Methods in
Molecular Biology 2932

Springer Protocols

Alexander Krasnitz
Pascal Belleau *Editors*

Cancer Bioinformatics

Second Edition

Humana Press

METHODS IN MOLECULAR BIOLOGY

Series Editor
John M. Walker
School of Life and Medical Sciences,
University of Hertfordshire,
Hatfield, UK

For further volumes:
http://www.springer.com/series/7651

For over 35 years, biological scientists have come to rely on the research protocols and methodologies in the critically acclaimed *Methods in Molecular Biology* series. The series was the first to introduce the step-by-step protocols approach that has become the standard in all biomedical protocol publishing. Each protocol is provided in readily-reproducible step-by-step fashion, opening with an introductory overview, a list of the materials and reagents needed to complete the experiment, and followed by a detailed procedure that is supported with a helpful notes section offering tips and tricks of the trade as well as troubleshooting advice. These hallmark features were introduced by series editor Dr. John Walker and constitute the key ingredient in each and every volume of the *Methods in Molecular Biology* series. Tested and trusted, comprehensive and reliable, all protocols from the series are indexed in PubMed.

Cancer Bioinformatics

Second Edition

Edited by

Alexander Krasnitz and Pascal Belleau

Cold Spring Harbor Laboratory, Cold Spring Harbor, NY, USA

Humana Press

Editors
Alexander Krasnitz
Cold Spring Harbor Laboratory
Cold Spring Harbor, NY, USA

Pascal Belleau
Cold Spring Harbor Laboratory
Cold Spring Harbor, NY, USA

ISSN 1064-3745 ISSN 1940-6029 (electronic)
Methods in Molecular Biology
ISBN 978-1-0716-4565-9 ISBN 978-1-0716-4566-6 (eBook)
https://doi.org/10.1007/978-1-0716-4566-6

Preface

This second edition of the Cancer Bioinformatics volume in the Methods in Molecular Biology (MiMB) protocol series reflects the new status of data management and analysis as an integral part of basic and translational research in cancer. Cancer bioinformatics' mission is twofold: to bring cutting-edge computational tools to bear on problems in cancer research and to provide computational support of novel molecular diagnostics and treatment modalities in the clinic of cancer. In keeping with these goals, the present volume contains chapters on cancer-related software repositories, databases, and cloud computing resources; computing techniques applied to recently developed molecular protocols in cancer biology; in-depth analysis of genomic alterations caused by cancer; methods to evaluate findings from liquid biopsies; and prognostic tools for immunotherapies. Written in the MiMB protocol style, these chapters provide step-by-step guidance to each respective computational method or resource, facilitating their adoption by the readers.

Cold Spring Harbor, NY, USA — *Alexander Krasnitz*
Pascal Belleau

Contents

Preface . . . . . *v*
Contributors . . . . . *ix*

1 Bioconductor's Computational Ecosystem for Genomic Data Science in Cancer 1
Marcel Ramos, Lori Shepherd, Nathan C. Sheffield, Alexandru Mahmoud, Hervé Pagès, Andres Wokaty, Dario Righelli, Davide Risso, Sean Davis, Sehyun Oh, Levi Waldron, Martin Morgan, and Vincent Carey

2 Building Portable and Reproducible Cancer Informatics Workflows for Scalable Data Analysis: An RNA Sequencing Tutorial 47
Rowan F. Beck, Zelia F. Worman, Gaurav Kaushik, and Brandi N. Davis-Dusenbery

3 Using the Cancer Epitope Database and Analysis Resource (CEDAR) 75
Zeynep Koşaloğlu-Yalçın, Randi Vita, Nina Blazeska, Bjoern Peters, and Alessandro Sette

4 Quantifying the Prevalence of Non-B DNA Motifs as a Marker of Non-B Burden in Cancer Using NBBC 93
Qi Xu and Jeanne Kowalski

5 Starfish: Deciphering Complex Genomic Rearrangement Signatures Across Human Cancers 105
Lisui Bao

6 Using FFPEsig to Remove Formalin-Induced Artifacts and Characterize Mutational Signatures in Cancer 125
Qingli Guo, Ann-Marie Baker, Ville Mustonen, and Trevor A. Graham

7 Inferring Phenotypes of Copy Number Clones in Cancer Populations Using TreeAlign 137
Hongyu Shi, Matthew Zatzman, Sohrab Shah, and Andrew McPherson

8 Inference of Genetic Ancestry from Cancer-Derived Molecular Data with RAIDS 153
Pascal Belleau, Astrid Deschênes, David A. Tuveson, and Alexander Krasnitz

9 Pruning-Assisted Modeling of Network Graph Connectivity from Spatial Transcriptomic Data 177
Antara Biswas and Subhajyoti De

10 Inferring Metabolic Flux from Gene Expression Data Using METAFlux 187
Yuchen Pan, Yuefan Huang, Vakul Mohanty, and Ken Chen

11 Functional Pathway Inference Analysis (FPIA) 203
Irbaz I. Badshah and Pedro R. Cutillas

12 NGP: A Tool to Detect Noncoding RNA-Gene Regulatory Pairs from Expression Data 231
Hongjie Ke and Tianzhou Ma

13 MODIG: An Attention Mechanism-Based Approach to Cancer Driver Gene Identification 247
Wenyi Zhao and Zhan Zhou

14 Predictive Modeling of Anticancer Drug Sensitivity Using REFINED CNN 259
Daniel Nolte, Omid Bazgir, and Ranadip Pal

15 Anticancer Monotherapy and Polytherapy Drug Response Prediction Using Deep Learning: Guidelines and Best Practices 273
Amin Emad and David Earl Hostallero

16 Identification of Somatic Variants in Cancer Genomes from Tissue and Liquid Biopsy Samples 291
Kiran Krishnamachari, Hanaé Carrié, and Anders Jacobsen Skanderup

17 SUMMER: A Practical Tool for Identifying Factors and Biomarkers Associated with Pan-cancer Survival 303
Junyi Xin, Silu Chen, Huiqin Li, Meilin Wang, and Mulong Du

18 Predicting Tumor Antigens Using the LENS Workflow Through RAFT 319
Steven P. Vensko, Dante Bortone, and Benjamin G. Vincent

Index *343*

Contributors

Irbaz I. Badshah • *Centre for Genomics and Computational Biology, Barts Cancer Institute, Queen Mary University of London, John Vane Science Centre, Charterhouse Square, London, UK*

Ann-Marie Baker • *Genomics and Evolutionary Dynamics Laboratory, Centre for Evolution and Cancer, Institute of Cancer Research, London, UK*

Lisui Bao • *Key Laboratory of Evolution & Marine Biodiversity (Ministry of Education) and Institute of Evolution & Marine Biodiversity, Ocean University of China, Qingdao, China*

Omid Bazgir • *Modeling & Simulation/Clinical Pharmacology, Genentech, South San Francisco, CA, USA*

Rowan F. Beck • *Velsera, Charlestown, MA, USA*

Pascal Belleau • *Simons Center for Quantitative Biology, Cold Spring Harbor Laboratory, Cold Spring, NY, USA; Cancer Center, Cold Spring Harbor Laboratory, Cold Spring, NY, USA*

Antara Biswas • *Rutgers Cancer Institute of New Jersey, Rutgers, the State University of New Jersey, New Brunswick, NJ, USA*

Nina Blazeska • *Center for Vaccine Innovation, La Jolla Institute for Immunology, La Jolla, CA, USA*

Dante Bortone • *Lineberger Comprehensive Cancer Center, University of North Carolina at Chapel Hill, Chapel Hill, NC, USA*

Vincent Carey • *Channing Division of Network Medicine, Mass General Brigham, Harvard Medical School, Buffalo, NY, USA*

Hanaé Carrié • *Genome Institute of Singapore (GIS), Agency for Science, Technology and Research (A*STAR), Singapore, Republic of Singapore*

Ken Chen • *Department of Bioinformatics and Computational Biology, The University of Texas MD Anderson Cancer Center, Houston, TX, USA*

Silu Chen • *Department of Environmental Genomics, Jiangsu Key Laboratory of Cancer Biomarkers, Prevention and Treatment, Collaborative Innovation Center for Cancer Personalized Medicine, Nanjing Medical University, Nanjing, China; Department of Genetic Toxicology, The Key Laboratory of Modern Toxicology of Ministry of Education, Center for Global Health, School of Public Health, Nanjing Medical University, Nanjing, China*

Pedro R. Cutillas • *Centre for Genomics and Computational Biology, Barts Cancer Institute, Queen Mary University of London, John Vane Science Centre, Charterhouse Square, London, UK*

Sean Davis • *University of Colorado Anschutz School of Medicine, Aurora, CO, USA*

Brandi N. Davis-Dusenbery • *Independent Advisor, Charlestown, MA, USA*

Subhajyoti De • *Rutgers Cancer Institute of New Jersey, Rutgers, the State University of New Jersey, New Brunswick, NJ, USA*

Astrid Deschênes • *Cancer Center, Cold Spring Harbor Laboratory, Cold Spring, NY, USA*

Mulong Du • *Department of Environmental Genomics, Jiangsu Key Laboratory of Cancer Biomarkers, Prevention and Treatment, Collaborative Innovation Center for Cancer*

Personalized Medicine, Nanjing Medical University, Nanjing, China; Department of Genetic Toxicology, The Key Laboratory of Modern Toxicology of Ministry of Education, Center for Global Health, School of Public Health, Nanjing Medical University, Nanjing, China; Department of Biostatistics, Center for Global Health, School of Public Health, Nanjing Medical University, Nanjing, China

Amin Emad • *Department of Electrical and Computer Engineering, McGill University, Montreal, QC, Canada; Mila, Quebec AI Institute, Montreal, QC, Canada; The Rosalind and Morris Goodman Cancer Institute, Montreal, QC, Canada*

Trevor A. Graham • *Genomics and Evolutionary Dynamics Laboratory, Centre for Evolution and Cancer, Institute of Cancer Research, London, UK*

Qingli Guo • *Genomics and Evolutionary Dynamics Laboratory, Centre for Evolution and Cancer, Institute of Cancer Research, London, UK; Organismal and Evolutionary Biology Research Programme, Department of Computer Science, University of Helsinki, Helsinki, Finland*

David Earl Hostallero • *Department of Electrical and Computer Engineering, McGill University, Montreal, QC, Canada; Mila, Quebec AI Institute, Montreal, QC, Canada*

Yuefan Huang • *Department of Bioinformatics and Computational Biology, The University of Texas MD Anderson Cancer Center, Houston, TX, USA; Department of Biostatistics & Data Science, School of Public Health, The University of Texas Health Science Center at Houston (UTHealth), Houston, TX, USA*

Gaurav Kaushik • *ScienceIO, New York, NY, USA*

Hongjie Ke • *Department of Epidemiology and Biostatistics, School of Public Health, University of Maryland, College Park, MD, USA*

Zeynep Koşaloğlu-Yalçın • *Center for Vaccine Innovation, La Jolla Institute for Immunology, La Jolla, CA, USA*

Jeanne Kowalski • *Department of Oncology, Dell Medical School, University of Texas at Austin, Austin, TX, USA*

Alexander Krasnitz • *Simons Center for Quantitative Biology, Cold Spring Harbor Laboratory, Cold Spring, NY, USA; Cancer Center, Cold Spring Harbor Laboratory, Cold Spring, NY, USA*

Kiran Krishnamachari • *Genome Institute of Singapore (GIS), Agency for Science, Technology and Research (A*STAR), Singapore, Republic of Singapore*

Huiqin Li • *Department of Environmental Genomics, Jiangsu Key Laboratory of Cancer Biomarkers, Prevention and Treatment, Collaborative Innovation Center for Cancer Personalized Medicine, Nanjing Medical University, Nanjing, China; Department of Genetic Toxicology, The Key Laboratory of Modern Toxicology of the Ministry of Education, Center for Global Health, School of Public Health, Nanjing Medical University, Nanjing, China*

Tianzhou Ma • *Department of Epidemiology and Biostatistics, School of Public Health, University of Maryland, College Park, MD, USA*

Alexandru Mahmoud • *Channing Division of Network Medicine, Mass General Brigham, Harvard Medical School, Boston, MA, USA*

Andrew McPherson • *Computational Oncology, Department of Epidemiology and Biostatistics, Memorial Sloan Kettering Cancer Center, New York, NY, USA*

Vakul Mohanty • *Department of Bioinformatics and Computational Biology, The University of Texas MD Anderson Cancer Center, Houston, TX, USA*

Martin Morgan • *Roswell Park Comprehensive Cancer Center, Buffalo, NY, USA*

Ville Mustonen • *Organismal and Evolutionary Biology Research Programme, Department of Computer Science, University of Helsinki, Helsinki, Finland*
Daniel Nolte • *Department of Electrical and Computer Engineering, Texas Tech University, Lubbock, TX, USA*
Sehyun Oh • *City University of New York School of Public Health, New York, NY, USA*
Hervé Pagès • *Fred Hutchinson Cancer Center, Seattle, WA, USA*
Ranadip Pal • *Department of Electrical and Computer Engineering, Texas Tech University, Lubbock, TX, USA*
Yuchen Pan • *Department of Bioinformatics and Computational Biology, The University of Texas MD Anderson Cancer Center, Houston, TX, USA; Department of Biostatistics & Data Science, School of Public Health, The University of Texas Health Science Center at Houston (UTHealth), Houston, TX, USA*
Bjoern Peters • *Center for Vaccine Innovation, La Jolla Institute for Immunology, La Jolla, CA, USA; Department of Medicine, University of California San Diego, La Jolla, CA, USA*
Marcel Ramos • *City University of New York School of Public Health, New York, NY, USA*
Dario Righelli • *Department of Electrical Engineering and Information Technology, University of Naples "Federico II", Naples, Italy*
Davide Risso • *Department of Statistical Sciences, University of Padova, Padova, Italy*
Alessandro Sette • *Center for Vaccine Innovation, La Jolla Institute for Immunology, La Jolla, CA, USA; Department of Medicine, University of California San Diego, La Jolla, CA, USA*
Sohrab Shah • *Computational Oncology, Department of Epidemiology and Biostatistics, Memorial Sloan Kettering Cancer Center, New York, NY, USA*
Nathan C. Sheffield • *University of Virginia, Charlottesville, VA, USA*
Lori Shepherd • *Roswell Park Comprehensive Cancer Center, Buffalo, NY, USA*
Hongyu Shi • *Computational Oncology, Department of Epidemiology and Biostatistics, Memorial Sloan Kettering Cancer Center, New York, NY, USA*
Anders Jacobsen Skanderup • *Genome Institute of Singapore (GIS), Agency for Science, Technology and Research (A*STAR), Singapore, Republic of Singapore*
David A. Tuveson • *Cancer Center, Cold Spring Harbor Laboratory, Cold Spring, NY, USA*
Steven P. Vensko II • *Lineberger Comprehensive Cancer Center, University of North Carolina at Chapel Hill, Chapel Hill, NC, USA*
Benjamin G. Vincent • *Lineberger Comprehensive Cancer Center, University of North Carolina at Chapel Hill, Chapel Hill, NC, USA*
Randi Vita • *Center for Vaccine Innovation, La Jolla Institute for Immunology, La Jolla, CA, USA*
Levi Waldron • *City University of New York School of Public Health, New York, NY, USA*
Meilin Wang • *Department of Environmental Genomics, Jiangsu Key Laboratory of Cancer Biomarkers, Prevention and Treatment, Collaborative Innovation Center for Cancer Personalized Medicine, Nanjing Medical University, Nanjing, China; Department of Genetic Toxicology, The Key Laboratory of Modern Toxicology of Ministry of Education, Center for Global Health, School of Public Health, Nanjing Medical University, Nanjing, China*
Andres Wokaty • *City University of New York School of Public Health, New York, NY, USA*
Zelia F. Worman • *Velsera, Charlestown, MA, USA*

Junyi Xin • *Department of Bioinformatics, School of Biomedical Engineering and Informatics, Nanjing Medical University, Nanjing, Jiangsu, China; Department of Environmental Genomics, Jiangsu Key Laboratory of Cancer Biomarkers, Prevention and Treatment, Collaborative Innovation Center for Cancer Personalized Medicine, Nanjing Medical University, Nanjing, China*

Qi Xu • *Department of Oncology, Dell Medical School, University of Texas at Austin, Austin, TX, USA; Department of Molecular Biosciences, College of Natural Sciences, University of Texas at Austin, Austin, TX, USA*

Matthew Zatzman • *Computational Oncology, Department of Epidemiology and Biostatistics, Memorial Sloan Kettering Cancer Center, New York, NY, USA*

Wenyi Zhao • *State Key Laboratory of Advanced Drug Delivery and Release Systems & Innovation Institute for Artificial Intelligence in Medicine, College of Pharmaceutical Sciences, Zhejiang University, Hangzhou, China*

Zhan Zhou • *State Key Laboratory of Advanced Drug Delivery and Release Systems & Innovation Institute for Artificial Intelligence in Medicine, College of Pharmaceutical Sciences, Zhejiang University, Hangzhou, China; The Fourth Affiliated Hospital, Zhejiang University School of Medicine, Yiwu, China*

Chapter 1

Bioconductor's Computational Ecosystem for Genomic Data Science in Cancer

Marcel Ramos, Lori Shepherd, Nathan C. Sheffield, Alexandru Mahmoud, Hervé Pagès, Andres Wokaty, Dario Righelli, Davide Risso, Sean Davis, Sehyun Oh, Levi Waldron, Martin Morgan, and Vincent Carey

Abstract

The Bioconductor project enters its third decade with over two thousand packages for genomic data science, over 100,000 annotation and experiment resources, and a global system for convenient distribution to researchers. Over 60,000 PubMed Central citations and terabytes of content shipped per month attest to the impact of the project on cancer genomic data science. This report provides an overview of cancer genomics resources in Bioconductor. After an overview of Bioconductor project principles, we address exploration of institutionally curated cancer genomics data such as TCGA. We then review genomic annotation and ontology resources relevant to cancer and then briefly survey analytical workflows addressing specific topics in cancer genomics. Concluding sections cover how new software and data resources are brought into the ecosystem and how the project is tackling needs for training of the research workforce. Bioconductor's strategies for supporting methods developers and researchers in cancer genomics are evolving along with experimental and computational technologies. All the tools described in this report are backed by regularly maintained learning resources that can be used locally or in cloud computing environments.

Key words Cancer genomics, open source software, data structures, transcriptomics, mutations, ontology, epigenomics, spatial transcriptomics

1 Introduction

Computation is a central component of cancer genomics research. Tumor sequencing is the basis of computational investigation of mutational, epigenetic and immunologic processes associated with cancer initiation and progression. Numerous computational workflows have been produced to profile tumor cell transcriptomes and proteomes. New technologies promise to unite sequence-based characterizations with digital histopathology, ultimately driving

Alexander Krasnitz and Pascal Belleau (eds.), *Cancer Bioinformatics*, Methods in Molecular Biology, vol. 2932,
https://doi.org/10.1007/978-1-0716-4566-6_1,

efforts in molecule design and evaluation to produce patient-centered treatments.

Bioconductor is an open source software project with a 20 year history of uniting biostatisticians, bioinformaticians, and genome researchers in the creation of an ecosystem of data, annotation, and analysis resources for research in genome-scale biology. This paper will review current approaches of the project to advancing cancer genomics. After a brief discussion of basic principles of the Bioconductor project, we will present a "top down" survey of resources useful for cancer bioinformatics. Primary sections address

- how to explore institutionally curated cancer genomics data
- genomic annotation resources relevant to cancer genomics
- analytical workflows
- components for introducing new data or analyses
- pedagogics and workforce development.

Two concluding sections offer discussion of possible paths forward, and a detailed description of software components underlying an exemplary integrative analysis of response to immune checkpoint blockade.

2 Bioconductor principles

2.1 R packages and vignettes

Software tools and data resources in Bioconductor are organized into "R packages". These are collections of folders with data, code (principally R functions), and documentation, following a protocol specified in the Writing R Extensions manual [1]. R packages have a DESCRIPTION file with metadata about package contents and provenance. Package structure can be checked for validity using the `R CMD check` facility. Documentation of code and data can be programmatically checked for existence and validity. The DESCRIPTION file for a package specifies its version and also gives precise definition of how an R package may depend upon versions of other packages.

At its inception, Bioconductor introduced a new approach to holistic package documentation called "vignette". Vignettes provide narrative and explanation interleaved with executable code describing package operations. While R function manual pages describe the operation of individual functions, vignettes illustrate the interoperation of package components and provide motivation for both package design but also context for its use.

2.2 R package repositories; repository evolution

Bioconductor software forms a coherent ecosystem that can be checked for consistency of versions of all packages available in a given installation of R. Bioconductor packages may specify

dependency on other Bioconductor packages, or packages that are available in the CRAN repository. Bioconductor does not include packages with dependencies on "github-only" packages. Later in this paper we will provide details on package quality assurance that provide a rationale for this restriction.

Major updates to the R language occur annually, and updates are preceded by careful assessment of effects of language change on Bioconductor package operations. These effects can be identified through changes in the output of R CMD check. The Bioconductor ecosystem is updated twice a year, once to coincide with update to R, and once about six months later. The semiannual updates reflect the need to track developments in the fast-moving field of genomic data science.

2.3 Package quality assessment; installation consistency

The BiocCheck function is used to provide more stringent assessment of package compliance with basic principles of the Bioconductor ecosystem.

The BiocManager package provides functionality for installing and updating packages and for verifying the coherence and version status of the currently installed package collection. This is important in the context of a language and package ecosystem that changes every six months, while analyses may take years to complete. Tools for recreating past package collections are available to assist in reproducing outputs of prior analyses.

2.4 Unifying assay and sample data: SummarizedExperiment and MultiAssayExperiment

Most of the data from genome-scale experiments to be discussed in this chapter are organized in special data containers rooted in the concepts of the SummarizedExperiment class. Briefly, assay data are thought of as occupying a $G \times N$ array, and sample level data occupy an $N \times K$ table. The array and the table are linked together in the SummarizedExperiment; see Figure 1.

Multiple representations of assay results may be managed in this structure, but all assay arrays must have dimensions $G \times N$.

For experiment collections in which the same samples are subjected to multiple genome-scale assays, MultiAssayExperiment containers are used. See Figure 2 for the layout.

Further details on these data structures will be provided in section 6.

2.5 Downloading and caching cancer genomics data and annotations

Downloading and managing data from various online resources can be excessively time consuming. Bioconductor encourages data caching for increased efficiency and reproducibility. The caching data methods employed in Bioconductor allow analysis code to concisely refer to data resources as needed, with minimal attention to how data are stored, retrieved or transformed. It allows for easy management and reuse of data that are on remote servers or in cloud, storing source location and providing information for data

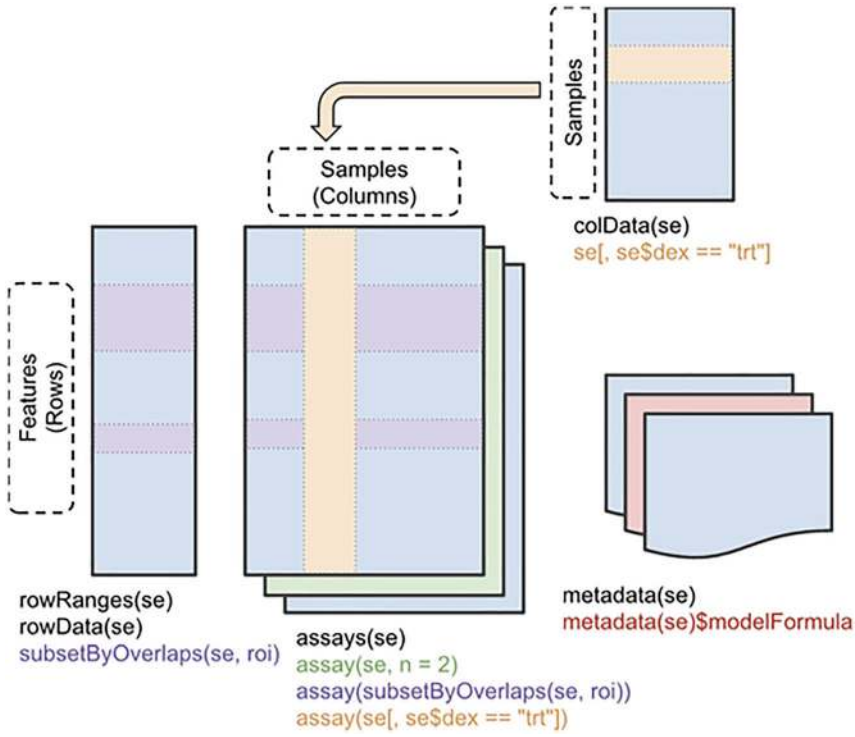


Fig. 1 SummarizedExperiment schematic.

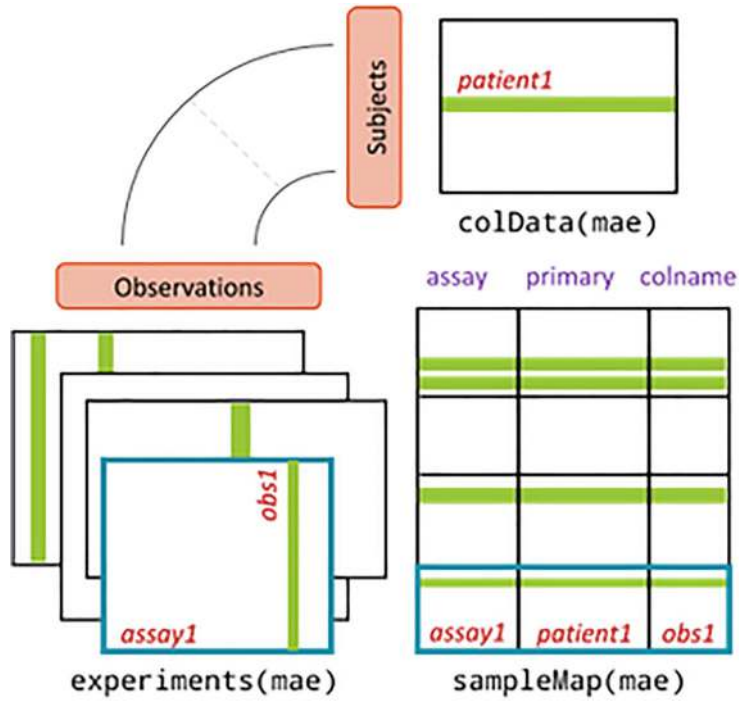


Fig. 2 MultiAssayExperiment schematic.

updates. The BiocFileCache Bioconductor package handles data management from within R.

BiocFileCache is a general-use caching system but Bioconductor also provides "Hubs", AnnotationHub and ExperimentHub, to help distributed annotation or experimental data hosted externally. Both AnnotationHub and ExperimentHub use BiocFileCache to handle download and caching of data.

AnnotationHub provides a centralized repository of diverse genomic annotations, facilitating easy access and integration into analyses. Researchers can seamlessly retrieve information such as genomic features, functional annotations, and variant data, streamlining the annotation process for their analyses.

ExperimentHub extends this concept to experimental data. It serves as a centralized hub for storing and sharing curated experiment-level datasets, allowing researchers to access a wide range of experimental designs and conditions. This cloud-based infrastructure enhances collaboration and promotes the reproducibility of analyses across different laboratories.

The curatedTCGAData package provides some resources through ExperimentHub, as do many other self-identified "CancerData" resources. Once the ExperimentHub is loaded, it can be queried for terms of interest.

Here and throughout, shading is used to indicate code operations in Bioconductor 3.19 with R 4.4. Lines of output are preceded by ##.

```
library(ExperimentHub)
eh = ExperimentHub()
query(eh, "CancerData")
## ExperimentHub with 1742 records
## # snapshotDate(): 2024-04-29
## # $dataprovider: Eli and Edythe L. Broad Institute
##             of Harvard and MIT, GEO, ...
## # $species: Homo sapiens, Mus musculus, NA
## # $rdataclass: SummarizedExperiment, RaggedExperiment,
##             matrix, list, DFrame,...
## # additional mcols(): taxonomyid, genome, description,
## #   coordinate_1_based, maintainer, rdatadateadded,
##         preparerclass, tags,
## #       rdatapath, sourceurl, sourcetype
## # retrieve records with, e.g., 'object[["EH558"]]'
##
##           title
##   EH558  | ACC_CNASNP-20160128
##   EH559  | ACC_CNVSNP-20160128
##   EH560  | ACC_colData-20160128
##   EH561  | ACC_GISTIC_AllByGene-20160128
```

```
##   EH562  | ACC_GISTIC_ThresholdedByGene-20160128
##   ...     ...
##   EH8533 | tcga_transcript_counts
##   EH8534 | target_rhabdoid_wgbs_hg19
##   EH8567 | xenium_hs_breast_addon
##   EH9482 | Capper_example_betas.rda
##   EH9483 | GIMiCC_Library.rda
```

Multiple terms can be used to narrow results before choosing a download.

```
query(eh, c("CancerData", "esophageal"))
## ExperimentHub with 2 records
## snapshotDate(): 2023-10-24
## $dataprovider: University of California San Francisco
## $species: Homo sapiens
## $rdataclass: RangedSummarizedExperiment, data.frame
## additional mcols(): taxonomyid, genome, description,
##   coordinate_1_based, maintainer, rdatadateadded,
##   preparerclass, tags,}
##   rdatapath, sourceurl, sourcetype }
## retrieve records with, e.g., object[["EH8527"]]
##          title
##   EH8527 | cao_esophageal_wgbs_hg19
##   EH8530 | cao_esophageal_transcript_counts
```

Similarly AnnotationHub files can be downloaded for annotating data. For example, the ensembl 110 release of gene and protein annotations are obtained with the following:

```
library(AnnotationHub)
ah = AnnotationHub()
query(ah, c("ensembl", "110", "Homo sapiens"))
#snapshotDate(): 2024-04-29
#AnnotationHub with 1 record
## snapshotDate(): 2024-04-29
## names(): AH113665
## $dataprovider: Ensembl
## $species: Homo sapiens
## $rdataclass: EnsDb
## $rdatadateadded: 2023-04-25
## $title: Ensembl 110 EnsDb for Homo sapiens
## $description: Gene and protein annotations
for
```

```
  Homo
##      sapiens based on Ensem...
## $taxonomyid: 9606
## $genome: GRCh38
## $sourcetype: ensembl
## $sourceurl: http://www.ensembl.org
## $sourcesize: NA
## $tags: c("110", "Annotation",
  "AnnotationHubSoftware",
##       "Coverage", "DataImport", "EnsDb",
        "Ensembl",
##       "Gene", "Protein",
##       "Sequencing", "Transcript")
## retrieve record with 'object[["AH113665"]]'
```

3 Exploring institutionally curated cancer genomics data

3.1 The Cancer Genome Atlas

An overview of Bioconductor's resource for the Cancer Genome Atlas (TCGA) is easy to obtain, with the curatedTCGAData package.

```
library(curatedTCGAData)
tcgatab = curatedTCGAData(version="2.1.1")
```

Records obtained for adrenocortical carcinoma (code ACC) are in Table 1.

Various conventions are in play in this table. The "title" field is of primary concern. The title string can be decomposed into substrings with interpretation `[tumorcode]_[assay]-[date]_[optional codes]`. The column `ah_id` will be explained in section 4, and entries in column `rdataclass` will be discussed in section 6 below.

3.1.1 Tumor code resolution

There are 33 different tumor types available in TCGA. The decoding of tumor codes for the first ten in alphabetical order is provided in Table 2.

3.1.2 Assay codes and counts

Assays performed on tumors vary across tumor types. For assay types shared between breast cancer, glioblastoma, and lung adenocarcinoma (code LUAD), the numbers of tumor and normal samples available in curatedTCGAData are provided in Table 3.

Table 1
Records returned by curatedTCGAData::curatedTCGAData(), filtered to those pertaining to adrenocortical carcinoma.

	ah_id	title	file_size	rdataclass
1	EH4737	ACC_CNASNP-20160128	0.8 Mb	RaggedExperiment
2	EH4738	ACC_CNVSNP-20160128	0.2 Mb	RaggedExperiment
3	EH4740	ACC_GISTIC_AllByGene-20160128	0.2 Mb	SummarizedExperiment
4	EH4741	ACC_GISTIC_Peaks-20160128	0 Mb	RangedSummarizedExperiment
5	EH4742	ACC_GISTIC_ThresholdedByGene-20160128	0.2 Mb	SummarizedExperiment
6	EH4744	ACC_Methylation-20160128_assays	239.2 Mb	SummarizedExperiment
7	EH4745	ACC_Methylation-20160128_se	6 Mb	RaggedExperiment
8	EH4747	ACC_Mutation-20160128	0.7 Mb	SummarizedExperiment
9	EH4748	ACC_RNASeq2Gene-20160128	2.7 Mb	SummarizedExperiment
10	EH4750	ACC_RPPAArray-20160128	0.1 Mb	SummarizedExperiment
414	EH8118	ACC_miRNASeqGene-20160128	0.2 Mb	SummarizedExperiment
415	EH8119	ACC_RNASeq2GeneNorm-20160128	5.4 Mb	SummarizedExperiment

Table 2
Decoding TCGA tumor code abbreviations.

Code	Tumor.Type
ACC	Adrenocortical Carcinoma
BLCA	Bladder Urothelial Carcinoma
BRCA	Breast Invasive Carcinoma
CESC	Cervical Squamous Cell Carcinoma and Endocervical Adenocarcinoma
CHOL	Cholangiocarcinoma
CNTL	Controls
COAD	Colon Adenocarcinoma
DLBC	Lymphoid Neoplasm Diffuse Large B-cell Lymphoma
ESCA	Esophageal Carcinoma
FPPP	FFPE Pilot Phase II
GBM	Glioblastoma Multiforme
HNSC	Head and Neck Squamous Cell Carcinoma
KICH	Kidney Chromophobe
KIRC	Kidney Renal Clear Cell Carcinoma

(continued)

Table 2
(continued)

Code	Tumor.Type
KIRP	Kidney Renal Papillary Cell Carcinoma
LAML	Acute Myeloid Leukemia
LCML	Chronic Myelogenous Leukemia
LGG	Brain Lower Grade Glioma
LIHC	Liver Hepatocellular Carcinoma
LUAD	Lung Adenocarcinoma
LUSC	Lung Squamous Cell Carcinoma
MESO	Mesothelioma
MISC	Miscellaneous
OV	Ovarian Serous Cystadenocarcinoma
PAAD	Pancreatic Adenocarcinoma
PCPG	Pheochromocytoma and Paraganglioma
PRAD	Prostate Adenocarcinoma
READ	Rectum Adenocarcinoma
SARC	Sarcoma
SKCM	Skin Cutaneous Melanoma
STAD	Stomach Adenocarcinoma
TGCT	Testicular Germ Cell Tumors
THCA	Thyroid Carcinoma
THYM	Thymoma
UCEC	Uterine Corpus Endometrial Carcinoma
UCS	Uterine Carcinosarcoma
UVM	Uveal Melanoma

Table 3
Numbers of assays available in TCGA on tumor and normal samples, for breast cancer, glioblastoma, and lung adenocarcinoma.

	BRCA	BRCAnormal	GBM	GBMnormal	LUAD	LUADnormal
CNASNP	1089	1120	577	527	516	579
CNVSNP	1080	1119	577	527	516	579
GISTIC_AllByGene	1080	0	577	0	516	0

(continued)

Table 3
(continued)

	BRCA	BRCAnormal	GBM	GBMnormal	LUAD	LUADnormal
GISTIC_Peaks	1080	0	577	0	516	0
GISTIC_ThresholdedByGene	1080	0	577	0	516	0
Mutation	988	5	283	7	230	0
RNASeq2Gene	1093	119	153	13	515	61
RPPAArray	887	50	233	11	365	0
RNASeq2GeneNorm	1093	119	153	13	515	61
Methylation_methyl27	314	29	285	0	65	24
Methylation_methyl450	783	102	140	14	458	34

3.1.3 An example dataset for RNA-seq from glioblastoma multiforme

We obtain normalized RNA-seq data on primary tumor samples for GBM with

```
gbrna = TCGAprimaryTumors(curatedTCGAData("GBM",
    "RNASeq2GeneNorm", dry.run=FALSE, version="2.1.1"))
gbrna
## A MultiAssayExperiment object of 1 listed
##experiment with a user-defined name and respective class.
##Containing an ExperimentList class object of length 1:
## [1] GBM_RNASeq2GeneNorm-20160128: SummarizedExperiment
##       with 18199 rows and 153 columns
##
## Functionality:
##  experiments() - obtain the ExperimentList instance
##  colData() - the primary/phenotype DataFrame
##  sampleMap() - the sample coordination DataFrame
##  $, [, [[ - extract colData columns, subset or
##      experiment
## *Format() - convert into a long or wide DataFrame
## assays() - convert ExperimentList to a SimpleList of
##      matrices
## exportClass() - save data to flat files
```

R functions defined in Bioconductor packages can operate on the variable `gbrna` to retrieve information of interest. Details on the underlying data structure are given in section 6 below. For most assay types, we think of the quantitative assay information as tabular

in nature, with table rows corresponding to genomic features such as genes, and table columns corresponding to samples.

Information on GBM samples employs the `colData` function.

```
dim(colData(gbrna))
## [1] 153 4380
```

For sample level information obtained with `colData`, we think of rows as samples, and columns as sample attributes.

3.1.4 Clinical and phenotypic data

TCGA datasets are generally provided as combinations of results for tumor tissue and normal tissue. The determination of a record's sample type is encoded in the sample "barcode". Decoding of sample barcodes is described at

`https://docs.gdc.cancer.gov/Encyclopedia/pages/TCGA_Barcode/`

with specific interpretation of sample types listed at

`https://gdc.cancer.gov/resources-tcga-users/tcga-code-tables/sample-type-codes`

separately. The TCGAutils package provides utilities for extracting data on primary tumor samples, excluding samples that may have been taken on normal tissue or metastases.

Clinical and phenotypic data on all TCGA samples are voluminous. For example, there are 2684 fields of sample level data for BRCA samples, and 4380 fields for GBM samples. Many of these fields are meaningfully populated for only a very small minority of samples. To see this for GBM:

```
mean(sapply(colData(gbrna), function(x) mean(is.na (x))>.90))
## [1] 0.8091324
```

In words, for 81% of clinical data fields in TCGA GBM data, at least 90% of entries are missing.

Nevertheless, with careful inspection of fields and contents, nearly complete clinical data can be extracted and combined with molecular and genetic assay data with modest effort.

The following code chunk illustrates a very crude approach to comparing survival profiles for BRCA, GBM, and LUAD donors. The result is in Figure 3.

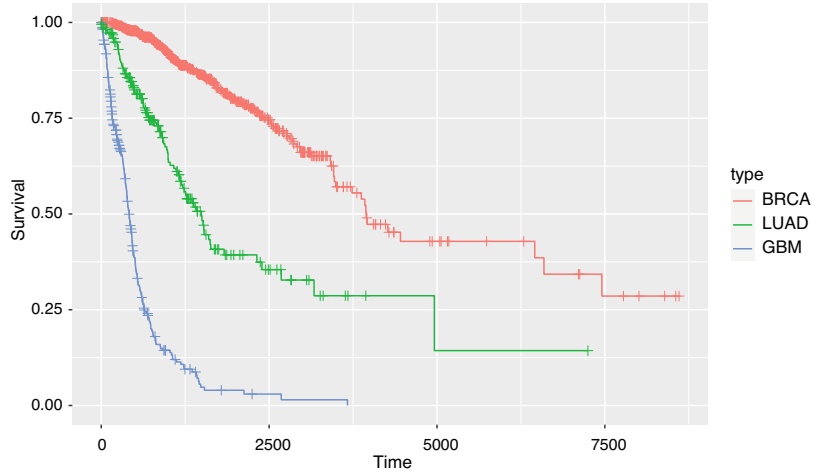


Fig. 3 Survival profile extraction from three MultiAssayExperiments produced with curatedTCGAData calls

```
# obtain mutation data for BRCA, GBM, LUAD; could use any or
#    all assay types
brmut = curatedTCGAData("BRCA", "Mutation", version = "2.1.1",
     dry.run = FALSE)
gbmut = curatedTCGAData("GBM", "Mutation", version = "2.1.1",
     dry.run = FALSE)
lumut = curatedTCGAData("LUAD", "Mutation", version = "2.1.1",
     dry.run = FALSE)
# extract survival times
library(survival)
getSurv = function(mae) {
days_on=with(colData(mae),ifelse(is.na(days_to_last_followup),
days_to_death,days_to_last_followup))
Surv(days_on,colData(mae)$vital_status)
}
ss=lapply(list(brmut,gbmut,lumut),getSurv)
codes=c("BRCA","GBM","LUAD")
type=factor(rep(codes,sapply(ss,length)))
allsurv=do.call(c,ss)
library(GGally)
ggsurv(survfit(allsurv~type))
```

At this point, survival times within tumor type can be stratified by any features of the mutation profiles of individual samples. The "RaggedExperiment" class is employed to test each BRCA sample for presence of any mutation in the gene TTN. See Figure 4.

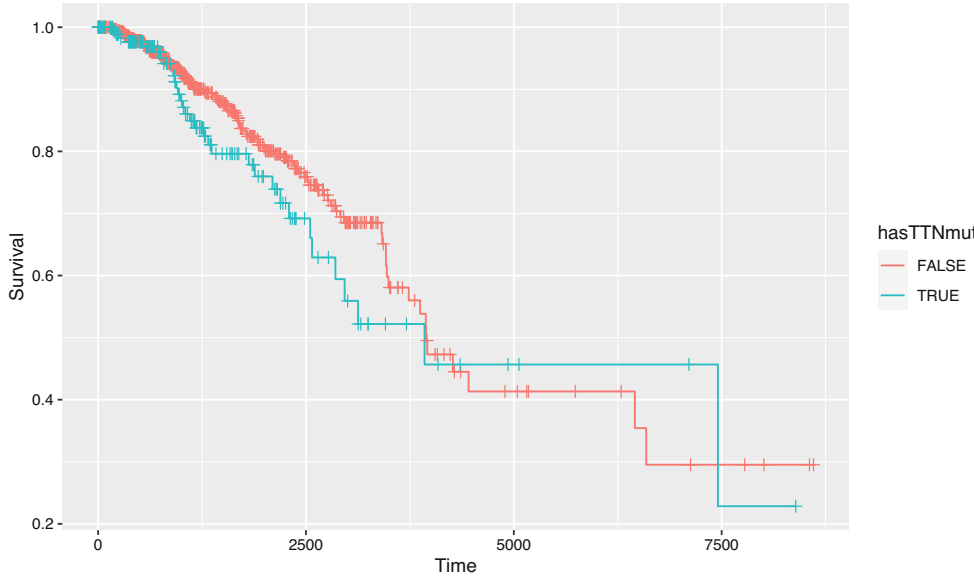


Fig. 4 Survival distributions for donors of breast tumors in TCGA, stratified by presence or absence of mutation in gene TTN.

```
bprim = TCGAprimaryTumors(brmut)
## harmonizing input:
## removing 5 sampleMap rows with 'colname' not
in
##     colnames of experiments
mutsyms    =    assay(experiments(bprim)[[1]],
"Hugo_Symbol")
cn = rownames(colData(bprim)) # short
cna = colnames(mutsyms) # long
cnas = substr(cna, 1, 12)
hasTTNmut = apply(assay(experiments(
   TCGAprimaryTumors(brmut))[[1]],
    "Hugo_Symbol"), 2,
    function(x) length(which(x=="TTN"))>0)
## harmonizing input:
## removing 5 sampleMap rows with 'colname' not in
##     colnames of experiments
names(hasTTNmut) = cnas
bsurv = getSurv(TCGAprimaryTumors(brmut))
## harmonizing input:
## removing 5 sampleMap rows with 'colname' not in
##     colnames of experiments
hasTTNmut = hasTTNmut[cn] # match mut records
to surv times
ggsurv(survfit(bsurv~hasTTNmut))
```

Similar manipulations permit exploration of relationships between any molecular assay outcomes and any clinical data collected in TCGA.

3.2 cBioPortal

The cBioPortal user guide at

`https://www.cbioportal.org/`

defines the goal of the portal to be reducing "the barriers between complex genomic data and cancer researchers by providing rapid, intuitive, and high-quality access to molecular profiles and clinical attributes from large-scale cancer genomics projects, and therefore to empower researchers to translate these rich data sets into biologic insights and clinical applications."

Bioconductor's cBioPortalData package simplifies access to over 300 genomic studies of diverse cancers in cBioPortal. The main unit of data access is the publication. The `cBioPortal` function mediates a connection between an R session and the cBioPortal API. `getStudies` returns a tibble with metadata on all studies.

```
library(cBioPortalData)
cbio = cBioPortal()
allst = getStudies(cbio)
dim(allst)
## [1] 397 13
```

A pruned selection of records from the cBioPortal studies table is given in Table 4.

To explore copy number alteration data from a study on angiosarcoma, we find the associated studyId field in `allst` and use the `cBioDataPack` function to retrieve a MultiAssayExperiment:

```
ann = "angs_project_painter_2018"
ang = cBioDataPack(ann)
ang
## A MultiAssayExperiment object of 3 listed
##experiments with user-defined names and respective classes.
####Containing an ExperimentList class object of length 3:
##[1] cna_hg19.seg: RaggedExperiment with 27835 rows
##         and 48 columns
##[2] cna: SummarizedExperiment with 23109 rows
##         and 48 columns
##[3] mutations: RaggedExperiment with 24058 rows
##         and 48 columns
```

Table 4
Excerpts from four fields on selected records in the cBioPortal getStudies output.

name	description	studyId
Adenoid Cystic Carcinoma of the Breast	Whole exome sequencing of 12 breast AdCCs.	acbc_mskcc_2015
Adenoid Cystic Carcinoma	Whole-exome or whole-genome sequencing analysis of 60 ACC tumor/normal pairs	acyc_mskcc_2013
Adenoid Cystic Carcinoma	Targeted Sequencing of 28 metastatic Adenoid Cystic Carcinoma samples.	acyc_fmi_2014
Adenoid Cystic Carcinoma	Whole-genome or whole-exome sequencing of 25 adenoid cystic carcinoma tumor/normal pairs.	acyc_jhu_2016
Adenoid Cystic Carcinoma	WGS of 21 salivary ACCs and targeted molecular analyses of a validation set (81 patients).	acyc_mda_2015
Adenoid Cystic Carcinoma	Whole-genome/exome sequencing of 10 ACC PDX models.	acyc_mgh_2016
Adenoid Cystic Carcinoma	Whole exome sequencing of 24 ACCs.	acyc_sanger_2013
Adenoid Cystic Carcinoma Project	Multi-Institute Cohort of 1045 Adenoid Cystic Carcinoma patients.	acc_2019
Basal Cell Carcinoma	Whole-exome sequencing of 126 basal cell carcinoma tumor/normal pairs; targeted sequencing of 163 sporadic samples (40 tumor/normal pairs) and 4 Gorlin symdrome basal cell carcinomas.	bcc_unige_2016

```
## Functionality:
##  experiments() - obtain the ExperimentList
    instance
##  colData() - the primary/phenotype DataFrame
##  sampleMap() - the sample coordination DataFrame
##  $, [, [[ - extract colData columns, subset
    , or
##      experiment
## *Format() - convert into a long or wide DataFrame
## assays() - convert ExperimentList to a SimpleList of
##      matrices
## exportClass() - save data to flat files
```

The copy number alteration outcomes are in the `assay` component of the experiment.

```
seg = experiments(ang)[[1]]
colnames(seg) = sapply(strsplit(colnames(seg),
"-"), "[", 5)
assay(seg)[1:4,1:4]
##
##                DAE1F DACME DADBW DAD34
## 1:12227-955755      71    NA    NA    NA
## 1:957844-1139868    62    NA    NA    NA
## 1:1140874-1471177  167    NA    NA    NA
## 1:1475170-1855370  113    NA    NA    NA
```

The rownames component of this matrix can be transformed to a GenomicRanges instance for concise manipulation.

```
allalt = GRanges(rownames(assay(seg)))
 allalt
## GRanges object with 27835 ranges and 0 meta-
data columns:
##         seqnames          ranges strand
##            <Rle>       <IRanges>  <Rle>
##      [1]       1     12227-955755      *
##      [2]       1    957844-1139868     *
##      [3]       1   1140874-1471177     *
##      [4]       1   1475170-1855370     *
##      [5]       1  1857786-17257894     *
##      ...     ...             ...    ...
##  [27831]      20    68410-1559342      *
##  [27832]      20   1585705-1592359     *
##  [27833]      20  1616247-62904955     *
##  [27834]      21  9907492-48084286     *
##  [27835]      22 16157938-51237572     *
##  -------
##  seqinfo: 22 sequences from an unspecified
    genome; no
##      seqlengths
```

We'll focus on chromosome 17, where TP53 is found. Regions of genomic alteration are summarized to their midpoints. The display in Figure 5 shows a strong peak in the vicinity of 7.5 Mb on chromosome 17, near TP53.

```
g17 = allalt[seqnames(allalt)=="17"]
df17 = as(g17, "data.frame")
df17$mid = .5*(df17$start+df17$end) # midpoint only
ggplot(df17,aes(x=mid))+geom_density(bw=.2)+xlab("chr 17 bp")
```

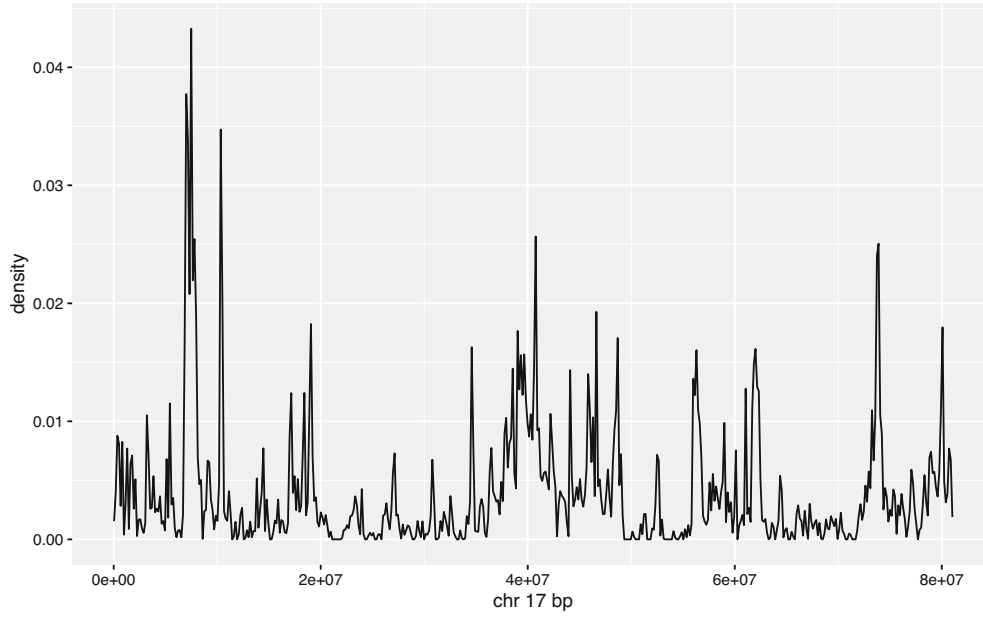


Fig. 5 Density of recurrent genomic alterations on chromosome 17 for 48 angiosarcoma patients

4 Genomic annotation resources relevant to cancer

4.1 Resources from UCSC, NCBI, and EMBL

Sequences for reference genome builds for human and other model organisms are supplied in BSgenome packages. BSgenome.Hsapiens.UCSC.hg19 provides all chromosomes and contigs for the 2009 build; the hg38 suffix may be used for the 2013 build. The recent "telomere to telomere" build is available as BSgenome.Hsapiens.NCBI.T2T.CHMv13v2.0.

NCBI's dbSNP catalog of genetic variants is provided in versioned packages. For example, SNPlocs.Hsapiens.dbSNP155.GRCh38 includes position and nucleotide content information for over 1 billion SNP identifiers ("rs numbers").

Tracks defined for the UCSC genome browser are also packaged. The package TxDb.Hsapiens.UCSC.knownGene.hg38 can be used to get gene, transcript, and exon location information for the hg38 build. The EnsDb packages provide similar information for annotations curated at EMBL.

```
library(EnsDb.Hsapiens.v86)
EnsDb.Hsapiens.v86
## EnsDb for Ensembl:
## |Backend: SQLite
## |Db type: EnsDb
```

```
## |Type of Gene ID: Ensembl Gene ID
## |Supporting package: ensembldb
## |Db created by: ensembldb package from Bioconductor
## |script\_version: 0.3.0
## |Creation time: Thu May 18 16:32:27 2017
## |ensembl\_version: 86
## |ensembl\_host: localhost
## |Organism: homo\_sapiens
## |taxonomy\_id: 9606
## |genome\_build: GRCh38
## |DBSCHEMAVERSION: 2.0
## | No. of genes: 63970.
## | No. of transcripts: 216741.
## |Protein data available.
```

The "genes" method provides addresses and additional annotations.

```
names(mcols(genes(EnsDb.Hsapiens.v86)))
## [1] "gene_id"            "gene_name"       "gene_biotype"
## [4] "seq_coord_system"   "symbol"          "entrezid"
head(table(genes(EnsDb.Hsapiens.v86)$gene_biotype))
##     3prime_overlapping_ncRNA   antisense
##                           30        5703
## bidirectional_promoter_lncRNA IG_C_gene
##                            4         23
##            IG_C_pseudogene    IG_D_gene
##                         11            64
```

More recent versions of Ensembl gene annotation are available from AnnotationHub, as illustrated above in section 2.5 with the creation of `ens110`.

4.2 Gene sets

Many methods have been developed to employ collections of genes for inference on hypotheses about cancer initiation or progression. The Molecular Signatures Database (MSigDB) is curated at Broad Institute, and can be harvested using the msigdb package.

Collect all gene sets for humans:

```
library(msigdb)
hssigs = getMsigdb(org="hs", id="SYM",
  version=getMsigdbVersions())
```

Find those with CANCER in their name:

```
nms = grep("CANCER", names(hssigs), value=TRUE)
head(nms)
## [1] "SOGA_COLORECTAL_CANCER_MYC_DN"
## [2] "SOGA_COLORECTAL_CANCER_MYC_UP"
## [3] "WATANABE_RECTAL_CANCER_RADIOTHERAPY_
  RESPONSIVE_UP"
## [4] "LIU_PROSTATE_CANCER_UP"
## [5] "BERTUCCI_MEDULLARY_VS_DUCTAL_BREAST_
  CANCER_UP"
## [6] "WATANABE_COLON_CANCER_MSI_VS_MSS_UP"
wangmet = hssigs[["WANG_METASTASIS_OF_BREAST_CANCER_ESR1_UP"]]
wangmet
## setName: WANG_METASTASIS_OF_BREAST_CANCER_ESR1_UP
## geneIds: KPNA2, HDGFL3, ..., PSMC2 (total: 22)
## geneIdType: Symbol
## collectionType: Broad
##bcCategory: c2 (Curated)
##bcSubCategory: CGP
## details: use 'details(object)'
```

Information on provenance is bound together with the gene list:

```
details(wangmet)
## setName: WANG_METASTASIS_OF_BREAST_CANCER_ESR1_UP
## geneIds: KPNA2, HDGFL3, ..., PSMC2 (total: 22)
## geneIdType: Symbol
## collectionType: Broad
##bcCategory: c2 (Curated)
##bcSubCategory: CGP
## setIdentifier: LVY1HGGWMJ7:35020:Fri May 26 13: 33:02
##        2023:1104005
## description: Genes whose expression in primary ER(+)
##     [GeneID=2099] breast cancer tumors positively correla
## (longDescription available)
## organism: Homo sapiens
## pubMedIds: 15721472
## urls: https://data.broadinstitute.org/gsea-msigdb/msigdb/
##   release/2023.1.Hs/msigdb_v2023.1.Hs.xml.zip
## contributor: Arthur Liberzon
```

4.3 Ontologies

Informal reasoning about cancer genomics employs conventional but frequently ambiguous terminology. In modern information science, ontologies are structured vocabularies (sets of "terms", which may be single words or natural language phrases) accompanied by explicit statements of semantic relationships among terms.

Bioconductor provides several approaches for using ontologies in cancer data science. The most familiar ontology in this domain is Gene Ontology (GO), which organizes vocabulary about genes and gene products in the areas of molecular function, cellular components, and biological processes.

4.3.1 Ontology usage with AnnotationDbi

A common use case is to find genes or proteins associated with some biological process, component, or function. A phrase like 'Golgi membrane' can be found in Gene Ontology using the select method with GO.db:

```
library(GO.db)
select(GO.db, keytype="TERM",
keys="Golgi membrane", columns=c("GOID",
    "DEFINITION",
      "ONTOLOGY"))
##           TERM      GOID
## 1 Golgi membrane GO:0000139
##                            DEFINITION
## 1 The lipid bilayer surrounding any of the
## compartments of the Golgi apparatus.
## ONTOLOGY
## 1    CC
```

Once the formal identifier is obtained, the org.Hs.eg.db package can be used to find mappings from the GO term to gene and protein identifiers. This generates a fairly large table:

```
library(org.Hs.eg.db)
go139 = select(org.Hs.eg.db, keys="GO:0000139", keytype="GO",
columns=c("ENTREZID", "SYMBOL", "PFAM"))
dim(go139)
## [1] 1212 6
head(go139)
##         GO EVIDENCE ONTOLOGY ENTREZID SYMBOL   PFAM
## 1 GO:0000139     TAS      CC       28   ABO     PF03414
## 2 GO:0000139     IEA      CC      102 ADAM10    PF00200
## 3 GO:0000139     IEA      CC      102 ADAM10    PF13574
## 4 GO:0000139     IEA      CC      102 ADAM10    PF01562
## 5 GO:0000139     TAS      CC      162  AP1B1    PF09066
## 6 GO:0000139     TAS      CC      162  AP1B1    PF01602
```

The evidence code TAS means that there is a "traceable author statement" associating the term of interest with the gene identified. The number of genes in traceable Golgi membrane:gene associations is found with

```
go139 |> dplyr::filter(EVIDENCE=="TAS") |>
   distinct(ENTREZID) |> count()
##      n
## [1] 327
```

4.3.2 Ontology usage with rols

Access to a vast collection of ontologies is afforded by the EBI's Ontology Lookup Service (OLS). The rols package uses the OLS API to discover ontologic mapping of terms of interest. Here we'll consider the term "golgi membrane dynamics", which is not found in GO. Again a multistep process is used.

```
library(rols)
lk1 = OlsSearch(q="golgi membrane dynamics", exact TRUE)
lk1
## Object of class 'OlsSearch':
##query: golgi membrane dynamics
##requested: 20 (out of 3)
##response(s): 0
```

In this first step, we find how extensive is the response to the query. Certain searches yield tens of thousands of hits. With the exact parameter setting, the yield is modest. Now we extract a data.frame after requesting all records with `olsSearch`. Results are excerpted in Table 5.

```
lk2 = olsSearch(lk1)
lk3 = as(lk2, "data.frame")
lk3$description = unlist(lk3$description)
```

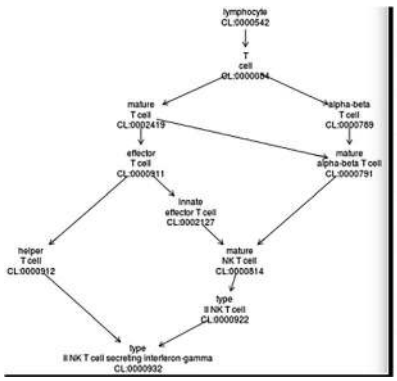
The detailed descriptions of the NCI Thesaurus entries show the exact nature of the search outcome.

4.3.3 Cross-ontology relationships

Philosophically, ontology is the study of what there is. For applications in information science, boundaries need to be established so that ontological resources can be managed with well-defined scopes. In Gene Ontology, three sub-ontologies are explicitly identified for cellular components, biological processes, and molecular functions.

Table 5
Using rols to obtain ontologic information related to golgi membrane dynamics.

short_form	description	label
NCIT_C119637	This gene is involved in both protein ubiquitination and Golgi membrane dynamics.	HACE1 Gene
NCIT_C119639	E3 ubiquitin-protein ligase HACE1 (909 aa, ~102 kDa) is encoded by the human HACE1 gene. This protein is involved in the regulation of both the ubiquitination and subsequent degradation of small GTPases, which modulates Golgi membrane dynamics.	E3 Ubiquitin-Protein Ligase HACE1
NCIT_C119638	Human HACE1 wild-type allele is located in the vicinity of 6q16.3 and is approximately 132 kb in length. This allele, which encodes E3 ubiquitin-protein ligase HACE1 protein, plays a role in the modulation of both Golgi membrane dynamics and ubiquitination. Mutations of the gene, including translocations that either reduce expression of the gene (t(6;15)(q21;q21)) or truncate the gene (t(5;6)(q21;q21)), are associated with Wilms tumor.	HACE1 wt Allele



query	tag	prtag	cond	entity	SYMBOL	name
CL:0000932	CL:0000932	PR:000001343	hasPMP	CD69 molecule	CD69	type II NK T cell secreting interferon-gamma
CL:0000932	CL:0000814	PR:000001874	hasPMP	KLRB1-like protein		mature NK T cell
CL:0000932	CL:0000789	GO:0042105	hasPMP	alpha-beta T cell receptor complex		alpha-beta T cell
CL:0000932	CL:0000542	CP:0000000	hasPart			lymphocyte
CL:0000932	CL:0000542	CP:0000037	hasPart			lymphocyte
CL:0000932	CL:0000542	GO:0000792	hasPart	heterochromatin		lymphocyte
CL:0000932	CL:0000542	GO:0030141	lacksPart	secretory granule		lymphocyte
CL:0000932	CL:0002419	GO:0042101	hasPMP	T cell receptor complex		mature T cell

Fig. 6 Ontology visualization and tabulation with ontoProc::ctmarks.

As knowledge of cell biology increases, the typology of cells becomes more and more intricate. Differentiation and definition of "cell types" involves concepts from immunology, protein science, anatomy, and other conceptual domains for which ontologies have been developed. Figure 6 presents, on the left, the hierarchy of cell type concepts starting at "lymphocyte", leading to "Type II Natural Killer T cell secreting interferon gamma". On the right, some of the GO and Protein Ontology (PR) cross-references in the Cell Ontology (CL) entry for the Type II NK cell are shown. The "cond" column of the table contains abbreviated tokens representing formal relationships linking the cell type to the protein or cellular component elements of PR and GO. The token "hasPMP" refers to the element of the Relation Ontology (RO) "has plasma membrane part" (RO:0002104).

Prospects for use of ontological discipline in the definition of new cell types are reviewed in a 2018 paper from the Venter Institute [2].

The field of biological ontology is rapidly advancing, and the integration of ontology search and inference with data analytic frameworks requires more effort at this time.

5 Analytical workflows

5.1 Overview

Table 6 presents an informal topical labeling for Bioconductor software packages with cancer mentioned in the Description field of package metadata.

The vignettes of each of these packages provide background and illustration of their roles in cancer genomics.

Table 6
Topical organization of packages with cancer applications.

topic	packages
Ancestry	RAIDS
Biomarkers	INDEED, iPath, RLassoCox
ceRNA	GDCRNATools
Clonal Evolution	CIMICE, LACE, OncoSimulR, TRONCO, CancerInSilico, cellscape
CNV	oncoscanR, SCOPE, ZygosityPredictor
DrugSensitivity	DepInfeR, octad, PharmacoGx, rcellminer
Epigenetics	MethylMix, AMARETTO, COCOA, methylclock, missMethyl
HotSpots/Drivers/signatures	compSPOT, MoonlightR, Moonlight2R,
	DriverNet, genefu, mastR, pathifier, RESOLVE, macat,
	SigCheck, signeR, signifinder, supersigs, decompTumor2Sig, YAPSA
ImmuneModulation	easier
IsoformSwitching	IsoformSwitchAnalyzeR
Literature mining	OncoScore
ncRNA	NoRCE
Radiomics	RadioGx
RecurrentFusion	copa, oppar
Spatial	SpatialDecon
SpecificCancers	consensusOV, PDATK, STROMA4
Splicing	OutSplice, psichomics
Subtyping	SCFA

5.2 Packages supporting epigenomic analysis

Bioconductor also provides a diverse array of packages for analysis of epigenome data. Cancer is often studied under a developmental lens, so increasingly, studies are measuring cell states using epigenomic methods. Epigenomics is the study of chemical modifications and chromosomal conformations of DNA in a nucleus; in cancer epigenomics, we study how the cancer epigenome differs among cancers and how these relate to healthy epigenomes. As of 2023, Bioconductor includes 89 packages under *Epigenetics* and 93 packages tagged under *FunctionalGenomics*, including dozens of tools for analyzing a variety of epigenome assays, such as ATAC-seq, ChIP-seq, or bisulfite-seq. Among these are also tools that handle more general analysis, such as genomic region set enrichment.

First, for ATAC-seq data, Bioconductor packages include general-purpose pipelines, including scPipe [3]. and esATAC [4] which start from FASTQ files and produce feature count matrices. Alternatively, many practitioners elect to do general-purpose pipeline processing outside of R, and then bring the processed data into R for statistical analysis, visualization, and quality control. In this approach, ATACseqQC provides a variety of QC plots specific to ATAC-seq data [5].

For DNA methylation, many popular packages have been developed to help with all stages of a DNA methylation analysis. These include minfi [6] which specializes in methylation array analysis, biseq and bsseq [7] which provide fundamental infrastructure for sequencing-based assays, and RnBeads [8], which provides a comprehensive general-purpose analysis of DNA methylation cohorts from arrays or sequencing-based assays. Other packages provide more specialized analysis approaches, such as MIRA [9], which infers regulatory activity of transcription factors using DNA methylation signals, or ELMER, which uses DNA methylation and gene expression in large cancer cohorts to infer transcription factor networks [10]. EpiDISH infers the proportions of cell-types present in a bulk sample on the basis of DNA methylation data [11].

DiffBind [12] facilitates differential binding analysis of ChIP-seq peak data.

GenomicDistributions [13] provides a variety of plots for visualization distributions of any type of genomic range data. The chromPlot package specializes in plots across chromosomes. Several packages deal with unsupervised exploration of variation in epigenomic data. PathwayPCA, MOFA2 [14] and COCOA [15] can process any epigenomic signal data. A variety of alternative approaches for enrichment analysis, which include LOLA [16], chipenrich, regioneR [17], and FGNet [18]. Annotation packages include ChIPpeakAnno [19] and annotatr [20].

5.3 Some details on prediction of responsiveness to immune checkpoint blockade

The National Cancer Institute website on checkpoint inhibitors in cancer immunotherapy ("Immune Checkpoint Inhibitors" [21]) lists 12 different cancer types amenable to treatment via immune checkpoint inhibition. The "easier" package in Bioconductor assembles multiple systems biology resources to produce patient-specific prediction of responsiveness to immune checkpoint blockade (ICB) [22].

Figure 7 presents on overview of results of immune response assessment in a cohort of patients with bladder cancer [23]. Patient's bulk RNA-seq data are used to develop multiple quantitative descriptors of the tumor microenvironment, and scores for processes regarded as hallmarks of anti-cancer immune responses.

This display encapsulates a) the capacity of measurements of genomic elements to discriminate patients who respond to ICB for bladder cancer (position of labeled item on x axis), b) the direction of association of element activity with immune response (shape of glyph) and c) the relative magnitudes of weights (size of glyph) estimated for features in initial model fitting.

The design of this package is noteworthy in its approach to information hiding. Parameters estimated in machine learning of

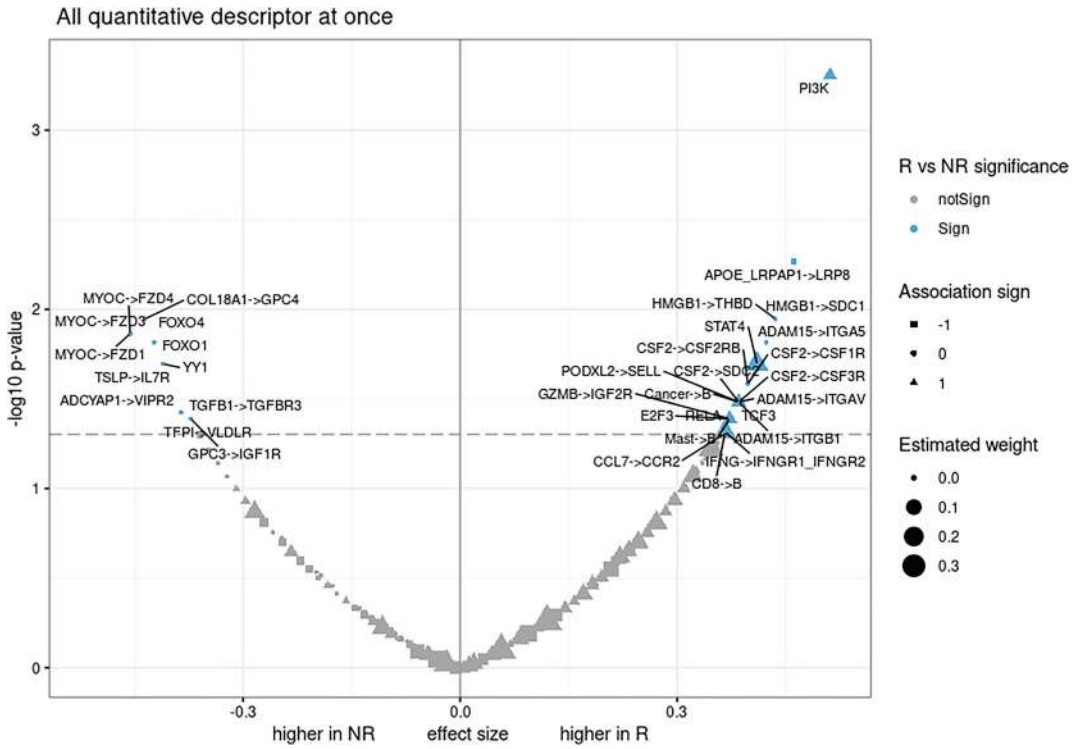


Fig. 7 Comparison of genomic features distinguishing patients non-responsive and responsive to immune checkpoint blockade.

tissue-specific relations between quantitative descriptors of the tumor microenvironment and hallmarks of immune response are stored in ExperimentHub.

```
library(easierData)
list_easierData()
##   eh_id                        title
##  EH6677  Mariathasan2018_PDL1_treatment
##  EH6678                    opt_models
##  EH6679              opt_xtrain_stats
##  EH6680           TCGA_mean_pancancer
##  EH6681             TCGA_sd_pancancer
##  EH6682             cor_scores_genes
##  EH6683            intercell networks
##  EH6684             lr_frequency_TCGA
##  EH6685               group_lr_pairs
##  EH6686              HGNC_annotation
##  EH6687         scores_signature_genes
```

The structure of the stored model weights resource can be sketched by probing list elements.

```
mw = eh[["EH6678"]]
## see ?easierData and browseVignettes('easierData') for
##    documentation
## loading from cache
names(mw)   # TCGA tumor types
## [1] "LUAD" "LUSC" "BLCA" "BRCA" "CESC" "CRC" "GBM" "HNSC" "KIRC"
## [10] "KIRP" "LIHC"   "OV" "PAAD" "PRAD" "SKCM" "STAD"
##       "THCA" "UCEC"
## [19] "NSCLC"
names(mw[["LUAD"]]) # TME descriptors
## [1] "pathways" "immunecells" "tfs" "lrpairs" "ccpairs"
rownames(mw[["LUAD"]]$pathways$CYT) # predict cytolytic
#                             # activity
##  [1] "(Intercept)" "Androgen" "EGFR" "Estrogen"   "Hypoxia"
## [6] "JAK-STAT"  "MAPK" "NFkB" "p53" "PI3K"
## [11]       "TNFa"   "Trail" "VEGF" "WNT"
```

The vignette of the easier package steps through phases, using these tumor-type-specific weights to compute patient-specific measures of transcription factor activity or cell-cell interaction on the basis of bulk RNA-seq (units are transcripts per million), and a patient-specific measure of pathway activity using raw RNA-seq counts. These metrics may be of interest in their own

right for applications other than establishing predictions of response to ICB.

Section 9 provides the names and versions of all packages used to produce this analysis.

5.4 Representing and visualizing spatial transcriptomics experiments

Spatial transcriptomics (ST) allows the quantification of RNA expression of large numbers of genes while preserving the spatial context of tissues and cells. This is important as cancer progression depends on a complex tumor microenvironment, and not just cell type composition, but also cell type spatial organization can be used to derive diagnostic or prognostic markers.

The Bioconductor project offers multiple approaches to handle and manipulate spatial transcriptomics data. The SpatialExperiment class [24] is designed to be a lightweight, technology-agnostic container. By inheriting from the SingleCellExperiment class, it unlocks the use in ST data of analysis packages designed for single-cell data, such as scater for exploration and QC, and scran for normalization. SpatialFeatureExperiment [25] extends SpatialExperiment to easily reuse polygons and other spatial geometry features from geospatial CRAN packages, such as sf. See also MoleculeExperiment [26] for a different approach based on the data.table package.

In addition to data containers, Bioconductor provides a rich set of ST data. The STexampleData and SFEData packages contain a collection of datasets from different technologies and tissues. As of December 2023, the TENxVisiumData package provides a collection of 13 in-house 10X Genomics Visium datasets from 23 samples across two organisms (human and mouse) and 13 tissues. The MerfishData package contains two annotated samples assayed with the MERFISH in-situ imaging protocol.

Finally, Bioconductor offers a growing collection of analysis methods tailored for spot-based and in-situ ST data, including methods for visualization, data exploration and quality control, spot deconvolution, spatially-aware clustering, and identification of spatially-variable genes.

To show a simple example of an analysis workflow on spot-based data, we explore a fresh frozen Invasive Ductal Carcinoma breast tissue assayed with the 10X Genomics Visium platform. First, we use the ggspavis package for visualization. See Figure 8.

```
library(TENxVisiumData)
## snapshotDate(): 2023-10-24
library(SpatialExperiment)
library(ggspavis)
hbc <- HumanBreastCancerIDC()
## see ?TENxVisiumData and browseVignettes ('TENxVisiumData')
##       for documentation
```

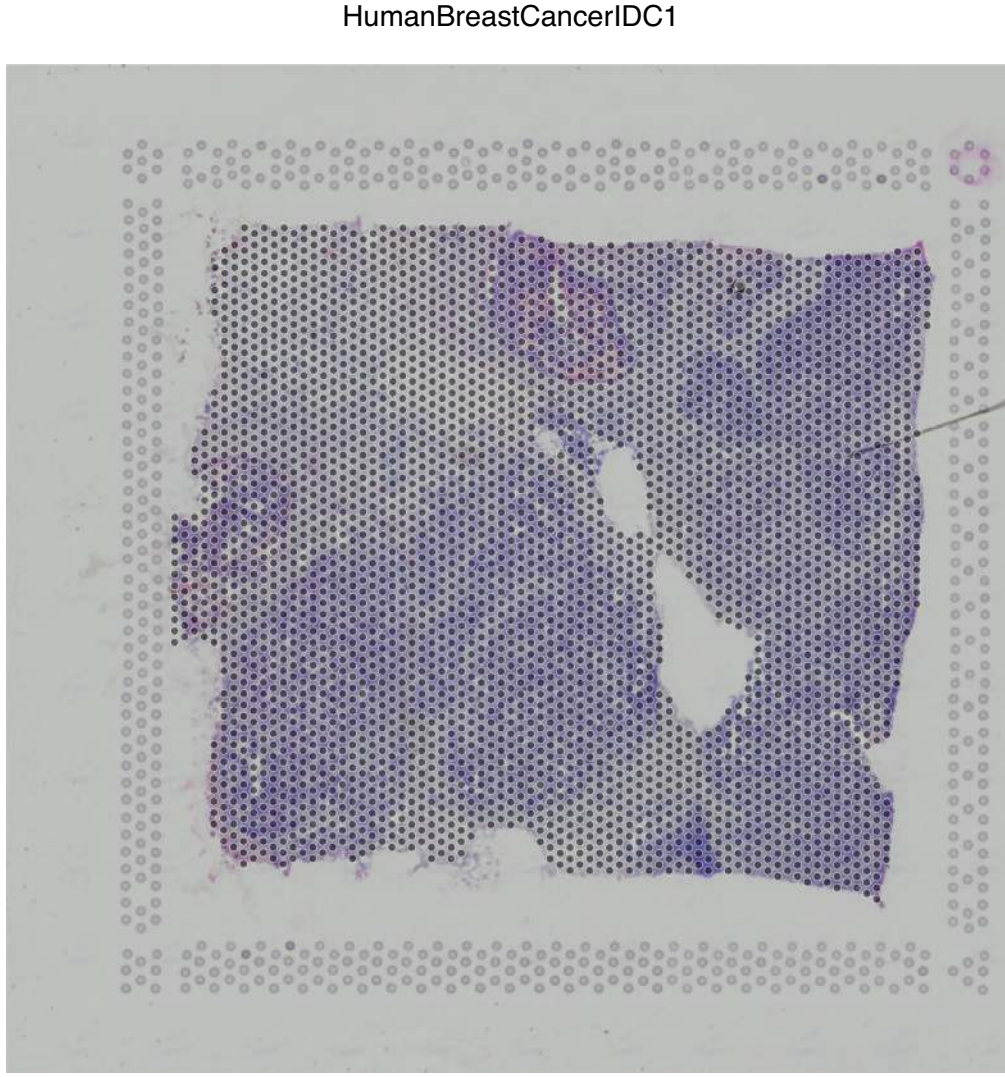


Fig. 8 Visualization of a Visium breast cancer sample

```
## loading from cache
hbc <- hbc[,hbc$sample_id=="HumanBreastCancerIDC1"]
hbc$in_tissue <- TRUE
hbc <- rotateImg(hbc, degrees=-90)
plotVisium(hbc, y_reverse = FALSE)
```

To investigate the spatially variable genes the nnSVG package implements a method for the detection of genes whose expression varies in the tissue spatial domains by fitting nearest-neighbor Gaussian processes [27].

```
library(scater)
library(nnSVG)
library(scran)
#add quality metrics
is_mito <- grepl("(^MT-)|(^mt-)", rowData(hbc)$symbol)
hbc <- addPerCellQC(hbc, subsets = list(mito = is_mito))
## needed because the column name is hard coded in
##     the nnSVG::filter_genes
rowData(hbc)$gene_name <- rowData(hbc)$symbol
## filter and normalize gene expression
hbc <- filter_genes(hbc)
## Gene filtering: removing mitochondrial genes
## removed 13 mitochondrial genes
## Gene filtering: retaining genes with at least 3 counts
##      in at least 0.5% (n = 19) of spatial locations
## removed 26583 out of 36588 genes due to low expression
hbc <- computeLibraryFactors(hbc)
hbc <- logNormCounts(hbc)
## select highly variable genes
hvgs <- getTopHVGs(hbc, n=1000)
hbc <- hbc[hvgs,]
## identify spatially variable genes
hbc <- nnSVG(hbc, n_threads=4)
## post-processing
hbc <- hbc[order(rowData(hbc)$rank),]
gnr1 <- rowData(hbc)$symbol[1]
rownames(hbc) <- rowData(hbc)$symbol
```

By ranking the results of nnSVG, we are able to detect the most spatially variable genes. As an example, Figure 9 shows how the most spatially variable gene varies across the tissue.

```
plotVisium(hbc, y_reverse = FALSE, fill = gnr1, palette="red")
```

Finally, we show an example of an in-situ ST technology, by visualizing a breast cancer sample assayed with the 10X Genomics Xenium platform.

```
library(SpatialFeatureExperiment)
library(SFEData)
jbr = JanesickBreastData("rep1")
jbr
## class: SpatialFeatureExperiment
## dim: 541 167782
## metadata(1): Samples
```

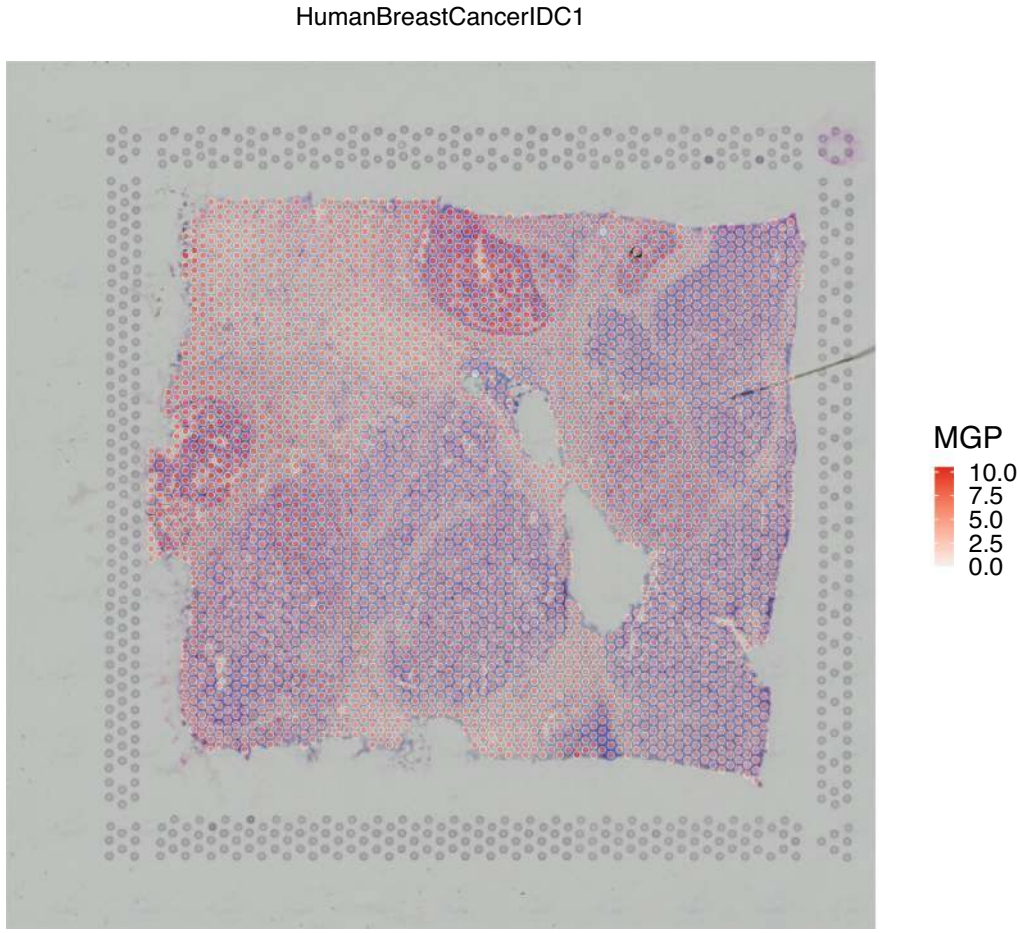


Fig. 9 Spatial expression of a highly variable gene

```
## assays(1): counts
## rownames(541): ABCC11 ACTA2 ... BLANK_0497 BLANK_0499
## rowData names(6): ID Symbol ... vars cv2
## colnames: NULL
## colData names(10): Sample Barcode ... nCounts nGenes
## reducedDimNames(0):
## mainExpName: NULL
## altExpNames(0):
## spatialCoords names(2) : x_centroid y_centroid
## imgData names(1): sample_id
##
## unit:
## Geometries:
## colGeometries: centroids (POINT), cellSeg (POLYGON),
```

```
##     nucSeg (GEOMETRY)
##
## Graphs:
## sample01:
```

We can leverage the nature of in-situ data to explore the cell density across the tissue, identifying the tissue's macrostructure, and the cell segmentation, zooming in on a small portion of the tissue. *See* Figure 10.

```
library(Voyager)
cellbins <- plotCellBin2D(jbr, hex = TRUE)
cellgeo <- plotGeometry(jbr, "cellSeg",
  bbox=c("xmin"=0, "ymin"=4000, "xmax"=1000, "ymax"=5000))
library(gridExtra)
grid.arrange(cellbins, cellgeo, ncol=2)
```

Finally, we can visualize the expression of marker genes after log-normalizing the data (Fig. 11).

```
jbr <- jbr[, jbr$nCounts >= 20]
jbr <- logNormCounts(jbr)
library(scattermore)
strom <- plotSpatialFeature(jbr, "POSTN",
                colGeometryName = "centroids",
scattermore = TRUE, ncol = 2, pointsize = 0.5) +
```

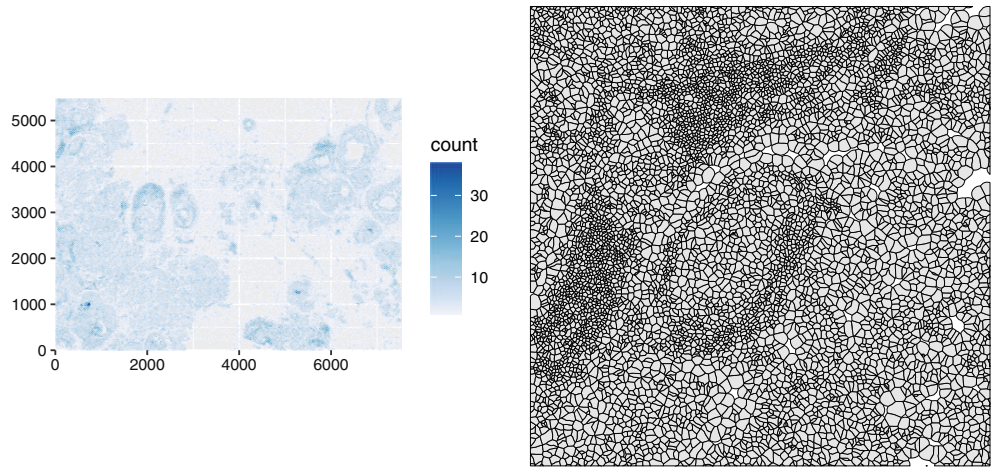


Fig. 10 Cell density and cell boundaries of a Xenium breast cancer sample

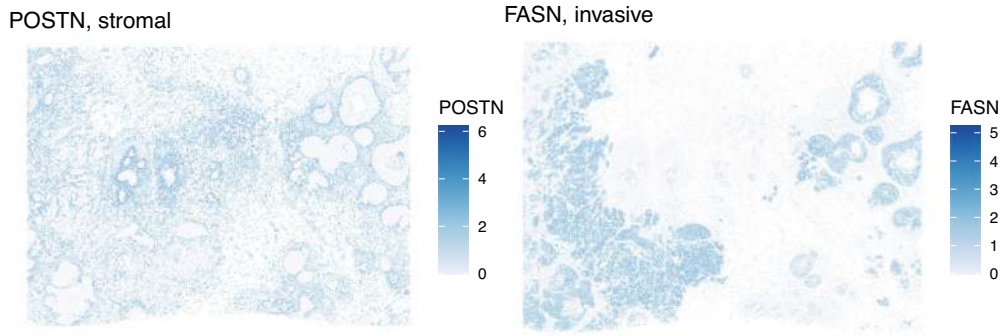


Fig. 11 Spatial expression of marker genes

```
                ggtitle("POSTN, stromal")
fasn <- plotSpatialFeature(jbr, "FASN",
                colGeometryName = "centroids",
scattermore = TRUE, ncol = 2, pointsize = 0.5) +
ggtitle("FASN, invasive")
grid.arrange(strom, fasn, ncol=2)
```

6 Components and processes for introducing new data, analytic tools, documents

6.1 Contributions and review

Proposed contributions to Bioconductor's ecosystem of software packages, data resources, and documentation are registered at

`https://github.com/bioconductor/contributions/issues`

Contributors identify a public github.com repository that houses their software, or some durable open data repository for a data contribution. The contributor provides schematized information on format, licensing, and commitment to maintenance of the contributed resource. After a series of automated and manual verification steps, the contributed resource enters the review process.

An example under review in December 2023 is the "methodical" package, submitted 27 September 2023. The issue number at the contributions site is 3169. This contribution is of particular interest as it addresses new data resources from whole genome and reduced representation bisulfite sequencing experiments. Specifics on these high-resolution studies of DNA methylation in a variety of clinical situtions are given below.

6.2 Data structures

Inheritance is a key feature of object-oriented programming (OOP) that allows us to define a new class out of existing classes and add new features, which provides reusability of code. Inheritance carries

over attributes and methods defined for base classes; 'Attributes' are variables that are bound in a class. They are used to define behavior and methods for objects of that class. 'Methods' are functions defined within a class that receive an instance of the class, conventionally called self, as the first argument. The attributes defined for a base class will automatically be present in the derived class, and the methods for the base class will work for the derived class. The R programming language has three different class systems: S3, S4, and Reference. Inheritance in S3 classes does not have any fixed definition, and hence attributes of S3 objects can be arbitrary. Derived classes, however, inherit the methods defined for the base class. Inheritance in S4 classes is more structured, and derived classes inherit both attributes and methods of the parent class. Reference classes are similar to S4 classes, but they are mutable and have reference semantics.

S4 classes are used extensively in Bioconductor to create data structures that store complex information, such as biological assay data and metadata, in one or more slots. The entire structure can then be assigned to an R object, and the types of information in each slot of the object are tightly controlled. S4 generics and methods define functions that can be applied to these objects, providing a rich software development infrastructure while ensuring interoperability, reusability, and efficiency.

Bioconductor have established Bioconductor classes to represent different types of biological data. Data and tools distributed through Bioconductor adopt Bioconductor classes, providing convenient methods and improving usability and interoperability within the Bioconductor ecosystem.

Table 7
Overview of key datatypes and associated classes in Bioconductor.

Data Types	Bioconductor Classes
Genomic coordinates (1-based, closed interval)	GRanges
Groups of genomic coordinates	GRangesList
Ragged genomic coordinates	RaggedExperiment
Gene sets	GeneSet
Rectangular Features x samples	SummarizedExperiment
Multi-omics data	MultiAssayExperiment
Single-cell data	SingleCellExperiment
Spatial Transcriptomics	SpatialExperiment
Mass spectrometry data	Spectra

The GRanges class represents a collection of genomic ranges and associated annotations. Each element in the vector represents a set genomic ranges in terms of the sequence name (seqnames, typically the chromosome), start and end coordinates (ranges, as an IRanges object), strand (strand, either positive, negative, or unstranded), and optional metadata columns (e.g., exon_id and exon_name in the below).

```
GRanges object with 4 ranges and 2 metadata columns:
     seqnames          ranges strand |   exon_id
     exon_name
       <Rle>       <IRanges>  <Rle> | <integer>
       <character>
  [1]       X 99883667-99884983      - |    667145
            ENSE00001459322
  [2]       X 99885756-99885863      - |    667146
            ENSE00000868868
  [3]       X 99887482-99887565      - |    667147
            ENSE00000401072
  [4]       X 99887538-99887565      - |    667148
            ENSE00001849132
  -------
  seqinfo: 722 sequences (1 circular) from an
  unspecified genome
```

The GRangesList object serves as a container for genomic features consisting of multiple ranges that are grouped by a parent features, such as spliced transcripts that are comprised of exons. A GRangesList object behaves like a list and many of the same methods for GRanges objects are available for GRangesList object as well.

The SummarizedExperiment class (see Figure 1 is a matrix-like container, where rows represent features of interest (e.g., genes, transcripts, exons, etc.) and columns represent samples. The attributes of this object include experimental results (in assays), information on observations (in rowData) and samples (in colData), and additional metadata (in metadata). SummarizedExperiment objects can simultaneouly manage several experimental results as long as they are of the same dimensions. The best benefit of using SummarizedExperiment class is the coordination of the metadata and assays when subsetting. SummarizedExperiment is similar to the historical ExpressionSet class, but more flexible in its row information, allowing both GRanges and DataFrames. ExpressionSet object can be easily converted to SummarizedExperiment.

RangedSummarizedExperiment inherits the SummarizedExperiment class, with the extended capability of storing genomic ranges (as a GRanges or GRangesList object) of interest instead of a DataFrame (S4-class objectcs similar to data.frame) of features in rows.

The MultiAssayExperiment class (presented above in Figure 2) is modeled after the SummarizedExperiment class. A MultiAssayExperiment instance `M` can be filtered as a three-dimensional array. When `G` is a vector of feature identifiers, `C` a vector of sample identifiers, and `E` a vector of experiment names, then `M[G, C, E]` is a MultiAssayExperiment with content restricted to the requested features, samples, and experiments. The MultiAssayExperiment package includes tooling to convert data content to "long" or "wide" formats. In long format, each element of the assay array occupies a row, accompanied by metadata associated with the element. In wide format, each sample occupies a row, accompanied by all assocated assay and metadata elements.

6.3 Out-of-memory data representation strategies

We return to the "methodical" package submission mentioned above. A number of whole-genome bisulfite sequencing experiments on tumors from various anatomic sites are available in ExperimentHub. Metadata in that package shows that the datasets are large, ranging from 2–40 gigabytes. One smaller dataset is provided for illustration.

```
library(TumourMethData)
demm = download_meth_dataset("mcrpc_wg ..." ... [TRUNCATED]
demm
## class: RangedSummarizedExperiment
## dim: 1333114 100
## metadata(5): genome is_h5 ref_CpG chrom_sizes
##     descriptive_stats
## assays(2): beta cov
## rownames: NULL
## rowData names(0):
## colnames(100): DTB_003 DTB_005 ... DTB_265 DTB_266
## colData names(4): metastatis_site subtype age sex rowRanges(demm)
## GRanges object with 1333114 ranges and 0 metadata columns:
##           seqnames    ranges strand
##              <Rle> <IRanges>  <Rle>
##       [1]    chr11     60077      *
##       [2]    chr11     60088      *
##       [3]    chr11     60365      *
##       [4]    chr11     60941      *
##       [5]    chr11     60979      *
##       ...      ...       ...    ...
##   [1333110]    chr11 135076482      *
##   [1333111]    chr11 135076496      *
##   [1333112]    chr11 135076502      *
##   [1333113]    chr11 135076507      *
##   [1333114]    chr11 135076510      *
##   -------
##   seqinfo: 25 sequences from an unspecified genome; no seqlengths
```

```
names(colData(demm))
## [1] "metastatis_site" "subtype"       "age"          "sex"
table(demm$metastatis_site)
##     Bone     Liver Lymph_node     Other
##       43        11        38         8
```

References to `demm` involve an 800MB excerpt of a prostate cancer atlas with a storage footprint of 40GB. Ideally, queries about particular genomic regions on particular samples, whole-sample statistical summaries, and searches for patterns can be carried out without specific accommodation of the data size or representation. The DelayedArray package helps pursue this aim. We'll illustrate by interrogating the prostate cancer WGBS data for "beta" (fraction of locus that is methylated) values in the vicinity of gene ATM.

```
library(EnsDb.Hsapiens.v86)
gg = genes(EnsDb.Hsapiens.v86)
# get gene addresses
atmpos = gg[gg$gene_name == "ATM" &
gg$gene_biotype == "protein_coding"] # filter to ATM
seqlevelsStyle(atmpos) = "UCSC"
assay(subsetByOverlaps(demm, atmpos+1e6))
## <18110 x 100> DelayedMatrix object of type "double":
##         DTB_003 DTB_005 DTB_008 ... DTB_265 DTB_266
##    [1,]  0.1053  0.7660  0.9206   .  0.6944  0.9412
##    [2,]  0.4062  0.9091  0.9318   .  0.5676  1.0000
##    [3,]  0.1379  0.0000  0.7400   .  0.4643  0.9231
##    [4,]  0.2308  0.9231  0.9149   .  0.8929  0.9286
##    [5,]  0.1481  0.8500  0.8864   .  0.8710  0.9762
##     ...      .       .       .   .       .       .
## [18106,]  0.4138  0.3143  0.3208   . 0.17647 0.10000
## [18107,]  0.2727  0.2745  0.4143   . 0.22500 0.32500
## [18108,]  0.2258  0.4800  0.5775   . 0.08889 0.25000
## [18109,]  0.5278  0.7059  0.8088   . 0.55263 0.97561
## [18110,]  0.2778  0.3137  0.6957   . 0.52632 0.35714
```

The numeric values presented above are just the "corners" of the associated array, presented as a "check" on the content requested. Transfer of array content to the CPU for numerical analysis only occurs on demand, which can be tailored to the quantity of RAM available at analysis time.

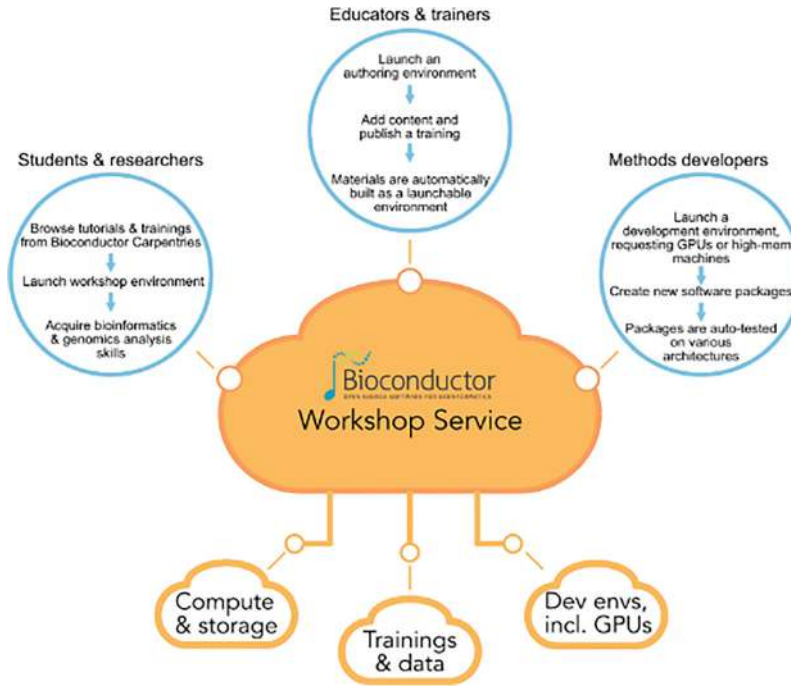


Fig. 13 Workshop.bioconductor.org schematic.

6.4 Quality assessment of Bioconductor resources

Figure 12 is an overview of the periodic ecosystem testing process for Bioconductor software packages in the release branch. All Bioconductor and CRAN packages on which they depend are present and are updated on change to sources.

The project distributes source tarballs for Linux-like systems, and compiled binaries for MacOS and Windows. Numbers in red boxes indicate failures to install, build, or check. Failure events are frequently platform-specific; full logs are provided on the build report pages to help developers isolate and fix build and check errors. When failures are persistent, developers are contacted by core. If contact cannot be made and failures continue, packages are deprecated for at least one release, and then removed.

7 Pedagogics and workforce development

The Bioconductor project has undertaken a number of initiatives to support growth of the scientific workforce's capacity to efficiently integrate and interpret genome-scale experiments.

- **Partnering with The Carpentries.** The Carpentries (https://carpentries.org) is a non-profit organization focused on teaching programming and data science to researchers. The organization defines "good practices in lesson design and development, and open source collaboration skills". Bioconductor community

members have created bioc-intro, bioc-project, and bioc-rnaseq repositories using The Carpentries Incubator template. This arrangement helps Bioconductor create and manage a "train the trainer" process according to tested pedagogical principles.

- **Curating monographs for topics in genomic data science.** The breadth of Bioconductor resources for genomics, combined with the energetic approach to software and annotation upkeep in the project, empowers Bioconductor developers to produce unified, wide-ranging, computable documents on topics of interest to the broader cancer genomics community. Books currently available at bioconductor.org include OSCA (Orchestrating Single Cell Analysis with Bioconductor), SingleRBook (Assigning cell types with SingleR), csawBook (Analysis of ChIP-seq data), OHCA (Orchestrating Hi-C Analysis with Bioconductor) and R for Mass Spectrometry. Very recently, Jacques Serizay of Institut Pasteur has contributed a book authoring framework called BiocBook. This transforms documents marked up in Posit's quarto format into web-based books backed up by Docker containers and maintained with templated GitHub actions. The OHCA book is produced and managed with BiocBook.
- **A system for authoring and deploying interactive workshops.**

Figure 13 gives an overview of the resources and objectives of the system underlying workshop.bioconductor.org. Given a kubernetes-enabled cluster the workshop system assembles

- compute and storage elements,
- static components (training texts and shareable data),

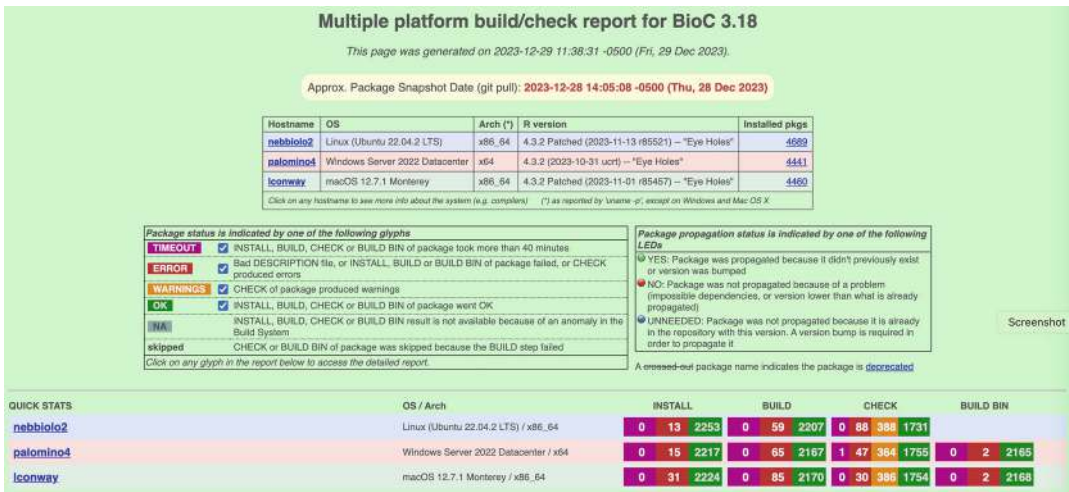


Fig. 12 Build report for Bioc 3.18, 12-29-2023.

- development environments (containers with all runtime elements required to compiled code, conduct analyses, communicate with GPUs).

A lightly customized deployment of the Galaxy system (usegalaxy.org) is used to deal with authentication and process initiation and termination.

This system has been used to serve interactive workshops in a number of international conferences. Content in R markdown or quarto can be produced by anyone interested in offering a workshop, and the "BiocWorkshopSubmit" app at workshop.bioconductor.org can be used to identify new content to the system. Markdown documents will be analyzed to determine what resources are needed for the containerization of workshop software and data components, and the container will be created and registered at the GitHub Container Registry. Arrangements to deploy the workshop over a given calendar period can be made with Bioconductor core. The workshop container can be used to conduct the workshop on any system with a Docker client.

8 Conclusions and paths forward

We have described several aspects of Bioconductor's approach to ecosystem management for cancer genomics data science resources. In light of the dynamism of biotechnological innovation, it is clear that the project must anticipate change. But it is challenging to introduce changes to processes on which a very large community depends for their daily research work. Commitments to supporting reproducible research entail that Bioconductor preserves decades worth of images of software and data for immediate retrieval via web request by parties unknown to the project.

We'll conclude this report with a few observations on general paths that the project is likely to take that should have favorable consequences to researchers in cancer genomics.

- **Language-agnostic data and annotation** The `alabaster.*` packages introduced in Bioconductor 3.17 are designed to convert existing Bioconductor data structures to formats that are more readily ingested by software in other languages. Thus the `alabaster.mae` package will convert a MultiAssayExperiment into a collection of files of metadata (serialized in JSON), sample-level data (serialized as CSV), and assay data (serialized to HDF5).
- **Zero-configuration genomic analysis environments** Users of Docker containers have long been able to take advantage of Bioconductor containers pre-populated with Rstudio and runtime resources to support installation of any desired

software packages. The bioc2u system (https://github.com/bioconductor/bioc2u) in conjunction with r2u (github.com/eddelbuettel/r2u) introduces the availability of Debian packages for all Bioconductor packages, made available in a CRAN-like repository. Given a system running Ubuntu 22 or 20, the apt package manager will resolve any package requests with tested, fully linked binary packages. Users do not have to perform any configuration or compilation of system utilities or package code. This practice can greatly reduce resource consumption that occurs when individuals or workflow systems need to compile every package and its dependencies to perform analyses.

- **Computation at the data** Several members of Bioconductor's development core are on the technical development team of NHGRI's Analysis and Visualization Laboratory (AnVIL). The aim of this project is to overthrow the prevalent model of downloading data for local analysis. AnVIL mobilizes commercial cloud computing and storage to support truly elastic genomic analysis – create and pay for only the computation you need. The basic strategy is described in Schatz et al. [28]. used in the production of the Telomere-to-Telomere genome build, see Aganezov et al. [29].

We hope that the project can continue to support researchers in cancer genomics for another 20 years!

9 Figure 7 software

Package	Version	Date(UTC)	Source
abind	1.4-5	2016-07-21	RSPM (R 4.2.0)
AnnotationDbi	1.64.1	2023-11-03	Bioconductor
AnnotationHub	3.10.0	2023-10-24	Bioconductor
backports	1.4.1	2021-12-13	RSPM (R 4.2.0)
bcellViper	1.38.0	2023-10-26	Bioconductor
Biobase	2.62.0	2023-10-24	Bioconductor
BiocFileCache	2.10.1	2023-10-26	Bioconductor
BiocGenerics	0.48.1	2023-11-01	Bioconductor
BiocManager	1.30.22	2023-08-08	RSPM (R 4.2.0)
BiocParallel	1.36.0	2023-10-24	Bioconductor
BiocVersion	3.18.0	2023-04-25	Bioconductor
Biostrings	2.70.1	2023-10-25	Bioconductor
bit	4.0.5	2022-11-15	RSPM (R 4.2.0)

(continued)

Package	Version	Date(UTC)	Source
bit64	4.0.5	2020-08-30	RSPM (R 4.2.0)
bitops	1.0-7	2021-04-24	RSPM (R 4.2.0)
blob	1.2.4	2023-03-17	RSPM (R 4.2.0)
broom	1.0.5	2023-06-09	RSPM (R 4.2.0)
bspm	0.5.5	2023-08-22	CRAN (R 4.3.1)
cachem	1.0.8	2023-05-01	RSPM (R 4.2.0)
car	3.1-2	2023-03-30	RSPM (R 4.2.0)
carData	3.0-5	2022-01-06	RSPM (R 4.2.0)
class	7.3-22	2023-05-03	RSPM (R 4.2.0)
cli	3.6.2	2023-12-11	RSPM (R 4.3.0)
codetools	0.2-19	2023-02-01	RSPM (R 4.2.0)
coin	1.4-3	2023-09-27	RSPM (R 4.3.0)
colorspace	2.1-0	2023-01-23	RSPM (R 4.2.0)
cowplot	1.1.2	2023-12-15	RSPM (R 4.3.0)
crayon	1.5.2	2022-09-29	RSPM (R 4.2.0)
curl	5.2.0	2023-12-08	RSPM (R 4.3.0)
DBI	1.1.3	2022-06-18	RSPM (R 4.2.0)
dbplyr	2.4.0	2023-10-26	RSPM (R 4.3.0)
decoupleR	2.8.0	2023-10-24	Bioconductor
DelayedArray	0.28.0	2023-10-24	Bioconductor
DESeq2	1.42.0	2023-10-24	Bioconductor
digest	0.6.33	2023-07-07	RSPM (R 4.2.0)
dorothea	1.14.0	2023-10-26	Bioconductor
dplyr	1.1.4	2023-11-17	RSPM (R 4.3.0)
e1071	1.7-14	2023-12-06	RSPM (R 4.3.0)
easier	1.8.0	2023-10-24	Bioconductor
easierData	1.8.0	2023-10-26	Bioconductor
ellipsis	0.3.2	2021-04-29	RSPM (R 4.2.0)
evaluate	0.23	2023-11-01	RSPM (R 4.3.0)
ExperimentHub	2.10.0	2023-10-24	Bioconductor
fansi	1.0.6	2023-12-08	RSPM (R 4.3.0)
farver	2.1.1	2022-07-06	RSPM (R 4.2.0)
fastmap	1.1.1	2023-02-24	RSPM (R 4.2.0)

(continued)

Package	Version	Date(UTC)	Source
filelock	1.0.3	2023-12-11	RSPM (R 4.3.0)
generics	0.1.3	2022-07-05	RSPM (R 4.2.0)
GenomeInfoDb	1.38.1	2023-11-08	Bioconductor
GenomeInfoDbData	1.2.11	<NA>	Bioconductor
GenomicRanges	1.54.1	2023-10-29	Bioconductor
ggplot2	3.4.4	2023-10-12	RSPM (R 4.3.0)
ggpubr	0.6.0	2023-02-10	RSPM (R 4.2.0)
ggrepel	0.9.4	2023-10-13	RSPM (R 4.3.0)
ggsignif	0.6.4	2022-10-13	RSPM (R 4.2.0)
glue	1.6.2	2022-02-24	RSPM (R 4.2.0)
gridExtra	2.3	2017-09-09	RSPM (R 4.2.0)
gtable	0.3.4	2023-08-21	RSPM (R 4.2.0)
htmltools	0.5.7	2023-11-03	RSPM (R 4.3.0)
htmlwidgets	1.6.4	2023-12-06	RSPM (R 4.3.0)
httpuv	1.6.13	2023-12-06	RSPM (R 4.3.0)
httr	1.4.7	2023-08-15	RSPM (R 4.2.0)
interactiveDisplayBase	1.40.0	2023-10-24	Bioconductor
IRanges	2.36.0	2023-10-24	Bioconductor
jsonlite	1.8.8	2023-12-04	RSPM (R 4.3.0)
KEGGREST	1.42.0	2023-10-24	Bioconductor
kernlab	0.9-32	2023-01-31	RSPM (R 4.2.0)
KernSmooth	2.23-22	2023-07-10	RSPM (R 4.2.0)
knitr	1.45	2023-10-30	RSPM (R 4.3.0)
labeling	0.4.3	2023-08-29	RSPM (R 4.2.0)
later	1.3.2	2023-12-06	RSPM (R 4.3.0)
lattice	0.22-5	2023-10-24	RSPM (R 4.3.0)
lazyeval	0.2.2	2019-03-15	RSPM (R 4.2.0)
libcoin	1.0-10	2023-09-27	RSPM (R 4.3.0)
lifecycle	1.0.4	2023-11-07	RSPM (R 4.3.0)
limSolve	1.5.7	2023-09-21	RSPM (R 4.3.0)
locfit	1.5-9.8	2023-06-11	RSPM (R 4.2.0)
lpSolve	5.6.20	2023-12-10	RSPM (R 4.3.0)
magrittr	2.0.3	2022-03-30	RSPM (R 4.2.0)

(continued)

Package	Version	Date(UTC)	Source
MASS	7.3-60	2023-05-04	RSPM (R 4.2.0)
Matrix	1.6-4	2023-11-30	RSPM (R 4.3.0)
MatrixGenerics	1.14.0	2023-10-24	Bioconductor
matrixStats	1.2.0	2023-12-11	RSPM (R 4.3.0)
memoise	2.0.1	2021-11-26	RSPM (R 4.2.0)
mime	0.12	2021-09-28	RSPM (R 4.2.0)
mixtools	2.0.0	2022-12-05	RSPM (R 4.2.0)
modeltools	0.2-23	2020-03-05	RSPM (R 4.2.0)
multcomp	1.4-25	2023-06-20	RSPM (R 4.2.0)
munsell	0.5.0	2018-06-12	RSPM (R 4.2.0)
mvtnorm	1.2-4	2023-11-27	RSPM (R 4.3.0)
nlme	3.1-164	2023-11-27	RSPM (R 4.3.0)
pillar	1.9.0	2023-03-22	RSPM (R 4.2.0)
pkgconfig	2.0.3	2019-09-22	RSPM (R 4.2.0)
plotly	4.10.3	2023-10-21	RSPM (R 4.3.0)
plyr	1.8.9	2023-10-02	RSPM (R 4.3.0)
png	0.1-8	2022-11-29	RSPM (R 4.2.0)
preprocessCore	1.64.0	2023-10-24	Bioconductor
progeny	1.24.0	2023-10-24	Bioconductor
promises	1.2.1	2023-08-10	RSPM (R 4.2.0)
proxy	0.4-27	2022-06-09	RSPM (R 4.2.0)
purrr	1.0.2	2023-08-10	RSPM (R 4.2.0)
quadprog	1.5-8	2019-11-20	RSPM (R 4.2.0)
quantiseqr	1.10.0	2023-10-24	Bioconductor
R6	2.5.1	2021-08-19	RSPM (R 4.2.0)
rappdirs	0.3.3	2021-01-31	RSPM (R 4.2.0)
Rcpp	1.0.11	2023-07-06	RSPM (R 4.2.0)
RCurl	1.98-1.13	2023-11-02	RSPM (R 4.3.0)
reshape2	1.4.4	2020-04-09	CRAN (R 4.0.1)
rlang	1.1.2	2023-11-04	RSPM (R 4.3.0)
rmarkdown	2.25	2023-09-18	RSPM (R 4.3.0)
ROCR	1.0-11	2020-05-02	RSPM (R 4.2.0)
RSQLite	2.3.4	2023-12-08	RSPM (R 4.3.0)

(continued)

Package	Version	Date(UTC)	Source
rstatix	0.7.2	2023-02-01	RSPM (R 4.2.0)
S4Arrays	1.2.0	2023-10-24	Bioconductor
S4Vectors	0.40.2	2023-11-23	Bioconductor 3.18 (R 4.3.2)
sandwich	3.1-0	2023-12-11	RSPM (R 4.3.0)
scales	1.3.0	2023-11-28	RSPM (R 4.3.0)
segmented	2.0-1	2023-12-19	RSPM (R 4.3.0)
sessioninfo	1.2.2	2021-12-06	RSPM (R 4.2.0)
shiny	1.8.0	2023-11-17	RSPM (R 4.3.0)
SparseArray	1.2.2	2023-11-07	Bioconductor
startup	0.21.0	2023-12-11	RSPM (R 4.3.0)
stringi	1.8.3	2023-12-11	RSPM (R 4.3.0)
stringr	1.5.1	2023-11-14	RSPM (R 4.3.0)
Summarized Experiment	1.32.0	2023-10-24	Bioconductor
survival	3.5-7	2023-08-14	RSPM (R 4.2.0)
TH.data	1.1-2	2023-04-17	RSPM (R 4.2.0)
tibble	3.2.1	2023-03-20	RSPM (R 4.3.0)
tidyr	1.3.0	2023-01-24	RSPM (R 4.2.0)
tidyselect	1.2.0	2022-10-10	RSPM (R 4.2.0)
utf8	1.2.4	2023-10-22	RSPM (R 4.3.0)
vctrs	0.6.5	2023-12-01	RSPM (R 4.3.0)
viper	1.36.0	2023-10-24	Bioconductor
viridisLite	0.4.2	2023-05-02	RSPM (R 4.2.0)
withr	2.5.2	2023-10-30	RSPM (R 4.3.0)
xfun	0.41	2023-11-01	RSPM (R 4.3.0)
xtable	1.8-4	2019-04-21	RSPM (R 4.2.0)
XVector	0.42.0	2023-10-24	Bioconductor
yaml	2.3.8	2023-12-11	RSPM (R 4.3.0)
zlibbioc	1.48.0	2023-10-24	Bioconductor
zoo	1.8-12	2023-04-13	RSPM (R 4.2.0)

Acknowledgements

This work was supported in part by NIH NCI 3U24CA180996-10S1, NHGRI 5U24HG004059-18, and NSF ACCESS allocation BIR190004.

References

1. R-Core. *Writing R Extensions*, 2024.
2. Aevermann, B. D., Novotny, M., Bakken, T., Miller, J. A., Diehl, A. D., Osumi-Sutherland, D., Lasken, R. S., Lein, E. S., and Scheuermann, R. H. Cell type discovery using single-cell transcriptomics: Implications for ontological representation. *Human Molecular Genetics*, 27:R40–R47, 2018.
3. Tian, L., Su, S., Dong, X., Amann-Zalcenstein, D., Biben, C., Seidi, A., Hilton, D. J., Naik, S. H., and Ritchie, M. E. scpipe: A flexible r/bioconductor preprocessing pipeline for single-cell rna-sequencing data. *PLOS Computational Biology*, 14(8):e1006361, 2018.
4. Wei, Z., Zhang, W., Fang, H., Li, Y., and Wang, X. esatac: An easy-to-use systematic pipeline for atac-seq data analysis. *Bioinformatics (Oxford, England)*, March 2018.
5. Ou, J., Liu, H., Yu, J., Kelliher, M. A., Castilla, L. H., Lawson, N. D., and Zhu, L. J. Atacseqqc: a bioconductor package for post-alignment quality assessment of atac-seq data. *BMC Genomics*, 19(1), 2018.
6. Aryee, M. J., Jaffe, A. E., Corrada-Bravo, H., Ladd-Acosta, C., Feinberg, A. P., Hansen, K. D., and Irizarry, R. A. Minfi: a flexible and comprehensive bioconductor package for the analysis of infinium dna methylation microarrays. *Bioinformatics*, 30(10): 1363–1369, 2014.
7. Hansen, K. D., Langmead, B., and Irizarry, R. A. Bsmooth: from whole genome bisulfite sequencing reads to differentially methylated regions. *Genome Biology*, 13(10):R83, 2012.
8. Müller, F., Scherer, M., Assenov, Y., Lutsik, P., Walter, J., Lengauer, T., and Bock, C. Rnbeads 2.0: comprehensive analysis of dna methylation data. *Genome Biology*, 20(1), 2019.
9. Lawson, J., Tomazou, E., Bock, C., and Sheffield, N. C. Mira: An R package for DNA methylation-based inference of regulatory activity. *Bioinformatics*, bty083, 3 2018.
10. Silva, T. C., Coetzee, S. G., Gull, N., Yao, L., Hazelett, D. J., Noushmehr, H., Lin, D.-C., and Berman, B. P. Elmer v.2: an r/bioconductor package to reconstruct gene regulatory networks from dna methylation and transcriptome profiles. *Bioinformatics*, 35(11): 1974–1977, 2019.
11. Zheng, S. C., Breeze, C. E., Beck, S., and Teschendorff, A. E. Identification of differentially methylated cell types in epigenome-wide association studies. *Nature Methods*, 15(12): 1059–1066, 2018.
12. Stark, R. and Brown, G. *DiffBind: differential binding analysis of ChIP-Seq peak data*, 2011.
13. Kupkova, K., Mosquera, J. V., Smith, J. P., Stolarczyk, M., Danehy, T. L., Lawson, J. T., Xue, B., Stubbs, J. T., LeRoy, N., and Sheffield, N. C. GenomicDistributions: fast analysis of genomic intervals with bioconductor. *BMC Genomics*, 23(1), apr 2022.
14. Argelaguet, R., Arnol, D., Bredikhin, D., Deloro, Y., Velten, B., Marioni, J. C., and Stegle, O. Mofa+: a statistical framework for comprehensive integration of multi-modal single-cell data. *Genome Biology*, 21(1), 2020.
15. Lawson, J. T., Smith, J. P., Bekiranov, S., Garrett-Bakelman, F. E., and Sheffield, N. C. COCOA: coordinate covariation analysis of epigenetic heterogeneity. *Genome Biology*, 21(1), sep 2020.
16. Sheffield, N. C. and Bock, C. Lola: enrichment analysis for genomic region sets and regulatory elements in R and bioconductor. *Bioinformatics*, 32(4):587–589, Oct 2016.
17. Gel, B., Diez-Villanueva, A., Serra, E., Buschbeck, M., Peinado, M. A., and Malinverni, R. regioneR: an r/bioconductor package for the association analysis of genomic regions based on permutation tests. *Bioinformatics*, page btv562, sep 2015.
18. Aibar, S., Fontanillo, C., Droste, C., and De Las Rivas, J. Functional gene networks: R/bioc package to generate and analyse gene networks derived from functional enrichment and clustering. *Bioinformatics*, 31(10): 1686–1688, 2015.
19. Zhu, L. J., Gazin, C., Lawson, N. D., Pagès, H., Lin, S. M., Lapointe, D. S., and Green, M. R. ChIPpeakAnno: a bioconductor package to annotate ChIP-seq and ChIP-chip data. *BMC Bioinformatics*, 11(1), may 2010.
20. Cavalcante, R. G. and Sartor, M. A. annotatr: genomic regions in context. *Bioinformatics*, 33(15):2381–2383, mar 2017.

21. Immune checkpoint inhibitors. https://www.cancer.gov/about-cancer/treatment/types/immunotherapy/checkpoint-inhibitors, 2022. Accessed: 2023-12-30.
22. Óscar Lapuente-Santana, van Genderen, M., Hilbers, P. A., Finotello, F., and Eduati, F. Interpretable systems biomarkers predict response to immune-checkpoint inhibitors. *Patterns*, 2, 8 2021.
23. Mariathasan, S., Turley, S. J., Nickles, D., Castiglioni, A., Yuen, K., Wang, Y., Kadel III, E. E., Koeppen, H., Astarita, J. L., Cubas, R., Jhunjhunwala, S., Banchereau, R., Yang, Y., Guan, Y., Chalouni, C., Ziai, J., Senbabaoglu, Y., Santoro, S., Sheinson, D., Hung, J., Giltnane, J. M., Pierce, A. A., Mesh, K., Lianoglou, S., Riegler, J., Carano, R. A. D., Eriksson, P., Hoglund, M., Somarriba, L., Halligan, D. L., van der Heijden, M. S., Loriot, Y., Rosenberg, J. E., Fong, L., Mellman, I., Chen, D. S., Green, M., Derleth, C., Fine, G. D., Hegde, P. S., Bourgon, R., and Powles, T. Tgfb attenuates tumour response to pd-l1 blockade by contributing to exclusion of t cells. *Nature*, 554(7693):544–548, Feb 2018.
24. Righelli, D., Weber, L. M., Crowell, H. L., Pardo, B., Collado-Torres, L., Ghazanfar, S., Lun, A. T., Hicks, S. C., and Risso, D. Spatialexperiment: infrastructure for spatially-resolved transcriptomics data in r using bioconductor. *Bioinformatics*, 38(11): 3128–3131, 2022.
25. Moses, L., Einarsson, P. H., Jackson, K., Luebbert, L., Booeshaghi, A. S., Antonsson, S., Bray, N., Melsted, P., and Pachter, L. Voyager: exploratory single-cell genomics data analysis with geospatial statistics. *bioRxiv*, 2023.
26. Couto, B. Z. P., Robertson, N., Patrick, E., and Ghazanfar, S. Moleculeexperiment enables consistent infrastructure for molecule-resolved spatial transcriptomics data in bioconductor. *bioRxiv*, 2023.
27. Weber, L. M., Saha, A., Datta, A., Hansen, K. D., and Hicks, S. C. nnsvg for the scalable identification of spatially variable genes using nearest-neighbor gaussian processes. *Nature communications*, 14(1):4059, 2023.
28. Schatz, M. C., Philippakis, A. A., Afgan, E., Banks, E., Carey, V. J., Carroll, R. J., Culotti, A., Ellrott, K., Goecks, J., Grossman, R. L., Hall, I. M., Hansen, K. D., Lawson, J., Leek, J. T., Luria, A. O., Mosher, S., Morgan, M., Nekrutenko, A., O'Connor, B. D., Osborn, K., Paten, B., Patterson, C., Tan, F. J., Taylor, C. O., Vessio, J., Waldron, L., Wang, T., Wuichet, K., Baumann, A., Rula, A., Kovalsy, A., Bernard, C., Caetano-Anollés, D., der Auwera, G. A. V., Canas, J., Yuksel, K., Herman, K., Taylor, M. M., Simeon, M., Baumann, M., Wang, Q., Title, R., Munshi, R., Chaluvadi, S., Reeves, V., Disman, W., Thomas, S., Hajian, A., Kiernan, E., Gupta, N., Vosburg, T., Geistlinger, L., Ramos, M., Oh, S., Rogers, D., McDade, F., Hastie, M., Turaga, N., Ostrovsky, A., Mahmoud, A., Baker, D., Clements, D., Cox, K. E., Suderman, K., Kucher, N., Golitsynskiy, S., Zarate, S., Wheelan, S. J., Kammers, K., Stevens, A., Hutter, C., Wellington, C., Ghanaim, E. M., Wiley, K. L., Sen, S. K., Francesco, V. D., s Yuen, D., Walsh, B., Sargent, L., Jalili, V., Chilton, J., Shepherd, L., Stubbs, B., O'Farrell, A., Vizzier, B. A., Overbeck, C., Reid, C., Steinberg, D. C., Sheets, E. A., Lucas, J., Blauvelt, L., Cabansay, L., Warren, N., Hannafious, B., Harris, T., Reddy, R., Torstenson, E., Banasiewicz, M. K., Abel, H. J., and Walker, J. Inverting the model of genomics data sharing with the nhgri genomic data science analysis, visualization, and informatics lab-space. *Cell Genomics*, 2:100085, 1 2022.
29. Aganezov, S., Yan, S. M., Soto, D. C., Kirsche, M., Zarate, S., Avdeyev, P., Taylor, D. J., Shafin, K., Shumate, A., Xiao, C., Wagner, J., McDaniel, J., Olson, N. D., Sauria, M. E., Vollger, M. R., Rhie, A., Meredith, M., Martin, S., Lee, J., Koren, S., Rosenfeld, J. A., Paten, B., Layer, R., Chin, C. S., Sedlazeck, F. J., Hansen, N. F., Miller, D. E., Phillippy, A. M., Miga, K. H., McCoy, R. C., Dennis, M. Y., Zook, J. M., and Schatz, M. C. A complete reference genome improves analysis of human genetic variation. *Science*, 376, 2022.

Chapter 2

Building Portable and Reproducible Cancer Informatics Workflows for Scalable Data Analysis: An RNA Sequencing Tutorial

Rowan F. Beck, Zelia F. Worman, Gaurav Kaushik, and Brandi N. Davis-Dusenbery

Abstract

The continued decrease in sequencing costs has led to an abundance of high-throughput data representing an increasing diversity of experimental conditions. These changes have been coupled with the adoption of cloud technologies and interoperability standards to share and analyze large primary and secondary data files. While 10 years ago analysis of hundreds or thousands of genomics samples was only practical at institutions with large local computational resources, these experiments can now be routinely performed by anyone with access to the Internet.

In this tutorial, we use the Seven Bridges Cancer Genomics Cloud (CGC) to analyze RNA sequencing data from the NIH Cancer Research Data Commons (CRDC). This tutorial demonstrates how to bring a new computational algorithm to the platform, combine it with an existing workflow, and execute an analysis on the cloud. We highlight best practices for designing command line tools, Docker containers, and CWL descriptions to enable massively parallelized and reproducible biomedical computation with cloud resources. The CGC's support for diverse analysis techniques and user-friendly interface simplifies the complex process of handling large datasets while promoting collaboration across disciplines.

Key words Cloud, Bioinformatics, Cancer informatics, Workflows, CWL, TCGA, AWS, Docker, Reproducibility, Software design

1 Introduction

The Seven Bridges Cancer Genomics Cloud (CGC) powered by Velsera is part of the National Cancer Institute (NCI) Cancer Research Data Commons (CRDC), which was created to accelerate and simplify use of petabyte-scale clinical, imaging, and multiomics data [1]. The CGC provides a secure, scalable, and reproducible environment to perform computational analysis in the cloud [2, 3]. Further, the use of interoperability standards enables connectivity of this resource to NCI multiple data nodes such as the

Alexander Krasnitz and Pascal Belleau (eds.), *Cancer Bioinformatics*, Methods in Molecular Biology, vol. 2932, https://doi.org/10.1007/978-1-0716-4566-6_2,

Genomics Data Commons (GDC) which house data from more than 130,000 participants as of 2023 [4]. The CGC enables data discovery through rich visual interfaces as well as interactive data analysis using popular tools like Python, R Studio, SAS, and Galaxy. Additionally, the CGC enables the automation of large-scale, reproducible analysis using software containers and workflow languages like Nextflow (NF) [5], Workflow Description Language (WDL) [6], and Common Workflow Language (CWL) [7]. Most high-throughput analysis techniques including next generation sequencing, proteomics, flow cytometry, imaging, etc. create large and complex primary data files which must be quality controlled, aggregated, and harmonized. The CGC democratizes these steps by providing both visual and programmatic interfaces for describing analytic workflows. Further, central management of utilities like job scheduling and orchestration, cost monitoring, and data security enables researchers to focus on addressing specific analytic questions rather than building and operating computational infrastructure.

Computational reproducibility remains a significant challenge when replicating studies or performing large-scale, collaborative cancer genomics [8–10]. Variations in software versions or parameters introduce errors and artifacts when attempting to compare analyses from various sources or when working in large collaborations or consortia.

In order to enable and simplify computational reproducibility, the CGC leverages open-source and community-driven technologies for supporting reproducibility with complementary software models that enable researchers to (1) replicate analyses performed previously, (2) readily analyze large volumes of data with identical workflows, and (3) track each step, input, parameter, and output of an analysis automatically.

One such technology is software containers, i.e., operating system-level virtualized environments with a complete filesystem and a unique set of resources and permissions. The Cancer Genomics Cloud specifically supports Docker [11], an implementation of software containers that is operable on all major operating systems. The only external dependency for "running" a Docker container is that the Docker daemon is installed. Because containers are isolated environments, they can be used as portable vehicles for software and their dependencies. For example, a Docker container may build upon a Linux distribution and contain a bioinformatics tool and its dependencies.

Docker containers are designed to be small and intended to be easily deployed to enable sharing among data analysts. Software within containers, if deterministic, will run exactly the same regardless of where the container is deployed or by whom. The use of Docker, therefore, solves a major issue in handling software dependencies in execution environments and comparing or replicating

results which may differ because of software versioning. Indeed, over the last 10 years, Docker and other containerization technologies have become widely adopted across industries.

One issue not solved by Docker is *how* to run bioinformatics tools within the container—in other words, the exact command line arguments which will be used for a given analysis. Bioinformatics use cases have inspired the development and adoption of multiple computational workflow languages which allow users to clearly define computational processes. Each of these languages has different benefits and drawbacks including speed of development, level of declaration, and stability. Bioinformaticians can modularize their pipelines to support reproducibility and reuse by utilizing a workflow language based on their needs by understanding the strengths and weaknesses of each language. Here we focus on CWL which is the most mature of frequently used bioinformatics workflow languages [7, 12]. Importantly, CWL is a formally governed specification which underlies a rich ecosystem of orchestrators, visualization engines, debugging tools, and workflow repositories. The robust nature of CWL enables inclusion of computational workflows within the BioCompute Object paradigm which is being explored as a standard for transmission of computational processes to regulatory agencies like the FDA [13, 14].

For an individual tool, the CWL description contains the URL or pointer to a Docker container residing in an online registry and a set of commands which are executable within the container. In addition, CWL defines "ports" or the input objects which can be used to run analysis (e.g., files, parameters, or simple objects that the user can provide to support execution) and the output objects expected from the analysis.

Based on experience with end users, two additional capabilities have been added on the Seven Bridges platform to extend the utility of Docker and CWL. First, because the CWL specification can be extended by plug-ins, we have added features to enable advanced features such as application revision history and on-the-fly optimization of compute resources. Second, the platform records the explicit parameters and files used in every execution, thus allowing researchers to clone prior analyses inclusive of all inputs for replication or application to new data.

The Seven Bridges Cancer Genomics Cloud aims to drive advancements in cancer research by providing an efficient, secure, and scalable platform for computational analysis. Its integration with interoperability standards and connectivity to NCI data nodes enable researchers from diverse institutions to efficiently leverage the power of cloud computing to perform new analysis. Below we focus on the process of bringing a new tool to the CGC using Docker and CWL. We note that these steps are applicable to other Seven Bridges platform deployments. Furthermore, we emphasize that more than 900 (and growing) popular tools and

workflows are immediately available and optimized for use on the platform (https://cgc.sbgenomics.com/public/apps). These tools can be modified and combined following the approach in Subheading 3.5. Typical bioinformatics workflows include primary and secondary processing of data using well-defined workflows, followed by interactive analysis and exploration using scripting languages and visualization techniques. While the platform provides rich capabilities for downstream analysis, a review of these is outside the scope of this chapter, and the reader is encouraged to explore the detailed documentation at https://docs.cancergenomicscloud.org/.

2 Materials

Researchers wishing to access the CGC need only a personal computer with reliable access to the Internet. For use of Docker containers on your personal computer, we recommend that researchers follow the minimal requirements guidelines from Docker (https://docs.docker.com). Public containers for bioinformatics are also available in online repositories, such as DockerHub (hub.docker.com), Quay.io (quay.io), and Dockstore (dockstore.org). All containers for this study were developed using Docker for Mac Version 1.12 on a 2020 MacBook Pro with 4 cores and 16GB RAM (2 cores and 2GB allocated for Docker). These methods and materials are intended for users familiar with basic bash commands and who have a working knowledge of bioinformatics methods and tools.

Academic users may create a free account on the CGC at https://www.cancergenomicscloud.org/ using their institutional email address. While not required for the following method, the CGC enables researchers to access controlled data from popular datasets like The Cancer Genome Atlas (TCGA) [15], the Human Tumor Atlas Network (HTAN) [16], and others. Users wishing to access controlled data must register using their eRA Commons account and have an approved Data Access Request (DAR) (*see* **Note 1**).

3 Methods

RNA sequencing (RNA-seq) is a molecular biology technique that provides a comprehensive and high-throughput method for profiling and quantifying the entire transcriptome of a biological sample, offering insights into gene expression, alternative splicing, and identification of novel transcripts. The CGC is connected to RNAseq data from thousands of tumors. While single cell RNASeq is increasingly common and the platform provides a number of

resources for this type of analysis, below we use common methods for batch analysis as an example as many bioinformaticians are intimately familiar with these tools.

In brief, we will build a workflow that first performs quality control using the popular tool FastQC, followed by quantification of gene expression using HISAT2-StringTie. FastQC is a widely used and highly efficient quality control tool used in bioinformatics to assess the quality of high-throughput sequencing data. It provides a comprehensive analysis of raw sequencing data and offers insights into various parameters that are crucial for downstream analysis [17]. Numerous approaches have been developed to quantify the abundance of RNA transcripts from sequencing data. Here we've selected hierarchical indexing for spliced alignment of transcripts (HISAT2) and StringTie [18]. Together, these methods align reads to a genome, assemble transcripts including novel splice variants, compute the abundance of these transcripts in each sample, and compare experiments to identify differentially expressed genes and transcripts.

The CGC enables researchers to upload private data and link to public open or controlled access data in the cloud. Here we will use open data from the HCC1143 cancer cell line available as part of the Cancer Cell Line Encyclopedia (CCLE) Project [19]. These data are linked and immediately on the CGC, so researchers are able to rapidly follow the provided method which can then be extended to use private data or other public RNAseq files. Further, the steps provided here can be applied to deploy any command line tool and/or modify existing workflows.

3.1 Workflow Design

Deploying new computational methods or tools and incorporating them into an existing workflow on the CGC entails four major steps:

1. A Docker image is created that contains the software and all of its dependencies. The Docker image should be tested to ensure that the software is correctly installed prior to deployment on the CGC platform.
2. Once tested, the Docker image is pushed to the CGC Docker registry using Docker shell commands and a CGC authentication token, as outlined in Subheading 3.3.
3. A private app is created using CWL to allow description of the Docker image.
4. The newly created app is incorporated into an existing public or private workflow.

In this example, we will incorporate our newly deployed quality control tool into a public RNA-seq workflow. This final step allows tools to be recombined to create complex and reproducible workflows. However, it should be noted that many software tools are perfectly functional as so-called "stand-alone" tools.

3.2 Creating Docker Containers and Testing Tools in Them

A major benefit to Docker containers is their portability; a single container running the Ubuntu Linux distribution can be smaller than 200 MB. This allows for easy and rapid sharing between researchers and platforms. We provide the following guidelines to maintain these benefits:

1. Use Dockerfiles to build your Docker containers (*see* **Note 2**).
2. Avoid or clearly document dependencies on cloud-specific services (e.g., AWS SageMaker).
3. Package each tool in your workflow as a separate container (*see* **Note 3**).
   (a) For example, use a unique container for FastQC and its dependencies and a unique container for any additional tools.
   (b) For Linux tools (e.g., cut, gzip, tar, grep), we recommend using a standard Ubuntu container (`ubuntu:latest`) or a container that builds from it (*see* **Note 4**).
4. In the Dockerfile, explicitly set the working directory as "`/`."
   (a) When we later describe how to execute commands within the container, all arguments and file paths must be relative to "`/`," so it's good practice to start thinking that way early.
5. In the Dockerfile, set the command option as "`/bin/bash`."

With these considerations in mind, we will now start by building a Docker container with FastQC which is available at https://www.bioinformatics.babrahmac.uk/projects/fastqc/fastqc_v0.11.8.zip.

Using the command line, create a file called "Dockerfile" in the top directory with the following content; each section of which is described below:

```
FROM ubuntu:16.04
MAINTAINER "YourFirstName YourLastName" <email@institution.
io>
# Update and install necessary tools
RUN apt-get update && apt-get install -y build-essential
zlib1g-dev \
libgsl0-dev wget unzip
RUN apt-get -y install software-properties-common
RUN apt-get update
RUN apt-get -y install openjdk-8-jdk openjdk-8-jre && apt-get
clean
WORKDIR /opt

# Add FastQC to container and give proper permissions
RUN wget https://www.bioinformatics.babraham.ac.uk/projects/
```

```
fastqc/fastqc_v0.11.8.zip
 RUN unzip fastqc_v0.11.8.zip && rm fastqc_v0.11.8.zip
 RUN chmod 755 /opt/FastQC/fastqc
 RUN ln -s /opt/FastQC/fastqc /usr/local/bin/fastqc

 COPY Dockerfile /opt/
```

This Dockerfile can then be built using the following command:

```
$ docker build -t <repo/image:tag> </path/to/Dockerfile>
```

The first line of the Dockerfile specifies "`ubuntu:16.04`" as the base image (using the `FROM` command; *see* **Note 5**). The "`MAINTAINER`" or contact information for the Dockerfile author can also be specified. The working directory is defined in the third line (`WORKDIR /`).

The second block of commands (`RUN apt-get...`) installs the dependencies for FastQC, which are defined by the tool authors in their documentation [5]. When using "`apt-get`," include the "`-y`" or "`--yes`" option to confirm all actions ahead of time, preventing the need for intervention during install.

Finally, we're changing into "/opt/" as the working directory, downloading and installing the FastQC software and copying the Dockerfile used to build the image into "/opt/" as well to store a record of how the image was created. In the third set of actions (`RUN mkdir...`), we create a directory for the FastQC executables in /opt/; copy the local directory to this directory within the container (*see* **Note 6**). To grant permissions to files in /opt/FastQC, the `chmod 755 /opt/FastQC is used,` and add the directory to $PATH in order to invoke it from the working directory (*see* **Note 7**). Finally, we set the command to "`/bin/bash`" (*see* **Note 8**).

Next, run the container in interactive mode to verify that the installation has occurred properly:

```
$ docker run -ti <repo/image:tag>
```

If changes need to be made to a container, we recommend recording each change within the original Dockerfile as a comment and then rebuilding the container (*see* **Note 9**). In this way, your Dockerfile will capture all dependencies to streamline reproducibility and repeatability. **Note 10** provides an example Dockerfile for another RNASeq tool, RSEM, which requires multiple dependencies.

3.3 Deploying Containers on the Seven Bridges Cancer Genomics Cloud

Once the container is tested, you can "push" it to an online registry. The Cancer Genomics Cloud can pull containers from any public registry, including DockerHub or Quay.io. However, we recommend pushing containers to the CGC image registry for increased reliability and reduced latency when pulling containers for computation.

A user will need to verify their CGC authentication token, which encodes that specific user's CGC credentials and uniquely identifies them on the CGC. To obtain this token, a user must log in to the CGC (cgc.sbgenomics.com) then click "Developer" in the top navigation bar. Next, select "Authentication token." Click "Generate Token" to generate an authentication token for the first time. The authentication token will be displayed in the input field, and information will be provided on when the token will expire (*see* Fig. 1).

To push the container to the CGC image registry, tag the container with the appropriate repo name:

```
 docker tag <image_id> cgc-images.sbgenomics.com/<cgc_user-
name>/<image>:<tag> docker login cgc-images.sbgenomics.com
 <cgc_username>
 <cgc_auth_token>
 docker push cgc-images.sbgenomics.com/<cgc_username>/<im-
age>:<tag>
```

Often, bioinformatics tools are very comprehensive which makes it difficult to capture all of its command line arguments or utilities in a single description. Attempting to do so may require very complex wrappers, which can increase the time for debugging and testing. To prevent this issue, wrap your analysis and reuse prior wrappers as your analyses change. Everything is versioned on the Seven Bridges Cancer Genomics Cloud which makes it easy to keep track of each application and its associated files. In addition, researchers have full visibility to when each task was executed and by whom.

3.4 Describing Tools Using the Cancer Genomics Cloud

On the Cancer Genomics Cloud, tools are considered single executables or runnables, which consist of a set of command line expressions run within a Docker container. Workflows are chains of tools, in which upstream files are passed downstream until a final result is achieved. To begin using the visual interface to write the CWL describing a tool, navigate to Apps, followed by "Command Line Tool" as shown in Fig. 2.

On the CGC, tools have the following five properties:

1. *Docker Container*: Defines the Docker container that will be used via the `<repo/image:tag>` (e.g., "`ubuntu:latest`" or "`cgc-images.sbgenomics.com/gauravcgc/rsem:1.2.31`")

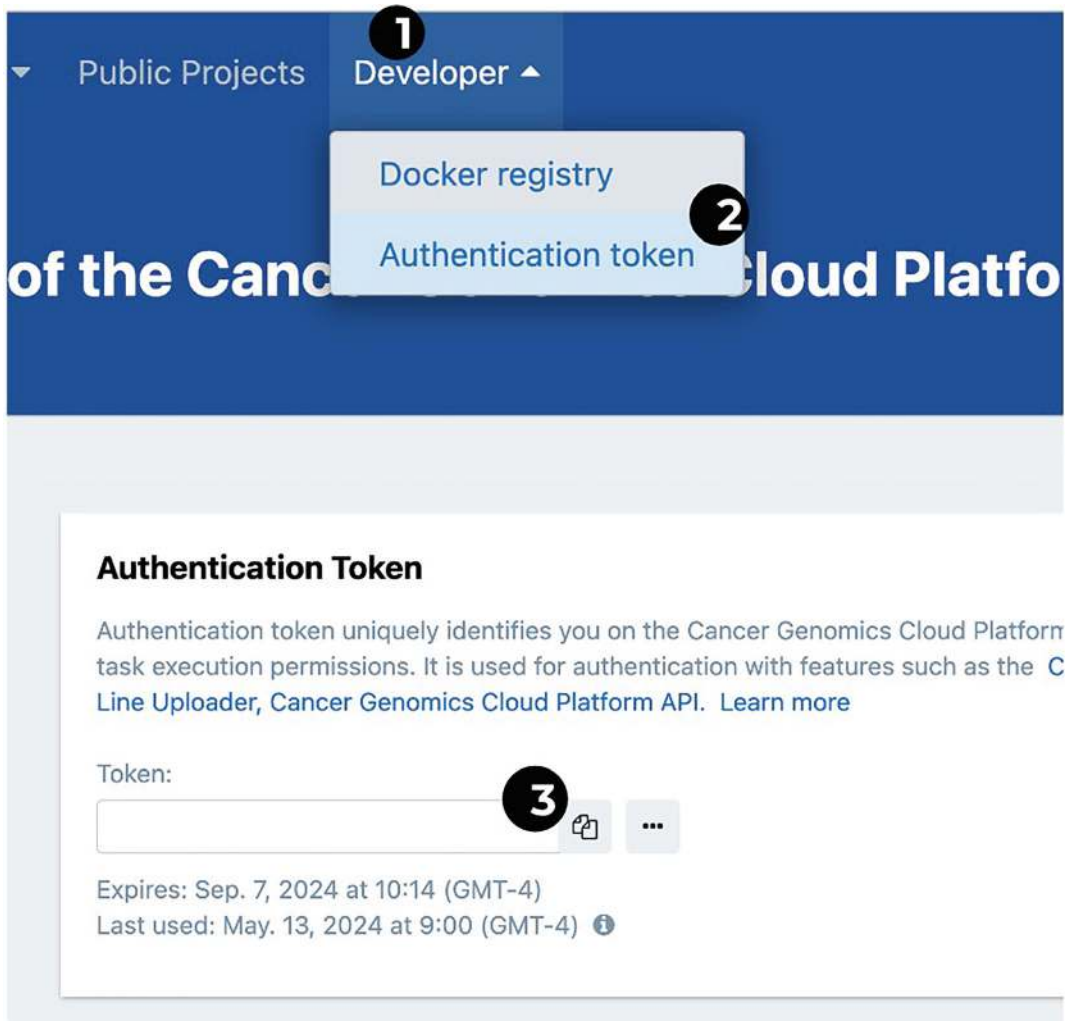


Fig. 1 Accessing the CGC Authentication Token. The authentication token encodes your Seven Bridges CGC Platform credentials and uniquely identifies you on the Seven Bridges Platform. This token can be used with a number of features of the Seven Bridges Platform instead of a login

2. *Base Commands*: The starting point on which the arguments and ports are layered
3. *Arguments*: Hard coded inputs which are not user configurable at runtime
4. *Input Ports*: Describe data objects that can be passed to the tool during execution
5. *Output Ports*: Describe data objects that will be saved from an execution, most commonly an output file or array of files

We'll walk through describing each of these properties below.

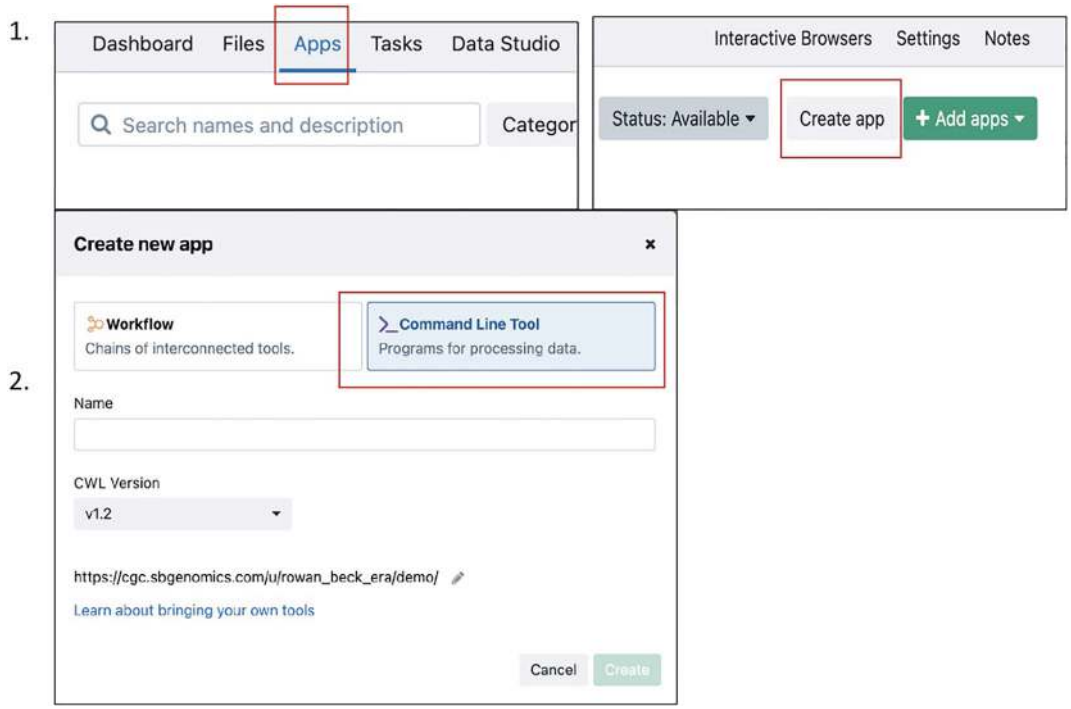


Fig. 2 Accessing the CGC Tool Editor. The tool editor is a visual editor for creating and editing CWL tools. (1) To access the tool editor, first select or create a project, and then navigate to the "Apps" tab along the top of the screen. Click the "Create app" button. (2) To create a new app, select "Command Line Editor" and give your app a name

Wrapping *FastQC in CWL*

The FastQC tool has several input and output parameter settings. Rather than go through every single one, we will add only a few of the parameters and define required Base Command, Arguments, Inputs, and Outputs as shown in Fig. 3.

The desired command line we are aiming to build is as follows:

```
fastqc --noextract --outdir . /path/to/input-1.ext /path/to/
input-2.ext
```

In this example, we are defining the base command as "`fastqc`" with an additional argument of "`--noextract`." This argument allows us to run FastQC without having to uncompress the output file after creating it. To create this command line, do the following:

1. Set the Docker image URL, which can be used with a "`docker pull`" command to pull an image to your local machine.
2. Set the Base Command as "`fastqc`."

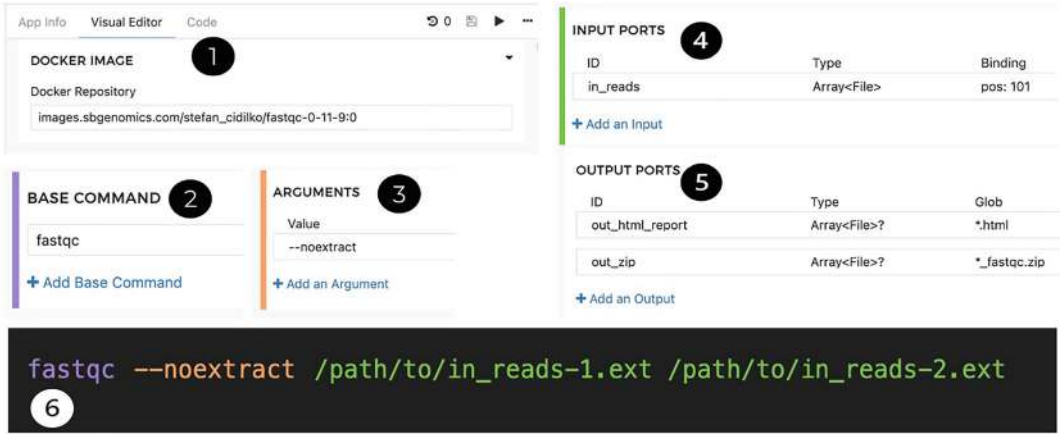


Fig. 3 Describing the FastQC tool with the CGC Tool Editor. (1) Each tool must have a Docker container within which the command line executions are run. (2) The Base Command field is for a set of core commands for the tool, including the necessary subcommand. (3) The argument command is where any arguments to the base command can be added or modified. (4) The inputs tab is where Input Ports are added and modified. The Input Port menu allows the user to set the parameters and behavior for an object that can be passed to the tool (e.g., a File). (5) The Outputs menu allows the user to specify Output Ports, which describe objects produced by the tool which the user wishes to capture when the execution is completed. (6) The Resulting command line of the app

3. Click the "Add an Argument" button. Configure new argument as following:
   (a) *Use command line binding*: YES
   (b) *Prefix*: (leave empty)
   (c) *Value*: --noextract
   (d) *Separate value and prefix*: YES or NO
   (e) *Position*: 0
      (i) *Note*: Position is set to 0 as we want this argument to be placed first after the end of the base command.

   For this argument, we have left the prefix empty as it is a Boolean type argument and it does not take an actual value. Since we want this flag to be added to the generated command line under all circumstances, we have added the "`--noextract`" argument directly as a value.

   Arguments differ from "inputs," as arguments are not directly open to the user manipulation during task creation. You may utilize arguments to lock down fundamental aspects of your tool execution. For instance, in this case we want to force FastQC to adopt a certain behavior: never create result directories, and instead rather keep results in compressed format to reduce file cluttering in the project.

   The settings we can define for an argument are as follows:

- *Use command line binding*: This Yes/No switch determines whether the specified argument should be included in the generated command line.
- *Prefix*: Prefix field allows tool wrapper to define the argument prefix.
- *Value*: This field allows tool wrapper to define the value that will come after the prefix. Tool wrappers may fill this field with fixed values or create a JavaScript expression using "</>" button for dynamic generation of the argument value. You may visit the Seven Bridges Knowledge Center (https://docs.sevenbridges.com/) to learn more about dynamic expressions generated with JavaScript.
- *Separate value and prefix*: This Yes/No switch determines whether "prefix" and "value" should be separated with a space in the command line.
- *Position*: This field determines the position of the argument within the command line. You may set this as 0 to position the argument at the first place after the base command or a very large number, such as 99, to position the argument at the very end of the command line.

4. Add an additional argument by selecting the "+ Add an Argument" button and configure as following:
   (a) *Use command line binding*: YES
   (b) *Prefix*: --outdir
   (c) *Value*:
   (d) *Separate value and prefix*: YES
   (e) *Position*: 1

   This time we have set a prefix and a value, since this is an argument that takes a value.
5. Add an INPUT PORT by selecting the "Add an Input" button and configure as following:
   (a) *Required*: YES
   (b) *ID*: input_files
   (c) *Type*: array
   (d) *Items Type*: File
   (e) *Include in the command line*: Yes
   (f) *Value Transform*:
   (g) *Prefix*:
   (h) *Position*: 100 (or any large number; we want this to be placed at the end of the command line)
   (i) *Separate value and prefix*: Yes

(j) *Item Separator*: repeat
(k) *Stage Input*: --none--
(l) *shellQuote*:
(m) *Load Content*: No
(n) *Add secondary file*: No Secondary Files defined.
(o) *Label*: Input FASTQ Files
(p) *Description*:
(q) *Alternative Prefix*:
(r) *Category*:
(s) *File type(s)*: FASTQ, FASTQ.GZ

You may find more information about how to define the input settings at the Seven Bridges Knowledge Center (https://docs.sevenbridges.com/).

6. (Optional) Add another INPUT PORT by selecting "+ *Add an Input*," and set the following fields in the inspector panel:
   (a) *Required*: NO
   (b) *ID*: threads
   (c) *Type*: int
   (d) *Include in the command line*: Yes
   (e) *Value Transform*:
   (f) *Prefix*: --threads
   (g) *Position*: 3
   (h) *Separate value and prefix*: Yes
   (i) *Label*: Number of Threads
   (j) *Description*: Specifies the number of files which can be processed simultaneously.
   (k) *Alternative Prefix*:
   (l) *Category*:
   (m) *Tool Defaults*: 1

Now that we have specified all inputs and arguments defined in the target command line structure shown at the beginning of this section, we may start defining the outputs that should be collected after the execution of the FastQC. When run with a "--noextract" argument, FastQC creates two files per sample:

- A ZIP file that contains FastQC report, graphics, and summary files
- An HTML report showing the FastQC results

As such, we must create output ports for both types of outputs.

7. Add an OUTPUT PORT by selecting the "Add an Output" button, and set the following fields in the inspector panel:
   (a) *Required*: No
   (b) *ID*: report_zip
   (c) *Type*: array
   (d) *Items Type*: File
   (e) *Glob*: *.zip
   (f) *Inherit*:
   (g) *Output eval*:
   (h) *Load content*: NO
   (i) *No Secondary Files defined Label*: Report zip
   (j) *Description*:
   (k) *File type(s)*: ZIP

   You may learn more about the output port settings at the Seven Bridges Knowledge Center (https://docs.sevenbridges.com/). Please note that adding an output port does not cause a change in the sample command line generated at the bottom of the screen.

   Now you can add our second output port for the HTML reports that you can display on the Platform.
8. Add an OUTPUT PORT by selecting the "+ Add an Output" button, and set the following fields in the inspector panel:
   (a) *Required*: No
   (b) *ID*: report_html
   (c) *Type*: array
   (d) *Items Type*: File
   (e) *Glob*: *.html
   (f) *Inherit*:
   (g) *Output eval*:
   (h) *Load content*: NO
   (i) *No Secondary Files defined*
   (j) *Label*: Report HTML
   (k) *Description*:
   (l) *File type(s)*: HTML
9. Save your tool to the Seven Bridges Platform by creating a new revision. Click the floppy disc icon located at the top right corner of the screen. You may add a revision note, and then select "Save."

   You will now notice that a new tab appears that displays the app version that you just pushed to the Platform project.

CWL provides a variety of features to enhance the usability and performance of described tools such as conditional execution, batching, and dynamic expressions (please *see* **Notes 11–15** for additional considerations and suggestions when optimizing tool descriptions). While we've provided instructions for describing tools using the visual interface on Seven Bridges platforms, it is worth emphasizing that CWL can be entirely written by hand and any valid CWL description can be imported into the platform. The CWL website (commonwl.org) provides numerous tutorials and resources for users getting started with CWL.

3.5 Chaining Tools into Workflows

Workflows on the Cancer Genomics Cloud are chains of tools where input data is passed from tool to tool in an orchestrated manner. In addition, workflows expand upon tools in specific ways:

1. Explicit values can be set for Input Ports (e.g., "threads" in the example above).
2. Workflows can be configured to "scatter" an array of parameters of values over an Input Port, creating a task for each index in the array.
3. Workflows can be "batched" based on metadata properties.

The Seven Bridges Platform includes a workflow editor for viewing and customizing workflows. For example, the addition or subtraction of tools from a workflow may be desirable to meet the users' analysis needs.

Here we will walk through the steps required to modify the public *HISAT2-StringTie Workflow* to include the FastQC tool which we just wrapped in CWL. These steps could also be applied to combine other public or private tools into a reproducible workflow.

Copy HISAT2-StringTie Workflow from Public Apps Gallery

The HISAT2-StringTie Workflow can be used to perform a gene abundance estimation of RNA-Seq data (i.e., quantification) for a unified set of genes common for all samples in an analysis. This workflow is based on the Nature protocol paper [18] (with the absence of the last step, Ballgown, which carries out testing for differential expression).

To copy an existing version of this workflow into your project space, click the "Apps" tab along the top banner of your project and then "Add apps," and lastly select "Public Apps."

Next, type "*HISAT2*" into the search box, and click "*Copy*" on the "*HISAT2-StringTie Workflow*" workflow panel. Select your project from the dropdown menu, or create a new project. You should now see a copy of the "*HISAT2-StringTie Workflow*" app in the project you created.

Chaining FastQC into the HISAT2-StringTie Workflow

Starting from a project containing the public apps version of the HISAT2-StringTie Workflow, first navigate to the "Apps" tab and click the "…" next to HISAT2-StringTie Workflow, and then select "Edit."

The platform will prompt you with a warning that if you make changes to this copy of the HISAT2-StringTie Workflow, you will stop getting notifications about updates to the original app. Click "Edit" again to open the workflow editor with the HISAT2-StringTie workflow featured in the workflow editor dashboard. This workflow consists of several independent apps each displayed as a colored circle and corresponding input or output files displayed as gray circles. There are nodes on the perimeter of each item in the Workflow Editor. These represent the tool's ports, which are used to enable data to flow in and out of the App. Nodes on the left of the circle represent input ports, whereas the ones on the right indicate output ports. Clicking on a port and dragging will reveal a smart connector which is used to chain tools together into workflows as shown in Fig. 4.

To connect our FastQC app, first navigate to the left side of the editor. Click the tab labeled "My Projects" and type "FastQC" to search for your app. Once you've identified your FastQC app, select it with your arrow, and then drag and drop the app onto the workflow editor dashboard.

Now, the "FastQC" app is available to be connected to the rest of the workflow. We must first connect the FASTQ files (labeled "Reads") as an input to the FastQC app. To do so, click the output port of the object labeled "Reads," and drag it to the input node on the left side of the "FastQC" app.

FastQC is now connected to our workflow; however, we need to specify which outputs we would like to capture. To do this, click the output node of the FastQC app and drag it to the dashboard. Finally, clicking the grid icon in the lower right-hand corner of the editor will realign the diagram.

Before executing the workflow, you need to save it by clicking the save icon in the upper right-hand corner of the editor. This will reveal a dialogue box for comments describing the changes made to the workflow. Here we'll type "Added FastQC to the workflow." Now, the workflow is complete and ready for analysis.

3.6 Running the Workflow

To run the workflow, we must first create a new task within the project space containing the modified HISAT2-StringTie workflow. To create a new task, navigate to the "Apps" tab within the project, and then select "Run" next to the workflow. A new task page will be created featuring two main tabs, "Task Inputs" and "Execution Settings."

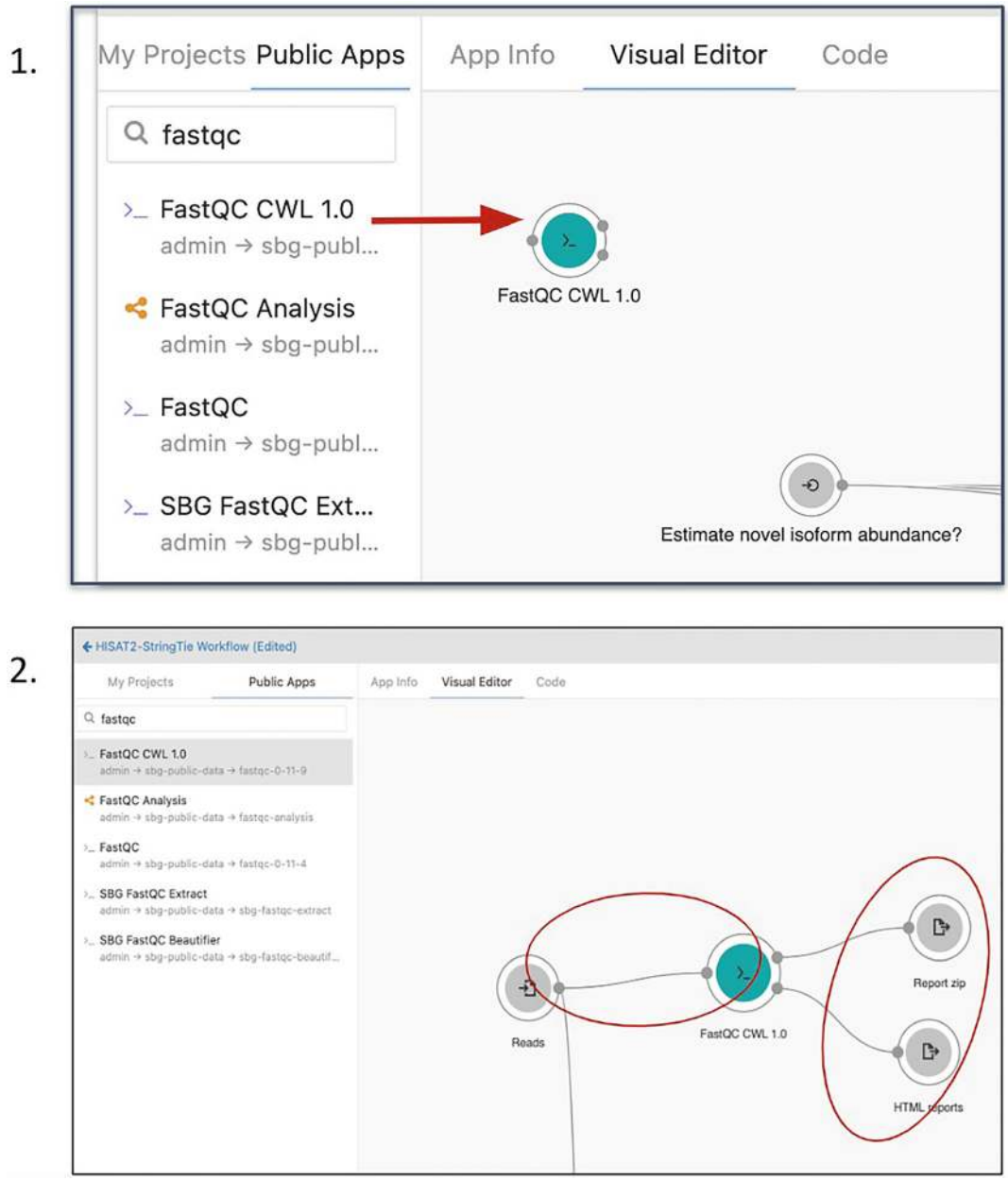


Fig. 4 Assembly of the FastQC-HISAT2-Stringtie workflow. Workflows are described as chains of tools, in which data objects are passed downstream. Data objects (inputs/outputs) can be passed between tools by connecting the appropriate ports or can be redirected to Output Ports, so they are captured and saved once execution is completed

Workflows are designed such that certain ports can be "locked" with or without values (if not required), thereby preventing a user of the workflow from modifying this parameter. (1) To connect an additional app to the workflow, select the desired app from the list on the left side of the screen, and then click and drag the app into the workspace editor. (2) Connect the input port of FastQC to the "reads" output port. Click and drag both output ports of FastQC to create and save output files

Task Inputs

The task inputs tab can be divided into two sections with which the user can interact: Inputs and App Settings. The "Inputs" section for this workflow specifies that a user must provide reads, a reference annotation file, and a reference or index file. These files can be selected from the current project, other projects a user is a member of, or public files. For this example, we will use input files from the Public Files repository. To select reads, click the "Select file(s)" icon for Reads input. Navigate to the "Public Files" tab along the top. Next, click the "Tags" dropdown menu and select "RNA-seq." From the list of sorted files, select the two files ending in 1Mreads. Click "Save Selection" to assign these files to your task, and then click "Copy" when prompted. You will be returned to the draft task page. Next, click the "Select file(s)" icon for the Reference annotation file, navigate to the "Public Files" tab, then search "Homo_sapiens.GRCh38.84.gtf," and save your selection. Finally, click "Select file(s)" for the "References or Index files" input field, navigate to "Public Files" and search "grch38_tran.tar.gz," select the file, and save the selection.

Under the "App Settings" section, set "Estimate novel isoform abundance?" as False. Click "Show all" under the Apps Settings tab to see all other parameters as shown in Fig. 5. These parameters are not set as "editable" by the developer of this workflow.

Execution Settings

Under Execution Settings, "*Spot Instances*" can be set to "*On*." Spot instances are cost-effective ways to run tasks. Our in-house analysis has shown that tasks on spot instances cost up to 90% less than tasks that run on on-demand instances. This comes at a low risk with an interruption rate of ~1%, in which case the platform will automatically restart the interrupted process(es) using an on-demand instance.

Memoization provides a mechanism for the automatic reuse of precomputed outputs inside your project. There are some important considerations to be aware of before using Memoization (https://docs.sevenbridges.com/docs/about-memoization).

Additional options such as "Instance type" and "Parallelization" can be accessed and specified in this section.

Click "*Run*" on the top-right to kick off the task. Task details page shows the task is in running state. Processing time is ~7 min and costs $0.17. After the task has finished, click the task link. This brings up the task details page, which contains a complete record of the executed app, used inputs, app settings, and the produced outputs.

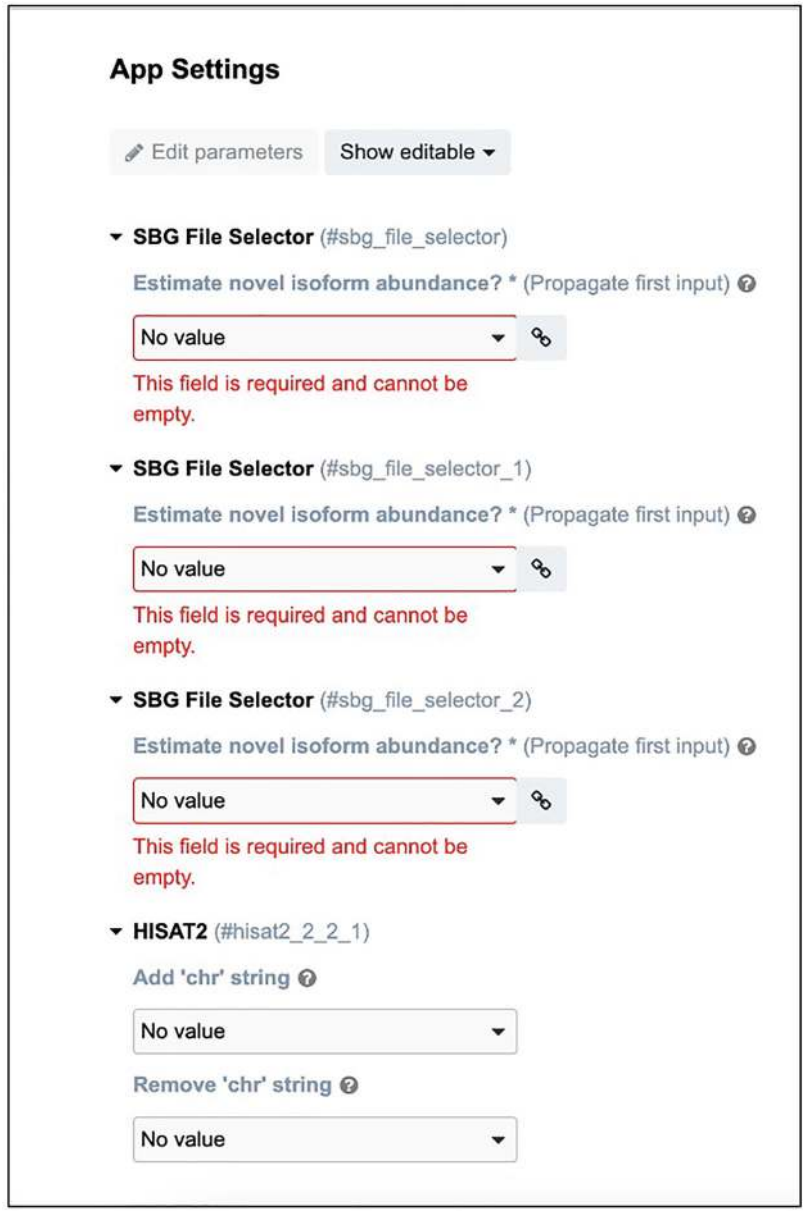


Fig. 5 Setting parameters in the FastQC-HISAT2-Stringtie workflow. Once inputs are provided to the app on the task run page, we can take a look at "App Settings." Only the setting, "*Estimate novel isoform abundance?*", can be altered in this workflow. Click "Show all" under the Apps Settings tab to see all other parameters. These parameters are not set as "editable" by the developer of this workflow

Viewing Outputs

StringTie provides a number of output files, among them a gene abundance file, which is a TAB file containing FPKM, TPM, and Coverage values for each gene as shown in Fig. 6. Two files are also produced containing transcript and gene expression values in the

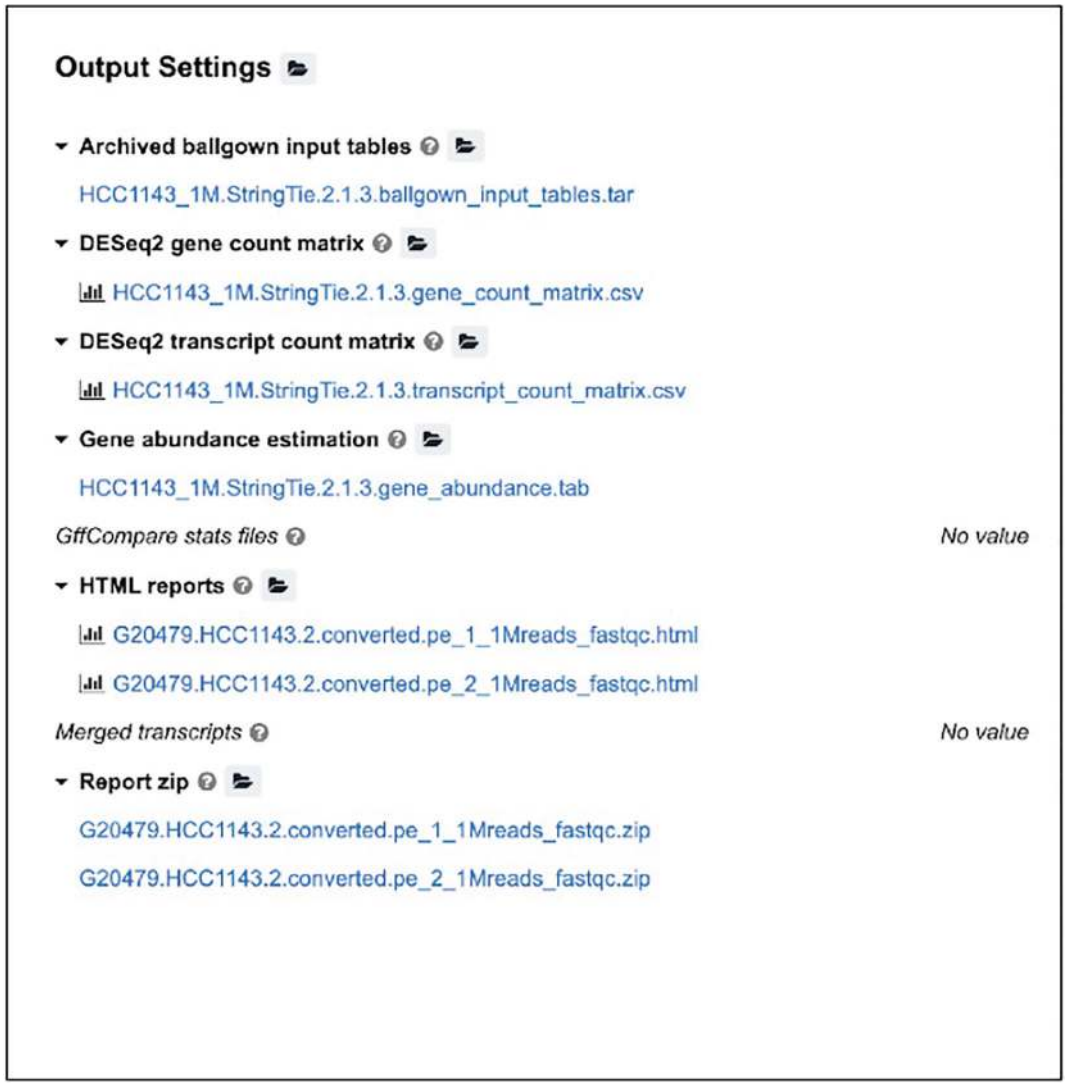


Fig. 6 Viewing task results. Execution details including computational runtime and price are printed below the task name. Output files from the task run are linked to directly from the task page

DESeq2 input format. The count matrix files are in .CSV format and can be previewed on the Platform and dynamically sorted to display genes with the highest read count values.

Viewing Stats and Logs

The Seven Bridges Platform gives you the tools you need to view statistics and logs related to your analysis. This can come in handy when you are diagnosing failed tasks. Click on "View stats & logs" at the top-right of the task details page. This brings up the "Task stats" page. You can find a detailed explanation of all its content in the Seven Bridges Knowledge Center (https://docs.sevenbridges.com/docs/view-task-stats). A detailed view of the selected area is

shown beneath the timeline. Most often, you will see several horizontal bars on the timeline which correspond to different apps and their jobs. Each app is executed in several steps, which are called jobs. To view these jobs individually, click the bar representing the app. The jobs appear as horizontal green bars underneath the app. Parallel jobs are aligned vertically, to indicate that they executed simultaneously.

If we select the gray box labeled HISAT2, you now see the pinned details for this app. Click "View Logs" under "HISAT2" to get more details about how this app was executed. The Task logs page gives us access to all the log files of the app and any output files generated. Logs we typically want to inspect are "cmd.log," which contains the executed command line, "sbg.worker.log," which provides details on Platform operations to coordinate the task execution, and "job.err.log," which contains all error/warning messages produced by the command (stderr).

After checking the logs, if you would like to investigate resource usage on the computational instances used throughout the execution, you can go close the task logs dialogue by clicking on the "X" button on top right corner, and click "Instance Metrics" button shown on the top right-hand side of the "Tasks stats & logs" page. The Platform lets you access instance metrics information for all instances used in task execution.

The following information is available during task execution and for 15 days after the task has been executed:

- Instance type, purchasing type and status
- Instance configuration, i.e., available vCPUs, Memory, Disk space
- CPU usage
- Disk usage
- Memory usage
- Load average
- I/O activity
- Swap activity

All files generated from the execution can be accessed from the task page as well as from the files tab. Note that selecting a file will link back to the execution that created it which helps to support traceability and reproducibility. We recommend running the workflow with several test files before setting up large batch tasks. This allows you to identify areas for cost optimization and avoid errors and expected outcomes at scale. Typically, researchers will perform interactive analysis using visualization and scripting tools once raw data are processed. The CGC has a number of tutorials and videos demonstrating these capabilities which can be found on the CGC home page at www.cancergenomicscloud.org.

3.7 Conclusions

The Seven Bridges Cancer Genomics Cloud plays a pivotal role within the NCI Cancer Research Data Commons and serves as an efficient, secure, and scalable computational analysis platform. Its adherence to interoperability standards and seamless connection to NCI data nodes allows users to easily integrate software and technologies to analyze publicly available data, as demonstrated in this guide. The CGC's facilitation of diverse analysis techniques, user-friendly interfaces, and automation capabilities not only streamlines the intricate process of managing extensive datasets but also empowers researchers to focus on the core analytical aspects of their investigations.

4 Notes

1. The CGC uses standards developed by the Global Alliance for Genomics and Health (GA4GH) to enable approved researchers to access petabytes of data. A listing of connected datasets can be found at https://www.cancergenomicscloud.org/datasets. A video tutorial for submitting dbGaP requests can be found at https://www.youtube.com/watch?v=m0xp_cCO7kA, while guidance for troubleshooting data access is available at https://docs.cancergenomicscloud.org/docs/dbgap-controlled-data-access.
2. Dockerfiles should be used as a precise description of how to rebuild the container used for an analysis. Since Dockerfiles are simple text descriptions of a container using a domain-specific language, they are extremely portable. By the time you're done building a container with a Dockerfile, you have a lightweight way to enable others to reproduce your execution environments. You may want to interactively build the container and then record the steps you take in a Dockerfile—but to ensure reproducibility, the final container you use should actually be built from the Dockerfile.
3. An important aspect of Docker containers is that a group of containers in the same volume (e.g., registry, hard drive) can share layers. For example, a container that is built from another container but adds additional tools doesn't duplicate the data from the previous container. This behavior alleviates any data burden of having many function-specific containers. For this reason, we encourage researchers to employ many lightweight containers for easy sharing and deployment, as opposed to a few containers with many tools inside.
4. For simple command line tools, we recommend using standard Linux distribution containers. For example, let us examine a simple tool which saves the first 1000 lines of a FASTQ file as a new file. We can use the "head" tool to trim the top "n" lines of

any file. Since `head` is a Linux command line tool, we can use an Ubuntu container as a bigger one is not needed. To save the first 1000 lines of a FASTQ in a new file, use `head` to print the first 1000 lines of the input FASTQ file, and then redirect stdout to a new file:

```
head -1000 [input_file] > [output_file]
```

This tool can then be reused on any FASTQ file to generate a plethora of outputs. Note that this tool is particularly useful for sampling FASTQs, which can then be used for rapidly testing tools or workflows which are computationally intensive. When the goal is to see if your tool runs properly, we recommend that you use sampled FASTQs to iterate more rapidly.

5. Dockerfiles use the "FROM" command to use another image as the "base" for the new image being built. We recommend that you build from trusted, standard containers (e.g., `ubuntu:latest`) instead of a third-party container. If you choose to rebuild your container from this Dockerfile in the future and the base image has been modified, the rebuilt container will inherit any changes within it. This may lead to errors or dependency issues if the base image has changed dramatically.
6. If you don't intend to use the software outside the container, you can use commands such as wget, curl, or other downloading software to directly download software into a container during the build.
7. Some software will automatically add itself to $PATH, and thus these steps may not be required.
8. Docker containers can be run to execute a command directly (using the CMD field). However, containers intended for use on the CGC should be built such that they can be run in "interactive mode" (e.g., `docker run -it <repo/image:tag>`) and a set of specified commands can be executed within them. On the Cancer Genomics Cloud, all commands are executed from the working directory ("/") by default.
9. The "`docker build`" command will begin to build the container layers starting from the beginning of the Dockerfile (FROM) until the end (CMD). The layers pertaining to each code block are cached during the build. If a change is made somewhere in the Dockerfile, all the layers pertaining to code blocks *above* the change will be loaded, and all layers built from the command *below* the change will be overwritten. It is recommended that any software-based changes that need to be made into a container are done by modifying the original Dockerfile and then rebuilding. However, there may be cases where you

wish to save an analysis or analysis output in a container for your own use (not for production). In such cases, you can commit changes made in a container back to the image with the following command: `docker commit <container_id> <repo/image:tag>`.

10. Here is an example Dockerfile for another RNAseq tool, RSEM 1.23.1.

```
FROM ubuntu:latest
MAINTAINER "YourFirstName YourLastName" <email@institution.
io>
WORKDIR /
# Update and install necessary tools
RUN apt-get update -y
RUN apt-get install -y \
gcc \
g++ \
libdb5.1 \
libdb5.1-dev \
make \
cmake \
libboost-dev \
libboost-thread-dev \
libboost-system-dev \
zlib1g-dev \
ncurses-dev \
libxml2-dev \
libxslt-dev \
build-essential \
python \
python-pip \
python-dev \
git \
apt-utils \
vim \
wget \
perl \
perl-base \
r-base \
r-base-core \
r-base-dev
# Get RSEM-1.2.31 and install RSEM and EBSeq
RUN wget -P opt --verbose --tries=5 https://github.com/
deweylab/RSEM/archive/v1.2.31.tar.gz
RUN tar xzf opt/v1.2.31.tar.gz
RUN cd RSEM-1.2.31/ && make && make install && make ebseq
# Install bowtie
```

```
RUN apt-get install bowtie -y
# Open container with bash terminal
CMD ["/bin/bash"]
```

11. There are a few important considerations when designing tools. On the CGC, you can create a "batch" of tasks based on the metadata properties of a set of input files. For example, let's say you have a tool which sorts a single BAM file and outputs the sorted BAM. You can create a workflow from this tool which can instead take an array of bam files and will create a single task for each set of files based on "Sample ID." Since each BAM file presumably has its own unique Sample ID, a single task will be created per BAM with far less user interaction (if done through the GUI) or code required (if done through the API). A major constraint for this feature is that you can only batch on a *single* Input port. This is obvious for BAM files, but when performing a batch of tasks for a set of paired-end FASTQ files, you must pass the FASTQ files as an array of files. If you create an individual port for each paired-end FASTQ (e.g., 1, 2), then you cannot set up a batchable workflow.
12. We can set a default value for an Input Port by taking two steps: (1) do not make the port "required," and (2) create a dynamic expression for the "Value" of this port such that the port has a value if unspecified. For example, if the Input Port type is an integer and we wish to specify "31" as a default value, the dynamic expression can be written as `($self || 31)` or `($job.inputs.<input_id> || 31)`. This expression means that if the user specifies a value for this port ($self), then that is used as the "Value" or, if unspecified, 31 is given as the value. In addition, the default value in the CWL description can be different from the tool's default.
13. Explicit values for each Input Port and Allocated Resources (CPU, Memory) are provided to the application for execution. This is referred to as the "job" object (or "job.json"). From this object, we can incorporate the properties of Input Ports into the dynamic expressions of the tool. For example, suppose that a user can specify the number of threads for a tool with an integer value (ID: threads). In order to pass this to the CPU requirements field, we can set the value for CPU as the following dynamic expression: `$job.inputs.threads`.
14. Consider how a tool will operate when it runs at scale and how you will manage the output data. Tools such as kallisto index take an input which generates the output filename or its prefix. When batching tasks, you cannot then specify an individual name per file. Rather, only an individual value can be given. On the CGC, this will produce n number of files with the same

name, with the addition of a prefix ("_#_," where # is the nth copy of a file). This will make file management difficult, as the origin of a file is not readily apparent from its name. Instead, we will solve this issue by automatically passing a unique identifier, such as the input file's "Sample ID," which will scale over dozens, hundreds, or thousands of tasks.

15. Within workflows, if an Output Node is not created from a port, then objects from that port are not saved. It is possible to both pass an output from an upstream tool downstream and create an Output Port to save it. In this way, you can also save "intermediate files" or files which can be passed to downstream tools, by creating Output Nodes for them in addition to connecting them to downstream tools.

Acknowledgments

The Seven Bridges Cancer Research Data Commons Cloud Resource has been funded in whole or in part with Federal funds from the National Cancer Institute, National Institutes of Health, Contract No. HHSN261201400008C, and ID/IQ Agreement No. 17X146 under Contract No. HHSN261201500003I and 75N91019D00024. We would like to acknowledge the contributions of past and present Velsera team members, including but not limited to Jack DiGiovanna, Alison Leaf, Liz Williams, Manisha Ray, Dennis Dean III, Divya Sain, Sai Subramanian, Nikola Mirkovic, Nina Skoko, Ivana Gornik, and Milos Trboljevac.

References

1. Kim E, Davidsen T, Davis-Dusenbery BN, Baumann A, Maggio A, Chen Z, Meerzaman D, Casas-Silva E, Pot D, Pihl T, Otridge J, Shalley E, CRDC Program, Barnholtz-Sloan JS, Kerlavage AR (2024) NCI Cancer Research Data Commons: lessons learned and future state. Cancer Res 84(9): 1404–1409. https://doi.org/10.1158/0008-5472.CAN-23-2730. PMID: 38488510; PMCID: PMC11063686
2. Pot D, Worman Z, Baumann A, Pathak S, Beck R, Beck E, Thayer K et al (2024) NCI Cancer Research Data Commons: cloud-based analytic resources. Cancer Res 84(9): 1396–1403
3. Lau JW, Lehnert E, Sethi A, Malhotra R, Kaushik G, Onder Z, Groves-Kirkby N, Mihajlovic A, DiGiovanna J, Srdic M, Bajcic D, Radenkovic J, Mladenovic V, Krstanovic D, Arsenijevic V, Klisic D, Mitrovic M, Bogicevic I, Kural D, Davis-Dusenbery B (2017) The Cancer Genomics Cloud: collaborative, reproducible, and democratized—a new paradigm in large-scale computational research. Cancer Res 77:e3–e6
4. Wang Z, Davidsen TM, Kuffel GR, Addepalli KD, Bell A, Casas-Silva E, Dingerdissen H et al (2024) NCI Cancer Research Data Commons: resources to share key cancer data. Cancer Res 84(9):1388–1395
5. Di Tommaso P, Chatzou M, Floden EW, Barja PP, Palumbo E, Notredame C (2017) Nextflow enables reproducible computational workflows. Nat Biotechnol 35:316–319
6. Voss K, Auwera GA, Gentry J (2017) Full-stack genomics pipelining with GATK4 + WDL + Cromwell. F1000Res 6:1–4
7. Crusoe MR, Abeln S, Iosup A, Amstutz P, Chilton J, Tijanić N, Ménager H, Soiland-Reyes S, Gavrilović B, Goble C, Community TC (2022) Methods included: standardizing computational reuse and portability with the Common Workflow Language. Commun ACM 65:54–63

8. Alioto TS, Buchhalter I, Derdak S, Hutter B, Eldridge MD, Hovig E, Heisler LE, Beck TA, Simpson JT, Tonon L, Sertier A-S, Patch A-M, Jäger N, Ginsbach P, Drews R, Paramasivam N, Kabbe R, Chotewutmontri S, Diessl N, Previti C, Schmidt S, Brors B, Feuerbach L, Heinold M, Gröbner S, Korshunov A, Tarpey PS, Butler AP, Hinton J, Jones D, Menzies A, Raine K, Shepherd R, Stebbings L, Teague JW, Ribeca P, Giner FC, Beltran S, Raineri E, Dabad M, Heath SC, Gut M, Denroche RE, Harding NJ, Yamaguchi TN, Fujimoto A, Nakagawa H, Quesada V, Valdés-Mas R, Nakken S, Vodák D, Bower L, Lynch AG, Anderson CL, Waddell N, Pearson JV, Grimmond SM, Peto M, Spellman P, He M, Kandoth C, Lee S, Zhang J, Létourneau L, Ma S, Seth S, Torrents D, Xi L, Wheeler DA, López-Otín C, Campo E, Campbell PJ, Boutros PC, Puente XS, Gerhard DS, Pfister SM, McPherson JD, Hudson TJ, Schlesner M, Lichter P, Eils R, Jones DTW, Gut IG (2015) A comprehensive assessment of somatic mutation detection in cancer using whole-genome sequencing. Nat Commun 6:10001
9. Ziemann M, Poulain P, Bora A (2023) The five pillars of computational reproducibility: bioinformatics and beyond. Brief Bioinform 24: bbad375. https://doi.org/10.1093/bib/bbad375
10. Trisovic A, Lau MK, Pasquier T, Crosas M (2022) A large-scale study on research code quality and execution. Sci Data 9:60
11. Merkel D (2014) Docker: lightweight Linux containers for consistent development and deployment. Linux J 2014:2
12. GitHub. common-workflow-language common-workflow-language/common-workflow-language. https://github.com/common-workflow-language/common-workflow-language. Accessed 22 Feb 2015
13. Xiao N, Koc S, Roberson D, Brooks P, Ray M, Dean D (2020) BCO App: tools for generating BioCompute Objects from next-generation sequencing workflows and computations. F1000Res 9:1144
14. Simonyan V, Goecks J, Mazumder R (2017) Biocompute objects—a step towards evaluation and validation of biomedical scientific computations. PDA J Pharm Sci Technol 71: 136–146
15. Hutter C, Zenklusen JC (2018) The Cancer Genome Atlas: creating lasting value beyond its data. Cell 173:283–285
16. Rozenblatt-Rosen O, Regev A, Oberdoerffer P, Nawy T, Hupalowska A, Rood JE, Ashenberg O, Cerami E, Coffey RJ, Demir E, Ding L, Esplin ED, Ford JM, Goecks J, Ghosh S, Gray JW, Guinney J, Hanlon SE, Hughes SK, Hwang ES, Iacobuzio-Donahue CA, Jané-Valbuena J, Johnson BE, Lau KS, Lively T, Mazzilli SA, Pe'er D, Santagata S, Shalek AK, Schapiro D, Snyder MP, Sorger PK, Spira AE, Srivastava S, Tan K, West RB, Williams EH, Human Tumor Atlas Network (2020) The Human Tumor Atlas Network: charting tumor transitions across space and time at single-cell resolution. Cell 181:236–249
17. Babraham Bioinformatics. FastQC A quality control tool for high throughput sequence data. https://www.bioinformatics.babraham.ac.uk/projects/fastqc/. Accessed 19 Jan 2024
18. Pertea M, Kim D, Pertea GM, Leek JT, Salzberg SL (2016) Transcript-level expression analysis of RNA-seq experiments with HISAT, StringTie and Ballgown. Nat Protoc 11:1650–1667
19. Barretina J, Caponigro G, Stransky N, Venkatesan K, Margolin AA, Kim S, Wilson CJ, Lehár J, Kryukov GV, Sonkin D, Reddy A, Liu M, Murray L, Berger MF, Monahan JE, Morais P, Meltzer J, Korejwa A, Jané-Valbuena J, Mapa FA, Thibault J, Bric-Furlong E, Raman P, Shipway A, Engels IH, Cheng J, Yu GK, Yu J, Aspesi P Jr, de Silva M, Jagtap K, Jones MD, Wang L, Hatton C, Palescandolo E, Gupta S, Mahan S, Sougnez C, Onofrio RC, Liefeld T, MacConaill L, Winckler W, Reich M, Li N, Mesirov JP, Gabriel SB, Getz G, Ardlie K, Chan V, Myer VE, Weber BL, Porter J, Warmuth M, Finan P, Harris JL, Meyerson M, Golub TR, Morrissey MP, Sellers WR, Schlegel R, Garraway LA (2019) Addendum: The Cancer Cell Line Encyclopedia enables predictive modelling of anticancer drug sensitivity. Nature 565:E5–E6

Chapter 3

Using the Cancer Epitope Database and Analysis Resource (CEDAR)

Zeynep Koşaloğlu-Yalçın, Randi Vita, Nina Blazeska, Bjoern Peters, and Alessandro Sette

Abstract

The Cancer Epitope Database and Analysis Resource (CEDAR) is a freely accessible catalog of cancer epitope and receptor data linked to the biological, immunological, and clinical contexts in which they were described. CEDAR data is populated by manual curation of the cancer literature and provides a central resource for researchers to access information about cancer antigens and their specific epitopes, which is relevant to our understanding of the role that the immune system plays in cancer progression, prevention, and treatment. In this chapter, we aim to provide a comprehensive overview of the database section of CEDAR. This includes a detailed description of all available query parameters, guidance on navigating through the query results, and a demonstration of how CEDAR can aid cancer research, featuring example research scenarios and queries.

Key words Tumor antigens, Epitopes, Neoantigens, Cancer immunology, Immunotherapy, Database

1 Introduction

Adaptive immunity relies on two main types of responses; one is mediated by B cells, while the other is mediated by T cells. Molecules that are recognized by immune cells are called antigens. Receptors on the surface of B cells (BCRs) and T cells (TCRs) bind to specific parts of antigens, called epitopes, which can trigger an immune response.

Antigens expressed by cancer cells, namely, tumor antigens, play an essential role in the diagnosis and treatment of cancer. Tumor antigens include proteins, glycoproteins, glycolipids, and carbohydrates. Tumor antigens can be broadly categorized into tumor-specific antigens (TSAs), which are restricted to tumor cells, and tumor-associated antigens (TAAs), antigens that are present in both tumor cells and normal cells [1]. TSAs include antigens derived from oncogenic viruses (e.g., human papillomavirus

Alexander Krasnitz and Pascal Belleau (eds.), *Cancer Bioinformatics*, Methods in Molecular Biology, vol. 2932,
https://doi.org/10.1007/978-1-0716-4566-6_3,

(HPV)), products of mutated genes (also called neoantigens), and cancer germline or cancer-testis antigens that are normally not expressed in adult tissue but reexpressed in tumor tissue (e.g., carcinoembryonic antigen (CEA), melanoma-associated antigens (MAGEs)) [2, 3]. TAAs include differentiation or tissue-specific antigens that are expressed by tumors and the normal tissues from which they arise (e.g., melanoma-associated antigen recognized by T cells (MART-1), Glycoprotein 100 (gp100)) and overexpressed antigens that are expressed in normal cells but are expressed at considerably higher levels on tumor cells (e.g., Her2/neu, Survivin, wild-type p53) [3, 4].

Cancer antigens are routinely used as diagnostic markers and provide targets for immunotherapeutic treatments [5]. Historically, antigen-targeting immunotherapies yielded limited success in clinical trials (reviewed in [6]). However, with the advent of immune-checkpoint blockade (ICB) therapies that can remove the immune inhibitory signals at the tumor site, interest in antigen-based immunotherapies has resurged. For example, epitope-based vaccines can elicit strong and durable immune responses when combined with ICB (reviewed in [7]). The transfer of epitope-specific T cells and T cell receptors, or T cells with chimeric antigen receptors (CAR T cells), is also being studied and has shown clinical success [8–12]. While epitopes from shared antigens expressed across cancers of different individuals provide potential targets for more broadly applicable immunotherapies, neoantigens that are highly tumor-specific are of particular interest for personalized strategies. Indeed, neoantigen-based immunotherapies were shown to be highly effective when compared to therapies based on shared antigens [13, 14]. Data about tumor antigens and their specific epitopes is relevant to our understanding of the role that the immune system plays in cancer progression, prevention, and treatment.

Given the importance of cancer epitopes, there is a clear need to catalog all cancer epitope-related data linked to the biological, immunological, and clinical contexts. Most importantly, this information must be freely available to the scientific community in a user-friendly format. Most resources do not capture all necessary epitope data granularly and/or are not freely available. The Cancer Epitope Database and Analysis Resource (CEDAR, cedar.iedb.org) [15] was initiated in 2021 and provides a central, freely accessible catalog of cancer epitope and receptor data linked to the biological, immunological, and clinical contexts in which they were described. It builds on technical and scientific knowledge obtained from the Immune Epitope Database (IEDB, iedb.org) [16] and utilizes similar infrastructure and processes adapted to the cancer research setting.

CEDAR data is populated by manual curation of the cancer literature, following detailed curation guidelines, and is assisted by automated validation [17]. Briefly, PhD-level curators read journal

articles and identify all experimental assays where an adaptive immune receptor was tested for recognition of an epitope, whether the outcome was positive or negative. All assay types that provide epitope-specific recognition data, including assays that generate BCR and TCR sequences and 3D structures, are captured. All epitope-specific data and details of the assays used to describe the immune responses are captured. Curation is performed on a per-publication basis, with all data, including positive and negative outcomes, from each publication curated fully as a stand-alone record. Thus, any single epitope may have been described in multiple papers and various contexts.

In this chapter, we provide a detailed description of the database part of CEDAR, describing all available query parameters and explaining how to navigate and interpret the query results. We also offer four research scenarios with example queries to demonstrate how CEDAR can facilitate cancer research.

2 CEDAR Query Interface

One of the challenges for biomedical databases is to develop intuitive query interfaces while allowing the user to perform granular queries. To ensure this, we conducted interviews with several experts in the cancer immunology research field. During several iterations, we developed a search interface that makes the most requested information immediately accessible (Fig. 1).

2.1 Epitope Panel

In the "Epitope" panel, users can select if they want to include linear or discontinuous epitopes in their query. A specific peptide sequence of interest can also be entered into the "Linear Peptide" field. If no specific antigen is entered in the "Epitope Source" panel below, this query might return results including different antigens, as the same peptide might occur in different proteins. Users can search for exact matches or include epitopes from which the peptide of interest is a substring. One could also search for epitopes that are similar to the peptide of interest by selecting the "BLAST" options from the drop-down menu.

2.2 Epitope Source Panel

Users can use the "Epitope Source" panel to query CEDAR for epitopes encoded by a specific gene by entering it into the text field. When the user starts to enter text into the field, matching proteins will show up in a drop-down menu. In the example query in Fig. 1, the antigen "NY-ESO-1" was selected. The "Molecule Finder" can be accessed by clicking the "Find" button next to the text field. Here, users can search for a gene/protein of interest by entering a synonym or the UniProt ID and the source organism.

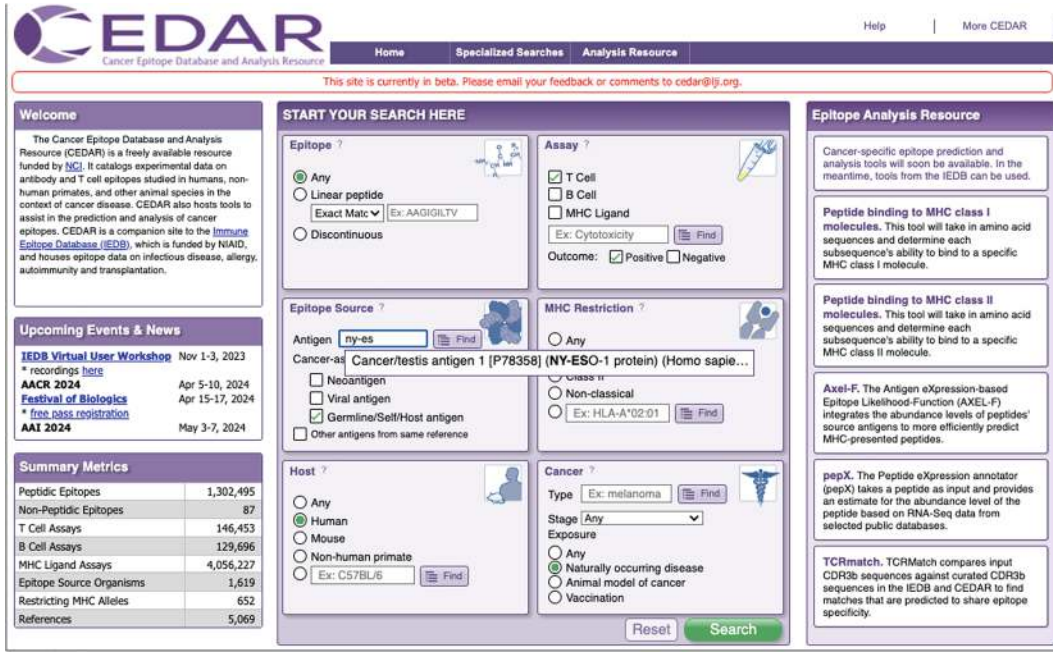


Fig. 1 CEDAR homepage: cedar.iedb.org. The CEDAR search interface makes the most requested information immediately accessible and allows intuitive yet granular queries

Additionally, users can select specific cancer-associated antigen subtypes as the epitope source, including "Neoantigen," "Viral antigen," and "Germline/Self/Host antigen." We defined these three broader categories of cancer-associated antigens that can be clearly distinguished and are mutually exclusive. Antigens that are not cancer-associated but were reported together with cancer-associated antigens in a cancer-related study can also be included in the query by selecting "Other antigens from same reference." The default selection on the CEDAR homepage excludes those antigens and only queries all cancer-associated antigens. In the example query in Fig. 1, the checkboxes "Neoantigen" and "Viral antigen" were deselected to include only germline epitopes.

2.3 Host Panel

In the "Host" panel, users can select the organism for which they want to retrieve epitope data for. While CEDAR hosts some epitope data from mouse and nonhuman primates, curation currently focuses on human epitope data. In the example query in Fig. 1, "Human" was selected as the host.

2.4 Assay Panel

In the "Assay" panel, users can select to query for epitopes that were tested using specific assay types. For a broader search, the three main assay groups, "T cell," "B cell," or "MHC ligand elution," can be selected using the checkboxes. Users can also search for a specific assay type. The assay types in CEDAR reflect

the assays being used in the literature. This includes all commonly used immunological methods such as ELISPOT, ELISA, FACs, bioassays, etc. The assay of interest can be found by typing into the text box, which will show all matching assay types in a drop-down menu.

The "Assay Finder" can be accessed by clicking the "Find" button next to the text box. The Assay Finder allows users to search for either the purpose of the assay (e.g., to measure IL-2) or the method used (e.g., ELISA). Users can also browse the "Assay Tree" to explore all available assays in CEDAR.

All experimental data entered into the CEDAR database are categorized as either positive or negative. Users can select which outcomes to include in the query using the checkboxes. In the example query in Fig. 1, the checkboxes "B Cell" and "MHC Ligand" were deselected to only include T cell assays. For the "Outcome" fields, only "Positive" was selected to only include assays with positive outcomes.

2.5 MHC Restriction Panel

In the "MHC Restriction" panel, users can select to query for the MHC molecules involved in an epitope's recognition. One can select to search for epitopes recognized in the context of MHC class I, class II, or nonclassical or narrow down the search and enter a specific MHC into the text box. The "MHC Restriction Finder" can be accessed using the finder button next to the text box. Here, users can search for an MHC or use the tree to explore all MHC molecules captured in CEDAR. In the example query in Fig. 1, "Class I" was selected to only query for MHC class I restricted epitopes.

2.6 Cancer Panel

Users can narrow down their query to only include epitopes tested in the context of a specific cancer using the "Cancer" panel. A cancer type can be selected by typing it into the text box, where matching cancer types appear in the drop-down menu. In the "Disease Finder," which can be accessed using the finder icon next to the text box, users can search for cancers by name or using Disease Ontology (DOID) [18] or the Ontology of Immune Epitopes (ONTIE) [19] identifiers. One can also explore the "Disease Tree" to select cancer types of interest. In the cancer panel, users can also select a cancer stage they want to query from the drop-down menu. The cancers can further be narrowed down according to how the host was exposed, i.e., "Naturally occurring disease" (e.g., occurrence of breast cancer), "Animal model of cancer" (e.g., C57BL/6 or BALB/c), and "Vaccination" (e.g., therapeutic vaccination with neoantigens or prophylactic vaccination against human papillomavirus (HPV)). In the example query in Fig. 1, the "Type" field was left blank to include all cancer types, and "Naturally occurring disease" was selected to only include cancers that occurred naturally without any intervention.

2.7 Querying CEDAR

As an example, we have populated the query interface to search for all MHC class I restricted germline epitopes from NY-ESO-1 that were observed in humans with naturally occurring cancer of any type and tested with any T cell assay that yielded a positive outcome. Executing this search queries the entire CEDAR content, meaning all epitope, antigen, assay, receptor records, and returns records meet these search criteria.

3 Results Display

Once a query has been executed, the search results are presented on a new page (Fig. 2a). The search criteria are displayed at the top of the results table, and any filter can also be removed at this stage by

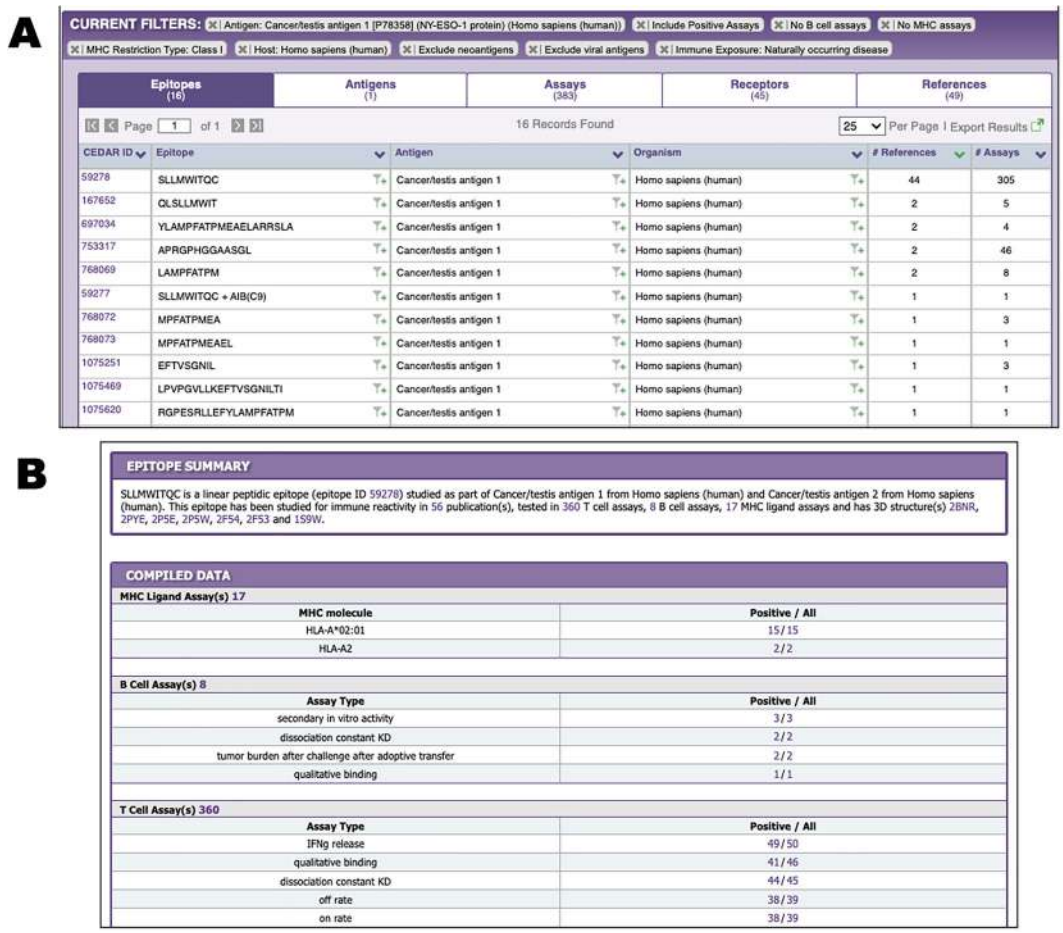


Fig. 2 CEDAR results presentation of epitope data. (**a**) The Epitopes tab displays one unique epitope per row together with its source antigen and organism, and all available assays and references are summarized as numbers. (**b**) The Epitope Details page provides information on all experimental contexts in which an epitope was tested. For each assay type the epitope was tested in, it displays how often it was tested, the outcome, and links to these assays

clicking "X" next to the parameter. Additional search panels added to the left side of the page allow the current results to be further refined by selecting additional search parameters (not shown in Fig. 2a).

The results are grouped into five tabs—"Epitopes," "Antigens," "Assays," "Receptors," and "References," which are filtered based on the selected search criteria. These different units of information reflect how users may want to utilize CEDAR; one can explore the literature, for example, on the References tab or use the Antigens tab to explore which specific antigens have been studied for immune reactivity for a cancer type of interest. Results on each tab are sorted by how much information is available, and data with the highest number of references is shown first.

3.1 Epitopes Tab

The "Epitopes" tab displays one unique epitope per row with its source antigen and organism. For each epitope, all available assays and references are summarized as numbers (Fig. 2a). If the user is interested in a specific epitope from this list, results can be filtered by clicking the funnel icon next to it. This will filter all records in the tabs Epitopes, Antigens, Assays, Receptors, and References to only show records related to the epitope of interest while maintaining all selected search parameters.

By clicking on the "Epitope ID" in the "Details" column of an epitope, a new browser tab containing the "Epitope Details" page is opened (Fig. 2b). The Epitope Details page provides information on all experimental contexts in which an epitope was tested, with a textual summary of the compiled data at the top of the page and in the data tables below. Each assay type, namely, MHC ligand, B cell, and T cell assay, is presented in a separate section of the data table and provides a summary of assay subtypes the epitope was tested in, how often it was tested, how often the outcome was positive, and links to these assays. Using the Epitope Details page, a user can quickly form opinions regarding an epitope of interest. On the Epitope Details page, all information will be displayed without considering the parameters of the initial query.

For example, the assays performed on the epitope in Fig. 2b suggest that the NY-ESO-1 epitope SLLMWITQC binds to HLA-A*02:01, as shown by 15 assays. The epitope can activate B cells, as shown by four different B cell assays. A plethora of T cell assays were performed with this epitope, all suggesting that it can activate T cells. All numbers in these data tables are links and can be clicked to access the details about the corresponding assays.

3.2 Antigens Tab

The "Antigens" tab displays one unique antigen per row with its source organism. The Antigen table also provides information on how often epitopes from each antigen were studied with counts for the number of epitopes, assays, and references. Users can further narrow their results to a single antigen using the funnel icon next to

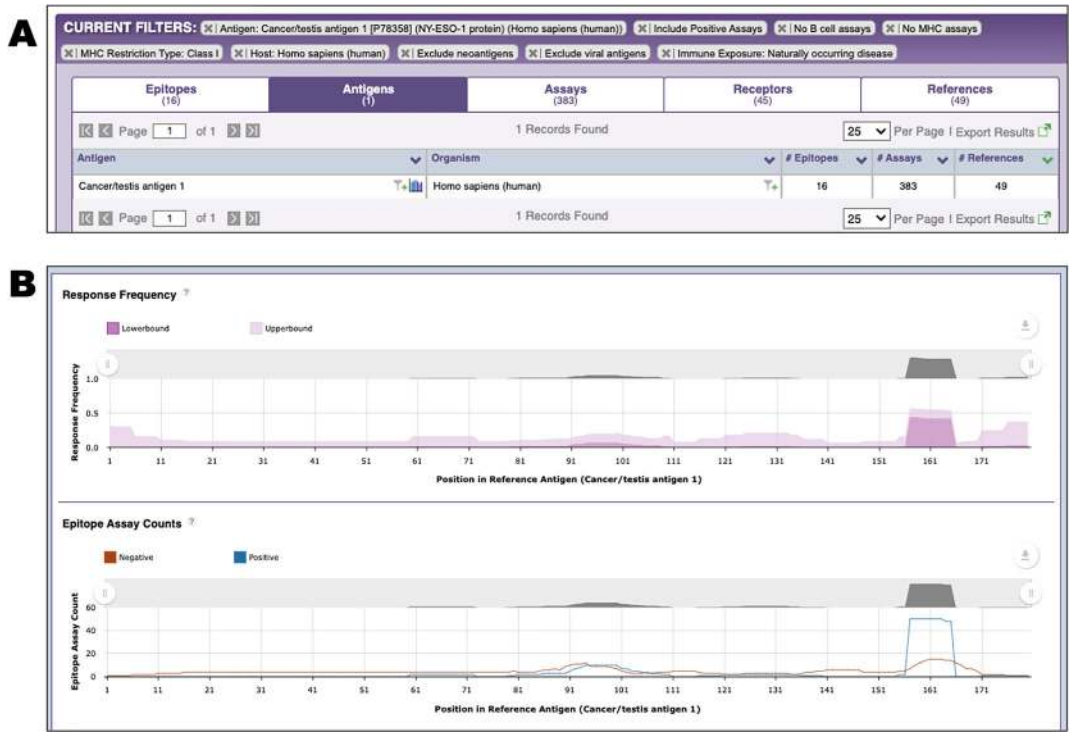


Fig. 3 CEDAR results presentation of antigen data. (**a**) The Antigens tab displays one unique antigen per row with its source organism and information on how often epitopes from each antigen were studied with counts for the number of epitopes, assays, and references. (**b**) The Immunome Browser visualizes linear epitopes along the length of the parent antigen based on sequence similarity and displays how often each protein region has been studied and in how many assays the immune response was positive or negative

the antigen of interest. In the example in Fig. 3a, only one antigen is displayed because our example query included a filter for NY-ESO-1.

The "Immunome Browser" can be accessed by clicking the bar chart icon next to the antigen of interest. The Immunome Browser visualizes linear peptidic epitopes along the length of the parent antigen based on sequence similarity. This displays how often each protein region has been studied in immune assays and in how many assays the immune response was positive or negative. Figure 3b shows the Immunome Browser output for the epitopes from NY-ESO-1 recognized in the human T cell response. The upper plot renders the lower and upper bounds of the 95% confidence interval of the response frequency for each target protein position, averaged over all epitopes mapped to that position and calculated as the number of positively responded subjects relative to the total number tested. The bottom plot shows the number of positive and negative assays averaged over epitopes mapped to each position in the protein sequence. A table below the graphs (not shown) presents results for each epitope and each protein position in a tabular

format that can be saved, along with the graph images, for further analysis and publication. The user can interactively zoom in and out of the plots to a specific protein region, and the table will update accordingly. Of note, the Immunome Browser is not available for neoantigens at this time, but future work is planned to implement a suitable version of the Immunome Browser for neoantigens.

3.3 Assays Tab

The "Assays" tab displays all assays used to identify and test the epitopes that meet the search criteria. If an epitope was tested with multiple assays, all assays that meet the search criteria will be listed here. The Assays parent tab is grouped into three subtabs: T cell, B cell, and MHC ligand assays. In our example in Fig. 4, only the T cell tab is populated because our example query included a filter for T cell assays.

Each row in the assays tab describes an assay with a set of fields providing relevant details such as the following:

(i) Reference. This field contains the reference the assay was reported in, and the first author, journal, and year of publication are displayed. By clicking the reference, the reference details page can be opened (described below).

(ii) Epitope. This field describes the specific epitope sequence utilized in the assay together with its source antigen and organism. By clicking here, the "Epitope Details" page can also be accessed.

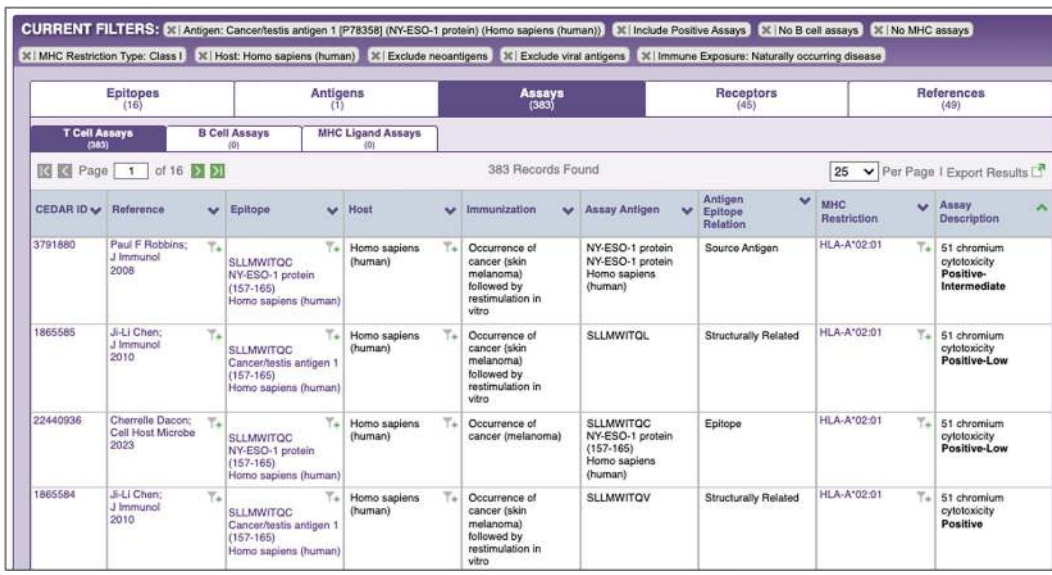


Fig. 4 CEDAR results presentation of assay data. The Assays tab displays all assays used to identify and test the epitopes that meet the search criteria and is grouped into three tabs: T cell, B cell, and MHC Ligand Assays. Each row in the assays tab describes an assay with a set of fields providing relevant details such as the type and outcome of the assay and the reference it was reported in

(iii) Host. This field described the host from which the immune cells or antibodies for the assay were derived.

(iv) Immunization. This field contains details about how the host was immunized. This includes, for example, "Occurrence of Disease" (e.g., naturally occurring lung cancer), "Administration in vivo" (e.g., injection of a mouse with a cancer antigen), "Environmental exposure" (e.g., natural exposure to HPV through known sexual contact), and "Vaccination" (e.g., therapeutic vaccination with neoantigens).

(v) Assay Antigen. This field contains the specific antigen that was used in the assay. This is not always the same as the epitope because CEDAR captures all experimental contexts in which an epitope-specific receptor is tested. For example, if an epitope-specific T cell line is tested for proliferation in response to a whole protein, this will be included in CEDAR.

(vi) Antigen Epitope Relation. This field describes the relationship between the epitope and the assay antigen. For example, "Epitope" describes a case where the epitope itself was used in the assay, whereas "Source Antigen" describes an assay where the whole protein was used and the specific epitope sequence within the antigen was also known.

(vii) MHC Restriction. This describes the MHC restriction of the epitope that was utilized in the assay. Depending on what was reported in the reference, this can be a general description like "HLA class II" or a specific MHC molecule like "HLA-DRB1*04:02."

(viii) Assay Description. This field provides a brief description of the assay, including the type of assay and the outcome. The assay type includes the method and what was measured, e.g., ELISPOT and IFNg release. All experimental data entered into CEDAR are categorized as positive or negative. If authors provide such information, additional granularity is available for positive data with values of positive-high, positive-intermediate, and positive-low. For assay types with quantitative measurements, the numerical values and units are also available.

3.4 Receptors Tab

The "Receptors" tab has two subtabs for T cell and B cell receptors (Fig. 5a). In each row of the TCR or BCR tab, a unique receptor is displayed along with the species it was reported in, the type (e.g., αβ TCRs), and the Chain 1 and Chain 2 CDR3 sequences, if available. Analogous to the "Epitope" tab, details about a receptor can be retrieved by clicking on its ID in the "Group ID" column.

On the "Receptor Details" page (Fig. 5b), relevant information about the species that the receptor was reported in is displayed, and links to the 3D structures in PDB are summarized at the top of the

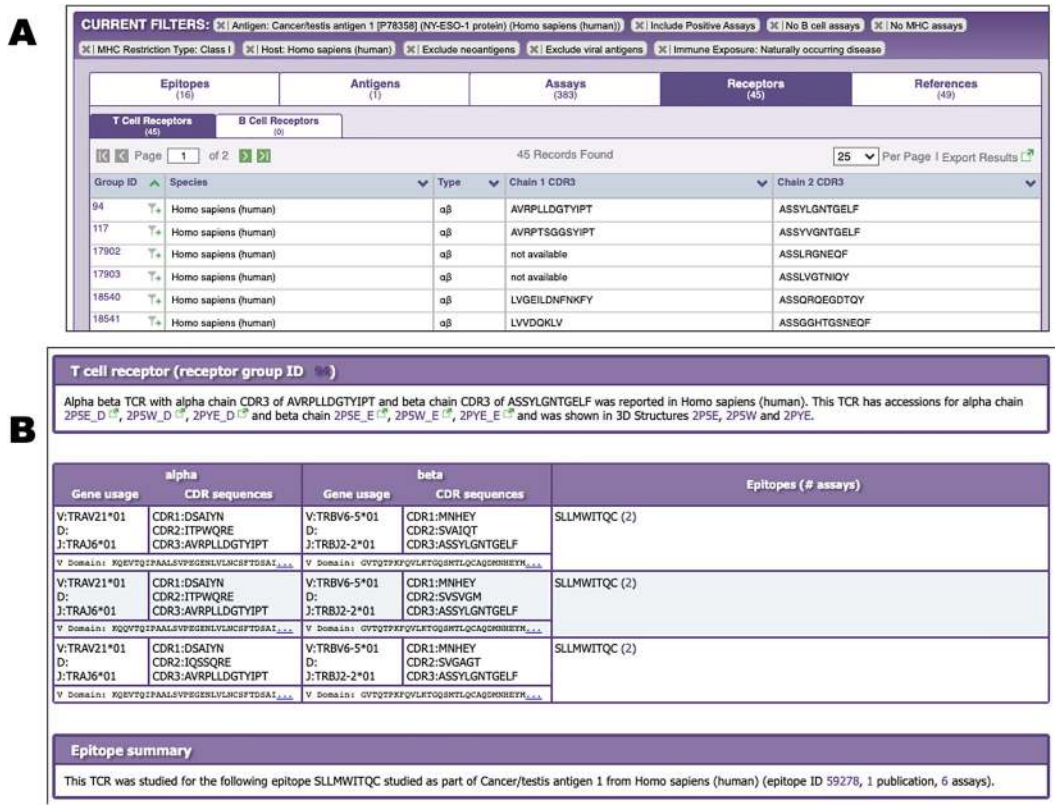


Fig. 5 CEDAR results presentation of receptor data. (**a**) The Receptors tab has two subtabs for T cell and B cell receptors. In each row of the TCR or BCR tab, a unique receptor is displayed along with the species it was reported in, the type, and the Chain 1 and Chain 2 CDR3 sequences, if available. (**b**) The Receptor Details page provides relevant information about the species that the receptor was reported in and links to the 3D structures in PDB. In the data table below, the gene usage and sequences for CDR1, CDR2, and CDR3 as well as the full-length receptor sequence are displayed if available for both alpha and beta chains. Sequences of the epitopes that the receptor was reported to recognize are also listed, together with links to the corresponding epitopes in CEDAR

page. In the data table below, the gene usage and sequences for CDR1, CDR2, and CDR3 as well as the full-length receptor sequence are displayed for both alpha and beta chains, when available. Some authors only provide a single CDR3 sequence for one chain, while others provide full-length receptor sequences for both chains. Sequences of the epitopes that the receptor was reported to recognize are also listed, together with links to the corresponding epitopes in CEDAR.

3.5 References Tab

The "References" tab (Fig. 6a) lists all references containing epitope data that meet the search criteria. Listed are the PubMed ID (PMID), author list, title, journal, and year of publication. Clicking on the PMID opens a new browser tab to the corresponding PubMed entry. The "Reference Details" page (Fig. 6b) can be

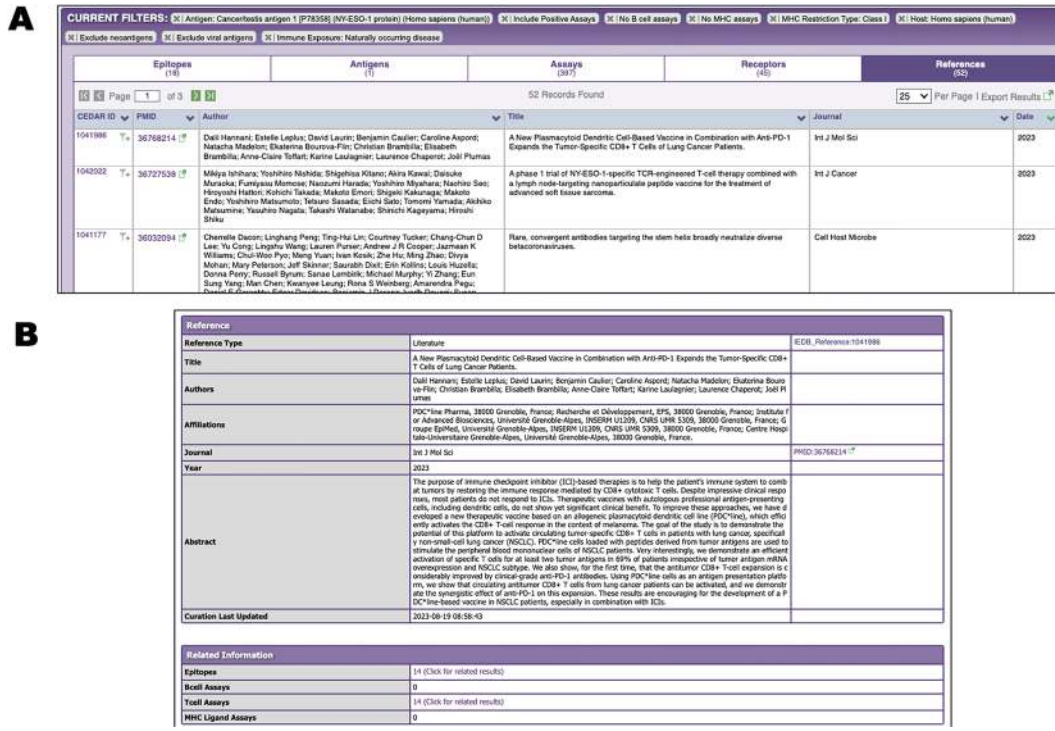


Fig. 6 CEDAR results presentation of curated references. (**a**) The references tab (lists all references containing epitope data that meet the search criteria together with PubMed ID, author list, title, journal, and year of publication). (**b**) The reference details page provides more details about the publication, like the abstract and the authors' affiliations. Also, all epitopes and assays reported in the reference are listed in a summary table

accessed by clicking the CEDAR ID of the reference. Here, more details about the publication are provided, like the abstract and the authors' affiliations. Additionally, all epitopes and assays reported in the reference are listed in a summary table.

4 Downloading Data from CEDAR

After executing a query, the corresponding results from each of the tabs, Epitopes, Antigens, Assays, Receptors, and References, can be downloaded by clicking "Export Results" in the upper right corner of the results display. The user has the option to choose what format to download the data in and which specific columns to include.

All data stored in CEDAR can also be retrieved as a bulk download. The download page can be accessed from the homepage by clicking "More CEDAR" on the top right corner and then selecting "Database Export" from the pop-up. Complete database exports are available in XML and MySQL formats. Most users prefer to download data in a tabular format, which can be found

under the section "CSV Metric Exports." Here, the user has the option to separately download T cell (tcell_full_v3.zip), B cell (bcell_full_v3.zip), and MHC ligand elution assays (mhc_ligand_full.zip). These tables contain all information on the assay level, and an epitope will be listed multiple times if it was reported in the context of multiple different assays.

5 Example Search Scenarios and Queries

To illustrate how CEDAR can facilitate cancer research, we present four example research questions along with example queries.

5.1 Research Scenario I

For a literature review, the user wants to retrieve all references that contain assays testing epitopes from the antigen "Prostate-specific antigen."

- In the Epitope Source panel, enter "Prostate-specific antigen" and select the first entry from the drop-down menu "Prostate-specific antigen [P07288] (Homo sapiens (human))."
- In the Assay panel, check both outcome types, "Positive" and "Negative."
- All other search panels remain in the default selection.
- Execute the query and navigate to the "References" tab in the Results display.
- Click "Export Results" in the upper right corner of the "References" Results display to download an Excel file containing details about 74 references.

5.2 Research Scenario II

The user conducted a study about recurrent mutations in cancer and identified putative neoantigens using prediction tools. The neoepitope "SYLDSGIHF" from Catenin beta 1 (CTNNB1) was found to be recurring in many patients and was predicted to be potentially immunogenic. The user wants to investigate the experimental evidence for immunogenicity reported in the literature.

- In the Epitope panel, enter the epitope sequence "SYLDSGIHF" into the "Linear peptide" text field.
- In the Assay panel, check both outcome types, "Positive" and "Negative."
- All other search panels remain in the default selection.
- Execute the query and inspect the results.
- In the "Antigen" tab, two antigens will be listed: "neoantigen: Catenin beta-1 Homo sapiens (human)" and "neoantigen: Catenin beta 1 (UniProt:A0A2R8Y7Z0)." These are two different isoforms of the protein, and the epitope "SYLDSGIHF" was reported in both in different references.

- Navigate to the "Epitopes" tab and open the "Epitope Details" page by clicking on the CEDAR ID of the epitope.
- In the "Epitope Details" page, under "MHC Ligand Assays," the user can see that the epitope was tested for binding HLA-A*24:02, three times with positive outcomes every time.
- In the "Epitope Details" page, under "T cell Assays," the user can see that the epitope was tested in eight T cell assays: three cytotoxicity assays, three IFNg release assays, one GM-CSF release assay, and one tetramer assay measuring qualitative binding. As most of these assays had positive outcomes, there is strong evidence that the epitope "SYLDSGIHF" can activate T cells.

5.3 Research Scenario III

Certain viruses can cause cancer by integrating viral DNA into the human genome. As viral antigens are foreign to the human immune system, they are attractive targets for immunotherapy [20]. To explore putative antigen and epitope targets, the user wants to retrieve a list of viral antigens and epitopes in head and neck cancer with positive outcomes in T cell recognition assays.

- In the Epitope Source panel, deselect the checkboxes "Neoantigen" and "Germline/Self/Host antigen" to select only "Viral antigen."
- In the Assay panel, deselect "B Cell" and "MHC Ligand," and only select the outcome type "Positive" to query for T cell assays with positive outcomes.
- In the Cancer panel, enter "head and neck cancer" into the Cancer "Type" field, and select "head and neck cancer (ID: DOID:11934, head/neck neoplasm" from the drop-down menu.
- All other search panels remain in the default selection.
- Executing the query will show 92 epitopes from four antigens of the organism "Alphapapillomavirus 9."
- Download the epitopes by clicking "Export Results" on the upper right corner of the "Epitopes" tab.

5.4 Research Scenario IV

Neoantigen vaccines can elicit strong and durable antitumor immune responses when paired with immune-checkpoint blockade therapy. The user wants to investigate which neoantigens have been successfully used for vaccinating cancer patients.

- In the Epitope Source panel, deselect the checkboxes "Viral antigen" and "Germline/Self/Host antigen" to select only "Neoantigen."
- In the Host panel select Human.
- In the Cancer panel, under "Exposure," select "Vaccination."

- All other search panels remain in the default selection.
- Executing the refined query results in 520 epitopes from 377 antigens, tested with 775 assays, reported in 44 references.
- We can further refine the query on the Results page to include only therapeutic vaccinations. On the left side of the Results display, under "Cancer" and "Exposure," select "Vaccination (therapeutic)."
- Executing the refined query results in 309 epitopes from 233 antigens, tested with 514 assays, reported in 30 references.
- Download the epitopes by clicking "Export Results" on the upper right corner of the "Epitopes" tab.

6 Conclusion and Future Plans

The Cancer Epitope Database and Analysis Resource (CEDAR) provides a freely accessible, comprehensive collection of cancer epitope and receptor data curated from the literature, linked to the biological, immunological, and clinical contexts in which they were described.

Overall, as of January 31, 2024, a total of 289,028 cancer-associated epitopes have been curated from 5001 references. These epitopes have been tested in 910,389 assays (87,903 T cell assays, 64,766 B cell assays, and 757,720 MHC ligand elution assays). Furthermore, 36,928 receptors have been curated: 36,555 T cell receptors and 373 antibodies. There are currently approximately 3145 possibly curatable cancer references still outstanding in the curation queue.

We have several plans to improve CEDAR's search interface. Given the increased interest in neoantigens, we are working on including the option to search for neoantigens encoded by a specific mutation. We are also working on including more information about the variant encoding the neoantigen. Currently, it is not possible to search for receptor sequences directly from the main CEDAR search interface, but it is only possible to filter for specific sequences once a query has been executed. We plan to provide this option directly on the main CEDAR search interface in the future. Finally, we also plan to provide an Application Programming Interface (API) to allow users to perform custom and more complex queries and larger batch queries.

With the advancement of technologies like high-throughput cancer genomics and single-cell analysis of immune repertoires, we expect that the amount of cancer epitope data will continue to increase in the future. As the amount of data continues to grow, it becomes increasingly important to extract meaningful insights from this data. Making cancer epitope data easily and freely

accessible creates an opportunity for the research community to reanalyze the data, generate new research hypotheses, and gain new insights into cancer immunology.

Acknowledgments

Research reported in this publication was supported by the National Institutes of Health grant U24CA248138 and contract 75N93019C00001.

References

1. Valilou SF, Rezaei N (2019) Chapter 4 – Tumor antigens. In: Rezaei N, Keshavarz-Fathi M (eds) Vaccines for cancer immunotherapy. Academic Press, London, pp 61–74. https://doi.org/10.1016/B978-0-12-814039-0.00004-7
2. Kufe DW, Holland JF, Frei E, American Cancer Society (2003) Holland Frei cancer medicine, vol 1, 6th edn. BC Decker, Hamilton
3. Jhunjhunwala S, Hammer C, Delamarre L (2021) Antigen presentation in cancer: insights into tumour immunogenicity and immune evasion. Nat Rev Cancer 21(5):298–312. https://doi.org/10.1038/s41568-021-00339-z
4. Kosaloglu-Yalcin Z, Blazeska N, Carter H, Nielsen M, Cohen E, Kufe D, Conejo-Garcia J, Robbins P, Schoenberger SP, Peters B, Sette A (2021) The cancer epitope database and analysis resource: a blueprint for the establishment of a new bioinformatics resource for use by the cancer immunology community. Front Immunol 12:735609. https://doi.org/10.3389/fimmu.2021.735609
5. Hellström KE, Hellström I (2002) Tumor antigens. In: Bertino JR (ed) Encyclopedia of cancer, 2nd edn. Academic Press, New York, pp 459–466. https://doi.org/10.1016/B0-12-227555-1/00251-3
6. Melero I, Gaudernack G, Gerritsen W, Huber C, Parmiani G, Scholl S, Thatcher N, Wagstaff J, Zielinski C, Faulkner I, Mellstedt H (2014) Therapeutic vaccines for cancer: an overview of clinical trials. Nat Rev Clin Oncol 11(9):509–524. https://doi.org/10.1038/nrclinonc.2014.111
7. Hollingsworth RE, Jansen K (2019) Turning the corner on therapeutic cancer vaccines. NPJ Vaccines 4:7. https://doi.org/10.1038/s41541-019-0103-y
8. Majzner RG, Ramakrishna S, Yeom KW, Patel S, Chinnasamy H, Schultz LM, Richards RM, Jiang L, Barsan V, Mancusi R, Geraghty AC, Good Z, Mochizuki AY, Gillespie SM, Toland AMS, Mahdi J, Reschke A, Nie EH, Chau IJ, Rotiroti MC, Mount CW, Baggott C, Mavroukakis S, Egeler E, Moon J, Erickson C, Green S, Kunicki M, Fujimoto M, Ehlinger Z, Reynolds W, Kurra S, Warren KE, Prabhu S, Vogel H, Rasmussen L, Cornell TT, Partap S, Fisher PG, Campen CJ, Filbin MG, Grant G, Sahaf B, Davis KL, Feldman SA, Mackall CL, Monje M (2022) GD2-CAR T cell therapy for H3K27M-mutated diffuse midline gliomas. Nature 603(7903):934–941. https://doi.org/10.1038/s41586-022-04489-4
9. Shah NN, Lee DW, Yates B, Yuan CM, Shalabi H, Martin S, Wolters PL, Steinberg SM, Baker EH, Delbrook CP, Stetler-Stevenson M, Fry TJ, Stroncek DF, Mackall CL (2021) Long-term follow-up of CD19-CAR T-cell therapy in children and young adults with B-ALL. J Clin Oncol 39(15):1650–1659. https://doi.org/10.1200/JCO.20.02262
10. Spiegel JY, Patel S, Muffly L, Hossain NM, Oak J, Baird JH, Frank MJ, Shiraz P, Sahaf B, Craig J, Iglesias M, Younes S, Natkunam Y, Ozawa MG, Yang E, Tamaresis J, Chinnasamy H, Ehlinger Z, Reynolds W, Lynn R, Rotiroti MC, Gkitsas N, Arai S, Johnston L, Lowsky R, Majzner RG, Meyer E, Negrin RS, Rezvani AR, Sidana S, Shizuru J, Weng WK, Mullins C, Jacob A, Kirsch I, Bazzano M, Zhou J, Mackay S, Bornheimer SJ, Schultz L, Ramakrishna S, Davis KL, Kong KA, Shah NN, Qin H, Fry T, Feldman S, Mackall CL, Miklos DB (2021) CAR T cells with dual targeting of CD19 and CD22 in adult patients with recurrent or refractory B cell malignancies: a phase 1 trial. Nat Med 27(8):1419–1431. https://doi.org/10.1038/s41591-021-01436-0

11. Brightman SE, Naradikian MS, Miller AM, Schoenberger SP (2020) Harnessing neoantigen specific CD4 T cells for cancer immunotherapy. J Leukoc Biol 107(4):625–633. https://doi.org/10.1002/JLB.5RI0220-603RR
12. Yamamoto TN, Kishton RJ, Restifo NP (2019) Developing neoantigen-targeted T cell-based treatments for solid tumors. Nat Med 25(10): 1488–1499. https://doi.org/10.1038/s41591-019-0596-y
13. Weber JS, Carlino MS, Khattak A, Meniawy T, Ansstas G, Taylor MH, Kim KB, McKean M, Long GV, Sullivan RJ, Faries M, Tran TT, Cowey CL, Pecora A, Shaheen M, Segar J, Medina T, Atkinson V, Gibney GT, Luke JJ, Thomas S, Buchbinder EI, Healy JA, Huang M, Morrissey M, Feldman I, Sehgal V, Robert-Tissot C, Hou P, Zhu L, Brown M, Aanur P, Meehan RS, Zaks T (2024) Individualised neoantigen therapy mRNA-4157 (V940) plus pembrolizumab versus pembrolizumab monotherapy in resected melanoma (KEYNOTE-942): a randomised, phase 2b study. Lancet 403:632. https://doi.org/10.1016/S0140-6736(23)02268-7
14. Rojas LA, Sethna Z, Soares KC, Olcese C, Pang N, Patterson E, Lihm J, Ceglia N, Guasp P, Chu A, Yu R, Chandra AK, Waters T, Ruan J, Amisaki M, Zebboudj A, Odgerel Z, Payne G, Derhovanessian E, Muller F, Rhee I, Yadav M, Dobrin A, Sadelain M, Luksza M, Cohen N, Tang L, Basturk O, Gonen M, Katz S, Do RK, Epstein AS, Momtaz P, Park W, Sugarman R, Varghese AM, Won E, Desai A, Wei AC, D'Angelica MI, Kingham TP, Mellman I, Merghoub T, Wolchok JD, Sahin U, Tureci O, Greenbaum BD, Jarnagin WR, Drebin J, O'Reilly EM, Balachandran VP (2023) Personalized RNA neoantigen vaccines stimulate T cells in pancreatic cancer. Nature 618(7963):144–150. https://doi.org/10.1038/s41586-023-06063-y
15. Kosaloglu-Yalcin Z, Blazeska N, Vita R, Carter H, Nielsen M, Schoenberger S, Sette A, Peters B (2023) The Cancer Epitope Database and Analysis Resource (CEDAR). Nucleic Acids Res 51(D1):D845–D852. https://doi.org/10.1093/nar/gkac902
16. Vita R, Mahajan S, Overton JA, Dhanda SK, Martini S, Cantrell JR, Wheeler DK, Sette A, Peters B (2019) The Immune Epitope Database (IEDB): 2018 update. Nucleic Acids Res 47(D1):D339–D343. https://doi.org/10.1093/nar/gky1006
17. Salimi N, Edwards L, Foos G, Greenbaum JA, Martini S, Reardon B, Shackelford D, Vita R, Zalman L, Peters B, Sette A (2020) A behind-the-scenes tour of the IEDB curation process: an optimized process empirically integrating automation and human curation efforts. Immunology 161(2):139–147. https://doi.org/10.1111/imm.13234
18. Schriml LM, Munro JB, Schor M, Olley D, McCracken C, Felix V, Baron JA, Jackson R, Bello SM, Bearer C, Lichenstein R, Bisordi K, Dialo NC, Giglio M, Greene C (2022) The Human Disease Ontology 2022 update. Nucleic Acids Res 50(D1):D1255–D1261. https://doi.org/10.1093/nar/gkab1063
19. Greenbaum JA, Vita R, Zarebski LM, Sette A, Peters B (2010) Ontology development for the immune epitope database. In: Bioinformatics for immunomics. Springer, New York, pp 47–56
20. von Witzleben A, Wang C, Laban S, Savelyeva N, Ottensmeier CH (2020) HNSCC: tumour antigens and their targeting by immunotherapy. Cells 9(9):2103. https://doi.org/10.3390/cells9092103

Chapter 4

Quantifying the Prevalence of Non-B DNA Motifs as a Marker of Non-B Burden in Cancer Using NBBC

Qi Xu and Jeanne Kowalski

Abstract

Alternative DNA structures, such as Z-DNA, G-quadruplexes, and mirror repeats, have shown potential involvement in cancer etiology. NBBC (Non-B DNA Burden in Cancer) is a web-based tool designed for quantifying and analyzing non-B DNA motifs within a cancer context. Herein, we provide a step-by-step protocol for employing NBBC, starting with data input and proceeding through the quantification and normalization of non-B DNA motifs that result in calculation of non-B burden. With detailed instructions for performing various motif-centric analyses based on cancer gene signatures, including DNA damage repair and response pathways for genomic stability, and other sample-level gene mutation signatures, users can explore non-B associative correlations within current cancer biology. We provide additional details on input queries into NBBC, interpret the quantitative results, and apply normalization techniques to ensure accurate comparisons across different genomic regions and non-B DNA structures.

NBBC offers a powerful and user-friendly interface for the cancer research community. This chapter serves as an essential, enhanced instructional guide for researchers to leverage NBBC in their cancer biomarker investigations for an understanding of the potential role of non-B DNA in contributing to them.

Key words NBBC, Non-B DNA, Cancer bioinformatics, Genome instability

1 Introduction

Noncanonical DNA refers to DNA structures that differ from the canonical B-DNA double helix structure, including G-quadruplexes, cruciform, slipped structures, triplexes, and Z-DNA [1–4]. It has been discovered that non-B DNA-forming sequences can induce genetic instability in human cancer genomes, suggesting a role in cancer development [1].

While there are several non-B DNA databases and prediction tools that exist, the majority of these tools primarily focus on individual motif sequences in isolation [5, 6]. We introduce the concept of "non-B burden" as a quantitative marker to provide the capacity to integrate information from non-B DNA motifs into a comprehensive, genome-wide perspective. This viewpoint has been

Alexander Krasnitz and Pascal Belleau (eds.), *Cancer Bioinformatics*, Methods in Molecular Biology, vol. 2932,
https://doi.org/10.1007/978-1-0716-4566-6_4,

notably absent in prior non-B DNA research, which underscores its innovative nature and potential. A parallel concept in cancer research can be found in the idea of tumor mutation burden. As tumor mutation burden quantifies the prevalence of mutations and can inform on cancer prognosis and treatment response, our introduction of "non-B burden" holds a similar promise for assessing non-B DNA motif prevalence and its potential for interpretation of biological processes, particularly within the realm of cancer research.

To accelerate the non-B burden analysis in cancer, NBBC is designed to serve as a new analysis and visualization platform for the exploration of non-B DNA. NBBC serves as a valuable resource for researchers investigating the role of non-B DNA structures in cancer and other genetic diseases.

In this chapter, we demonstrate a "how to" guide to access and explore NBBC [7] for gene signature analyses that may be further combined with familiar downstream correlatives. In doing so, we present details on the quantitative approach and normalization methods that are available at various genomic levels, including the gene level, signature level, and sample level. Altogether, we provide a guided approach to the use of NBBC, including input query, quantification, and normalization of non-B DNA motifs, and how to use non-B burden in downstream applications.

2 Materials

2.1 The Web Server, NBBC

To simplify the use of non-B burden calculation and introduce it for wide, non-bioinformatic research uses, we introduce NBBC, a Non-B DNA Burden Explorer in Cancer. NBBC is an online web server that provides non-B burden calculation, non-B burden visualization, and non-B motif exploration. NBBC serves to conduct non-B burden computations and offers normalizations that enable comparisons across genes or non-B structures. It provides visualizations for descriptive analysis of burden values, burden distribution, and burden-based gene clustering. The NBBC webserver is accessible without any login requirements and is completely free to use (https://kowalski-labapps.dellmed.utexas.edu/NBBC/).

2.2 Non-B DNA Motif Data

The non-B DNA-forming motif data in NBBC are downloaded from the Non-B DB 2.0 database (hg19 build) [5, 6, 8]. There are seven non-B structure motifs included: A-phased repeat (APR, $n = 2386$ motifs), G-quadruplexes (G4, $n = 361{,}232$ motifs), Z-DNA ($n = 404{,}192$ motifs), inverted repeats (IR, $n = 5{,}771{,}570$ motifs), mirror repeats (MR, $n = 1{,}378{,}864$ motifs), direct repeats (DR, $n = 1{,}113{,}354$ motifs), and short tandem repeats (STR, $n = 2{,}826{,}360$ motifs). A subset of MR and IR motifs are further delineated within the application to represent

Triplex (Triplex-MR, n = 412,028 motifs) and Cruciform (Cruciform-IR, n = 147,152 motifs) motifs, respectively. To ensure the reliability and relevance of our data, we have sourced our non-B DNA data in the NBBC web server, and so users have no need to download the non-B DNA data.

2.3 Query Input

To summarize, the input for NBBC includes three major types: a single gene (or a list of genes), a gene signature, or genomic coordinates (see below). To accommodate these input types, we have provided four options for users to choose from. In the web server ("Input Page" tab), Option 1 offers pre-populated gene sets related to cancer, while Option 2 provides molecular signatures of cancer cell lines. Alternatively, users may manually input gene symbols in Option 3 or upload genomic coordinates of interesting regions in Option 4.

2.3.1 Non-B Burden at Gene Level

(a) *A single gene*. The typical use case is a quick single gene search by typing the "hgnc symbol" of gene, such as KRAS, BRCA, etc.

(b) *Gene signatures*. The input includes popular cancer signatures, cell line molecular signatures, or user-defined signatures. NBBC offers built-in cancer-related gene sets for quick query, including cancer hallmark gene signatures from the MSigDB database [9], DNA damage repair and response gene signatures [10, 11], and molecular signatures (option 2) from the Genomics of Drug Sensitivity in Cancer database [12].

(c) *Genomic coordinates of regions*. This is a general option where the query region is not a full region of a gene but a subregion of the gene, such as cancer-specific mutation sites or regions with copy number alterations. In this case, user can upload the region in query (genomic coordinates) in a table (*see* **Note 1**).

2.3.2 Non-B Burden at Sample Level (Burden in Batch)

Furthermore, NBBC allows users to upload "multiple groups" of genomic regions (genes, mutation regions) in batch and calculate the non-B burden for each group. We named it as "Burden in Batch." Using "Burden in Batch," it allows users to calculate the non-B burden for each sample (*see* **Note 2**) to enable further downstream analysis of associations. The input format of "Burden in Batch" is a table with four columns (*group_id*, *chromosome*, *start*, *end*). Each row represents a genomic region and the "*group_id*" is used to group the genomic regions.

3 Methods

3.1 How to Use NBBC to Calculate Non-B Burden

The NBBC web server provides a user-friendly platform for calculating the non-B DNA burden within genomic sequences. Our protocol illustrates this calculation through two exemplar use cases.

(a) *Gene-Level Non-B Burden (Basic Use)*
NBBC can process both single and multiple genes to calculate the non-B burden. Here, we utilize the "Homologous Recombination" gene signature [10, 11], the maintenance of which plays an important role in cancer genome stability. This illustration demonstrate the foundational computation of non-B burden and emphasize the critical role of normalization techniques in the analysis.

(b) *Sample-Level Non-B Burden (Advanced Use)*
This second application showcases the "Burden in Batch" feature of NBBC. We employed mutation site data from 511 samples from the TCGA-LGG (Lower Grade Glioma) dataset [13]. Each sample comprises a list of mutation sites, complete with chromosomal coordinates. The "Burden in Batch" tool in NBBC is utilized to quantify the number of non-B motifs that overlap with these mutation sites, thereby summarizing the non-B burden for each sample. The objective here is to illustrate the integration of non-B burden analysis with clinical datasets, highlighting its utility in sample-level evaluation.

3.2 Start with the "Input Page"

The "Input Page" of the NBBC web server is the starting point for users to input their data for non-B burden analysis. This page is designed to accommodate various user requirements through four distinct input options:

(a) *Option 1: Built-In Cancer-Related Gene Signatures.* Users can choose from predefined gene sets related to cancer, such as those involved in DNA damage repair, response gene pathways, cancer hallmark gene sets, and oncogenes, among others.

(b) *Option 2: Cancer Cell Line-Specific Features.* This option provides a selection of molecular features specific to cancer cell lines, including mutations and copy number alterations.

(c) *Option 3: Manual Gene Input.* For users interested in conducting a quick query, this interface allows the manual input of single or multiple genes.

(d) *Option 4: Genomic Coordinates Upload.* Users can upload genomic coordinates that define regions of interest, such as mutation sites, to assess the non-B burden in the context of mutation-localized regions.

With the integration of these input capabilities, NBBC offers comprehensive coverage for non-B burden calculations at varying levels of genomic detail, from precise mutation sites to expansive gene signatures (Fig. 1).

3.3 Non-B Burden at Gene Level (Basic Use)

1. *Select genes as input*

   In the input page, under option 1, Select the "Homologous recombination" gene set under the "DNA Damage and Repair" category. There will be a preview window at the bottom to show the selected genes (34 genes). Click "Next > Gene Screen" to navigate to the "Gene Screen" module.

2. *The quantification of non-B DNA motif*

   Non-B burden is calculated by counting the occurrence of non-B forming regions associated with each specific non-B DNA type. After the query gene list is selected, the non-B burden will be calculated in the background by the web server. And the calculation value will be visualized in bar plots on the right panel of the "Gene Screen" tab page. There are two tabs called "Total Burden" and "Burden by type."

   (a) *Total Burden.* NBBC includes a Total Burdens plot that visualizes the total non-B burden in each gene as a bar plot. This visual provides users with a summary of the non-B burdens for each query unit.

   (b) *Burden by non-B type.* The stacked bar plot shows the non-B burdens by type for each gene in the query. This allows users to easily compare the non-B burdens across different genes and identify genes with potentially higher burdens from certain non-B structures.

   (c) *Interactive plots.* Since this is an interactive plot, users can select hovers for more details of each data point. This feature allows users to obtain a more detailed understanding of the non-B burden for each gene in the query and can aid in identifying potential targets for further investigation.

   (d) *Other options.* On the left side panel of this page, there are checkbox options, where NBBC provides users with the option to display only a subset of genes or non-B types. This flexible functionality allows users to tailor their analysis to their specific research needs and enables them to focus on the genes or non-B types of interest.

3. *The normalization of non-B burden*

   Normalization ensures meaningful comparisons of non-B burden across diverse genes and non-B DNA structure types (*see* **Note 3**). The NBBC provides several metrics for normalization to facilitate these comparisons (Fig. 2).

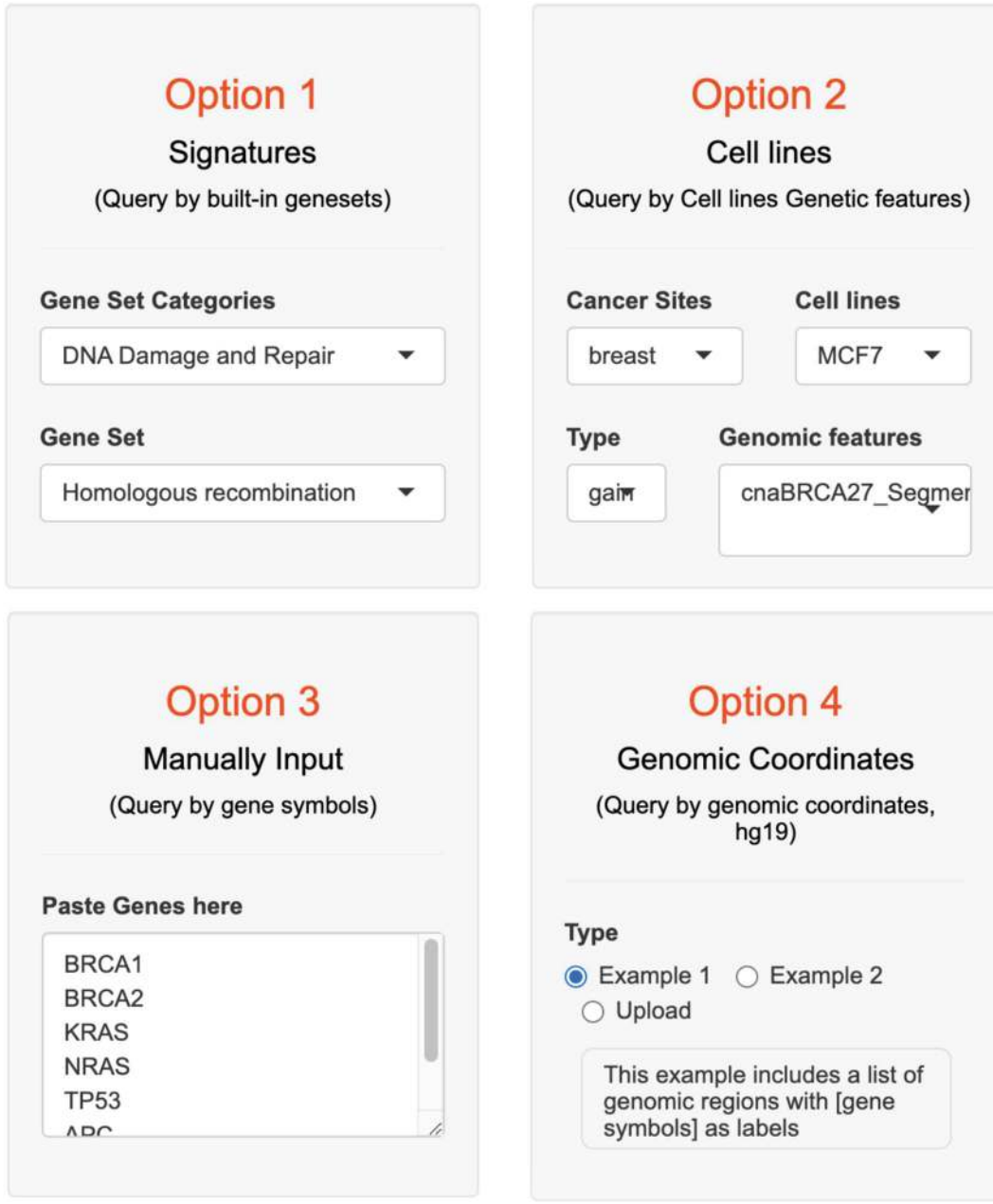


Fig. 1 The input page of NBBC. In the "Input Page," users can select from four distinct options to begin their analysis with NBBC. Option 1 provides a selection of predefined gene sets pertinent to cancer research. Option 2 allows users to explore molecular signatures from various cancer cell lines. For a more customized approach, Option 3 enables the manual entry of specific gene symbols. Option 4 permits the uploading of genomic coordinates for regions of interest

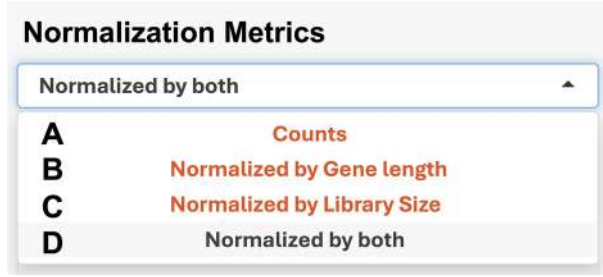


Fig. 2 The normalization options for non-B burden. (A) "Count" indicates raw motif counts, serving as the unnormalized base measure. (B) "By Gene Length" adjusts non-B burden based on gene size for comparability across genes. (C) "By Library Size" scales non-B burden to the non-B motif library size, aiding in analyzing various non-B DNA types. (D) "Normalization by Both" offers a measure standardizing non-B burden by both gene length and motif library and is the default normalized metric in NBBC

(a) *Raw Motif Counts*: The basic measure of non-B burden without any normalization, representing the number of non-B motifs present.

(b) *Normalization by Gene Length*: Adjusts the non-B burden by the length of the gene region. This is critical when comparing different genes, as it accounts for the varying sizes of genomic regions.

(c) *Normalization by Motif Library Size*: Tailors the non-B burden relative to the size of the non-B motif library. This normalization is particularly useful when assessing the burden of different types of non-B DNA within a single gene.

(d) *Normalization by Both (CPKM)*: This comprehensive measure normalizes non-B burden both by the length of query regions (per kilobase, 10^3) and by the library sizes of non-B motifs (per million, 10^6), facilitating a comparison across both genes and non-B motif types. This default unit, CPMK (counts per kilobase per million), ensures a standardized comparison by considering both the prevalence of non-B motifs and the scale of the genomic and motif libraries.

Normalization allows for the comparison of non-B burden across both different genomic regions and among distinct types of non-B DNA for the assessment of differences in motif prevalence among them. Specific applications of normalization include the following (Fig. 3):

(a) *Motif Library Size Normalization*: Recommended when the aim is to compare the non-B burden across various non-B DNA types within the same gene. It adjusts for the

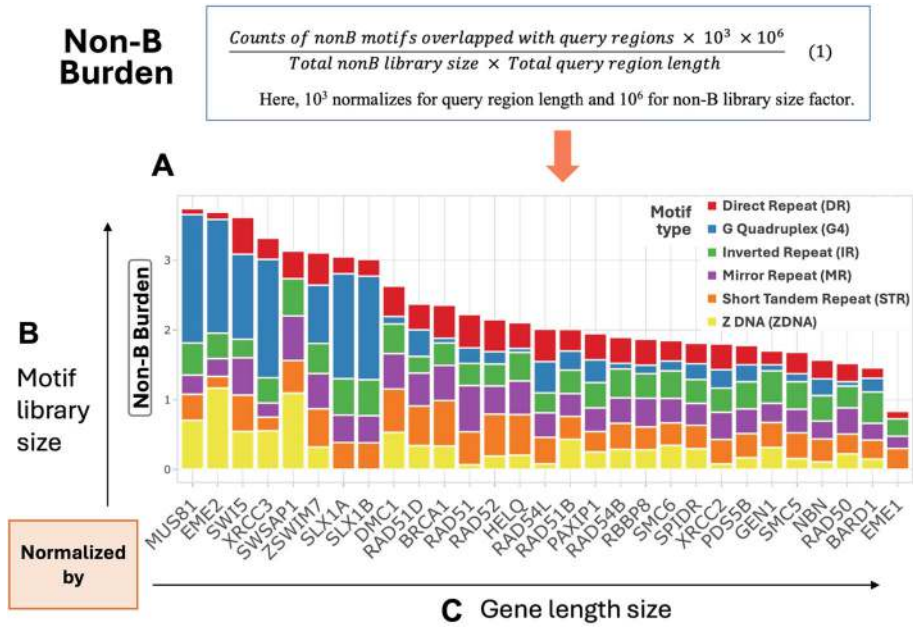


Fig. 3 The non-B burden distribution in homologous recombination signature queried in NBBC. (A) The non-B burden is visualized in a bar plot with normalization across both genes and non-B types. (B) The normalization by non-B motif library sizes allow burden comparison across non-B types. (C) The normalization by gene length enables comparisons across genes

total number of motifs in each non-B motif type, providing a direct comparison of prevalence across different structures.

(b) *Gene Length Normalization*: Advised for comparing non-B burden across multiple genes within a signature. This accounts for the potential bias where longer genes might inherently contain more non-B motifs, thus offering a more accurate reflection of non-B motif density, rather than sheer quantity, within the gene's region.

The aim of this part (*Gene-Level Non-B Burden*) in NBBC is to conduct a gene-level analyses of non-B representation that could prove helpful to focus on a single or subset of genes of interest for hypothesis generation.

3.4 Non-B Burden at Sample Level (Advance Use)

1. *Objective of "Burden in Batch"*

- *Conception*. The "Burden in Batch" function extends NBBC's capabilities, enabling the computation of non-B burden at the sample level. It allows for the analysis of multiple groups of genomic regions, such as mutated regions across different tumor samples, facilitating a deeper exploration of non-B burden in relation to clinical data.

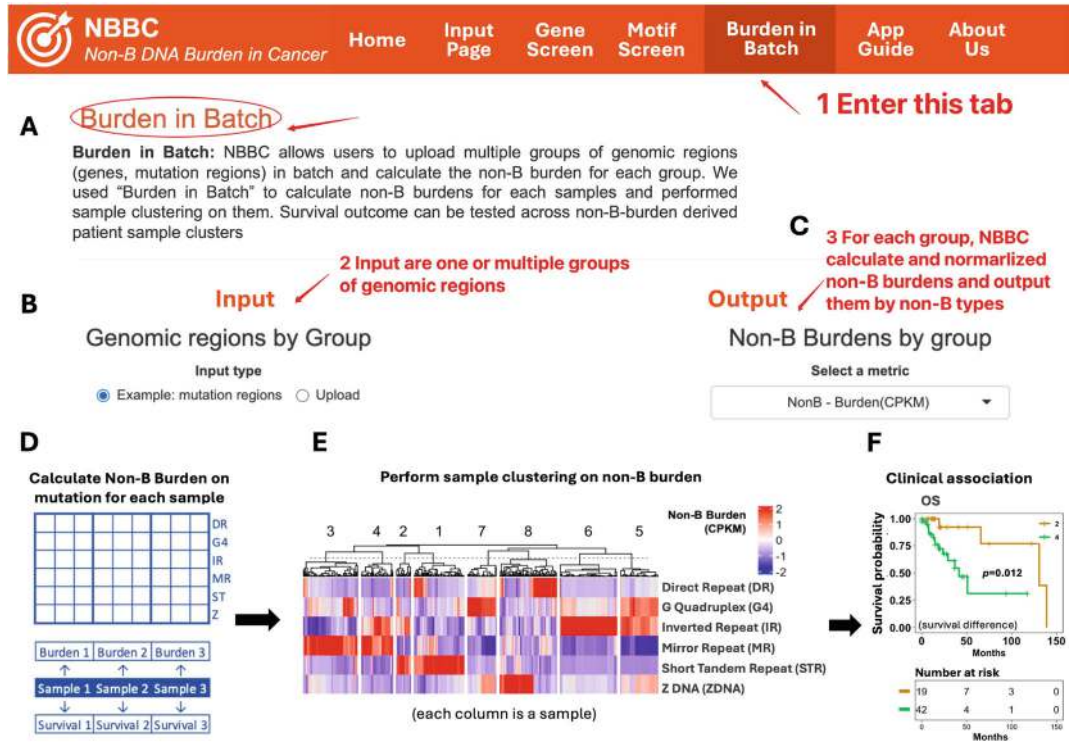


Fig. 4 The "burden in batch" analysis in NBBC. (**a**) The web interface to "Burden in Batch" analysis to calculate non-B burden at sample level in batch. (**b**) The input area to upload the query genomic region grouped by group ids. (**c**) The output area to output non-B burden with normalization options. (**d**) A graphical summary of non-B burden calculation at the sample level with mapped survival data. (**e**) The heatmap visualizes the clustering results of TCGA-LGG sample based on mutation-localized, sample-level non-B burdens. (**f**) Cluster 1 (STR high) and Cluster 4 (MR-IR high) show a significant overall survival (OS) difference ($p = 0.0012$)

- *Application*. This feature is especially beneficial for generalized queries based on extensive sets of genomic regions. These can represent distinct tumor samples, thereby enabling the calculation of sample-specific non-B burden. This analysis can reveal potential links between non-B burden and clinical outcomes.

This function can be accessed under the "Burden in Batch" tab from the home page in NBBC (Fig. 4a).

2. How to perform "Burden in Batch" analysis in NBBC
   - *Prepare input table*. The input format of "Burden in Batch" is a table with four columns (*group_id*, *chromosome*, *start*, *end*)—collectively referred to as the "query table." The example data can be downloaded by clicking the "Download Example" button on the page. Each row represents a genomic region and the "*group_id*" is used to group the genomic regions. The typical use case for "Burden in Batch" is to calculate non-B burden for each tumor sample using a list of genomic coordinates.

- *Case study input*: For our example, we calculate mutation-localized non-B burden for TCGA Lower Grade Glioma (LGG) samples ($n = 511$), using genome-wide mutation site regions as the query input. In other words, 511 groups of genomic mutation regions from 511 LGG samples will be used as input for the calculation (*see* **Note 4**).
- *Calculate non-B burden for each sample*. Users compile a query table containing the genomic mutation regions for all samples, ensuring columns are accurately labeled. Upon navigating to the "Burden in Batch" input page, users upload their table and initiate the analysis (Fig. 4b). The server quantifies non-B motifs coinciding with the input regions and applies the default CPKM normalization. All the non-B motifs that overlapped with mutation regions of each sample will be summarized for non-B burden at sample level. On the right side of the page, a matrix will be output to show non-B burden from different types (columns) of each sample (rows). Users can choose to download the metrics by clicking the "Download Burden Output" button.
- *Choose an appropriate normalization method*. Similar to the normalization methods gene-level non-B burden, the sample-level non-B burden also supports four types of metrics, including "counts," "normalize by motif library size," "normalize by motif region length," and "normalize by both (CPKM)." One difference here is the "region length." Different from "gene length", the region length here refers to the total length of all query regions for each sample (Fig. 4c).
- *Perform sample clustering on non-B burden*. Subsequent to calculating the non-B burden matrix, we perform clustering analysis on the samples based on their non-B burden. The data is standardized using Z-score normalization, with k-means clustering applied in this instance. But users can choose other preferred clustering methods. The purpose here is to showcase how this sample-level quantification of non-B burden can be used for downstream analysis (Fig. 4e).
- *Associate clusters with clinical data*. The clustering process categorizes samples into groups characterized by distinct non-B burden profiles. These groups can then be utilized in further analyses, such as survival or association studies, to elucidate the potential clinical significance of non-B burden variations in cancer (Fig. 4f).

NBBC serves as a valuable resource for researchers investigating the role of non-B DNA structures in cancer and other genetic diseases. By offering an accessible platform for analyzing and visualizing non-B DNA burden within a cancer context, NBBC enables the quantification and exploration of non-B DNA by a wide, non-bioinformatic user base.

4 Notes

1. Each row in the table represents a region of query. The four column names are required to be the same as the example [*hgnc_symbol*, *chromosome*, *start*, *end*]. [*hgnc_symbol*] can be either gene name or any tags to annotate the region but may not be duplicated. [*chromosome*, *start*, *end*] takes genomic coordinates as input.
2. "Burden in Batch" is different from "Genomic coordinates of regions." Although they both take genomic coordinate as input, "Genomic coordinates of regions" only allows one region under one label, such as one subregion in a specific gene. So in its input table, the "*hgnc_symbol*" column should be unique. In contrast, "Burden in Batch" allows querying a list of regions under one label, such as all the mutated regions in one tumor sample. Therefore, in its input table, the "*group_id*" column (such sample id) will be repeating for different genomic coordinate. Then non-B burden can be summarized for this sample.
3. The concept of normalization in RNA-seq analysis [14], exemplified by metrics like CPM (counts per million), RPKM (reads per kilobase of transcript, per million mapped reads), plays a similar role as in our approach for normalizing non-B burden. Similar to RNA-seq, where these normalization techniques ensure the comparability of gene expression values across diverse samples, normalization methods employed in non-B burden calculations serve a similar purpose.
4. The mutation data for TCGA-LGG is downloaded from UCSC Xena [13, 15] (https://tcga-xena-hub.s3.us-east-1.amazonaws.com/download/mc3%2FLGG_mc3.txt.gz). The Xena hub is at https://tcga.xenahubs.net, and the title is "dataset: somatic mutation (SNP and INDEL)—MC3 public version."

Acknowledgments

The NBBC website is hosted on an AWS server provided by the Kowalski Lab and supported by UT-Austin IT Solutions and Dell Medical School at UT-Austin.

References

1. Zhao J, Bacolla A, Wang G, Vasquez KM (2010) Non-B DNA structure-induced genetic instability and evolution. Cell Mol Life Sci 67: 43–62
2. Bansal A, Kaushik S, Kukreti S (2022) Non-canonical DNA structures: diversity and disease association. Front Genet 13:959258
3. Makova KD, Weissensteiner MH (2023) Noncanonical DNA structures are drivers of genome evolution. Trends Genet 39:109–124
4. Wang G, Vasquez KM (2023) Dynamic alternative DNA structures in biology and disease. Nat Rev Genet 24:211–234
5. Cer RZ, Donohue DE, Mudunuri US, Temiz NA, Loss MA, Starner NJ, Halusa GN, Volfovsky N, Yi M, Luke BT et al (2012) Non-B DB v2.0: a database of predicted non-B DNA-forming motifs and its associated tools. Nucleic Acids Res 41:D94–D100
6. Cer RZ, Bruce KH, Mudunuri US, Yi M, Volfovsky N, Luke BT, Bacolla A, Collins JR, Stephens RM (2011) Non-B DB: a database of predicted non-B DNA-forming motifs in mammalian genomes. Nucleic Acids Res 39: D383–D391
7. Xu Q, Kowalski J (2023) NBBC: a non-B DNA burden explorer in cancer. Nucleic Acids Res 51(W1):W357–W364
8. Cer RZ, Donohue DE, Mudunuri US, Temiz NA, Loss MA, Starner NJ, Halusa GN, Volfovsky N, Yi M, Luke BT et al (2013) Non-B DB v2.0: a database of predicted non-B DNA-forming motifs and its associated tools. Nucleic Acids Res 41:D94–D100
9. Liberzon A, Birger C, Thorvaldsdottir H, Ghandi M, Mesirov JP, Tamayo P (2015) The Molecular Signatures Database (MSigDB) hallmark gene set collection. Cell Syst 1:417–425
10. Wood RD, Mitchell M, Sgouros J, Lindahl T (2001) Human DNA repair genes. Science 291:1284–1289
11. Wood RD, Mitchell M, Lindahl T (2005) Human DNA repair genes, 2005. Mutat Res 577:275–283
12. Yang W, Soares J, Greninger P, Edelman EJ, Lightfoot H, Forbes S, Bindal N, Beare D, Smith JA, Thompson IR et al (2013) Genomics of Drug Sensitivity in Cancer (GDSC): a resource for therapeutic biomarker discovery in cancer cells. Nucleic Acids Res 41:D955–D961
13. Ellrott K, Bailey MH, Saksena G, Covington KR, Kandoth C, Stewart C, Hess J, Ma S, Chiotti KE, McLellan M (2018) Scalable open science approach for mutation calling of tumor exomes using multiple genomic pipelines. Cell Syst 6:271–281.e7
14. Zhao Y, Li M-C, Konaté MM, Chen L, Das B, Karlovich C, Williams PM, Evrard YA, Doroshow JH, McShane LM (2021) TPM, FPKM, or normalized counts? A comparative study of quantification measures for the analysis of RNA-seq data from the NCI patient-derived models repository. J Transl Med 19: 1–15
15. Goldman M, Craft B, Hastie M, Repečka K, McDade F, Kamath A, Banerjee A, Luo Y, Rogers D, Brooks AN (2018) The UCSC Xena platform for public and private cancer genomics data visualization and interpretation. *bioRxiv*. https://doi.org/10.1101/326470

Chapter 5

Starfish: Deciphering Complex Genomic Rearrangement Signatures Across Human Cancers

Lisui Bao

Abstract

Complex genomic rearrangements (CGRs) in cancer often originate from abnormal cellular structures such as micronuclei and chromatin bridges. However, the primary mechanisms responsible for CGR formation in disease tissues remain unclear, particularly due to the challenges in fully capturing these processes. To address this, we have developed "Starfish," a computational algorithm to decipher CGR signatures and infer their forming mechanisms by analyzing distinctive copy number variations and breakpoint patterns. Here, we provide practical guidance on the application of "Starfish," available as an R package, to study CGR signatures in human cancers.

Key words Starfish, Algorithm, Complex genomic rearrangements, Copy number variation, Structural variation, Cancer

1 Introduction

In a range of human cancers, structural variations (SVs) manifest themselves in numerous forms [1–3], from simple ones like deletions, duplications, inversions, and translocations to complex genomic rearrangements (CGRs) like chromothripsis [4, 5]. Understanding the molecular mechanisms behind SV formation is crucial for clinical advancements. Recent in vitro studies of CGR formation have pinpointed two primary mechanisms: chromosomal fragmentation in micronuclei [6] and chromatin bridge-induced chromothripsis [7, 8]. Both mechanisms can lead to extrachromosomal DNA (ecDNA) amplification [9], all contributing to the genomic complexity and instability in cancer. Despite these insights, the complex nature of these genomic rearrangements makes it challenging to study their full extent. A comprehensive understanding of the diverse mechanisms fueling CGR in disease tissues remains elusive.

Alexander Krasnitz and Pascal Belleau (eds.), *Cancer Bioinformatics*, Methods in Molecular Biology, vol. 2932,
https://doi.org/10.1007/978-1-0716-4566-6_5,

To address this gap, our study introduces a computational algorithm, termed "Starfish," designed to deduce CGR formation mechanisms based on the analysis of copy number variation (CNV) patterns and breakpoint distributions [10]. By integrating five distinct features—CGR breakpoint dispersion score, copy loss percentage, copy gain percentage, telomere loss percentage, and maximum copy number—we executed unsupervised consensus clustering on CGRs from the Pan-Cancer Analysis of Whole Genomes (PCAWG) project, identifying six unique CGR signatures (1, ecDNA/DM/HSR; 2, BFB cycles/chromatin bridge; 3, Large loss; 4, Micronuclei; 5, Large gain; 6, Hourglass). Signature 1, 2, and 4 were assigned to known CGR mechanisms using data from five experimental studies on chromothripsis. Further, we developed a neural network classifier, the "Starfish classifier," trained on these features and CGR signatures to categorize CGRs in additional samples.

2 Methods

2.1 Software Environment

Starfish is entirely written in R and has been tested on R 3.5.0, 3.6.0, and 4.0.0, but any version above 3.0.1 should be fine to use.

2.2 Package Dependencies

Starfish depends on several R packages including graph, BiocGenerics, S4Vectors, foreach, Genome-InfoDb, GenomicRanges, IRanges, ConsensusClusterPlus, neuralnet, plyr, data.table, MASS, ggplot2, gridExtra, dplyr, factoextra, dendextend, gplots, ggpubr, reshape2, cowplot, patchwork, Cairo, and ggforce (*see* **Note 1**). It also requires a modified version of ShatterSeek (ShatterSeeky) [4] which is provided as ShatterSeeky_0.4.tar.gz (*see* **Note 2**). As ShatterSeeky depends on the other packages, please install it as the last one. In case you cannot reproduce the example results, you may need to install the versions of packages specified below. It is possible that the newer versions are not compatible.

The versions of tested packages are graph 1.68.0, BiocGenerics 0.36.0, S4Vectors 0.28.0, foreach 1.5.1, Genome-InfoDb 1.26.0, GenomicRanges 1.42.0, IRanges 2.24.0, ConsensusClusterPlus 1.54.0, neuralnet 1.44.2, plyr 1.8.6, data.table 1.14.0, MASS 7.3.53.1, ggplot2 3.3.3, gridExtra 2.3, dplyr 1.0.5, factoextra 1.0.7, dendextend 1.15.1, gplots 3.1.1, ggpubr 0.4.0, reshape2 1.4.4, cowplot 1.1.1, patchwork 1.1.1, Cairo 1.5.12.2, ggforce 0.3.3.

2.3 Installation

Starfish can be installed remotely by

```
if (!requireNamespace("devtools", quietly = TRUE)) install.
packages("devtools")
library(devtools)
install_github("yanglab-computationalgenomics/Starfish")
```

Alternatively, one can download the latest release of Starfish by running the following command in a bash terminal:

```
git clone https://github.com/yanglab-computationalgenomics/Starfish.git

R CMD INSTALL
```

2.4 Load SV, CNV, and Sample Data into R

One can load Starfish and the test data provided by the package.

```
library(Starfish)
data("example_sv")
data("example_cnv")
data("example_sample")
```

Running this command loads three R data frames, corresponding to the somatic SVs, CNVs, and gender information of six tumors of the PCAWG project. Starfish accepts SV and CNV calls from any variant caller, as long as they are encoded in the required format (see below).

2.5 SV Data

Starfish requires the SV data to be stored in a data frame with the following columns:

"chrom1" (character): chromosome for the first breakpoint (*see* **Note 3**)

- "pos1" (numeric): genomic coordinate for the first breakpoint, and pos1 should be smaller than pos2 for intrachromosomal SVs (i.e., DEL, DUP, h2hINV, and t2tINV)
- "chrom2" (character): chromosome for the second breakpoint
- "pos2" (numeric): genomic coordinate for the second breakpoint
- "svtype" (character): type of SV, encoded as DEL (deletion-like; +/−), DUP (duplication-like; −/+), h2hINV (head-to-head inversion; +/+), t2tINV (tail-to-tail inversion; −/−), and TRA (translocation)
- "strand1" (character): strand information for the first breakpoint (e.g., + for DEL)
- "strand2" (character): strand information for the second breakpoint (e.g., − for DEL)
- "sample" (character): sample ID

```
print(head(example_sv), row.names = FALSE)
```

chrom1	pos1	chrom2	pos2	strand1	strand2	svtype	sample
11	36219433	2	52018778	+	−	TRA	07f16397-71bb-4594-ad4d-caa7d2baeabd
11	39939312	11	39941515	+	−	DEL	07f16397-71bb-4594-ad4d-caa7d2baeabd
12	40811811	12	63685824	+	+	h2hINV	07f16397-71bb-4594-ad4d-caa7d2baeabd
12	40838496	12	85876878	−	−	t2tINV	07f16397-71bb-4594-ad4d-caa7d2baeabd
12	57913369	12	75990126	−	+	DUP	07f16397-71bb-4594-ad4d-caa7d2baeabd

2.6 CNV Data

Starfish requires the CNV data to be stored in a data frame with the following columns:

- "chromosome" (character): chromosome
- "start" (numeric): start coordinate for the CN segment
- "end" (numeric): end coordinate for the CN segment
- "total_cn" (numeric): total copy number, which could be either integer or decimal
- "sample" (character): sample ID

```
print(head(example_cnv), row.names = FALSE)
```

chromosome	start	end	total_cn	sample
4	68582077	191154275	2	07f16397-71bb-4594-ad4d-caa7d2baeabd
12	112420925	112603567	1	07f16397-71bb-4594-ad4d-caa7d2baeabd
6	68336278	69355562	2	07f16397-71bb-4594-ad4d-caa7d2baeabd

Starfish will calculate the copy number (CN) baseline for each chromosome to identify loss and gain fragments in a chromosome-wise manner (*see* **Note 4**). It will use the gender information to call CN losses and gains on chromosome X.

2.7 Gender Data

Starfish requires the gender data to be stored in a data frame with following columns:

- "sample" (character): sample ID, which should match the ones in SV and CNV data frames.
- "gender" (character): gender for the sample, which could be "Female," "female," "F," "f," "Male," "male," "M," or "m." If the gender is unknown, any other characters could be given,

such as "unknown," and the gender will be inferred by the CN baseline of chromosome X (see details in Subheading 2.8.2).

```
print(head(example_sample), row.names = FALSE)
```

sample	gender
3db6e6cc-1a06-49b9-834e-b6611cde4c4b	Male
2b6d4d66-7f0b-4bc0-b3d6-171956a937c5	Female
18f5e75e-c623-11e3-bf01-24c6515278c0	Male
fc8130df-897d-5404-e040-11ac0d485e0a	Female

2.8 Running Starfish

There are four components of Starfish: **starfish_link**, **starfish_feature**, **starfish_sig**, and **starfish_plot**. Once the input data are loaded, we could run them step by step to obtain and examine intermediate outputs, or we could use **starfish_all** to run them all at once.

2.8.1 starfish_link

starfish_link loads an SV data frame and identifies "seed" CGR regions (see below), "linked" CGR regions, and complex SVs using modified ShatterSeek. Oscillating-copy-state criteria is removed from the original ShatterSeek package. In each sample, linked regions are identified if they are connected by at least two translocations within 10 kb of any seed regions. The search is performed iteratively until no new linked regions could be found. A CGR event is defined as all connected seed and linked regions combined (Fig. 1).

Usage

```
starfish_link(sv_file, prefix = "")
```

Input

- "sv_file": the SV data frame defined previously.
- "prefix": the prefix for all intermediate files; default is none.

Example

Starfish will output the progress of "seed" region and "linked" region calling.

```
starfish_link_out = starfish_link(example_sv, prefix = "exam-
ple")
```

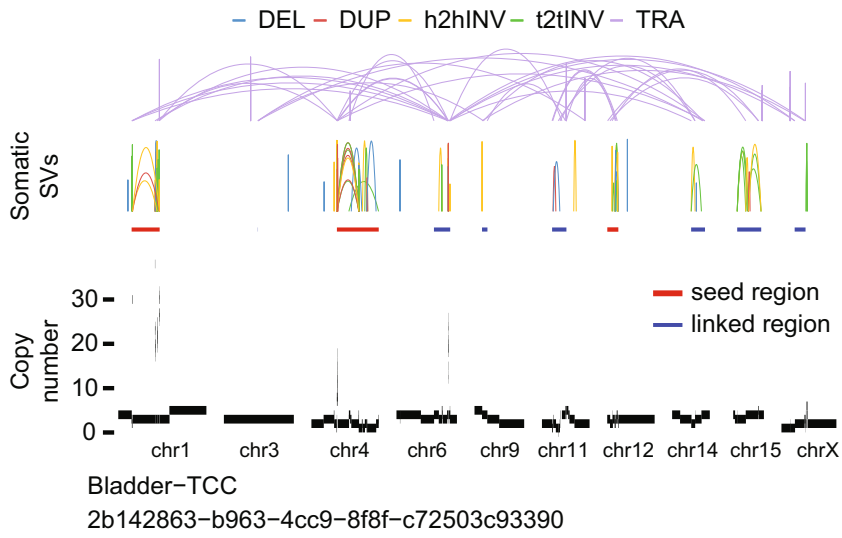


Fig. 1 "Seed" regions and "linked" regions in a CGR of a Bladder-TCC sample "2b142863-b963-4cc9-8f8f-c72503c93390." SVs are shown as arcs, and SV types are shown in different colors. "Seed" and "linked" regions are plotted as colored bars

```
##  Running…
##  Evaluating the statistical criteria
##  Successfully finished!
##  [1] "Iteration 1"
##  [1] "Starfish is connecting chromothriptic regions…"
```

Output

The function **starfish_link** returns an instance that contains three data frames: *interleave_tra_complex_sv*, *mergecall*, and *starfish_call*.

interleave_tra_complex_sv contains complex SVs in the CGR regions, and "complex = 2" means both breakpoints are in the CGR regions. A CSV file "example_connected_CGR_complex_SV.csv" will be generated under the working space:

```
print(head(starfish_link_out$interleave_tra_complex_sv), row.
names = FALSE)
```

chrom1	pos1	chrom2	pos2	svtype	complex	sample	cluster_id
19	22936736	4	4188431	TRA	2	18f5e75e-c623-11e3-bf01-24c6515278c0	18f5e75e-c623-11e3-bf01-24c6515278c0_4_19
19	22937258	4	4188272	TRA	2	18f5e75e-c623-11e3-bf01-24c6515278c0	18f5e75e-c623-11e3-bf01-24c6515278c0_4_19

mergecall contains "seed" CGR regions which is identified by the modified ShatterSeek. "Call3" refers to the criteria of "at least three interleaved intrachromosomal SVs and four or more interchromosomal SVs" to call CGRs used by ShatterSeek, and "call6" refers to the criteria of "at least six interleaved intrachromosomal SVs." The regions pass either "call3" or "call6" criteria will be defined as "seed" CGR regions.

```
print(head(starfish_link_out$mergecall), row.names = FALSE)
```

chr	start	end	sample	call3	call6
3	162683868	169237872	07f16397-71bb-4594-ad4d-caa7d2baeabd	no	CGR
5	30217747	163056821	07f16397-71bb-4594-ad4d-caa7d2baeabd	no	CGR
X	126784598	152948223	07f16397-71bb-4594-ad4d-caa7d2baeabd	no	CGR

starfish_call contains final CGR regions (both seed and linked regions), and a CSV file "example_connected_CGR_event.csv" will be generated under the working space.

```
print(head(starfish_link_out$starfish_call), row.names = FALSE)
```

chr	start	end	sample	CGR_status	link_chromosome	cluster_id
19	19053346	23741257	18f5e75e-c623-11e3-bf01-24c6515278c0	CGR	4_19	18f5e75e-c623-11e3-bf01-24c6515278c0_4_19
4	4188272	4188431	18f5e75e-c623-11e3-bf01-24c6515278c0	link	4_19	18f5e75e-c623-11e3-bf01-24c6515278c0_4_19

The columns are as follows:

- "chr": chromosome of the CGR region.
- "start": start coordinate of the CGR region.
- "end": end coordinate of the CGR region.
- "sample": sample ID.
- "CGR_status": "CGR" means the region is the "seed" region identified by modified ShatterSeek, and "link" means the region is the "linked" region connected to "seed" regions.
- "link_chromosome": the chromosomes linked together in the CGRs.

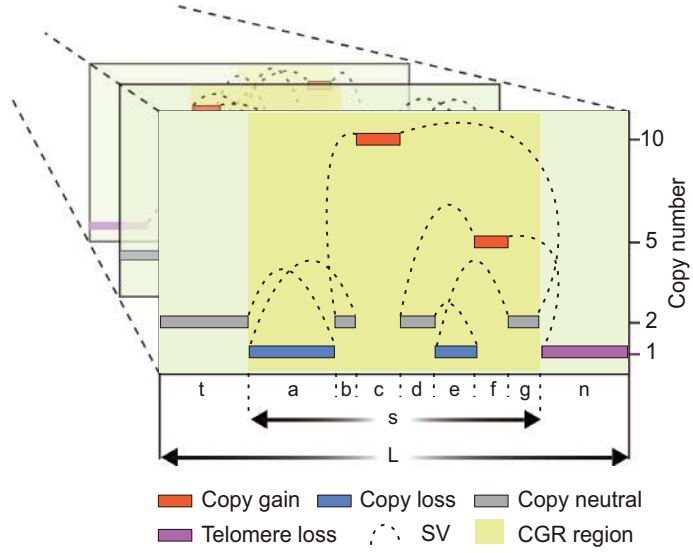


Fig. 2 Genomic features in CGR regions. Each layer represents a unique CGR event with CGR region highlighted in green. Different CN fragments are painted in different colors. Letters "*a*" to "*g*," "*n*," "*t*," "*s*," and "*L*" denote the lengths of DNA segments, which are used for feature computation

- "cluster_id": sample ID plus link_chromosome, which is the unique CGR event ID.

2.8.2 starfish_feature

This function loads CGR regions and SVs in the CGRs reported by starfish_link and then combines CNV calls and gender to construct a feature matrix for clustering and classification in Subheading 2.8.3. In Fig. 2, the size of CGR region $[s]$ equals to $\Sigma a{:}g$. Breakpoint dispersion score is defined as mean absolute deviation of $[a{:}g]$ which measures the randomness of breakpoint distribution of a CGR. Copy loss and copy gain percentages are calculated by Σa, $e/[s]$, and Σc, $f/[s]$. Telomere loss percentage is n/L. The maximum CN is 10 in this example.

Usage

```
starfish_feature(cgr, complex_sv, cnv_file, gender_file, prefix = "", genome_v = "hg19", cnv_factor = "auto", arm_del_rm = TRUE)
```

Input

- "cgr": the output of starfish_link_out$starfish_call.
- "complex_sv": the output of starfish_link_out $interleave_tra_complex_sv.
- "cnv_file": the CNV data frame defined previously.

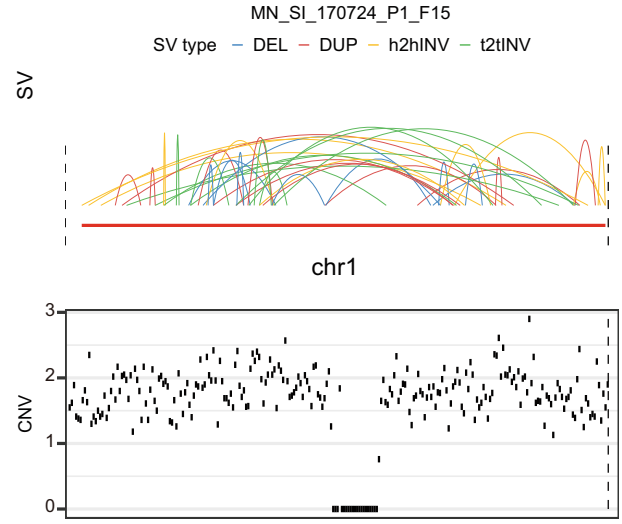


Fig. 3 SVs and CNVs identified from single-cell sequencing data with experimentally induced micronuclei. CN data from single-cell sequencing are noisy

- "gender_file": the sample gender defined previously.
- "prefix": the prefix for all intermediate files; default is none.
- "genome_v": the genome assembly used to call SV and CNV. It could be "hg19" or "hg38"; default is "hg19."
- "cnv_factor": the CN fluctuation beyond or below baseline to identify loss and gain fragments for samples with decimal CN; default is "auto," or users can provide a value between 0 and 1.
- "arm_del_rm": logical value for whether or not arm-level deletions should be removed; default is TRUE.

For each chromosome, **starfish_feature** will calculate the longest CN and set it as the chromosome CN baseline. In tumors, aneuploidy is common, so each chromosome can have a different baseline. By doing this, chromosome-specific copy loss and copy gain fragments can be identified. Sometimes, the CNV data may be noisy, such as from single-cell WGS. It would be challenging to determine the CN baselines and CNVs. For example, Fig. 3 shows a noisy CN profile from single-cell sequencing data of an experimentally induced chromothriptic cell [11]. The "cnv_factor" parameter is designed particularly for these samples. Copy loss fragments are defined as fragments with CN less than (CN baseline)*(1−cnv_factor), and copy gain fragments are those with CN greater than (CN baseline)*(1+cnv_factor). The default value is "auto," and Starfish will decide the value automatically based on the CN profile. In "auto" mode, if the CN segments all have integer CNs, the algorithm will assume the CN profile comes from bulk tumor sequencing samples with high quality CN profiles, such as

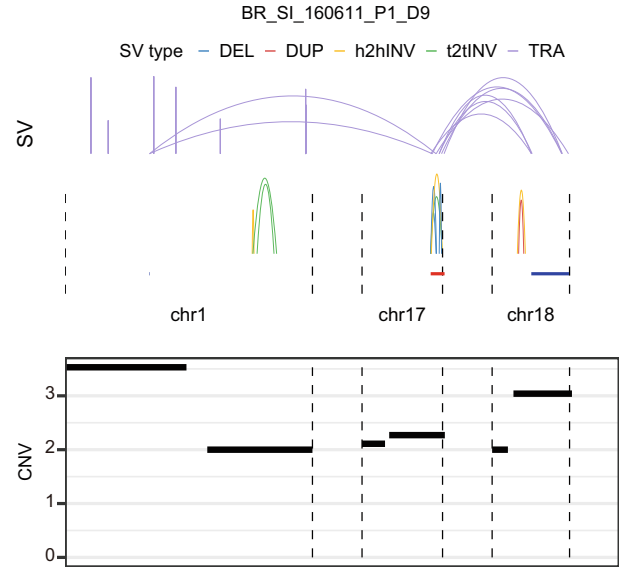


Fig. 4 SVs and CNVs identified from single-cell sequencing data with experimentally induced chromatin bridges. Induced chromatin bridge generates the telomere loss signature in the CGR region

PCAWG samples, and then cnv factor will be automatically set to 0. If the CN segments are decimal values, the algorithm will assume the CN profile is noisy, such as single-cell sequencing data in Fig. 3, and then cnv_factor will be automatically set to 0.15. The users may set this parameter manually to a value ranging from 0 to 1 based on the quality of CN profiles. For each sample without gender data, Starfish will infer the gender based on the CN baseline of chromosome X. If (1−cnv_factor)≤(CN baseline of chromosome X)≤(1+cnv_factor), the gender will be inferred as "Male"; otherwise, it will be inferred as "Female."

"arm_del_rm" controls whether arm-level deletions (at least 95% of one chromosome arm is lost) should be removed or not. To avoid the impact of aneuploidy on identifying telomere loss, arm-level deletions are removed from bulk tumor sequencing data by default, since arm-level copy losses are common in cancer and independent of CGRs. However, in single-cell sequencing of experimentally induced CGRs (Fig. 4), all alterations are generated in one cell cycle including the somatic SVs and arm-level CNVs. This function should be turned off in that case.

Example

Starfish will output the progress of CGR feature computation.

```
starfish_feature_out = starfish_feature(starfish_link_out
$starfish_call, starfish_link_out$interleave_tra_complex_sv,
```

```
example_cnv, example_sample, prefix = "example", genome_v =
"hg19", cnv_factor = "auto", arm_del_rm = TRUE)
```

```
## [1] "Starfish is computing the feature matrix, please be
    patient…"
## [1] "6% is done…"
## [1] "Event 07f16397-71bb-4594-ad4d-caa7d2baeabd_12"
## …
## [1] "62% is done…"
## [1] "Event 2b6d4d66-7f0b-4bc0-b3d6-171956a937c5_12"
## …
## [1] "100% is done…"
## [1] "Event fc8130df-897d-5404-e040-
    11ac0d485e0a_2_3_7_17"
## [1] "CGR feature computing is done!"
```

Output

The function **starfish_feature** returns an instance that contains a data frame—*cluster_feature*—which is the feature matrix. A CSV file "example_CGR_feature_matrix.csv" will be generated under the working space:

```
print(head(starfish_feature_out$cluster_feature), row.names =
FALSE)
```

sample	cluster_id	max_CNV	Loss_size _percentage	Gain_size _percentage	max_telo_loss _percentage	Brk_dispersion _MAD_mean_total
07f16397-71bb-4594-ad4d-caa7d2baeabd	07f16397-71bb-4594-ad4d-caa7d2baeabd_12	2	0.072	0	0	1.428
07f16397-71bb-4594-ad4d-caa7d2baeabd	07f16397-71bb-4594-ad4d-caa7d2baeabd_3	2	0.180	0	0	0.962
07f16397-71bb-4594-ad4d-caa7d2baeabd	07f16397-71bb-4594-ad4d-caa7d2baeabd_6	2	0.968	0	0	1.711

The columns are

- "sample": sample ID
- "cluster_id": the unique CGR event ID
- "max_CNV": maximum copy number (log scale)
- "Loss_size_percentage": copy loss percentage
- "Gain_size_percentage": copy gain percentage

- "max_telo_loss_percentage": highest telomere loss percentage
- "Brk_dispersion_MAD_mean_total": breakpoint dispersion score

2.8.3 starfish_sig

This function loads the feature matrix and performs either signature classification (classifier) or de novo signature decomposition (clustering).

Usage

```
starfish_sig(cluster_feature, prefix = "", cmethod = "class")
```

Input

- "cluster_feature": feature matrix, which is the output from starfish_feature_out$cluster_feature.
- "prefix": the prefix for intermediate files; default is none.
- "cmethod": method to infer signatures from the CGR feature matrix. It could be either "class," the preconstructed classifier of PCAWG dataset, or "cluster," the de novo unsupervised clustering. Default is "class."

Example 1

If the user selects "class" method, Starfish will run the signature classification based on a preconstructed classifier built upon 2428 tumors from PCAWG and output the progress:

```
starfish_sig_out = starfish_sig(starfish_feature_out$cluster_feature, prefix = "example", cmethod = "class")
```

```
## Using cluster_id as id variables
## [1] "Signature classification is done!"
```

Output 1

The function **starfish_sig** returns a data frame which contains the normalized CGR feature values and the CGR signature classification. A CSV file "example_pcawg_6signatures_class.csv" will be generated under the working space:

```
print(head(starfish_sig_out), row.names = FALSE)
```

cluster_id	Brk_dispersion _MAD_mean_total	Loss_size _percentage	Gain_size _percentage	log_max_CN	max_telo _loss_percentage	CGR_signature
3db6e6cc-1a06-49b9-834e-b6611cde4c4b_1_6_7_12_X	1.987	0	0.839	3.714	0	1 ecDNA/ double minutes

(continued)

cluster_id	Brk_dispersion _MAD_mean_total	Loss_size _percentage	Gain_size _percentage	log_max_CN	max_telo _loss_percentage	CGR_signature
18f5e75e-c623-11e3-bf01-24c6515278c0_4_19	0.110	0	2.699	1.794	4.054	2 BFB cycles/ chromatin bridge
07f16397-71bb-4594-ad4d-caa7d2baeabd_6	2.955	3.543	0	0.597	0	3 Large loss
07f16397-71bb-4594-ad4d-caa7d2baeabd_3	0.325	0.661	0	0.597	0	4 Micronuclei

The columns are

- "cluster_id": the unique CGR event ID
- "log_max_CN": normalized maximum copy number (log scale)
- "Loss_size_percentage": normalized copy loss percentage
- "Gain_size_percentage": normalized copy gain percentage
- "max_telo_loss_percentage": normalized highest telomere loss percentage
- "Brk_dispersion_MAD_mean_total": normalized breakpoint dispersion score
- "CGR_signature": CGR signature classification

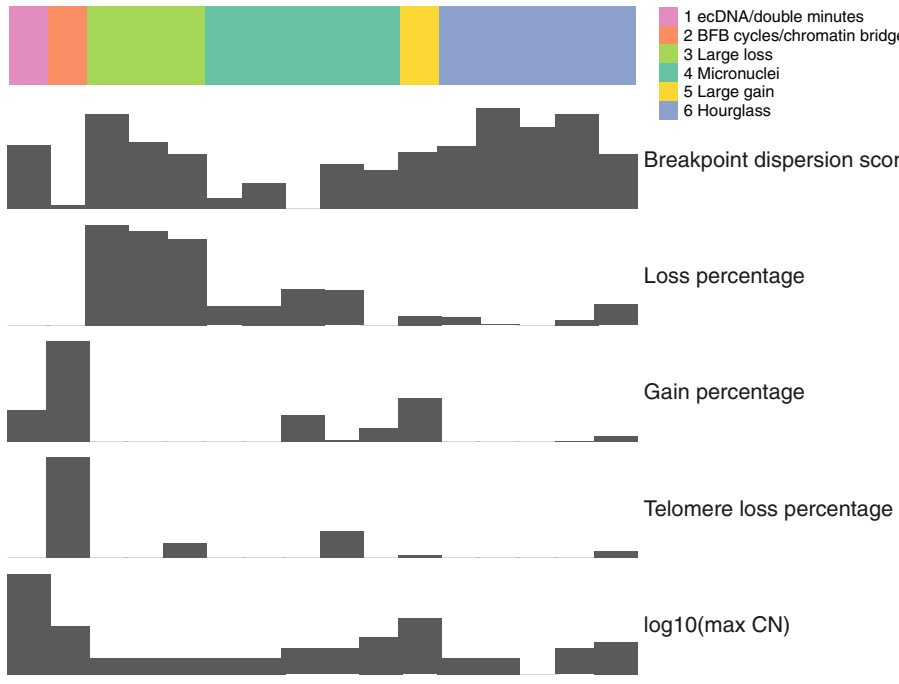


Fig. 5 Six CGR signatures with normalized feature values in the test data. Each column is a unique CGR event. Top panel shows the predicted signature of the event, and bottom panels show the normalized values of five features

A signature plot will be generated for users to quickly check the proportion of six CGR signatures in the study cohort (Fig. 5).

Example 2
If the user selects "cluster" method, Starfish will invoke R package ConsensusClusterPlus and run the unsupervised consensus clustering for de novo signature decomposition and output the progress:

```
starfish_sig(starfish_feature_out$cluster_feature, prefix =
"example", cmethod = "cluster")
```

```
##   end fraction
##   clustered
##   [1] "Clustering is done! The clustering results are stored
      under 'CGR_cluster' folder!"
```

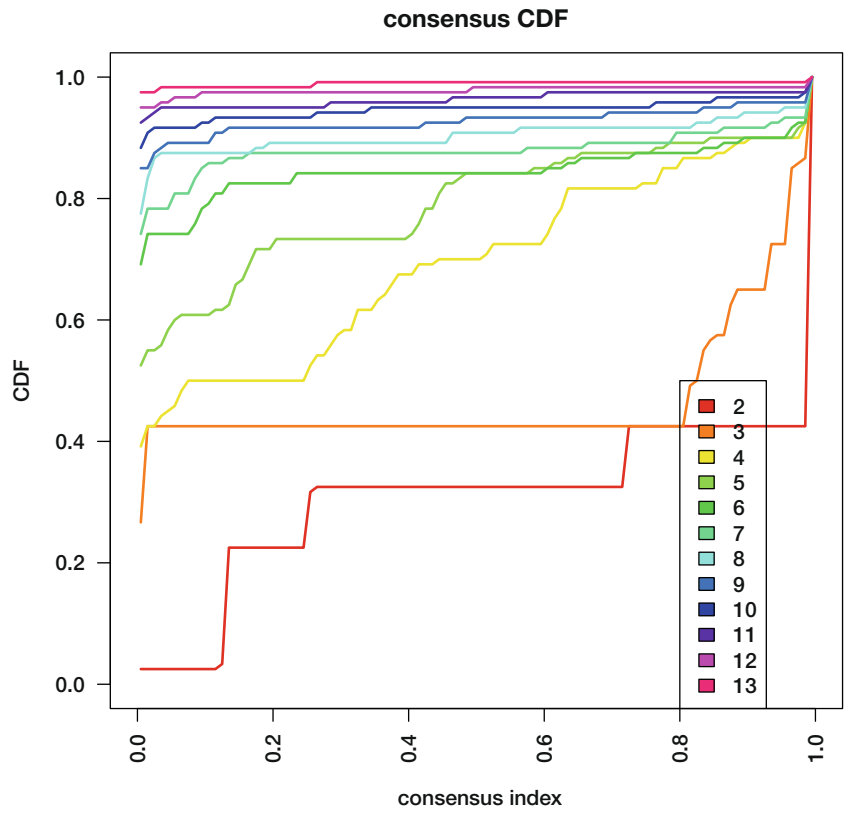


Fig. 6 CDF plot for test data. This graphic shows the cumulative distribution functions of the consensus matrix for each cluster number K (indicated by colors). This figure allows the user to determine at what number of K the CDF reaches an approximate maximum. The optimal *K* would be 6 as the curve approaches the plateau in this example

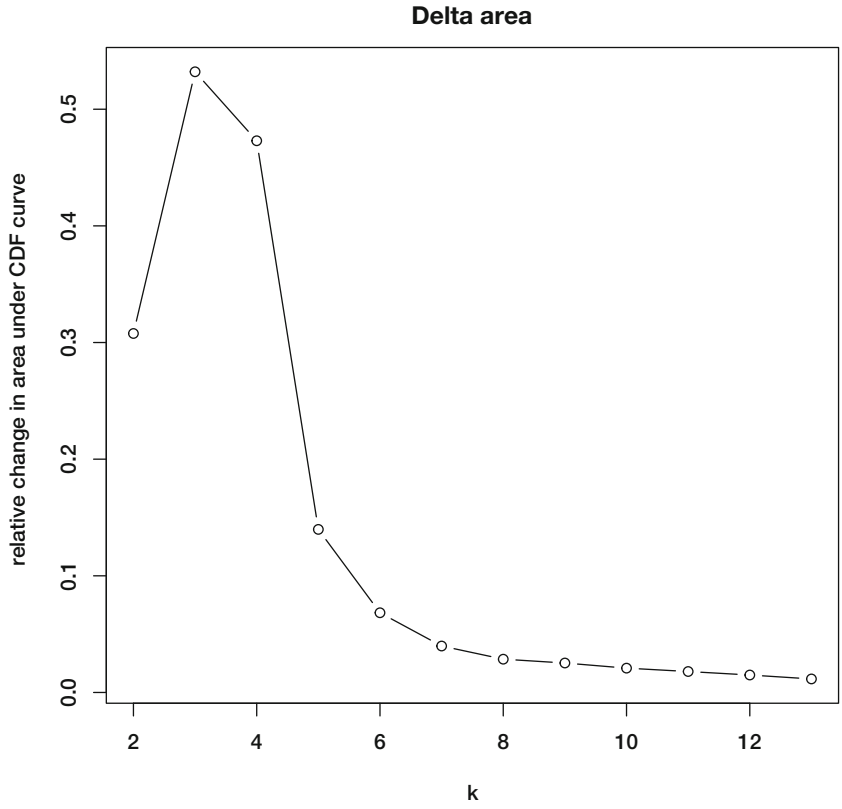


Fig. 7 Delta area plot for test data. This graphic shows the relative change in area under the CDF curve comparing cluster number K and $K - 1$. This plot allows the user to determine the relative increase in consensus and determine K at which there is no appreciable increase. Beginning from cluster 6, there is no more dramatic increase, so six clusters would be optimal in this example

Output 2

The clustering results are stored under "CGR_cluster" folder. Users could check either the "Delta Area" plot (Fig. 6) or the "Consensus Cumulative Distribution Function (CDF) Plot" (Fig. 7) to determine the optimal cluster number K:

The user can find clustering tables "CGR_cluster.k=K.consensusClass.csv" under the "CGR_cluster" folder, where different K represents different clustering numbers. Taking $K = 6$ as an example, column "V1" is the unique CGR event ID, and column "V2" is the clustering ID.

```
starfish_sig_cluster = read.csv("CGR_cluster.k=6.consensusClass.csv", header = F)
print(head(starfish_sig_cluster), row.names = FALSE)
```

V1	V2
07f16397-71bb-4594-ad4d-caa7d2baeabd_12	1

(continued)

V1	V2
07f16397-71bb-4594-ad4d-caa7d2baeabd_3	2
07f16397-71bb-4594-ad4d-caa7d2baeabd_6	3
07f16397-71bb-4594-ad4d-caa7d2baeabd_5	4
07f16397-71bb-4594-ad4d-caa7d2baeabd_7	2
07f16397-71bb-4594-ad4d-caa7d2baeabd_X	4

2.8.4 starfish_plot

This function loads an SV data frame, a CNV data frame, and CGR regions reported by starfish_link, to draw the CGR regions in a linear setting.

```
starfish_plot(sv_file, cnv_file, cgr, genome_v = "hg19")
```

Input

- "sv_file": the SV data frame defined previously.
- "cnv_file": the CNV data frame defined previously.
- "cgr": the output of starfish_link_out$starfish_call.
- "genome_v": the genome assembly version. It could be either "hg19" or "hg38," and default is "hg19."

Example

```
starfish_plot(example_sv, example_cnv, starfish_link_out
$starfish_call)
```

Output

Linear plots of CGR regions (Fig. 8)

2.8.5 starfish_all

Starfish also provides a function "starfish_all" to run all four functions described above at once.

Usage

```
starfish_all(sv_file, cnv_file, gender_file, prefix = "",
genome_v = "hg19", cnv_factor = "auto",
        arm_del_rm = TRUE, plot = TRUE, cmethod = "class")
```

Input

- "sv_file": the SV data frame defined previously.
- "cnv_file": the CNV data frame defined previously.
- "gender_file": the gender data frame defined previously.
- "prefix": the prefix for all intermediate files; default is none.

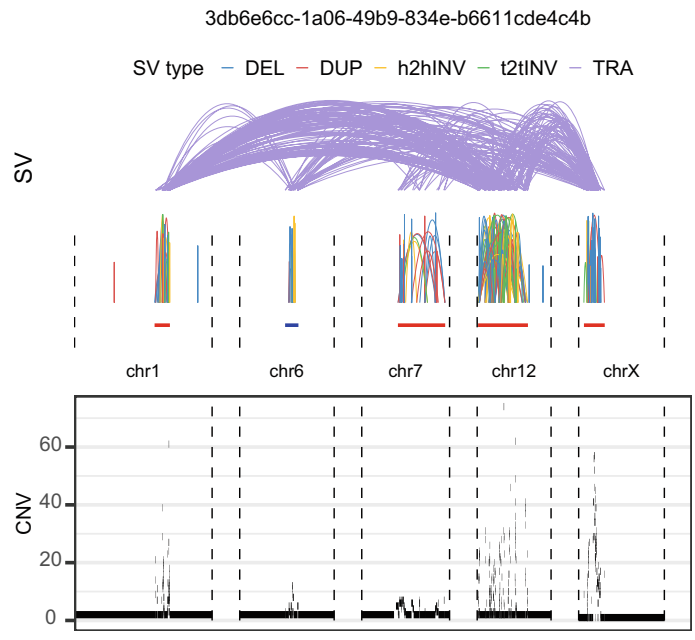


Fig. 8 Example plot of SVs and CNVs in a CGR region involving chr1, 6, 7, 12, and X in tumor sample "3db6e6cc-1a06-49b9-834e-b6611cde4c4b" with connected "seed" CGR regions shown in red and "linked" CGR regions shown in blue

- "genome_v": the genome assembly version. It could be "hg19" or "hg38"; default is "hg19."
- "cnv_factor": the CN fluctuation beyond or below baseline to identify loss and gain fragments for samples with decimal CN; default is "auto," or users can provide a value between 0 and 1.
- "arm_del_rm": logical value for whether or not arm-level deletions should be removed; default is TRUE.
- "plot": the logical value of plotting CGRs; default is TRUE.
- "cmethod": method to infer signatures from the CGR feature matrix, which can be "class" or "cluster"; default is "class."

Example

```
starfish_all(example_sv, example_cnv, example_sample, prefix
= "example", genome_v = "hg19",
        cnv_factor = "auto", arm_del_rm = TRUE, plot = TRUE,
cmethod = "class")
```

3 Notes

1. The users can load all packages at once as

```
Packages <- c("ShatterSeeky", "GenomeInfoDb", "plyr", "data.table", "GenomicRanges", "IRanges", "MASS", "ggplot2", "grid", "gridExtra", "dplyr", "ConsensusClusterPlus", "factoextra", "gplots", "ggpubr", "reshape2", "cowplot", "scales", "patchwork", "Cairo", "ggforce")
lapply(Packages, library, character.only = TRUE)
```

2. ShatterSeeky could be installed as
   git clone https://github.com/yanglab-computationalgenomics/Starfish.git
   cd Starfish
   R CMD INSTALL ShatterSeeky_0.4.tar.gz
3. Chromosome names could be either Ensembl style or UCSC style, e.g., "Chr1," "chr1," and "1" are all accepted, and only chromosomes 1–22, X, and Y are considered.
4. For bulk tumor sequencing samples, it is strongly recommended to use algorithms, such as Batternberg and Sequenza, to derive integer copy numbers, rather than using copy ratios between tumor and normal.

References

1. Negrini S, Gorgoulis VG, Halazonetis TD (2010) Genomic instability--an evolving hallmark of cancer. Nat Rev Mol Cell Biol 11(3): 220–228. https://doi.org/10.1038/nrm2858
2. Yang L, Luquette LJ, Gehlenborg N et al (2013) Diverse mechanisms of somatic structural variations in human cancer genomes. Cell 153(4):919–929. https://doi.org/10.1016/j.cell.2013.04.010
3. The ICGC/TCGA Pan-Cancer Analysis of Whole Genomes Consortium (2020) Pan-cancer analysis of whole genomes. Nature 578(7793):82–93. https://doi.org/10.1038/s41586-020-1969-6
4. Cortes-Ciriano I, Lee JJ, Xi R et al (2020) Comprehensive analysis of chromothripsis in 2,658 human cancers using whole-genome sequencing. Nat Genet 52(3):331–341. https://doi.org/10.1038/s41588-019-0576-7
5. Stephens PJ, Greenman CD, Fu B et al (2011) Massive genomic rearrangement acquired in a single catastrophic event during cancer development. Cell 144(1):27–40. https://doi.org/10.1016/j.cell.2010.11.055
6. Zhang CZ, Spektor A, Cornils H et al (2015) Chromothripsis from DNA damage in micronuclei. Nature 522(7555):179–184. https://doi.org/10.1038/nature14493
7. Maciejowski J, Li Y, Bosco N et al (2015) Chromothripsis and kataegis induced by telomere crisis. Cell 163(7):1641–1654. https://doi.org/10.1016/j.cell.2015.11.054
8. Maciejowski J, Chatzipli A, Dananberg A et al (2020) APOBEC3-dependent kataegis and TREX1-driven chromothripsis during telomere crisis. Nat Genet 52(9):884–890. https://doi.org/10.1038/s41588-020-0667-5
9. Shoshani O, Brunner SF, Yaeger R et al (2021) Chromothripsis drives the evolution of gene amplification in cancer. Nature 591(7848): 137–141. https://doi.org/10.1038/s41586-020-03064-z

10. Bao L, Zhong X, Yang Y et al (2022) Starfish infers signatures of complex genomic rearrangements across human cancers. Nat Cancer 3(10):1247–1259. https://doi.org/10.1038/s43018-022-00404-y

11. Umbreit NT, Zhang CZ, Lynch LD et al (2020) Mechanisms generating cancer genome complexity from a single cell division error. Science 368(6488):eaba0712. https://doi.org/10.1126/science.aba0712

Chapter 6

Using FFPEsig to Remove Formalin-Induced Artifacts and Characterize Mutational Signatures in Cancer

Qingli Guo, Ann-Marie Baker, Ville Mustonen, and Trevor A. Graham

Abstract

The wealth of routinely processed formalin-fixed and paraffin-embedded (FFPE) cancer biopsies is potentially a tremendous resource for cancer genomics research. However, the presence of formalin-induced artifactual mutations in FFPE material can confound mutational analyses. Our de-noising algorithm, FFPEsig, removes FFPE-related artifactual mutations enabling the inference of biological mutational signatures. In this chapter, we focus on the practical use of FFPEsig, offering a detailed guidance from both the wet-lab experimental and bioinformatics analysis perspectives. Our aim is to assist users to generate robust and significant results using FFPEsig.

Key words FFPE samples, Signature analysis, Mutational processes, De-noising, Biological mutation signal

1 Introduction

Characterizing mutational signatures in cancer FFPE (formalin-fixed paraffin-embedded) specimens can help us to understand cancer genome evolution [1–3]. However, the use of formalin in fixing tissues can result in significant DNA damage, leading to not only a reduced quantity but also a diminished quality in resulting DNA sequencing libraries [4, 5]. In our earlier research, we introduced FFPEsig, a comprehensive computational tool designed to eliminate noise and unveil genuine biological signals within DNA sequencing data derived from FFPE blocks [6]. Our de-noising method assumes that formalin induces a characteristic pattern of "mutational noise" that will be consistent across FFPE samples; therefore, removal of this predictable pattern from the observed mutational profiles is plausible.

We discovered that the mutational spectrum of formalin-induced artifactual mutations is highly similar to that of two biological mutational processes—C > T mutations at CpG sites

Alexander Krasnitz and Pascal Belleau (eds.), *Cancer Bioinformatics*, Methods in Molecular Biology, vol. 2932,
https://doi.org/10.1007/978-1-0716-4566-6_6,

that occur as part of aging [7] and C > T mutations that occur because of a specific base excision repair deficiency that results from *NTHL1* mutations [8]. These two biological processes produce patterns of mutations highly similar to formalin-induced signatures in scenarios where sequencing library preparation is performed with and without a chemical "FFPE repair" agent, respectively [6]. To subtract FFPE generated noise, FFPEsig takes the observed mutation profile and the known noise mutational signature as input, subtracts the noise from the observed profile, and outputs the predicted biological mutational profile for the given sample.

Our previous results revealed two main factors that can impact the correction performance of FFPEsig: signal-to-noise ratio (SNR) (the true mutational load vs the formalin-induced artifacts) and signal-to-noise similarity (SNS) (the similarity of the mutational signatures produced by biological processes compared to that of formalin-induced mutagenesis) [6].

While we briefly addressed the implementation of this knowledge in the analysis of mutational signatures in FFPE samples, this chapter provides a more comprehensive usage manual. We focus on the practical solutions and suggestions derived from our own research. Additionally, we have incorporated notes covering frequently asked questions and areas that users commonly overlook.

2 User Manual

In this section, we first summarize the major steps of using FFPEsig, including the recommended wet-lab protocols. Next, we provide a detailed description for each of the components of the bioinformatics analysis. Finally, we focus on the practical usage of FFPEsig with additional notes.

2.1 Overview

Here we provide an overview about the major steps of applying FFPEsig (Fig. 1), including the wet-lab "FFPE repair" treatment (Step 1), downstream bioinformatics analysis (Step 2), and a final noise correction using FFPEsig (Step 3). We included the first two steps (wet lab and mutational calling) in our guidance as they directly impact the two main factors, SNS and SNR, which jointly determine the success of noise removal by FFPEsig.

In Step 1, the decision-making process involves determining whether the target FFPE sample(s) should undergo repair using a chemical agent (e.g., uracil DNA glycosylase, UDG) or be left unrepaired during the DNA extraction process. This choice results in the generation of distinct formalin-induced mutational signature "error profiles", which can further add complexity to the bioinformatics process if the error profile is very similar to the dominant true biological mutational processes expected to be active in the sample.

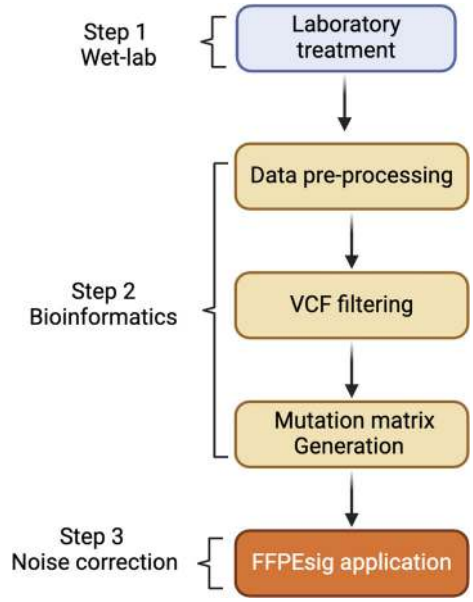


Fig. 1 Overview of the major steps of using FFPEsig

For instance, colorectal cancers (CRCs) often exhibit a prominent aging signature (SBS1, COSMIC version 3). If the CRC FFPE DNA has undergone FFPE repair, the artifactual mutations would also display an SBS1-like FFPE noise pattern, leading to a high SNS [6]. In such a scenario, FFPEsig will likely "overcorrect" the noise (i.e., erroneously remove true biological signal) because it is challenging to differentiate the highly similar authentic signals from the artifactual ones. Therefore, we strongly recommend using unrepaired FFPE DNA for mutational signature analysis with FFPEsig. The following subsection 2.2 outlines how prior knowledge of a given cancer type can guide this decision-making process.

Step 2 represents a bioinformatics analysis pipeline for calling somatic mutations and generating mutational profiles from the mutation list. The incorporation of artifact-filtering methods is crucial at this stage to ensure a reasonable SNR for FFPEsig to operate effectively. In our observations, FFPEsig demonstrates effective performance when the SNR is greater than 0.1 [6]. Nevertheless, its efficacy declines rapidly when the true mutation count is less than 10% of the noise count, which is due to the stochastic variability of the noise outweighing the signal itself. Therefore, the upstream artifact-filtering steps are crucial for addressing easily removable FFPE artifacts and maximizing SNR values. To achieve this goal, we offer additional suggestions in the following sections.

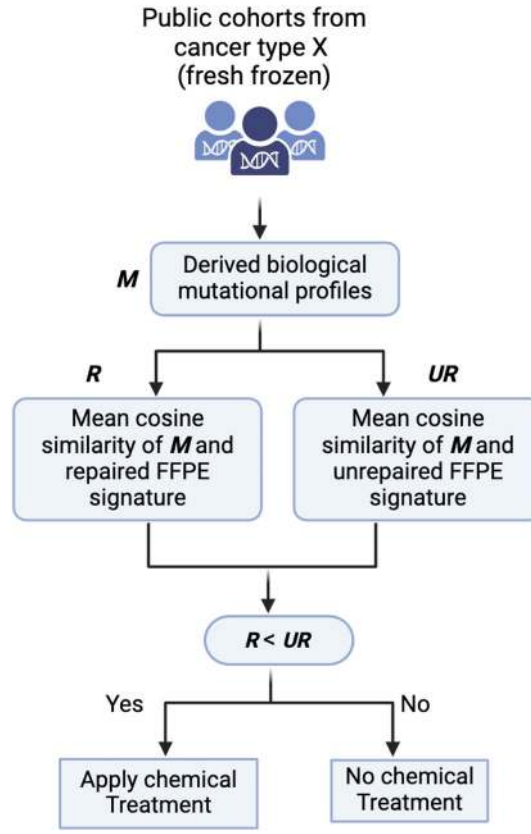


Fig. 2 Using public datasets to determine suitable laboratory treatment of FFPE samples based on SNS. (This figure is adapted from Guo et al. [6])

2.2 Defining the Suitable Laboratory Protocols

To decide whether chemical FFPE repair treatment should be applied to the target FFPE DNA sample Υ of cancer type X, FFPEsig users can apply the existing knowledge by examining high quality tumor sequencing data from the same cancer type X (Fig. 2), e.g., from the pan-cancer data cohort [9]. In this context, we assume that the biological mutation profile of target FFPE sample Υ is likely to align with publicly available mutational spectrum of the same cancer type.

For example, users can compare the averaged SNS values, R and NR, calculated between the biological mutation patterns (M) of fresh-frozen tumors of cancer type X and the two established repaired and unrepaired FFPE signatures, respectively. The UR value represents the mean pair-wise cosine similarities between biological profiles and the unrepaired signature, with R representing the same value computed against the repaired signature. If the biological mutation patterns share a higher similarity with the unrepaired signature ($R < UR$), we recommend applying UDG treatment to repair the FFPE DNA extracted from sample Υ. Otherwise, if the similarity is greater with the repaired signature ($R > UR$), FFPE DNA sample Υ should remain unrepaired.

Table 1
Recommended application of FFPEsig on common cancer type

Group	Common cancer types	Recommendation (Step 1)
Confident	Lung-SCC	Repair
	Bladder-TCC	
	Lung-AdenoCA	
	Head-SCC	
	Skin-Melanoma	
	Eso-AdenoCA	Unrepair
	ColoRect-AdenoCA	
	Stomach-AdenoCA	
	CNS-GBM	
Less confident (proceed with care)	Liver-HCC	Repair
	Biliary-AdenoCA	
	Ovary-AdenoCA	
	Breast-AdenoCA	
	Kidney-RCC	
	Bone-Osteosarc	
	Uterus-AdenoCA	Unrepair
	Panc-AdenoCA	
	Lymph-BNHL	
	Prost-AdenoCA	
Not confident	Lymph-CLL	NA
	Kidney-ChRCC	
	Panc-Endocrine	
	Myeloid-MPN	
	Thy-AdenoCA	
	CNS-Medullo	
	CNS-PiloAstro	

This table is adapted from Guo et al. [6]

We applied the above principle to sequencing data from fresh-frozen tumors in Pan-Cancer Analysis of Whole Genomes (PCAWG) [9] and summarized the recommended protocol for the common cancer types reported in PCAWG (Table 1). However, users are encouraged to apply these principles using additional public datasets as these become available.

2.3 Generating FFPE Mutational Profile with Reasonable SNR

Upon obtaining the raw sequencing data from the target FFPE cancer sample Υ along with its matched normal DNA data, the initial step involves processing the sample pair to call somatic mutations. It is crucial to note that the removal of FFPE noise mutations also carries a risk of excluding true somatic mutations. Therefore, achieving a balanced mutation list with a reasonable SNR is essential in Step 2.

In our laboratory, we call SNVs using the mutation callers, such as Mutect2 [10] and Platypus [11]. Additional filters are then applied, including (1) checking if the FILTER flag was marked as

PASS or other acceptable filters (alleleBias, Q20, QD, SC, HapScore); (2) ensuring the variant is not a known germline variant; (3) confirming a genotype is called for all samples and the genotype Phred score is 10 or higher in all samples; and (4) verifying that the normal sample has no reads containing the variant and at least three or more reads support the variant in a tumor sample. For a more detailed overview, please refer to the relevant Method section in our original paper [6]. Finally, SigProfilerMatrixGenerator is used to derive mutational profiles [12].

Following the mutation calling and filtering steps, we can once again use available cancer profiles from existing fresh-frozen tumors to estimate the SNR for sample Υ (Fig. 3). To obtain this value, we calculate the number of point mutations in the resulting FFPE mutational profile, denoted as symbol O. We aggregate the averaged mutation load of non-hypermutated tumors in cancer type X (with a mutation load $<5 \times 10^4$) (*see* **Note 1**), denoted as symbol A. The approximate estimation of SNR is then calculated by computing $A/(O\text{-}A)$. If the estimated SNR is greater than 0.1, the FFPE mutation profile and the known FFPE repair status (from Step 1) are ready for use in FFPEsig for noise correction in Step 3. However, if the estimated SNR falls below 0.1, additional mutation filters or adjustments to the threshold for the current filters (in Step 2) may be necessary to regenerate the FFPE mutation profile.

In our previous analysis, we observed that certain cancer types exhibit a relatively lower mutational load, making it challenging to adjust the bioinformatics pipeline to achieve a reasonable SNR (Table 1). Consequently, we do not recommend applying FFPEsig to cancer samples collected from these tissues.

2.4 Applying FFPEsig

FFPEsig is a standalone command-line tool and implemented in python (version >3). The following code can be used to run the tool:

```
python FFPEsig.py -i [input file] -s [sample ID] -l [repair mode] -o [output file]
```

In the above code, the script "FFPEsig.py" is available to download from here. In addition, four arguments are required to run the tool:

1. *--input/-i <string>*, refers to the input file containing the observed mutation counts in 96-channel format for one or more FFPE samples (*see* **Note 2**).
2. *--sample/-s <string>*, refers to a sample ID contained in the input file. This sample ID identifies which sample to process further, and it also serves as the file identifier in the output folder (*see* **Note 3**). Running multiple samples simultaneously is also possible (*see* **Note 4**).

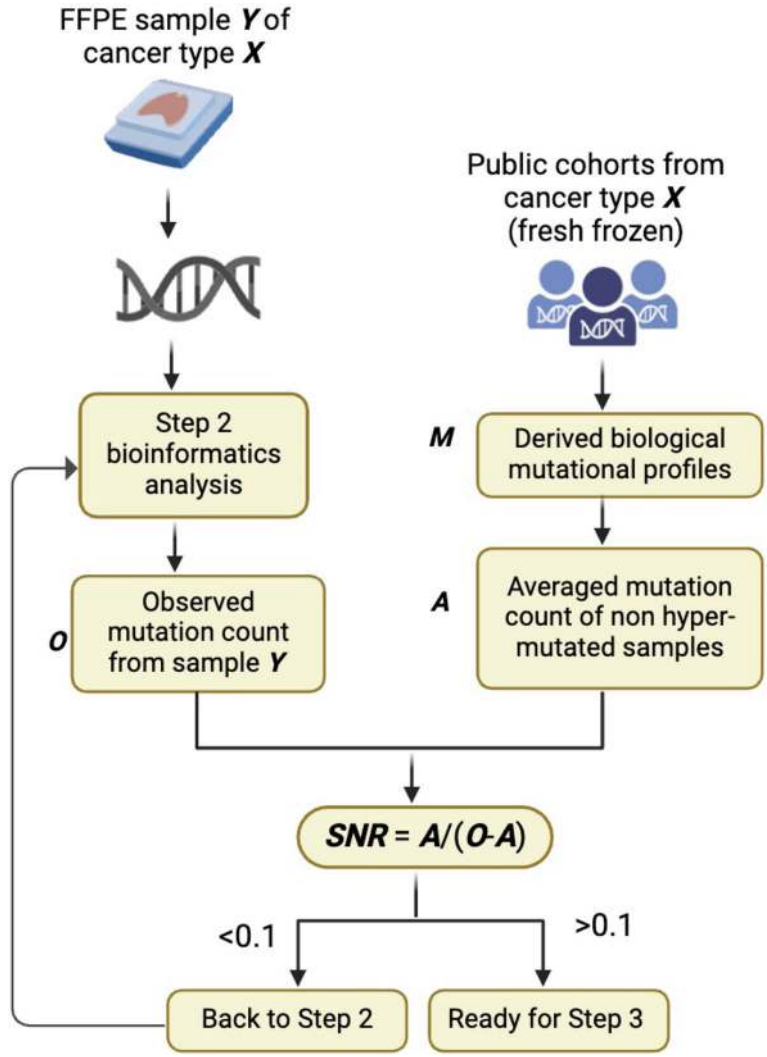


Fig. 3 Adjusting bioinformatics analysis pipeline to meet the minimal SNR requirement using public datasets. (This figure is adapted from Guo et al. [6])

3. *--label/-l <string>*, refers to the label of the repair status of the sample. The label can be either "Repaired" or "Unrepaired." When the "Repaired" label is specified, the repaired FFPE noise signature (similar to SBS1) is used as the noise mutation pattern. On the other hand, when the "Unrepaired" label is provided, the unrepaired FFPE noise signature (similar to SBS30) is utilized by FFPEsig. It is crucial to clearly specify the label to avoid error messages. Additionally, users have the option to update the error profiles (*see* **Note 5**) or repurpose the tool to remove noise in a different setting (*see* **Note 6**).
4. *--output/-o <string>*: provides the path to the folder where all the output files will be stored. This folder must be created ("mkdir path-to-the-folder") before running FFPEsig if it does not exist.

FFPEsig generates four files in the output folder when provided with appropriate input arguments. The output files include the following:

1. *'SampleID'_corrected_profile.csv*, contains the final corrected mutation profile, which can be further used for the decomposition of individual mutational signatures (*see* **Note 7**).
2. *'SampleID'_all_solutions.csv*, contains all solutions derived from using different random initial values (*see* **Note 3**). This file is helpful to help the users to estimate the variances among all predicted values, if interested.
3. *'SampleID'_before_correction.pdf*, visualizes original mutational profile of the given sample over the 96-mutational channels. Here, we provide a representative example of an uncorrected profile of a simulated FFPE sample (Fig. 4, top panel).
4. *'SampleID'_after_correction.pdf*, visualizes the corrected mutational profile of the given sample over the 96-mutational channels. Again, we show an example of a corrected profile from a simulated FFPE sample (Fig. 4, bottom panel).

3 Possible Batch Effect and Strategies

In our previous article, we have observed an excess number of artifactual T > C mutations in FFPE samples in study 1 [13], but not in study 2 [5] and 3 [14]. Additionally, we reviewed an

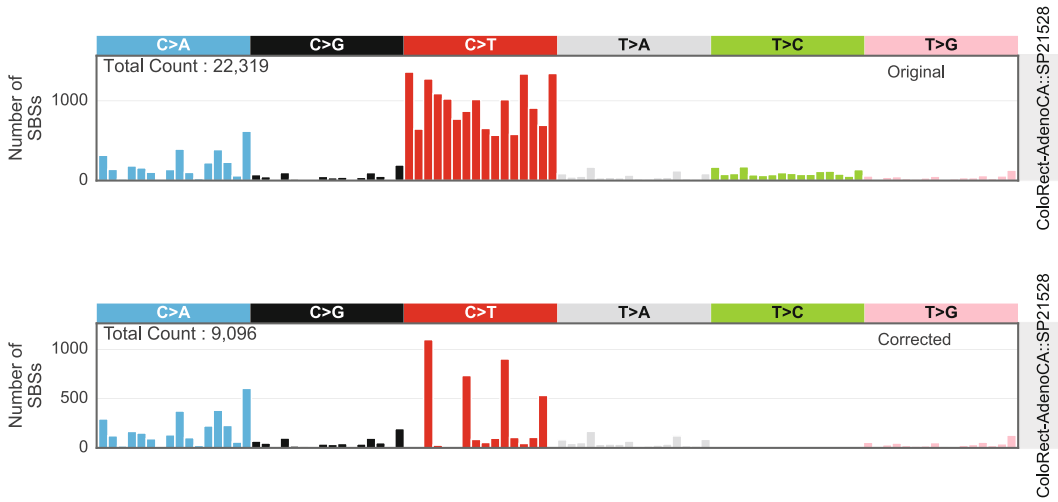


Fig. 4 Examples of output plots from FFPEsig. In the output folder, users can find two figures generated by FFPEsig for a given sample. The top panel displays the 96-mutational channel plot of the original mutation count, while the bottom panel shows the same plot of the predicted biological mutational profile for the same sample. The total mutation count, sample ID, and correction status are annotated in the plots

additional 20 publications regarding this issue, and only 3 of 20 articles (15%) observed T > C mutations from their FFPE samples [6]. These collective findings suggest that these T > C mutations are likely batch-related artifacts produced by lab-specific DNA extraction or library generation protocols and/or sequencing and basic processing of these libraries.

Our previous investigation suggested that the choice of polymerase used in PCR (and reaction conditions) might be responsible for the batch-related artifacts. DNA polymerases have different levels of fidelity and bypass efficiency, also known as translesion synthesis, leading to polymerase associated artifacts [15, 16]. For example, T:G mispair is one of the most frequently produced and most easily extended base substitution errors for *Taq* DNA polymerase, which will lead to T > C accounting for 67% of the artifacts [17]. The study also found when dNTP concentration is lowered from 800 to 6 μm, only T:G mismatches or perfect base pairs are extended [18]. Further, Υ family bypass DNA polymerase would cause a great number of A:T > G:C artifacts as the misincorporation of dGTP opposite of T is even more efficient than inserting dATP in this family [16].

In situations where there is an abundance of T > C mutations, biological interpretation of these mutations should be approached with extra care. In our current signature analysis, we treated T > C mutation counts as missing data and assigned them zero values. Our analysis revealed that the remaining 80 channels (non-T > C) still retained sufficient information for robust signature decomposition. As more samples are sequenced, it will become feasible to replace these missing data with corrected values. For instructions on updating this information in the error profile, users can refer to **Notes 5** and **6** in this chapter.

4 Notes

1. FFPEsig is designed to filter the mutational spectra of samples with an overwhelming amount of noise that obscures the genuine biological signal. However, if the volume of the true signal is sufficiently strong to the extent that the FFPE-related noise is no longer the dominant component, FFPEsig may not be necessary. For instance, we observed that FFPEsig consistently performs well on hypermutated tumors, irrespective of the noise level. This is likely because the dominant biological mutational process can be reliably identified without the need for FFPE artifact removal.
2. The mutation count in the 96-mutational channel format can be obtained using installed methods like SigProfilerMatrixGenerator [12]. It is important to note that the input data must be

in the same order as presented in our signature file—because FFPEsig does not read the row names of the input file, which may be displayed in different formats by different matrix generation tools.

3. The input file should be a standardized CSV file (comma-separated) with the 96-channel mutational counts displayed in the columns, where the column names represent the sample IDs. While it can contain more than one sample, only the specified sample ID will be processed by using the command line. For instructions on running multiple samples automatically, please refer to **Note 4**.
4. FFPEsig focuses on one sample at a time. For each given sample, the pipeline runs the correction algorithm 100 times using different random initial values and collects predicted mutational profiles from each interaction. The final predicted profile takes the median values from each mutational channel over the 100 candidate solutions.

   To run multiple samples, the users can utilize the following codes in a shell script (see also the further discussion in our tool github page):

   for sample in "cat sample_names.txt" `do`

   ```
   python FFPEsig.py --input <Path-to-the-DataFrame> --sample $sample --label <Unrepaired|Repaired> --output <Path-of-output-folder>
   Done
   ```

   In the above code, *sample_names.txt* refers to a file containing all the targeted sample IDs. Each line within this file should contain one sample ID.
5. In the current version of FFPEsig, the formalin-induced FFPE signatures were derived from targeted panel sequencing data and then projected into the genome scale. However, we acknowledge that confounders can arise at both wet-lab and dry-lab steps, leading to some variability in error profiles. In such cases, users may consider using their own error profiles from benchmarked pilot studies. To update or replace the current error profiles, users can modify the relevant mutational frequency values in the instances named "ffpe_sig_repaired" or "ffpe_sig_unrepaired" in our FFPEsig.py script. Alternatively, you can provide the noise mutation profile directly to the "W1" argument in the function named "correct_FFPE_profile." For more details, please refer to our analysis notebook.
6. FFPEsig can be repurposed to correct noise in other settings where the dominant noise obscures the signal. In such cases, users can update the new error profile, which could also be generated from other types of mutations, such as double base

substitutions and small insertions and deletions, not necessarily limited to SBS mutations. In this scenario, the visualization codes, designed for SBS mutations, should be muted to avoid error messages.

7. The corrected mutation profile by FFPEsig is considered a combined biological mutational profile, representing a linear combination of various biological mutational processes. To explore the active mutational processes in the given sample, a signature refitting analysis is required. Our GitHub page includes a function named "sig_refitting" specifically designed for this task. Users can easily utilize it by downloading the setup.py. Detailed instructions for running this analysis can be found on our analysis codes page.

Acknowledgments

This article was supported by funding from the Schottlander Research Charitable Trust (to QG and TG) and Cancer Research UK (DRCNPG-May21_100001 to TG). V.M. acknowledges funding from the Academy of Finland (345829).

Conflict of Interest

TG is named as a coinventor on patent applications that describe a method for TCR sequencing (GB2305655.9, also with AMB), a method to measure evolutionary dynamics in cancers using DNA methylation (GB2317139.0), and a method to infer drug resistance mechanisms from barcoding data (GB2501439.0). TG has received honorarium from Genentech and consultancy fees from DAiNA therapeutics.

References

1. Alexandrov LB, Nik-Zainal S, Wedge DC et al (2013) Signatures of mutational processes in human cancer. Nature 500:415–421
2. Van Hoeck A, Tjoonk NH, van Boxtel R et al (2019) Portrait of a cancer: mutational signature analyses for cancer diagnostics. BMC Cancer 19:457
3. Koh G, Degasperi A, Zou X et al (2021) Mutational signatures: emerging concepts, caveats and clinical applications. Nat Rev Cancer 21: 619–637
4. Wong SQ, Li J, Tan AYC et al (2014) Sequence artefacts in a prospective series of formalin-fixed tumours tested for mutations in hotspot regions by massively parallel sequencing. BMC Med Genet 7:23
5. Bhagwate AV, Liu Y, Winham SJ et al (2019) Bioinformatics and DNA-extraction strategies to reliably detect genetic variants from FFPE breast tissue samples. BMC Genomics 20:689
6. Guo Q, Lakatos E, Bakir IA et al (2022) The mutational signatures of formalin fixation on the human genome. Nat Commun 13:4487
7. Alexandrov LB, Jones PH, Wedge DC et al (2015) Clock-like mutational processes in human somatic cells. Nat Genet 47:1402–1407
8. Drost J, van Boxtel R, Blokzijl F et al (2017) Use of CRISPR-modified human stem cell organoids to study the origin of mutational signatures in cancer. Science 358:234–238

9. Alexandrov LB, Kim J, Haradhvala NJ et al (2020) The repertoire of mutational signatures in human cancer. Nature 578:94–101
10. Van der Auwera GA, Carneiro MO, Hartl C et al (2013) From FastQ data to high confidence variant calls: the Genome Analysis Toolkit best practices pipeline. Curr Protoc Bioinformatics 43:11.10.1–11.10.33
11. Rimmer A, Phan H, Mathieson I et al (2014) Integrating mapping-, assembly- and haplotype-based approaches for calling variants in clinical sequencing applications. Nat Genet 46:912–918
12. Bergstrom EN, Huang MN, Mahto U et al (2019) SigProfilerMatrixGenerator: a tool for visualizing and exploring patterns of small mutational events. BMC Genomics 20:685
13. Prentice LM, Miller RR, Knaggs J et al (2018) Formalin fixation increases deamination mutation signature but should not lead to false positive mutations in clinical practice. PLoS One 13:e0196434
14. Van Allen EM, Wagle N, Stojanov P et al (2014) Whole-exome sequencing and clinical interpretation of formalin-fixed, paraffin-embedded tumor samples to guide precision cancer medicine. Nat Med 20:682–688
15. Quach N, Goodman MF, Shibata D (2004) In vitro mutation artifacts after formalin fixation and error prone translesion synthesis during PCR. BMC Clin Pathol 4:1
16. Kunkel TA (2004) DNA replication fidelity. J Biol Chem 279:16895–16898
17. McInerney P, Adams P, Hadi MZ (2014) Error rate comparison during polymerase chain reaction by DNA polymerase. Mol Biol Int 2014: 287430
18. Eckert KA, Kunkel TA (1991) DNA polymerase fidelity and the polymerase chain reaction. PCR Methods Appl 1:17–24

Chapter 7

Inferring Phenotypes of Copy Number Clones in Cancer Populations Using TreeAlign

Hongyu Shi, Matthew Zatzman, Sohrab Shah, and Andrew McPherson

Abstract

Somatic copy number changes modify gene expression and drive cancer development and progression. Single-cell techniques now allow for the profiling of both gene expression and copy number, opening the possibility of linking expression changes with copy number changes at a single-cell level. However, joint measurement of both expression and copy number from the same cell is not commonplace, and thus joint analysis of expression and copy number requires computational integration of the two modalities. TreeAlign is a method for matching cells in single-cell RNA (scRNA) data to clones inferred from single-cell whole genome sequence (scWGS) data. TreeAlign is phylogeny aware and capable of robustly modeling the effect of gene dosage on gene expression. In this chapter, we provide a practical guide for using TreeAlign to jointly analyze copy number and gene expression from single-cell whole genome sequencing and single-cell RNA sequencing datasets.

Key words Cancer genomics, Tumor evolution, Single-cell RNA sequencing, Single-cell whole genome sequencing, Probabilistic modeling, Tumor microenvironment

1 Introduction

Copy number alterations (CNAs) are frequent events in cancer and are known to contribute to transcriptional diversity among cancer cells. Amplification of oncogenes and deletion of tumor suppressors can lead to dysregulated expression of affected genes and change the fitness landscape of cancer cells [1]. CNAs spanning larger genomic regions of chromosomal arms or whole chromosomes can impact the expression of hundreds of genes through copy number (CN) dosage effects. CN dosage effects are defined as the positive correlation between CN (or gene dosage) and the corresponding gene expression [2]. Previous studies using bulk sequencing techniques have investigated the association between clonal CNAs and gene expression [3–5]. The expression level of a gene can be influenced by copy number dosage effects reflected by the significant positive correlation between gene expression and the

Alexander Krasnitz and Pascal Belleau (eds.), *Cancer Bioinformatics*, Methods in Molecular Biology, vol. 2932,
https://doi.org/10.1007/978-1-0716-4566-6_7,

underlying copy number (CN) [2]. However, gene dosage effects are not deterministic and may be subject to compensatory mechanisms, rendering the impact of CNAs on expression as highly variable across the genome. Transcriptional adaptive mechanisms [6] including epigenetic modifications and downstream transcriptional regulation can modulate CN dosage effects [7–9], further obscuring the direct impact of gene dosage. It remains an open question to investigate how CNAs impact gene expression through both dosage-dependent and dosage-independent mechanisms and how dosage effects contribute to intratumor heterogeneity and clonal evolution in cancer.

Establishing a robust connection between genetic compositions and cancer cell phenotypes is challenging. Conventional bulk sequencing methods have been widely employed to delineate somatic alterations and concomitant phenotypic modifications [3–5, 10]. Although paired datasets with both DNA and RNA sequencing make it possible to correlate these two aspects, the resolution of these studies is still limited at the level of patient-derived tumor samples and unable to provide a comprehensive view of genetic and transcriptomic diversity at the subclonal or single-cell level. Conducting single-cell RNA and DNA sequencing separately offers the capacity to profile a large number of individual cells and thereby provides a more comprehensive depiction of cell populations in tumors. In recent years, an increasing number of studies have appeared, generating multimodal datasets with single-cell DNA and single-cell RNA profiles [1, 11–13]. The measurements of both genetic alterations and gene expression at single-cell level allow us to further dissect different aspects of intratumor heterogeneity and understand the role of CNAs in subclonal phenotypic divergence. However, a comprehensive understanding of the intricate interplay between subclonal CNAs and phenotypic changes requires the development of computational frameworks for integrating these diverse data modalities.

To quantify how CNAs influence gene expression at a subclone level, we developed TreeAlign [14], which computationally integrates independently sampled single-cell DNA and RNA sequencing data from the same cell population and explicitly models gene dosage effects from subclonal alterations (Fig. 1). TreeAlign implements a Bayesian probabilistic model to assign single-cell expression profiles to a scWGS-based single-cell phylogeny while inferring CN dosage effects. To further improve the accuracy of clone assignment and dosage effect prediction, TreeAlign also allows explicit modeling of allele-specific expression from allelic copy number imbalance. The software for TreeAlign (https://github.com/shahcompbio/TreeAlign) is implemented in Python using Pyro [15] and is publicly available. The principles and benchmarking of TreeAlign have been previously described in our publication. This

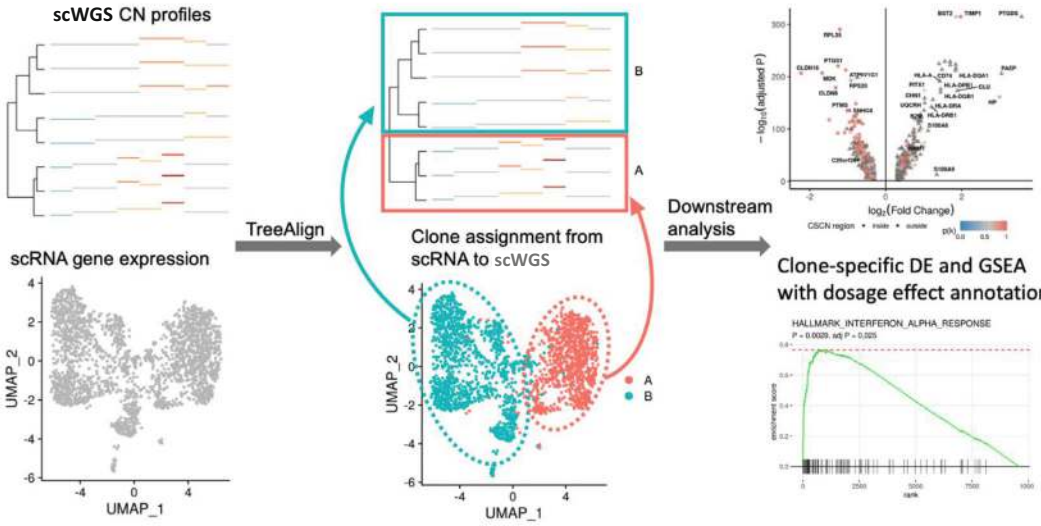


Fig. 1 A TreeAlign analysis integrates scWGS CN profiles (top left) and scRNA gene expression (bottom left) resulting in an assignment of scRNA cells to scWGS clones (middle) allowing downstream differential expression and gene set enrichment analysis

chapter focuses on a practical guide to apply TreeAlign on single-cell sequencing datasets.

2 Methods

In this section, we outline the steps to preprocess both scRNA and scWGS from raw reads to input tables for TreeAlign. We describe multiple options for some steps depending on the scWGS or scRNA platform used to generate the data or the preference of the analyst. Quality control (QC) steps are described to allow for checks on data quality and amount of relevant signal. We then describe how to run the TreeAlign model and interpret the outputs generated by the software.

TreeAlign can be run using either clones or a phylogeny as input. In addition, the model can be run in either total copy number or allele-specific mode, meaning there are four possible operating modes overall. We summarize the required inputs based on the operating mode below.

Universal inputs

- Total copy number per scWGS cell.
- Gene expression per scRNA cell.

Clonal input

- Assignments of scWGS cells to clusters representing clones.

Phylogeny input

- A phylogeny relating scWGS cells.

Total copy number mode

- No additional inputs required.

Allele-specific copy number mode

- Allele-specific copy number per heterozygous SNP per scWGS cell.
- Read counts for heterozygous SNPs in scRNA cells.

2.1 Preprocessing

The complete preprocessing pipeline involves the following steps:

- scWGS preprocessing,
  - Alignment and QC.
  - Compute cell specific total copy number (required).
  - Compute cell and allele-specific copy number (optional, allele-specific model).
  - Infer copy number clones (clone mode).
  - Infer a phylogenetic tree (phylogeny mode).
  - Call heterozygous SNPs from matched normal WGS (optional, allele-specific model).
- scRNA preprocessing,
  - Alignment and QC.
  - Classify cells as tumor or normal.
  - Genotype heterozygous SNPs in scRNA (optional, allele-specific model).

A flowchart of the preprocessing pipeline is shown in Fig. 2.

We describe the workflow in the following sections and provide examples based on data generated from DLP+ and 10X sequencing of a pretreatment high grade serous ovarian tumor (OV-105) [16].

2.1.1 Preprocessing and Alignment of scWGS Data

TreeAlign has been tested with both DLP+ [17] and 10X scWGS data although other scWGS data types should also be possible. For alignment and QC of scWGS data, we recommend using the pipeline designed for each data type. For 10X CNV, the cellranger DNA pipeline [18] will generate a BAM file with each read properly tagged with the corresponding cell barcode. For DLP+ data, we recommend using Mondrian [17, 19]. The Mondrian pipeline will perform alignment and total copy number calling using HMMCopy and provide metrics for filtering poor quality and replicating cells. When using DLP+, we recommend filtering cells

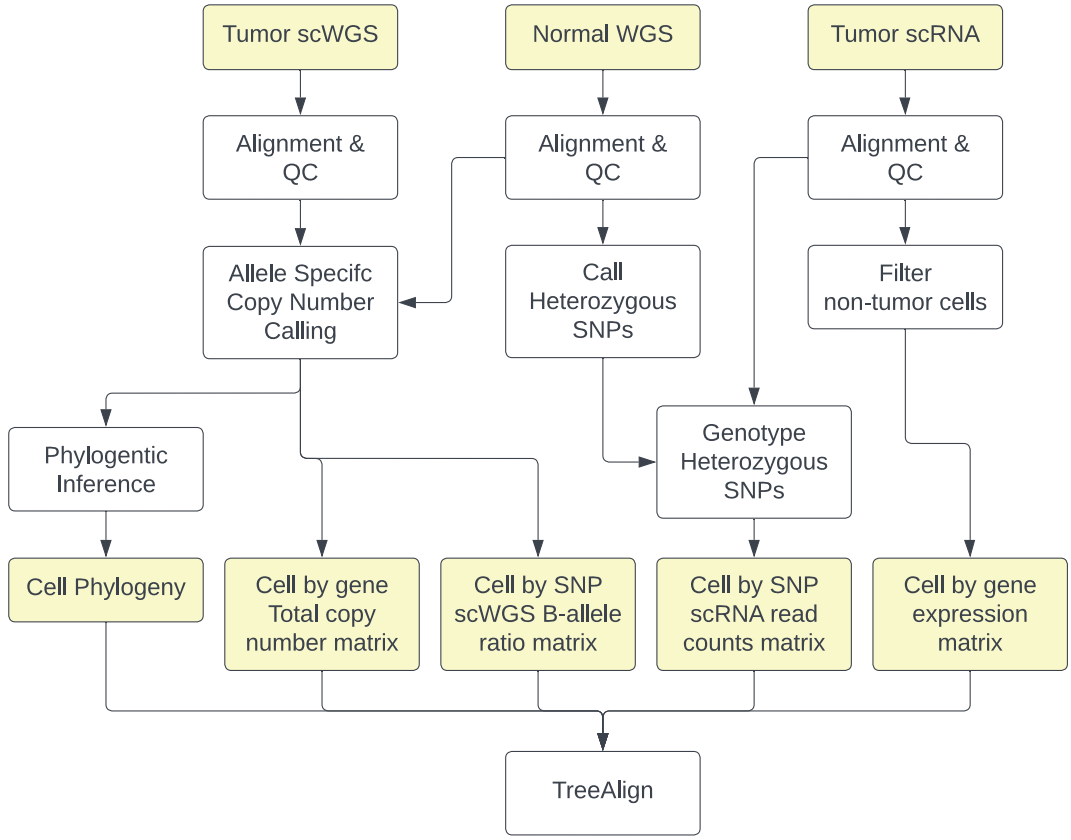


Fig. 2 TreeAlign preprocessing workflow

with `quality` < 0.75 in addition to cells labeled `is_s_phase`. For 10X CNV data, the default filtering on DIMAPD and mapped read count generally suffice.

2.1.2 Total Copy Number Calling from scWGS

Both Mondrian and cellranger DNA provide subcommands to call cell specific total copy number. To QC the resulting copy number calls, it is generally helpful to plot a heatmap of copy number sorted either by a clustering of the cells or a phylogenetic tree (see Fig. 3). A heatmap will allow you to see where segmentation has failed or where the default cell filtering has not adequately removed noisy cells. For a TreeAlign analysis, we advise removing outlier cells. A cell may be an outlier because either the sequencing performed poorly for that cell or because the cell has undergone mitotic failure resulting in significant cell specific differences. In either case, cells with large numbers of cell specific copy number changes will impede accurate clonal or phylogenetic inference and are unlikely to be represented in the scRNA data. Removing outliers can be accomplished with the `scgenome.tl.detect_outliers` function in Mondrian/scgenome [20].

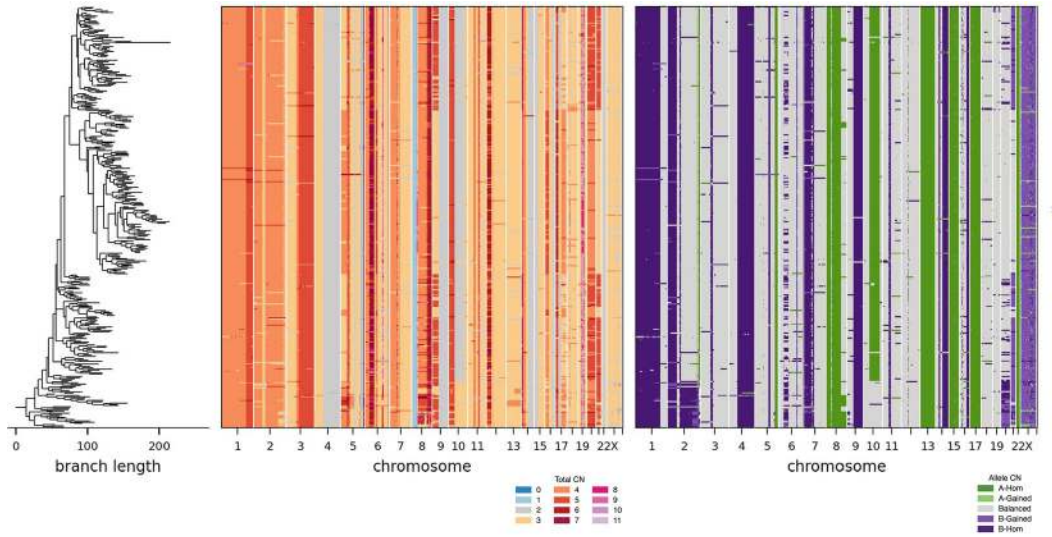


Fig. 3 OV-105 copy number and phylogenetic tree, with clones annotated to the right of the heatmaps

Additional options for calling copy number also worth mentioning include SCOPE [21], FLCNA [22], and Chisel [23]. Both FLCNA and Chisel will also output copy number clones that may be used with TreeAlign. Chisel will also output allele-specific copy number calls that can be used with an allele-specific TreeAlign analysis, obviating the need for Signals (see below).

2.1.3 Allele-Specific Copy Number Calling from scWGS

We recommend using Signals [1] for allele-specific copy number calling, especially when working with DLP+ and using Mondrian for preprocessing and copy number calling. The Mondrian pipeline provides two additional subcommands that can be used to generate inputs for Signals. The `inferhaps` subcommand operates on matched normal WGS, required for Signals, to produce patient specific haplotype blocks of phased SNPs. The `counthaps` subcommand takes the haplotype blocks and scWGS data as input and generates a table of cell specific read counts for haplotype blocks. Signals is then run in R, with the haplotype block read counts and total copy number as input. See the online documentation [24] for a description of the commands used.

As with total copy number, a sorted heatmap of allele-specific copy number will help identify segmentation issues and inadequate cell filtering (see Fig. 3). In addition, a plot of B-Allele Frequency (BAF) per inferred copy number state will also help determine how well the allele-specific copy number fits the raw BAF signal (see Fig. 4).

An important secondary output of Signals is the phase of each SNP relative to the allele-specific copy number assigned to the A and B alleles. Each haplotype block inferred by Mondrian is an

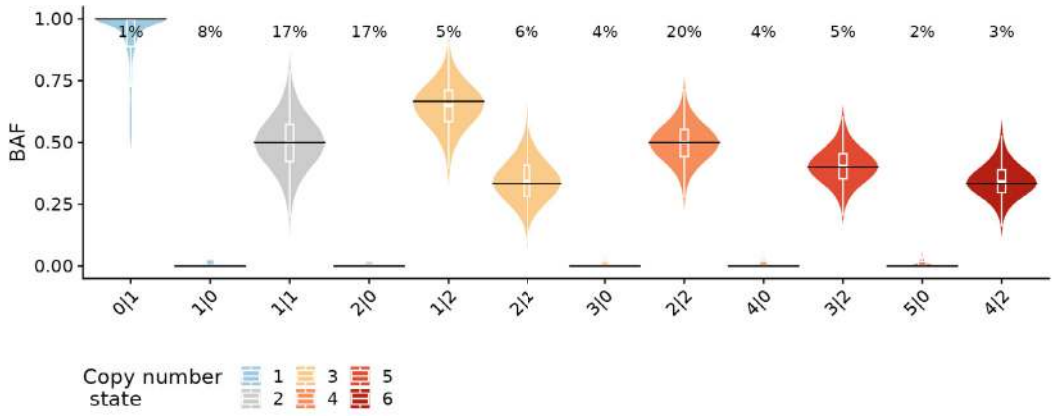


Fig. 4 BAF QC plot for Signals showing distribution of BAF (y-axis) per allele-specific copy number state (x-axis)

assignment of either the reference or alternate allele to haplotype allele 0 or 1. During allele-specific copy number inference Signals will assign the B allele as either haplotype allele 0 or 1 for each block, thereby assigning the B allele as the reference or alternate allele of each SNP. To compute the phase of each SNP, merge the Signals output with the haplotype blocks table produced by the `inferhaps` subcommand of Mondrian.

2.1.4 Inferring a Phylogenetic Tree

Accurate phylogenetic inference of clonal populations from scWGS copy number data is a difficult problem. Cell specific variation can be significant in some datasets, obscuring clonal signal. Copy number changes are homoplastic, and independent copy number changes can produce convergent genomic profiles. As such, this step may require careful tuning of the parameters of the selected method.

For inferring a phylogenetic tree, we recommend using either MEDICC2 [25] or Sitka [26]. MEDICC2 can be run with either allele-specific or only total copy number inputs. We recommend using allele-specific input if available, even if you do not intend to run TreeAlign in allele-specific mode, as the additional information will improve tree inference. To run MEDICC2 in allele-specific mode, reformat the Signals output to the input format required by MEDICC2, using copy number columns `A` and `B` output from Signals as the `cn_a` and `cn_b` inputs for MEDICC2. For total copy number input, use the `state` column output from Mondrian/HMMCopy. To run sitka, follow the Tree Inference Tutorial [27] using total copy number inputs from Mondrian/HMMCopy. Leaves of the resulting tree will be either cells or loci. To ensure the sitka tree is compatible with TreeAlign, prune the tree output by sitka until the only leaves are those representing cells.

A plot of the inferred phylogeny alongside total and allele-specific copy number is often informative for QC of the phylogenetic tree (Fig. 3). Long branches of a small number of cells may result when those cells are outliers, either because they are noisy or replicating or harbor significant cell or subclone-specific change due to complex structural variation or whole genome doubling. These sets of cells can be filtered post hoc, though rerunning phylogenetic inference without these cells may improve the quality of the phylogeny overall. The combined phylogeny copy number plot also gives a global view of how well the phylogenetic method is able to explain copy number changes across cells by grouping them together as having a shared origin in a single clone. Identical copy number changes scattered throughout the phylogeny indicate that noise, particularly at segment boundaries for MEDICC2, is inhibiting the method from grouping together common copy number changes into a shared evolutionary history. Further smoothing of the copy number subsequent to phylogenetic inference may be required in such instances. Finally, comparison with a non-phylogenetic clustering can often inform the quality of the phylogeny.

2.1.5 Inferring Copy Number Clones

A single-cell phylogeny is required to leverage the full capabilities of TreeAlign. Nevertheless, a clustering of cells into putative clonal populations can also be used as input. Several methods have been used to cluster scWGS into clonal populations in previous work. Dimensionality reduction using umap followed by HDBSCAN [1] can be accomplished using the `umap_clustering` function from the Signals R package. K-means or Gaussian mixture model based clustering [17] can be accomplished using the `scgenome.tl.cluster_cells` function in Mondrian/scgenome [20]. If you have run cellranger DNA [18], then hierarchical clustering will be included as an output of the pipeline and can be post-processed to generate clusters of cells. Alternatively, methods specifically tailored for scWGS data including FLCNA [22] and Chisel [23] will produce a clustering of the scWGS cells.

2.1.6 Calling Heterozygous Germline SNPs in Matched Normal WGS

Where possible, heterozygous SNPs should be obtained from matched normal WGS data. WGS data should be subject to alignment and QC following GATK best practices [28]. Two options are possible for calling SNPs. De novo calling can be accomplished using `bcftools mpileup` to pileup reads from a BAM file, followed by `bcftools call` to call and genotype SNPs [29]. Alternatively, you can restrict to SNPs identified from an existing panel such as 1000 genomes [30], accessible through IGSR [31]. Add the panel vcf file as the `--regions-file` option of `bcftools mpileup` to restrict to genotyping SNPs in the panel. Using a panel will result in reduced runtimes and reduce the need to QC the resulting SNP data.

2.1.7 Alignment and QC of scRNA Data

scRNA data should be aligned to a reference genome using one of several dedicated alignment tools, typically CellRanger from 10X genomics [32], though other methods, such as STARsolo [33], alevin [34], or Kallisto [35], can be used depending on the user's needs. These aligners will also typically output a gene expression count matrix that will serve as the basis for downstream expression analysis using either Seurat or ScanPy. These steps can be easily performed using the Nextflow scRNA workflow [36], which is set up for scRNA best practices, allows users to select between the various alignment options, and can be deployed locally on a user's laptop or a local or cloud compute server using Docker or Singularity containers.

Downstream quality control to remove low-quality cells can be performed by filtering for a minimum read count per cell (minimum 200 counts per cell) and removing cells with a high proportion of reads aligned to mitochondrial or ribosomal genes (max 20%). For smaller datasets, filtering thresholds should be inspected and selected based on the overall data quality. For larger datasets, quality filters can be automated by using the median absolute deviation (MAD) to remove cells that differ by 5 MADs in one of these metrics. Ambient RNA correction can be performed using SoupX [37], DecontX [38], or CellBender [39]. Doublet detection and removal can be performed using scDblFinder [40].

2.1.8 Classifying scRNA Cells as Tumor or Normal

Patient-derived scRNA tumor samples are typically composed of both malignant and nonmalignant cells, which must be accurately labeled prior to downstream analysis. Complementary approaches are usually recommended to achieve this task. First, scRNA based copy number inference methods, such as inferCNV [41], Numbat [42], SCEVAN [43], or others [44–46], may be used to identify cells with copy number alterations consistent with malignant cells. In parallel, exploration of the gene expression using variable feature selection, PCA, and clustering should reveal patient specific clusters of epithelial cells (in the case of epithelial-derived tumors), which should coincide with copy number alteration containing cells. These clusters should be distinct from nonmalignant epithelial cells, which should cluster together between patient samples derived from the same tissue more readily (assuming minimal batch effects between patients).

2.1.9 Genotyping Heterozygous SNPs in scRNA

Heterozygous loci can be measured in the scRNA data using CellSNP [47] or alternatively Vartrix [48] to compute allele-specific scRNA read counts for each heterozygous SNP. CellSNP or Vartrix will produce a matrix of total depth at each SNP position and number of reads supporting the alternate allele. For optimal performance, TreeAlign should be provided with not just the read counts of each SNP but the phase aware read counts relative to the phasing inferred during allele-specific copy number calling from

scWGS. For each SNP, the alternate allele is either on the A or B allele as called by Signals or other allele-specific copy number methods. The phasing information from Signals or other tools can then be used to compute the number of reads supporting the B allele, as required for input to TreeAlign. Note that if phasing information is not available, TreeAlign can infer the B allele of each SNP during model inference.

2.2 Running TreeAlign

TreeAlign models: (1) clone-specific gene expression level based on corresponding copy numbers and (2) proportions of reads from B alleles in scRNA based on B allele frequencies estimated from scWGS. After successfully preprocessing and QC of input raw data, running TreeAlign involves preparing input data matrices and executing a small number of TreeAlign API calls in python.

2.2.1 Preparing Input Data

Prepare inputs to TreeAlign as follows:

Phylogenetic tree: A tree representing the evolution of cells from a diploid ancestor with cells as leaf nodes. The tree input should be provided as `Bio.Phylo.BaseTree` which can be constructed by reading the phylogeny stored in newick format using the `Phylo.read` function from the `Biopython` package.

Clone assignments: Cell to clone cluster assignment table, used in place of a phylogenetic tree. The clone assignment table provided should contain two columns: `cell_id` to identify each cell and `clone_id` for defining the clone to which each scWGS cell is assigned.

Gene expression: Gene by cell matrix of raw read counts from scRNA.

Total copy number from scWGS: Gene by cell matrix of integer copy number from scWGS. This input typically necessitates a conversion from a matrix of genomic bin by cell copy number output from a copy number inference method like Mondrian/HMMCopy. The `scgenome.tl.aggregate_genes` function in Mondrian/scgenome [20] can be used to calculate a gene by cell matrix from a bin by cell matrix and gene start and end positions.

Allele-specific copy number from scWGS: SNP by cell matrix of B allele fraction for each heterozygous SNP position. B allele fractions should range from 0 to 1. From Signals, B allele fractions are calculated as the copy number of the B allele (`B` column) as a fraction of total copy number (`A` column + `B` column). The `scgenome.tl.intersect_positions` function in Mondrian/scgenome [20] can be used to calculate a position by cell matrix from a bin by cell matrix and a set of positions representing locations of heterozygous SNPs.

SNP allele evidence in scRNA: SNP by cell read count matrix for which each entry is the number of reads supporting the B allele at a heterozygous SNP position for a given cell in the scRNA data. Computing this matrix involves extracting reference and alternate read counts from CellSNP or Vartrix and then using the per-SNP B allele assignments from Signals or similar to produce a matrix of read counts supporting the B allele of each SNP. If such an assignment is not possible, the number of reads supporting the reference can be provided, but the `infer_b_allele` option should be used (see below).

SNP depth in scRNA: SNP by cell read count matrix for which each entry is the total number of reads overlapping a heterozygous SNP position for a given cell in the scRNA data. This input can be constructed directly from the output of CellSNP or Vartrix.

2.2.2 Running TreeAlign

The input datasets described in the previous section can be used to construct the python `CloneAlignTree` object for data cleaning and preprocessing. The phylogenetic tree, gene expression, and total copy number inputs are required. The allele-specific copy number, SNP B allele evidence, and SNP depth inputs are optional. If allele-specific scRNA evidence is phased with scWGS allele-specific copy number, set the `infer_b_allele` to False. Otherwise, set `infer_b_allele` to True to specify that TreeAlign should learn B allele assignment during inference. If allele-specific inputs are not provided, the allele-specific extension of TreeAlign will not be run and the clone assignment results will be purely based on total copy number and gene expression. After construction of the `CloneAlignTree` object, `assign_cells_to_tree` function can be called to initialize the inference process.

In addition to assigning expression profiles to a scWGS-based phylogeny, TreeAlign also allows assigning expression profiles to predefined clones similar to CloneAlign [49]. Instead of providing single-cell phylogenetic tree based on scWGS, users can provide a clone assignment table to construct a `CloneAlignClone` object to initiate assignment of expression profiles to predefined CN clones. Set the `infer_b_allele` option accordingly. After construction of the `CloneAlignClone` object, `assign_cells_to_clones` function can be called to initialize the inference process.

2.2.3 Tunable Parameters

Additional parameters can be set to customize the running of TreeAlign.

`repeat`: number of repeated runs of inference in TreeAlign. As the final clone assignment results are determined by the majority votes from repeated runs, a larger value for `repeat` parameter may generate more robust results but take longer time to finish running.

`min_cell_count_expr`: minimum number of cells in the scRNA data to allow assigning expression profiles to smaller subtrees.

`min_cell_count_cnv`: minimum number of cells in the scWGS data to allow TreeAlign to keep assigning expression profiles to smaller subtrees.

`min_clone_assign_prob`: minimum probability required to assign expression profiles to a subclone.

`min_clone_assign_freq`: minimum proportion of repeated runs with consistent clone assignment needed to assign expression profiles to a subclone.

By adjusting these four parameters of `min_clone_assign_freq`, `min_clone_assign_prob`, `min_cell_count_expr`, and `min_cell_count_cnv`, users can tune how far they want TreeAlign to proceed down a phylogenetic tree in the alignment of scRNA cells to tree clades. Forcing TreeAlign to proceed further down a phylogenetic tree allows it to assign expression profiles to smaller subclones but may reduce the overall performance of the model and leave more expression profiles in the unassigned state.

`min_consensus_gene_freq`: The purity threshold for a gene to be included in the CN input matrix. It is defined as the proportion of cells with the modal copy number in a clone for that gene. This tunable parameter was inspired by the observation that within the same clones, the integer CNs inferred from hmmcopy are still heterogeneous at a cell level; therefore, we want to preserve the more homogeneous and clone-specific copy number signals to be used as input for TreeAlign. min_consensus_gene_freq >0.5 should be a reasonable starting point as we want to ensure that all input CN values are consensus in given clones. Users may want to further increase this value to allow genes which are more representative of a clone to be used as input, especially if there are already sufficient numbers of genes to serve as input, and TreeAlign assignment is confident and stable across different runs.

`min_consensus_snv_freq`: The purity threshold for a heterozygous SNP to be included in the BAF input matrix—similar to `min_consensus_gene_freq`.

2.2.4 TreeAlign Outputs

After successful inference, `generate_output` function can be called to generate results. The results consist of three dataframes:

Clone assignment: A table assigning scRNA cells (`cell_id` column) to scWGS tree clades or clones (`clone_id` column).

Dosage effect: inferred level of dosage effect (`gene_type_score` column) for each gene (`gene` column) determined to be in a clone-specific copy number region during TreeAlign's internal prefiltering step. The gene_type_score column contains `p(k)` scores for genes. `p(k)` ranges from 0 to 1 and represents

the probability of a gene's expression depending on copy number dosage effects.

Inferred B allele: A table of SNP allele assignments inferred if allele-specific inputs are provided and `infer_b_allele` is set to `True`. If allele-specific inputs were not provided or parameter `infer_b_allele` is set to False, the return dataframe will be empty.

2.3 Interpreting TreeAlign Results

TreeAlign results provide insight into the clone-specific expression patterns in a population of tumor cells. scRNA cells will cluster at clades throughout the phylogeny, with each cluster best explained by the mode of allele-specific copy number of the given clade. A cluster of scRNA cells assigned closer to the root of the phylogeny implies that the cells in that cluster are unlikely to harbor the copy number changes that differentiate any smaller subclades. Some scRNA cells will not be assigned to a clade, indicating that these cells have divergent copy number profiles from those found in the scWGS population. A helpful visual aid is to plot scWGS copy number alongside scRNA copy number, with scWGS cells ordered by the inferred phylogenetic tree and scRNA cells ordered by their assignment to clades in the tree (Fig. 5). For OV-105, this plot reveals a small subclone with divergent copy number on chromosomes 2, 3, and 19, with evidence in both modalities.

In addition, TreeAlign clone assignment results can be used to compare expression profiles between cancer cell subclones and characterize clone-specific transcriptional phenotypes (Fig. 6). TreeAlign assigned cancer cell expression profiles from ovarian cancer patient OV-105 to six clones. Using differential expression analysis, we can identify expression programs that are specific to each clone. For example, compared to other clones in patient OV-105, clone E has upregulated expression of genes in IFN

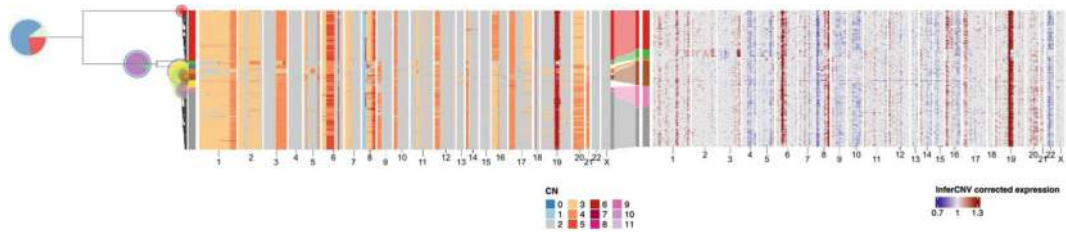


Fig. 5 Integrated model of TreeAlign assigns expression profiles of OV-105 to phylogeny. Heatmaps of copy number profiles from scWGS (left) and InferCNV corrected expression profiles from scRNA (right). The Sankey chart in the middle shows clone assignment from expression profiles to copy number based clones by integrated TreeAlign. Pie charts on the tree showing how TreeAlign assigns cell expression profiles to subtrees recursively. The pie charts are colored by the proportions of cell expression profiles assigned to downstream subtrees. The outer ring color of the pie charts denotes the current subtree

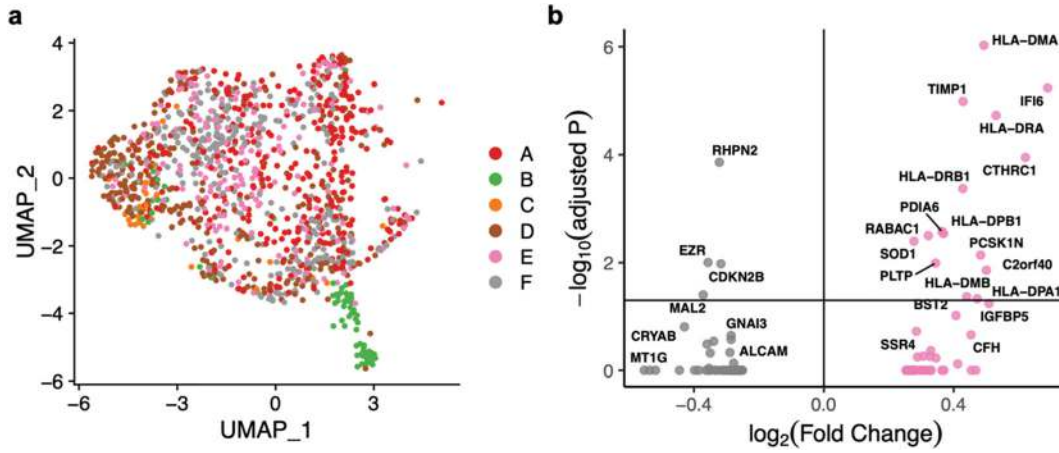


Fig. 6 TreeAlign results for OV-105 showing (**a**) UMAP embedding of scRNA cell expression profiles colored by TreeAlign inferred clones and (**b**) differentially expressed genes between clone E and all other subclones

responses and antigen presentation. IFN signaling has important immune modulatory effects and has been previously linked to immune evasion and resistance to immunotherapy [50]. In addition to patient OV-105, IFN signaling was also found to be highly variable between clones in ovarian cancers [14]. The recurrent differential expression of immune related pathways between subclones suggests their importance in clonal divergence in ovarian cancers.

References

1. Funnell T, O'Flanagan CH, Williams MJ et al (2022) Single-cell genomic variation induced by mutational processes in cancer. Nature 612: 106–115
2. Henrichsen CN, Vinckenbosch N, Zöllner S et al (2009) Segmental copy number variation shapes tissue transcriptomes. Nat Genet 41: 424–429
3. Bhattacharya A, Bense RD, Urzúa-Traslaviña CG et al (2020) Transcriptional effects of copy number alterations in a large set of human cancers. Nat Commun 11:715
4. Ding J, McConechy MK, Horlings HM et al (2015) Systematic analysis of somatic mutations impacting gene expression in 12 tumour types. Nat Commun 6:8554
5. Jörnsten R, Abenius T, Kling T et al (2011) Network modeling of the transcriptional effects of copy number aberrations in glioblastoma. Mol Syst Biol 7:486
6. Sztal TE, Stainier DYR (2020) Transcriptional adaptation: a mechanism underlying genetic robustness. Development 147
7. El-Brolosy MA, Stainier DYR (2017) Genetic compensation: a phenomenon in search of mechanisms. PLoS Genet 13:e1006780
8. Fehrmann RSN, Karjalainen JM, Krajewska M et al (2015) Gene expression analysis identifies global gene dosage sensitivity in cancer. Nat Genet 47:115–125
9. Veitia RA, Bottani S, Birchler JA (2013) Gene dosage effects: nonlinearities, genetic interactions, and dosage compensation. Trends Genet 29:385–393
10. Pollack JR, Sørlie T, Perou CM et al (2002) Microarray analysis reveals a major direct role of DNA copy number alteration in the transcriptional program of human breast tumors. Proc Natl Acad Sci USA 99:12963–12968

11. Andor N, Lau BT, Catalanotti C et al (2020) Joint single cell DNA-seq and RNA-seq of gastric cancer cell lines reveals rules of in vitro evolution. NAR Genom Bioinform 2:lqaa016
12. Guo L, Yi X, Chen L et al (2022) Single-cell DNA sequencing reveals punctuated and gradual clonal evolution in hepatocellular carcinoma. Gastroenterology 162:238–252
13. Gonzalo Parra R, Przybilla MJ, Simovic M, et al (2021), Single cell multi-omics analysis of chromothriptic medulloblastoma highlights genomic and transcriptomic consequences of genome instability., https://www.biorxiv.org/content/10.1101/2021.06.25.449944v1
14. Shi H, Williams MJ, Satas G et al (2024) Allele-specific transcriptional effects of subclonal copy number alterations enable genotype-phenotype mapping in cancer cells. Nat Commun 15:2482
15. Bingham E, Chen JP, Jankowiak M et al. Pyro: Deep universal probabilistic programming, https://www.jmlr.org/papers/volume20/18-403/18-403.pdf
16. Vázquez-García I, Uhlitz F, Ceglia N et al (2022) Ovarian cancer mutational processes drive site-specific immune evasion. Nature 612:778–786
17. Laks E, McPherson A, Zahn H et al (2019) Clonal decomposition and DNA replication states defined by scaled single-cell genome sequencing. Cell 179:1207–1221.e22
18. What is cell ranger DNA? -software -single cell CNV -official 10x genomics support. https://support.10xgenomics.com/single-cell-dna/software/pipelines/latest/what-is-cell-ranger-dna
19. mondrian-scwgs, Github
20. scgenome, Github
21. Wang R, Lin D-Y, Jiang Y (2020) SCOPE: a normalization and copy-number estimation method for single-cell DNA sequencing. Cell Syst 10:445–452.e6
22. Qin F, Cai G, and Xiao F (2023) A statistical learning method for simultaneous copy number estimation and subclone clustering with single cell sequencing data. bioRxiv
23. Zaccaria S, Raphael BJ (2020) Characterizing allele- and haplotype-specific copy numbers in single cells with CHISEL. Nat Biotechnol. 39(2):207–214
24. Signals. https://shahcompbio.github.io/signals
25. Kaufmann TL, Petkovic M, Watkins TBK et al (2022) MEDICC2: whole-genome doubling aware copy-number phylogenies for cancer evolution. Genome Biol 23:241
26. Dorri F, Salehi S, Chern K et al (2020) Efficient Bayesian inference of phylogenetic trees from large scale, low-depth genome-wide single-cell data. https://www.biorxiv.org/content/10.1101/2020.05.06.058180v1
27. sitkatree, Github
28. DePristo MA, Banks E, Poplin R et al (2011) A framework for variation discovery and genotyping using next-generation DNA sequencing data. Nat Genet 43:491–498
29. Danecek P, Bonfield JK, Liddle J et al (2021) Twelve years of SAMtools and BCFtools. Gigascience 10
30. 1000 Genomes Project Consortium, Auton A, Brooks LD et al (2015) A global reference for human genetic variation. Nature 526:68–74
31. Fairley S, Lowy-Gallego E, Perry E et al (2020) The International Genome Sample Resource (IGSR) collection of open human genomic variation resources. Nucleic Acids Res 48:D941–D947
32. Zheng GXY, Terry JM, Belgrader P et al (2017) Massively parallel digital transcriptional profiling of single cells. Nat Commun 8
33. Kaminow B, Yunusov D, Dobin A (2021) STARsolo: accurate, fast and versatile mapping/quantification of single-cell and single-nucleus RNA-seq data. https://www.biorxiv.org/content/10.1101/2021.05.05.442755
34. Srivastava A, Malik L, Smith T et al (2019) Alevin efficiently estimates accurate gene abundances from dscRNA-seq data. Genome Biol 20:65
35. Melsted P, Booeshaghi AS, Liu L et al (2021) Modular, efficient and constant-memory single-cell RNA-seq preprocessing. Nat Biotechnol 39:813–818
36. scrnaseq: Introduction. https://nf-co.re/scrnaseq
37. Young MD, Behjati S (2020) SoupX removes ambient RNA contamination from droplet-based single-cell RNA sequencing data. Gigascience 9:giaa151
38. Yang S, Corbett SE, Koga Y et al (2020) Decontamination of ambient RNA in single-cell RNA-seq with DecontX. Genome Biol 21:57
39. Fleming SJ, Chaffin MD, Arduini A et al (2023) Unsupervised removal of systematic background noise from droplet-based single-cell experiments using CellBender. Nat Methods 20:1323–1335
40. Germain P-L, Lun A, Garcia Meixide C et al (2021) Doublet identification in single-cell sequencing data using scDblFinder. F1000Res 10:979

41. inferCNV. https://github.com/broadinstitute/inferCNV
42. Gao T, Soldatov R, Sarkar H et al (2023) Haplotype-aware analysis of somatic copy number variations from single-cell transcriptomes. Nat Biotechnol 41:417–426
43. De Falco A, Caruso F, Su X-D et al (2023) A variational algorithm to detect the clonal copy number substructure of tumors from scRNA-seq data. Nat Commun 14:1074
44. Gao R, Bai S, Henderson YC et al (2021) Delineating copy number and clonal substructure in human tumors from single-cell transcriptomes. Nat Biotechnol 39:599–608
45. Serin Harmanci A, Harmanci AO, Zhou X (2020) CaSpER identifies and visualizes CNV events by integrative analysis of single-cell or bulk RNA-sequencing data. Nat Commun 11:89
46. Fan J, Lee H-O, Lee S et al (2018) Linking transcriptional and genetic tumor heterogeneity through allele analysis of single-cell RNA--seq data. Genome Res 28:1217–1227
47. Huang X, Huang Y (2021) Cellsnp-lite: an efficient tool for genotyping single cells. Bioinformatics 37:4569–4571
48. vartrix: Single-Cell Genotyping Tool, Github
49. Campbell KR, Steif A, Laks E et al (2019) Clonealign: statistical integration of independent single-cell RNA and DNA sequencing data from human cancers. Genome Biol 20:54
50. Benci JL, Xu B, Qiu Y et al (2016) Tumor interferon signaling regulates a multigenic resistance program to immune checkpoint blockade. Cell 167:1540–1554.e12

Chapter 8

Inference of Genetic Ancestry from Cancer-Derived Molecular Data with RAIDS

Pascal Belleau, Astrid Deschênes, David A. Tuveson, and Alexander Krasnitz

Abstract

There has recently been increasing appreciation of ancestral effects on cancer genotypes and phenotypes. Consequently, the need has grown for ancestry annotation of cancer-derived molecular data. In response, we created a computational tool termed RAIDS (Robust Ancestry Inference using Data Synthesis). RAIDS is designed to infer genetic ancestry using as input sequence data from a variety of molecular protocols, even in the absence of matching cancer-free genotypes of the patient. Implemented as an R language package, RAIDS is available from the Bioconductor repository. Here we describe functionalities of RAIDS, provide instructions for its installation, give examples of its usage, and explain the interpretation of its output. While RAIDS is being actively developed, the guidance provided here is expected to apply to future refined and expanded versions of this software tool.

Key words Genetic ancestry, Genotyping, Continental populations, Synthetic data, Principal-component analysis

1 Introduction

There is ample epidemiological evidence that race and ethnicity are important determinants of incidence, clinical course, and outcome in multiple types of cancer [1–5]. As such, these categories must be taken into account in the analysis of molecular data derived from cancer. A number of recently published large-scale genomic studies of cancer [6–11] point to differences in the molecular makeup of the disease among groups of different ancestral background. These differences extend to frequencies of somatic mutations in driver genes [6, 10, 11], the degree of genomic instability and somatic copy number variation (CNV) [8, 9], and immune response to cancer [7]. These ancestral differences, in turn, have been found to have a major effect on response to treatment [12]. Knowledge of

Alexander Krasnitz and Pascal Belleau (eds.), *Cancer Bioinformatics*, Methods in Molecular Biology, vol. 2932,
https://doi.org/10.1007/978-1-0716-4566-6_8,

patient genetic ancestry may be decisive in other contexts, e.g., to discover that patients from an ancestral group are at higher risk for side effects from treatment [13].

More molecular data are needed to power discovery of ancestry-specific effects in cancer. The data shortage for this purpose is especially acute for many non-European ancestries that have not been sufficiently sampled to date. For example, a massive cancer dataset of The Cancer Genome Atlas (TCGA) only lists 27 self-identified Alaskan natives/American Indians and only 13 native Hawaiians/Pacific islanders. Even for populations with higher aggregate numbers of patient cases, the case count is often very small for individual cancer types. For example, while a total of 934 African American (AA) patients are listed by TCGA, only 7 AA cases are listed for prostate and pancreatic cancers each, despite the high incidence of both among AA.

With computational tools described here, we seek to alleviate this data shortage by facilitating reliable, detailed, data-driven ancestry annotation of cancer-derived molecular data from two major sources: (a) existing data available for secondary analysis and (b) cancer-derived specimens, including, notably, those in tissue archives. Secondary data analysis on a massive scale is by far the most efficient, rapid, and economically feasible way to study ancestral impact on the molecular features of cancer. Although data acquisition from tumor-derived specimens is far more expensive and time-consuming by comparison, it is often made necessary by a study design. In such cases, matching cancer-free material may not always be available for ancestry analysis by conventional methods, very often with no possibility of a follow-up specimen collection from the patient. This is especially likely to be the case with archived tumor tissues. Ancestry inference from newly generated data originating in archival tumor tissues is still necessary in such settings and is the second major application for the tools described here.

Ancestry annotation of cancer-derived data draws on two sources. One is a patient's self-identified race and/or ethnicity (SIRE), designed to capture social and cultural factors affecting health. SIRE correlates with but is distinct from ancestry. SIRE is often missing, sometimes inaccurate, and usually incomplete. As a recent analysis [14] of PubMed database entries since 2010 reveals, patients' SIRE is massively underreported in genome and exome sequencing studies of cancer, with only 37% of these reporting race and 17% reporting ethnicity. Furthermore, SIRE has recently been found missing from 56% of electronic health records [15] and is therefore unavailable for annotation of many archived specimens. When available, SIRE is not always consistent with genetic ancestry. A self-declaring patient is often given a choice from a small number of broad racial or ethnic categories. As a result, SIRE fails to capture complete ancestral information and to quantify ancestral admixtures.

A far more accurate and detailed ancestral characterization may be obtained by genotyping a patient's DNA from a cancer-free tissue. Powerful methods exist for ancestry inference from germline DNA sequence [16–18]. These methods were recently used to determine ancestry of approximately 10,000 patients profiled by TCGA [7, 8]. However, genotyping of DNA from matched normal specimens is not part of standard clinical practice, where the purpose of DNA profiling is often identification of mutations with known oncogenic effects, such as those in the Catalog Of Somatic Mutations In Cancer (COSMIC). As a result, it is not performed routinely outside academic clinical centers or major research projects. There also are studies yielding sequence data from tumors, whose purpose does not require germline profiling. RNA sequencing (RNA-seq) for expression quantification, DNA methylation analysis of cytosine-converted DNA, chromatin accessibility profiling by sequencing (ATAC-seq), and low-coverage whole-genome sequencing (WGS) are in this category. Finally, peripheral blood is most often the source of germline DNA in the clinic, but this is not the case for diseases of the hematopoietic system, e.g., leukemias, wherein cancer cells are massively present in circulation. In summary, matched germline DNA sequence is not universally available for cancer-derived molecular data. In such cases, it is necessary to infer ancestry from the nucleic acids of the tumor itself.

Standard methods of ancestry inference rely on population specificity of germline single-nucleotide variants (SNVs). WGS or whole-exome sequences (WES), at depths sufficient for reliably calling single-nucleotide variants, and readouts from genotyping microarrays, are therefore most suitable for this purpose. However, detailed DNA profiling is often not performed in molecular studies of cancer. In such cases, it is necessary to infer ancestry from other types of tumor-derived data, including RNA-seq, DNA sequence for a small panel of genes (e.g., FoundationOne CDx [19]), low-coverage WGS, cytosine-converted sequences, and ATAC-seq.

Ancestry inference from molecular data other than germline DNA sequence faces two challenges. One is cancer-specific, common to all types of tumor-derived sequence. Tumor genomes are often replete with somatic alterations, including copy number variants, translocations, loss of heterozygosity (LOH), microsatellite instabilities, and SNV. All these alterations are, to various degrees, potential obstacles to accurate ancestry inference. For example, an LOH event may render one of the parental alleles inaccessible to ancestry analysis in a broad chromosomal fragment. The other challenge is data-specific. RNA-seq yields extremely uneven coverage of the transcript both due to a broad range of RNA expression levels and to molecular protocol design. Further distortions arise from allele-specific expression. Similar nonuniformity of coverage is found in ATAC-seq data. Gene panels represent a small fraction of the genome, whose sufficiency for ancestry inference is not clear a

priori and varies from panel to panel. In addition, such panels are enriched in cancer-driver genes, which tend to undergo somatic alteration more frequently than other parts of the genome. Confident SNV profiling, required by all existing ancestry inference methods, may be challenging with low-coverage WGS. Cytosine-converted data present unique challenges: any observed thymine next to a guanine may result from conversion.

Inferential tools described here are designed to address two critical questions. (a) How can the accuracy of genetic ancestry inference be assessed and optimized for a given molecular profile? The profile in question may have its unique set of sequence properties. These include the underlying genotype and phenotype, the target sequence and uniformity of its coverage depth, read length, and sequencing quality. (b) How can such assessment and optimization be accomplished in the absence of a control dataset from a large, ancestrally diverse cohort? The requisite cohort would have to yield molecular profiles with underlying biology and technical properties closely similar to those of the given profile and to have genetic ancestry of the donors inferred from the germline genotypes. Such a set of controls is not available for the vast majority of existing molecular profiles.

In the following, we present computational methodology for global, continental-level ancestry calls from tumor-derived molecular data, including whole-exome sequences, specialized gene panels, and RNA sequences [20, 21]. This methodology combines existing algorithms for ancestry inference with a novel adaptive procedure for inference parameter optimization and rigorous performance assessment, termed "data synthesis." The resulting tools have been validated using a representative subset of public cancer-derived data and made publicly available in Bioconductor [21], as an R language package termed RAIDS. We find their accuracy to be consistently high across ancestral groups, sequencing modalities, and the four cancer types examined in our paper [20].

The remainder of this Protocol is structured as follows. In the Methods section, we explain how RAIDS is installed and specify the reference data it requires. We next describe the key functionalities of RAIDS. This section ends with an example of RAIDS applied to a cancer-derived molecular profile, explained step by step. The chapter ends with Notes, providing further details on installation and execution of RAIDS.

2 Methods

The genetic ancestry inference procedure consists of three main steps as shown in Fig. 1.

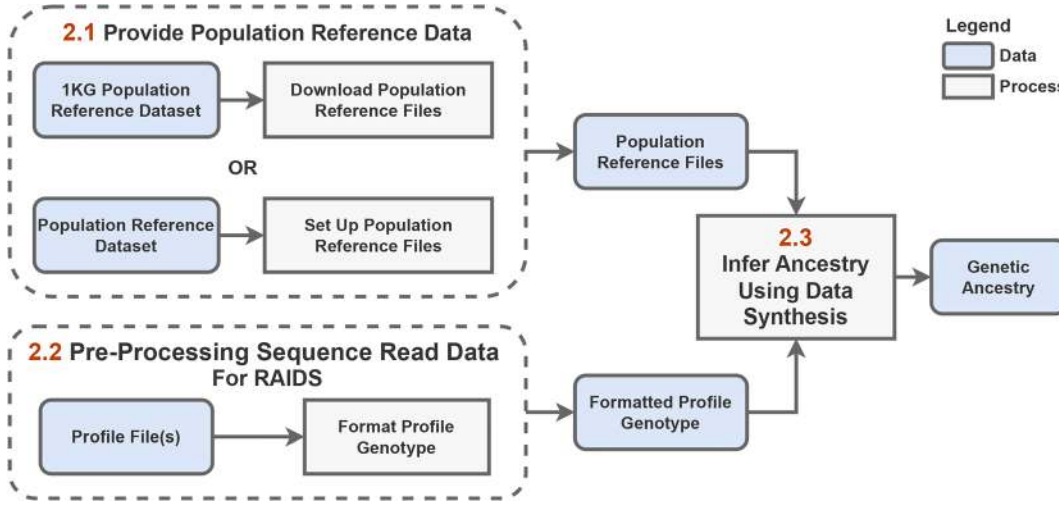


Fig. 1 An overview of the genetic ancestry inference procedure. The preparation of population reference files is explained in Subheading 2.1, computation of profile genotypes in Subheading 2.2, and the donor genetic ancestry in Subheading 2.3, respectively

2.1 Population Reference Data

The 1000 Genomes (1KG in the following) population reference dataset [22], hosted by the International Genome Sample Resource, is at present the most detailed and complete dataset of its kind. This dataset provides reference for five continental super-populations: African (AFR), American (AMR), East Asian (EAS), European (EUR), and South Asian (SAS). The full set of 1KG genotypes with at least 1% frequency in at least one super-population is available at https://labshare.cshl.edu/shares/krasnitzlab/aicsPaper in the genomic data structure (GDS) format. GDS is the most suitable format for handling large volumes of genomic data using the R programming language. We will use this population reference input throughout this protocol. Alternatively, users can build their own reference datasets. Instructions for doing are provided in Subheading 2.4.

The population reference input into RAIDS consists of two files: the reference genotype file (`matGeno1000g.gds` provided for the 1KG reference) and the reference annotation file (`matAnnot1000g.gds` provided for the 1KG reference). The reference genotype file contains, for each donor to the population reference data, the genotypes at all positions in the genome, where the alternative allele frequency in at least one super-population is at least 1%. The reference annotation file tabulates, for each super-population, its haplotype blocks. Further details on the data structures contained in these files are provided in Subheading 2.4.

2.2 Preprocessing Sequence Read Data for RAIDS

A file containing sequence reads mapped to the same human genome build as the population reference data in the BAM format is required as input for each input profile. If mapping or remapping is necessary, it should follow the Genome Analysis Toolkit (GATK) [23] best practice guidelines. If 1KG reference data provided with the RAIDS package are used for inference, the reads should be mapped to the GRCh38 (hg38) build of the human genome [24]. These guidelines are different for sequences derived from DNA and those derived from RNA.

The GATK best practice guidelines are found at GATK website for DNA-derived sequences [25] and for RNA-derived sequences [26]. The BAM file must be sorted and indexed, e.g., using Samtools [27].

2.3 Ancestry Inference Using Data Synthesis

In this section, we will describe in detail the inference procedure used by RAIDS. This procedure relies on data synthesis for inference parameter optimization and inference performance evaluation. The main steps of Subheading 2.3.1 are as follows:

1: Setup and data preprocessing for RAIDS.

2: Infer the ancestry with RAIDS.

2) A: Sample the reference data for donor genotypes, to be used for synthesis. A fixed number of donor genotypes are sampled from each population. For example, there are 26 populations in the 1KG reference. One can sample 30 genotypes from each of the 26 populations defined in the 1KG reference, to the total of 780.

2) B: Execute a function call to one of the RAIDS functions, `inferAncestryDNA()` or `inferAncestryDNA()`, depending on the molecular source of the input profile. Each of these calls will result in a series of operations as follows:

- Genotype the input sequence data at all biallelic single-nucleotide polymorphic (SNP) positions in the genome where the frequency of either allele is above a minimum frequency `cutOff` in the reference data.
- Prune the set of positions resulting from (A) to reduce the linkage disequilibrium between any two such positions below a value of $\sqrt{0.1}$ (*see* **Note 1**).
- Infer genetic ancestry for the entire set of synthetic profiles and a range of inference parameters. Use the results to optimize the inference parameters.
- Infer genetic ancestry of the input profile using the optimal parameters found above.

3: Present and interpret the results of (2) (see ***Note 2****)*

3) A: Output the inferred ancestry of the input profile, the optimal set of parameters, and the final state of the inference algorithm.

3) B: Evaluate the performance of the inference algorithm on the synthetic data.

2.3.1 Example: Genetic Ancestry of an RNA Sequence Profile

In this example, we shall use as input an RNA-seq profile from the Encode collection [28] (the accession numbers: ENCFF001RFH and ENCFF001RFG).

1: Setup and data preprocessing for RAIDS

1) A: Create a directory structure for the example.

First you need a working directory (`workingDirectory`) that will contain all the other directories. You have to recreate the following structure on your computer.

```
workingDirectory/
    data/
        refGDS
        fastq
        genomeReference
        GATK
        010_star
        020_picard
        030_splitNCigar
        040_recalibration
        profileGDS
        res.out
```

1) B: Download the population reference files.

```
cd workingDirectory
cd data/refGDS

wget  https://labshare.cshl.edu/shares/krasnitzlab/aicsPaper/
matGeno1000g.gds
wget  https://labshare.cshl.edu/shares/krasnitzlab/aicsPaper/
matAnnot1000g.gds
cd -
```

1) C: Download the two unmapped paired-end sequence read data files for the input profile in the FASTQ format (approximately 17Gb in total):

```
cd data/fastq

wget  https://www.encodeproject.org/files/ENCFF001RFH/@@down-
load/ENCFF001RFH.fastq.gz -O ENCFF001RFH.fastq.gz
```

```
wget https://www.encodeproject.org/files/ENCFF001RFG/@@down-
load/ENCFF001RFG.fastq.gz -O ENCFF001RFG.fastq.gz
cd -
```

1) *D:* Download and uncompress the GRCh38 human genome reference sequence file and the corresponding annotation file [29]:

```
export PICARD_JAR=[path_to_picard_jar]
cd data/genomeReference

wget https://ftp.ebi.ac.uk/pub/databases/gencode/Gencode_hu-
man/release_46/GRCh38.p14.genome.fa.gz
wget https://ftp.ebi.ac.uk/pub/databases/gencode/Gencode_hu-
man/release_46/gencode.v46.annotation.gtf.gz
gunzip gencode.v46.annotation.gtf.gz
gunzip GRCh38.p14.genome.fa.gz

java -Xmx2G -jar $PICARD_JAR CreateSequenceDictionary \
    REFERENCE=GRCh38.p14.genome.fa \
    OUTPUT=GRCh38.p14.genome.dict

samtools faidx ./GRCh38.p14.genome.fa

cd -
```

1) *E:* Create a genome index for the STAR [30] RNA-seq read mapper (see the GitHub site [31] for details), then run STAR in the two-pass mode with the default parameters, and the read length of 100 bases as used for sequencing in this case.

```
STAR \
--runThreadN 1 \
--runMode genomeGenerate \
--genomeDir data/genomeReference/ \
--sjdbGTFfile data/genomeReference/gencode.v46.annotation.gtf
\
--genomeFastaFiles data/genomeReference/GRCh38.p14.genome.fa
\
--sjdbOverhang 100

STAR \
--runThreadN 1 --genomeDir data/genomeReference/ \
--twopassMode Basic \
--twopass1readsN -1 \
--readFilesIn data/fastq/ENCFF001RFH.fastq.gz data/fastq/
ENCFF001RFG.fastq.gz \
--outSAMtype BAM SortedByCoordinate \
```

```
--readFilesCommand zcat \
--outFileNamePrefix data/010_star/ENCFF001RF
```

1) F: Post-process the mapped sequences following the GATK guidelines [26].

```
cd data/GATK

wget https://storage.googleapis.com/genomics-public-data/re-
sources/broad/hg38/v0/Homo_sapiens_assembly38.dbsnp138.vcf
wget https://storage.googleapis.com/genomics-public-data/re-
sources/broad/hg38/v0/Homo_sapiens_assembly38.dbsnp138.vcf.
idx
wget https://storage.googleapis.com/genomics-public-data/re-
sources/broad/hg38/v0/Mills_and_1000G_gold_standard.indels.
hg38.vcf.gz
wget https://storage.googleapis.com/genomics-public-data/re-
sources/broad/hg38/v0/Mills_and_1000G_gold_standard.indels.
hg38.vcf.gz.tbi

cd -

java -Xmx16g -jar $PICARD_JAR AddOrReplaceReadGroups \
    -I data/010_star/ENCFF001RFAligned.sortedByCoord.out.bam \
     -O data/020_picard/ENCFF001RF.ARG.bam \
     -SO coordinate --RGID 3 --RGLB lib1 --RGPL illumina \
     --RGSM CURRENT --RGPU unit1 --RGCN encode

java -Xmx16g  -jar $PICARD_JAR MarkDuplicates \
    I=data/020_picard/ENCFF001RF.ARG.bam \
    O=data/020_picard/ENCFF001RF_sorted_dedupped.bam \
    CREATE_INDEX=true \
    VALIDATION_STRINGENCY=SILENT \
    M=data/020_picard/ENCFF001RF_sorted_dedupped.metrics

gatk --java-options "-Xmx16g -XX:+UseParallelGC -XX:Paral-
lelGCThreads=1 -XX:ConcGCThreads=1" SplitNCigarReads \
       --reference data/genomeReference/GRCh38.p14.genome.fa \
        --input data/020_picard/ENCFF001RF_sorted_dedupped.bam
\
        --output data/030_splitNCigar/ENCFF001RF_split.bam

gatk --java-options "-Xmx16g -XX:+UseParallelGC -XX:Paral-
lelGCThreads=1 -XX:ConcGCThreads=1" BaseRecalibrator \
   --input data/030_splitNCigar/ENCFF001RF_split.bam \
   --reference data/genomeReference/GRCh38.p14.genome.fa \
    --known-sites data/GATK/Homo_sapiens_assembly38.dbsnp138.
vcf \
```

```
    --known-sites data/GATK/Mills_and_1000G_gold_standard.in-
dels.hg38.vcf.gz \
   --output data/040_recalibration/ENCFF001RF_recal_data.table

gatk --java-options "-Xmx16g -XX:+UseParallelGC -XX:Paral-
lelGCThreads=1 -XX:ConcGCThreads=1" ApplyBQSR \
   --input data/030_splitNCigar/ENCFF001RF_split.bam \
   --reference  data/genomeReference/GRCh38.p14.genome.fa \
   --bqsr-recal-file data/040_recalibration/ENCFF001RF_recal_-
data.table \
   --output data/040_recalibration/ENCFF001RF_recalibrated.bam

samtools sort -O BAM data/040_recalibration/ENCFF001RF_recali-
brated.bam > data/040_recalibration/ENCFF001RF_recalibrated_-
sort.bam
samtools index data/040_recalibration/ENCFF001RF_recalibra-
ted_sort.bam
# Note if all the steps before are ok you can only keep
# data/040_recalibration/ENCFF001RF_recalibrated_sort.ba[mi]
```

2: Ancestry inference with RAIDS

Invoke R from the directory `workingDirectory`.

Install and load RAIDS, and set up the required directory paths.

```
if (!require("BiocManager", quietly = TRUE))
    install.packages("BiocManager")

BiocManager::install("RAIDS")

library(RAIDS)

setwd("workingDirectory")

pathReference <- file.path("data/refGDS")

refGenotype <- file.path(pathReference, "matGeno1000g.gds")
refAnnotation <- file.path(pathReference, "matAnnot1000g.
gds")
pathProfileGDS <- file.path("data", "profileGDS")

pathOut <- file.path("data", "res.out")
# The output directories must exist
if (!dir.exists(pathProfileGDS))
    dir.create(pathProfileGDS)
if (!dir.exists(pathOut))
    dir.create(pathOut)
```

2) A: Sample reference donor profiles from the reference data.

With the 1KG reference, we recommend sampling 30 donor profiles per population. For reproducibility, be sure to use the same random-number generator seed.

```
set.seed(3043)

dataRef <- select1KGPopForSynthetic(fileReferenceGDS=refGen-
otype,
    nbProfiles=30L)
```

2) B: Perform the ancestry inference.

Within a single function call, data synthesis is performed, the synthetic data are used to optimize the inference parameters, and, with these, the ancestry of the input profile donor is inferred.

```
pathToBam <- file.path("data/040_recalibration/ENCFF001RF_re-
calibrated_sort.bam")
chrInfo <- GenomeInfoDb::seqlengths(BSgenome.Hsapiens.UCSC.
hg38::Hsapiens)[1:25]

resOut <- inferAncestryGeneAware(profileFile=pathToBam,
    pathProfileGDS =pathProfileGDS,
    fileReferenceGDS=refGenotype,
    fileReferenceAnnotGDS=refAnnotation,
    chrInfo=chrInfo,
    syntheticRefDF=dataRef,
    genoSource=c("bam"),
    blockTypeID="GeneS.Ensembl.Hsapiens.v86")
saveRDS(resOut, file.path(pathOut, "resOut.rds"))
```

3: Examine the value of the ancestry inference as follows

3) A: The inferred ancestry and the optimal parameters. For the global ancestry inference using PCA followed by nearest neighbor classification these parameters are D, the number of the top principal directions retained, and k, the number of nearest neighbors [20].

```
print(resOut$Ancestry)
```

3) B: Visualize the RAIDS performance for the synthetic data, as a function of *D* and *k* (Fig. 2).

```
createAUROCGraph(dfAUROC=resOut$paraSample$dfAUROC,
    title=" Encode ENCFF001 RNA")
```

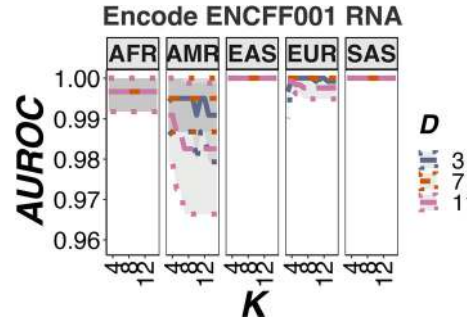


Fig. 2 The area under receiver operating characteristic (AUROC) measure of performance for global ancestry inference, as a function of the inference parameters *D* and *K* computed for each super-population (AFR, AMR, EAS, EUR, SAS)

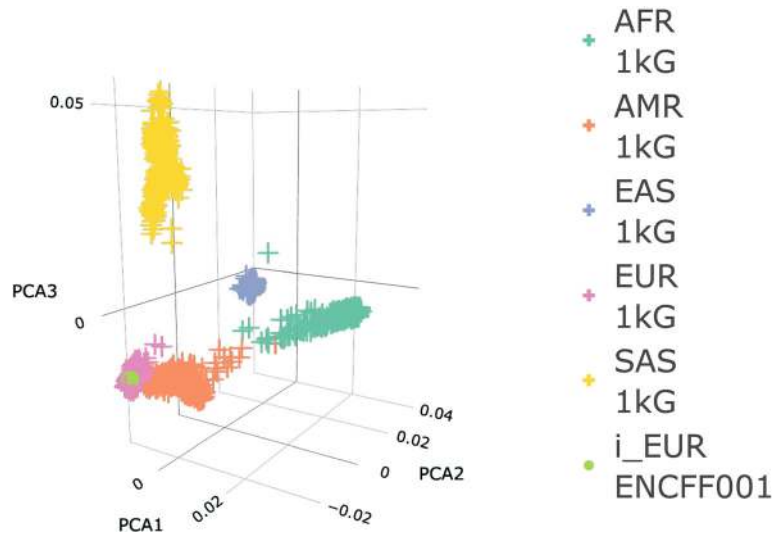


Fig. 3 The reference (1KG) profiles and a donor RNA-seq profile ENCFF001 (green circle) are projected onto the top 3 principal components of the reference data

Visualize the inference results in three top principal coordinates (Fig. 3).

```
if (! requireNamespace("GenomeInfoDb", quietly=TRUE)){
    install.packages("plotly")
    library(plotly)
}

dfPop <- getRefSuperPop(refGenotype)

eigenvect <- rbind(resOut$pcaSample$eigenvector.ref,
```

```
                    resOut$pcaSample$eigenvector)
rownames(eigenvect) <- c(row.names(resOut$pcaSample$eigenvec-
tor.ref),
                      row.names(resOut$pcaSample$eigenvector))

sampleInfo  <- data.frame(id=rownames(eigenvect),
                  superPop=c(dfPop[rownames(eigenvect[-1* nrow
(eigenvect),])],
                      paste0("i_", resOut$Ancestry$SuperPop)),
           Type=c(rep("1kG", nrow(eigenvect)-1), "ENCFF001"),
            stringsAsFactors=FALSE)

pcaSel3dAA <- data.frame(PCA1=eigenvect[,1], PCA2=eigenvect
[,2], PCA3=eigenvect[,3])

plot_ly(pcaSel3dAA, x=~PCA1, y=~PCA2, z=~PCA3,
        type='scatter3d', mode="markers",
        color=as.factor(sampleInfo$superPop),
        symbol=factor(sampleInfo$Type,
                      levels = c("ENCFF001","1kG"),
                      labels = c("ENCFF001","1kG")),
        symbols=c("circle","cross"))
```

2.4 Build a Population Reference Dataset (Optional)

The population reference input into RAIDS consists of two files: the reference genotype file and the reference annotation file. The reference genotype file contains, for each donor, to the population reference data, the genotypes at all positions in the genome, where the alternative allele frequency in a population is above a user-defined threshold. The reference annotation file tabulates, for each super-population, its haplotype blocks and the phase information for the reference. Both these files adhere to the genome data structure (GDS) format [32–34]. We use the gdsfmt [32] package to format both files.

2.4.1 Build the Reference Genotype File

Here we will explain how to construct the 1KG reference file as an example (*see* **Note 3**).

We will set up a directory structure for the reference, namely, a directory tree for the input:

```
workingDirectory
    data
        1000GGenotypeGRCh38      # PATHVCF1KG
        geno                     # PATHGENO
        info1kg
        testpopulationRef        # PATHOUT
        ldBlock                  # pathBlockPop
```

```
PATHVCF1KG=data/1000GGenotypeGRCh38
if [ ! -d $PATHVCF1KG ]
then
mkdir $PATHVCF1KG
fi
```

and two output directories:

```
PATHOUT=data
PATHGENO=${PATHOUT}/geno
PATHINFO=${PATHOUT}/info1kg

if [ ! -d $PATHGENO ]
then
mkdir $PATHGENO
fi
if [ ! -d data/testpopulationRef ]
then
mkdir data/testpopulationRef
fi
if [ ! -d $PATHINFO ]
then
mkdir $PATHINFO
fi
```

The genotypes of the donors to 1KG are defined in VCF-formatted files (one per chromosome) [35].

Currently, these files follow the GRCh38 build of the human genome [24].

The table header in the VCF file is as follows:

```
#INFO=<ID=DP,Number=1,Type=Integer,Description="Approximate
read depth; some reads may have been filtered">
#CHROM  POS     ID      REF     ALT     QUAL    FILTER  INFO
FORMAT  HG00096 HG00097 ...
```

where the first nine columns provide the sequence variant information, including the allele frequencies in the *INFO* column. The remaining columns contain the donor genotypes.

In addition, we shall need the donor genotype metadata file in the PED format [36] located at The International Genome Sample Resource (IGSR) [37].

Download the input files:

```
cd  $PATHVCF1KG
for i in `seq 1 22`
do
```

```
wget http://ftp.1000genomes.ebi.ac.uk/vol1/ftp/data_collections/1000_genomes_project/release/20181203_biallelic_SNV/ALL.chr${i}.shapeit2_integrated_v1a.GRCh38.20181129.phased.vcf.gz
wget http://ftp.1000genomes.ebi.ac.uk/vol1/ftp/data_collections/1000_genomes_project/release/20181203_biallelic_SNV/ALL.chr${i}.shapeit2_integrated_v1a.GRCh38.20181129.phased.vcf.gz.tbi
done
wget ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/technical/working/20130606_sample_info/20130606_g1k.ped
cd -
```

From this input, we first create separate genotype files for each 1KG donor (*see* **Note 4**).

```
FILE1KG=${PATHVCF1KG}/ALL.chr1.shapeit2_integrated_v1a.GRCh38.20181129.phased.vcf.gz
NF=$(($(zcat $FILE1KG|head -n 5000|grep "#CHROM"|grep -o $'\t'|wc -l) + 1))
TITLE=$(zcat $FILE1KG|head -n 5000|grep "#CHROM")
for j in `seq 10 $NF`
do
CURRENT=$(echo $TITLE|cut -d $' ' -f ${j})
echo ${CURRENT} > ${PATHGENO}/${CURRENT}.csv
done
for i in `seq 1 22`
do
FILE1KG=${PATHVCF1KG}/ALL.chr${i}.shapeit2_integrated_v1a.GRCh38.20181129.phased.vcf.gz

for j in `seq 10 $NF`
do
CURRENT=$(echo $TITLE|cut -d $' ' -f ${j})
zcat $FILE1KG|grep -v "#"|cut -d $'\t' -f ${j}|head -n 10000 >> ${PATHGENO}/${CURRENT}.csv
done
done

cd ${PATHGENO}
for j in `seq 10 $NF`
do
CURRENT=$(echo $TITLE|cut -d $' ' -f ${j})
bzip2 ${CURRENT}.csv
done
cd -
```

We also create a single variant information table which, in addition to columns describing the variants, provides the variant allele frequencies in each 1KG super-population.

```
echo  $'CHROM,POS,REF,ALT,AF,EAS_AF,EUR_AF,AFR_AF,AMR_AF,SA-
S_AF'>${PATHINFO}/snvAnnotation.csv
for i in `seq 1 22`
do
FILE1KG=${PATHVCF1KG}/ALL.chr${i}.shapeit2_integrated_v1a.
GRCh38.20181129.phased.vcf.gz

zcat $FILE1KG|grep -v "#"|cut -d $'\t' -f 1,2,4,5,8|perl -n -e
'@line=split("\t",$_);
@info=split(";", $line[4]);
my %indCur=();
foreach( @info){
@val=split("\=", $_);
$indexCur{$val[0]}=$val[1]};
print($line[0].",". $line[1].",". $line[2] . "," . $line[3].
",".
$indexCur{"AF"}.
",". $indexCur{"EAS_AF"}.
",". $indexCur{"EUR_AF"}.
",". $indexCur{"AFR_AF"}.
",". $indexCur{"AMR_AF"}.
",". $indexCur{"SAS_AF"} ."\n")' |head -n 10000 >>${PATHINFO}/
snvAnnotation.csv
done

cd ${PATHINFO}
bzip2 snvAnnotation.csv
cd -
```

Next, in an R session, we reformat the input metadata:

```
library(RAIDS)

pathGeno1Kg <- "data/1000GGenotypeGRCh38"
pathGeno <- "data/geno"
pathInfo <- "data/info1kg"
fileName <- "20130606_g1k.ped"
ped <- prepPed1KG(filePed=file.path(pathGeno1Kg,fileName),
    pathGeno=pathGeno)

saveRDS(ped, file.path(pathInfo, "ped1kg.rds"))
```

Only variants with a user-set minimum frequency in at least one super-population will be retained for further analysis. Here the minimum frequency is set to 0.01. The requisite function call is.

```
outSNVIndex <- "snvSel0.01.Index.rds"
outSNVSelected <- "snvSel0.01.rds"
generateMapSnvSel(cutOff=0.01,
    fileSNV=file.path(pathInfo, "snvAnnotation.csv.bz2"),
    fileSNPsRDS=file.path(pathInfo, outSNVIndex),
    fileFREQ=file.path(pathInfo, outSNVSelected))
```

, where `cutOff` is the minimum frequency; `pathSNVAnnotation` is the path to a variant information file (`snvAnnotation.csv.bz2` in this example); and `outSNVIndex` and `outSNVSelected` are names of the output files: the first containing the index for the population reference genotypes and the second containing the description of the variants retained.

We next create a reference genotype file required by RAIDS.

```
pathReference <- "data/testpopulationRef"
refGenotype <- file.path(pathReference, "matGeno1000gTest.
gds")

generateGDS1KG(pathGeno=pathGeno,
    filePedRDS=file.path(pathInfo, "ped1kg.rds"),
    fileSNVIndex=file.path(pathInfo, outSNVIndex),
    fileSNVSelected=file.path(pathInfo, outSNVSelected),
    fileNameGDS=refGenotype)
```

where `pathGeno` is the path to the directory containing all the reference genotype files; `filePedRDS`, `fileSNVIndex`, and `fileSNVSelected` are input file names; and `fileNameGDS` is the output file name.

Finally, we test all pairs of the reference donors for kinship and add a field in the reference genotype file identifying a subset of unrelated donors:

```
identifyRelativeRef(fileReferenceGDS=refGenotype,
    maf=0.05,
    thresh=2^(-11/2),
    fileIBD=file.path(pathInfo, "ibd1kg.rds"),
    filePart=file.path(pathInfo, "unrelated1kg.rds"))

addRef2GDS1KG(fileNameGDS=refGenotype,
    filePart=file.path(pathInfo, "unrelated1kg.rds"))
```

In the final form, the population reference structure is (*see* **Note 5**).

```
|--+ sample.id   { Str8 2548, 19.9K }
|--+ sample.annot   [ data.frame ] *
|  |--+ sex   { Str8 2548, 5.0K } #
|  |--+ pop.group   { Str8 2548, 10.0K }
|  |--+ superPop   { Str8 2548, 10.0K }
|  \--+ batch   { Float64 2548, 19.9K }
|--+ snp.id   { Str8 24516859, 223.2M }
|--+ snp.chromosome   { UInt16 24516859, 46.8M }
|--+ snp.position   { Int32 24516859, 93.5M }
|--+ snp.allele   { Str8 24516859, 93.5M }
|--+ snp.AF   { PackedReal24 24516859, 70.1M }
|--+ snp.EAS_AF   { PackedReal24 24516859, 70.1M } #
|--+ snp.EUR_AF   { PackedReal24 24516859, 70.1M } #
|--+ snp.AFR_AF   { PackedReal24 24516859, 70.1M } #
|--+ snp.AMR_AF   { PackedReal24 24516859, 70.1M } #
|--+ snp.SAS_AF   { PackedReal24 24516859, 70.1M } #
|--+ genotype   { Bit2 24516859x2548, 14.5G }
\--+ sample.ref   { Bit1 2548, 319B }
```

The fields are as follows:

`sample.id`: a character string used as unique identifier for each sample.

`sample.annot`: a data.frame object where each row corresponds to a donor containing those columns:

`sex`: a character string used as identifier of the sex of the donor.

`pop.Group`: a character string representing the subpopulation ancestry of the donor (e.g., GBR).

`superPop`: a character string representing the super-population ancestry of the donor.

`batch`: an integer field reserved for future use (e.g., addition of donors to the reference collection).

`snp.id`: a character string used as a unique identifier for each variant.

`snp.chromosome`: an integer denoting the chromosome for the variant.

`snp.position`: genomic coordinate of the variant.

`snp.allele`: a character string representing the reference and alternative alleles.

`snp.AF`: a numeric value for the frequency of the alternative allele in the reference collection.

snp.[SPR]_AF: a numeric value between 0 and 1 representing the allele frequency of the alternative allele in the super-population SPR.

sample.ref: an integer indicating whether the reference donor is retained (=1) or removed following the kinship test.

2.4.2 Build the Reference Annotation File

The reference annotation file is created by the following function call:

```
refAnnotation <- file.path(pathReference, "matAnnot1000gTest.
gds")

generatePhaseRef(fileReferenceGDS=refGenotype,
    pathGeno=pathGeno,
    fileSNVIndex=file.path(pathInfo, outSNVIndex),
    fileReferenceAnnotGDS=refAnnotation)
```

where `fileReferenceGDS`, `pathGeno`, and `fileSNVIndex` are character strings containing the names of population reference file, the index file for the population reference genotypes, and the path to the directory containing all the reference genotype files, respectively; `fileReferenceAnnotGDS` is a character string for the reference annotation file name.

At this point, the annotation file contains haplotype phasing information for each donor.

Next, for each super-population, we add to the annotation file a partition of the genome into blocks of variants in linkage disequilibrium (LD):

```
pathBlockPop <- "data/ldBlock"
addBlockFromDetFile( fileReferenceGDS=refGenotype,
    gdsRefAnnotFile=refAnnotation,
    pathBlock=pathBlockPop,
    superPop="AFR")
```

where `fileReferenceGDS` and `gdsRefAnnotFile` are character strings containing the names of population reference file and the reference annotation file name and `pathBlockPop` is a character string for a directory containing, for each chromosome, a file with a table for super-population-specific LD blocks. The file names must be of the form `*.chr[chrNum].blocks.det`, where `chrNum` is an integer between 1 and 22. The LD block table must have at least the following three columns:

CHR: the chromosome.

BP1: the start position of the block.

BP2: the end position of the block.

(*see* **Note 6**);

and the `superPop` argument is a character string containing the name of the super-population.

Finally, in order to enable inference from RNA-seq data, we add to the annotation file a transcript-oriented partition of the genome. For all transcribed regions annotated in the `EnsDb.Hsapiens`, we provide the transcription start and end coordinates. The remainder of the genome is partitioned into windows of equal size.

```
if (! requireNamespace("EnsDb.Hsapiens.v86", quietly=TRUE)){
    BiocManager::install("EnsDb.Hsapiens.v86")
    library(EnsDb.Hsapiens.v86)
}

edb <- EnsDb.Hsapiens.v86::EnsDb.Hsapiens.v86

addGeneBlockRefAnnot(fileReferenceGDS=refGenotype,
    gdsRefAnnotFile=refAnnotation,
    winSize=10000,
    ensDb=edb,
    suffixBlockName="Ensembl.Hsapiens.v86")
```

where

`winSize`: an integer window size for the partition of the genome outside the annotated regions.

`edb`: an object available from Bioconductor.

`suffixBlockName`: a character string containing the name of the edb object.

In the final form, the reference annotation file contains the following structure:

```
--+ phase   { Bit2 24516859x2548 LZ4_ra(35.0%), 5.1G }
|--+ block.annot   [ data.frame ] *
|  |--+ block.id   { Str8 7, 123B }
|  \--+ block.desc   { Str8 7, 388B }
\--+ block   { Int32 24516859x7 LZ4_ra(3.60%), 23.6M }
```

The field descriptions are as follows:

`phase`: an integer representing the phase of the SNVs in the Population Annotation GDS file; 0 means the first allele is a reference; 1 means the first allele is the alternative and 3 means unknown. The first allele combined with the genotype of the variant determines the phase for a biallelic variant. The variants in `phase` are in the same order than the variants in the Population reference dataset.

block.annot: a data.frame object containing the following columns.

block.id: a character string representing an identifier of block group. A block can be linkage disequilibrium block relative to a population or a gene.

block.desc: a character string describing the block group.

block: a matrix of integer values where each row represents a SNV in the same order as that of the variants in Population reference dataset. The columns are the block groups described in `block.annot`. Each element in the matrix is a block identifier.

3 Future Expansion and Refinement of RAIDS

Future versions of RAIDS will include methods for in-depth inference of genetic ancestry, specifically inference of ancestral admixtures and local ancestry. RAIDS will also be enabled to handle sequence data from additional molecular protocols, such as ATAC-seq, cytosine-conversion assays, and low-coverage whole-genome sequences.

4 Notes

Note 1: In RAIDS, the function `snpgdsLDpruning()` from the Bioconductor package SNPRelate is invoked to prune variants in order to reduce the linkage disequilibrium.

Note 2: We tune the inference parameters to optimal performance on a set of synthetic profiles. A strict assessment of performance at the optimum would require another set of synthetic profiles. Instead, we examine whether the inference performance remains nearly unchanged in a range of parameters near the optimum.

Note 3: We restricted the reference to the first 10,000 variants for accelerate the example. If you want to generate all the reference, you must remove the 2 "`|head -n 10000`" in shell scripts.

Note 4: The genotype file name must be of the form "`donorID.csv.bz2`," where the string "`donorID`" is unique for each donor to the reference. The first line of the genotype file contains the name of the donor, and the subsequent lines each contain a genotype in the format "`Allele1/Allele2`" if no phasing is available, "`Allele1|Allele2`" otherwise.

Note 5: The fields for allele frequencies must have names of the form "`snp.[SPR]_AF`" where `SPR` is the super-population reference code, e.g., EAS for the East Asian super-population.

These fields correspond to the values present in the field `superPop` of the population reference structure file.

Note 6: We recommend Plink [38] software package for computing LD blocks [39]. Computing LD blocks genome-wide is computationally demanding. An example of Plink output is provided at https://labshare.cshl.edu/shares/krasnitzlab/aicsPaper/ldBlock2024.09.10.tar.gz.

Acknowledgments

We acknowledge support by the National Institute of Health (awards 1U01CA289357–01 to A.K.; P30CA45508, P20CA192996, U01CA224013, U01CA210240, R01CA188134, R01CA249002, and R01CA229699 to D.A.T.); the New York Genome Center Polyethnic-1000 Project (award 33,350,211, A.K. and D.A.T.) and MacMillan Center for the Study of the Non-Coding Genome (awards 36,620,111 and 36,620,112); Simons Foundation awards 519,054 (A.K.) and 552,716 (D.A.T.), the Simons Center for Quantitative Biology at Cold Spring Harbor Laboratory (A.K.); the Lustgarten Foundation (LF; awards 33,470,211, 36,900,101, 36,900,201), where D.A.T. is a Distinguished Scholar and Director of LF-designated laboratory; and the Pershing Square Foundation, William Ackman, and Neri Oxman (all D.A.T.).

References

1. Siegel RL, Miller KD, Jemal A (2020) Cancer statistics, 2020. CA Cancer J Clin 70(1):7–30. https://doi.org/10.3322/caac.21590
2. Cronin KA, Lake AJ, Scott S, Sherman RL, Noone AM, Howlader N, Henley SJ, Anderson RN, Firth AU, Ma J, Kohler BA, Jemal A (2018) Annual Report to the Nation on the Status of Cancer, part I: national cancer statistics. Cancer 124(13):2785–2800. https://doi.org/10.1002/cncr.31551
3. Ashktorab H, Kupfer SS, Brim H, Carethers JM (2017) Racial disparity in gastrointestinal cancer risk. Gastroenterology 153(4):910–923. https://doi.org/10.1053/j.gastro.2017.08.018
4. Huang BZ, Stram DO, Le Marchand L, Haiman CA, Wilkens LR, Pandol SJ, Zhang ZF, Monroe KR, Setiawan VW (2019) Interethnic differences in pancreatic cancer incidence and risk factors: the Multiethnic Cohort. Cancer Med 8(7):3592–3603. https://doi.org/10.1002/cam4.2209
5. Tan DS, Mok TS, Rebbeck TR (2016) Cancer genomics: diversity and disparity across ethnicity and geography. J Clin Oncol 34(1):91–101. https://doi.org/10.1200/JCO.2015.62.0096
6. Mahal BA, Alshalalfa M, Kensler KH, Chowdhury-Paulino I, Kantoff P, Mucci LA, Schaeffer EM, Spratt D, Yamoah K, Nguyen PL, Rebbeck TR (2020) Racial differences in genomic profiling of prostate cancer. N Engl J Med 383(11):1083–1085. https://doi.org/10.1056/NEJMc2000069
7. Carrot-Zhang J, Chambwe N, Damrauer JS, Knijnenburg TA, Robertson AG, Yau C, Zhou W, Berger AC, Huang KL, Newberg JY, Mashl RJ, Romanel A, Sayaman RW, Demichelis F, Felau I, Frampton GM, Han S, Hoadley KA, Kemal A, Laird PW, Lazar AJ, Le X, Oak N, Shen H, Wong CK, Zenklusen JC, Ziv E, Cancer Genome Atlas Analysis N, Cherniack AD, Beroukhim R (2020) Comprehensive analysis of genetic ancestry and its molecular correlates in cancer. Cancer Cell 37(5):639–654 e636. https://doi.org/10.1016/j.ccell.2020.04.012
8. Yuan J, Hu Z, Mahal BA, Zhao SD, Kensler KH, Pi J, Hu X, Zhang Y, Wang Y, Jiang J,

Li C, Zhong X, Montone KT, Guan G, Tanyi JL, Fan Y, Xu X, Morgan MA, Long M, Zhang Y, Zhang R, Sood AK, Rebbeck TR, Dang CV, Zhang L (2018) Integrated analysis of genetic ancestry and genomic alterations across cancers. Cancer Cell 34(4):549–560 e549. https://doi.org/10.1016/j.ccell.2018.08.019

9. Sinha S, Mitchell KA, Zingone A, Bowman E, Sinha N, Schäffer AA, Lee JS, Ruppin E, Ryan BM (2020) Higher prevalence of homologous recombination deficiency in tumors from African Americans versus European Americans. Nat Cancer 1(1):112–121. https://doi.org/10.1038/s43018-019-0009-7
10. Bhatnagar B, Kohlschmidt J, Mrozek K, Zhao Q, Fisher JL, Nicolet D, Walker CJ, Mims AS, Oakes C, Giacopelli B, Orwick S, Boateng I, Blachly JS, Maharry SE, Carroll AJ, Powell BL, Kolitz JE, Stone RM, Byrd JC, Paskett ED, de la Chapelle A, Garzon R, Eisfeld AK (2021) Poor survival and differential impact of genetic features of black patients with acute myeloid leukemia. Cancer Discov 11(3):626–637. https://doi.org/10.1158/2159-8290.CD-20-1579
11. Carrot-Zhang J, Soca-Chafre G, Patterson N, Thorner AR, Nag A, Watson J, Genovese G, Rodriguez J, Gelbard MK, Corrales-Rodriguez L, Mitsuishi Y, Ha G, Campbell JD, Oxnard GR, Arrieta O, Cardona AF, Gusev A, Meyerson M (2021) Genetic ancestry contributes to somatic mutations in lung cancers from admixed Latin American populations. Cancer Discov 11(3):591–598. https://doi.org/10.1158/2159-8290.CD-20-1165
12. Weiner AB, Vidotto T, Liu Y, Mendes AA, Salles DC, Faisal FA, Murali S, McFarlane M, Imada EL, Zhao X, Li Z, Davicioni E, Marchionni L, Chinnaiyan AM, Freedland SJ, Spratt DE, Wu JD, Lotan TL, Schaeffer EM (2021) Plasma cells are enriched in localized prostate cancer in Black men and are associated with improved outcomes. Nat Commun 12(1):935. https://doi.org/10.1038/s41467-021-21245-w
13. Zaaijer S, Capes-Davis A (2021) Ancestry matters: building inclusivity into preclinical study design. Cell 184(10):2525–2531. https://doi.org/10.1016/j.cell.2021.03.041
14. Nugent A, Conatser KR, Turner LL, Nugent JT, Sarino EMB, Ricks-Santi LJ (2019) Reporting of race in genome and exome sequencing studies of cancer: a scoping review of the literature. Genet Med 21(12):2676–2680. https://doi.org/10.1038/s41436-019-0558-2
15. Polubriaginof FCG, Ryan P, Salmasian H, Shapiro AW, Perotte A, Safford MM, Hripcsak G, Smith S, Tatonetti NP, Vawdrey DK (2019) Challenges with quality of race and ethnicity data in observational databases. J Am Med Inform Assoc 26(8–9):730–736. https://doi.org/10.1093/jamia/ocz113
16. Pritchard JK, Stephens M, Donnelly P (2000) Inference of population structure using multilocus genotype data. Genetics 155(2):945–959
17. Price AL, Patterson NJ, Plenge RM, Weinblatt ME, Shadick NA, Reich D (2006) Principal components analysis corrects for stratification in genome-wide association studies. Nat Genet 38(8):904–909. https://doi.org/10.1038/ng1847
18. Alexander DH, Novembre J, Lange K (2009) Fast model-based estimation of ancestry in unrelated individuals. Genome Res 19(9):1655–1664. https://doi.org/10.1101/gr.094052.109
19. Frampton GM, Fichtenholtz A, Otto GA, Wang K, Downing SR, He J, Schnall-Levin M, White J, Sanford EM, An P, Sun J, Juhn F, Brennan K, Iwanik K, Maillet A, Buell J, White E, Zhao M, Balasubramanian S, Terzic S, Richards T, Banning V, Garcia L, Mahoney K, Zwirko Z, Donahue A, Beltran H, Mosquera JM, Rubin MA, Dogan S, Hedvat CV, Berger MF, Pusztai L, Lechner M, Boshoff C, Jarosz M, Vietz C, Parker A, Miller VA, Ross JS, Curran J, Cronin MT, Stephens PJ, Lipson D, Yelensky R (2013) Development and validation of a clinical cancer genomic profiling test based on massively parallel DNA sequencing. Nat Biotechnol 31(11):1023–1031. https://doi.org/10.1038/nbt.2696
20. Belleau P, Deschenes A, Chambwe N, Tuveson DA, Krasnitz A (2023) Genetic ancestry inference from cancer-derived molecular data across genomic and transcriptomic platforms. Cancer Res 83(1):49–58. https://doi.org/10.1158/0008-5472.CAN-22-0682
21. Belleau P, Deschênes A, Tuveson DA, Krasnitz A (2023) Accurate inference of genetic ancestry from cancer sequences. Bioconductor. https://doi.org/10.18129/B9.bioc.RAIDS
22. Lowy-Gallego E, Fairley S, Zheng-Bradley X, Ruffier M, Clarke L, Flicek P, Genomes Project C (2019) Variant calling on the GRCh38 assembly with the data from phase three of the 1000 Genomes Project. Wellcome Open Res 4:50. https://doi.org/10.12688/wellcomeopenres.15126.2
23. GATK Team (2024) Genome Analysis Toolkit. https://gatk.broadinstitute.org/hc/en-us
24. Schneider VA, Graves-Lindsay T, Howe K, Bouk N, Chen HC, Kitts PA, Murphy TD, Pruitt KD, Thibaud-Nissen F, Albracht D, Fulton RS, Kremitzki M, Magrini V, Markovic C,

McGrath S, Steinberg KM, Auger K, Chow W, Collins J, Harden G, Hubbard T, Pelan S, Simpson JT, Threadgold G, Torrance J, Wood JM, Clarke L, Koren S, Boitano M, Peluso P, Li H, Chin CS, Phillippy AM, Durbin R, Wilson RK, Flicek P, Eichler EE, Church DM (2017) Evaluation of GRCh38 and de novo haploid genome assemblies demonstrates the enduring quality of the reference assembly. Genome Res 27(5):849–864. https://doi.org/10.1101/gr.213611.116

25. GATK Team (2024) How to map and clean up short read sequence data efficiently. Broad Institute. https://gatk.broadinstitute.org/hc/en-us/articles/360039568932%2D%2DHow-to-Map-and-clean-up-short-read-sequence-data-efficiently
26. GATK Team (2024) RNAseq short variant discovery (SNPs + Indels). https://gatk.broadinstitute.org/hc/en-us/articles/360035531192-RNAseq-short-variant-discovery-SNPs-Indels
27. Danecek P, Bonfield JK, Liddle J, Marshall J, Ohan V, Pollard MO, Whitwham A, Keane T, McCarthy SA, Davies RM, Li H (2021) Twelve years of SAMtools and BCFtools. Gigascience 10(2). https://doi.org/10.1093/gigascience/giab008
28. Luo Y, Hitz BC, Gabdank I, Hilton JA, Kagda MS, Lam B, Myers Z, Sud P, Jou J, Lin K, Baymuradov UK, Graham K, Litton C, Miyasato SR, Strattan JS, Jolanki O, Lee JW, Tanaka FY, Adenekan P, O'Neill E, Cherry JM (2020) New developments on the Encyclopedia of DNA Elements (ENCODE) data portal. Nucleic Acids Res 48(D1):D882–D889. https://doi.org/10.1093/nar/gkz1062
29. Frankish A, Diekhans M, Ferreira AM, Johnson R, Jungreis I, Loveland J, Mudge JM, Sisu C, Wright J, Armstrong J, Barnes I, Berry A, Bignell A, Carbonell Sala S, Chrast J, Cunningham F, Di Domenico T, Donaldson S, Fiddes IT, Garcia Giron C, Gonzalez JM, Grego T, Hardy M, Hourlier T, Hunt T, Izuogu OG, Lagarde J, Martin FJ, Martinez L, Mohanan S, Muir P, Navarro FCP, Parker A, Pei B, Pozo F, Ruffier M, Schmitt BM, Stapleton E, Suner MM, Sycheva I, Uszczynska-Ratajczak B, Xu J, Yates A, Zerbino D, Zhang Y, Aken B, Choudhary JS, Gerstein M, Guigo R, Hubbard TJP, Kellis M, Paten B, Reymond A, Tress ML, Flicek P (2019) GENCODE reference annotation for the human and mouse genomes. Nucleic Acids Res 47(D1):D766–D773. https://doi.org/10.1093/nar/gky955
30. Dobin A, Davis CA, Schlesinger F, Drenkow J, Zaleski C, Jha S, Batut P, Chaisson M, Gingeras TR (2013) STAR: ultrafast universal RNA-seq aligner. Bioinformatics 29(1):15–21. https://doi.org/10.1093/bioinformatics/bts635
31. Dobin A (2024) STAR. https://github.com/alexdobin/STAR?tab=readme-ov-file
32. Zheng X, Gogarten S, Gailly J-L, Adler M, Collet Y (2024) R Interface to CoreArray Genomic Data Structure (GDS) Files. https://bioconductor.org/packages/release/bioc/html/gdsfmt.html
33. Zheng X, Gogarten SM, Lawrence M, Stilp A, Conomos MP, Weir BS, Laurie C, Levine D (2017) SeqArray-a storage-efficient high-performance data format for WGS variant calls. Bioinformatics 33(15):2251–2257. https://doi.org/10.1093/bioinformatics/btx145
34. Zheng X, Levine D, Shen J, Gogarten SM, Laurie C, Weir BS (2012) A high-performance computing toolset for relatedness and principal component analysis of SNP data. Bioinformatics 28(24):3326–3328. https://doi.org/10.1093/bioinformatics/bts606
35. Schneider VA, Graves-Lindsay T, Howe K, Bouk N, Chen HC, Kitts PA, Murphy TD, Pruitt KD, Thibaud-Nissen F, Albracht D, Fulton RS, Kremitzki M, Magrini V, Markovic C, McGrath S, Steinberg KM, Auger K, Chow W, Collins J, Harden G, Hubbard T, Pelan S, Simpson JT, Threadgold G, Torrance J, Wood JM, Clarke L, Koren S, Boitano M, Peluso P, Li H, Chin CS, Phillippy AM, Durbin R, Wilson RK, Flicek P, Eichler EE, Church DM (2018) 1000 genomes 20181203 biallelic SNV. http://ftp.1000genomes.ebi.ac.uk/vol1/ftp/data_collections/1000_genomes_project/release/20181203_biallelic_SNV
36. IGSR (2013) Technical note on the pedigree file for the 1000 Genomes project. https://ftp.1000genomes.ebi.ac.uk/vol1/ftp/technical/working/20130606_sample_info/README_20130606_sample_info
37. IGSR (2013) Pedigree file for the 1000 Genomes project. ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/technical/working/20130606_sample_info/20130606_g1k.ped
38. Purcell S, Neale B, Todd-Brown K, Thomas L, Ferreira MA, Bender D, Maller J, Sklar P, de Bakker PI, Daly MJ, Sham PC (2007) PLINK: a tool set for whole-genome association and population-based linkage analyses. Am J Hum Genet 81(3):559–575. https://doi.org/10.1086/519795
39. Chang C, Chow C, Vattikuti S, Tellier L, Lee J (2024) Plink 1.9 Haplotype block estimation documentation. https://www.cog-genomics.org/plink/1.9/ld#blocks

Chapter 9

Pruning-Assisted Modeling of Network Graph Connectivity from Spatial Transcriptomic Data

Antara Biswas and Subhajyoti De

Abstract

Functional interactions within and between different types of somatic cells are crucial for executing complex organ-level biological processes in multicellular organisms. Spatial transcriptomic technologies have allowed for high throughput characterization of cell communities and associated cellular processes in the tissue contexts. However, analytical resources for characterization and quantitative inference of spatial interactions among somatic cells that can potentially impact complex biological functions in tissue microenvironment are still limited. Here, we describe a framework to use network graph-based spatial statistical models on spatially annotated molecular data to gain insights into cellular relationship and connectivity in the local tumor microenvironment and evaluate the effects of network graph connectivity on the model inference.

Key words Network, Community connectivity, Pruning, Trimming, Spatial transcriptomics, Tumor microenvironment, Quantitative inference

1 Introduction

In multicellular organisms, somatic cells are organized into different tissue layers, where functional interactions among the cells lead to complex, organ-level functions [1, 2]. Examining gene expression in somatic cell types along with their respective cellular neighborhoods in the tissue layers provides information about the broader context of biological processes and their interrelation in the tissue microenvironment—offering valuable insights and interpretation of their functional significance in tissues, organs, and general body system [3–5]. Emerging technologies such as the high throughput spatial transcriptomics (ST) approaches have allowed for capture of gene expression profiles at specific, spatially annotated spots in tissues at unprecedented resolution, enabling a systematic assessment of the patterns of spatial connectedness of

Alexander Krasnitz and Pascal Belleau (eds.), *Cancer Bioinformatics*, Methods in Molecular Biology, vol. 2932,
https://doi.org/10.1007/978-1-0716-4566-6_9,

cellular transcriptomes, and also have prompted the development of numerous statistical models for characterizing the makeup of highly heterogeneous tissue samples [5–9].

While substantial work has been done to detect somatic cell types and their activities in the tissue contexts using spatial transcriptomics, the efforts to statistically model the spatial aspects of the community structure of cell types and connectivity of cellular processes are still at an earlier stage [10–12]. Appropriate modeling of these attributes can shed light into complex biological processes in the tissue microenvironment and their deregulation in diseases such as cancer. In previous work [13, 14], we presented some results from a network graph model built to capture the characteristics of spatial heterogeneity. In this work, we provide more details on the network science underpinning the model and discuss the strategies for model optimization by removing redundancy in connectivity from a dense network. Specifically, this chapter describes the methodological details of a neighborhood graph-based approach to analyze spatial transcriptomic data and focuses on the community structure of cell types and connectivity of cellular processes and examines strategies to prune the neighborhood graph at different levels based on cellular makeups, without affecting topology of the spatial network. This particularly highlights how the functionalities offered by the analyses described here address the challenges of spatial data setting and can be widely used to assess the effects of the cell-type and proximity-based trimming strategies to capture the fundamental attributes of cellular relationship and local patterns in the cancer microenvironment.

2 Materials

The R packages eSDM, spdep,adegraphics, adespatial, ade4, sp., adegenet, adegraphics, gstat, raster, spatialreg, geometry, lsa, sp., and gstat were used for manipulating, modelling, representation and statistical analysis tools for spatial data analysis [15, 16]. Data processing relied heavily on the Tidyverse v1.3.2 R packages (https://www.tidyverse.org/). All statistical analyses were performed using R version 4.2.3 (https://www.r-project.org/). In the following sections, we focus on the fundamentals of analysis and trimming techniques. Detailed code used for performing relevant analyses is available from our GitHub directory (https://github.com/sjdlabgroup/BLCA-resources). In this chapter, we will use the terms pruning and trimming interchangeably.

3 Methods

3.1 Spatial Collection and Initial Wet Lab Processing

Deidentified human bladder urothelial carcinoma samples were obtained from Rutgers Cancer Institute of New Jersey Biorepository under an IRB approved protocol (Pro2019002924, PI: De). A typical spatial transcriptomic sequencing workflow involves initial tissue preparation, capturing and library preparation, sequencing, raw data processing, and downstream analyses (Fig. 1a). In this study, we used the Visium platform from 10x Genomics to perform spatial transcriptomic sequencing. 5 μm tissue sections were placed on the Visium Spatial Gene Expression Slide for FFPE, hybridized, and prepared for sequencing according to the manufacturer's protocols. The slides were then used with Visium Spatial Gene Expression for FFPE User Guide (10X Genomics, CG000407) to generate Visium Spatial Gene Expression-FFPE libraries and sequenced on Illumina NovaSeq S4 300 cycle. Using this approach, we profiled tissue slices from four bladder cancer patients (S1-S4), as reported elsewhere [14], and here we discuss the modeling of spatial processes from this dataset using the network graph approach. Samples included Sample 1 (S1), a high-grade invasive urothelial transitional cell carcinoma with lymph node metastasis but no distant metastasis; Sample 2 (S2), a high-grade invasive localized urothelial transitional cell carcinoma without lymph

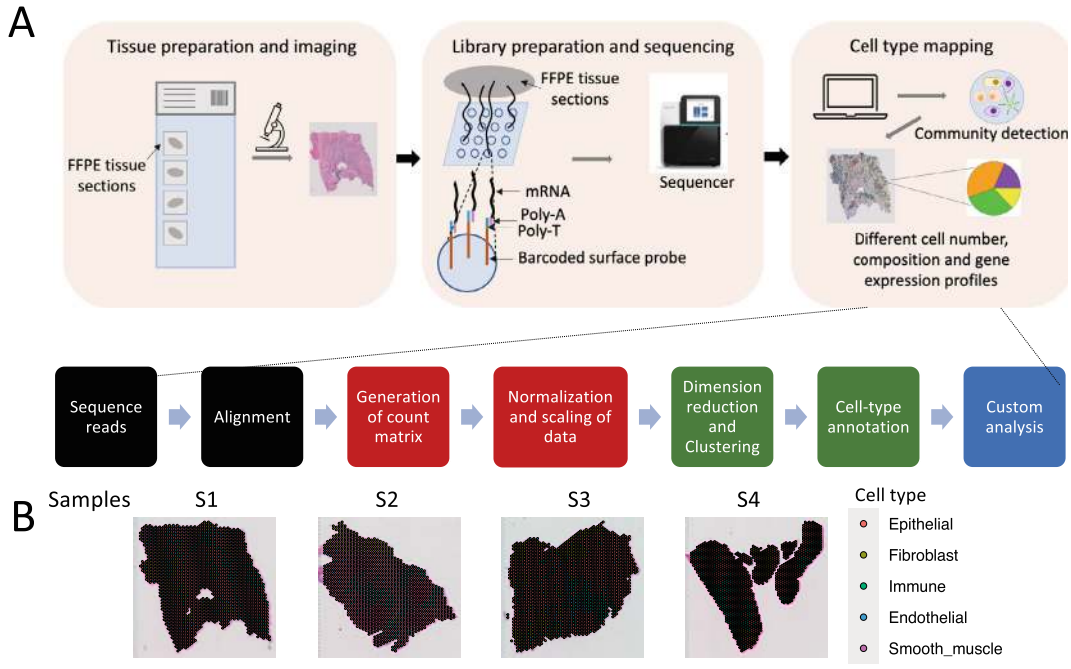


Fig. 1 (**a**) Schematic diagram of integral components of the spatial transcriptomics workflow, including key analysis steps. (**b**) Spatial composition of cell types in the four spatial transcriptomics samples, S1–S4

node or distant metastasis; Sample 3 (S3), a high-grade noninvasive papillary urothelial carcinoma without lymph node or distant metastasis; and Sample 4 (S4), a high-grade invasive urothelial transitional cell carcinoma with squamous differentiation and negative for lymph node or distant metastasis [14].

3.2 Processing and Initial Analysis of Spatial Transcriptomic Data to Examine Spatial Heterogeneity in Tumor and Non-tumor Cells in Bladder Cancer

As indicated elsewhere [14], the sequence data (FASTQ files) were processed using Space Ranger (v2.0.1) pipeline to align transcriptomic reads to the human reference genome (GRCh38), map them to the microscopic images of the tissue samples from which the reads were obtained and generate feature barcode matrices. Feature-barcode matrices and associated H&E images for each sample were imported into the R package "Seurat" (v4.3.0) for normalization, quality control, batch effect correction, dimensionality reduction, clustering, and cell-type estimation [17, 18] (Fig. 1b; *see* **Notes 1** and **2**).

3.3 Modeling Spatial Transcriptomic Data Using Neighborhood Connectivity Graph

We used the network graph-based approach to model the biological interactions in the spatial transcriptomic dataset and assessed the effects of network connectivity on the key inferences [13]. A neighborhood connectivity graph describes pairwise relationships between two or more nodal entities in a connected network. Spatial transcriptomic data are typically presented as nodal features (e.g., proportions of different cell types, pathway scores, etc.) at each regularly spaced spatially profiled spots in a tissue section (Fig. 2a). In the basic model, each spatially annotated data point from a tissue sample is a node, and adjacent nodes are connected by an edge—which enables statistical modeling of interactions among the adjacent nodes. For more complex models, it is represented as a collection of nodes connected by edges, where the set of neighbors of a given node is the node's neighborhood and the number of its neighbors is its connectivity.

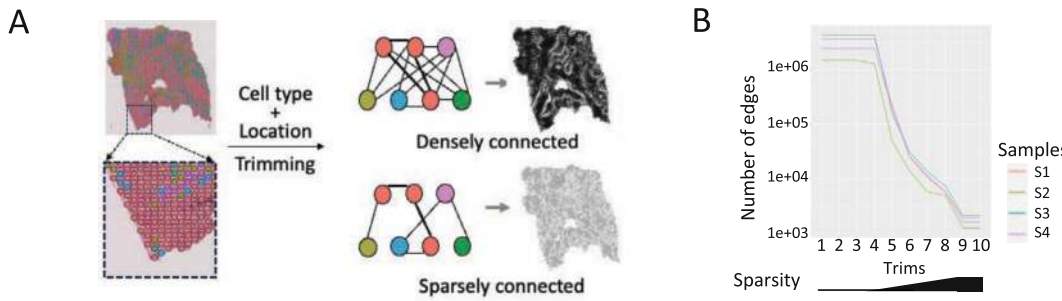


Fig. 2 (**a**) A schematic representation of construction and trimming of connections in neighborhood graph. As shown here for sample S1, using gene expression and spatial coordinate matrices from spatial transcriptomics platform, spatial network is constructed, which consists of two components—nodes (spots) and edges (connecting spots)—wherein edge thickness is determined by the attributes of the nodes (spatial annotation of spots) and the structure of the graph. (**b**) The number of edges changing with trimming of neighborhood graphs for samples S1–S4 from most dense to most sparse

3.4 Network Graph Trimming Strategy

Considering the spatially prominent features of tissue architectures, it is apparent that a majority of the tissue-level biological processes (i) depend on the local cell-type composition, (ii) show weak correspondence between dissimilar tissue layers, and (iii) their effects decay over distance within and across tissues. Therefore, one may argue that uniform nodal connectivity based on spatial adjacency alone may not be adequate and utility of alternative connectivity patterns should be evaluated. Starting with a fully connected network graph, we considered strategies to systematically trim the edges. All edges in the neighborhood connectivity graph had equal weight before network trimming. Since most of the tissue-level biological processes depend on distance and cellular community structures, we trimmed the network based on both geographic distance between the nodes in the graph and also their cell-type compositions. Network trimming, based on weights of network connection, was performed using a min-max module. We first computed pairwise Euclidean distances and cosine similarity values in terms of cell-type composition, between all the nodes in a graph. Here we used a product of cosine similarity of cell spots based on cell-type composition and inverse distance matrix of spatial spots to construct a spatial weight matrix, $\Psi = \{w_{ij}; 1 \le i, j \le N\}$, described as

$$w_{ij} = c_{ij} \times d_{ij},$$

where c_{ij} is cosine similarity value and d_{ij} is inverse distance between the i and j-th node and N is the number of spots. For each sample, the network graph based on the spatial transcriptomic data was trimmed to generate sparse network submatrices by network disassembly using edge removal. It was trimmed in such a way that the edges between adjacent nodes in the graph were progressively eliminated based on increasingly higher cutoff for w, which reflects increasingly higher threshold for similarity in cell-type composition as well as spatial proximity of the spots in the 10X spatial transcriptomic data (Fig. 2a, b). In simpler terms, $w_{ij} \to 0$ when the distance between the nodes is large or their cell-type compositions are fundamentally different. In contrast, $w_{ij} \to 1$ when the nodes are adjacent and also their cell-type compositions are very similar, reflecting homologous cellular neighborhood. We binned w_{ij} into ten-sequence vector and chose the sequence partitions as possible values of w'_{cutoff}. To trim the network, we tested different w'_{cutoff} and retained the edges between the nodes i and j if $w_{ij} >= w'_{\text{cutoff}}$ (Fig. 3; *see* **Notes 3** and **4**).

3.5 Calculation of Different Spatial Statistics Using Network Graph

When cells are mapped to their spatial context, they often exhibit some degree of spatial relatedness at some scale. One popular measure of spatial autocorrelation is the Moran's index (Moran's I for short) coefficient. Moran's I is an index for measuring spatial

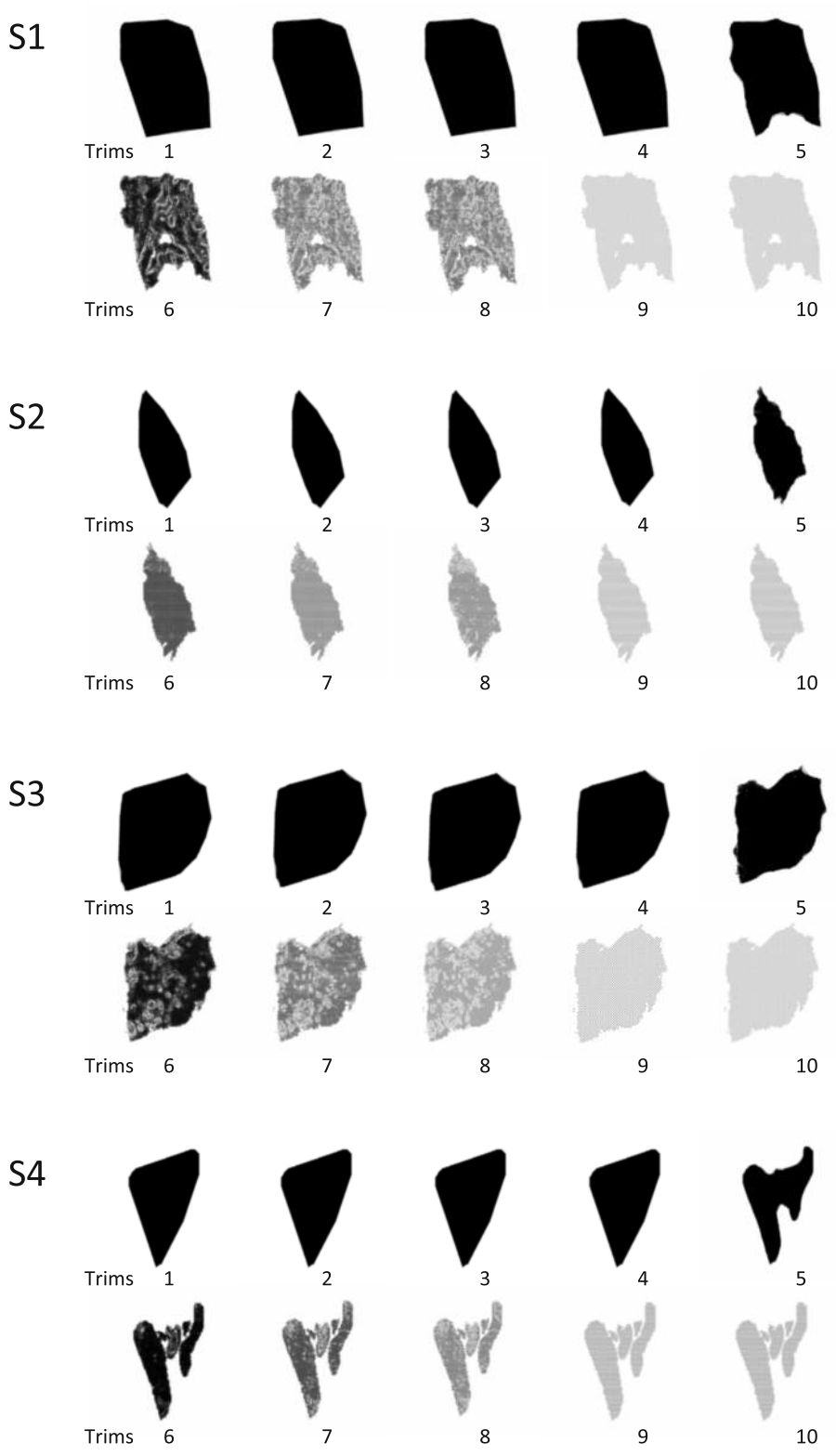


Fig. 3 Spatial network graphs for samples S1–S4, trimmed by using similarity/proximity measures between the nodes, wherein the spatial information is considered as a spatial weight matrix and the trimming ranges, in a ten-sequence vector, from minima (most dense graph) to maxima (most sparse graph) of the weight matrix

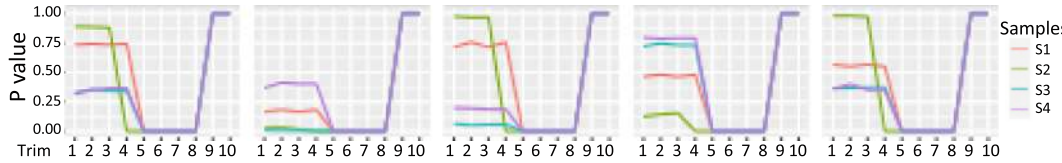


Fig. 4 Spatial analysis showing joint variation in spatial localization of the cell types in the four samples, S1–S4, with trimming, wherein *P* values associated with Moran's *I* statistic are shown

autocorrelation of a given feature considering its values at different spatial locations, as in this case, a phenotype score of spots in a tissue microenvironment. It is the correlation coefficient indicating the relationship for a variable (e.g., proportion of cell type) between the neighboring nodes. Given a set of features and associated attributes, Moran's I evaluates whether the pattern expressed is clustered, dispersed, or random. We used the moran.randtest function, which is based on the moran.mc function of the spdep package, using a published approach [13]. Moran's spatial autocorrelation can be expressed as

$$I = \frac{N}{W\sum_k (x_k - \overline{x})} \Sigma_i \Sigma_j (x_i - \overline{x})\left(x_j - \overline{x}\right),$$

where I refers to Moran's I; N is the total number of spatial units indexed with i and j; x is the random variable, in this case, a phenotype score for tissue microenvironment in the spatial units; $\bar{x}$ is the mean of x; w_{ij} is a spatial weight matrix; and W is the sum of all w_{ij}. We computed spatial autocorrelation values using this formula with the above-described spatial weights for the edges in the network (Fig. 4; *see* **Note 5**).

3.6 Visualization of Analysis Results

These analyses produced a set of results (*see* **Note 6**):

(a) Cell-type estimation.

(b) Spatial network graphs.

(c) Network trimming.

(d) Spatial autocorrelation analysis.

4 Notes

1. After strict quality control and filtration, a total of 6823 spots in ST data were retained for downstream analysis (S1-1923, S2-1230, S3-2097, S4-1573). Tumors are spatially heterogeneous tissues that comprise numerous cell types with intricate structures. Expression matrices were combined from all ST samples; then, gene and spot filtering, dimensionality reduction, and clustering of all spots were performed. We recommend using well-characterized cell-type markers to identify different cell, including endothelial cells, epithelial cells,

fibroblasts, immune cells, and smooth muscle cells. As expected from tumor tissues, epithelial cancer cells emerged as the dominant cell type, given that bladder carcinomas are considered as the malignancies of epithelial tissue [19, 20].

2. Tumor microenvironment is inherently heterogeneous and thus spatial heterogeneity is expected within and between tumors. In our dataset, the proportions of cell types varied in each region, revealing heterogeneity and complexity by showing differential gene expression profiles of regions on the same slide and by revealing differential patterns across patients. Focusing on the tumor tissues, we found clusters of epithelial cells both in the center of the tumor and in the invasive front in all tumor samples. We observed heterogeneous distribution of other cell populations. T cells and tumor cells were generally in proximity, although we observed differences in immune cell infiltration within and across tumors. On the other hand, peritumoral zone was rich in fibroblasts, which protects tumor cells from enhanced T cell accumulation [21–23]. These results reinforce that immune and stromal cells within the TME play a key role in cancer progression by interacting with tumor cells by secreting different chemokines, cytokines, and other signaling molecules.
3. To examine the effects of the dense-to-sparse representation of the network graph on key spatial inference from spatial transcriptomic data, it is recommended that results from different trimming are compared. We started with the densely connected graph to sparse and calculated a weight (w) for each edge that depends on the cell-type composition similarity and geographic distance between pairs of spatially annotated spots. As we pruned the weak connections with increasing stringency (w'_{cutoff}: 0–10) to attain dense-to-sparse representation of the network graph, the overall connectedness of the graph declined, as expected, and the network gradually disintegrated into disjoint subgraphs and isolated nodes. At an extreme, at the highest level of w_{cutoff}, all nodes were isolated. Trimming at an intermediate level, i.e., w'_{cutoff}, within 4–8 created interpretable sparse networks that revealed biologically interesting local microenvironmental architectures and spatial patterns in cellular processes.
4. We observed that there were sample-to-sample variations in the tissue architecture, network graph topology, and specific inferences about cell-type compositions and cell-type specific local autocorrelation—indicating that single network trimming cutoff may be unsuitable for all situations. Instead, the gradual trimming from dense-to-sparse network using the strategy described above can be a generic approach to identify context-specific trimming ranges and resulting biologically relevant patterns.

5. Moran's *I* is a popular measure of spatial autocorrelation across adjacent nodes in a graph to examine spatial patterns of cell-type abundance for respective cell types and assessed the effects of trimming on this index. Negative Moran's *I* values indicate an inverse relationship between abundance of cells of specific cell types with respect to that in the neighboring spots, and Moran's *I* values above 0 suggest a synergistic relationship among adjacent spots in terms of abundance of specific cell types. We observed that Moran's *I* for trims 4–8 was statistically significant for all cell types; however, nonsignificant *P* values were associated with extreme trims. This is due to organization of tissue spatial architecture into discrete subunits based on cell types and by pruning the weakest connections—the collection of these cellular subunits becomes concentrated in the locality [24].
6. Overall, our observations suggest that refinement of the network graph based on both cell type and proximity captures spatial relationships among cell types and biological processes and connectivity in the local tissue microenvironment. We found that intermediate levels of network connectivity are more useful in identifying local structures in cell-type composition and interdependent biological processes. Autocorrelation of cell-type abundance scores, measured using Moran's *I*, revealed local tissue-level microstructures, such that regions of clonal tumor growth or fibrosis at this range. Importantly, it showcases the effects of different levels of network trimming on the reliability of biological inferences. Our method is of course not without limitations. The major limitation is that tumor images represent a single timepoint, meaning that our method is not temporally resolved [25, 26]. Another limitation is that the current spatial transcriptomic technologies lack single-cell resolution, but emerging techniques might provide highly defined annotations to locate fine-grained histopathological regions and further improve trimming tools and metrices [25, 27]. Constructing spatial-temporal connectivity graph, with 3D perspectives, and designing trimming operations may help resolve the challenging issues of incoherent predictions and enabling demonstration of transitions in connectivity. By doing so, one could establish statistical metrices for the biological network analysis focusing on intercellular communication patterns.

Acknowledgments

The authors acknowledge financial support from New Jersey Commission for Cancer Research (to A.B.) R01GM129066, P01CA250957, and R35GM149224 (to S.D.). The authors

thank the other members of the laboratory of S.D. and also Rutgers Cancer Institute for helpful discussions.

Conflict of Interest The authors declare no potential conflicts of interest with respect to research, authorship, and/or publication of this chapter.

References

1. Bryant DM, Mostov KE (2008) From cells to organs: building polarized tissue. Nat Rev Mol Cell Biol 9(11):887–901
2. Dos Santos AXdS, Liberali P (2019) From single cells to tissue self-organization. FEBS J 286(8):1495
3. Anderson NM, Simon MC (2020) The tumor microenvironment. Curr Biol 30(16):R921–R925
4. Baghban R et al (2020) Tumor microenvironment complexity and therapeutic implications at a glance. Cell Commun Signal 18:1–19
5. Palla G et al (2022) Spatial components of molecular tissue biology. Nat Biotechnol 40(3):308–318
6. Ståhl PL et al (2016) Visualization and analysis of gene expression in tissue sections by spatial transcriptomics. Science 353(6294):78–82
7. Rao A et al (2021) Exploring tissue architecture using spatial transcriptomics. Nature 596(7871):211–220
8. Tian L, Chen F, Macosko EZ (2023) The expanding vistas of spatial transcriptomics. Nat Biotechnol 41(6):773–782
9. Rao N, Clark S, Habern O (2020) Bridging genomics and tissue pathology: 10x genomics explores new frontiers with the visium spatial gene expression solution. Genet Eng Biotechnol News 40(2):50–51
10. Moses L, Pachter L (2022) Museum of spatial transcriptomics. Nat Methods 19(5):534–546
11. Velten B, Stegle O (2023) Principles and challenges of modeling temporal and spatial omics data. Nat Methods 20(10):1462–1474
12. Liu T et al (2023) A comprehensive overview of graph neural network-based approaches to clustering for spatial transcriptomics T. Liu et al. Overview of Spatial Transcriptomics' Spatial Clutering. Comput Struct Biotechnol J 23: 106–128
13. Biswas A et al (2022) Inference on spatial heterogeneity in tumor microenvironment using spatial transcriptomics data. Comput Syst Oncol 2(3):e21043
14. Biswas A et al (2023) Transcriptional state dynamics lead to heterogeneity and adaptive tumor evolution in urothelial bladder carcinoma. Commun Biol 6(1):1292
15. Hornik K (2012) The comprehensive R archive network. Wiley Interdiscip Rev Comput Stat 4(4):394–398
16. Team, R.D.C., R: A language and environment for statistical computing. (No Title), 2010
17. Satija R et al (2015) Spatial reconstruction of single-cell gene expression data. Nat Biotechnol 33(5):495–502
18. Hao Y et al (2021) Integrated analysis of multimodal single-cell data. Cell 184(13): 3573–3587.e29
19. Sanli O et al (2017) Bladder cancer. Nat Rev Dis Prim 3(1):1–19
20. Dyrskjøt L et al (2023) Bladder cancer. Nat Rev Dis Primers 9(1):58
21. Gajewski TF (2015) The next hurdle in cancer immunotherapy: overcoming the non–T-cell–inflamed tumor microenvironment. In: Seminars in oncology. Elsevier
22. Fearon DT (2014) The carcinoma-associated fibroblast expressing fibroblast activation protein and escape from immune surveillance. Cancer Immunol Res 2(3):187–193
23. Joyce JA, Fearon DT (2015) T cell exclusion, immune privilege, and the tumor microenvironment. Science 348(6230):74–80
24. Seferbekova Z et al (2023) Spatial biology of cancer evolution. Nat Rev Genet 24(5): 295–313
25. De S (2021) Signatures beyond oncogenic mutations in cell-free DNA sequencing for non-invasive, early detection of cancer. Front Genet 12:759832
26. Biswas A, De S (2021) Drivers of dynamic intratumor heterogeneity and phenotypic plasticity. Am J Phys Cell Phys 320(5):C750–C760
27. Moffitt JR, Lundberg E, Heyn H (2022) The emerging landscape of spatial profiling technologies. Nat Rev Genet 23(12):741–759

Chapter 10

Inferring Metabolic Flux from Gene Expression Data Using METAFlux

Yuchen Pan, Yuefan Huang, Vakul Mohanty, and Ken Chen

Abstract

Metabolic dysregulation is a hallmark of malignant cells, which contributes significantly to tumor proliferation, persistence, and therapeutic resistance. Further, metabolic interplay between malignant cells and cells in the tumor microenvironment (TME) has a significant impact on tumor phenotype. Examining the reconfiguration of metabolic pathways within tumors and TME is therefore critical to understand cancer biology and improve patient care. Current limitations of metabolomic techniques, however, restrict broad and deep characterization of tumor metabolome. To address this gap, we developed METAFlux (METAbolic Flux balance analysis), a computational technique that uses flux balance analysis (FBA) to infer activity or flux of metabolic reactions from bulk and single-cell RNA sequencing data (scRNA-seq). Here, we describe the workflow along with a detailed step-by-step explanation for calculating metabolic fluxes using METAFlux from bulk RNA-seq and scRNA-seq data and the extension to characterize metabolic heterogeneity and metabolic interaction among cell types.

Key words Metabolism, Flux balance analysis, Bulk RNA-seq, Single-cell RNA-seq, Tumor microenvironment

1 Introduction

Research has shown that metabolic characteristics of tumors and their TME are linked to tumor phenotype and patient outcome and can potentially unveil therapeutic vulnerabilities [1]. Understanding metabolic patterns in cancers can therefore facilitate greater understanding of the underlying cancer biology and development of new therapeutics. However, contemporary metabolomic technologies only assess a subset of metabolites, thus providing an incomplete view of tumor metabolism. Large multi-omics datasets used to study molecular biology of cancers also often lack detailed metabolic profiling [2]. Flux balance analysis (FBA), a quantitative technique, that models the flow of metabolites through metabolic networks [3]. FBA's application to gene expression data can exploit the relative abundance of transcriptomic data to comprehensively

Alexander Krasnitz and Pascal Belleau (eds.), *Cancer Bioinformatics*, Methods in Molecular Biology, vol. 2932,
https://doi.org/10.1007/978-1-0716-4566-6_10,

study tumor metabolism [4–7]. While computational methods have been developed to infer metabolic flux from bulk and single-cell(sc)RNA-seq, methods that model the metabolic interplay among various cell populations in the TME are underdeveloped.

To address these challenges, we developed METAFlux, a computational framework to deduce metabolic fluxes from bulk and single-cell transcriptomic data using FBA. METAFlux utilizes Human1, a genome-scale metabolic model (GEM) that encodes the relationships between genes, metabolites, and reactions in a human cell to perform FBA. METAFlux is capable of simultaneously inferring a nondegenerate solution for flux of all reactions in the metabolic network from cancer gene expression data while considering media or TME nutrient conditions as binary parameters, corresponding to their availability. The metabolic fluxes generated by METAFlux can be a valuable resource for detailed metabolic characterization of tumors from gene expression and identification of potential metabolic targets for detailed follow-up studies in the context of precision medicine. In conventional FBA analysis, fluxes represent the velocities or rates at which metabolic reactions occur. However, in METAFlux, the predicted fluxes are calculated and normalized based on gene expression data, which means the results are relative flux scores. A positive flux indicates the reaction progressing in the forward direction, and a negative flux indicates the reaction is reversed. In the context of nutrient uptake, positive fluxes indicate secretion of the nutrient into the interstitial space and negative uptake of nutrients. Here, we present the workflow with detailed step-by-step explanation to obtain metabolic fluxes from bulk RNA-seq and scRNA-seq data using METAFlux.

2 Materials

2.1 Software and Packages Version

METAFlux_1.1.0

R version 4.3.2

Seurat_4.3.0

Matrix_1.6.3

2.2 Human1 (Human-GEM File)

Human1 [8] is a publicly accessible consensus human genome-scale metabolic model (GEM) consisting of 13,082 metabolic reactions and 8378 metabolites. These reactions are distributed across nine distinct compartments, including the extracellular space, peroxisome, mitochondria, cytosol, lysosome, endoplasmic reticulum, Golgi apparatus, nucleus, and inner mitochondria.

2.3 Cell Medium and Human Blood Nutrient Profiles

METAFlux requires a user-defined input that defines a set of metabolites (nutrient profile) that are available for uptake by cells. The input is formatted as a table with nutrient names in the "metabolite" column and their corresponding exchange reaction IDs in the "reaction_name" column. METAFlux offers 2 default mediums to define the nutrient profiles: (1) "cell_medium" consisting of 44 metabolites that can be used to analyze cells grown in culture and (2) "human_blood" which is an approximation of nutrients available in human blood for use on expression profiles derived from human tissues. These nutrient profiles can be used by the user in the absence of experiment specific nutrient profiles.

2.4 Nutrient Lookup Files

METAFlux offers a table "nutrient_lookup_files" containing 1648 exchange reactions. These exchange reactions are artificial reactions that serve as a mathematical representation exchange of metabolites between cells and the extracellular space.

3 Methods

3.1 Installation

METAFlux R package can be easily installed from GitHub using devtools:

```
devtools::install_github('KChen-lab/METAFlux')
```

Installation of other dependencies

- Install the osqp package for optimization using.

```
install.packages('osqp')
```

- Install the dplyr package using.

```
install.packages('dplyr')
```

- For single-cell data analysis, we provide a pipeline to work with Seurat. Please install Seurat package using.

```
install.packages('Seurat')
```

3.2 Bulk RNA-Seq Pipeline

3.2.1 Quick Workflow for Bulk RNA-Seq Sample

This workflow can be applied to human cell line and tissue derived bulk RNAseq. Gene expression is input as a matrix with genes as rows and samples as columns. Users will need to choose the appropriate nutrient profile (either cell medium for cells in culture or human blood medium for tissues samples) as the input for METAFlux. Users can also provide custom nutrient profiles when appropriate. Note: METAFlux estimates flux in each sample based on the same nutrient profile in any given run.

```
library(METAFlux)
https://htmlpreview.github.io/?https://github.com/KChen-lab/
METAFlux/blob/main/Tutorials/pipeline.html - cb2-3data
("bulk_test_example")
data("cell_medium")
data("human_blood")
scores <- calculate_reaction_score(bulk_test_example)
flux <- compute_flux(mras = scores,medium = human_blood)
cbrt <- function(x) {
     sign(x) * abs(x)^(1/3)
}
flux <- cbrt(flux)
```

3.2.2 Step-by-Step Bulk RNA-Seq Pipeline

```
library(METAFlux)
```

Load the Library

Load Data

(1) Load gene expression data.

METAFlux requires gene expression data as input.

- The gene expression matrix should be gene by sample matrix where row names are human gene names (gene symbols), and column names should be sample names. Please note that METAFlux does not support other gene IDs.
- The input gene expression matrix should be normalized (e.g., log-transformed TPM, etc.) before using METAFlux. METAflux will not perform any normalization on expression data.
- Gene expression data cannot have negative values.

```
data("bulk_test_example")
head(bulk_test_example)
##           Sample1    Sample2  Sample3  Sample4    Sample5
## TSPAN6   4.433587 4.06179073 4.777144 5.501764 5.32881296
## TNMD     0.000000 0.04264398 0.000000 0.000000 0.08406697
## DPM1     4.467942 5.61354161 5.125975 4.926973 4.74574474
## SCYL3    2.286859 2.75061766 2.356148 2.548451 1.87972673
## C1orf112 2.575287 2.82777973 2.087475 1.682574 1.76127531
## FGR      4.005357 2.56314415 1.673564 2.272007 2.08067237
```

(2) Load Human 1 GEM.

We use the Human 1 GEM as the underlying metabolic model. For each reaction, there is one unique Reaction ID and SUBSYSTEM.

```
data("human_gem")
head(human_gem[,c("ID","EQUATION","EC-NUMBER","GENE ASSOCIA-
TION","SUBSYSTEM")])
```

In Table 1, we can get the following information from each column:

ID represents the reaction ID in Metabolic Atlas.

Equation shows the detailed chemical equation for this reaction; here, "=>" means this reaction is irreversible and "<=>" means reversible.

EC number represents the Enzyme Commission number; every enzyme code consists of four numbers separated by periods. These numbers represent a progressively finer classification of the enzyme. For more detailed expansions, please refer to BiteSizeBio link [9].

Gene association represents the Ensembl ID of the genes associated with this reaction; gene reaction associations from HMR2, Recon3D, and iHsa were combined and integrated with enzyme complex information from Recon3D, iHsa, and the comprehensive resource of mammalian protein complexes database (CORUM [10]) to obtain gene reaction rules for Human1.

Subsystem corresponds to a set of reactions that share a similar metabolic function; it can help organize and categorize metabolic reactions based on their functional roles or participation in specific cellular processes.

For each reaction, we can also get other important information from this file such as whether the reaction is reversible, the specific compartment in which it occurs, and a list of the metabolites and genes involved.

(3) Load the METAFlux medium.

Two general medium files, "cell medium" and "human blood medium," are available in METAFlux for users for general purpose use for analysis of human cell lines in culture and tissue samples, respectively. METAFlux also allows for input of user-defined medium when the information is available (see below).

```
data("cell_medium")
data("human_blood")
```

Table 1
Preview of Human-GEM: the generic genome-scale metabolic model of *Homo sapiens*

ID	Equation	EC number	Gene association	Subsystem
HMR_3905	ethanol[c] + NAD+[c] = > acetaldehyde [c] + H+[c] + NADH[c]	1.1.1.1;1.1.1.71	ENSG00000147576 or ENSG00000172955 or ENSG00000180011 or ENSG00000187758 or ENSG00000196344 or ENSG00000196616 or ENSG00000197894 or ENSG00000198099 or ENSG00000248144	Glycolysis/ gluconeogenesis
HMR_3907	ethanol[c] + NADP +[c] = > acetaldehyde[c] + H +[c] + NADPH[c]	1.1.1.2	ENSG00000117448	Glycolysis/ gluconeogenesis
HMR_4097	acetate[c] + ATP[c] + CoA [c] = > acetyl-CoA[c] + AMP[c] + PPi [c]	6.2.1.1	ENSG00000131069	Glycolysis/ gluconeogenesis
HMR_4099	acetate[m] + ATP[m] + CoA [m] = > acetyl-CoA[m] + AMP [m] + PPi[m]	6.2.1.1	ENSG00000111058 or ENSG00000154930	Glycolysis/ gluconeogenesis
HMR_4108	acetyl adenylate[c] + CoA[c] = > acetyl-CoA[c] + AMP[c] + H+[c]	6.2.1.1	ENSG00000131069	Glycolysis/ gluconeogenesis
HMR_4133	acetate[c] + ATP[c] + H+[c] = > acetyl adenylate[c] + PPi[c]	6.2.1.1	ENSG00000131069	Glycolysis/ gluconeogenesis

Look at the first six rows of cell_medium.

(4) Load the METAFlux medium if users have prior knowledge about their medium composition.

Metabolite	reaction_name
Arginine	HMR_9066
Histidine	HMR_9038
Lysine	HMR_9041
Methionine	HMR_9041
Phenylalanine	HMR_9043
Tryptophan	HMR_9045

Users can provide custom medium files that need to be formatted as a data.frame with two columns, "metabolite" and "reaction_name," for the name of the metabolite and its exchange reaction, respectively. The exchange IDs can be looked up in the nutrient_lookup_files. For example, metabolite "naphthalene" has an exchange reaction ID "HMR_7110."

```
data("nutrient_lookup_files")
view("nutrient_lookup_files")
```

Note: After viewing the nutrient lookup file, users can use the search box on the top right corner to locate the corresponding nutrient reaction and its equation

Calculate MRAS (Metabolic Reaction Activity Score)

METAFlux utilizes Gene-Protein-Reaction (GPR) rules [11] to decipher the Boolean logic relationships among genes within a specific reaction and use it to compute a metabolic reaction activity score (MRAS) for each reaction. MRAS represents the activity of a reaction as a function of gene expression of enzymes catalyzing it. Using the approach described below, the expression of 3625 metabolic genes, a MRAS score is calculated for each reaction in Human 1.

In GPR, AND operator is employed to link genes that encode for different subunits of the same enzyme, and the OR operator is used to connect genes encoding for isoenzymes. For an enzyme complex where all the subunits need to be expressed to catalyze a reaction, the metabolic activity is determined by the lowest expression value among all the genes associated with this enzyme complex. For more details about the calculation of each operator and the steps of deriving MRAS, please refer to our Nature Communications (NatComm) paper [12].

Given a gene expression matrix *calculate_reaction_score* function in METAFlux can be used to calculate MRAS scores across all samples:

```
scores <- calculate_reaction_score(bulk_test_example)
```

Calculate Flux

MRAS calculated above can be used to connect transcriptome and fluxes. Normalized MRAS is used as the flux upper bound to their corresponding metabolic reactions, and the lower bound is set to zero if the reaction is nonreversible or to (-normalized MRAS) if reversible. For more details about how to construct the constraints of fluxes, please refer to our NatComm paper [12].

To guide the optimization search in a biologically relevant sup-space, the metabolite availability needs to be defined. We use the *cell line culture* medium containing 44 metabolites as the growth medium, which includes major components from Hams F-12 medium and other essential nutrients and ions from serum supplements. The uptake or secretion rates of these 44 metabolites are not limited. However, for the remaining metabolites in the model, cells can only secrete them into the medium rather than uptake from the medium. For patients' tissue sample, we derived a list of 64 metabolites in *human blood* based on a human plasma-like medium (HPLM) developed by Cantor et al. [13] For more general use purposes, users can also define their own metabolite list.

Predicted fluxes are inferred from gene expression, so the results are relative flux scores. A positive flux represents secretion, while a negative flux indicates uptake of the metabolite.

In R code, we calculate the metabolic fluxes for the 13,082 reactions.

```
https://htmlpreview.github.io/?https://github.com/KChen-lab/METAFlux/blob/main/Tutorials/pipeline.html - cb23-2flux <-compute_flux(mras = scores,medium = human_blood)
```

Inspecting and Interpreting the Flux Data

- The sign of flux represents the direction of a reaction. In the context of nutrient uptake/release reactions (1648 exchange reactions in the nutrient lookup file), a positive value signifies the release of metabolites into extracellular space, while a negative value indicates the uptake of metabolites. In other reactions, a positive flux denotes a net forward direction, whereas a negative flux implies a net backward direction. The absolute values represent the magnitude of the flux.
- As we aim to minimize the sum of all fluxes in the model, the resulting flux data output tends to be parsimonious, with many reactions approaching zero flux. For instance, reactions that

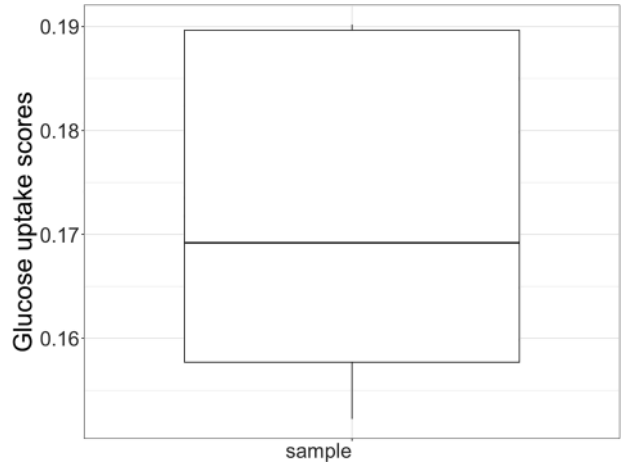


Fig. 1 The glucose uptake result of sample data in the step-by-step bulk-RNA seq pipeline

should predominantly proceed in the forward direction might have a predicted flux with a small negative value, effectively approaching zero flux.

- The "nutrient_lookup_files" and "human_gem" tables can be used to focus on specific reactions and nutrient exchanges of interest. For instance, if we seek information on glucose uptake, a search for "glucose" would yield the glucose uptake reaction (HMR_9034). These values can be considered as the rates of glucose metabolite uptake. Subsequently, we can extract the relevant data (Fig. 1):

```
 data("nutrient_lookup_files")
glucose <- data.frame(glucose = flux[grep("HMR_9034",human_-
gem$ID),])
library(ggplot2)
ggplot(glucose, aes(y = -glucose, x = "sample")) + geom_box-
plot() + ggtitle("Glucose uptake level") +
 xlab("") + ylab("Glucose uptake scores") + theme_bw()
# result shown in Figure1
```

If one wants to explore other reactions such as glycolysis, oxphos, etc., Reaction_ID is required by using human_gem file to search it. As an illustration, suppose our focus is on the reaction HMR_4363 within the glycolysis pathway:

```
HMR_4363 <- data.frame(hmr4363 = flux[grep("HMR_4363",human_-
gem$ID),])
```

3.3 Single-Cell RNA-Seq Pipeline

3.3.1 Quick Workflow for Single-Cell RNA-Seq Sample

When applied to scRNAseq, METAFLux models the entire TME as a community to consider metabolic interactions between groups. In this approach, we model fluxes at the level of cell groups such as cluster or cell-type level rather than the individual cells, to mitigate the effect of sparsity in scRNAseq data and characterizing metabolic heterogeneity and interaction among various cell types or clusters. Clusters can be obtained function *FindNeighbors* and *FindClusters* in Seurat, which is based on nearest neighbor graph, and cell types can be assigned based on the known knowledge or clustering results. The application is showcased using Seurat object. Since bootstrap samples will be generated later by sampling with replacement, this step is necessary to estimate the properties of each group when sampling from an approximating distribution.

To estimate metabolic fluxes, the following inputs are required:

1. METAFlux's single-cell workflow accepts gene expression as a Seurat object. Please note that the expression should be normalized prior to using the object to run METAFLux. If users have an expression matrix, then *CreateSeuratObject* function in Seurat package can be used to create a Seurat object compatible with this workflow.
2. Group assignment for each cell. This can be cell type, cluster, or other user-defined grouping. This should be specified in a column in the metadata file of the Seurat object.
3. Cluster/cell-type fractions. As single-cell disassociation protocols can have uneven sampling of cell types in a tissue, estimating cell-type proportions directly from single-cell data can introduce biases. When available, we encourage users to use cell proportions inferred from experiments or derived from in silico deconvolution of corresponding bulk gene expression data.

Note: As METAFlux models cell types/groups as a collective with the same TME, it is more intuitive to run METAFlux independently on each sample rather than on single-cell data after integration. However, based on specific biological inquiries and data quality, one may choose to explore the average effect across subjects, where the proportions of each cluster or cell type are estimated using all available data, and flux computation will be performed assuming that all the cell groups exist within a shared TME.

Here we present the quick workflow to run METAFlux on a Seurat object containing cells from a single sample.

```
library(METAFlux)
data("sc_test_example")
data("human_blood")
mean_exp = calculate_avg_exp(
```

```
 myseurat = sc_test_example,
 myident = 'Cell_type',
 n_bootstrap = 3,
 seed = 1
)
scores <- calculate_reaction_score(data = mean_exp)
round(table(sc_test_example$Cell_type)/nrow(sc_test_example@-
meta.data),1)
flux <- compute_sc_flux(
 num_cell = 4,
 fraction = c(0.1, 0.3, 0.3, 0.3),
 fluxscore = scores,
 medium = human_blood
)
cbrt <- function(x) {
 sign(x) * abs(x)^(1/3)
flux <- cbrt(flux)
```

In cases where the Seurat object contains multiple samples, we recommend running METAFlux on each sample independently. The workflow below presents an outline of how this can be achieved.

```
obj.list <- SplitObject(seurat, split.by = "patient_id")
for (i in c(1:length(obj.list))) {
  sc <- obj.list[[i]]
  mean_exp = calculate_avg_exp(
    myseurat = sc_test_example,
    myident = 'Cell_type',
    n_bootstrap = 50,
    seed = 1
  )
  scores <- calculate_reaction_score(data = mean_exp)
  g <- table(sc$Cell_type) / nrow(sc@meta.data)
  print(g)
  flux = compute_sc_flux(
    num_cell = 4,
    fraction = c(g[1], g[2], g[3], g[4]),
    fluxscore = scores,
    medium = human_blood
  )
}
```

```
 library(METAFlux)
data("human_blood")
data("cell_medium")
```

3.3.2 Step-by-Step Single-Cell RNA-Seq Pipeline

```
data("human_gem")
```

Load Library, METAFlux Medium, and GEM Information

Load the Single-Cell Data

```
data("sc_test_example")
```

Please note that this Seurat toy example is based on a single patient. For multiple samples, refer to the example code in Quick workflow for single-cell RNA-seq sample.

Create an Average Expression Profile for Stratified Bootstrapped Samples for This Patient

We only provide the built-in function for computing mean expression on bootstrap samples; here, METAFlux utilizes *AverageExpression* function in Seurat to get the average expression in each identity class; it returns a matrix with genes as rows and identity classes as columns. There are various methods to aggregate samples, such as median or geometric mean. Users have the flexibility to calculate their own "average expression profile" according to their preferences, but it should adhere to the same data format as illustrated below.

```
mean_exp <-
  calculate_avg_exp(
    myseurat = sc_test_example,
    myident = 'Cell_type',
    n_bootstrap = 3,
    seed = 1
  )
```

Calculate MRAS (Metabolic Reaction Activity Score)

MRAS can be computed from individual samples using GPR. The scores are normalized by dividing each element by the maximum value in that vector. The whole vector includes the reaction score of all relationships that a gene is involved in. This step is the same as the process in the bulk RNA-seq pipeline.

Since bootstrap samples are used, MRAS matrix has a repeating motif.

```
scores <- calculate_reaction_score(data=mean_exp)
```

Compute Flux

The flux computation and GEM setup are quite similar to the bulk pipeline. However, there are some differences for single cell: GEMs are merged across cell types, cell-type proportions are taken into consideration, and the biomass of the full community is maximized.

specifically, we generate bootstraped samples, and each generated sample has the same size and the same group proportion as the original data, then using these samples to calculate MRAS for each mean gene expression vector, and group fraction parameter is also defined, which indicates the proportions of groups of interest with respect to the whole sample. To merge multiple metabolic networks, we create a "TME metabolite reservoir" for different cell groups to interact. Our model is designed to optimize the biomass of the entire community while minimizing the sum of squares of overall fluxes.

To calculate the metabolic fluxes, OSQP solver is also used as in bulk pipeline. For the mathematical details of how the model is constructed and how the fluxes are calculated, please refer to our NatComm paper [12]. Users should keep the order of *fraction* parameters consistent with the order of *scores* results (MRAS scores matrix). For example, Cell type 1 (B lymphocytes) is the first column of scores, and cell type 2 (epithelial cells) is the second column of scores. And fractions need to sum up to 1.

```
round(table(sc_test_example$Cell_type)/nrow(sc_test_example@-
meta.data),1)
flux <- compute_sc_flux(
 num_cell = 4,
 fraction = c(0.1, 0.3, 0.3, 0.3),
 fluxscore = scores,
 medium = human_blood
)
```

Inspecting and Interpreting the Flux Data

The total dimension of the predicted flux data is calculated as (num_cell * 13082 + 1648) * number_of_bootstrap (13,082 reactions in *Human1* and 1648 exchange reactions in nutrient lookup file). Rows labeled "external_medium" represent reactions involving the entire TME exchanging metabolites with the external environment. These reactions pertain to the overall tumor community rather than a specific cell type. Rows labeled "internal_medium" correspond to reactions where a specific cell type or cluster exchanges metabolites with the TME.

Signs of fluxes are biologically meaningful as previously discussed in the bulk pipeline.

The fluxes specific to cell types represent average fluxes, calculated as the mean or unit flux for each cell type. To clarify if the flux of reaction A for cell type 1 is 0.2, it denotes that the average cell within cell type 1 exhibits a flux of 0.2 for reaction A. In contrast, external_medium fluxes are not average fluxes; instead, they are weighted total fluxes. The relationship between external_medium and internal_medium is defined as follows:

$$Proportion^{celltype1} \cdot Internal\_Medium^{Celltype1}_{exchange\_reaction_i} + Proportion^{celltype2} \cdot Internal\_Medium^{Celltype2}_{exchange\_reaction_i} + \cdots\cdots = External\_Medium_{exchange\_reaction_i}$$

We can extract the reactions of interest. Once again, for those solely focused on metabolite uptake or release, a search in the "nutrient lookup file" can provide the Reaction ID for the metabolite exchange (refer to "Bulk Step-by-Step Breakdown" section in Subheading 2.3 for detailed guidance). For instance, to inquire about glucose uptake, a search for "glucose" would yield the glucose uptake reaction, identified as HMR_9034. Subsequently, the relevant data can be extracted by the following:

```
data("nutrient_lookup_files")
glucose <- data.frame(glucose=flux[grep("HMR_9034",rownames
(flux)),])
cbrt <- function(x) {
     sign(x) * abs(x)^ (1/3)
}
```

Considering all these key points collectively, we observe that Glucose.V1, Glucose.V2, and Glucose.V3 represent three bootstrap samples. One can examine the distribution of all bootstraps to compare the nutrient uptake profiles of B cells, epithelial cells, myeloid cells, and T cells (Fig. 2).

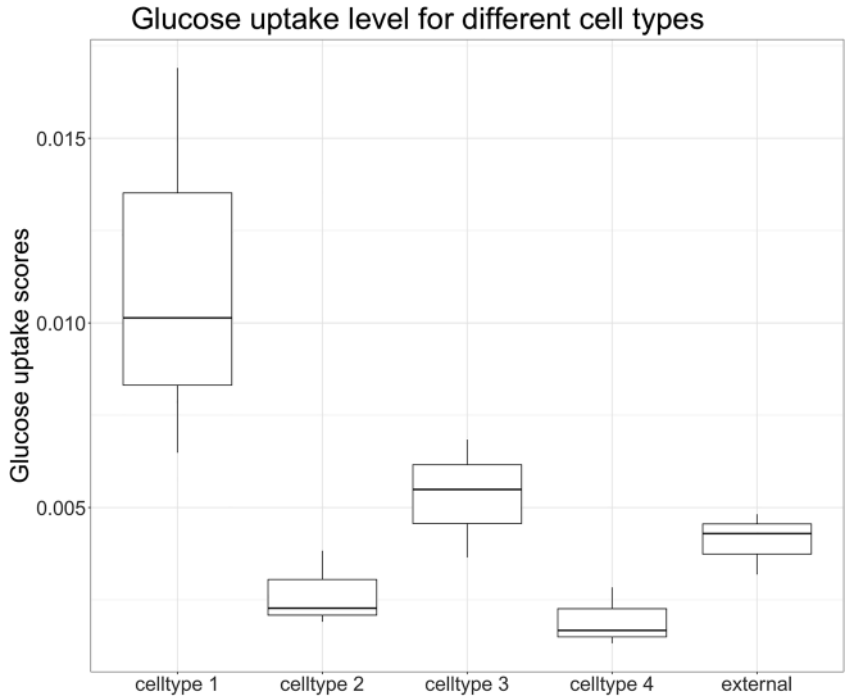


Fig. 2 The glucose uptake result of sample data in the step-by-step single-cell RNA seq pipeline

```
library(ggplot2)
glucose$celltype = rownames(glucose)
long_glucose = reshape2::melt(glucose, id.vars = 'celltype')
ggplot(long_glucose, aes(y = -value, x = celltype)) + geom_-
boxplot() + ggtitle("Glucose uptake level for different cell
types") +
  xlab("") + ylab("Glucose uptake scores") + theme(axis.text.x
= element_text(
    angle = 90,
    vjust = 0.5,
    hjust = 1
  ))
# result shown in Figure2
```

- "celltype 1 internal_medium HMR_9034" is the glucose uptake flux for cell type 1 (referring to B cells in toy example).
- "celltype 2 internal_medium HMR_9034" is the glucose uptake flux for cell type 2(referring to epithelial cells in toy example).
- "celltype 3 internal_medium HMR_9034" is the glucose uptake flux for cell type 2(referring to myeloid cells in toy example).
- "celltype 4 internal_medium HMR_9034" is the glucose uptake flux for cell type 2(referring to T cells in toy example).
- "external_medium HMR_9034" is the glucose uptake flux for the whole tumor community(referring to combined TME community in toy example).

References

1. Faubert B, Solmonson A, DeBerardinis RJ (2020) Metabolic reprogramming and cancer progression. Science 368(6487):eaaw5473. https://doi.org/10.1126/science.aaw5473
2. Zhou B, Xiao JF, Tuli L, Ressom HW (2012) LC-MS-based metabolomics. Mol Biosyst. 8(2):470–481. https://doi.org/10.1039/c1mb05350g. Epub 2011 Nov 1. PMID: 22041788; PMCID: PMC3699692
3. Orth J, Thiele I, Palsson B (2010) What is flux balance analysis? Nat Biotechnol 28:245–248. https://doi.org/10.1038/nbt.1614
4. Lee D, Smallbone K, Dunn WB, Murabito E, Winder CL, Kell DB, Mendes P, Swainston N (2012) Improving metabolic flux predictions using absolute gene expression data. BMC Syst Biol 6:73. https://doi.org/10.1186/1752-0509-6-73. PMID: 22713172; PMCID: PMC3477026
5. Jensen PA, Papin JA (2011) Functional integration of a metabolic network model and expression data without arbitrary thresholding. Bioinformatics. 27(4):541–547. https://doi.org/10.1093/bioinformatics/btq702. Epub 2010 Dec 20. PMID: 21172910; PMCID: PMC6276961
6. Colijn C, Brandes A, Zucker J, Lun DS, Weiner B, Farhat MR, Cheng TY, Moody DB, Murray M, Galagan JE (2009) Interpreting expression data with metabolic flux models: predicting Mycobacterium tuberculosis mycolic acid production. PLoS Comput Biol. 5(8): e1000489. https://doi.org/10.1371/journal.pcbi.1000489. Epub 2009 Aug 28. PMID: 19714220; PMCID: PMC2726785
7. Chandrasekaran S, Price ND (2010) Probabilistic integrative modeling of genome-scale metabolic and regulatory networks in Escherichia coli and Mycobacterium tuberculosis. Proc Natl Acad Sci U S A 107(41):

17845–17850. https://doi.org/10.1073/pnas.1005139107. Epub 2010 Sep 27. PMID: 20876091; PMCID: PMC2955152

8. Robinson JL, Kocabaş P, Wang H, Cholley PE, Cook D, Nilsson A, Anton M, Ferreira R, Domenzain I, Billa V, Limeta A, Hedin A, Gustafsson J, Kerkhoven EJ, Svensson LT, Palsson BO, Mardinoglu A, Hansson L, Uhlén M, Nielsen J (2020) An atlas of human metabolism. Sci Signal 13(624):eaaz1482. https://doi.org/10.1126/scisignal.aaz1482. PMID: 32209698; PMCID: PMC7331181
9. Understand EC numbers in 5 minutes part I: how EC numbers work. Protein Expression and Analysis July 8, 2013. https://bitesizebio.com/10683/understand-ec-numbers-in-5-minutes-part-i-how-ec-numbers-work/
10. Tsitsiridis G, Steinkamp R, Giurgiu M, Brauner B, Fobo G, Frishman G, Montrone C, Ruepp A (2023) CORUM: the comprehensive resource of mammalian protein complexes-2022. Nucleic Acids Res 51(D1): D539–D545. https://doi.org/10.1093/nar/gkac1015. PMID: 36382402; PMCID: PMC9825459
11. Gu C, Kim GB, Kim WJ, Kim HU, Lee SY (2019) Current status and applications of genome-scale metabolic models. Genome Biol 20(1):121. https://doi.org/10.1186/s13059-019-1730-3. PMID: 31196170; PMCID: PMC6567666
12. Huang Y, Mohanty V, Dede M, Tsai K, Daher M, Li L, Rezvani K, Chen K (2023) Characterizing cancer metabolism from bulk and single-cell RNA-seq data using METAFlux. Nat Commun 14(1):4883. https://doi.org/10.1038/s41467-023-40457-w. PMID: 37573313; PMCID: PMC10423258
13. Cantor JR, Abu-Remaileh M, Kanarek N, Freinkman E, Gao X, Louissaint A Jr, Lewis CA, Sabatini DM (2017) Physiologic Medium Rewires Cellular Metabolism and Reveals Uric Acid as an Endogenous Inhibitor of UMP Synthase. Cell. Apr 6;169(2):258–272.e17. https://doi.org/10.1016/j.cell.2017.03.023. PMID: 28388410; PMCID: PMC5421364.

Chapter 11

Functional Pathway Inference Analysis (FPIA)

Irbaz I. Badshah and Pedro R. Cutillas

Abstract

Pathway inference methods allow the mapping of biochemical networks, the discovery of signaling components, and the assignment of functions to understudied proteins and genes. Literature and automated text mining have been successfully used to reconstruct metabolic and signaling circuits, while gene regulatory networks may be inferred from gene expression data. As an alternative approach to map members of proliferative pathways, functional pathway inference analysis (FPIA) is based on the premise that genes producing similar phenotypes following perturbation across multiple cell lines belong to a common pathway. We have demonstrated this concept with the use of gene dependency datasets that allow the provision of probabilistic values of pathway membership for thousands of genes. Here, we provide a detailed protocol for the implementation of FPIA in the *'cordial'* R package. As an illustration of how FPIA may be used to identify new pathway members, we present a step-by-step description of its use for the investigation of genes functionally associated to PI3K and TP53.

Key words Cancer, Cell signaling, Cordial, CRISPR-Cas9, FPIA, Functional pathway inference analysis, Gene dependency, Network, Pathway, R, RNAi

1 Introduction

Bioinformatic methods play a crucial role in inferring biochemical pathways, providing insights into the mechanisms of biochemical processes, and identifying potential drug targets [1]. These methods complement experimental approaches and are essential for annotating the genome and discovering signaling members within biochemical networks [2]. Once mapped into pathways, protein and gene sets may be used to infer pathway activities from transcriptomic or proteomic data by performing enrichment analysis, e.g., gene set enrichment analysis (GSEA) or overrepresentation analysis, focusing on groups/sets of genes sharing common biological functions [3, 4].

An approach to biochemical pathway inference involves systematic automated literature search, which reconstructs biochemical pathways by providing structured data derived from experimental

Alexander Krasnitz and Pascal Belleau (eds.), *Cancer Bioinformatics*, Methods in Molecular Biology, vol. 2932,
https://doi.org/10.1007/978-1-0716-4566-6_11,

observations in published literature [5]. These reconstructions serve as valuable resources that abstract essential information on the biochemical transformations and can be restricted to the analysis of specific target organisms [6]. Automated text-mining techniques and the integration of biological databases are employed to initiate the reconstruction of molecular circuits, providing the foundational data for pathway reconstruction [7, 8]. However, some representations of biochemical pathways may reduce network complexity and result in ambiguous representations [9], and inconsistencies between public resources containing literature-curated signaling pathways have been noted [10].

As an alternative to systematic literature search and text mining, gene regulatory networks (GRNs) have attracted significant interest as a means to infer cell-type specific pathway circuitry, leading to the development of numerous methods for their statistical inference from gene expression data [11]. These methods aim to understand gene interactions and regulatory mechanisms, with the aim of deciphering gene functions and cellular dynamics [12]. Various computational approaches have been proposed for inferring GRNs, including bidirectional recurrent neural networks for single-cell transcriptomic data [13], multilevel strategies for large-scale network inference [14], and relaxed graph matching for network inference [15]. Additionally, fuzzy cognitive maps and Bayesian networks have been utilized for improved inference of GRNs based on time series data and multi-omics data integration, respectively [16, 17].

We recently proposed a methodology for identifying components of biochemical pathways involved in cell proliferation and viability [1]. The approach is based on the premise that genes belonging to the same pathway would produce the same antiproliferative phenotypes across cell models when such genes are silenced using genetic means, such as by CRISPR or RNA interference (RNAi). To test this concept, we developed a method named functional pathway inference analysis (FPIA), which uses datasets of systematic gene perturbations across large panels of cell line data as input. Through the use of gene dependency data of genome-wide systematic genetic perturbation screens across multiple cancer cell lines, the calculation of pairwise correlations reveals signaling mediators with concordant survival phenotypes that are potential elements of a shared pathway [1]. This chapter provides a protocol for FPIA implemented in the freely available R package *'cordial'* [1].

Table 1
Recommended hardware

	Specification
System type	64-bit operating system; x64-based processor
Processor	Intel® Core™ i7-10610U CPU @ 1.80 GHz; Base speed 2.30 GHz
Installed RAM	64.0 GB (minimum recommended: 16 GB)

Hardware used in the creation and testing of *'cordial'*

2 Materials

2.1 Hardware

- The *'cordial'* package was developed and tested on a system with the hardware specification detailed in Table 1.

2.2 Software

- The required software is presented in Table 2.
- The R programming language and environment, its packages, additional build tools, and RStudio integrated development environment (IDE) are available for several Linux distributions, macOS, and Windows operating systems (OS); consult the official websites of the software for more information (Table 2).

2.3 'cordial'

- The *'cordial'* R package provides functions to compute pairwise Pearson's correlations of a dataset in whole or of specified targets simultaneously in parallel. Conveniently, it includes the ability to filter the input dataset and select a subset of columns to compute correlations. The functions of the package output Pearson's product moment correlation coefficients, p-values, adjusted p-values (q-values), linear model slope, and observation counts in a long-format data structure (*'data.table'*).
- The *'cordial'* R package was built on a 64-bit Windows 10 and Windows 11 system; however, the codebase was constructed to be OS-independent. Specifically, an asynchronous multisession parallel backend was implemented which is compatible with Windows and Unix-like systems (Linux, macOS).

3 Methods

- The methods herein assume the use of the R programming language and environment via the RStudio IDE, where R code is run from an R console or source editor.

3.1 Installation

- Consult the official websites of the software listed in Table 2 for OS-specific installation instructions for the current releases.

Table 2
Software requirements

Software	Version	Notes	Website URL
R [33]	≥3.5.0	OS-specific installers can be found by first selecting a Comprehensive R Archive Network (CRAN) mirror close to the user's location	https://cran.r-project.org/
Additional build tools	(Specific to OS and R versions)	Required to build R packages from source, such as from a GitHub repository: Linux, macOS, Windows	https://support.posit.co/hc/en-us/articles/200486498-Package-Development-Prerequisites https://cran.r-project.org/bin/linux/ https://cran.r-project.org/bin/macosx/tools https://cran.r-project.org/bin/windows/Rtools/
RStudio	(Latest available)	Although not an absolute requirement, for ease of use, an IDE is recommended	https://posit.co/download/rstudio-desktop/
'devtools' [34]	(Latest available)	R package (Subheading 3.1)	https://devtools.r-lib.org/index.html
'magrittr' [35]	≥2.0.1	R package (Subheading 3.1)	https://magrittr.tidyverse.org/index.html
'tidyr' [36]	≥1.1.4	R package (Subheading 3.1)	https://tidyr.tidyverse.org/index.html
'purrr' [37]	≥0.3.4	R package (Subheading 3.1)	https://purrr.tidyverse.org/index.html
'future' [38]	≥1.23.0	R package (Subheading 3.1)	https://future.futureverse.org/index.html
'furrr' [39]	≥0.2.3	R package (Subheading 3.1)	https://furrr.futureverse.org/index.html
'collapse' [40]	≥1.7.2	R package (Subheading 3.1)	https://sebkrantz.github.io/collapse/index.html
'data.table' [41]	≥1.14.2	R package (Subheading 3.1)	https://rdatatable.gitlab.io/data.table/index.html https://rdatatable.gitlab.io/data.table/articles/datatable-intro.html
'cordial' [1]	(Latest available)	R package (Subheading 3.1)	https://github.com/CutillasLab/cordial
'ggplot2' [42]	(Latest available)	R package; optional (Subheading 3.1). For creating graphics and plots	https://ggplot2.tidyverse.org/index.html
'ggrepel' [43]	(Latest available)	R package; optional (Subheading 3.1). Provides additional features for *'ggplot2'*	https://ggrepel.slowkow.com/

Required software for FPIA; install in the order listed

1. Install R (Table 2).
2. Install tools for building R packages from source (such as from GitHub), specific to the version of R and OS (Table 2).
3. Install RStudio (Table 2).
4. Install *'cordial'* package dependencies by executing in an R console or editor (Table 2):
   (a) *'devtools'*
   (b) *'magrittr'*
   (c) *'tidyr'*
   (d) *'purrr'*
   (e) *'future'*
   (f) *'furrr'*
   (g) *'collapse'*
   (h) *'data.table'*
   (i) *'ggplot2'* (optional)
   (j) *'ggrepel'* (optional)

```
if (!require("devtools")) { # check if not installed
 install.packages("devtools") # install
 library(devtools) # load
}
install.packages("magrittr")
install.packages("tidyr")
install.packages("purrr")
install.packages("future")
install.packages("furrr")
install.packages("collapse")
install.packages("data.table")
install.packages("ggplot2")
install.packages("ggrepel")
```

5. Install the *'cordial'* package by any of the following methods (Table 2):
   (a) Direct from GitHub.

```
devtools::install_github("CutillasLab/cordial@*release")
```

   (b) Manual download.
      (i) Download the Package Archive File (cordial_x.x.x.tar.gz) of the latest release from the GitHub repository (Table 2).

(ii) Run the following, replacing the string argument to '*path*' with the actual location:

```
devtools::install_local(path = "C:/path/to/cordial_x.x.x.tar.gz")
```

(c) RStudio graphical user interface (GUI).

(i) Download the Package Archive File (cordial_x.x.x.tar.gz) of the latest release from the GitHub repository (Table 2).

(ii) In RStudio, click *Tools* menu.

(iii) Select *Install Packages…* .

(iv) In the *Install from* list box, select *Package Archive File (.zip; .tar.gz)*.

(v) Click *Browse* to select the downloaded '*cordial*' Package Archive File.

(vi) Select *Install*.

3.2 *Load* 'cordial'

- The *'cordial'* R package must be loaded (makes functions, data, and code available) and attached (places the package in the search path of available R objects so that function definitions can be found) by executing at the beginning, once per session:

```
library(cordial)
```

- The package documentation can be consulted by prepending *'?'* to the package name; this method can be used for any object including functions and data:

```
?cordial
```

- All subsequent code examples assume the *'cordial'* package has been loaded, in addition to any optional packages (*'ggplot2'*, *'ggrepel'*).

3.3 *Data*

- Datasets included in *'cordial'* are from the Cancer Dependency Map (DepMap)—a collaboration of the Broad Institute (Cambridge, Massachusetts, USA) and the Wellcome Sanger Institute (Hinxton, Cambridgeshire, UK) [18–22].
- *'cordial'* contains three datasets from the DepMap project:
  1. *'cellmeta_DT'*: cancer cell line metadata
  2. *'rnai_DT'*: cancer cell line genetic dependencies from RNAi screens
  3. *'crispr_DT'*: cancer cell line genetic dependencies from CRISPR-Cas9 screens

3.3.1 'cellmeta_DT'

- Cancer cell line sample information [23–25].
- A '*data.table*' with 1811 rows (cell lines) and 26 columns (cell line metadata).
  1. *depmap_id*: cell line DepMap ID. The '*data.table*' key (improves performance of binary search, joins, grouping, and indexing),
  2. *cell_line_name*: cell line name,
  3. *stripped_cell_line_name*: stripped cell line name,
  4. *CCLE_Name*: Cancer Cell Line Encyclopaedia name,
  5. *Alias*: alias,
  6. *COSMICID*: Catalogue Of Somatic Mutations In Cancer ID,
  7. *sex*: sex of individual from which sample was derived,
  8. *source*: tissue sample source,
  9. *Achilles_n_replicates*: Achilles number of replicates,
  10. *cell_line_NNMD*: cell line null-normalized median difference,
  11. *culture_type*: cell culture type,
  12. *culture_medium*: culture medium,
  13. *cas9_activity*: Cas9 activity,
  14. *RRID*: research resource identifier,
  15. *WTSI_Master_Cell_ID*: Wellcome Trust Sanger Institute ID,
  16. *sample_collection_site*: sample collection site,
  17. *primary_or_metastasis*: primary or metastatic cancer cell line,
  18. *primary_disease*: primary disease,
  19. *Subtype*: subtype disease,
  20. *age*: age of individual from which sample was derived,
  21. *Sanger_Model_ID*: Sanger model ID,
  22. *depmap_public_comments*: additional information,
  23. *lineage*: cancer cell line lineage,
  24. *lineage_subtype*: cancer cell line lineage subtype,
  25. *lineage_sub_subtype*: cancer cell line lineage sub-subtype,
  26. *lineage_molecular_subtype*: cancer cell line lineage molecular subtype.

3.3.2 'rnai_DT'

- Cancer cell line genetic dependencies estimated using the DEMETER2 model applied to three large-scale RNAi screening datasets: the Broad Institute Project Achilles, Novartis Project DRIVE, and the Marcotte et al. breast cell line dataset [21, 26–29].
- A *'data.table'* with 711 rows (cell lines) and 16,816 columns (6 cell line metadata, 16,810 genes).
    1. *depmap_id*: cell line DepMap ID. The *'data.table'* key (improves performance of binary search, joins, grouping, and indexing),
    2. *cell_line_display_name*: cell line display name,
    3. *lineage_1*: cancer cell line lineage,
    4. *lineage_2*: cancer cell line lineage subtype,
    5. *lineage_3*: cancer cell line lineage sub-subtype,
    6. *lineage_4*: cancer cell line lineage molecular subtype,
    7. … 16,816: [Genes].

3.3.3 'crispr_DT'

- Cancer cell line genetic dependencies estimated using the CERES model applied to the Avana library CRISPR-Cas9 genome-scale knockout (prefixed with Achilles) [20, 25, 30, 31].
- A *'data.table'* with 945 rows (cell lines) and 17,648 columns (6 cell line metadata, 17,642 genes).
    1. *depmap_id*: cell line DepMap ID. The *'data.table'* key (improves performance of binary search, joins, grouping and indexing),
    2. *cell_line_display_name*: cell line display name,
    3. *lineage_1*: cancer cell line lineage,
    4. *lineage_2*: cancer cell line lineage subtype,
    5. *lineage_3*: cancer cell line lineage sub-subtype,
    6. *lineage_4*: cancer cell line lineage molecular subtype,
    7. … 17,648: [Genes].

3.3.4 Usage

- The datasets can be accessed as follows; for further information, consult *'data.table'* documentation (Table 2):

```
# access data object
cellmeta_DT
rnai_DT
crispr_DT
```

```
# view in tabulated spreadsheet presentation
View(cellmeta_DT)
View(rnai_DT)
View(crispr_DT)

# index data
# syntax: DT[i, j, by]
# DT[
# subset/reorder rows using i,
# select/calculate columns using j,
# grouped according to by
# ]
# an empty argument in i, j or by indicates no subsetting
# change `by` to `keyby` to sort results

# EXAMPLES ####

# all rows, column subset, grouped ----
# `.()` = alias of `list()` in `data.table`
cellmeta_DT[
 ,
 .(stripped_cell_line_name, sample_collection_site),
 by = .(primary_disease, Subtype)
]

# Boolean index, column subset using indices ----
rnai_DT[
 lineage_1 == "Breast"
 & lineage_2 == "Breast Carcinoma",
 1:6
]

# Boolean index using variable, column subset excluding
indices ----

# create character vector of cell lines
cell_primary <- cellmeta_DT[
 primary_or_metastasis == "Primary",
 stripped_cell_line_name
]

# index and return all dependency data
crispr_DT[cell_line_display_name %in% cell_primary, !1:6]

# index and return dependency data for specific genes ----

# create character vector of specific genes
```

```
# 'data.table' supports chained operations
# '.SD' = subset of data:
# contains data for the current group in 'by'
targets <- crispr_DT[, 7:16][, colnames(.SD)]

# index first 5 rows for specific genes
# '..' = access variable from outside the 'data.table'
crispr_DT[1:5, ..targets]

# calculate means for each column of target genes, ----
# per 'lineage_1' group

# create character vector of specific genes
targets <- crispr_DT[, 7:16][, colnames(.SD)]

# apply mean for each 'lineage_1' to each gene column in
'targets'
# '.SDcols' = columns to compute on
crispr_DT[, lapply(.SD, mean), keyby = lineage_1, .SDcols =
targets]
```

3.4 Functions

- There are three main functions of *'cordial'* that perform a correlation analysis to implement FPIA:
  1. *'cor_map()'*: correlation analysis of a dataset
  2. *'cor_target_map()'*: correlation analysis of multiple targets in parallel
  3. *'cor_target()'*: correlation analysis of a single target in parallel
- There are two additional functions to set up an asynchronous multisession backend for parallel computing:
  1. *'start_parallel()'*: begin a parallel processing plan
  2. *'end_parallel()'*: begin a sequential processing plan

3.4.1 'start_parallel()'

- *'start_parallel()'* creates a multisession *'future::plan()'* for asynchronous (parallel) processing in separate R sessions running in the background of the same machine; replaces any previously set *'plan'*.
- If a multisession *'plan'* is set, *'cor_target()'* and *'cor_target_map()'* execute in parallel; the result is returned to the main R session.
- Run *'start_parallel()'* at the beginning of working with *'cordial'* functions. Alternatively, users may set their own *'plan'* for more

control. To return to sequential processing, see *'end_parallel()'* (Subheading 3.4.2).

Call

```
start_parallel(logical_cores = FALSE)
```

Arguments

- ***'logical_cores'***: a logical scalar. The number of parallel R sessions is set to the number of physical CPUs/cores if *'FALSE'* (default) or logical CPUs/cores if *'TRUE'*.

Usage

```
# create parallel multisessions equal to the number of physical
cores ----
start_parallel() # Default
start_parallel(FALSE) # Same as above

# create parallel multisessions equal to the number of logical
cores ----
start_parallel(TRUE)

# execute 'cordial' functions...
```

3.4.2 'end_parallel()'

- *'end_parallel()'* creates a sequential *'future::plan()'* for synchronous processing in the current R session; replaces any previously set *'plan'*.
- Run *'end_parallel()'* at the end of working with *'cordial'* functions to return to sequential processing. Alternatively, users may set their own *'plan'* explicitly for more control.

Call

```
end_parallel()
```

Arguments

- [None].

Usage

```
# execution of 'cordial' functions...

# begin sequential processing ----
end_parallel()
```

3.4.3 'cor_map()': Correlation Analysis of a Dataset

Correlation Analysis

- *'cor_map()'* uses *'collapse::pwcor()'* to test for an association between paired samples computed with Pearson's product moment correlation coefficient (same as *'cor.test()'*). Correlations are computed on complete pairs of observations for each pair of variables (same as *'cor(..., use = "pairwise.complete.obs")'*).
- Adjusted *p*-values are computed with *'p.adjust()'* using one of the *'p.adjust.methods'*: *'c("holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "none")'*. Default *'"BH"'* (alias *'"fdr"'*) is the Benjamini-Hochberg false discovery rate multiple testing adjustment method [32].

Subset

- *'cor_map()'* conveniently allows filtering (*'filter_rows'*) of the input dataset by performing a cross-join (*'CJ'*) with a named list referring to values present within the dataset itself, or a separate metadata *'data.table'*.
- If the dataset contains non-numeric columns, they must be omitted by selecting (*'select_cols'*) the columns to compute pairwise correlations. This mechanism also allows limiting of the correlations to perform.
- The subsetting algorithm is identical to that in *'cor_target_map()'* and *'cor_target()'*.

Context

- *'cor_map()'* differs from *'cor_target()'* in that correlations are computed for all pairs of columns specified in *'select_cols'*, whereas *'cor_target()'* computes pairwise correlations for a single specified *'target'* column, with correlations limited to the columns specified in *'select_cols'*.
- *'cor_target_map()'* varies from *'cor_target()'* in that it allows specifying multiple elements in *'target'*.

Output

- A *'data.table'* in long-format is returned with no pairwise duplicates (*corr(X, Y)* without *corr(Y, X)*); if *'filter_rows'* has been supplied, the filters are included:
  1. *Target*,
  2. *Correlation*,
  3. Pearson's product moment correlation coefficients (*r*),
  4. *p*-values (*p*),
  5. Adjusted *p*-values (*q*),
  6. Observation counts (*n*),
  7. … [Filters].

Call

```
cor_map(
 dataset,
 select_cols = colnames(dataset),
 filter_rows = NULL,
 metadata = NULL,
 self = "yes",
 method = "BH"
)
```

Arguments

- *'dataset'*: a *'data.table'*. Must be in column-major order.
- *'select_cols'*: a vector of column names (character) or indices (numeric). The columns to use for computing correlations, which must be of type numeric.
- *'filter_rows'*: a named *'list()'*. Values specify which rows to subset. Names correspond to column names in *'dataset'* or *'metadata'* if supplied.
- *'metadata'*: a *'data.table'*. Must be in column-major order. Optional input containing data to filter *'dataset'* by. If supplied, *'metadata'* and *'dataset'* must possess the same key (*'data.table::setkey()'*) column.
- *'self'*: a character scalar. Self-correlations are included if *'"yes"'* (default) or omitted if *'"no"'*.
- *'method'*: A character scalar. Correction method for *p*-value adjustment, passed to *'p.adjust()'*. One of *'c("holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "none")'*; *'"BH"'* (alias *'"fdr"'*) (default).

Usage: Dataset

```
# execute function
results <- cor_map(
 dataset = crispr_DT,
 select_cols = colnames(crispr_DT[, !1:6])
)

# add column for annotating plots
results <-
 results[, `=`(
  Direction = data.table::fifelse(r > 0, "Positive", "Nega-
tive")
 )]
```

Usage: Dataset Filtered by Metadata

```
# create vector of gene targets
genes <- c(
 "AKT1", "AKT2", "AKT3", "MDM2", "PIK3CA", "PIK3CB", "PIK3CD",
 "PIK3R1", "PIK3R2", "PIK3R3", "TP53"
)

# filter using columns in 'metadata'
# (to filter 'dataset' directly without 'metadata':
# use column names and values in 'dataset' for 'filter_rows')
results <- cor_map(
 dataset = crispr_DT,
 select_cols = genes,
 filter_rows = list(lineage = "ovary"),
 metadata = cellmeta_DT
)

# add column for annotating plots
results <-
 results[, `:=`(
  Direction = data.table::fifelse(r > 0, "Positive", "Nega-
tive")
 )]
```

Usage: Dataset by Grouping Variable

```
# create vector of gene targets
genes <- c(
 "AKT1", "AKT2", "AKT3", "MDM2", "PIK3CA", "PIK3CB", "PIK3CD",
 "PIK3R1", "PIK3R2", "PIK3R3", "TP53"
)

# create vector of grouping variables
unique_lineages <- crispr_DT[, unique(lineage_1)]

# map function to multiple groups
results <- purrr::map(
 .x = unique_lineages,
 .f = ~ cor_map(
 dataset = crispr_DT,
 select_cols = genes,
 filter_rows = list(lineage_1 = .x)
 )
)

# assign names to list elements
```

```
results <- purrr::set_names(results, unique_lineages)

# keep data that is not empty (from too few observations)
results <- purrr::compact(.x = results, .p = ~ nrow(.x))

# combine output into single `data.table`
results <- data.table::rbindlist(results)

# add column for annotating plots
results <-
 results[, `:=`(
  Direction = data.table::fifelse(r > 0, "Positive", "Negative")
 )]
```

3.4.4 'cor_target_map()': Correlation Analysis of Multiple Targets

- *'cor_target_map()'* uses *'cor.test()'* to test for an association between paired samples computed with Pearson's product moment correlation coefficient. Correlations are computed with incomplete cases removed (*'cor.test(..., na.action = "na.omit")'*).

Correlation Analysis

- Adjusted *p*-values are computed with *'p.adjust()'* using one of the *'p.adjust.methods'*: *'c("holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "none")'*. Default *'"BH"'* (alias *'"fdr"'*) is the Benjamini-Hochberg false discovery rate multiple testing adjustment method [32].

Subset

- *'cor_target_map()'* conveniently allows filtering (*'filter_rows'*) of the input dataset by performing a cross-join (*'CJ'*) with a named list referring to values present within the dataset itself or a separate metadata *'data.table'*.
- If the dataset contains non-numeric columns, they must be omitted by selecting (*'select_cols'*) the columns to compute pairwise correlations. This mechanism also allows limiting of the correlations to perform.
- The subsetting algorithm is identical to that in *'cor_target()'* and *'cor_map()'*.

Parallelization

- *'cor_target_map()'* computes correlations in parallel if an asynchronous *'future::plan()'* is set prior to executing *'cor_target_map()'*. See *'start_parallel()'* (Subheading 3.4.1).
- Internally, *'furrr::future_map()'* is used to map simultaneously in parallel each element in *'target'* for processing via *'cor.test()'*. Specifically, *'cor_target_map()'* by default maps *'cor_targets()'* (a sequential variant of *'cor_target()'*) to avoid nested parallel

operations; otherwise, the nested parallel operations both attempt to utilize the full complement of CPUs/cores which would result in inefficient load balancing.

Context

- *'cor_target_map()'* differs from *'cor_map()'* in that pairwise correlations are computed for multiple specified *'target'* columns, with correlations limited to the columns specified in *'select_cols'*, whereas *'cor_map()'* computes correlations for all pairs of columns specified in *'select_cols'*.
- *'cor_target_map()'* varies from *'cor_target()'* in that it allows specifying multiple elements in *'target'*.

Output

- A *'data.table'* in long-format is returned with no pairwise duplicates (*corr(X, Y)* without *corr(Y, X)*); if *'filter_rows'* has been supplied, the filters are included:
  1. *Target*,
  2. *Correlation*,
  3. *Slope*,
  4. Pearson's product moment correlation coefficients (*r*),
  5. *p*-values (*p*),
  6. Adjusted *p*-values (*q*),
  7. … [Filters].

Call

```
cor_target_map(
 dataset,
 target,
 select_cols = colnames(dataset),
 filter_rows = NULL,
 metadata = NULL,
 self = "yes",
 method = "BH",
 fun = cordial::cor_targets
)
```

Arguments

- ***'dataset'***: a *'data.table'*. Must be in column-major order.
- ***'target'***: a character vector. Column names in *'dataset'* to compute correlations with (specified in *'select_cols'*), which must be of type numeric.

- *'select_cols'*: a vector of column names (character) or indices (numeric). The columns to use for computing correlations, which must be of type numeric.
- *'filter_rows'*: a named *'list()'*. Values specify which rows to subset. Names correspond to column names in *'dataset'* or *'metadata'* if supplied.
- *'metadata'*: a *'data.table'*. Must be in column-major order. Optional input containing data to filter *'dataset'* by. If supplied, *'metadata'* and *'dataset'* must possess the same key (*'data.table::setkey()'*) column.
- *'self'*: a character scalar. Self-correlations are included if *'"yes"'* (default) or omitted if *'"no"'*.
- *'method'*: a character scalar. Correction method for *p*-value adjustment, passed to *'p.adjust()'*. One of *'c("holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "none")'*; *'"BH"'* (alias *'"fdr"'*) (default).
- *'fun'*: a function. Currently, only compatible with *'cordial::cor_targets()'* (default) or *'cordial::cor_target()'*. See Parallelization (Subheading 3.4.4.3).

Usage: Multiple Targets

```
# create vector of gene targets
genes <- c(
   "AKT1", "AKT2", "AKT3", "MDM2", "PIK3CA", "PIK3CB",
"PIK3CD",
  "PIK3R1", "PIK3R2", "PIK3R3", "TP53"
)

# execute function for multiple targets
# 'select_cols' must also include the 'target' columns
results <- cor_target_map(
  dataset = crispr_DT,
  target = genes,
  select_cols = c(genes, colnames(crispr_DT[, 7:17]))
)

# add column for annotating plots
results <-
  results[, `:=`(
      Direction = data.table::fifelse(r > 0, "Positive",
"Negative")
 )]
```

Usage: Multiple Targets Filtered by Metadata

```
# create vector of gene targets
genes <- c(
     "AKT1", "AKT2", "AKT3", "MDM2", "PIK3CA", "PIK3CB",
"PIK3CD",
   "PIK3R1", "PIK3R2", "PIK3R3", "TP53"
)

# filter using columns in `metadata`
# (to filter `dataset` directly without `metadata`:
# use column names and values in `dataset` for `filter_rows`)
results <- cor_target_map(
  dataset = crispr_DT,
   target = genes,
   select_cols = genes,
   filter_rows = list(lineage = c("breast", "ovary")),
 metadata = cellmeta_DT
)

# add column for annotating plots
results <-
  results[, `=`(
      Direction = data.table::fifelse(r > 0, "Positive",
"Negative")
 )]
```

Usage: Multiple Targets by Grouping Variable

```
# create vector of gene targets
genes <- c(
   "AKT1", "AKT2", "AKT3", "MDM2", "PIK3CA", "PIK3CB",
"PIK3CD",
   "PIK3R1", "PIK3R2", "PIK3R3", "TP53"
)

# create vector of grouping variables
lineages <- c("Breast", "Ovary")

# map function to multiple targets for each group
results <- purrr::map(
  .x = lineages,
  .f = ~ cor_target_map(
  dataset = crispr_DT,
  target = genes,
  select_cols = genes,
  filter_rows = list(lineage_1 = .x)
 )
)
```

```
# assign names to list elements
results <- purrr::set_names(results, lineages)

# keep data that is not empty (from too few observations)
results <- purrr::compact(.x = results, .p = ~ nrow(.x))

# combine output into single 'data.table'
results <- data.table::rbindlist(results)

# add column for annotating plots
results <-
  results[, `:=`(
    Direction = data.table::fifelse(r > 0, "Positive", "Negative")
  )]
```

3.4.5 'cor_target()': Correlation Analysis of a Single Target

Correlation Analysis

- *'cor_target()'* uses *'cor.test()'* to test for an association between paired samples computed with Pearson's product moment correlation coefficient. Correlations are computed with incomplete cases removed (*'cor.test(..., na.action = "na.omit")'*).
- Adjusted *p*-values are computed with *'p.adjust()'* using one of the *'p.adjust.methods'*: *'c("holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "none")'*. Default *'"BH"'* (alias *'"fdr"'*) is the Benjamini-Hochberg false discovery rate multiple testing adjustment method [32].

Subset

- *'cor_target()'* conveniently allows filtering (*'filter_rows'*) of the input dataset by performing a cross-join (*'CJ'*) with a named list referring to values present within the dataset itself or a separate metadata *'data.table'*.
- If the dataset contains non-numeric columns, they must be omitted by selecting (*'select_cols'*) the columns to compute pairwise correlations. This mechanism also allows limiting of the correlations to perform.
- The subsetting algorithm is identical to that in *'cor_target_map()'* and *'cor_map()'*.

Parallelization

- *'cor_target()'* computes correlations in parallel if an asynchronous *'future::plan()'* is set prior to executing *'cor_target()'*. See *'start_parallel()'* (Subheading 3.4.1).
- Internally, *'furrr::future_map2()'* is used to map simultaneously in parallel each column in *'select_cols'* with the *'target'* for processing via *'cor.test()'*.

Context

- *'cor_target()'* differs from *'cor_map()'* in that pairwise correlations are computed for a single specified *'target'* column, with correlations limited to the columns specified in *'select_cols'*, whereas *'cor_map()'* computes correlations for all pairs of columns specified in *'select_cols'*.
- *'cor_target_map()'* varies from *'cor_target()'* in that it allows specifying multiple elements in *'target'*.

Output

- A *'data.table'* in long-format is returned with no pairwise duplicates (*corr(X, Y)* without *corr(Y, X)*); if *'filter_rows'* has been supplied, the filters are included:
  1. *Target*,
  2. *Correlation*,
  3. *Slope*,
  4. Pearson's product moment correlation coefficients (*r*),
  5. *p*-values (*p*),
  6. Adjusted *p*-values (*q*),
  7. … [Filters].

Call

```
cor_target(
  dataset,
  target,
  select_cols = colnames(dataset),
  filter_rows = NULL,
  metadata = NULL,
  self = "yes",
  method = "BH"
)
```

Arguments

- ***'dataset'***: a *'data.table'*. Must be in column-major order.
- ***'target'***: a character scalar. A column name in *'dataset'* to compute correlations with (specified in *'select_cols'*), which must be of type numeric.
- ***'select_cols'***: a vector of column names (character) or indices (numeric). The columns to use for computing correlations, which must be of type numeric.
- ***'filter_rows'***: a named *'list()'*. Values specify which rows to subset. Names correspond to column names in *'dataset'* or *'metadata'* if supplied.

- *'**metadata**'*: a *'data.table'*. Must be in column-major order. Optional input containing data to filter *'dataset'* by. If supplied, *'metadata'* and *'dataset'* must possess the same key (*'data.table::setkey()'*) column.
- *'**self**'*: a character scalar. Self-correlations are included if *'"yes"'* (default) or omitted if *'"no"'*.
- *'**method**'*: a character scalar. Correction method for *p*-value adjustment, passed to *'p.adjust()'*. One of: *'c("holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "none")'*; *'"BH"'* (alias *'"fdr"'*) (default).

Usage: Single Target

```
# execute function
results <- cor_target(
  dataset = crispr_DT,
  target = "A1BG",
  select_cols = colnames(crispr_DT[, !1:6])
)

# add column for annotating plots
results <-
   results[, `:=`(
      Direction = data.table::fifelse(r > 0, "Positive",
"Negative")
 )]
```

Usage: Single Target Filtered by Metadata

```
# filter using columns in 'metadata'
# (to filter 'dataset' directly without 'metadata':
# use column names and values in 'dataset' for 'filter_rows')
results <- cor_target(
  dataset = crispr_DT,
  target = "A1BG",
  select_cols = colnames(crispr_DT[, !1:6]),
  filter_rows = list(lineage_subtype = "breast_carcinoma"),
  metadata = cellmeta_DT
)

# add column for annotating plots
results <-
  results[, `:=`(
      Direction = data.table::fifelse(r > 0, "Positive",
"Negative")
 )]
```

Usage: Single Target by Grouping Variable

```
# create vector of unique grouping variables
unique_lineages <- crispr_DT[, unique(lineage_1)]

# map function to a single target for each group
results <- purrr::map(
  .x = unique_lineages,
  .f = ~ cor_target(
  dataset = crispr_DT,
  target = "A1BG",
  select_cols = colnames(crispr_DT[, !1:6]),
  filter_rows = list(lineage_1 = .x)
 )
)

# assign names to list elements
results <- purrr::set_names(results, unique_lineages)

# keep data that is not empty (from too few observations)
results <- purrr::compact(.x = results, .p = ~ nrow(.x))

# combine output into single `data.table`
results <- data.table::rbindlist(results)

# add column for annotating plots
results <-
 results[, `:=`(
  Direction = data.table::fifelse(r > 0, "Positive", "Nega-
tive")
 )]
```

4 Anticipated Results

- Example usage of visualizing correlations for a subset of exemplar genes in PI3K and TP53 pathways across all cell lines is shown below. The output of such analysis returned genes known to be associated with these pathways (Fig. 1):

```
# create vector of gene targets
genes <- c(
  "AKT1", "AKT2", "AKT3", "MTOR", "PDPK1", "PIK3CA", "PIK3CB",
  "PIK3CD", "PIK3R1", "PIK3R2", "PIK3R3", "PTEN", "TP53"
)

# execute function for specified genes
results <- cor_target_map(
```

```
 dataset = crispr_DT,
 target = genes,
 select_cols = colnames(crispr_DT[, !1:6]),
 self = "no"
)

# create subset for labels:
# top 6 correlations by q-value for each `Target`
annotations <- results[order(q)][q < 0.05, head(.SD, 6), by =
Target]

# get maximum absolute value of 'Slope'
max_abs_slope <- results[, max(abs(Slope))]

# create plot
plot_results <- ggplot(
  results[q < 0.05],
  aes(x = r, y = -log10(q), colour = Slope)) +
  geom_point(
  data = results[q > 0.05], # non-significant points
  colour = "grey",
  alpha = 0.5
 ) +
 geom_point(alpha = 0.5) + # significant points from initial
`ggplot()`
   geom_label_repel(
    data = annotations,
    aes(label = Correlation),
    colour = "black",
    size = 2.5,
    force = 5,
    force_pull = 0.5,
    direction = "y",
    nudge_y = 3,
    nudge_x = 2,
    label.padding = 0.15,
    segment.size = 0.3
 ) +
 geom_hline(
   yintercept = -log10(0.05),
   linetype = "dashed",
   linewidth = 0.4,
   alpha = 0.5
 ) +
 geom_vline(
   xintercept = 0,
   linetype = "dashed",
   linewidth = 0.4,
```

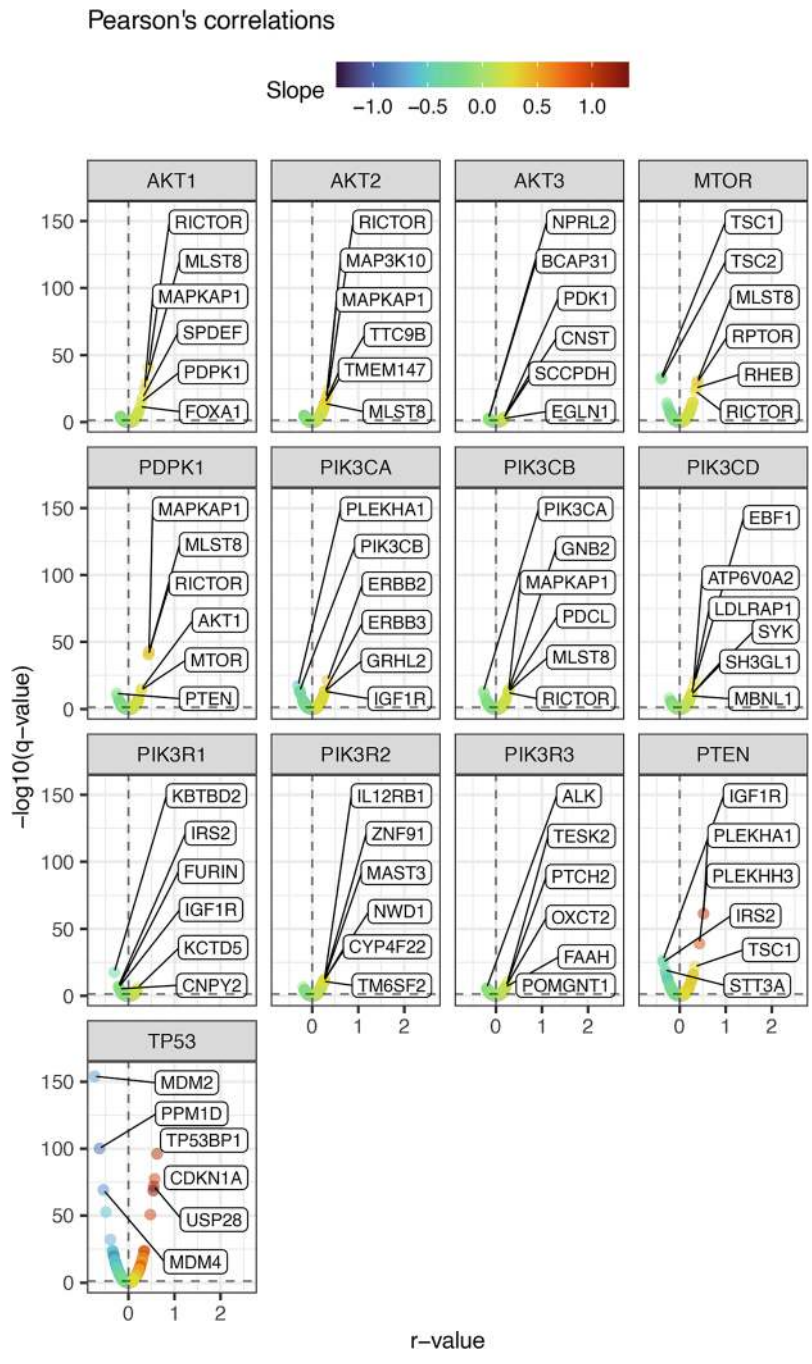


Fig. 1 Pearson's correlations across all cell lines. Pearson's correlations of genes significantly associated with specified target genes across multiple cancer cell lines from CRISPR-Cas9 knockout data. Top 6 correlations (labeled) by *q*-value for each target (shown in graph headers) with self-correlations omitted; horizontal line: $-\log_{10}(q\text{-value}) = 0.05$; vertical line: $r\text{-value} = 0$

```
    alpha = 0.5
  ) +
  facet_wrap(vars(Target)) +
  scale_color_viridis_c(
    option = "H",
    limits = c(-max_abs_slope, max_abs_slope),
    guide = guide_colourbar(barwidth = 8, barheight = 0.7)
  ) +
  theme_bw() +
  theme(
    legend.position = "top",
    legend.title = element_text(size = 9),
    legend.text = element_text(size = 8),
    plot.title = element_text(size = 10),
    strip.text = element_text(size = 8),
    axis.title = element_text(size = 9),
    axis.text = element_text(size = 8)
  ) +
  labs(
    title = "Pearson's correlations",
    x = "r-value",
    y = "-log10(q-value)",
  )

# view plot
plot_results

# save plot
ggsave("Fig 1.pdf", width = 4.5, height = 7.5, units = "in")
```

References

1. Badshah II, Cutillas PR (2023) Systematic identification of biochemical networks in cancer cells by functional pathway inference analysis. Bioinformatics 39(1). https://doi.org/10.1093/bioinformatics/btac769
2. Hijazi M et al (2020) Reconstructing kinase network topologies from phosphoproteomics data reveals cancer-associated rewiring. Nat Biotechnol 38(4):493–502. https://doi.org/10.1038/s41587-019-0391-9
3. Subramanian A et al (2005) Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. Proc Natl Acad Sci. https://doi.org/10.1073/pnas.0506580102
4. Kim SY, Volsky DJ (2005) PAGE: parametric analysis of gene set enrichment. BMC Bioinformatics 6:144. https://doi.org/10.1186/1471-2105-6-144
5. Chowdhury S, Sarkar RR (2015) Comparison of human cell signaling pathway databases—evolution, drawbacks and challenges. Database. https://doi.org/10.1093/database/bau126
6. Thiele I, Palsson BØ (2010) A protocol for generating a high-quality genome-scale metabolic reconstruction. Nat Protoc. https://doi.org/10.1038/nprot.2009.203
7. Usié A et al (2014) Biblio-MetReS for user-friendly mining of genes and biological processes in scientific documents. Peerj. https://doi.org/10.7717/peerj.276
8. Wang H et al (2018) RAVEN 2.0: a versatile toolbox for metabolic network reconstruction

and a case study on Streptomyces Coelicolor. PLoS Comput Biol. https://doi.org/10.1371/journal.pcbi.1006541

9. Sandefur CI, Mincheva M, Schnell S (2013) Network representations and methods for the analysis of chemical and biochemical pathways. Mol BioSyst. https://doi.org/10.1039/c3mb70052f
10. Turei D, Korcsmaros T, Saez-Rodriguez J (2016) OmniPath: guidelines and gateway for literature-curated signaling pathway resources. Nat Methods 13(12):966–967. https://doi.org/10.1038/nmeth.4077
11. Seçilmiş D, Hillerton T, Sonnhammer ELL (2022) GRNbenchmark – a web server for benchmarking directed gene regulatory network inference methods. Nucleic Acids Res. https://doi.org/10.1093/nar/gkac377
12. Cai X, Bazerque JA, Giannakis GB (2013) Inference of gene regulatory networks with sparse structural equation models exploiting genetic perturbations. PLoS Comput Biol. https://doi.org/10.1371/journal.pcbi.1003068
13. Gan Y et al (2022) Inferring gene regulatory networks from single-cell transcriptomic data using bidirectional RNN. Front Oncol. https://doi.org/10.3389/fonc.2022.899825
14. Wu J, Zhao X, Lin Z, Shao Z (2016) Large scale gene regulatory network inference with a multi-level strategy. Mol BioSyst. https://doi.org/10.1039/c5mb00560d
15. Weighill D et al (2020) Gene regulatory network inference as relaxed graph matching. https://doi.org/10.1101/2020.06.23.167999
16. Emadi M, Boroujeni FZ, Pirgazi J (2021) Improved fuzzy cognitive maps for gene regulatory networks inference based on time series data. https://doi.org/10.21203/rs.3.rs-770358/v1
17. Zarayeneh N et al (2017) Integration of multi-omics data for integrative gene regulatory network inference. Int J Data Min Bioinform. https://doi.org/10.1504/ijdmb.2017.10008266
18. Boehm JS, Golub TR (2015) An ecosystem of cancer cell line factories to support a cancer dependency map. Nat Rev Genet 16(7):373–374. https://doi.org/10.1038/nrg3967
19. Corsello SM et al (2020) Discovering the anti-cancer potential of non-oncology drugs by systematic viability profiling. Nat Cancer 1(2):235–248. https://doi.org/10.1038/s43018-019-0018-6
20. Meyers RM et al (2017) Computational correction of copy number effect improves specificity of CRISPR-Cas9 essentiality screens in cancer cells. Nat Genet 49(12):1779–1784. https://doi.org/10.1038/ng.3984
21. Tsherniak A et al (2017) Defining a cancer dependency map. Cell 170(3):564–576.e516. https://doi.org/10.1016/j.cell.2017.06.010
22. Yu C et al (2016) High-throughput identification of genotype-specific cancer vulnerabilities in mixtures of barcoded tumor cell lines. Nat Biotechnol 34(4):419–423. https://doi.org/10.1038/nbt.3460
23. Cancer Cell Line Encyclopedia C and Genomics of Drug Sensitivity in Cancer C (2015) Pharmacogenomic agreement between two cancer cell line data sets. Nature 528(7580):84–87. https://doi.org/10.1038/nature15736
24. Barretina J et al (2012) The cancer cell line encyclopedia enables predictive modelling of anticancer drug sensitivity. Nature 483(7391):603–607. https://doi.org/10.1038/nature11003
25. DepMap B (2021) public_21q1. In, 1 edn
26. Marcotte R et al (2016) Functional genomic landscape of human breast cancer drivers, vulnerabilities, and resistance. Cell 164(1–2):293–309. https://doi.org/10.1016/j.cell.2015.11.062
27. McDonald ER et al (2017) Project DRIVE: a compendium of cancer dependencies and synthetic lethal relationships uncovered by large-scale, deep RNAi screening. Cell 170(3):577–592.e510. https://doi.org/10.1016/j.cell.2017.07.005
28. McFarland JM et al (2018) Improved estimation of cancer dependencies from large-scale RNAi screens using model-based normalization and data integration. Nat Commun 9(1). https://doi.org/10.1038/s41467-018-06916-5
29. Science CD (2020) DEMETER2 data. In, 6 edn
30. Dempster JM et al (2019) Extracting biological insights from the project Achilles genome-scale CRISPR screens in cancer cell lines. bioRxiv:720243. https://doi.org/10.1101/720243
31. Ghandi M et al (2019) Next-generation characterization of the cancer cell line encyclopedia. Nature 569(7757):503–508. https://doi.org/10.1038/s41586-019-1186-3
32. Benjamini Y, Hochberg Y (1995) Controlling the false discovery rate: a practical and powerful approach to multiple testing. J R Stat Soc Ser B (Methodolog) 57(1):289–300

33. Team RC (2020) R: a language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria
34. Wickham H, Hester J, Chang W, Bryan J (2022) Devtools: tools to make developing R packages easier. In, R package version 2.4.5 edn
35. Bache SM, Wickham H (2022) Magrittr: a forward-pipe operator for R. In, R package version 2.0.3 edn
36. Wickham H, Vaughan D, Girlich M (2023) Tidyr: tidy messy data. In, R package version 1.3.0 edn
37. Wickham H, Henry L (2023) Purrr: functional programming tools. In, R package version 1.0.2 edn
38. Bengtsson H (2021) A unifying framework for parallel and distributed processing in R using futures. R J 13(2):208–227. https://doi.org/10.32614/RJ-2021-048
39. Vaughan D, Dancho M (2022) Furrr: apply mapping functions in parallel using futures. In, R package version 0.3.1 edn
40. Krantz S (2022) Collapse: advanced and fast data transformation. In
41. Barrett T, Dowle M, Srinivasan A (2023) data.table: Extension of 'data.frame'. In, R package version 1.14.10 edn
42. Wickham H (2016) ggplot2: elegant graphics for data analysis. Springer, New York
43. Slowikowski K (2024) ggrepel: automatically position non-overlapping text labels with 'ggplot2'. In, R package version 0.9.5 edn

Chapter 12

NGP: A Tool to Detect Noncoding RNA-Gene Regulatory Pairs from Transcriptomic Data

Hongjie Ke and Tianzhou Ma

Abstract

Noncoding RNAs (ncRNAs) play key roles in cancer initiation, promotion, and progression via regulating the expression of critical genes. Existing methods performed simple bivariate analysis on each pair of ncRNA and gene separately without considering the complex interactions among ncRNAs and genes. We developed a statistically rigorous and computationally efficient software tool to identify essential ncRNA-gene regulatory pairs from transcriptome-wide ncRNA and gene expression data. Here we provide a practical guidance with real data examples on the use of the tool implemented in the R package "NGP."

Key words Noncoding RNA, Gene regulation, Gene expression, NGP

1 Introduction

A majority (>95%) of the human genome is transcribed into RNAs that do not further encode for proteins called noncoding RNAs (ncRNAs), which include micro RNAs (miRNAs), small interfering RNAs (siRNAs), and long noncoding RNAs (lncRNAs), among others [1]. Though long regarded as the "dark matter" of the genome [2], ncRNAs played critical roles in human malignancies. For example, deregulation of miRNAs and lncRNAs has been linked to all cancer types and impacts major cancer hallmarks [3–8]. ncRNAs can regulate gene expression at both transcriptional and posttranscriptional levels, as mechanisms to affect cancer onset and progression [1, 9]. However, the study of ncRNAs and their target genes and how they regulate gene expression in cancer is still in its infancy.

The advent of high-throughput technology including microarray and RNA sequencing (RNA-seq) has enabled the expression analysis for a large number of ncRNAs and genes over the whole genome simultaneously. Existing methods typically performed simple bivariate analysis on each pair of ncRNA and gene separately,

Alexander Krasnitz and Pascal Belleau (eds.), *Cancer Bioinformatics*, Methods in Molecular Biology, vol. 2932,
https://doi.org/10.1007/978-1-0716-4566-6_12,

which ignores the complex interaction among ncRNAs and genes and is subject to severe multiple testing issue [10, 11]. Jointly analyzing candidate ncRNAs and genes expressed in a condition and investigating how their interactions changed from status to status (e.g., from early stage to late stage cancer) are critical to our understanding of their roles in disease pathogenesis. However, several analytical challenges exist when handling these two sets of expression data together. First, the numbers of ncRNAs and genes in the human genome are both huge (~10k–100k each), resulting in a vast number of candidate ncRNA-gene interactive pairs (~1 billion) to search from. Second, ncRNAs and genes are highly correlated with other ncRNAs and genes, further increasing the computational burden. Third, the high-dimensional data of ncRNA and gene expression are usually featured by being highly skewed, heavy-tailed, and noisy (e.g., due to low expression of ncRNA). Robust methods are needed to mitigate the bias brought by these non-normal data.

To fill the gap, we recently developed a statistically rigorous and computational efficient method to robustly screen and select non-coding RNA regulators of gene expression [12]. Here, we present a bioinformatic tool built on the basis of the method, namely, "NGP," for the detection of *N*oncoding RNA-*G*ene regulatory *P*airs from transcriptome-wide ncRNA and gene expression data.

2 Materials

This protocol requires a computer with R installed. We assume both ncRNA and gene expression data have undergone standard pipeline for microarray (e.g., quality control, background adjustment, and normalization by affy or lumi package in R [13, 14]) or RNA-seq (e.g., alignment and quantification by HISAT + StringTie [15, 16]) and are well annotated following standardized gene nomenclature. The inputs of the tool are the normalized expression values (or normalized counts, e.g., Reads Per Kilobase of transcript per Million mapped reads (RPKM)/Fragments Per Kilobase of transcript per Million mapped reads (FPKM)/Transcripts Per Million (TPM)/Reads Per Million (RPM) for RNA-seq) of annotated ncRNAs and genes for matched samples in a particular condition (e.g., patients with kidney cancer):

(a) Normalized expression values (or normalized count for RNA-seq) of ncRNAs for matched samples in csv or txt files

(b) Normalized expression values (or normalized count for RNA-seq) of genes for matched samples in csv or txt files

The current *NGP* tool only allows the input of normalized expression data from either microarray or RNA-seq. Users can use

the tool to further perform filtering, imputation, and transformation if needed. For RNA-seq, the tool so far only considers the continuous RPKM/FPKM/TPM values but not raw count data. After reading into R, the two input data items will be in data frame/matrix format with rows representing the ncRNAs/genes and columns representing the samples. The columns (i.e., samples) of the two data matrices need to be matched and put in the same order.

3 Methods

Figure 1 shows a workflow of the *NGP* tool. We assume both ncRNA and gene expression data have undergone standard pipeline to generate normalized expression values. The tool provides a function to perform filtering, imputation, and transformation when needed as a preprocessing step before the analysis. The main analytical component of the tool is implemented in two stages: in the first dimension reduction stage, as both ncRNA and gene expression data are of high dimension, the tool runs an edge-wise screening (remove ncRNA-gene pairs) followed by a node-wise screening (remove ncRNA or genes) to reduce the number of ncRNAs, genes, and ncRNA-gene pairs to a more manageable

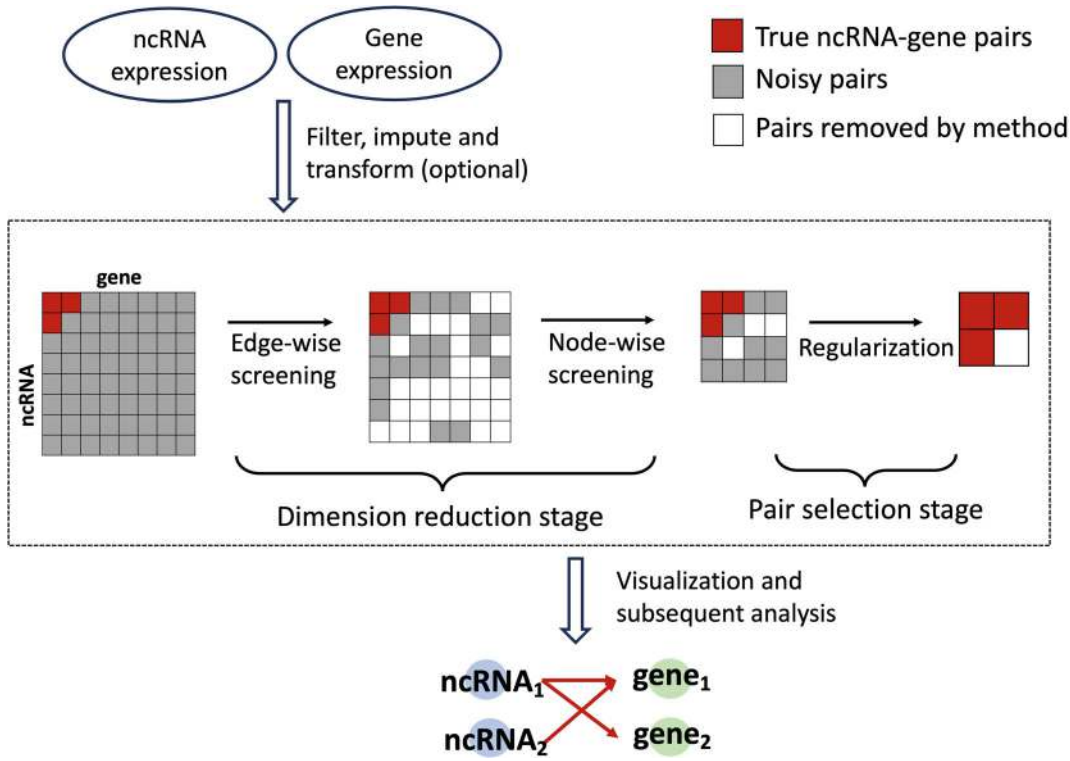


Fig. 1 A workflow of the "NGP" tool. Dashed part is the main analytical component of the tool

scale. In the second pair selection stage, the tool applies a multivariate regularization method to identify the most critical ncRNA-gene pairs from a sparse ncRNA x gene adjacency matrix outputted from the first stage. Our tool can also be coupled with existing visualization software (e.g., Cytoscape [17]) to provide both tabular and graphical outputs for users to visually explore the identified ncRNA-gene regulatory pairs and perform subsequent analysis (e.g., pathway enrichment analysis, validation using external database).

3.1 Install NGP in R

Download *NGP* from its GitHub repository:

```
devtools::install_github('kehongjie/NGP')
```

Dependency packages of *NGP* including impute, rPCor, CCA, CCP, remMAP, and FarmTest will be automatically installed with the package.

3.2 Preprocessing Step (Optional)

"Junk in, junk out". Expression data can be noisy which will lead to biased results if not carefully prepared. Before performing any analysis, the tool employs an optional preprocessing step, following the common practice of analyzing gene expression data [18], by filtering out ncRNAs or genes with low expression (e.g., typical for ncRNAs) and low variance (e.g., housekeeping genes), conducting log2 transformation, and imputing any missing values whenever necessary if users have not done so yet.

Sample code

```
ncrna_data <- preproc(data=ncrna_input, type="ncRNA", platform="RNAseq", filter.mean.cutoff=T, filter.mean.quantile=F, filter.var=F, mean.cutoff=0.3, log=FALSE, impute=FALSE)
gene_data <- preproc(data=gene_input, type="gene", platform="RNAseq", filter.mean.cutoff=T, filter.mean.quantile=F, filter.var=T, mean.cutoff=5, var.quantile=0.5, log=FALSE, impute=FALSE)
```

We provide two options for filtering by mean: cutoff vs. quantile. Considering the relatively lower expression of ncRNAs as compared to genes, we suggest using a lower mean cutoff for ncRNA (e.g., 0.3 for ncRNA vs. 5 for gene). The preprocessing pipeline for ncRNA has not reached a consensus yet, so we follow the common practice in gene expression to process ncRNA, but users can apply other existing ncRNA specific processing pipelines to process the ncRNA expression data and treat it as input for our tool. Log2 transformation and imputation are optional if already performed in previous steps.

3.3 Edge-wise Screening of ncRNA-Gene Pairs

Both ncRNA and gene expression data are of extremely high dimension (10–100k each, with up to 1 billion potential edges to search from), so we need to reduce the dimension before selecting important pairs. In the dimension reduction stage, we first perform an edge-wise screening. The main method used here follows from our recently published paper [12], and users may refer to the paper for technical details. In short, we use the robust partial correlation based statistics as a screening utility and run an iterative approach to remove ncRNA-gene pairs/edges with the statistics not passing a threshold defined by the parameter α_e (*see* Fig. 2). The expected output from this step is a highly sparse ncRNA x gene adjacency matrix ("1" indicating the corresponding ncRNA-gene pair is kept, "0" indicating the pair is removed) with much fewer number of pairs, ncRNAs, and genes (when all pairs of an ncRNA or a gene are removed) kept than the original matrix. The optimal choice of α_e will be determined by stability or pseudo-F-score based approaches introduced next (*see* Subheading 3.5).

Sample code

```
fit_edge <- screen.edge(X=ncrna_data, Y=gene_data, alpha=1e-
5, X.thres=0.5, Y.thres=0.5, C=0.5)
```

In the above function, "alpha" is the key threshold parameter users need to tune using our recommended approaches to achieve the best performance. Other than that, we do not suggest users to change the default values for options determining the neighbors in conditional sets (X.thres, Y.thres) and the robustification related

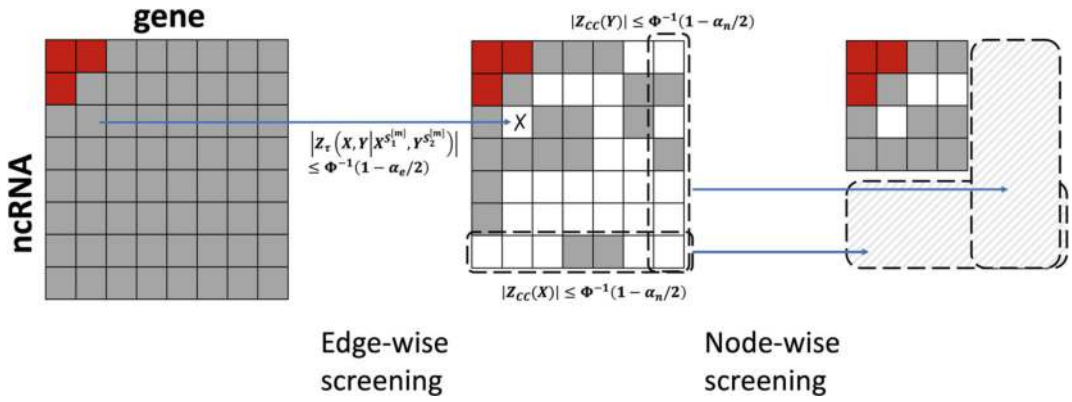


Fig. 2 Demonstration of screening methods in dimension reduction stage. Two screening utilities, robust partial correlation and marginal canonical correlation, are used in edge-wise and node-wise screening, respectively. In edge-wise screening, when the robust partial correlation based statistics of a ncRNA(X)-gene(Y) pair $| Z_\tau\left(X, Y|X^{S_1^{[m]}}, Y^{S_2^{[m]}}\right) |$ does not pass the threshold $\Phi^{-1}(1-\alpha_e/2)$ at any iteration m, the pair will be removed. In node-wise screening, when the marginal canonical correlation based statistics between a node (either ncRNA(X) or gene(Y)) and its remaining edges $| Z_{CC}(X) |$ or $| Z_{CC}(Y) |$ does not pass the threshold $\Phi^{-1}(1-\alpha_n/2)$, the node will be removed

parameter (C) in the calculation of robust partial correlation, unless they fully understand the method. The output of this function will be an ncRNA x gene (p x q where p is the number of ncRNAs and q is the number of genes) adjacency matrix, with each entry indicating whether the corresponding ncRNA-gene pair is kept (1) or not (0).

3.4 Node-Wise Screening of ncRNA and Gene Nodes

The ncRNA x gene matrix after edge-wise screening tends to have highly sparse rows (ncRNAs) and columns (genes). We further apply a node-wise screening by using marginal canonical correlation (CC) of a node with all its remaining neighbors as a screening utility [19] and remove ncRNA or gene nodes having the statistics not passing a threshold defined by the parameter α_n (*see* Fig. 2).

Sample code

```
fit_node <- screen.node(X=ncrna_data, Y=gene_data, edge.ind=fit_edge, alpha=1e-6)
```

Edge.ind is an ncRNA x gene adjacency matrix with each entry indicating whether the pair is kept from the previous step (1) or not (0). The above function will calculate the marginal CC for all nodes with their remaining neighbors and remove nodes when relatively low CC. Alpha is the threshold parameter in node-wise screening (less sensitive than the one for edge-wise screening; *see* Subheading 3.5 for recommended value). The node-wise screening may have computational bottlenecks with an ultrahigh-dimensional dense matrix, so we suggest node-wise screening to be implemented after edge-wise screening. Like the previous step, the output of this function will be an ncRNA × gene ($p \times q$: p is number of ncRNAs, q the number of genes) adjacency matrix, with each entry indicating whether the corresponding ncRNA-gene pair is kept (1) or not (0) after node-wise screening step.

3.5 Selecting Optimal Threshold Parameter

One key tuning parameter of the tool is the threshold parameter α_e (α for short from here on) in the edge-wise screening step. On one hand, a too stringent α will have the risk of losing important signals from the screening step; on the other hand, an overly conservative α will reduce the screening power without achieving the dimension reduction purpose. In *NGP* tool, following from [12], we propose two procedures to select the optimal α. The first procedure is based on stability selection to make sure an optimal α can remove as many pairs as possible while keeping the selection frequency of top pairs above a cutoff with false-discovery rate (FDR) controlled.

Sample code

```
p <- ncol(ncrna_data) # number of ncRNAs
q <- ncol(gene_data) # number of genes
```

```
alpha <- screen.tune(X=ncrna_data, Y=gene_data, B=100,
D=100, eta=min(p,q), method="stability")
```

Eta is the number of top pairs for which selection frequency will be calculated. B and D are the number of subsampling and permutations to be performed to calculate selection frequency and get its FDR cutoff. We do not suggest users to change the default values for these options unless they fully understand the method and the algorithm. Figure 3 shows an example on how to select the optimal threshold parameter α using a stability based approach. At $\alpha = 1e-6$, the selection frequency of top pairs bypasses the FDR cutoff; thus, the threshold is chosen too stringent. $\alpha = 1e-5$ is an optimal threshold where the selection frequency of top pairs does not bypass the FDR cutoff but also achieves the screening purpose (i.e., screen out as many noisy pairs as one can).

The stability selection algorithm, however, can be computationally heavy and becomes infeasible in some real data with ultra-high dimension. Alternatively, we propose a much faster and relatively accurate procedure using a pseudo-F-score (like F-score = 2*precision*sensitivity/{precision + sensitivity}). A pseudo-F-score is defined using the same formula but from a pseudo sensitivity (true ncRNA-gene pairs defined as those pairs with largest marginal correlations) and a pseudo precision as we do not know the ground truth in real data to balance between effective dimension reduction (i.e. maintaining high precision) and high sensitivity in the remaining pairs.

Sample code

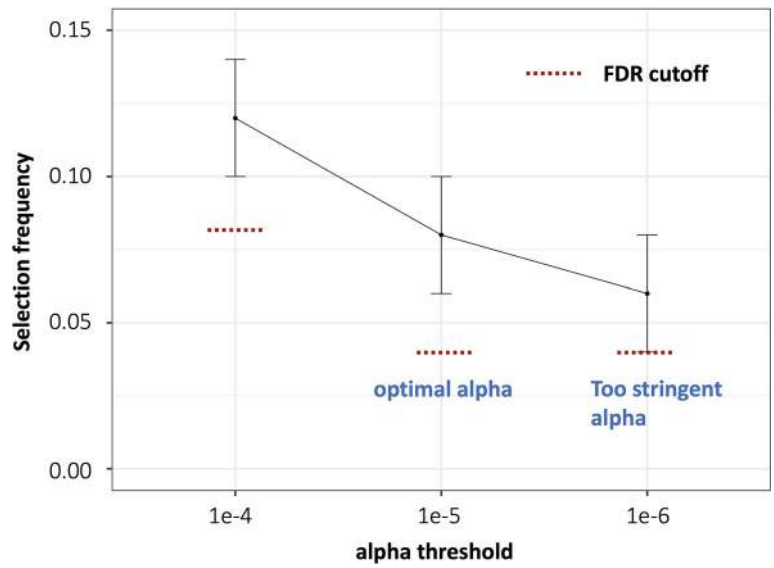


Fig. 3 Selecting optimal threshold parameter α using a stability based approach

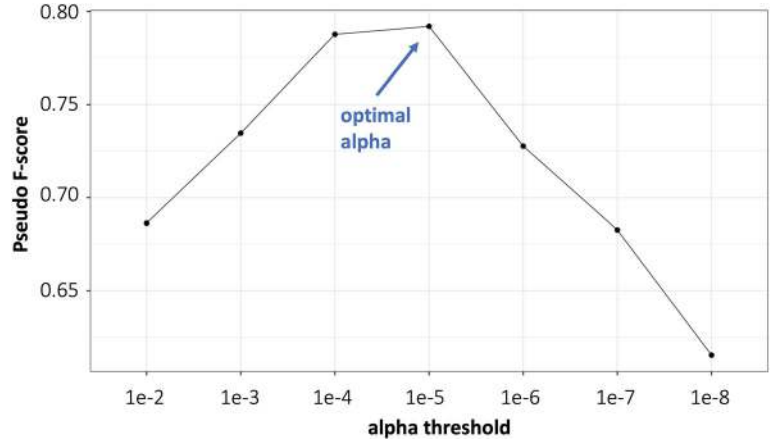


Fig. 4 Selecting optimal threshold parameter α using pseudo-*F*-score

```
alpha <- screen.tune(X=ncrna_data, Y=gene_data, eta=min(p, q), method="pseudo")
```

Eta is the number of top pairs from which the pseudo-F-score will be calculated. Figure 4 shows an example on how to select the optimal threshold parameter α using a pseudo-F-score. $\alpha = 1e-5$ is an optimal threshold that has the largest pseudo-F-score value here.

The choice of the threshold parameter α_n in the node-wise screening step is relatively stable based on our exploration of different real data examples, so we do not further suggest any approaches to select its optimal value. We generally suggest a slightly more stringent α_n (e.g., $\alpha_n = \alpha/10$) to make sure highly sparse nodes can be removed.

3.6 Multivariate Regularization to Identify Final ncRNA-Gene Regulatory Pairs

After screening (or dimension reduction stage), the dimensions of both ncRNAs and genes are reduced to a more manageable scale (e.g., comparable to sample size), and the matrix of remaining ncRNA-gene pairs is highly sparse. In the final pair selection stage, the *NGP* tool performs a multivariate regularization method to identify the most probable ncRNA-gene regulatory pairs from the highly sparse matrix of remaining ncRNA-gene edge pairs [20].

Sample code

```
fit_reg <- multi.reg(X=ncrna_data, Y=gene_data, edge.ind=fit_node, lambda1=seq(1,100,10), lambda2=seq(1,100,10))
```

Edge.ind is an ncRNA x gene adjacency matrix with each entry indicating whether the pair is kept from the previous steps (1) or not (0). Lambda1 and lambda 2 are the sequence of tuning parameters ("seq(1,100,10)": start from 1, end at 100 with step size of

10) to be used in the regularization step, and the method will automatically use cross-validation to select the optimal lambdas from the specified sequences. We generally do not suggest users to change the default values to ensure accuracy. However, to avoid excessive computation in irrelevant ranges, users can first run coarser search using a larger step size (e.g., 20 or 30) in a wider range and then fine search for the optimal lambda with smaller step size within a narrower range.

3.7 NGP *Output and Subsequent Analysis*

The major output of the *NGP* tool includes a matrix of coefficient estimate for all remaining ncRNA-gene pairs and a list of two columns of corresponding ncRNA and gene names for those pairs. The two-column output will be treated as an input (ncRNA column as the source node, gene column as the target node) for network visualization platform "CytoScape" [17] to investigate the ncRNA-gene regulation pattern in a network graph.

Sample code

```
fit_reg$cyto_input
```

The identified ncRNA-gene pairs and remaining ncRNA and gene nodes can be further validated using existing database on ncRNAs, genes, and ncRNA-gene interactions specific to a condition or a disease [21–23]. These databases are typically experimentally validated or sequence based, thus providing further evidence and additional biological information to the interactions identified from expression data by *NGP* tool. In addition, we also suggest users to perform pathway enrichment analysis on the pool of genes regulated by ncRNAs for more biological insight of the ncRNA regulatory path in the disease.

4 Examples

We use ncRNA and gene expression data from The Cancer Genome Atlas Program (TCGA) cohort to demonstrate two examples (one for lncRNA-gene regulation, the other for miRNA-gene regulation) of applying the *NGP* tool to identify critical ncRNA-gene regulatory pairs in cancer study.

4.1 *LncRNA Regulation of Gene Expression in Kidney Renal Papillary Cell Carcinoma (KIRP)*

LncRNAs are essential regulators of genes in major pathways of cell growth, proliferation, differentiation, and survival and are critical to the tumor formation, progression, and pathogenesis of kidney cancer [24, 25]. KIRP accounts for 10–15% of all renal cell carcinoma and has a poor prognosis [26]. In this example, we retrieve lncRNA (in RPM) and gene expression data (in RPKM; both measured by RNA-seq) of $N = 198$ matched KIRP samples in TCGA from The Atlas of Non-coding RNAs in Cancer [27] and

LinkedOmics [28], respectively, and use our tool to identify critical lncRNAs, genes, and lncRNA-gene regulatory pairs in KIRP.

In the first step, we carefully preprocessed the data and filtered out features with low expression (lncRNAs with mean RPM ≤ 0.3 and genes with mean RPKM ≤ 5) following the general guideline [27, 29]. We also filtered out potential housekeeping genes and only kept the most variant 50% of genes. After preprocessing, 2170 lncRNAs and 6704 genes were left with a total of 14,547,680 possible pairs.

We then applied the edge-wise screening to reduce dimension, and only 3054 pairs were left after screening, generating a highly sparse lncRNA x gene matrix. We further applied the node-wise screening and removed those lncRNAs and genes with very few edges left (mostly with one or two edges per node, as the number of pairs removed are about the same as the number of nodes removed after node-wise screening). Lastly, we applied the multivariate regulation to select the final set of 877 ncRNA-gene regulatory pairs from the interactions of 298 lncRNAs and 781 genes. The final pool was highly enriched with lncRNAs related to kidney cancer based on EVLncRNA database [22] (*see* Table 1; 18 out of 32 with Fisher's exact test p-value$<1e{-}4$). We plotted one example of lncRNA-gene interaction network from these 877 pairs in Cytoscape (Fig. 5). Both the lncRNA "lnc-IRX3–80" and the gene "IRX5" were Iroquois transcription factors found to be related to kidney development and tumorigenesis [30, 31]. Their interaction we identified here has also been validated in the LncTarD database [32], a comprehensive lncRNA-target regulation database, which may help reveal the underlying regulatory mechanism in KIRP and have potential diagnostic and therapeutic values.

Table 1
Results after each step of the *NGP* tool for the TCGA-KIRP lncRNA-gene regulation example

Step	Number of lncRNAs left	Number of genes left	Number of lncRNA-gene pairs left
After preprocessing	2170 (32)	6704	14,547,680
After edge-wise screening	1336 (23)	2251	3054
After node-wise screening	1153 (21)	1737	2471
After regularization	298 (18)	781	877

Numbers inside parentheses indicate the numbers of lncRNAs that are shown to be related to kidney cancer according to EVLncRNAs database

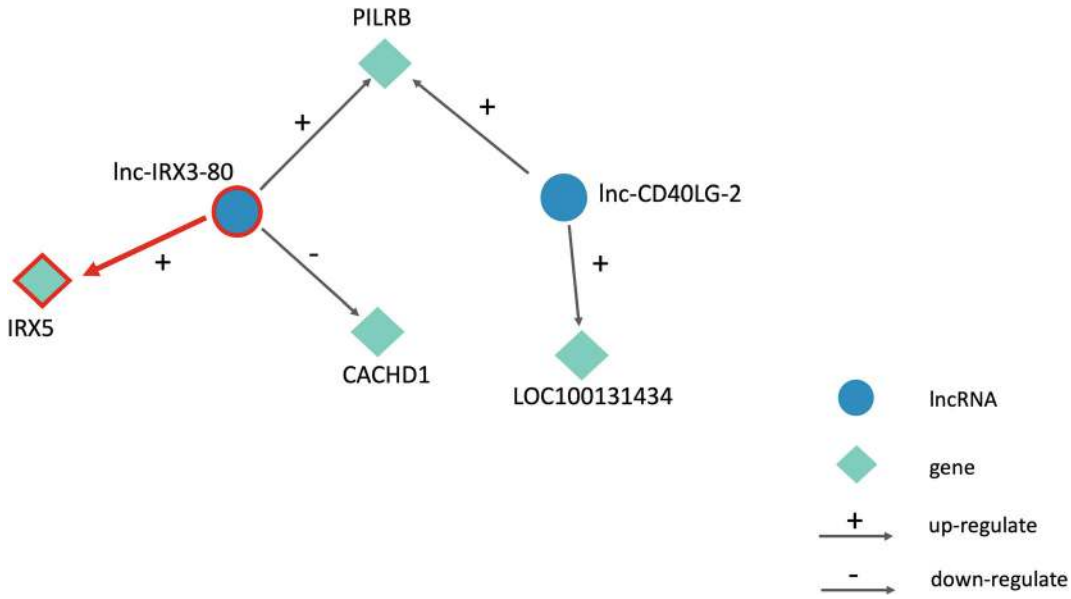


Fig. 5 An example of a network of lncRNA-gene regulatory pairs identified in TCGA KIRP by our method. Red highlighted indicates the lncRNAs, genes, or the lncRNA-gene pairs have been validated in large lncRNA/gene databases (e.g., EVLncRNA, LncTarD, Oncogene database)

4.2 miRNA Regulation of Gene Expression in PRostate ADenocarcinoma (PRAD)

miRNAs are critical noncoding RNAs that play indispensable roles in regulating gene expression and have been found as key factors and potentially serve as clinical tools for diagnosis, prognosis, and therapy in prostate cancer [33–35]. In the second example, we retrieve miRNA (in RPM) and gene expression data (in RPKM) of $N = 493$ matched PRAD samples in TCGA from LinkedOmics [28] and use our tool to identify critical miRNAs, genes, and miRNA-gene regulatory pairs in PRAD.

We first preprocessed the data and filtered out low-expressed genes with mean RPKM ≤ 5 and only kept the most variant 50% of genes. After preprocessing, 765 miRNAs and 6715 genes were left with a total of 5,136,975 possible pairs.

We then applied the edge-wise screening to reduce dimension, and only 1396 pairs were left after screening, generating a highly sparse miRNA x gene matrix. We further applied the node-wise screening and removed those miRNAs and genes with very few edges left (mostly with one or two edges per node). Lastly, we applied the multivariate regulation to select the final set of 618 ncRNA-gene regulatory pairs from the interactions of 147 miRNAs and 547 genes. These 147 miRNAs were highly enriched with miRNAs related to prostate cancer based on miR-Cancer database [21] (*see* Table 2; 33 out of 79 with Fisher's exact test p-value$<1e-4$). We plotted one example of miRNA-gene interaction network from these 618 pairs in Cytoscape (Fig. 6). The

Table 2
Results after each step of the *NGP* tool for the TCGA-PRAD miRNA-gene regulation example

Step	Number of miRNAs left	Number of genes left	Number of miRNA-gene pairs left
After preprocessing	765 (79)	6715	5,136,975
After edge-wise screening	381 (50)	1132	1396
After node-wise screening	294 (46)	729	962
After regularization	147 (33)	547	618

Numbers inside parentheses indicate the numbers of miRNAs that are shown to be related to kidney cancer according to miRCancer database

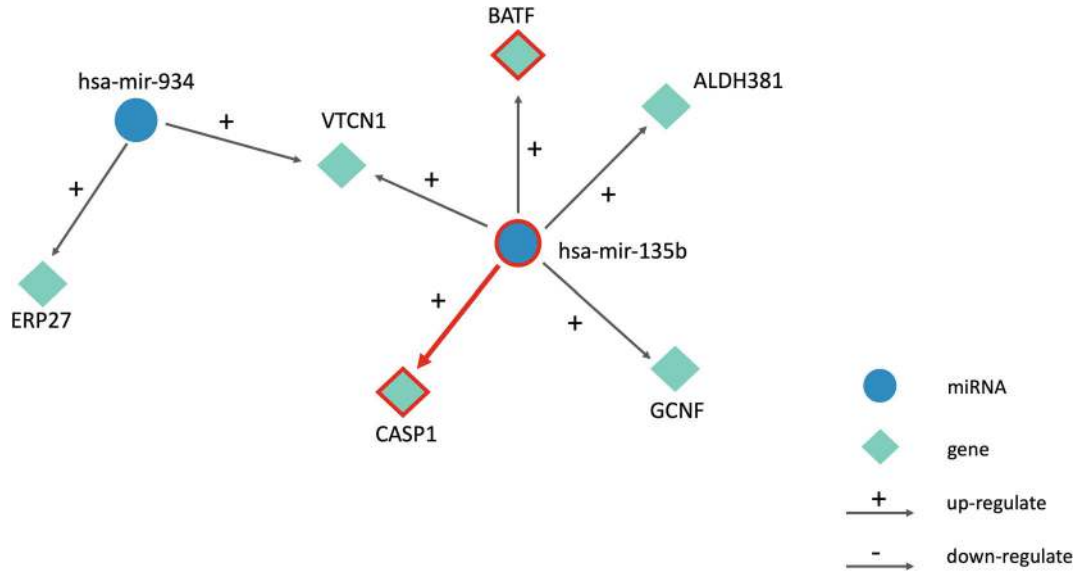


Fig. 6 An example of a network of miRNA-gene regulatory pairs identified in TCGA PRAD by our method. Red highlighted indicates the miRNAs, genes, or the miRNA-gene pairs have been validated in large miRNA/gene databases (e.g., miRCancer, miRDB, Oncogene database)

miRNA "hsa-mir-135b" and the genes "BATF" and "CASP1" were found to be related to prostate cancer (BATF and CASP1 are both potential oncogenes) [36–38]. Their interactions we identified here have also been validated in the miRDB database [39], revealing the potential miRNA→gene regulatory pathways in PRAD.

5 Notes

1. Screening implemented in the *NGP* tool is different from filtering by mean or variance as usually done in bioinformatics. Stemmed from the concept of sure screening in statistics, screening is a theoretically justified step to reduce the dimension for the analysis while keeping most of the true signals in the pool after screening, a theoretical property known as sure screening property [40]. In our context, we have a theoretical guarantee that most of the true ncRNA-gene regulatory pairs should remain in the pool after screening.
2. As the *NGP* tool first applies screening methods to reduce the dimension, it is computationally efficient and runs significantly fast for high-dimensional ncRNA and gene expression data. However, the computational cost can depend on the detailed implementation of each step (*see* our paper [12] for a benchmark of computational time of the screening step using simulations), e.g., the threshold parameter used in screening, the range of tuning parameter values to search from in regularization. For example, setting a more stringent screening threshold may speed up the procedure by greatly reducing the dimension but may also have the danger of losing important signals. We recommend users to follow instructions in the above protocol for the most efficient and accurate implementation of the tool.
3. To further reduce the computational cost, we highly recommended users to filter out ncRNAs or genes of less interest (low means or low variance) and focus only on the ncRNAs or genes of biological interest. If some differential expression (DE) analysis for ncRNAs and genes can be performed between, e.g., cancer and normal tissues, one can also prioritize analyzing the top DE ncRNAs and genes most related to disease development.
4. We assume both ncRNA and gene expression data were well annotated using standard nomenclature before the preprocessing step. If annotation was not completed yet, we recommend users apply the functions in existing R packages (e.g., "*preproc*" function in the "metaOmic" package [18]) and use existing annotation databases available in R to annotate before proceeding to the next steps.
5. The tool so far only considers continuously valued noncoding and gene expression data generated by microarray or RNA-seq. As the technology advances, new types of expression data (e.g., those generated by single cell RNA-seq) will emerge, and the tool will be gradually improved to accommodate the features of these newly emerging data types.

6. In its current form, the *NGP* tool relies on "Cytoscape" to help generate graphical and network output. The tool will be coupled with existing packages in R (e.g., igraph) to generate a graphical user interface (e.g., by R Shiny) with interactive visualization in future development.
7. The tool identifies critical ncRNA-gene regulatory pairs. On one hand, we recommend users to further perform downstream pathway analysis on the identified genes and post hoc validation using external databases (e.g., EVLncRNA, LncTarD, miRCancer); on the other hand, we do want to point out that as the research of noncoding RNA and its regulation of gene expression is still in an early stage, our tool might help identify new ncRNA-gene links underlying important ncRNA regulatory mechanism that are not yet available in existing database but worth further exploration.
8. The tool gives a static snapshot of all potential regulatory ncRNA-gene pairs in a condition or cancer type (e.g., in KIRP) based on expression data. As more clinical data become available, we will further consider how these paired links will change as the disease progresses in different stages/grades and whether any of these links have potential prognostic values (i.e., predictive of survival) in cancer in future development. In addition, the tool currently selects edge by edge without considering the overall regulatory pattern of ncRNAs and genes (e.g., in network) nor the flow of biological information. Bayesian network method to identify the overall causal pattern will be an important plus to the tool in future work.

Acknowledgments

This work was supported by the University of Maryland Grand Challenge grant, University of Maryland MPower Brain Health and Human Performance seed grant, University of Maryland Department of Epidemiology and Biostatistics Pilot Award and National Institutes of Health under award number 1K01DA059603-01A1.

References

1. Yan H, Bu P (2021) Non-coding RNA in cancer. Essays Biochem 65(4). https://doi.org/10.1042/EBC20200032
2. Patil VS, Zhou R, Rana TM (2014) Gene regulation by non-coding RNAs. Crit Rev Biochem Mol Biol 49(1). https://doi.org/10.3109/10409238.2013.844092
3. Schmitt AM, Chang HY (2016) Long noncoding RNAs in cancer pathways. Cancer Cell 29(4). https://doi.org/10.1016/j.ccell.2016.03.010
4. Lenkala D, LaCroix B, Gamazon ER, Geeleher P, Im HK, Huang RS (2014) The impact of microRNA expression on cellular

proliferation. Hum Genet 133(7). https://doi.org/10.1007/s00439-014-1434-4
5. Calin GA, Croce CM (2006) MicroRNA-cancer connection: the beginning of a new tale. Cancer Res 66(15). https://doi.org/10.1158/0008-5472.CAN-06-0800
6. Ivey KN, Srivastava D (2015) microRNAs as developmental regulators. Cold Spring Harb Perspect Biol 7(7). https://doi.org/10.1101/cshperspect.a008144
7. Winkle M, El-Daly SM, Fabbri M, Calin GA (2021) Noncoding RNA therapeutics – challenges and potential solutions. Nat Rev Drug Discov 20(8). https://doi.org/10.1038/s41573-021-00219-z
8. Gutschner T, Diederichs S (2012) The hallmarks of cancer: a long non-coding RNA point of view. RNA Biol 9(6). https://doi.org/10.4161/rna.20481
9. Grillone K, Riillo C, Scionti F, Rocca R, Tradigo G, Guzzi PH, Alcaro S, Di Martino MT, Tagliaferri P, Tassone P (2020) Non-coding RNAs in cancer: platforms and strategies for investigating the genomic "dark matter". J Exp Clin Cancer Res 39(1). https://doi.org/10.1186/s13046-020-01622-x
10. Zhang J, Le TD, Liu L, Li J (2019) Inferring and analyzing module-specific lncRNA-mRNA causal regulatory networks in human cancer. Brief Bioinform 20(4). https://doi.org/10.1093/bib/bby008
11. Madhumita M, Paul S (2022) A review on methods for predicting miRNA-mRNA regulatory modules. J Integr Bioinform 19(3). https://doi.org/10.1515/jib-2020-0048
12. Ke H, Ren Z, Qi J, Chen S, Tseng GC, Ye Z, Ma T (2022) High-dimension to high-dimension screening for detecting genome-wide epigenetic and noncoding RNA regulators of gene expression. Bioinformatics 38(17): 4078–4087. https://doi.org/10.1093/bioinformatics/btac518
13. Gautier L, Cope L, Bolstad BM, Irizarry RA (2004) affy—analysis of Affymetrix GeneChip data at the probe level. Bioinformatics (Oxford, England) 20(3). https://doi.org/10.1093/bioinformatics/btg405
14. Du P, Kibbe WA, Lin SM (2008) lumi: a pipeline for processing Illumina microarray. Bioinformatics (Oxford, England) 24(13). https://doi.org/10.1093/bioinformatics/btn224
15. Pertea M, Kim D, Pertea GM, Leek JT, Salzberg SL (2016) Transcript-level expression analysis of RNA-seq experiments with HISAT, StringTie and Ballgown. Nat Protoc 11(9). https://doi.org/10.1038/nprot.2016.095
16. Sahraeian SME, Mohiyuddin M, Sebra R, Tilgner H, Afshar PT, Au KF, Bani Asadi N et al (2017) Gaining comprehensive biological insight into the transcriptome by performing a broad-spectrum RNA-seq analysis. Nat Commun 8(1). https://doi.org/10.1038/s41467-017-00050-4
17. Shannon P, Markiel A, Ozier O, Baliga NS, Wang JT, Ramage D, Amin N, Schwikowski B, Ideker T (2003) Cytoscape: a software environment for integrated models of biomolecular interaction networks. Genome Res 13(11). https://doi.org/10.1101/gr.1239303
18. Ma T, Huo Z, Kuo A, Zhu L, Fang Z, Zeng X, Lin CW et al (2019) MetaOmics: analysis pipeline and browser-based software suite for transcriptomic meta-analysis. Bioinformatics (Oxford, England) 35(9). https://doi.org/10.1093/bioinformatics/bty825
19. He D, Zhou Y, Zou H (2021) On sure screening with multiple responses. Stat Sin 31(4): 1749–1777
20. Peng J, Zhu J, Bergamaschi A, Han W, Noh DY, Pollack JR, Wang P (2010) Regularized multivariate regression for identifying master predictors with application to integrative genomics study of breast cancer. Ann Appl Stat 4(1). https://doi.org/10.1214/09-AOAS271SUPP
21. Xie B, Ding Q, Han H, Wu D (2013) miRCancer: a microRNA-cancer association database constructed by text mining on literature. Bioinformatics (Oxford, England) 29(5). https://doi.org/10.1093/bioinformatics/btt014
22. Zhou B, Zhao H, Yu J, Guo C, Dou X, Song F, Hu G et al (2018) EVLncRNAs: a manually curated database for long non-coding RNAs validated by low-throughput experiments. Nucleic Acids Res 46(D1). https://doi.org/10.1093/nar/gkx677
23. Cheng L, Wang P, Tian R, Wang S, Guo Q, Luo M, Zhou W, Liu G, Jiang H, Jiang Q (2019) LncRNA2Target v2.0: a comprehensive database for target genes of lncRNAs in human and mouse. Nucleic Acids Res 47 (D1). https://doi.org/10.1093/nar/gky1051
24. Martens-Uzunova ES, Böttcher R, Croce CM, Jenster G, Visakorpi T, Calin GA (2014) Long noncoding RNA in prostate, bladder, and kidney cancer. Eur Urol 65(6). https://doi.org/10.1016/j.eururo.2013.12.003
25. Zhou S, Wang J, Zhang Z (2014) An emerging understanding of long noncoding RNAs in kidney cancer. J Cancer Res Clin Oncol 140(12). https://doi.org/10.1007/s00432-014-1699-y

26. Lan H, Zeng J, Chen G, Huang H (2017) Survival prediction of kidney renal papillary cell carcinoma by comprehensive LncRNA characterization. Oncotarget 8(67). https://doi.org/10.18632/oncotarget.22732
27. Li J, Han L, Roebuck P, Diao L, Liu L, Yuan Y, Weinstein JN, Liang H (2015) TANRIC: an interactive open platform to explore the function of lncRNAs in cancer. Cancer Res 75(18). https://doi.org/10.1158/0008-5472.CAN-15-0273
28. Vasaikar SV, Straub P, Wang J, Zhang B (2018) LinkedOmics: analyzing multi-omics data within and across 32 cancer types. Nucleic Acids Res 46(D1). https://doi.org/10.1093/nar/gkx1090
29. Ricketts CJ, De Cubas AA, Fan H, Smith CC, Lang M, Reznik E, Bowlby R et al (2018) The cancer genome atlas comprehensive molecular characterization of renal cell carcinoma. Cell Rep 23(1). https://doi.org/10.1016/j.celrep.2018.03.075
30. Marra AN, Wingert RA (2014) Roles of iroquois transcription factors in kidney development. Cell Dev Biol 3(1). https://doi.org/10.4172/2168-9296.1000131
31. Holmquist Mengelbier L, Lindell-Munther S, Yasui H, Jansson C, Esfandyari J, Karlsson J, Lau K et al (2019) The Iroquois homeobox proteins IRX3 and IRX5 have distinct roles in Wilms tumour development and human nephrogenesis. J Pathol 247(1). https://doi.org/10.1002/path.5171
32. Zhao H, Yin X, Xu H, Liu K, Liu W, Wang L, Zhang C et al (2023) LncTarD 2.0: an updated comprehensive database for experimentally-supported functional lncRNA-target regulations in human diseases. Nucleic Acids Res 51(D1). https://doi.org/10.1093/nar/gkac984
33. Ghamlouche F, Yehya A, Zeid Y, Fakhereddine H, Fawaz J, Liu YN, Al-Sayegh M, Abou-Kheir W (2023) MicroRNAs as clinical tools for diagnosis, prognosis, and therapy in prostate cancer. Transl Oncol 28. https://doi.org/10.1016/j.tranon.2022.101613
34. Rana S, Valbuena GN, Curry E, Bevan CL, Keun HC (2022) MicroRNAs as biomarkers for prostate cancer prognosis: a systematic review and a systematic reanalysis of public data. Br J Cancer 126(3). https://doi.org/10.1038/s41416-021-01677-3
35. Schitcu VH, Raduly L, Nutu A, Zanoaga O, Ciocan C, Munteanu VC, Cojocneanu R et al (2022) MicroRNA dysregulation in prostate cancer. Pharmgenomics Pers Med 15. https://doi.org/10.2147/PGPM.S348565
36. Olivan M, Garcia M, Suárez L, Guiu M, Gros L, Méndez O, Rigau M et al (2021) Loss of microRNA-135b enhances bone metastasis in prostate cancer and predicts aggressiveness in human prostate samples. Cancers 13(24). https://doi.org/10.3390/cancers13246202
37. Winter RN, Kramer A, Borkowski A, Kyprianou N (2001) Loss of caspase-1 and caspase-3 protein expression in human prostate cancer. Cancer Res 61(3):1227
38. Siltanen S, Fischer D, Rantapero T, Laitinen V, Mpindi JP, Kallioniemi O, Wahlfors T, Schleutker J (2013) ARLTS1 and prostate cancer risk—analysis of expression and regulation. PLoS One 8(8). https://doi.org/10.1371/journal.pone.0072040
39. Chen Y, Wang X (2020) miRDB: an online database for prediction of functional microRNA targets. Nucleic Acids Res 48(D1). https://doi.org/10.1093/nar/gkz757
40. Fan J, Lv J (2008) Sure independence screening for ultrahigh dimensional feature space. J R Stat Soc Series B Stat Methodol 70(5): 849–911. https://doi.org/10.1111/j.1467-9868.2008.00674.x

Chapter 13

MODIG: An Attention Mechanism-Based Approach to Cancer Driver Gene Identification

Wenyi Zhao and Zhan Zhou

Abstract

Identifying genes that play a causal role in carcinogenesis remains one of the major challenges in cancer biology. With the accumulation of high-throughput multi-omics data over decades, it has become a great challenge to effectively integrate these data into the identification of cancer driver genes. Here, we propose MODIG, a graph attention network (GAT)-based framework, to identify cancer driver genes by combining multi-omics pan-cancer data (mutations, copy number variants, gene expression, and methylation levels) with multidimensional gene networks. Among them, the multidimensional gene network is constructed by using genes as nodes and five types of gene associations (protein-protein interaction, gene sequence similarity, KEGG pathway co-occurrence, gene co-expression patterns, and gene ontology terms) as multiplex edges. We apply a GAT encoder to model within-dimension interactions to generate a gene representation for each dimension based on this graph, introduce a joint learning module to fuse multiple dimension-specific representations to generate general gene representations, and use the obtained gene representation to perform a semi-supervised driver gene identification task. The MODIG program is available at https://github.com/zjupgx/modig. The code and data are also available on Zenodo, at https://doi.org/10.5281/zenodo.7057241.

Key words Driver gene, Multi-omics data, Gene network, Protein-protein interaction, Graph attention network, Attention mechanism

1 Introduction

It is widely accepted that cancer progression is due to the accumulation of mutations in driver genes that confer a selective growth advantage to cells [1–4]. As a key issue in cancer genomics, identifying genes that play a causal role in carcinogenesis can help to better understand the molecular mechanisms of cancer development, facilitate the discovery of drug targets and biomarkers, and guide the development of precise therapeutic approaches. Over the past decades, several large-scale cancer genomics projects, such as The Cancer Genome Atlas (TCGA) [5] and the International

Alexander Krasnitz and Pascal Belleau (eds.), *Cancer Bioinformatics*, Methods in Molecular Biology, vol. 2932,
https://doi.org/10.1007/978-1-0716-4566-6_13,

Cancer Genome Consortium (ICGC) [6], have accumulated a large amount of genomics, epigenomics, transcriptomics, and proteomics data from thousands of cancer patients.

Benefitting from these omics data, many computational tools have been developed to identify cancer driver genes, yet most of them focus only on genomics data [7–11]. Several studies have found that some cancer driver genes are not altered at their DNA sequence level but dysregulated through various cellular mechanisms [1, 12, 13], suggesting that diverse omics data are likely to shape gene function at different biological scales and that effective integration of such information is valuable for identifying the potential driver and passenger genes. Thus, there is an increasing need to develop a framework that can harness and integrate complementary information among multi-omics data for downstream cancer driver gene prediction tasks.

Biological networks are abstract representations of biological systems as graphs, where genes are used as nodes, gene associations as edges, and omics features as node attributes. As a research hotspot in recent years, graph neural networks (GNN) are deep learning models developed specifically for graphs, which apply to high-dimensional and complex biological data and are potentially suitable integration frameworks. Recently, several studies have proposed cancer driver gene prediction methods integrating multi-omics features based on graph deep learning. EMOGI is an interpretable machine learning method based on graph convolutional networks (GCN) that combine genomics, epigenomics, and transcriptomics data as gene features with protein-protein interaction (PPI) networks to learn more abstract gene features [14]. MTGCN is a GCN-based multitask learning framework that simultaneously optimizes the node prediction and the link prediction task during the learning of node embedding features [15]. These existing methods are designed based on PPI networks.

However, considering the limitations of PPI networks (e.g., incomplete interaction profiles and the existence of research bias) and the fact that the performance of graph deep learning models relies heavily on the reliability of graph structure, we introduce heterogeneous information of multiple types of gene associations (gene similarity, gene co-expression patterns, etc.) into the construction of multidimensional gene networks, in which multi-omics features are used as node features, and develop MODIG, a graph attention network-based (GAT-based) model for efficient integration of cancer multi-omics data to facilitate the prediction of cancer driver genes [16].

2 Materials and Methods

MODIG is a GAT-based cancer driver gene identification method that generates gene representation by integrating multi-omics data and multiple gene associations; it not only utilizes the labels of labeled nodes in a multidimensional gene network but also the topological features of unlabeled nodes [17]. The steps involved in MODIG are as follows:

1. *Generation of omics feature matrix*: Calculate mutation rate, differential DNA methylation level, and differential expression rate for each gene by using multi-omics data.
2. *Generation of gene association profiles*: Measure gene associations based on gene co-expression pattern, gene sequence similarity, gene pathway co-occurrence, gene semantic similarity, and PPI.
3. *Construction of multidimensional gene network*: Use these gene association profiles as multiple edges and multi-omics features as node attributes.
4. *Multi-omics and multidimensional graph attention network*: MODIG contains three modules: a multidimensional GAT encoder, a joint learning module, and a multilayer perceptron (MLP) classifier for learning knowledge from multidimensional graphs to help in cancer driver gene prediction.

2.1 Generation of Omics Feature Matrix

Gene features containing the mutation rate, differential DNA methylation level, and differential expression rate are calculated by using the cancer genomics (somatic mutations and copy number variants), epigenomics (DNA methylations), and transcriptomics (gene expressions) data collected from TCGA, covering over 8000 samples and 16 different cancer types.

1. *Gene mutation rate*: Calculate the average of single nucleotide variations and copy number aberrations in a cancer type. Use the number of non-silent mutations in that gene divided by the exon length as the mutation frequency of each gene and the number of times that gene is amplified or deleted in a given cohort as the copy number mutation rate for each gene.
2. *Differential DNA methylation rate*: Calculate the average of the differences in methylation signal between tumor and matched normal samples in a cancer type as in Eq. 1. For each gene, the beta(β) values of all CpG sites within the defined promoter region were averaged to calculate the average promoter methylation level after removing batch effects using ComBat [18]:

$$dm_i^c = \frac{1}{|S_C|} \sum_{s \in S_C} \left(\beta_{si}^t - \beta_{si}^n\right) \quad (1)$$

where c denotes the specific cancer type, β_i^t, β_i^n are the methylation levels of gene i in tumor and matched normal samples, respectively, and S_C is the number of all samples for a given cancer type.

3. *Differential expression rate*: Calculate the mean of log2 fold change between gene expression values in tumor versus matched normal samples by using gene expression data after quantile normalization and batch effect correction by Combat [18].
4. *Gene feature matrix*: Concatenate these features as a gene feature matrix $\mathbb{R}^{N \times F}$, in which each row indicates a feature vector for each gene, after a column-by-column (feature-wise) min-max normalization. Set the missing values to 0.

2.2 Generation of Gene Association Profiles

The methods for gene association calculations are based on five diverse metrics (gene co-expression pattern, gene sequence similarity, pathway co-occurrence, gene semantic similarity, and PPI). For these gene association profiles, the associated values are in the range of [0, 1], where 1 indicates the highest association and 0 is the lowest association.

1. *Tissue co-expression*: The co-expression pattern between a pair of genes G_1 and G_2 is measured as the absolute Pearson's correlation coefficient of their gene expression vectors as in Eq. 2, based on the gene expression profiles of 79 normal human tissues (GEO code: GSE1133).

$$R_{\text{coexp}}(G_1, G_2) = \left|\frac{\text{cov}(G_1, G_2)}{\sigma(G_1)\sigma(G_2)}\right| \quad (2)$$

2. *Gene sequence similarity*: The protein sequences of all genes, downloaded from the NCBI RefSeq database (released February 2019), are aligned against each other by using the BLASTP program [19] with defaulted parameters, and then the sequence similarity between a pair of genes G_1 and G_2 is calculated as in Eq. 3. The gene semantic similarity should be normalized.

$$R_{\text{seq}}(G_1, G_2) = \frac{\text{BLAST}_{\text{bitscore}}(G_1, G_2) + \text{BLAST}_{\text{bitscore}}(G_2, G_1)}{\text{BLAST}_{\text{bitscore}}(G_1, G_1) + \text{BLAST}_{\text{bitscore}}(G_2, G_2)} \quad (3)$$

3. *Pathway co-occurrence*: The gene co-occurrence relationship is calculated using cosine similarity. A 337-dimensional vector is constructed for each gene using the human cancer pathways

from the KEGG database (released June 2021), assigned 1 to the dimension if the gene is included in the corresponding pathway and 0 otherwise. Then the gene co-occurrence relationship of two genes G_1 and G_2 is measured as follows:

$$R_{\text{path}}(G_1, G_2) = \frac{P_{G_1} \cdot P_{G_2}}{|P_{G_1}||P_{G_2}|} \tag{4}$$

4. *Gene semantic similarity*: The functional similarity is measured based on the semantic similarity among the gene ontology (GO) terms annotating genes by using an R package GOSemSim [20]. Gene semantic similarity score is calculated by using a best-match average strategy to combine the semantic similarity scores of multiple GO terms measured as the Wang method [21], which is implemented by setting the parameters of the mgeneSim function with the "measure" parameter as "Wang" and the "combine" parameter as "BMA." The GO terms used in measurement can be restricted by assigning the corresponding parameter to "BP" (biological process), "MF" (molecular function), and "CC" (cellular component). Thus, given a pair of genes G_1 and G_2 annotated by GO terms, the functional similarity is calculated as follows:

$$\begin{aligned}&\text{sim}_{\text{BMA}}(G_1, G_2)\\ &= \max\left(\frac{\sum_{i=1}^{m} \max_{1 \le j \le n} \text{sim}\left(\text{go}_i, \text{go}_j\right)}{m}, \frac{\sum_{j=1}^{n} \max_{1 \le i \le m} \text{sim}\left(\text{go}_i, \text{go}_j\right)}{n}\right)\end{aligned} \tag{5}$$

$$\begin{aligned}&R_{\text{GO}}(G_1, G_2)\\ &= \sqrt[3]{\text{sim}^{\text{BP}}_{\text{BMA}}(G_1, G_2)\text{sim}^{\text{MF}}_{\text{BMA}}(G_1, G_2)\text{sim}^{\text{CC}}_{\text{BMA}}(G_1, G_2)}\end{aligned} \tag{6}$$

5. *PPI*: The PPIs are collected from different sources, while most of them are downloaded from NDEx v2.5.1 [22], except for the CPDB network (version 35 and version 34) [23] and the STRING (version 11) network [24], the former being downloaded from http://cpdb.molgen.mpg.de/ and the latter being collected from https://stringdb-static.org/download/protein.links.v11.0/9606.protein.links.v11.0.txt.gz. The CPDB and STRING networks only consider high-confidence (probability of an interaction between two proteins) interactions. Thus, "complex" interactions (more than two partners) and interactions with scores <0.5 are excluded from the CPDB network, and interactions with scores <0.85 are removed from the STRING network.

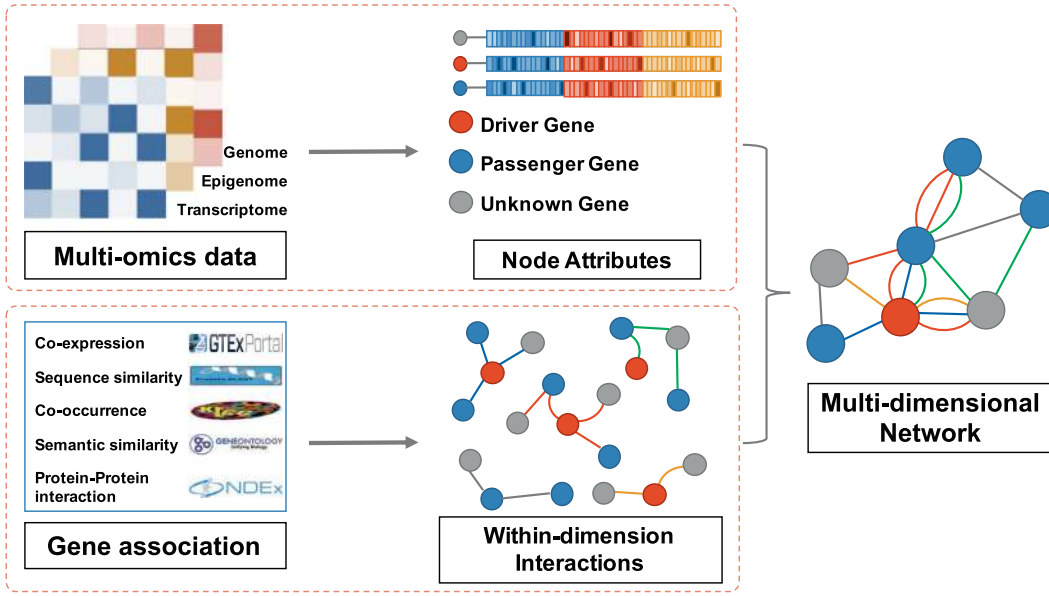


Fig. 1 Flowchart of multidimensional gene network construction

2.3 Construction of Multidimensional Gene Network

A multidimensional graph with about 20,000 nodes and two million edges (containing 5 types) is constructed by integrating diverse gene association profiles into a unified, gene-centric network, after filtering out small association values (Fig. 1).

1. *Filter out weak associations in gene association profiles*: To retain strongly correlated gene associations, the pathway co-occurrence is set to 0.5. Since the network of tissue co-expression and gene semantic similarity network is too dense, the threshold is set to a higher value of 0.8. The gene co-expression network retains approximately the top 1% of highly correlated edges (*see* **Notes 1** and **2**).
2. *Construct the multidimensional graph*: The nodes in the graph represent genes and the links represent their respective relationships, and it consists of a set of N nodes $\mathcal{V} = \{v_1, \cdots, v_N\}$ and D sets of edges $\{\mathcal{E}_1, \cdots, \mathcal{E}_D\}$. Each edge set $\mathcal{E}_d$ describes the d-th type of relation between the nodes in the corresponding d-th dimension. These D types of relations can be expressed by D adjacency matrices $A^{(1)}$,..., $A^{(D)}$. Besides, each node is characterized as a multi-omics feature vector (*see* **Note 3**).

2.4 Multi-Omics and Multidimensional Graph Attention Network

To learn knowledge from the multidimensional graph, instead of fusing different edges into a single edge to form a homogeneous graph, MODIG applies a GAT block for within-dimension interactions to get the dimension-specific gene representations and a joint

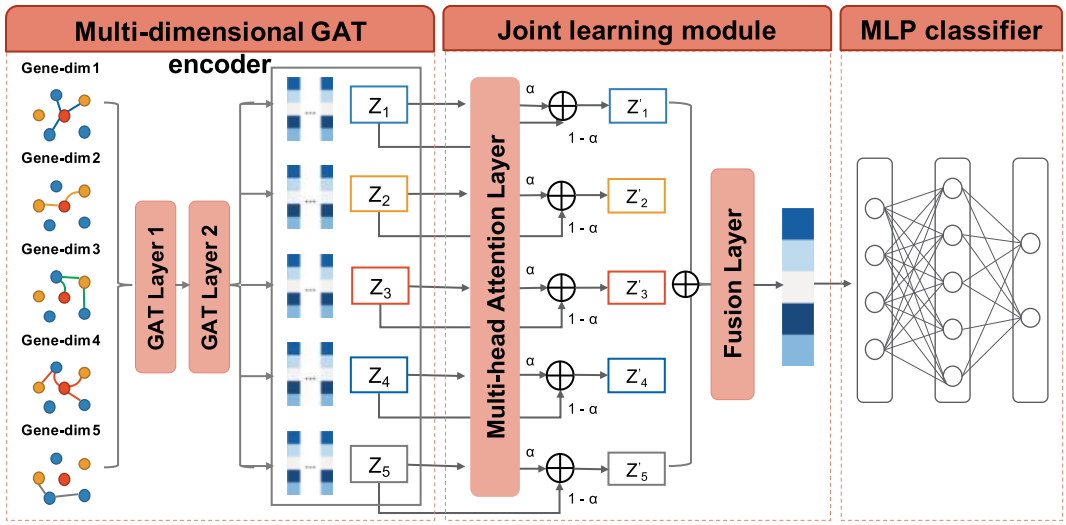


Fig. 2 Schematic diagram of cancer driver gene prediction method, MODIG

learning module to adaptatively learn the importance of different dimensional representations and fuse them by an attention mechanism for downstream cancer driver gene prediction (Fig. 2).

2.4.1 Multidimensional GAT Encoder

The GAT encoder is used to learn the within-dimension representation of genes, which is a GNN method that employs an attention mechanism to aggregate node features [25]. Given the input vertex feature $h = \left\{ \vec{h}_1, \vec{h}_2, \cdots, \vec{h}_n \right\}$, $\vec{h}_i \in \mathbb{R}^{N \times F}$, where N is the number of nodes and F is the number of features in each node. The 2 GAT layers with 300 and 100 hidden channels and 3 attention heads with a 0.25 dropout rate together are stacked as a GAT block. Then, the GAT block is applied on intra-dimensional interactions to generate dimension-specific gene representations Z_1, Z_2, Z_3, Z_4, and Z_5, respectively, by updating the vertex representations through the following steps:

$$e_{ij} = a\left(\mathrm{W}\vec{\mathrm{h}}_i, \mathrm{W}\vec{\mathrm{h}}_j\right) \tag{7}$$

$$e_{ij} = \mathrm{LeakReLU}\left(\vec{a}^{\mathrm{T}}\left[\mathrm{W}\vec{\mathrm{h}}_i \mathrm{W}\vec{\mathrm{h}}_j\right]\right) \tag{8}$$

$$\alpha_{ij} = \mathrm{softmax}_j\left(e_{ij}\right) = \frac{\exp e_{ij}}{\sum_{k \in \mathcal{N}_i} \exp e_{ik}} \tag{9}$$

$$\vec{h}'_i = \sigma\left(\frac{1}{K}\sum_{K=1}^{K}\sum_{j \in N_i} \alpha_{ij}^k W^k \vec{h}_j\right) \tag{10}$$

where $\boldsymbol{W}$ and $\vec{a}$ are trainable parameters and $\|$ is the concatenation operation.

2.4.2 Joint Learning Module

The joint learning module can adaptatively learn the importance of different dimensional representations and fuse them by an attention mechanism for downstream cancer driver gene prediction.

1. *Cross-dimension information sharing*: Given the representation from D different dimensions as $\{Z_1, \cdots, Z_M\}$, a self-attention layer is employed to share information across all dimensions. For each dimension-specific representation, associate a key matrix $K_i \in \mathbb{R}^{n \times k}$ and a query matrix $Q_i \in \mathbb{R}^{n \times k}$ with it as follows:

$$K_i = Z_i W_k, \quad Q_i = Z_i W_q \tag{11}$$

Then, propagate information among all dimensions as follows:

$$\widehat{Z_i} = \sum_{i=1}^{D} \text{softmax}\left(\left[\frac{Q_i K_i^{\mathrm{T}}}{\sqrt{k}} \right]_{i=1}^{D} \right) Z_i \tag{12}$$

Next, the final representation for i-th dimension w is calculated by incorporating the relevant global information $\widehat{Z_i}$ of the i-th dimension with a weight α, as in Eq. 13.

$$Z_i' = \alpha \widehat{Z_i} + (1-\alpha) Z_i, \quad 0 \le \alpha \le 1 \tag{13}$$

2. *Multidimension fusion*: A fusion layer learns the corresponding importance of dimension-specific representations and combines all dimension-specific representations to obtain the final gene representation Z_f as follows:

$$Z_f = \sum_{i}^{D} w_i Z_i', \quad w_i = \text{softmax}\left(q^{\mathrm{T}} \cdot \tanh\left(W \cdot \left(Z_i' \right)^T + b \right) \right) \tag{14}$$

where q is a shared attention vector and w_i is the weight of the i-th dimension.

2.4.3 MLP Classifier

After obtaining the final gene representation, a semi-supervised classification task is performed with the MLP classifier to identify cancer driver genes. The loss function is binary cross-entropy by adding a weight of 2.7 for positive samples owing to the unbalance of two classes (non-cancer and cancer genes) as in Eq. 15.

$$L = -(py * \log \sigma(x) + (1-y) * \log(1-\sigma(x))) \tag{15}$$

3 Implementation

3.1 Input Data

1. *Omics feature matrix*: Calculate the mutation rate, differential DNA methylation level, and differential expression rate for each gene in different types of cancer to obtain a gene feature matrix, based on the multi-omics data collected from TCGA or your cohort.

2. *Labeled genes for training*: The positive samples are obtained from multiple sources, such as the Network of Cancer Genes (NCG) v6.0 [26], COSMIC Cancer Gene Census (CGC v91) [27], and DigSEE [28]. The negative samples are generated by recursively removing potentially cancer-related genes from all protein-coding genes, such as genes included in the NCG, genes associated with cancer pathways in the KEGG database, genes present in the Online Mendelian Inheritance in Man (OMIM) database, MutSigdb predicted genes associated with cancer, and genes whose expressions are associated with cancer gene expression. As a result, 796 positive samples and 2187 negative samples were obtained for MODIG training (*see* **Note 4**).
3. *Independent test sets*: The first set (Independent Set 1) comprises manually curated cancer genes annotated according to validated oncogenic effects from the OncoKB [29] database and literature-curated cancer genes from the ONGene [30] database. The second set (Independent Set 2) comprises candidate cancer genes from the NCG [26] which are nonoverlapping with the known cancer gene set used for MODIG's training as well as high-confidence cancer genes compiled using different computational tools [11].

3.2 Software

To run MODIG downloading from GitHub (https://github.com/zjupgx/modig), the following tools and packages must be installed:

1. MODIG is written in Python 3.8, Pytorch 1.8.1, and Pytorch geometric library 2.0.0 [31, 32]. In addition, the recommendation parameters for MODIG are the use of the Adam optimizer with a learning rate of 0.001, a weight decay of 0.0005, and a dropout rate of 0.25 for 1000 epochs.
2. The command to run MODIG is *python main.py -t output -ppi CPDB*. Several parameters can be tuned: *--thr_go*, *--thr_seq*, *--thr_exp*, *--thr_path*, etc. Refer to the *main.py* file for a detailed description of all parameters.
3. In the prediction results, genes with scores more than 0.99 are considered potential cancer driver genes. In particular, the thresholds are set artificially and can be adjusted accordingly to the task.

4 Notes

1. The thresholds of gene association profiles are set artificially based on a priori knowledge when constructing the edges for the multidimensional graphs and can be adjusted according to the specific task and input data.

2. To further improve the performance of MODIG, some network noise reduction methods can be considered to improve the reliability of gene networks.
3. Assigning multi-properties to edges might comprise the scalability of the model, especially for different cancer types. Constructing cancer-specific multidimensional gene networks may be more suitable for cancer-specific driver gene prediction tasks.
4. Due to issues such as label scarcity and lack of cancer-specific networks, MODIG is currently not sufficiently reliable and not recommended for driver gene prediction at the cancer-specific level.

Acknowledgments

This work was supported by the National Natural Science Foundation of China [Grant No. 32370712], the Zhejiang Provincial Natural Science Foundation of China [Grant No. LQ24C060005], the China Postdoctoral Science Foundation [Grant No. 2023M743050], and the Postdoctoral Research Project in Zhejiang Province. We thank the Information Technology Center and State Key Lab of CAD&CG; the Innovation Institute for Artificial Intelligence in Medicine, Zhejiang University; and Alibaba Cloud for the support of computing resources. We also gratefully acknowledge the clinical contributors and data producers from the TCGA Research Network for referencing the TCGA datasets.

References

1. Vogelstein B, Papadopoulos N, Velculescu VE et al (2013) Cancer genome landscapes. Science 340:1546–1558. https://doi.org/10.1126/science.1235122
2. Martincorena I, Campbell PJ (2015) Somatic mutation in cancer and normal cells. Science 349:1483–1489. https://doi.org/10.1126/science.aab4082
3. Stratton MR, Campbell PJ, Futreal PA (2009) The cancer genome. Nature 458:719–724. https://doi.org/10.1038/nature07943
4. Zhu X, Zhao W, Zhou Z, Gu X (2023) Unraveling the drivers of tumorigenesis in the context of evolution: theoretical models and bioinformatics tools. J Mol Evol 91:405–423. https://doi.org/10.1007/s00239-023-10117-0
5. Cancer Genome Atlas Research Network, Weinstein JN, Collisson EA et al (2013) The Cancer Genome Atlas Pan-Cancer analysis project. Nat Genet 45:1113–1120. https://doi.org/10.1038/ng.2764
6. Zhang J, Bajari R, Andric D et al (2019) The International Cancer Genome Consortium Data Portal. Nat Biotechnol 37:367–369. https://doi.org/10.1038/s41587-019-0055-9
7. Cheng F, Zhao J, Zhao Z (2016) Advances in computational approaches for prioritizing driver mutations and significantly mutated genes in cancer genomes. Brief Bioinform 17:642–656. https://doi.org/10.1093/bib/bbv068
8. Zhou Z, Zou Y, Liu G et al (2017) Mutation-profile-based methods for understanding selection forces in cancer somatic mutations: a comparative analysis. Oncotarget 8:58835–58846. https://doi.org/10.18632/oncotarget.19371

9. Zhao W, Yang J, Wu J et al (2021) CanDriS: posterior profiling of cancer-driving sites based on two-component evolutionary model. Brief Bioinform 22:bbab131. https://doi.org/10.1093/bib/bbab131
10. Martínez-Jiménez F, Muiños F, Sentís I et al (2020) A compendium of mutational cancer driver genes. Nat Rev Cancer:1–18. https://doi.org/10.1038/s41568-020-0290-x
11. Bailey MH, Tokheim C, Porta-Pardo E et al (2018) Comprehensive characterization of cancer driver genes and mutations. Cell 173:371–385.e18. https://doi.org/10.1016/j.cell.2018.02.060
12. Bradner JE, Hnisz D, Young RA (2017) Transcriptional addiction in cancer. Cell 168:629–643. https://doi.org/10.1016/j.cell.2016.12.013
13. Bell CC, Gilan O (2020) Principles and mechanisms of non-genetic resistance in cancer. Br J Cancer 122:465–472. https://doi.org/10.1038/s41416-019-0648-6
14. Schulte-Sasse R, Budach S, Hnisz D, Marsico A (2021) Integration of multiomics data with graph convolutional networks to identify new cancer genes and their associated molecular mechanisms. Nat Mach Intell 3:513–526. https://doi.org/10.1038/s42256-021-00325-y
15. Peng W, Tang Q, Dai W, Chen T (2021) Improving cancer driver gene identification using multi-task learning on graph convolutional network. Brief Bioinform 23:bbab432. https://doi.org/10.1093/bib/bbab432
16. Zhao W, Gu X, Chen S et al (2022) MODIG: integrating multi-omics and multi-dimensional gene network for cancer driver gene identification based on graph attention network model. Bioinformatics 38:4901–4907. https://doi.org/10.1093/bioinformatics/btac622
17. Kipf TN, Welling M (2016) Semi-supervised classification with graph convolutional networks. ICLR. arXiv:1609.02907
18. Johnson WE, Li C, Rabinovic A (2007) Adjusting batch effects in microarray expression data using empirical Bayes methods. Biostatistics 8:118–127. https://doi.org/10.1093/biostatistics/kxj037
19. Shiryev SA, Papadopoulos JS, Schäffer AA, Agarwala R (2007) Improved BLAST searches using longer words for protein seeding. Bioinformatics 23:2949–2951. https://doi.org/10.1093/bioinformatics/btm479
20. Yu G, Li F, Qin Y et al (2010) GOSemSim: an R package for measuring semantic similarity among GO terms and gene products. Bioinformatics 26:976–978. https://doi.org/10.1093/bioinformatics/btq064
21. Wang JZ, Du Z, Payattakool R et al (2007) A new method to measure the semantic similarity of GO terms. Bioinformatics 23:1274–1281. https://doi.org/10.1093/bioinformatics/btm087
22. Pillich RT, Chen J, Churas C et al (2021) NDEx: accessing network models and streamlining network biology workflows. Curr Protoc 1:e258. https://doi.org/10.1002/cpz1.258
23. Herwig R, Hardt C, Lienhard M, Kamburov A (2016) Analyzing and interpreting genome data at the network level with ConsensusPathDB. Nat Protoc 11:1889–1907. https://doi.org/10.1038/nprot.2016.117
24. Szklarczyk D, Gable AL, Nastou KC et al (2021) The STRING database in 2021: customizable protein–protein networks, and functional characterization of user-uploaded gene/measurement sets. Nucleic Acids Res 49:D605–D612. https://doi.org/10.1093/nar/gkaa1074
25. Veličković P, Cucurull G, Casanova A et al (2018) Graph attention networks. ICLR. arXiv:1710.10903
26. Repana D, Nulsen J, Dressler L et al (2019) The Network of Cancer Genes (NCG): a comprehensive catalogue of known and candidate cancer genes from cancer sequencing screens. Genome Biol 20:1. https://doi.org/10.1186/s13059-018-1612-0
27. Tate JG, Bamford S, Jubb HC et al (2019) COSMIC: the catalogue of somatic mutations in cancer. Nucleic Acids Res 47:D941–D947. https://doi.org/10.1093/nar/gky1015
28. Kim J, So S, Lee H-J et al (2013) DigSee: disease gene search engine with evidence sentences (version cancer). Nucleic Acids Res 41:W510–W517. https://doi.org/10.1093/nar/gkt531
29. Chakravarty D, Gao J, Phillips S et al (2017) OncoKB: a precision oncology knowledge base. JCO Precis Oncol:PO.17.00011. https://doi.org/10.1200/po.17.00011
30. Liu Y, Sun J, Zhao M (2017) ONGene: a literature-based database for human oncogenes. J Genet Genomics 44:119–121. https://doi.org/10.1016/j.jgg.2016.12.004
31. Paszke A, Gross S, Massa F, et al (2019) PyTorch: an imperative style, high-performance deep learning library. NeurIPS. arXiv:1912.01703
32. Fey M, Lenssen JE (2019) Fast graph representation learning with PyTorch Geometric. ICLR. arXiv:1903.02428

Chapter 14

Predictive Modeling of Anticancer Drug Sensitivity Using REFINED CNN

Daniel Nolte, Omid Bazgir, and Ranadip Pal

Abstract

Over the past decade, convolutional neural networks (CNNs) have revolutionized predictive modeling of data containing spatial correlations, specifically excelling at image analysis tasks due to their embedded feature extraction and improved generalization. However, outside of image or sequence data, datasets typically lack the structural correlation needed to exploit the benefits of CNN modeling. This is especially true regarding anticancer drug sensitivity prediction tasks, as the data used is often tabular without any embedded information in the ordering or locations of the features when utilizing data other than DNA or RNA sequences. This chapter provides a computational procedure, REpresentation of Features as Images with NEighborhood Dependencies (REFINED), that maps high-dimensional feature vectors into compact 2D images suitable for CNN-based deep learning. The pairing of REFINED mappings with CNNs enables enhanced predictive performance through reduced model parameterization and improved embedded feature extraction as compared to fully connected alternatives utilizing the high-dimensional feature vectors.

Key words Deep learning, Convolutional neural networks, Drug sensitivity prediction

1 Introduction

A crucial intent of data-driven precision medicine for cancer therapeutics is to accurately identify the most effective anticancer drug or combination of drugs for each individual tumor [1]. Anticancer drug sensitivity prediction from high-dimensional molecular fingerprints and genomics has substantially benefited from the surge in the availability of high-throughput screening data [2]. Given the vast number of features typically found in these high-dimensional datasets, feature selection is a common critical step to achieve enhanced predictive performance [3, 4]. With the rise in the amount of data, numerous deep learning-based approaches have shown outstanding drug sensitivity prediction performance as they enable built-in feature extraction when supplied with enough samples [5]. In addition to improving the generalization performance,

Alexander Krasnitz and Pascal Belleau (eds.), *Cancer Bioinformatics*, Methods in Molecular Biology, vol. 2932,
https://doi.org/10.1007/978-1-0716-4566-6_14,

the built-in feature extraction removes the need to perform feature selection, which can be time-consuming and expensive if done by a subject matter expert. Since molecular features and genomics data are typically represented as 1D vectors, these methods commonly employ fully connected deep neural networks as there are no spatial neighborhood correlations within the data to utilize the full potential of CNNs.

CNNs have emerged as the most established deep learning algorithm in the past decade due to their ability to handle the spatial correlations in images along with the rising availability of labeled data. Throughout their dominant performance in the ImageNet Large Scale Visual Recognition Competition (ILSVRC) from 2012 to 2017 [6], tremendous research and advancements were developed which allowed deeper model architectures while improving the generalization performance [7, 8]. Utilizing the developments of CNNs, astounding results have been achieved in medical imaging research [9], such as radiology [10] and magnetic resonance imaging analysis [11], with some tasks achieving expert-level performance. Regarding drug sensitivity prediction tasks, CNNs have only been applied on DNA or RNA sequencing data, as the sequences contain a natural ordering which can be processed by one-dimensional (1D) CNNs. However, for other modalities of data such as gene expression or molecular descriptors, the ordering of the features does not contain relevant dependencies.

To make CNN-based learning amenable to these modalities, REFINED [12] was developed as a representation learning methodology with the aim of arranging high-dimensional vectors into compact images conducive to CNN-based modeling. In this arrangement, similar features are positioned close by, while dissimilar features are placed far apart in the image. The REFINED algorithm maps tabular data into images in two main phases: (1) initial coordinate learning of each feature in the 2D space using a manifold learning technique and unique nearest pixel placement to avoid feature overlap and (2) hill climbing to optimize the feature pixel locations to match the true feature distances while maintaining unique pixels for each feature. REFINED CNN can be applied to any tabular data and has achieved superior performance on various drug sensitivity prediction and survival analysis tasks [12–15]. This chapter presents a detailed overview of the procedure to generate a REFINED mapping of input features and subsequently train a CNN on the mapped samples for accurate anticancer drug sensitivity prediction.

2 Methods

REFINED is a general methodology that can be applied on any type of tabular data and, as with all machine learning methods, starts by preparing the data into a quantifiable form suitable for

computer consumption. For drug sensitivity tasks, we start by extracting relevant features such as molecular drug descriptors or individual tumor gene expression values. Once the data is in a tabular form, the REFINED process begins initializing a mapping to optimally place features on a 2D square grid. The feature mapping can be viewed as a mapping of features to unique 2D coordinates of a square grid large enough to fit all features. REFINED initializes the mapping by learning a 2D manifold of the features and assigning each feature to its closest unique pixel. Then REFINED optimizes the feature mapping to mimic the true feature distances calculated from all samples. Each learned coordinate is associated with a pixel in an image, and a single image mapping is universal across all samples. In other words, we learn for the entire dataset, where the location of each feature resides in a 2D space such that similarity/dissimilarity among the features remains as close as possible to the initial sample space before applying the manifold learning technique. Once the REFINED mapping is optimized, every sample can be converted into an image and used to train a CNN for prediction using conventional training methods.

The general steps involved in generating a REFINED CNN model are depicted in Fig. 1 and are detailed as follows:

1. Data extraction and preprocessing: Extract and process the molecular descriptors or genomics data such that each sample is represented by a complete 1D numerical vector.
2. Initialize REFINED mapping: Apply an initial manifold learning technique, such as multidimensional scaling (MDS), on the features, and assign each feature to its closest unique coordinate in a 2D grid space.
3. REFINED hill climbing: Optimize the feature mapping by minimizing the difference between the new locations and true distances among the features using hill climbing.
4. CNN model training: Use the learned mapping to transform each 1D sample into compact 2D images, and use the images to train a CNN.

2.1 Data Extraction and Preprocessing

A wide range of data modalities have been identified as potential predictors of drug sensitivity, such as various genetic characterizations and molecular descriptors. This data is typically extracted using task specific quantification techniques which represent different characterizations as 1D vectors of features. Here we focus on molecular descriptors, although this method can be beneficial for any 1D representation when there are enough high-dimensional samples (*see* **Note 1**). For drug sensitivity prediction datasets, these characterizations are conducted across an extensive range of potential cancer-inhibiting drugs, each representing a sample in the dataset (*see* **Note 2**). Once the characterization has been collected

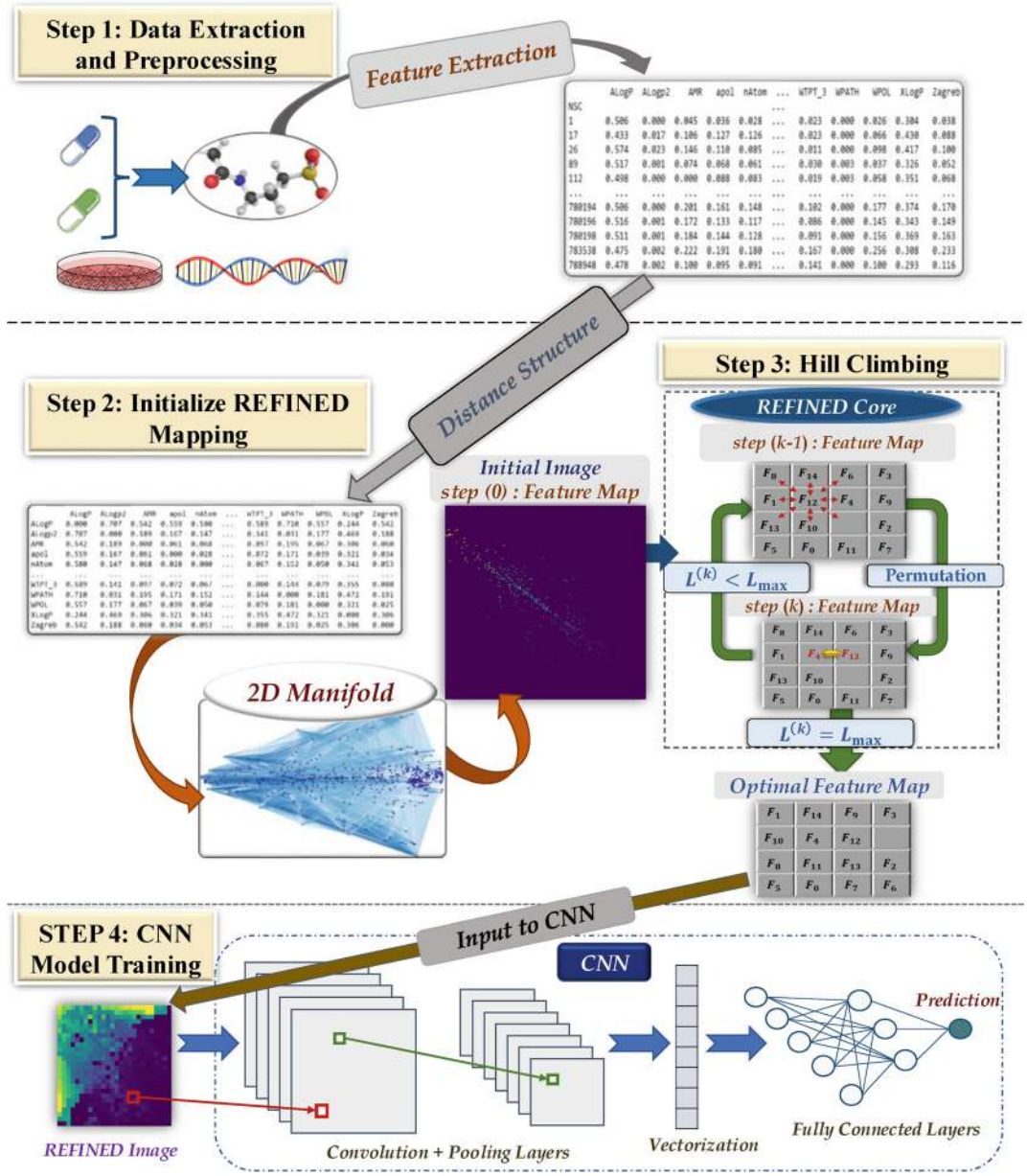


Fig. 1 Schematic overview of the REFINED CNN procedure. (1) Extract and preprocess data. (2) Initialize REFINED by learning a 2D manifold and assigning each feature to its closest unique pixel on a unit square grid. (3) Optimize the mapping to match the true feature distances in the initial sample space through hill climbing. (4) Train a CNN on the mapped samples. (Reproduced from [12] (CC-BY 4.0))

for each sample and centralized into one matrix, the data must be preprocessed. Some characterizations can lead to missing values due to numerous factors, and those features must be delt with either through removal or imputation. Common imputation techniques include filling the missing values with the mean of the

feature, while more sophisticated techniques exist such as k-nearest-neighbor [16] (kNN) or MissForest imputation [17]. Depending on the characterization, one should ensure the features are normalized such that they are all on the same scale (*see* **Note 3**). Lastly, drug responses are often recorded as measures of concentration resulting in half-maximal response which are logarithmic in nature due to the dose administration protocol. To correct for this, the concentrations are typically converted to sensitivities using a negative log transformation, $y = -\log(\text{concentration})$.

2.2 REFINED Manifold Learning

Now the preprocessed molecular features can be represented as a complete $n \times p$ matrix, where n is the number of samples and p is the number of features. The goal of using a manifold learning technique is to generate a $p \times 2$ matrix where each of the p features across the entire dataset are represented by two values, which are particular features' coordinates in a 2D space. Using the 2D manifold, each feature can be assigned to its closest unique pixel on a square grid in an iterative manner to arrive at an initial feature mapping.

We start by learning a 2D manifold of the features rather than the samples as normally done. We employ scikit-learn [18] manifold functions such as multidimensional scaling [19] or t-SNE [20]. This requires transposing the input data matrix as the scikit-learn methods are programmed to transform row (sample) wise rather than column (feature) wise. Using the learned manifold, the features can then be projected into two dimensions representing each features' coordinates on a 2D plane as a $p \times 2$ matrix, L. Subsequently, each feature must be mapped to a unique pixel to avoid sparse images with multiple features occupying the same location as shown in Fig. 2. Notice the sparsity in the top row of images generated by directly utilizing the learned 2D manifold which would result in deficient performance with CNNs as the overlapping features would induce interference. To ensure unique feature locations, we normalize the 2D coordinates onto the range [0, 1] by ranking the x and y coordinates separately and dividing each rank by p, essentially replacing each column of L with its fractional ranking equivalent to acquire L_{Ranked}.

Next, we iteratively assign each feature to its closest unique pixel. The process starts by initializing a $q^2 \times 3$ matrix, $M^{(0)}$, whose first two columns represent the x and y integers of pixel locations on a $q \times q$ grid that acts as our 2D pixel space, with q calculated as $\lceil \sqrt{p} \rceil$ and the third column of $M^{(0)}$ reserved for selected feature indices. The calculation of q ensures more pixels than features, allowing each feature to have its own exclusive pixel on a unit square grid with the unselected pixels set to null. Using the first two columns of $M^{(0)}$, the centroids of each pixel location can be calculated on the range [0,1] through the element wise operation $C_{ij} = \frac{M_{ij}}{q} + \frac{1}{2q}$.

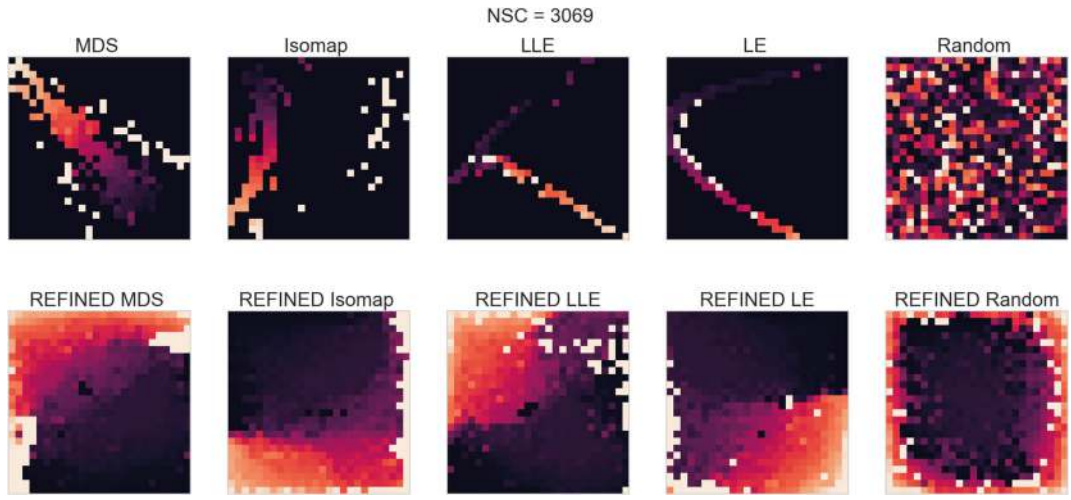


Fig. 2 Depiction of REFINED images employing different manifold learning techniques, which generate REFINED images with varying patterns. The top row of mappings depicts the overlapping features after directly applying the initial manifold, while the bottom row shows the learned REFINED mappings with unique pixels for each feature. Here, we used the molecular descriptors extracted from the NCI60 dataset with chloramphenicol (NSC ID = 3069) as the sample drug in all images. (Reproduced from [13] (CC-BY 4.0))

Now the distances between feature locations in the 2D manifold space and pixel centroids on the 2D square grid are easily calculated using the pairwise Euclidean distance between the 2D centroids, C, and the rank normalized 2D feature coordinates, L_{Ranked}.

Utilizing the pairwise distances, the iterative unique pixel mapping begins by evaluating each feature for which pixel resides closest to them. Since there will likely be feature collisions where multiple features are closest to a particular pixel, all available pixels are then evaluated, and the nearest feature among the ones who selected that pixel is assigned to it by placing the feature index into the third column of M at the specific pixel's row. This iteratively repeats for all unassigned features until all features have been assigned a unique pixel in the 2D mapping. Once each feature has been assigned a pixel, we have arrived at the initial REFINED feature mapping, $M^{(0)}$, which we subsequently further optimize. Note that, depending on the choice of manifold learning technique, the different similarity/dissimilarity metrics can lead to diverse patterns in the generated REFINED mappings, which can be combined to create an ensemble of REFINED CNN models (*see* **Note 4**).

2.3 Hill Climbing

In an image, we have discrete 2D coordinates, but the 2D coordinates learned through manifold learning are continuous and often so sparse that a cluster of features likely has multiple features occupying a single pixel. When solving the previous initial mapping, compromises occur for all multiple-feature to single-pixel

collisions. To resolve this issue, we apply a hill climbing technique such that, in an iterative approach, we arrive at a locally optimal mapping that closely mimics the true feature distances in the n-dimensional sample space.

The hill climbing process begins by calculating the pairwise distances between the features on the initial $n \times p$ tabular data, resulting in a $p \times p$ symmetric matrix, δ. This matrix serves as the ground-truth feature distances that will be approximated in the 2D mapping using the hill climbing procedure. Next, the initial 2D feature mapping, $M^{(0)}$, can be evaluated by calculating the pairwise Euclidean distance in the mapped 2D grid space, resulting in a $p \times p$ symmetric matrix, $\widehat{\delta}$. Using these distances, we can now compute a loss function of the difference between feature distances in the initial and mapped spaces that will be minimized through the hill climbing process:

$$\mathcal{L}\left(\delta, \widehat{\delta}\right) = \sqrt{\frac{\sum_{i=1}^{p}\sum_{j=1}^{p}\left(\delta_{i,j} - \widehat{\delta}_{i,j}\right)^2}{\sum_{i=1}^{p}\sum_{j=1}^{p}\left(\delta_{i,j}\right)^2}}$$

With the loss function defined, the iterative process begins performing permutations by swapping each pixel with its eight adjacent pixels in the case of selecting a 3 × 3 kernel, resulting in nine configurations for each pixel including its initial location. Each configuration is evaluated using $\mathcal{L}$, and the swap that results in the lowest loss is the one that is performed for each pixel. This procedure can be computationally expensive and would be computationally impracticable if considering all potential feature location combinations sequentially. Therefore, a heuristic permutation method was used in the implementation of [12] by breaking the full $q \times q$ mapping into separate adjacent 3 × 3 grids, with the pixels under evaluation occupying the center of each 3 × 3 grid. For example, for the first grid structure, the top-left most pixel, and every pixel spaced three pixels apart vertically or horizontally until the limit of the image has been reached, is evaluated. This allows parallel processing of each grid separately by distributing a subset of centroid coordinates to separate processors (*see* **Note 5**). Using this method, the only pixels that are evaluated are spaced three pixels apart vertically or horizontally, removing the potential for collisions between the separate processes. To evaluate every pixel on the 2D mapping, each 3 × 3 grid structure is shifted to nine distinct locations such that every pixel of the top-left most 3 × 3 grid of the image resides at the center once. This moves every grid in the same direction and is done iteratively, with the centroid pixels being selected and distributed to each process and the processes returning the best locations for each of their given centroids. Then the best

swap for each centroid pixel is performed for the current grid evaluation structure, and subsequently, the next group of centroid pixels are considered until all pixels have been evaluated. At the end of evaluating and swapping all pixels to their locally optimal locations, we have arrived at the next feature map, $M^{(k)}$, with k being the iteration number. This process repeats for the user specified number of iterations, or until a suitable loss has been achieved. After the optimal mapping, M, is obtained, we can map each 1D feature vector into 2D images suitable for CNN utilization. This involves transforming each of the n rows of samples into images by applying the learned mapping.

2.4 Model Training

Training a REFINED CNN model is highly dependent on the complexity of the problem, the availability of enough training samples, and sufficient computational resources. CNNs contain many hyperparameters, and finding an optimal architecture is often task specific. For instance, in [12], the CNN architecture we used for the NCI60 dataset takes a single REFINED image of molecular drug descriptors as the input for prediction since we had a relatively large training dataset for each cell line. However, in the same paper, the CNN model for the GDSC dataset takes two REFINED images, one associated with molecular descriptors and another associated with cell line gene expressions, to predict the drug response. Based on our experimentation, the width of the CNN is as important as its depth since we want to extract as much information as possible from the entire raw features in the REFINED images. We have noticed that increasing the number of kernels in the convolutional layers often helps extract more abstract feature maps heuristically.

Once the samples have been converted into images, we typically employ PyTorch [21] or TensorFlow [22], regardless of the version, for the implementation and training of deep neural networks including CNNs. Throughout the rest of the chapter, we will be focusing on a PyTorch implementation such as the code found in [14, 15], although a TensorFlow implementation can be found in the code of [12]. First, the model architecture is initialized by the user specifying the desired layers, such as convolutional layers, pooling layers, or dense layers, along with activation functions and hyperparameters that match the task complexity. In PyTorch, this is done by initializing a class that inherits the PyTorch Module class, leaving the user to define two functions: one for the model architecture in an *__init__* method and another method, *forward*, that takes a data matrix as input and performs a forward pass of the data through each of the previously defined layers.

With the model defined, the user can select which optimizer to use for training such as stochastic gradient decent (SGD) or Adam. We typically had success using the Adam optimizer. Next, the images should be split into training and evaluation datasets and

be prepared for input into another user-defined method for training. In the user-defined training method, the process will repeat for the specified number of training epochs, or until a valid spotting point is reached. This process consists of multiple steps made easy through PyTorch functions. At the beginning of an epoch, the gradient of the optimizer must be set to zero or NONE using the optimizer objects *zero_grad* method. Next, using an object of the user-defined model class from before, make a forward pass of the data by calling the object with the data as input. Finally, a specific loss, such as mean squared error (MSE), can be calculated using a PyTorch evaluation metric or user-defined method, and the loss can be backpropagated through the model using the *backward* method on the loss object before calling the *step* method on the optimizer object. These two methods will handle the backpropagation of the gradients and updating of the model parameters, and this process can be run for the specified number of epochs or until training is halted (*see* **Note 6**).

With the model now trained to predict drug sensitivity using input REFINED images, it is important to verify the generalization performance of the model by evaluating the predictive performance on the held-out evaluation set. This can be done by using the trained model to predict the evaluation dataset responses and calculate a desired metric using the true and predicted responses which is typically done with a user-defined evaluation method that evaluates the model on an input dataset.

3 Notes

1. In [12], we conducted an experiment to compare REFINED CNN predictive performance using normalized root mean squared error (NRMSE) metric with synthetic data, where we created a dataset with various sample sizes (ranging between 50 and 10,000), various feature sizes (ranging between 20 and 4000), and different spurious feature ratios (20%, 50%, and 80%). We observed that, in the case of availability of a minimal number of samples (< 200), or small number of features (<100), it is more efficient to use shallow models (e.g., support vector machines or random forests) rather than REFINED CNNs. Figure 3 shows the results of the abovementioned experiment in the case of 20% spurious features—complete results could be found in [12].
2. Molecular descriptors or fingerprints can be extracted using various descriptor calculation software packages such as PaDEL [23], Mordred [24], and RDKit [25]. In [12–14], we used the PaDEL software package in conjunction with each unique inhibitor's NSC identifier to extract 672 descriptors after removal of features with missing values.

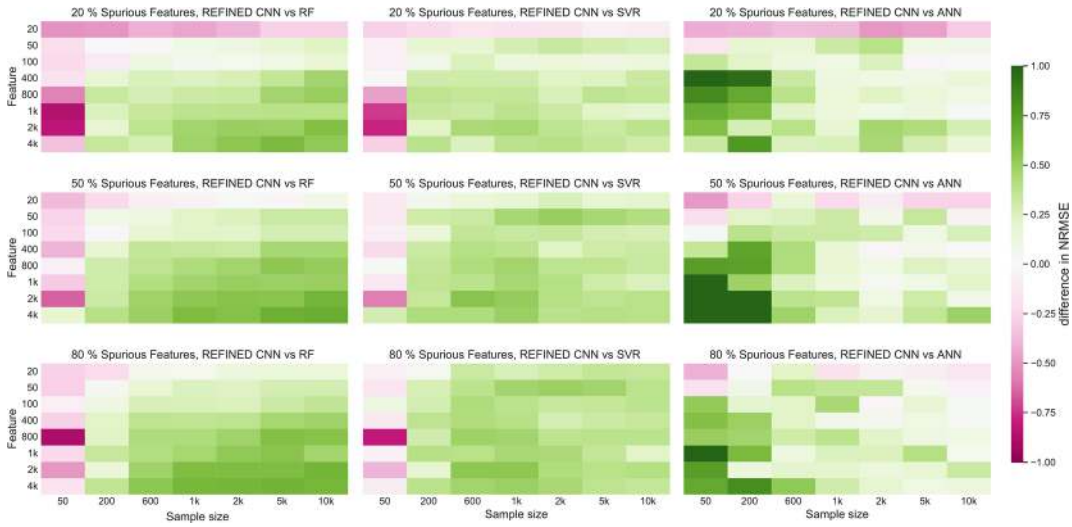


Fig. 3 Experimental results showing the predictive performance of REFINED CNN compared with random forests, support vector machines, and fully connected deep neural network methods while varying the number of samples and features available for model training using a synthetic dataset with 20% spurious features. (Reproduced from [12] (CC-BY 4.0))

3. For preprocessing steps such as normalization and imputation, we utilize the python package scikit-learn due to its versatile and user efficient functionality. Scikit-learn has a range of modules for various machine learning tasks such as imputation, feature selection, manifold learning, normalization, and various predictive modeling algorithms, all packaged with the same user-friendly functionality.
4. Depending on the choice of the initial manifold learning technique, the pattern of the REFINED image and consequently the predictive performance of the REFINED CNN would be different. In [13, 15], we have shown that an ensemble of REFINED CNN models, trained on separate REFINED mappings from different manifold learning techniques, can achieve improved model performance through stacking the associated CNN predictions with a simple linear regression. Figure 4 depicts a setup where four CNN models are trained using four separate REFINED mappings initialized from different manifold learning techniques. This can be performed by repeating **steps 2–4** for different manifold learning methods and learning an additional linear regression on a separate held-out dataset to combine the individual predictions into an accurate ensemble prediction.
5. For parallel processing of the hill climbing method, we used the MPI for Python package [26] which allowed one processor to act as the coordinating server and the remaining processors to act as clients. With this architecture, the server communicates

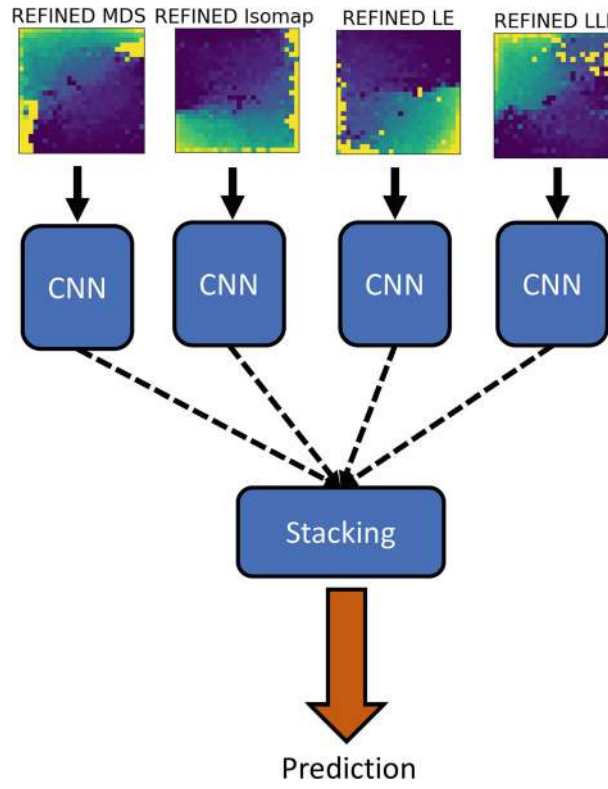


Fig. 4 Depiction of Ensemble REFINED: Four different REFINED images were created using MDS, Isomap, Laplacian eigenmaps (LE), and locally linear embedding (LLE), and for each set of REFINED images, a CNN was trained stacked using a linear regression model. (Reproduced from [13] (CC-BY 4.0))

the centroids to each of the clients and performs the swaps given the optimal swaps received from the clients. That leaves the clients to evaluate each of the nine swaps for all their given centroids and communicate to the server the optimal locations.

6. When training deep networks, employing early stopping and dynamic learning rate reduction is common to improve generalization error by reducing model overfitting. This is done at the cost of samples as the training and evaluation partitions must also accommodate another partition, termed the validation set, for model evaluation at each epoch. Although any percentage can be used depending on the number of samples available, a common splitting percentage is 60%, 20%, and 20% for the training validation and test datasets, respectively, as this reduces the potential sampling variation noise for validation and testing. Early stopping and learning rate schedulers track the number of successive epochs in which the model does not improve. After a specified number of epochs without improvement, the model reduces the learning rate by a user-defined

factor or halts training for learning rate scheduling and early stopping, respectively. In [12–15], early stopping was employed as it led to appropriately fit models with greater generalization performance and [14] also employed learning rate reduction to help fine-tune the models.

References

1. Krzyszczyk P et al (2018) The growing role of precision and personalized medicine for cancer treatment. Technology (Singap World Sci) 6(3–4):79–100. https://doi.org/10.1142/S2339547818300020
2. Ling A, Gruener RF, Fessler J, Huang RS (2018) More than fishing for a cure: the promises and pitfalls of high throughput cancer cell line screens. Pharmacol Ther 191:178–189. https://doi.org/10.1016/j.pharmthera.2018.06.014
3. Dong Z et al (2015) Anticancer drug sensitivity prediction in cell lines from baseline gene expression through recursive feature selection. BMC Cancer 15:489. https://doi.org/10.1186/s12885-015-1492-6
4. Koras K et al (2020) Feature selection strategies for drug sensitivity prediction. Sci Rep 10(1):9377. https://doi.org/10.1038/s41598-020-65927-9
5. Baptista D, Ferreira PG, Rocha M (2021) Deep learning for drug response prediction in cancer. Brief Bioinform 22(1):360–379. https://doi.org/10.1093/bib/bbz171
6. Russakovsky O et al (2015) ImageNet large scale visual recognition challenge. Int J Comput Vis 115(3):211–252. https://doi.org/10.1007/s11263-015-0816-y
7. Krizhevsky A, Sutskever I, Hinton GE (2012) ImageNet classification with deep convolutional neural networks. Association for Computing Machinery, New York
8. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Paper presented at the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) IEEE, Las Vegas, NV, USA
9. Sarvamangala DR, Kulkarni RV (2022) Convolutional neural networks in medical image understanding: a survey. Evol Intell 15(1):1–22. https://doi.org/10.1007/s12065-020-00540-3
10. Yamashita R, Nishio M, Do RKG, Togashi K (2018) Convolutional neural networks: an overview and application in radiology. Insights Imaging 9(4):611–629. https://doi.org/10.1007/s13244-018-0639-9
11. Bernal J et al (2019) Deep convolutional neural networks for brain image analysis on magnetic resonance imaging: a review. Artif Intell Med 95:64–81. https://doi.org/10.1016/j.artmed.2018.08.008
12. Bazgir O et al (2020) Representation of features as images with neighborhood dependencies for compatibility with convolutional neural networks. Nat Commun 11(1):4391. https://doi.org/10.1038/s41467-020-18197-y
13. Bazgir O, Ghosh S, Pal R (2021) Investigation of REFINED CNN ensemble learning for anticancer drug sensitivity prediction. Bioinformatics 37(Suppl_1):i42–i50. https://doi.org/10.1093/bioinformatics/btab336
14. Nolte D, Bazgir O, Ghosh S, Pal R (2023) Federated learning framework integrating REFINED CNN and Deep Regression Forests. Bioinform Adv 3(1):vbad036. https://doi.org/10.1093/bioadv/vbad036
15. Bazgir O, Lu J (2023) REFINED-CNN framework for survival prediction with high-dimensional features. iScience 26(9):107627. https://doi.org/10.1016/j.isci.2023.107627
16. Troyanskaya O et al (2001) Missing value estimation methods for DNA microarrays. Bioinformatics 17(6):520–525
17. Stekhoven DJ, Bühlmann P (2012) MissForest—non-parametric missing value imputation for mixed-type data. Bioinformatics 28(1):112–118
18. Pedregosa F et al (2011) Scikit-learn: machine learning in Python. J Mach Learn Res 12:2825–2830
19. Borg I, Groenen PJ (2005) Modern multidimensional scaling: theory and applications. Springer Science & Business Media, New York
20. Van der Maaten L, Hinton G (2008) Visualizing data using t-SNE. J Mach Learn Res 9(11):2579–2605
21. Paszke A et al (2019) Pytorch: an imperative style, high-performance deep learning library. Adv Neural Inf Process Syst 32:8026–8037

22. Abadi M et al (2016) Tensorflow: large-scale machine learning on heterogeneous distributed systems. arXiv preprint arXiv:160304467
23. Yap CW (2011) PaDEL-descriptor: an open source software to calculate molecular descriptors and fingerprints. J Comput Chem 32(7): 1466–1474
24. Moriwaki H, Tian Y-S, Kawashita N, Takagi T (2018) Mordred: a molecular descriptor calculator. J Chem 10(1):1–14
25. Landrum G. RDKit: open-source cheminformatics. https://www.rdkit.org
26. Dalcin L, Fang Y-LL (2021) mpi4py: status update after 12 years of development. Comput Sci Eng 23(4):47–54

Chapter 15

Anticancer Monotherapy and Polytherapy Drug Response Prediction Using Deep Learning: Guidelines and Best Practices

Amin Emad and David Earl Hostallero

Abstract

Cancer precision medicine aims to identify the best course of treatment for an individual. To achieve this goal, two important questions include predicting the response of an individual to a treatment strategy and identifying molecular markers that determine the response. The rapid growth of large publicly available databases containing clinical and molecular characteristics of cancer-derived samples paired with their response to single or multiple drugs, has enabled the development of computational models to answer these questions. In recent years, various deep learning models have been proposed to predict the response to polytherapy and monotherapies. However, selecting among all available options or developing new models for a particular study requires careful considerations and best practices to avoid various pitfalls. In this chapter, and drawing from our own studies, we will discuss various important points for choosing, utilizing, and developing such deep learning tools.

Key words Drug response prediction, Drug synergy prediction, Deep learning, Machine learning, Cancer precision medicine, Omics

1 Introduction

The rapid growth of large databases containing molecular and clinical properties of cancer derived samples and their response to different drugs or drug combinations such as TCGA [1], CCLE [2], CTRP [3], DepMap [4], GDSC [5], DrugComb [6], DrugCombDB [7], as well as advances in the area of deep learning (DL) [8] has fueled the development of computational models for precision cancer medicine. In particular, various DL models have been recently proposed to predict the response to monotherapies (henceforth referred to as drug response prediction (DRP)) and predicting the synergism of drug-pairs (henceforth referred to as drug synergy prediction (DSP)). Given the number of existing tools and potential approaches for these tasks, careful

Alexander Krasnitz and Pascal Belleau (eds.), *Cancer Bioinformatics*, Methods in Molecular Biology, vol. 2932,
https://doi.org/10.1007/978-1-0716-4566-6_15,

considerations are required to select the best approach for a particular application or dataset. Drawing examples from different tools that we have developed and studies that we have performed in this domain [9–14], we will discuss various important points for choosing, utilizing, and developing DL tools for DRP and DSP. While our focus will be on DL models, most of the guidelines provided here also apply to traditional machine learning or statistical approaches.

In this chapter, we use the term "samples" to refer to a wide range of cancer derived samples such as cancer cell lines (CCLs), patient-derived xenografts (PDX), and tumors, for which one wants to predict the effect of one or multiple drugs. We define DRP as the problem of predicting the response of samples to a specific compound. Similarly, DSP is concerned with predicting the synergy of two compounds, when used as a combinational treatment for a specific sample. We emphasize having a *sample* as an important distinction from related problems, such as drug combination recommendation and drug repurposing, in which the recommendations are not specific to a sample.

2 Methods

2.1 Preliminary Considerations and Overall Computational Pipeline

Prior to choosing the best tool or developing a new one, several questions should be answered. These questions clarify the specifications of the problem and will affect different aspects of the model. Some examples are provided below.

1. What are the input features for the model? The input features may involve different data modalities representing samples or drugs, each requiring different preprocessing and quality control steps.
2. What is the output that needs to be predicted? The type of output (e.g., Boolean, categorical, continuous) will affect the choice of the model, its architecture, and the loss function.
3. What is the final goal of the prediction? In some applications, one is interested in imputing missing values. For example, due to the large number of possible drug-pairs and cancer cell lines (CCLs), large databases of synergy scores (e.g., DrugComb [6]) contain many missing values, and one may need to impute those values. MARSY [14] is an example of such computational model for the DSP task. Alternatively, one may want a model that can predict response to a drug in new samples or for a new drug. Each of these tasks may require different model architectures, data processing, and evaluation.
4. Is the test set from the same domain as the training data or not? In some applications, the test set for which one needs to make predictions has distinct statistical characteristics. For example,

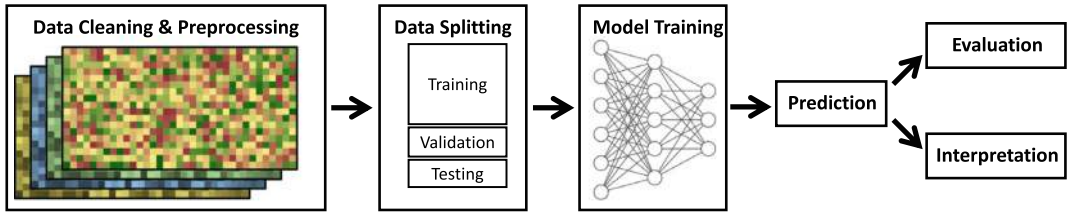


Fig. 1 The typical computational pipeline for the prediction of response to monotherapies or combinational therapies

we developed TG-LASSO [10] and TINDL [13] to predict the response of cancer tumors (test set) based on models completely trained on in vitro CCLs. In such an application, various biological and technical differences may exist between the train and test set (*see* [10, 13] for a discussion), which would require models that can generalize to out of distribution datasets.

While the answers to the questions above affect problem formulation, the overall computational pipeline typically includes data cleaning and preprocessing, data splitting, model training, prediction, evaluation, and interpretation (Fig. 1). In what follows, we will explain various aspects of these steps.

2.2 Model Formulation

One of the main factors influencing the model formulation is the type of variable to be predicted. In the context of DRP, when the drug response is presented as area under the dose-response curve (AUC), half-maximal inhibitory concentration (IC50), or similar measures, regression models are used to predict these continuous values. On the other hand, when the aforementioned measures are binarized or categorical representations such as Response Evaluation Criteria in Solid Tumours (RECIST) [15] are used, a classification model is utilized. In the context of DSP, most commonly used measures of synergism, which rely on Loewe additivity [16], Bliss independence [17], and zero interaction potency (ZIP) [18], are continuous values and can be predicted directly using a regression model, or their binarized versions can be predicted using a classifier.

Typically, there are three main frameworks used to formulate the DRP (or DSP) problem, depicted in Fig. 2. The first approach is the single-task learning (STL) (Fig. 2a) framework, in which a separate model is trained for each drug (or drug-pair). The input to each model is molecular and clinical features of samples (such as CCLs or tumors), but drug-specific features such as chemical structures are not used. For example, TG-LASSO [10], a computational model that we developed for DRP of cancer tumors based on a model trained on CCLs, falls under this category (even though it is based on traditional ML). Similarly, TINDL [13], a DL model

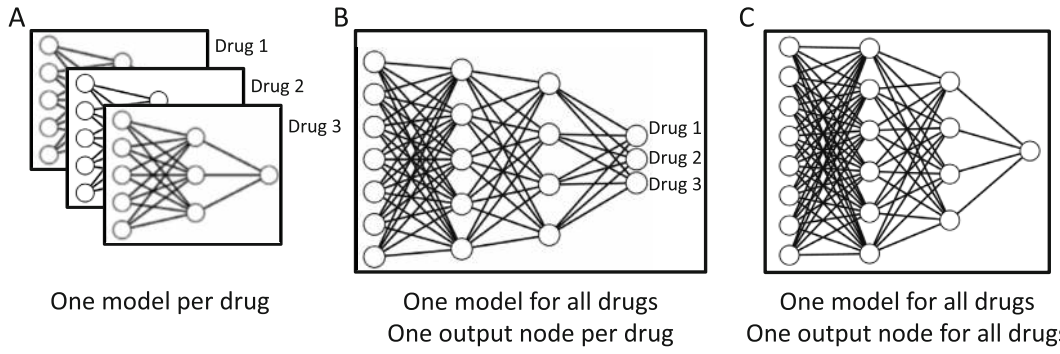


Fig. 2 The three main frameworks used for problem formulation. Neural networks are used to present these frameworks, but traditional machine learning (ML) or statistical models can also be categorized in a similar manner

with specialized tissue-informed normalization for the same task, also falls in this category. On one hand, this framework directly uses the knowledge of the drug identity, since the model is drug-specific; however, when the number of drugs is large, this approach will be computationally expensive. Moreover, information is not shared across different drugs. Since drugs with similar targets or mechanisms of action could be informative about each other, approaches that jointly model the response of multiple drugs (Fig. 2b, c) can be beneficial.

In the frameworks depicted in Fig. 2b, c, a single model is trained for all drugs, and information about the similarities or differences of drugs is shared. In the approach depicted in Fig. 2b, a multitask learning (MTL) framework is used, and each output node corresponds to a different drug. This approach shares parameters and information across drugs, but the similarity and differences of the drugs must be learned by the model during training without relying on drug-specific features. Similar to the previous framework, the input to the model are only the sample features. As such, it is still not possible to predict the response of drugs that do not exist in the training set.

In the third framework (Fig. 2c), both the sample and drug features are used as input, and a single model is trained to predict the response to any of the drugs in any of the samples. Generally, this framework is more flexible as it does not limit the model's generalizability to drugs that are already in the training set. However, the quality and choice of representations for samples and drugs play a larger part in the model's predictive capacity. Depending on the priorities of the user, this flexibility can be traded off for superior performance on the main task. As an example, BiG-DRP [11], which is a deep learning model based on heterogenous graph convolutional networks (HGCN) for DRP, uses this framework. In this model, CCLs are represented using their baseline gene

expression profile (the mRNA abundance of their genes), and drugs are represented based on their chemical structure (drug descriptors [19] or Morgan fingerprints [20]). Additionally, a drug-CCL bipartite graph is formed in which each drug is connected to the 1% of the training CCLs that are most sensitive to it (using "sensitive" edges) and to the 1% of the CCLs that are most resistant to it (using "resistant" edges). This graph enables information-sharing across all CCLs and drugs. A two-layer HGCN is used to integrate different representations to obtain better drug embeddings to improve DRP performance.

The DSP problem can also be formulated using the same three frameworks, but with some modifications. The main difference is that instead of predicting the response to a single drug, synergism of two or multiple drugs are predicted. As a result, in the first framework, one model per drug combination should be trained, and in the second framework, a different output node for each drug combination is required. For large numbers of unique drugs, the number of all possible combinations (even pairs) grows rapidly, resulting in the first two frameworks becoming computationally infeasible. The third framework, however, can be used in which features of multiple drugs need to be provided as input. We have developed MARSY [14] to predict the synergism of different drug-pairs in CCLs using the third framework (Fig. 2c). In this model, to enable learning better representations, we also defined two auxiliary tasks in a MTL framework such that the model also predicted response of the CCLs to each individual drug. In this model, CCLs were represented using their baseline gene expression profile, but the drugs were represented using the gene expression profile changes they would induce in two cancer cell lines. We observed that in this task, this drug representation improves performance compared to using the chemical structure of the drugs.

It is important to note that different variations of the three frameworks above can also be used. For example, in the first framework, one could train one model per sample by treating different drugs as training instances. Or in the second framework, one could have one output node for each CCL, instead of each drug. As a result, the advantages of information-sharing as well as the limitations in generalizability would be shifted from drugs to samples (since the role of the two are swapped).

2.3 Model Inputs

Various data modalities have been used for the DRP and DSP tasks. For training purposes, the models require a measure of drug response or drug synergism (mentioned earlier). Additionally, various features can be used as input to the model to represent samples and drugs. Molecular "omics" data are one category of features widely used to represent samples. These can include protein abundance, gene expression, methylation profile, mutations, copy number alternations, and other data modalities. Arguably, gene

expression profiles (nowadays mostly from RNA-seq) are the most commonly used set of features and have been shown to be most predictive of drug response in previous studies [21, 22]. One of the challenges of using multi-omics data is the excessive number of features that results in the "big-p, little-n" problem. More specifically, adding extra data modalities increases the number of features without increasing the number of training instances. Consequently, the computational model will be more prone to overfitting, negatively affecting its generalizability and performance. As a result, we recommend to start model training using one data modality and then add additional data modalities carefully while controlling and assessing overfitting using the validation set. Additionally, it is possible that these extra modalities are not available for some of the samples. While it may be tempting to drop samples with missing modalities or employ some imputation strategies, the risk of overfitting (when dropping a significant number of samples) and propagating error (when imputing) should be carefully assessed. In spite of these considerations, using multiple data modalities can improve prediction performance [12].

To represent different drugs (e.g., in the third framework in Fig. 2c), different drug features can be used. Drug targets [23], Morgan fingerprints [20], drug descriptors [19], change in gene expression profile of cell lines treated with the drug [14, 24], or other representations can be used to represent drugs from different views. In a recent study, we benchmarked four state-of-the-art interpretable DL models for DRP using different input features [12]. Comparing the performance of these models when drug targets or Morgan fingerprints (MFP) were used to represent drugs, we observed that MFPs were more informative for predicting the drug response of unseen CCLs. Similarly, we used MFPs and drug descriptors in BiG-DRP [11] and observed that using either of them provides good comparable performances (Table 3 of [11]). In the context of DSP, we used these chemical representations, but also the gene expression changes of MCF7 and PC3 CCLs in response to each drug (from the LINCS dataset [24]) in MARSY [14]. Our results showed that LINCS signatures resulted in better synergy prediction, which may suggest the use of this data modality as an alternative to chemical structure features (Table 5 of [14]). The use of "raw" chemical structures, represented by graphs [25], is also getting traction due to the increasing popularity of graph neural networks (GNNs). Graph-based representations provide additional depth to data that is otherwise lost when using vector-formatted drug features. For a comprehensive comparison of different chemical structure-based representation, we refer to [26]. Although GNNs seem to perform well, users have to take into account the additional nuisances of training neural networks, such as stochasticity and sample sizes.

Other information such as protein-protein interaction (PPI) networks or signaling pathways can also be useful in the context of DRP or DSP. Several studies have used signaling pathways to make DL models for DRP interpretable [27–30]. While this information can improve interpretability, our analyses [12] showed that they may not be able to improve the prediction performance compared to baselines. Similarly, PPI or other biological networks can be used for network-guided gene prioritization [9] or gene set characterization [31], but the capability of these networks in improving prediction performance is yet unclear. Given the increasingly large experimental PPI studies [32] as well as computational models that can accurately predict such networks [33], network information may prove to be a significant contributor in not only improving interpretability but also the prediction performance of computational models in this field.

2.4 Preprocessing

The first step of the computational pipeline (Fig. 1) involves data cleaning and preprocessing. In this section, we will describe steps that are applicable to different data modalities, instead of focusing on the details of one particular data type.

2.4.1 Data Cleaning

Depending on the quality of data, number of missing values, pattern of missing values, presence of outliers, etc., different data cleaning steps may be necessary. One of the main considerations is how to deal with missing values in the input feature dataset. To deal with missing values, two strategies are typically used. One is to impute missing values. There are various methods for imputation, and many of them work based on the principle that missing values can be estimated based on the values of other samples or other features. However, when the number of missing values is large, it is recommended to exclude samples or features with many missing values, since the imputation accuracy deteriorates as the number of missing values increases. When the choice is between removing a feature and a training instance, it is usually recommended to remove features, since reducing the number of training instances can negatively affect prediction performance and generalizability of the model. Other methods for removing features could include removing low-variance features or selecting a set of features based on prior information.

Labels should also be inspected in advance. For instances where drug response/synergy (labels) has duplicates, users should first check the range of concentration. Unless the concentration range is used to normalize the response/synergy, it is recommended that all samples should be in the same concentration ranges for a specific drug(pair); otherwise, the model may learn unwanted and unexpected relationships. Choose only one of the concentration ranges if there are multiple; otherwise, it is also common to average the labels.

2.4.2 Normalization

Most databases would provide FPKM or TPM data for gene expression. Although considered as normalized, the values of these can still be very large. This can lead to extremely low/high magnitude parameter values and gradients, which are not ideal for learning. We recommend using log2(TPM + 1) or log2(FPKM + 1) for gene expression data. Furthermore, we also recommend z-score normalization per gene, so that the distributions of features are centered around 0 and have a standard deviation of 1, which is more efficient for deep learning (also *see* **Note 1**). However, normalization is not necessary if the feature is binary, as in the case of mutation and MFPs.

In addition to the normalization of features, normalization of drug responses is also recommended in regression tasks, especially in multitask frameworks (framework 2 and 3) (also *see* **Note 2**). Specifically, you can z-score the drug responses prior to training and save the summary statistics (mean and standard deviation) so that you can use them to scale the predictions back to the original distribution. This is because some drugs have distinct response signatures (e.g., very high/low IC50 for all samples, low variance). Normalizing the responses prevents the models from focusing on the obvious biases of each drug and encourages the model to learn more meaningful differences between samples.

2.4.3 Feature Selection and Dimensionality Reduction

One advantage of deep learning is its capacity to learn features of high dimensional data without the need for feature selection. However, feature selection and dimensionality reduction can still improve deep learning models in efficiency and performance. For continuous-valued features, low-variance features can be dropped as they are less likely to be associated by the model with the output. In other applications of machine learning, features that highly correlate with each other are typically dropped or combined into one feature. However, it is unclear whether this should be the case for gene expression data because genes are not independent. As a result, highly correlated genes can still be kept, or users can rely on their intuition based on their prior knowledge.

Dimensionality reduction can be performed as part of preprocessing. Principal component analysis (PCA) should be used with care because PCA can only be applied for transductive tasks. Different types of autoencoders can also be useful, especially if you have an abundance of unlabeled samples. In a typical scenario, the autoencoder should only be trained using the training samples. In some cases, it is inevitable that test samples should be used in training the [auto]encoder. Researchers should carefully define the scope of their model to prevent misinterpretations.

2.4.4 Data Homogenization

One recurring theme in DRP is the idea of preclinical-to-clinical (P2C) DRP. In this scenario, the model is trained using preclinical (in vitro) datasets, but with the end-goal of predicting for clinical samples typically from a different dataset [10, 13]. It is important to address statistical discrepancies and homogenize data when multiple datasets are involved in the study. This can be performed as part of data preprocessing, training, post hoc analysis, or a combination of these. This step can typically be skipped when working with a single dataset, except when starting with raw data where the data has been gathered in multiple batches. One simple visual test to check whether such homogenization is necessary is to plot the data in two dimensions using PCA or other dimensionality reduction methods. When there is an obvious separation of points between the batches/datasets, then the data should be homogenized.

Data homogenization can be done during the preprocessing using batch effect removal tools. Although different datasets have much more distinction than different batches, multiple studies [10, 34] have shown batch effect removal tools such as ComBat [35] can still be effective in homogenizing gene expression data from GDSC (cancer cell lines) and TCGA (tumors) in the context of DRP.

In deep learning, we refer to these different datasets as different "domains". Domain adaptation methods such as DANN [36] and ADDA [37] allow the model to generalize across different domains using adversarial learning. These methods typically have two adversarial components, the encoder and the discriminator. Encoders are used to encode the inputs into embeddings in a common latent space across datasets, while discriminator classifies the original domain from the encoder's output. The goal of the encoder is to confuse the discriminator by making sure that the domain-specific artifacts are removed. In theory, once the discriminator can no longer tell the samples apart, then the embeddings are already "homogenized" across the dataset. However, it should be noted that unlike computer vision tasks, domain adaptation for biological data is quite difficult to assess. In our previous study [13], we found that a simple tissue-informed normalization could suffice in removing such discrepancies in preclinical-to-clinical DRP.

2.5 Data Splitting and Its Implications for the Application of the Model

One of the most important considerations is data splitting for training, hyperparameter tuning, and evaluation [12, 38]. As a general rule, the training set is used to learn model parameters, the validation set is used to tune the hyperparameters, and the test set is used to evaluate the performance and generalizability. As a result, the performance of the model on completely held-out test data is the main measure of performance, since both the training and validation sets have been used during training, and the performance on those sets will be inflated and overoptimistic.

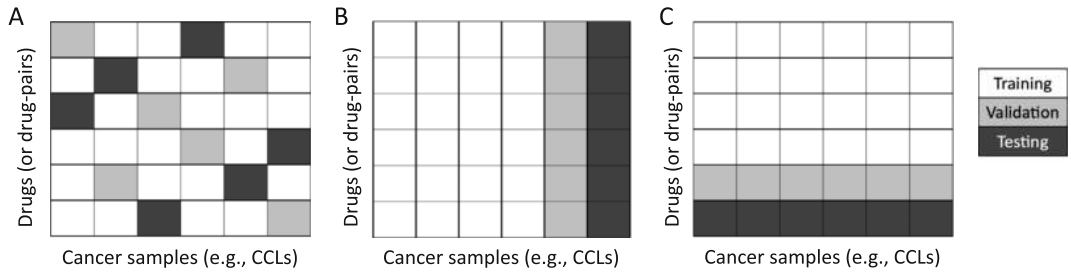


Fig. 3 Three data splitting strategies for DRP and DSP tasks

In addition to the general rule above, data splitting strategy has significant implications on the application and usability of a computational model for DRP or DSP (*see* **Note 3**). The most common approach for splitting the data into train/validation/test is by randomly selecting (sample, drug) pairs in DRP or (sample, drug-pair) triples in DSP (Fig. 3a). As a result of this strategy, each sample or each drug (or drug-pair) in the test set is present in the training set, but not simultaneously together. For example, if (CCL1, drug1) is in the test set, the training set may include (CCL1, drug2) and (CCL2, drug1). Such a model is not recommended in making predictions for completely unseen samples or completely unseen drugs because there is a high risk that the model has been overfitted on the seen drugs and samples. However, this model is useful to impute missing values or to obtain biological insights about the effect of drugs in different cancer types. In the context of DSP in CCLs, even if the synergy score of drug-pairs is of interest, the set of all possible triples of (CCL, drug1, drug2) is extremely large. Since experimentally measuring the synergy score of all such triples is costly and labor intensive, even the large databases of synergy scores such as DrugComb contain many missing values. In such cases, a model that is trained to impute these missing values can be quite useful. In [14], we developed MARSY to address this issue. We showed that MARSY can accurately predict the drug synergy scores (ZIP and S_{mean}) both in the leave-triple-out (described above) and in the more challenging leave-pair-out setup, in which a drug-pair in the test set is never seen in the training set. In addition to imputation, models trained using this framework can be used to characterize drugs' mechanisms of action and biomarkers of drug sensitivity as novel drug targets.

Another data splitting strategy is leave-cancer-sample-out (or cancer-sample-blind). In this strategy, any sample in the test set must not exist in the training or validation sets (Fig. 3b). This strategy is particularly useful to train computational models for precision medicine, in which the model should make predictions for a new sample. We commonly use this data splitting strategy (e.g., in TG-LASSO [10], TINDL [13], and BiG-DRP [11]) due to its usefulness for precision medicine applications. However, to

train models capable of accurately making predictions on unseen samples, a large number of training samples are necessary. As a result, this strategy is not applicable when only a few (e.g., tens of) samples are available. Another data splitting strategy is leave-drug (or drug-pair)-out, which is also known as drug-blind in DRP or drug-pair-blind in DSP (Fig. 3c). This strategy is similar to leave-cancer-sample-out, with the difference that now drugs (or drug-pairs) in the test set must not be present in the training or validation set. This strategy is particularly useful for drug discovery. In [14], we showed that MARSY can accurately predict synergy scores in a leave-drug-pair-out setup.

Depending on the application of a computational model, one should carefully choose the data splitting strategy for training and for evaluation. We recently benchmarked four interpretable DL models for DRP using all three data splitting strategies [12] and showed that accurate predictions in one strategy does not necessarily translate into accurate predictions in another.

2.6 Baseline Models and Ablation Study

One of the main steps in choosing the best computational model or in evaluating the performance of a novel predictor is benchmarking against alternative solutions. There are several categories of baselines that in our opinion are essential to achieve these goals, which we have used in previous studies [10–14]. The first category are baselines that we call naïve predictors [12]. These are simple models that make predictions without learning the relationship between inputs and outputs. The main goal of these baselines is to characterize potential inflation of performance metrics and provide a lower bound that any useful predictor should surpass. As we have discussed in our prior work [12], one such predictor works as follows. Assuming a random (Fig. 3a) or a leave-cancer-sample-out (Fig. 3b) split, for each drug (or drug-pair) the model reports the mean (or median) response (or synergy score) of that drug (drug-pair) across all samples in the training set. Alternatively, for a random or a leave-drug or (drug-pair)-out split, for each sample the model reports the mean (or median) response of that sample to all drugs (or drug-pairs) in its training set. Alternative variations of this model can also be designed that take into account drug families or tissue types in prediction. Naive predictors are one major category of baselines that are typically missing from publications in this domain, which may contribute to the misinterpretation of the quality of predictions. For example, the DRP methods that we benchmarked in [12] showed higher performance metrics in a leave-CCL-out framework compared to a leave-drug-out framework; however, when those metrics were calibrated using the corresponding naive predictors, their performances in these setups were not too different. This is due to specific biases that may exist in the dataset, which are not always easy to detect in advance. For example, when using IC50, typically the naïve predictor performs

well, since different drugs have vastly different IC50 values, and the knowledge of the drug identity by itself is quite informative about the response (also *see* **Note 4**).

Ablated and alternative versions of a model are also a very important category of baselines that must be assessed. Such baselines not only reveal which component of the model contributes most to its performance but also ensure that a simpler model cannot provide better or comparable results. For example, in BiG-DRP [11], we observed that when both Morgan fingerprints and drug descriptors are used simultaneously (requiring a more complex model with more parameters), the performance was slightly better than using each drug feature alone (Table 3 of [11]). However, we opted for drug descriptors, since the performance improvement was not large enough to justify the additional parameters that increase the computational complexity and training time. Including alternative versions of the model that have a comparable number of parameters while changing the architecture is also quite useful. When there are many training instances, a model with more parameters can perform better. To disentangle whether an improved performance stems from an architecture modification or simply the number of parameters, one needs to control for this. For example, in [14], we evaluated various versions of MARSY (Table 3 of [14]) to fully assess the trade-off between type of encoder and number of parameters.

In addition to the aforementioned baselines, state-of-the-art models designed for the same (or a similar) task as well as traditional machine learning algorithms (e.g., random forests, support vector machines, elastic net) should be included. For example, when assessing four interpretable DL models for DRP [12], we observed that a similar prediction performance could be achieved using a random forest or a simple fully connected neural network.

As a final note, it is recommended that all baseline models be trained and tested on the exact same training/validation/test sets used for the original model. In addition, cross-validation can ensure that a specific choice of data splitting is not the reason behind performance variations across models. Finally, the hyperparameter of baseline models should be tuned on the validation set or why such an approach has not been taken clearly discussed.

2.7 Model Interpretation

An important question in cancer precision medicine is the identification of molecular markers (e.g., genes) that are predictive of response to treatments. Such markers can reveal drug-cancer dependencies, can characterize drugs' mechanisms of action, and can identify novel drug targets to overcome drug resistance. The association between the molecular features of samples (e.g., gene mRNA expression) and drug response (or drug-pair synergy score) is one way to identify such markers. Simple correlation analysis or methods that additionally incorporate known interactions among

molecular features can be used for this purpose. In [9], we developed ProGENI, a method based on random walks with restart (RWR) for gene prioritization while incorporating known protein-protein interactions (PPIs). In this method, first a small number of genes are identified based on the correlation of their expression across many samples to the response to a specific drug. These genes are then used as the restart set in an RWR on a PPI network to obtain a ranked list of genes associated with the drug of interest. This approach enables the integration of PPIs with drug-gene dependencies to obtain a more informative list of genes. Our study (including wet-lab gene knockdown experiments) allowed us to identify genes whose mRNA expression significantly influences the response of a CCL to a drug, many of which were not identifiable using correlation analysis (or other methods) alone.

An alternative to directly assessing the association between molecular features and drug response is to utilize machine learning models for DRP and DSP. Achieving this is related to the concept of interpretable machine learning [39, 40]. Many traditional ML methods (e.g., LASSO) perform feature selection and prediction simultaneously, enabling the identification of a list of features. For DL models, methods that quantify feature attribution scores [41, 42] can be used to identify features that contribute most to the prediction performance (e.g., see TINDL [13] and BiG-DRP [11] for examples of how these methods can be used in DRP and *see* **Note 5** for some considerations).

Irrespective of how a list of top molecular features are obtained, one typically needs to characterize their functional role. When top genes or proteins are identified, gene ontology (GO) and pathway enrichment analysis are widely used to this end. We have found the gene set characterization (GSC) pipeline of KnowEnG (an online computational platform) [31] to be particularly powerful due to the wide variety of options it provides. In addition to supporting standard Fisher's exact test analysis with a wide range of GO and pathway collections, it supports a network-guided mode of operation. This mode is an implementation of an algorithm called DRaWR [43], which utilizes discriminative random walks to incorporate information from gene-level networks in this task, providing a richer and more comprehensive view of involved pathways.

As a final note, we would like to point out that in spite of the availability of the methods discussed here and other methods, the field of interpretable DL for DRP and DSP is a lively field of study with many recent innovations. Recent advances in this field rely on developing DL architectures that are motivated by prior biological knowledge [44] such as pathway information or PPIs. These innovations coupled with large databases of experimentally characterized biological networks and computational models that can predict them [33, 45] can have a significant impact in this field.

3 Notes

1. When performing transformations that require statistics about the data, consider calculating the statistics only using the training set to prevent data leakage. For example, when applying z-score normalization to the features, calculate the mean and standard deviation using only the training set. However, when the entities in the test set and training set are the same, there is no need for such consideration. For example, since drugs can simultaneously be in the training and test sets in leave-cancer-sample-out splitting, drug features can be normalized using all the drugs.
2. Normalizing the drug responses is not recommended for tasks that rely on drug rankings per sample because the normalization changes the order of the drugs. Normalizing drug responses is also not necessary if there are no obvious biases in the responses.
3. When benchmarking models, data splits should be fixed and consistent across different models to prevent unintended advantages/disadvantages for some models.
4. The naïve predictor is only one way to recalibrate our perception of evaluations. Another way is to calculate these metrics per drug (e.g., Pearson's correlation per drug) to remove the main source of these biases. These numbers may appear less impressive, but this highlights the difficulty of the task.
5. When interpreting the model, feature contributions can end up being inconsistent across identical models with different initializations due to possible feature correlations and stochasticity of training neural networks. For this reason, it is recommended to repeat the pipeline (training and explaining) multiple times with different initializations.

References

1. Cancer Genome Atlas Research N, Weinstein JN, Collisson EA, Mills GB, Shaw KR, Ozenberger BA, Ellrott K, Shmulevich I, Sander C, Stuart JM (2013) The Cancer Genome Atlas Pan-Cancer analysis project. Nat Genet 45(10):1113–1120. https://doi.org/10.1038/ng.2764
2. Barretina J, Caponigro G, Stransky N, Venkatesan K, Margolin AA, Kim S, Wilson CJ, Lehar J, Kryukov GV, Sonkin D, Reddy A, Liu M, Murray L, Berger MF, Monahan JE, Morais P, Meltzer J, Korejwa A, Jane-Valbuena J, Mapa FA, Thibault J, Bric-Furlong E, Raman P, Shipway A, Engels IH, Cheng J, Yu GK, Yu J, Aspesi P Jr, de Silva M, Jagtap K, Jones MD, Wang L, Hatton C, Palescandolo E, Gupta S, Mahan S, Sougnez C, Onofrio RC, Liefeld T, MacConaill L, Winckler W, Reich M, Li N, Mesirov JP, Gabriel SB, Getz G, Ardlie K, Chan V, Myer VE, Weber BL, Porter J, Warmuth M, Finan P, Harris JL, Meyerson M, Golub TR, Morrissey MP, Sellers WR, Schlegel R, Garraway LA (2012) The Cancer Cell Line Encyclopedia enables predictive modelling of anticancer drug sensitivity. Nature 483(7391):603–607. https://doi.org/10.1038/nature11003

3. Rees MG, Seashore-Ludlow B, Cheah JH, Adams DJ, Price EV, Gill S, Javaid S, Coletti ME, Jones VL, Bodycombe NE, Soule CK, Alexander B, Li A, Montgomery P, Kotz JD, Hon CS, Munoz B, Liefeld T, Dancik V, Haber DA, Clish CB, Bittker JA, Palmer M, Wagner BK, Clemons PA, Shamji AF, Schreiber SL (2016) Correlating chemical sensitivity and basal gene expression reveals mechanism of action. Nat Chem Biol 12(2):109–116. https://doi.org/10.1038/nchembio.1986
4. Ghandi M, Huang FW, Jane-Valbuena J, Kryukov GV, Lo CC, McDonald ER 3rd, Barretina J, Gelfand ET, Bielski CM, Li H, Hu K, Andreev-Drakhlin AY, Kim J, Hess JM, Haas BJ, Aguet F, Weir BA, Rothberg MV, Paolella BR, Lawrence MS, Akbani R, Lu Y, Tiv HL, Gokhale PC, de Weck A, Mansour AA, Oh C, Shih J, Hadi K, Rosen Y, Bistline J, Venkatesan K, Reddy A, Sonkin D, Liu M, Lehar J, Korn JM, Porter DA, Jones MD, Golji J, Caponigro G, Taylor JE, Dunning CM, Creech AL, Warren AC, McFarland JM, Zamanighomi M, Kauffmann A, Stransky N, Imielinski M, Maruvka YE, Cherniack AD, Tsherniak A, Vazquez F, Jaffe JD, Lane AA, Weinstock DM, Johannessen CM, Morrissey MP, Stegmeier F, Schlegel R, Hahn WC, Getz G, Mills GB, Boehm JS, Golub TR, Garraway LA, Sellers WR (2019) Next-generation characterization of the Cancer Cell Line Encyclopedia. Nature 569(7757):503–508. https://doi.org/10.1038/s41586-019-1186-3
5. Yang W, Soares J, Greninger P, Edelman EJ, Lightfoot H, Forbes S, Bindal N, Beare D, Smith JA, Thompson IR, Ramaswamy S, Futreal PA, Haber DA, Stratton MR, Benes C, McDermott U, Garnett MJ (2013) Genomics of Drug Sensitivity in Cancer (GDSC): a resource for therapeutic biomarker discovery in cancer cells. Nucleic Acids Res 41 (Database issue):D955–D961. https://doi.org/10.1093/nar/gks1111
6. Zagidullin B, Aldahdooh J, Zheng S, Wang W, Wang Y, Saad J, Malyutina A, Jafari M, Tanoli Z, Pessia A, Tang J (2019) DrugComb: an integrative cancer drug combination data portal. Nucleic Acids Res 47(W1):W43–W51. https://doi.org/10.1093/nar/gkz337
7. Liu H, Zhang W, Zou B, Wang J, Deng Y, Deng L (2020) DrugCombDB: a comprehensive database of drug combinations toward the discovery of combinatorial therapy. Nucleic Acids Res 48(D1):D871–D881. https://doi.org/10.1093/nar/gkz1007
8. LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521(7553):436–444. https://doi.org/10.1038/nature14539
9. Emad A, Cairns J, Kalari KR, Wang L, Sinha S (2017) Knowledge-guided gene prioritization reveals new insights into the mechanisms of chemoresistance. Genome Biol 18(1):153. https://doi.org/10.1186/s13059-017-1282-3
10. Huang EW, Bhope A, Lim J, Sinha S, Emad A (2020) Tissue-guided LASSO for prediction of clinical drug response using preclinical samples. PLoS Comput Biol 16(1):e1007607. https://doi.org/10.1371/journal.pcbi.1007607
11. Hostallero DE, Li Y, Emad A (2022) Looking at the BiG picture: incorporating bipartite graphs in drug response prediction. Bioinformatics 38(14):3609–3620. https://doi.org/10.1093/bioinformatics/btac383
12. Li Y, Hostallero DE, Emad A (2023) Interpretable deep learning architectures for improving drug response prediction performance: myth or reality? Bioinformatics 39(6). https://doi.org/10.1093/bioinformatics/btad390
13. Hostallero DE, Wei L, Wang L, Cairns J, Emad A (2023) Preclinical-to-clinical anti-cancer drug response prediction and biomarker identification using TINDL. Genomics Proteomics Bioinformatics 21:535. https://doi.org/10.1016/j.gpb.2023.01.006
14. El Khili MR, Memon SA, Emad A (2023) MARSY: a multitask deep-learning framework for prediction of drug combination synergy scores. Bioinformatics 39(4). https://doi.org/10.1093/bioinformatics/btad177
15. Eisenhauer EA, Therasse P, Bogaerts J, Schwartz LH, Sargent D, Ford R, Dancey J, Arbuck S, Gwyther S, Mooney M, Rubinstein L, Shankar L, Dodd L, Kaplan R, Lacombe D, Verweij J (2009) New response evaluation criteria in solid tumours: revised RECIST guideline (version 1.1). Eur J Cancer 45(2):228–247. https://doi.org/10.1016/j.ejca.2008.10.026
16. Loewe S (1953) The problem of synergism and antagonism of combined drugs. Arzneimittelforschung 3(6):285–290
17. Liu Q, Yin X, Languino LR, Altieri DC (2018) Evaluation of drug combination effect using a Bliss independence dose-response surface model. Stat Biopharm Res 10(2):112–122. https://doi.org/10.1080/19466315.2018.1437071
18. Yadav B, Wennerberg K, Aittokallio T, Tang J (2015) Searching for drug synergy in complex

dose-response landscapes using an interaction potency model. Comput Struct Biotechnol J 13:504–513. https://doi.org/10.1016/j.csbj.2015.09.001

19. Landrum G (2013) RDKit: a software suite for cheminformatics, computational chemistry, and predictive modeling. Greg Landrum, 8:1–31
20. Rogers D, Hahn M (2010) Extended-connectivity fingerprints. J Chem Inf Model 50(5):742–754. https://doi.org/10.1021/ci100050t
21. Jang IS, Neto EC, Guinney J, Friend SH, Margolin AA (2014) Systematic assessment of analytical methods for drug sensitivity prediction from cancer cell line data. Pac Symp Biocomput:63–74
22. Costello JC, Heiser LM, Georgii E, Gonen M, Menden MP, Wang NJ, Bansal M, Ammad-uddin M, Hintsanen P, Khan SA, Mpindi JP, Kallioniemi O, Honkela A, Aittokallio T, Wennerberg K, Community ND, Collins JJ, Gallahan D, Singer D, Saez-Rodriguez J, Kaski S, Gray JW, Stolovitzky G (2014) A community effort to assess and improve drug sensitivity prediction algorithms. Nat Biotechnol 32(12):1202–1212. https://doi.org/10.1038/nbt.2877
23. Szklarczyk D, Santos A, von Mering C, Jensen LJ, Bork P, Kuhn M (2016) STITCH 5: augmenting protein-chemical interaction networks with tissue and affinity data. Nucleic Acids Res 44(D1):D380–D384. https://doi.org/10.1093/nar/gkv1277
24. Subramanian A, Narayan R, Corsello SM, Peck DD, Natoli TE, Lu X, Gould J, Davis JF, Tubelli AA, Asiedu JK, Lahr DL, Hirschman JE, Liu Z, Donahue M, Julian B, Khan M, Wadden D, Smith IC, Lam D, Liberzon A, Toder C, Bagul M, Orzechowski M, Enache OM, Piccioni F, Johnson SA, Lyons NJ, Berger AH, Shamji AF, Brooks AN, Vrcic A, Flynn C, Rosains J, Takeda DY, Hu R, Davison D, Lamb J, Ardlie K, Hogstrom L, Greenside P, Gray NS, Clemons PA, Silver S, Wu X, Zhao WN, Read-Button W, Wu X, Haggarty SJ, Ronco LV, Boehm JS, Schreiber SL, Doench JG, Bittker JA, Root DE, Wong B, Golub TR (2017) A next generation connectivity map: L1000 platform and the first 1,000,000 profiles. Cell 171(6):1437–1452. e1417. https://doi.org/10.1016/j.cell.2017.10.049
25. Liu Q, Hu Z, Jiang R, Zhou M (2020) DeepCDR: a hybrid graph convolutional network for predicting cancer drug response. Bioinformatics 36(Suppl_2):i911–i918. https://doi.org/10.1093/bioinformatics/btaa822
26. Zagidullin B, Wang Z, Guan Y, Pitkanen E, Tang J (2021) Comparative analysis of molecular fingerprints in prediction of drug combination effects. Brief Bioinform 22(6). https://doi.org/10.1093/bib/bbab291
27. Deng L, Cai Y, Zhang W, Yang W, Gao B, Liu H (2020) Pathway-guided deep neural network toward interpretable and predictive modeling of drug sensitivity. J Chem Inf Model 60(10):4497–4505. https://doi.org/10.1021/acs.jcim.0c00331
28. Jin I, Nam H (2021) HiDRA: hierarchical network for drug response prediction with attention. J Chem Inf Model 61(8):3858–3867. https://doi.org/10.1021/acs.jcim.1c00706
29. Tang YC, Gottlieb A (2021) Explainable drug sensitivity prediction through cancer pathway enrichment. Sci Rep 11(1):3128. https://doi.org/10.1038/s41598-021-82612-7
30. Zhang H, Chen Y, Li F (2021) Predicting anticancer drug response with deep learning constrained by signaling pathways. Front Bioinform 1:639349. https://doi.org/10.3389/fbinf.2021.639349
31. Blatti C 3rd, Emad A, Berry MJ, Gatzke L, Epstein M, Lanier D, Rizal P, Ge J, Liao X, Sobh O, Lambert M, Post CS, Xiao J, Groves P, Epstein AT, Chen X, Srinivasan S, Lehnert E, Kalari KR, Wang L, Weinshilboum RM, Song JS, Jongeneel CV, Han J, Ravaioli U, Sobh N, Bushell CB, Sinha S (2020) Knowledge-guided analysis of "omics" data using the KnowEnG cloud platform. PLoS Biol 18(1):e3000583. https://doi.org/10.1371/journal.pbio.3000583
32. Luck K, Sheynkman GM, Zhang I, Vidal M (2017) Proteome-scale human interactomics. Trends Biochem Sci 42(5):342–354. https://doi.org/10.1016/j.tibs.2017.02.006
33. Szymborski J, Emad A (2022) RAPPPID: towards generalizable protein interaction prediction with AWD-LSTM twin networks. Bioinformatics 38(16):3958–3967. https://doi.org/10.1093/bioinformatics/btac429
34. Geeleher P, Cox NJ, Huang RS (2014) Clinical drug response can be predicted using baseline gene expression levels and in vitro drug sensitivity in cell lines. Genome Biol 15(3):R47. https://doi.org/10.1186/gb-2014-15-3-r47
35. Johnson WE, Li C, Rabinovic A (2007) Adjusting batch effects in microarray expression data using empirical Bayes methods. Biostatistics 8(1):118–127
36. Ganin Y, Lempitsky V (2015) Unsupervised domain adaptation by backpropagation. In: International conference on machine learning (ICML), pp 1180–1189, Lille, France

37. Tzeng E, Hoffman J, Saenko K, Darrell T (2017) Adversarial discriminative domain adaptation. In: IEEE computer vision and pattern recognition (CVPR), pp 7167–7176, IEEE, Honolulu, HI, USA
38. Tabe-Bordbar S, Emad A, Zhao SD, Sinha S (2018) A closer look at cross-validation for assessing the accuracy of gene regulatory networks and models. Sci Rep 8(1):6620. https://doi.org/10.1038/s41598-018-24937-4
39. Malioutov DM, Varshney KR, Emad A, Dash S (2017) Learning interpretable classification rules with boolean compressed sensing. In: Transparent data mining for big and small data, pp 95–121, Springer
40. Arrieta AB, Díaz-Rodríguez N, Del Ser J, Bennetot A, Tabik S, Barbado A, García S, Gil-López S, Molina D, Benjamins R (2020) Explainable Artificial Intelligence (XAI): concepts, taxonomies, opportunities and challenges toward responsible AI. Inf Fusion 58: 82–115
41. Schwab P, Karlen W (2019) CXPlain: causal explanations for model interpretation under uncertainty. In: Neural information processing systems (NeurIPS), Vancouver, BC, Canada
42. Lundberg SM, Lee S-I (2017) A unified approach to interpreting model predictions. In: Advances in neural information processing systems. Curran Associates Inc, Red Hook, p 30
43. Blatti C, Sinha S (2016) Characterizing gene sets using discriminative random walks with restart on heterogeneous biological networks. Bioinformatics 32(14):2167–2175. https://doi.org/10.1093/bioinformatics/btw151
44. Kuenzi BM, Park J, Fong SH, Sanchez KS, Lee J, Kreisberg JF, Ma J, Ideker T (2020) Predicting drug response and synergy using a deep learning model of human cancer cells. Cancer Cell 38(5):672–684.e676. https://doi.org/10.1016/j.ccell.2020.09.014
45. Emad A, Sinha S (2021) Inference of phenotype-relevant transcriptional regulatory networks elucidates cancer type-specific regulatory mechanisms in a pan-cancer study. NPJ Syst Biol Appl 7(1):9. https://doi.org/10.1038/s41540-021-00169-7

Chapter 16

Identification of Somatic Variants in Cancer Genomes from Tissue and Liquid Biopsy Samples

Kiran Krishnamachari, Hanaé Carrié, and Anders Jacobsen Skanderup

Abstract

Somatic variant detection is an important step in the analysis of cancer genomes for basic research as well as precision oncology. Here, we review existing computational methods for identifying somatic mutations from tissue as well as liquid biopsy samples. We then describe steps to run VarNet (Krishnamachari et al., Nat Commun 13:4248, 2022), a variant caller using deep learning, to accurately identify single nucleotide variants (SNVs) and short insertion-deletion (indels) mutations from next-generation sequencing (NGS) of tumor tissue samples.

Key words Somatic variant calling, Cancer genomics, Mutations, Next-generation sequencing

1 Introduction

1.1 Overview of Somatic Variant Calling

Acquired (somatic) DNA mutations and genetic instability play a significant role in tumorigenesis [2]. Somatic variants are caused by errors in the machinery for DNA replication and repair, which can be promoted by aberrant growth or environmental factors such as exposure to carcinogens.

Somatic variant detection is the computational problem of identifying acquired genetic mutations in cancer genomes. This is most commonly achieved by analyzing sequencing data obtained from matched normal and tumor samples of a given patient. Somatic variants can be categorized as single nucleotide variants (SNVs), short insertion-deletions (indels), and structural variants (SVs). In this chapter, we focus on the detection of SNVs and indels only.

Somatic variant detection is commonly performed on tumor tissue biopsies obtained from cancer patients. Somatic variant calling on tumor samples is confounded both by biological variation

Kiran Krishnamachari and Hanaé Carrié contributed equally with all other contributors.

Alexander Krasnitz and Pascal Belleau (eds.), *Cancer Bioinformatics*, Methods in Molecular Biology, vol. 2932,
https://doi.org/10.1007/978-1-0716-4566-6_16,

(e.g., tumor heterogeneity) and technical noise (e.g., sequencing errors). This process is further complicated when somatic variant calling must be performed on tumor samples that have been preserved using formalin fixation and paraffin embedding (FFPE), which is the predominant method to store and preserve clinical tissue samples worldwide. The FFPE process causes DNA damage that can cause variant callers to produce a large number of false positive mutation calls [3, 4].

Somatic variant detection is also valuable in the analysis of cell-free DNA (cfDNA) from blood liquid biopsy samples in precision oncology. cfDNA refers to degraded DNA fragments released into the plasma through apoptosis or necrosis or via passive or active secretion from various tissues in the body. In cancer patients, a portion of cfDNA, termed circulating tumor DNA (ctDNA), originates from tumor cells [5]. Tracking ctDNA mutations represents a promising approach to minimal residual disease (MRD) detection [6], treatment response monitoring [7], or targeted therapy attribution [8]. Liquid biopsy offers accessible, minimally invasive, and repeatable detection of cancer mutations in contrast to tissue biopsy; nevertheless, it is accompanied by unique challenges.

1.2 Somatic Variant Calling in Tumor Tissue Samples

Somatic variant callers have traditionally used statistical or probabilistic models of variant alleles in tumor samples in combination with multiple heuristic filters to remove false positives. Strelka2 [9] and Mutect2 [10] are two such popular methods. Recently, machine learning has proven to be an effective approach that leverages the many cancer genomes publicly available today. SMuRF [11] uses a random forest model trained using features from an ensemble of somatic mutation callers. Strelka2 [9] augments its probabilistic variant model using machine learning to predict an aggregate confidence score for each candidate variant. VarNet [1] and NeuSomatic [12] both use convolutional deep learning models to predict a probability of mutation. While NeuSomatic was trained on synthetic data from the DREAM challenge [13], VarNet was trained on real cancer genomes using weak supervision. AIVariant [14] is another deep learning based caller recently proposed for highly contaminated tumor samples (*see* Table 1 for an overview).

1.3 Somatic Variant Calling in Formalin-Fixed Paraffin-Embedded (FFPE) Tumor Samples

Fresh-frozen tumor tissue samples are preferred for basic research due to their lower levels of DNA degradation. In contrast, FFPE tumor tissue samples are the most common starting point for translational research and clinical diagnostics. However, the FFPE preservation process introduces significant DNA degradation and damage that results in misread bases during NGS. This makes the accurate determination of somatic mutations from FFPE tumor samples significantly more challenging compared to fresh-frozen (FF) tumor samples [3, 4]. FFPE artifacts tend to occur at low-allele frequencies, and the most prominent of these are

Table 1
Overview of somatic variant callers and different approaches used in the literature

Caller	Approach	Publication
Varscan2	Fisher's exact test	Koboldt et al.) [41]
Freebayes	Bayesian inference	Garrison and Marth [42]
Vardict	Fisher's exact test	Lai et al. [43]
Strelka2	Bayesian inference	Kim et al. [9]
Mutect2	Bayesian inference	Benjamin et al. [10]
SMuRF	Ensemble (random forest)	Huang et al. [11]
NeuSomatic	Deep learning	Sahraeian et al. [12]
VarNet	Deep learning	Krishnamachari et al. [1]
AIVariant	Deep learning	Jeon et al. [14]

C > T/G > A changes due to the hydrolytic deamination of cytosines [15]. FFPE artifacts especially confound the detection of true low-allele-frequency somatic mutations that could be of clinical relevance. Hence, simply filtering low-allele-frequency mutation calls may mislead downstream clinical analysis.

Recent work has attempted to improve the quality of FFPE variant call-sets by performing post hoc removal of artifacts from the outputs of popular variant callers (*see* **Note 5**). For example, IdeaFix [16] filters likely mutation artifacts from the output of Mutect2 using a decision tree-based approach exploiting multiple features such as read pair orientation bias, genomic context, and variant allele frequency. IdeaFix annotates C > T/G > A calls made by Mutect2 as either true variants or artifacts. FIREVAT [17] removes sequencing artifacts from variant call-sets using known mutational and error signatures. SOBDetector [18] proposed a method to filter artifacts from the output of any mutation caller using the *strand orientation bias* feature. FFPolish [19] proposed a logistic regression model of multiple features including variant allele frequency and variant read-quality metrics to filter artifacts from the outputs of mutation callers. Other work has also proposed using mutation calls made by *any two callers* on FFPE tumor samples as a simple baseline to reduce artifacts [20]. This strategy however would not exclude artifacts that are misclassified as mutations by more than one caller.

1.4 Somatic Variant Calling Approaches in Liquid Biopsy Samples

Calling cancer mutations from cfDNA is challenging due to typically low ctDNA fractions, leading to cancer mutations presenting at low variant allele frequencies (VAFs). While obtaining comprehensive tumor profiles from cfDNA could yield valuable biological

information, it entails significant technical complexity, as subclonal and metastatic mutations may manifest at even lower VAFs. Moreover, these mutations can be mistaken for clonal hematopoiesis of indeterminate potential (CHIP) variants from noncancerous blood cell subpopulations [21, 22]. The degraded nature of cfDNA amplifies sequencing errors, PCR artifacts, and mapping errors, all falling within the same VAF range as cancer mutations, complicating their distinction from noise. Moreover, nonrandom cfDNA fragmentation, attributable to nucleosome wrapping, results in uneven sequencing coverage compared to tissue biopsy, posing both a technical challenge and an informative feature that could be exploited for variant calling. Although increased sequencing depth enhances detection sensitivity, the limited input of molecules imposes a maximum informative coverage depth. For instance, a typical 5 mL blood draw yields on average 2 mL of plasma containing 10 ng of DNA, corresponding to 3000 diploid genome equivalents [23].

At ultrahigh sequencing depths, the utilization of unique molecular identifiers (UMIs) is recommended for consensus correction of PCR and sequencing errors. Furthermore, in silico error correction methods, such as iDES [24] and DREAMS [25], contribute to enhanced accuracy. The iDES method employs a locus-level error model derived from healthy cfDNA samples, eliminating noise loci through statistical tests. On the other hand, DREAMS constructs a machine learning read-level error model trained on error-free and sequencing error reads from filtered post-surgery low ctDNA samples. It considers read-level and local sequence-context features, albeit necessitating retraining for changes in laboratory protocols.

The tissue-specific callers in Table 1 can be applied to cfDNA with a reduced (e.g., $\frac{1}{\text{coverage}}$) or disabled VAF threshold (*see* **Note 7**). Table 2 gives an overview of existing cfDNA-specific callers. ABEMUS [26] models VAF distributions using a binomial model and employs global and per-base error filters derived from

Table 2
Overview of cfDNA somatic variant callers

Caller	Approach	Publication
SiNVICT	Poisson model	Kockan et al. [28]
ABEMUS	Binomial model	Casiraghi et al. [26]
cfSNV	Error suppression using overlapping read mate Joined genotype model Iterative search of clusters Adjusted site-level post-filtration Machine learning (random forest) read level post-filtration	Li et al. [27]

normal buffy coat samples. cfSNV [27] utilizes overlapping read mates to suppress errors, applies a joined genotype model on mutation clusters iteratively to address tumor heterogeneity, and performs adjusted site-level and machine learning-based read level post-filtration. SiNVICT [28], a plasma-only approach, incorporates local assembly, read realignment, and a statistical Poisson model. However, the lack of annotated real cfDNA datasets, akin to the ICGC benchmark dataset for tumor tissue [29], currently hinders comparative evaluation of the accuracy of these specialized cfDNA callers relative to tissue-based callers (Table 1) when applied to cfDNA samples.

Clinical ultra-deep targeted cfDNA-only assays (>1000x, targeting 10–200 genes) have been receiving US Food and Drug Administration (FDA) approval since 2020 as Companion Diagnostic tests (CDx) for detecting actionable mutations in confirmed cancer patients, facilitating targeted therapy stratification. The SEQC2 benchmark [30] compared five industry-leading ctDNA assays, emphasizing challenges in accurately detecting variants below 0.5% variant allele frequency (VAF).

Innovative minimal residual disease (MRD) detection tools employed for early relapse detection after surgery or treatment, such as INVAR [31] and MRDetect/MRD-Edge [32, 33], are built on ctDNA mutation detection. These methods harness cumulative signals from rare, error-corrected mutated ctDNA reads found in hundreds to thousands of somatic mutations detected in 30x WGS or 200x WES plasma samples, thereby overcoming the limitations in ctDNA abundance associated with targeted assays. INVAR implements error correction through overlapping read pairs and trinucleotide context-based filtering. MRDetect employs a support vector machine classifier, while MRD-Edge utilizes a convolutional neural network and a multilayer perceptron for read-level error correction. However, these methods rely on prior tissue biopsy, limiting their applicability for cancer screening. Recent proof-of-concept studies propose cfDNA-only cancer detection methods, such as Bae et al.'s [34] and Pointy [35], based on de novo mutation discovery from low-coverage WGS plasma samples, combined with other molecular characteristics. Enhancements, such as including a matched buffy coat sample for efficient germline subtraction (*see* **Note 8**), increasing sequencing depth, and validating on additional external datasets, are necessary to achieve significant performance improvements in future studies.

2 Materials

Here we describe steps to run VarNet, a deep learning based somatic variant caller that has demonstrated superior accuracy compared to other methods on real tumor tissue samples, including low tumor-purity settings (*see* **Note 1**) [1].

2.1 Environment

VarNet is implemented as a Python package. The source code is available on GitHub (https://github.com/skandlab/VarNet). To get started, download the latest release of VarNet from GitHub (https://github.com/skandlab/VarNet/releases). The dependencies to run VarNet can be installed using the pip package manager.

```
pip install -r requirement.txt
```

Alternatively, download the Docker image to run VarNet:

```
docker pull kiranchari/varnet:latest
```

VarNet relies on Tensorflow-cpu [36], Pysam (https://github.com/pysam-developers/pysam), NumPy [37], Pandas [38], Pybedtools [39], and Joblib (https://github.com/joblib/joblib). These dependencies will be automatically installed by the above command or can be found preinstalled in the docker image.

2.2 Input Requirements

VarNet performs somatic variant calling using matched normal and tumor genomes input as binary alignment map (BAM) files (.bam files) along with the reference genome (.fa file) that was used to perform alignment of the genomes. BAM files can be generated from sequencing data (.fastq files) using a pipeline such as the bcbio-nextgen (https://bcbio-nextgen.readthedocs.io) workflow. The recommended configuration to run bcbio-nextgen is listed in Table 3 (*see* **Note 4**).

3 Methods

3.1 Running VarNet

Minimally, VarNet requires the following parameters to run: (1) paths to normal and tumor BAM files, (2) path to reference genome file (*see* **Note 3**), and (3) an output directory. Optionally, the number of processes to use a BED (Browser Extensible Data) file containing a list of genomic regions to limit variant calling to and the run *mode* (SNV or indel calling) can be specified.

Table 3
Recommended bcbio-nextgen parameters

bcbio-nextgen parameter	Value
mark_duplicates	True
Recalibrate	False
Realign (GATK4)	False

First, the filtering step must be run to scan the genome to identify mutation candidates. An example command to perform filtering is as follows:

```
python filter.py \
 --sample_name dream1 \
 --normal_bam /path/to/dream1_normal.bam \
 --tumor_bam /path/to/dream1_tumor.bam \
 --processes 6 \
 --output_dir /path/to/varnet_outputs \
 --reference /path/to/GRCh38.fa
```

In the above filtering step, VarNet uses heuristic filters to exclude genomic sites with very low or no evidence for a variant in the tumor sample. By default, VarNet scans the genome for both SNV and indel candidates. Subsequently in the prediction step, VarNet generates input image encodings for each candidate variant (Fig. 1; *see* **Note 2**). The deep learning model is applied on the fly for each generated input image encoding. The prediction step can be run as follows:

```
python predict.py \
 --sample_name dream1 \
 --normal_bam /path/to/dream1_normal.bam \
 --tumor_bam /path/to/dream1_tumor.bam \
 --processes 6 \
 --output_dir /path/to/varnet_outputs \
 --reference /path/to/GRCh38.fa
```

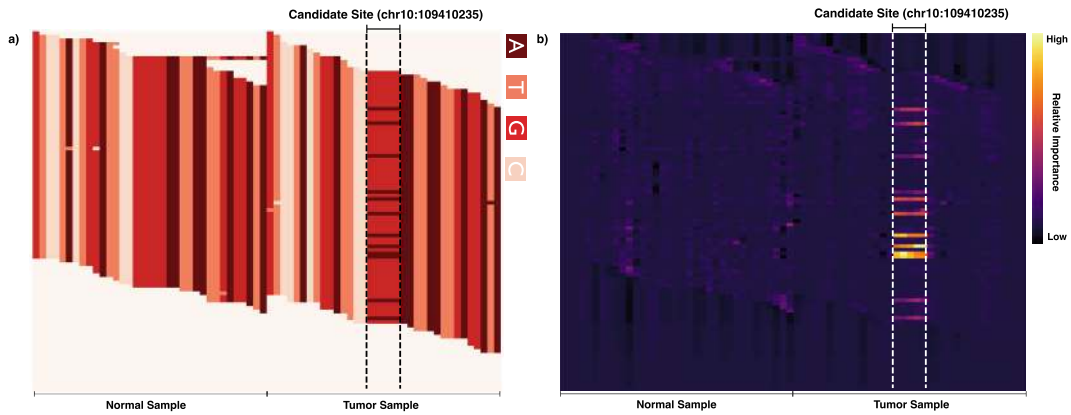


Fig. 1 (**a**) VarNet encoding (base channel) of an SNV on chromosome 10 in the medulloblastoma (MBL) sample. The candidate position is repeated 5× in both the normal and tumor image. Variant alleles are visible at the candidate site in the tumor sample image. (**b**) Heatmap visualization showing "pixels" in the base channel most important to VarNet's deep learning model. VarNet has identified variant alleles at the candidate site in the tumor. Pixel-wise importance scores were computed using guided backpropagation [40]

Note that VarNet does not require a GPU (graphics processing unit) to perform prediction. Upon successful completion of the prediction step, VarNet outputs a standard VCF (variant call format) file in the root of the sample's output directory. VarNet includes metadata for each variant in its output including a probability SCORE. The probability SCORE is the confidence given by VarNet's deep learning model that the locus contains a somatic mutation.

3.2 Performance on ICGC Benchmark Samples

We benchmarked VarNet on the International Cancer Genome Consortium (ICGC) Gold Set comprising manually verified somatic mutations in chronic lymphocytic leukemia (CLL) and medulloblastoma (MBL) tumor-normal pairs [29]. VarNet made calls at higher accuracy compared to existing state-of-the-art callers (Fig. 2).

3.3 Tuning the Performance of VarNet

VarNet uses a default SCORE threshold of 0.5 to classify variants as PASS or REJECT. However, this threshold can be varied to extract higher recall (sensitivity) or precision from VarNet. Decreasing the threshold would increase recall at the expense of precision, and vice versa (*see* **Note 6**).

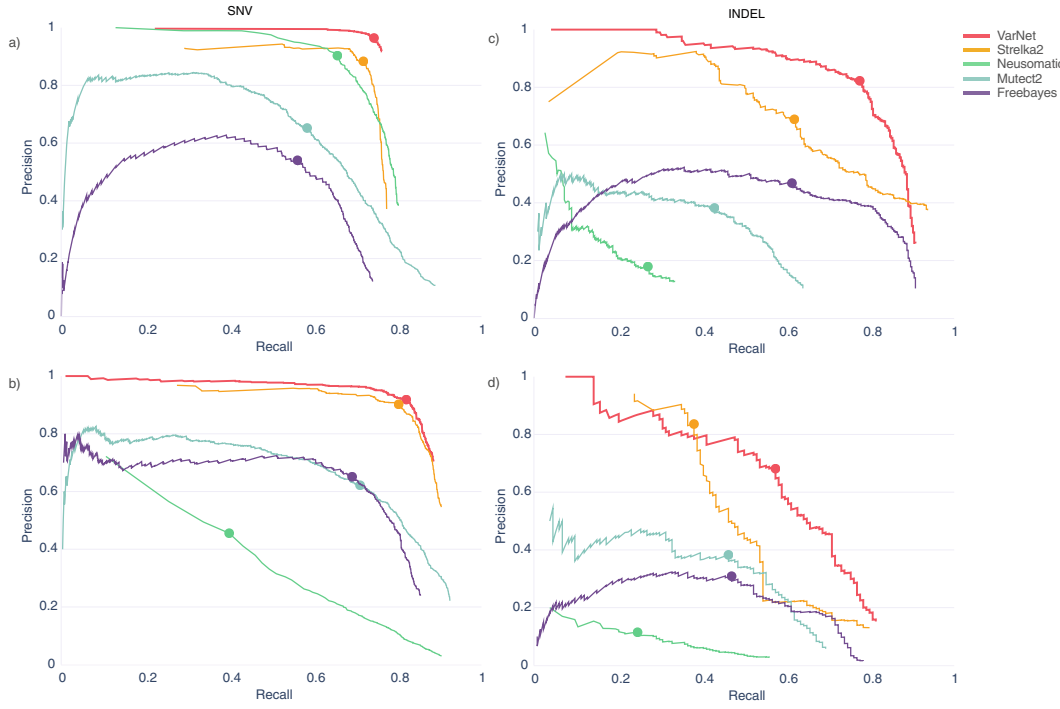


Fig. 2 (**a**, **b**) Precision/recall curves for SNV calling in the MBL and CLL samples, respectively. (**c**, **d**) Precision/recall curves for indel calling in the MBL and CLL samples, respectively. Solid circles indicate the highest F1 accuracy score obtained by each method

Minimum allele frequency thresholds: VarNet has a sensitivity threshold of 3.5% for the AF of variants it can detect. Variants below 3.5% are excluded in the filtering step. Hence, VarNet is not suitable for detecting very low VAF variants such as those in liquid biopsy samples.

4 Notes

1. VarNet is a deep learning based computational method to detect somatic variants from tumor tissue samples.
2. VarNet generates "images" of reads overlapping each candidate mutation in both the tumor and matched normal sample.
3. When running VarNet, ensure that the same reference genome used for aligning reads is also employed during variant calling.
4. Mark and remove duplicate reads in your sequencing data prior to somatic variant calling. This step can be done by libraries such as bcbio-nextgen.
5. Somatic variant callers can generate many false positive mutation calls on FFPE tumor samples. Post hoc variant filtering methods have been proposed to alleviate this problem.
6. Modifying the prediction threshold of callers can be important to achieve high accuracy, especially in FFPE tumor samples.
7. When using tissue-specific callers on liquid biopsy samples, lower (e.g., $\frac{1}{\text{coverage}}$) or disable the default VAF threshold.
8. Clinical ultra-deep targeted liquid biopsy assays are currently limited to cfDNA-only tests. For intermediate depth to deep WES/WGS, it is recommended to sequence the matched buffy coat sample for efficient germline subtraction.

References

1. Krishnamachari K, Lu D, Swift-Scott A et al (2022) Accurate somatic variant detection using weakly supervised deep learning. Nat Commun 13:4248. https://doi.org/10.1038/s41467-022-31765-8
2. Jassim A, Rahrmann EP, Simons BD, Gilbertson RJ (2023) Cancers make their own luck: theories of cancer origins. Nat Rev Cancer 23:710–724. https://doi.org/10.1038/s41568-023-00602-5
3. Oh E, Choi Y-L, Kwon MJ et al (2015) Comparison of accuracy of whole-exome sequencing with formalin-fixed paraffin-embedded and fresh frozen tissue samples. PLoS One 10:1–13. https://doi.org/10.1371/journal.pone.0144162
4. Srinivasan M, Sedmak D, Jewell S (2002) Effect of fixatives and tissue processing on the content and integrity of nucleic acids. Am J Pathol 161:1961–1971. https://doi.org/10.1016/S0002-9440(10)64472-0
5. Heitzer E, Haque IS, Roberts CES, Speicher MR (2019) Current and future perspectives of liquid biopsies in genomics-driven oncology. Nat Rev Genet 20:71–88. https://doi.org/10.1038/s41576-018-0071-5
6. Salvianti F, Gelmini S, Mancini I et al (2021) Circulating tumour cells and cell-free DNA as a prognostic factor in metastatic colorectal cancer: the OMITERC prospective study. Br J Cancer 125:94–100. https://doi.org/10.1038/s41416-021-01399-6

7. Abbosh C, Frankell AM, Harrison T et al (2023) Tracking early lung cancer metastatic dissemination in TRACERx using ctDNA. Nature 616:553–562. https://doi.org/10.1038/s41586-023-05776-4
8. Rolfo C, Mack PC, Scagliotti GV et al (2018) Liquid biopsy for advanced non-small cell lung cancer (NSCLC): a statement paper from the IASLC. J Thorac Oncol 13:1248–1268. https://doi.org/10.1016/j.jtho.2018.05.030
9. Kim S, Scheffler K, Halpern AL et al (2018) Strelka2: fast and accurate calling of germline and somatic variants. Nat Methods 15:591–594. https://doi.org/10.1038/s41592-018-0051-x
10. Benjamin D, Sato T, Cibulskis K et al (2019) Calling somatic SNVs and Indels with Mutect2. bioRxiv. https://doi.org/10.1101/861054
11. Huang W, Guo YA, Muthukumar K et al (2019) SMuRF: portable and accurate ensemble prediction of somatic mutations. Bioinformatics (Oxford, England). https://doi.org/10.1093/bioinformatics/btz018
12. Sahraeian SME, Liu R, Lau B et al (2019) Deep convolutional neural networks for accurate somatic mutation detection. Nat Commun 10. https://doi.org/10.1038/s41467-019-09027-x
13. Ewing AD, Houlahan KE, Hu Y et al (2015) Combining tumor genome simulation with crowdsourcing to benchmark somatic single-nucleotide-variant detection. Nat Methods 12:623–630. https://doi.org/10.1038/nmeth.3407
14. Jeon H, Ahn J, Na B et al (2023) AIVariant: a deep learning-based somatic variant detector for highly contaminated tumor samples. Exp Mol Med 55:1734–1742. https://doi.org/10.1038/s12276-023-01049-2
15. Steiert TA, Parra G, Gut M et al (2023) A critical spotlight on the paradigms of FFPE-DNA sequencing. Nucleic Acids Res: gkad519. https://doi.org/10.1093/nar/gkad519
16. Tellaetxe-Abete M, Calvo B, Lawrie C (2021) Ideafix: a decision tree-based method for the refinement of variants in FFPE DNA sequencing data. NAR Genom Bioinform 3:lqab092. https://doi.org/10.1093/nargab/lqab092
17. Kim H, Lee AJ, Lee J et al (2019) FIREVAT: finding reliable variants without artifacts in human cancer samples using etiologically relevant mutational signatures. Genome Med 11:81. https://doi.org/10.1186/s13073-019-0695-x
18. Diossy M, Sztupinszki Z, Krzystanek M et al (2021) Strand orientation bias detector to determine the probability of FFPE sequencing artifacts. Brief Bioinform 22. https://doi.org/10.1093/bib/bbab186
19. Dodani DD, Nguyen MH, Morin RD et al (2022) Combinatorial and machine learning approaches for improved somatic variant calling from formalin-fixed paraffin-embedded genome sequence data. Front Genet 13. https://doi.org/10.3389/fgene.2022.834764
20. de Schaetzen van Brienen L, Larmuseau M, Van der Eecken K et al (2020) Comparative analysis of somatic variant calling on matched FF and FFPE WGS samples. BMC Med Genet 13:94. https://doi.org/10.1186/s12920-020-00746-5
21. Chan HT, Chin YM, Nakamura Y, Low S-K (2020) Clonal hematopoiesis in liquid biopsy: from biological noise to valuable clinical implications. Cancers (Basel) 12:2277. https://doi.org/10.3390/cancers12082277
22. Jaiswal S, Ebert BL (2019) Clonal hematopoiesis in human aging and disease. Science 366:eaan4673. https://doi.org/10.1126/science.aan4673
23. Song P, Wu LR, Yan YH et al (2022) Limitations and opportunities of technologies for the analysis of cell-free DNA in cancer diagnostics. Nat Biomed Eng 6:232–245. https://doi.org/10.1038/s41551-021-00837-3
24. Newman AM, Lovejoy AF, Klass DM et al (2016) Integrated digital error suppression for improved detection of circulating tumor DNA. Nat Biotechnol 34:547–555. https://doi.org/10.1038/nbt.3520
25. Christensen MH, Drue SO, Rasmussen MH et al (2023) DREAMS: deep read-level error model for sequencing data applied to low-frequency variant calling and circulating tumor DNA detection. Genome Biol 24:99. https://doi.org/10.1186/s13059-023-02920-1
26. Casiraghi N, Orlando F, Ciani Y et al (2020) ABEMUS: platform-specific and data-informed detection of somatic SNVs in cfDNA. Bioinformatics 36:2665–2674. https://doi.org/10.1093/bioinformatics/btaa016
27. Li S, Noor ZS, Zeng W et al (2021) Sensitive detection of tumor mutations from blood and its application to immunotherapy prognosis. Nat Commun 12:4172. https://doi.org/10.1038/s41467-021-24457-2
28. Kockan C, Hach F, Sarrafi I et al (2017) SiNVICT: ultra-sensitive detection of single

nucleotide variants and indels in circulating tumour DNA. Bioinformatics 33:26–34. https://doi.org/10.1093/bioinformatics/btw536

29. Alioto TS, Buchhalter I, Derdak S et al (2015) A comprehensive assessment of somatic mutation detection in cancer using whole-genome sequencing. Nat Commun 6. https://doi.org/10.1038/ncomms10001
30. Deveson IW, Gong B, Lai K et al (2021) Evaluating the analytical validity of circulating tumor DNA sequencing assays for precision oncology. Nat Biotechnol 39:1115–1128. https://doi.org/10.1038/s41587-021-00857-z
31. Wan JCM, Heider K, Gale D et al (2020) ctDNA monitoring using patient-specific sequencing and integration of variant reads. Sci Transl Med 12:eaaz8084. https://doi.org/10.1126/scitranslmed.aaz8084
32. Zviran A, Schulman RC, Shah M et al (2020) Genome-wide cell-free DNA mutational integration enables ultra-sensitive cancer monitoring. Nat Med 26:1114–1124. https://doi.org/10.1038/s41591-020-0915-3
33. Widman AJ, Shah M, Øgaard N et al (2022) Machine learning guided signal enrichment for ultrasensitive plasma tumor burden monitoring. 2022.01.17.476508
34. Bae M, Kim G, Lee T-R et al (2023) Integrative modeling of tumor genomes and epigenomes for enhanced cancer diagnosis by cell-free DNA. Nat Commun 14:1–15
35. Wan JCM, Stephens D, Luo L et al (2022) Genome-wide mutational signatures in low-coverage whole genome sequencing of cell-free DNA. Nat Commun 13:4953. https://doi.org/10.1038/s41467-022-32598-1
36. Abadi M, Barham P, Chen J, et al (2016) TensorFlow: A system for large-scale machine learning. In: 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16). pp 265–283
37. Harris CR, Millman KJ, van der Walt SJ et al (2020) Array programming with NumPy. Nature 585:357–362. https://doi.org/10.1038/s41586-020-2649-2
38. Reback J, McKinney W, Jbrockmendel et al (2020) pandas-dev/pandas: Pandas 1.1.1
39. Dale RK, Pedersen BS, Quinlan AR (2011) Pybedtools: a flexible Python library for manipulating genomic datasets and annotations. Bioinformatics 27:3423–3424. https://doi.org/10.1093/bioinformatics/btr539
40. Springenberg JT, Dosovitskiy A, Brox T, Riedmiller MA (2015) Striving for simplicity: the all convolutional net. In: Bengio Y, LeCun Y (eds) 3rd international conference on learning representations, ICLR 2015, San Diego, CA, USA, 7–9 May 2015, Workshop Track Proceedings
41. Koboldt DC, Zhang Q, Larson DE, et al (2012) VarScan 2: somatic mutation and copy number alteration discovery in cancer by exome sequencing. Genome research 22:568–576
42. Garrison E, Marth G (2012) Haplotype-based variant detection from short-read sequencing
43. Lai Z, Markovets A, Ahdesmaki M, Johnson J (2015) VarDict: A novel and versatile variant caller for next-generation sequencing in cancer research. Cancer Research 75:4864 LP – 4864. https://doi.org/10.1158/1538-7445.AM2015-4864

Chapter 17

SUMMER: A Practical Tool for Identifying Factors and Biomarkers Associated with Pan-cancer Survival

Junyi Xin, Silu Chen, Huiqin Li, Mulong Du, and Meilin Wang

Abstract

The application of Mendelian randomization (MR) analytical framework based on genome-wide association study (GWAS) datasets has uncovered hundreds of risk factors involving disease development that included tumorigenesis, but the practice of MR in cancer survival remains limited. Here, we will provide abundant details of our previously established tool, *SU*rvival related cancer *M*ulti-omics database via *ME*ndelian *R*andomization (SUMMER; http://njmu-edu.cn:3838/SUMMER/), which would help users systematically evaluate the causal effects of risk factors and circulating biomarkers on pan-cancer survival.

Key words Cancer survival, Mendelian randomization, Risk factors, Biomarkers

1 Introduction

Genome-wide association study (GWAS), as a common genetic epidemiology method based on comparing the allele frequency of genetic variants between affected cases and unaffected controls, has uncovered thousands of genetic loci involved in susceptibility to disease, especially for cancer [1–3]. Notably, with the increasing number of GWAS loci, Mendelian randomization (MR), an integrative "post-GWAS" approach [4], provides a way to explore the potential risk factors or biomarkers related to the development of cancer [5]. Briefly, MR is a well-known causal inference method that uses single nucleotide polymorphisms (SNPs) as instrumental variables (IVs, i.e., genetic predictors), to assess the causal association between exposures (e.g., BMI and smoking) and outcomes (e.g., cancer risk) [6–8]. For example, our previous MR findings

Junyi Xin, Silu Chen and Huiqin Li contributed equally with all other contributors.

Alexander Krasnitz and Pascal Belleau (eds.), *Cancer Bioinformatics*, Methods in Molecular Biology, vol. 2932,
https://doi.org/10.1007/978-1-0716-4566-6_17,

indicated plausible noncausal associations between circulating vitamin E and the risk of ten common cancers [9].

It is noteworthy that cancer also ranks as a leading cause of death [10], with an estimated 10.0 million cancer deaths that occurred in 2020 worldwide, indicating that cancer death remains an important barrier to life expectancy [11]. Accurate estimation of survival probability can provide valuable insights into the precision therapy of cancer patients [12, 13]. Currently, MR has not been widely applied to identify the risk factors and biomarkers associated with cancer survival.

To address this gap, we have constructed an online pan-cancer survival database, SUMMER (*SU*rvival related cancer *M*ulti-omics database via *ME*ndelian *R*andomization; http://njmu-edu.cn:3838/SUMMER/), that includes (i) available survival GWAS summary statistics of 17 cancer types from the UK Biobank cohort (Table 1), followed by (ii) causal risk factors and biomarkers involving cancer survival obtained via MR analysis, to help users query, browse, and download results [14].

Totally, across 17 cancer types, SUMMER identifies a total of 1209 cancer overall survival (OS)-associated and 1539 cancer-specific survival (CSS)-associated SNPs at a suggestive genome-wide significance threshold ($P \leq 1 \times 10^{-6}$), as well as an average of 11 phenotypes, 716 genes, and 4828 CpG sites associated with cancer OS and an average of 11 phenotypes, 705 genes, and 4702 CpG sites associated with cancer CSS (Table 2).

In this chapter, we will provide details on the construction of SUMMER database and instructions on its usage and notes, to help users systematically evaluate causal effects of risk factors and circulating biomarkers on pan-cancer survival.

2 Materials

The construction of SUMMER database was conducted in a two-stage design (Fig. 1) as follows:

(i) *Construction of pan-cancer survival GWAS datasets*: to systematically evaluate the effects of genome-wide genetic variants on cancer survival that included OS and CSS, leveraging a total of 17 cancer types derived from the UK Biobank cohort.

(ii) *Integrative analysis to identify cancer prognostic risk factors and circulating biomarkers*: We evaluate effects causally related to risk factors and circulating biomarkers on cancer prognosis via a comprehensive MR approach that integrates pan-cancer survival GWAS datasets, with phenome-wide association study (PheWAS) and blood gene expression/DNA methylation quantitative trait loci (eQTL/meQTL) datasets.

Table 1
Basic characteristics of cancer cases in the UK Biobank cohort

Cancer type	Cases	Median follow-up Time (months)	Gender (%)		Age[a] (Mean ± SD)	BMI (Mean ± SD)	Death (%)	
			Male	Female			All-cause	Cancer-specific
Bladder cancer	526	49.63	426 (80.99)	100 (19.01)	67.11 ± 5.61	28.25 ± 4.34	170 (32.32)	113 (21.48)
Brain cancer	397	8.83	246 (61.96)	151 (38.04)	64.24 ± 7.04	27.65 ± 4.76	354 (89.17)	334 (84.13)
Breast cancer	4350	62.13	0 (0)	4350 (100)	61.84 ± 7.78	27.46 ± 5.10	319 (7.33)	233 (5.36)
Colorectal cancer	2621	48.57	1555 (59.33)	1066 (40.67)	65.25 ± 6.53	27.94 ± 4.59	779 (29.72)	569 (21.71)
Corpus uteri	698	57.58	0 (0)	698 (100)	64.17 ± 6.29	30.30 ± 6.95	105 (15.04)	78 (11.17)
Esophagus cancer	460	19.57	344 (74.78)	116 (25.22)	66.42 ± 5.81	28.63 ± 5.61	296 (64.35)	255 (55.43)
Gastric cancer	303	14.60	222 (73.27)	81 (26.73)	66.30 ± 6.63	28.68 ± 4.91	220 (72.61)	141 (46.53)
Lung cancer	1700	11.47	945 (55.59)	755 (44.41)	66.65 ± 5.99	27.46 ± 4.73	1287 (75.71)	1113 (65.47)
Lymphoid leukemia	350	51.90	209 (59.71)	141 (40.29)	65.42 ± 6.04	27.96 ± 5.11	58 (16.57)	26 (7.43)
Multiple myeloma	355	43.10	207 (58.31)	148 (41.69)	65.86 ± 6.78	27.79 ± 4.54	122 (34.37)	90 (25.35)
Oral and pharynx cancer	458	50.45	312 (68.12)	146 (31.88)	62.80 ± 6.94	27.27 ± 4.92	120 (26.2)	71 (15.5)
Ovarian cancer	437	40.33	0 (0)	437 (100)	63.65 ± 7.26	27.29 ± 4.86	201 (46)	177 (40.5)
Pancreatic cancer	506	5.35	274 (54.15)	232 (45.85)	66.27 ± 6.29	28.21 ± 5.02	460 (90.91)	422 (83.4)
Prostate cancer	4882	57.93	4882 (100)	0 (0)	66.77 ± 5.32	27.55 ± 3.83	460 (9.42)	258 (5.28)
Renal cancer	649	44.40	425 (65.49)	224 (34.51)	65.21 ± 6.38	29.18 ± 5.26	209 (32.2)	147 (22.65)
Skin melanoma	1402	56.27	717 (51.14)	685 (48.86)	63.39 ± 7.68	27.58 ± 4.48	119 (8.49)	79 (5.63)
Thyroid cancer	179	60.27	57 (31.84)	122 (68.16)	62.06 ± 7.56	27.66 ± 4.70	16 (8.94)	6 (3.35)

Note: *BMI* body mass index
[a]Age at diagnosis

Table 2
Summary of the significant associations of risk factors and circulating biomarkers with cancer survival via Mendelian randomization analysis

		Significant associations with overall survival						Significant associations with cancer-specific survival					
Cancer type	No. of SNPs	Lambda	SNPs[a]	Loci[a]	Phenotypes[b]	Genes[c]	CpG sites[c]	Lambda	SNPs[a]	Loci[a]	Phenotypes[b]	Genes[c]	CpG sites[c]
Bladder cancer	8,326,282	1.03	34	23	12	710	4692	1.03	134	38	13	737	5017
Brain cancer	8,390,743	1.12	108	39	4	847	5514	1.12	98	38	4	877	5574
Breast cancer	8,338,638	1.01	99	15	11	705	4681	1.01	29	12	14	677	4731
Colorectal cancer	8,334,629	1.01	7	5	5	692	4425	1.01	10	7	15	689	4301
Corpus uteri	8,360,934	1.01	161	35	7	629	4774	0.98	138	31	8	708	4700
Esophagus cancer	8,296,714	1.07	47	36	9	717	5034	1.07	36	31	7	729	5198
Gastric cancer	8,283,963	1.06	177	50	11	743	4953	1.07	270	54	12	716	5117
Lung cancer	8,355,227	1.01	24	4	12	677	4518	1.01	13	9	13	666	4678
Lymphoid leukemia	8,425,952	0.97	98	57	12	757	5208	0.84	271	36	15	792	5142
Multiple myeloma	8,258,091	1.07	73	39	12	680	4861	1.05	104	41	13	710	5041
Oral and pharynx cancer	8,272,464	1.05	99	30	9	762	4802	0.99	80	39	13	760	4819
Ovarian cancer	8,351,777	1.07	138	48	11	713	4978	1.06	90	46	6	671	4856
Pancreatic cancer	8,299,001	1.07	29	22	9	700	4746	1.06	86	26	10	678	4512
Prostate cancer	8,333,069	1.01	15	11	13	703	4559	1.00	29	9	18	719	4415
Renal cancer	8,376,852	1.03	32	17	11	754	5007	1.03	91	24	6	728	4712
Skin melanoma	8,339,443	0.99	55	22	14	668	4350	0.96	60	25	16	679	4635
Thyroid cancer	8,308,306	0.77	13	7	23	721	4977	0.37	0	0	12	451	2491

Note: *SNP* single nucleotide polymorphism, *IVW* inverse variance weighted, *SMR* summary-data-based Mendelian randomization
[a]*P*-value for Cox regression model $\leq 1 \times 10^{-6}$
[b]*P*-value for IVW analysis ≤ 0.05, *P*-value for egger intercept >0.05, and *P*-value for heterogeneity >0.05
[c]*P*-value for SMR analysis ≤ 0.05 and *P*-value for HEIDI >0.05

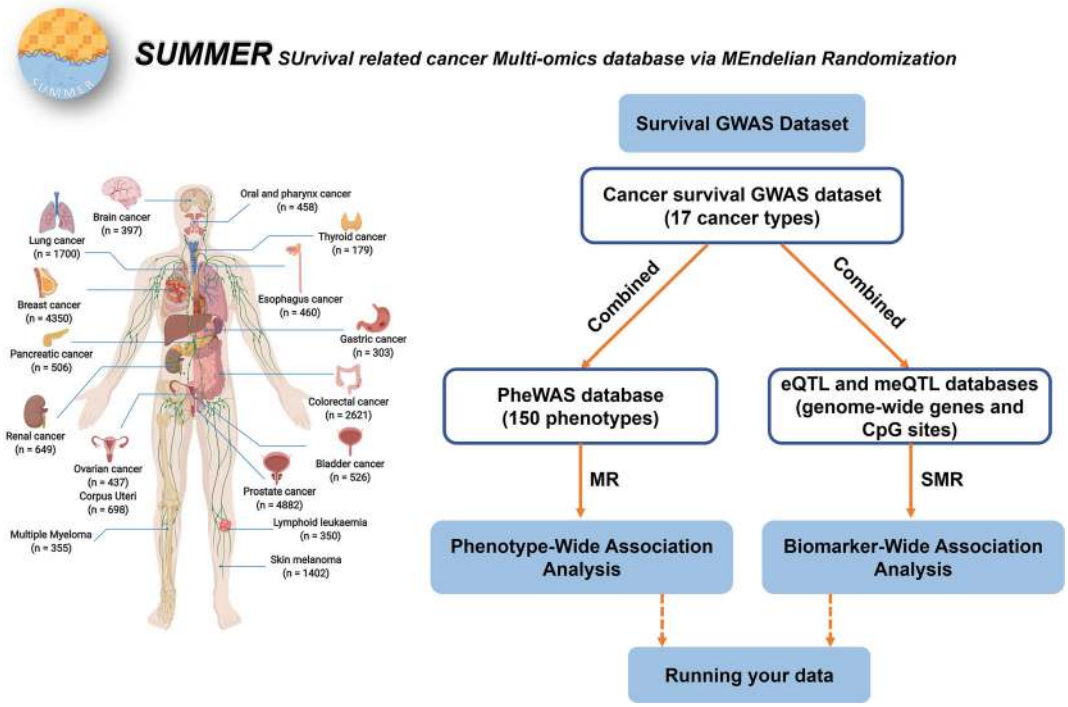


Fig. 1 Summary of the design of SUMMER database

2.1 Preparation of SNP-Exposure Association Datasets

2.1.1 PheWAS Dataset

The GWAS summary statistics of common traits in the PheWAS dataset [15] were accessed through the IEU Open GWAS project (https://gwas.mrcieu.ac.uk/), with the R package *TwoSampleMR* [16, 17].

Based on a strict QC process consisting of (i) limited in European population (*see* **Note 1**) and (ii) with ≥3 independent [linkage disequilibrium (LD) $r^2 < 0.01$] genetic instruments (defined by SNPs with *P*-value $\leq 5 \times 10^{-8}$), we included a total of 150 traits in this database, which spanned the categories of anthropometric, autoimmune/inflammatory, behavioral, cardiovascular, International Classification of Diseases, 10th revision (ICD-10) codes-related, miscellaneous, non-cancer illness, and psychiatric/neurological traits.

2.1.2 QTL Dataset

In addition to PheWAS dataset, we also obtained QTL datasets for biomarker-level analysis, including (i) eQTL datasets from the eQTLGen consortium (https://eqtlgen.org/) that incorporated a total of 31,684 blood samples mostly from donors of European ancestry [18] and (ii) meQTL dataset from Hannon et al.'s study that included a total of 1175 blood samples from donors of European ancestry (*see* **Note 1**) [19].

2.2 Preparation of SNP-Outcome Association (That Is, Cancer Survival GWAS) Datasets

2.2.1 Data Collection and Quality Control in UK Biobank Cohort

The UK Biobank cohort is a prospective, population-based study that recruited 502,528 adults aged 40–69 years from the general population between 2006 and 2010 [20]. Participants visited 1 of 22 assessment centers across England, Scotland, and Wales, where they completed touchscreen and nurse-led questionnaires and provided biological samples. The study protocol and information about data access are available online (https://www.ukbiobank.ac.uk/; Application #45611).

A total of 355,543 participants remained for analysis after the following individual-level quality control (QC) process which (i) excluded individuals with prevalent cancer (except nonmelanoma skin cancer, based on the ICD-10 [C44]) at baseline; (ii) excluded individuals of sex discordance; (iii) excluded outliers for genotype missingness or excess heterozygosity; (iv) retained unrelated participants; (v) restricted the cohort to "white British" individuals of European ancestry (*see* **Note 1**); and (vi) removed individuals who decided not to participate in this program. The follow-up time of cancer survival was measured from cancer diagnosis (defined by ICD-10 codes [21]) to death or the last follow-up (February 14, 2018). We determined whether an individual died of a specific cancer by considering the ICD-10 codes listed as the primary cause of death. Finally, of the 355,543 individuals, 19,628 were newly diagnosed with 1 or more of 17 cancer types, ranging from 179 thyroid cancer cases to 4882 prostate cancer cases (Table 1).

2.2.2 Cancer Survival GWAS Analysis

All samples derived from UK Biobank were genotyped using the UK BiLEVE Axiom Array or UK Biobank Axiom Array by Affymetrix [22]. The genotyping data were imputed using SHAPEIT3 and IMPUTE3 based on the reference panels of Haplotype Reference Consortium (HRC), UK10K, and 1000 Genomes Project (Phase 3). The study protocol and information about data access are available online (http://www.ukbiobank.ac.uk/wp-content/uploads/2011/11/UKBiobank-Protocol.pdf).

We kept variants based on a strict QC process with the following criteria: (i) SNPs located within autosomal chromosomes, (ii) imputation info score ≥ 0.3, (iii) minor allele frequency (MAF) ≥ 0.01, (iv) call rate $\geq 95\%$, and (v) Hardy-Weinberg equilibrium (HWE) P value $\geq 1 \times 10^{-6}$. Subsequently, the Cox proportional hazards regression analysis in an additive genetic model was applied to evaluate the association between each SNP and cancer survival that included OS and CSS as endpoints, with adjustment for sex, age at diagnosis, BMI, smoking status, drinking status, and the top 10 principal components of population stratification when appropriate.

2.3 MR Analysis Framework

2.3.1 MR Analysis in Identification of Cancer Survival-Associated Risk Factors

Here, we used the R package *TwoSampleMR* to apply multiple MR methods [6] in the phenotype-survival association analysis, including inverse variance weighted (IVW), weighted median, penalized weighted median, and MR Egger methods. In addition, the heterogeneity test was used to assess whether a genetic variant's effect on outcome was proportional to its effect on exposure, and the MR-Egger intercept test was fitted to evaluate the presence of horizontal pleiotropy [23]. The suggestive evidence between phenotypes and cancer survival was identified when three nominal thresholds were met, including *P*-value for IVW analysis ≤ 0.05, *P*-value for Egger intercept >0.05, and *P*-value for heterogeneity >0.05.

2.3.2 Summary-Data-Based MR (SMR) Analysis for Identification of Cancer Survival-Associated Circulating Biomarkers

The associations between biomarkers and cancer survival were evaluated using the SMR analytical framework with default settings (--peqtl-smr 5E-08 --peqtl-heidi 1.57E-03 --cis-wind 2000) by integrating the cancer survival GWAS summary statistics data with cis-eQTL and cis-meQTL results (i.e., with a window of 2000 kb to select SNPs centered around the target biomarker) [24, 25]. The genotype data from the European population of the 1000 Genomes Project Phase 3 were used for the LD estimation. The suggestive colocalized signals were determined at a nominal threshold of *P*-value for SMR analysis ≤ 0.05 and *P*-value for HEIDI (i.e., heterogeneity test in dependent instruments) >0.05.

3 Methods

3.1 Design of SUMMER Database

We applied the R package *Shiny* to develop SUMMER database (Fig. 1) with the following four modules:

(i) *Survival GWAS Dataset* module, to help users browse the association effects of over eight million genetic variants on pan-cancer survival

(ii) *Phenotype-Wide Association Analysis* module, to help users browse the causal effects of 150 phenotypes on pan-cancer survival

(iii) *Biomarker-Wide Association Analysis* module, to help users browse the causal effects of genome-wide genes and CpG sites on pan-cancer survival

(iv) *Running your data* module, to allow users to evaluate their own data on pan-cancer survival.

The "About" page provides more details about the function of this database.

3.2 "Survival GWAS Dataset" Module

1. *Search and browse*: Select a cancer type and enter a batch of SNP IDs or a genetic region. A table with cancer type (*see* **Note 2**), chromosome ID, SNP ID, SNP genomic position, SNP alleles (A1: minor/effect allele; A2: major/reference allele), MAF, hazard ratio (HR), standard error (SE), and *P*-value will be built to display the associations of SNPs with cancer survival that includes OS and CSS. Additional function annotation links are also provided (*see* **Note 3**).
2. *Download*: Download the results by clicking the "Download" button.
3. *Plot*: Select one SNP-survival pair and click the "Plot" button. The diagrams of Kaplan-Meier (KM) plot will be provided to display the associations.
4. *Example*: Colorectal cancer patients with the SNP rs17123527 GA or AA genotypes had shorter OS times than patients with the rs17123527 GG genotype (HR = 2.20, $P = 1.27 \times 10^{-5}$; *P* for log-rank test = 1.21×10^{-6}; Fig. 2).

3.3 "Phenotype-Wide Association Analysis" Module

1. *Search and browse*: Select a cancer type (*see* **Note 2**), a phenotype category (e.g., anthropometric and autoimmune/inflammatory; *see* **Note 4**) and a survival type (e.g., OS or CSS). A table with phenotype category, trait, trait ID, cancer type, survival type, MR method, number of IVs, and beta, SE, and *P*-value from the MR analysis will be built to display the associations of selected phenotypes with cancer survival.
2. *Download*: Download the results by clicking the "Download" button.
3. *Plot*: Select one trait-survival pair and click the "Plot" button. The diagrams of MR scatter plot will be provided to display the associations.
4. *Example*: Blood clot in the leg (DVT) was associated with a poorer OS of colorectal cancer ($\text{beta}_{\text{IVW}} = 8.45$, $P_{\text{IVW}} = 0.013$, $P_{\text{egger intercept}} = 0.375$, $P_{\text{IVW heterogeneity}} = 0.509$; Fig. 3).

3.4 "Biomarker-Wide Association Analysis" Module

1. *Search and browse*: Select a cancer type (*see* **Note 2**), a biomarker type (e.g., gene expression or CpG site; *see* **Note 4**), and a survival type (e.g., OS or CSS). A table with cancer type, survival type, probe ID, probe genomic position, top eQTL/meQTL SNP, top SNP genomic position, MAF from 1000 Genomes EUR population, top SNP-associated eQTL and survival GWAS results (including beta, SE, and *P*-value), and beta, SE, and *P*-value (including P_{SMR}, $P_{\text{multi-SMR}}$, and P_{HEIDI}) from SMR analysis will be built to display the associations of selected biomarkers with cancer survival.
2. *Download*: Download the results by clicking the "Download" button.

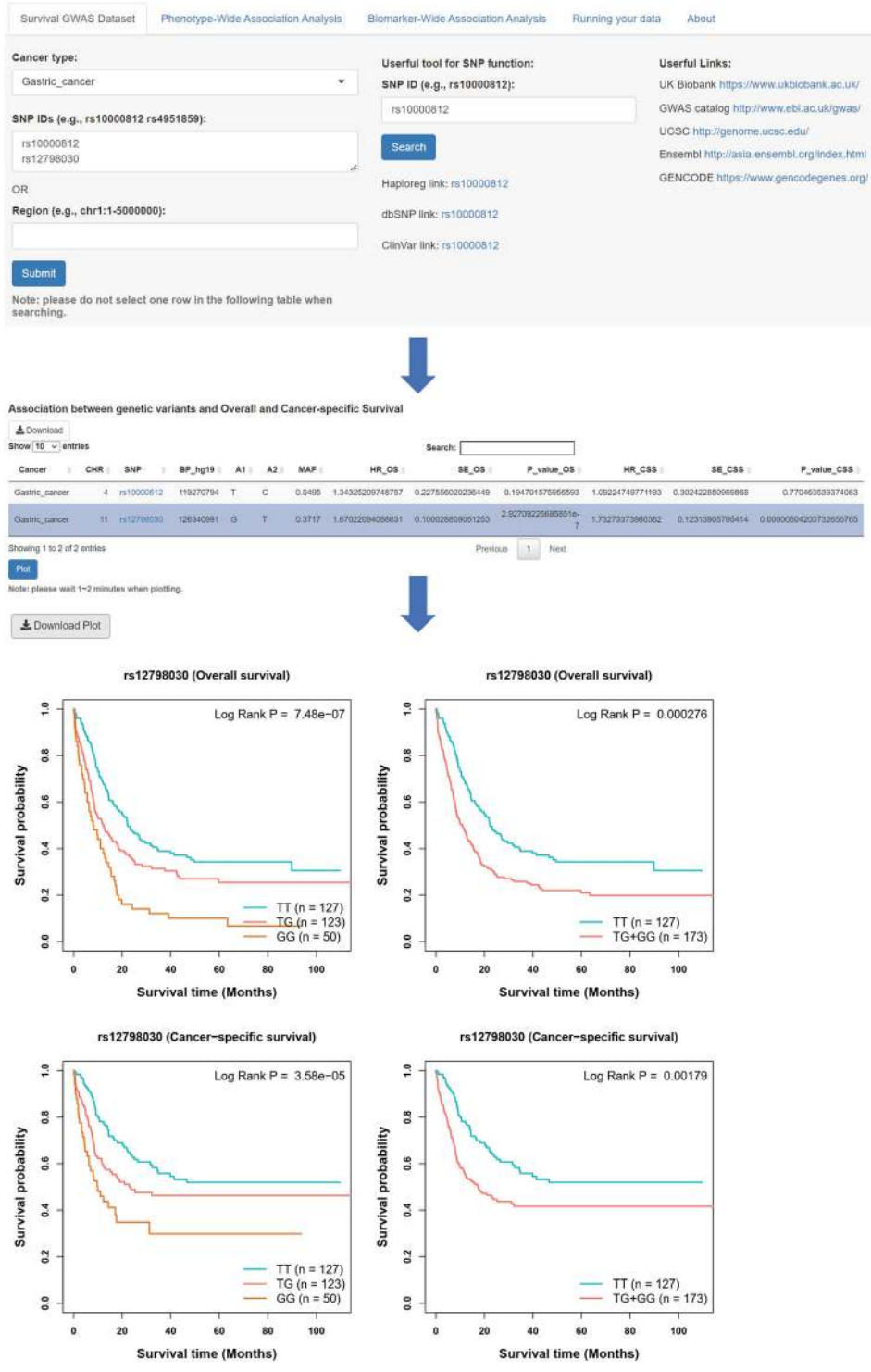
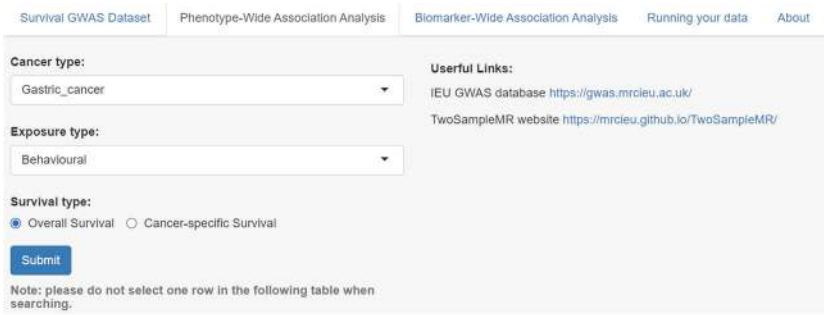
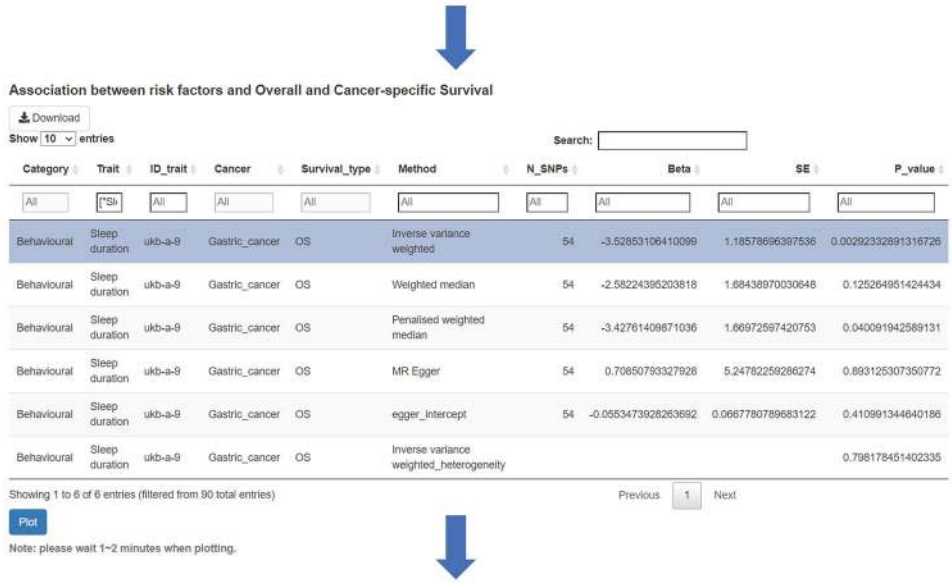


Fig. 2 Overview of the "Survival GWAS Dataset" module in the SUMMER database





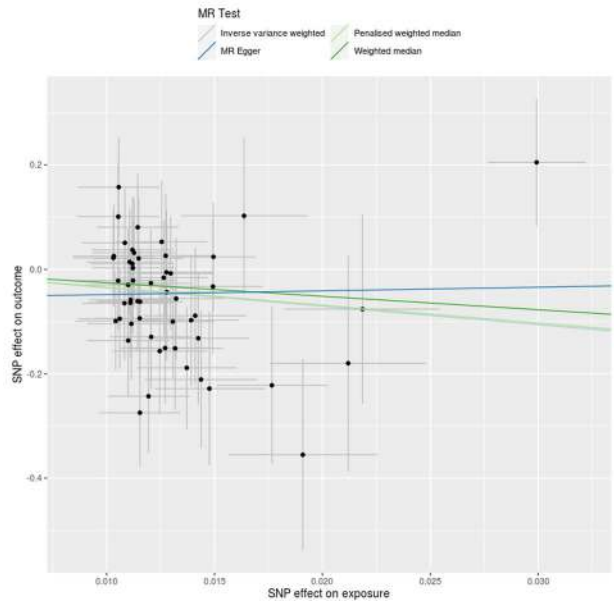


Fig. 3 Overview of the "Phenotype-Wide Association Analysis" module in the SUMMER database

3. *Plot*: Select one biomarker-survival pair and click the "Plot" button. The diagrams of SMR scatter plot will be provided to display the associations.
4. *Example*: Higher expression of *HTR6* was associated with poorer OS in colorectal cancer ($\text{beta}_{\text{SMR}} = 0.72$, $P_{\text{SMR}} = 2.38 \times 10^{-4}$, $P_{\text{multi-SMR}} = 0.007$, $P_{\text{HEIDI}} = 0.692$; Fig. 4).

3.5 "Running Your Data" Module

1. *Selection of type of exposures and outcomes*: Select a cancer type (*see* **Note 2**), a data type (e.g., phenotype or biomarker), and a survival type (e.g., OS or CSS), and enter a data name and email address (optional).
2. *Data uploading*: Upload your summary statistic data (.csv format).
3. *Data analysis, browse, and download*: Submit your data and perform analysis (*see* **Note 5**). A table derived from the MR or SMR analysis will be built to display the associations of related phenotypes/biomarkers with cancer survival, which can be downloaded by clicking the "Download" button or received by email.
4. *Plot*: Select one pair and click the "Plot" button. The diagrams of MR/SMR scatter plots will be provided to display the associations.
5. *Example*: The uploaded trait_1 was not associated with the OS of colorectal cancer ($\text{beta}_{\text{IVW}} = 6.15$, $P_{\text{IVW}} = 0.212$, $P_{\text{egger intercept}} = 0.518$, $P_{\text{IVW heterogeneity}} = 0.458$; Fig. 5).

4 Notes

1. Our database is mainly used to identify cancer survival-relevant risk factors and biomarkers in European populations, of which the findings may not be directly transferred in other ancestries.
2. The cancer outcomes are focused on OS and CSS of 17 cancers from UK Biobank cohort. More cancer survival GWAS datasets with larger sample sizes and longer follow-up times will be updated in our database, or at the user's own survival GWAS datasets.
3. Our database does not provide sufficient functional annotation for the identified genetic variants and biomarkers but shows several informative links (e.g., Haploreg, dbSNP, and ClinVar) on the website.
4. The phenotypes of our database only include 150 traits, and there is a limited number of biomarkers based on the gene expression and DNA methylation at the circulating level.

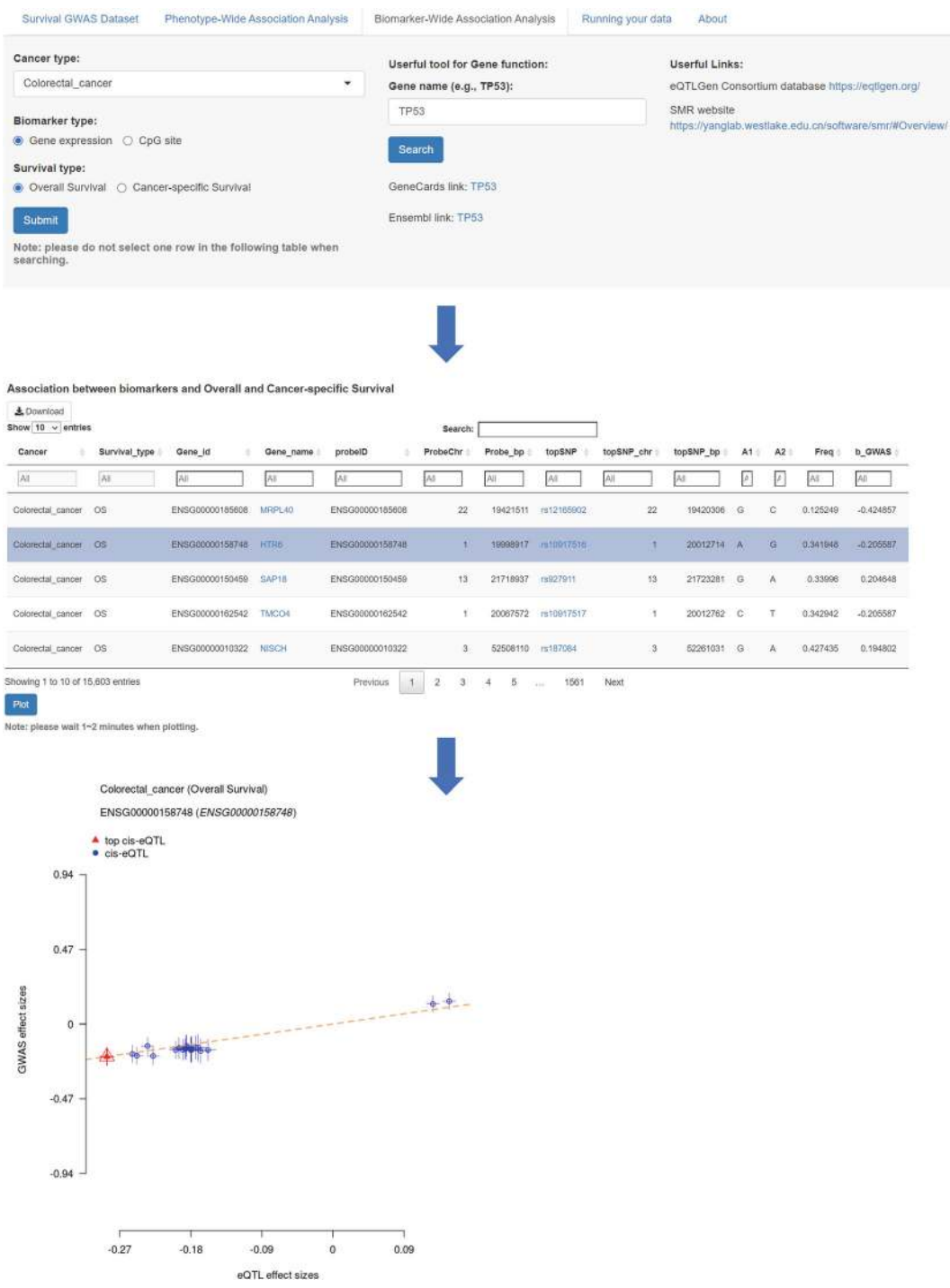


Fig. 4 Overview of the "Biomarker-Wide Association Analysis" module in the SUMMER database

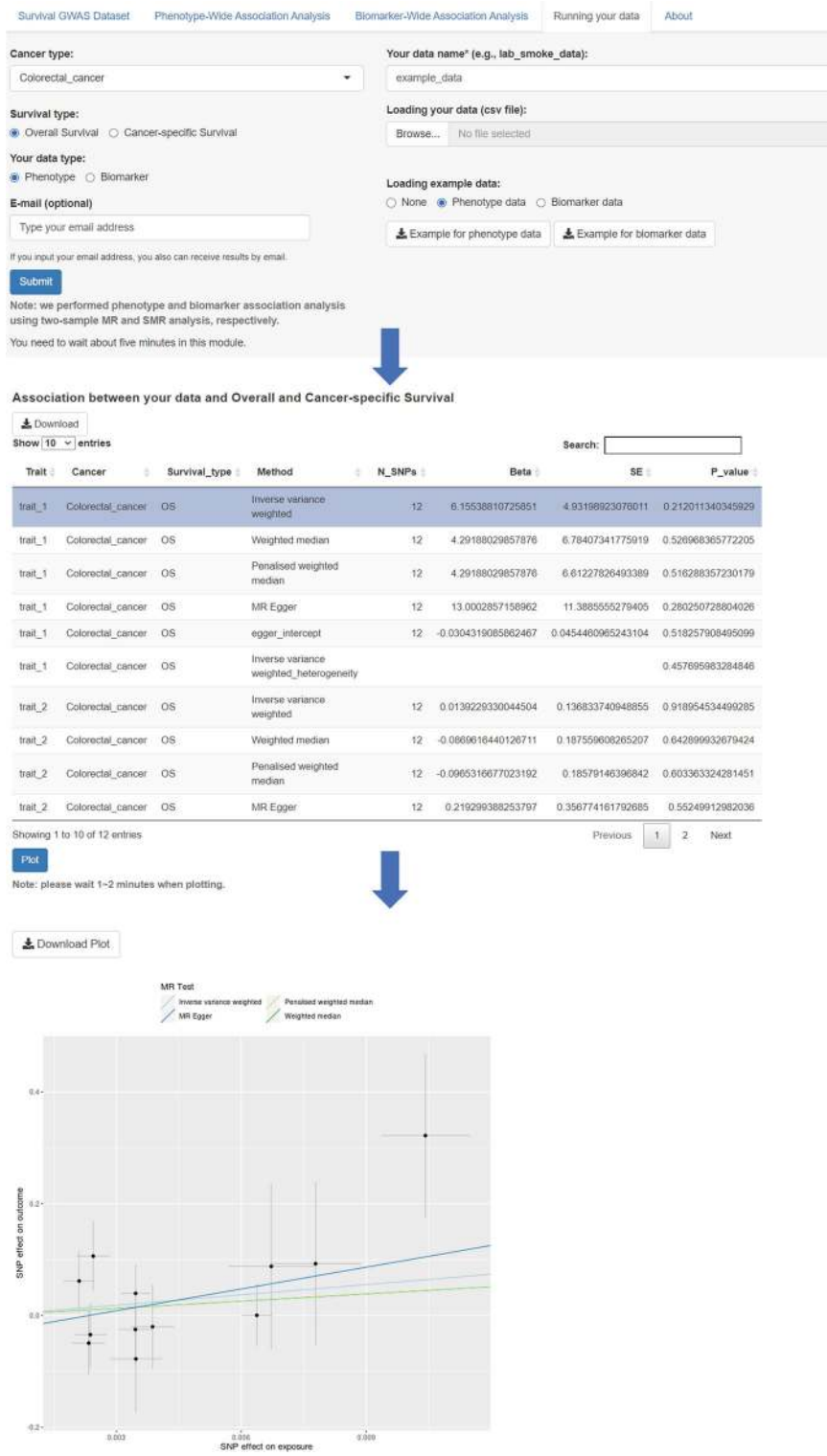


Fig. 5 Overview of the "Running your data" module in the SUMMER database

Additional risk factors and multi-tissue biomarkers will be updated in our database.

5. In the *Running your data* module, users must upload the SNP-exposure association file in the example format (e.g., . csv file) and cannot choose other parameters (e.g., the distance and LD r^2 for selection of genetic instrument). In addition, when running data with more genetic instruments, our tool needs more time for analysis. If errors occur, they are likely due to problems for the data format, the number of genetic instruments, or others.

Acknowledgments

This work was supported by the Natural Science Foundation of Jiangsu Province (BK20230003; BK20240524). We are grateful to the participants and study staff of the UK Biobank.

Conflict of Interest None

References

1. Sud A, Kinnersley B, Houlston RS (2017) Genome-wide association studies of cancer: current insights and future perspectives. Nat Rev Cancer 17:692–704
2. Buniello A, MacArthur J, Cerezo M et al (2019) The NHGRI-EBI GWAS Catalog of published genome-wide association studies, targeted arrays and summary statistics 2019. Nucleic Acids Res 47:D1005–D1012
3. Visscher PM, Brown MA, McCarthy MI et al (2012) Five years of GWAS discovery. Am J Hum Genet 90:7–24
4. Zuber V, Grinberg NF, Gill D et al (2022) Combining evidence from Mendelian randomization and colocalization: review and comparison of approaches. Am J Hum Genet 109: 767–782
5. Gallagher MD, Chen-Plotkin AS (2018) The post-GWAS era: from association to function. Am J Hum Genet 102:717–730
6. Smith GD, Ebrahim S (2003) 'Mendelian randomization': can genetic epidemiology contribute to understanding environmental determinants of disease? Int J Epidemiol 32: 1–22
7. Davies NM, Holmes MV, Davey SG (2018) Reading Mendelian randomisation studies: a guide, glossary, and checklist for clinicians. BMJ 362:k601
8. Davey SG, Hemani G (2014) Mendelian randomization: genetic anchors for causal inference in epidemiological studies. Hum Mol Genet 23:R89–R98
9. Xin J, Jiang X, Ben S et al (2022) Association between circulating vitamin E and ten common cancers: evidence from large-scale Mendelian randomization analysis and a longitudinal cohort study. BMC Med 20:168
10. Lin L, Li Z, Yan L et al (2021) Global, regional, and national cancer incidence and death for 29 cancer groups in 2019 and trends analysis of the global cancer burden, 1990–2019. J Hematol Oncol 14:197
11. Sung H, Ferlay J, Siegel RL et al (2021) Global cancer statistics 2020: GLOBOCAN estimates of incidence and mortality worldwide for 36 cancers in 185 countries. CA Cancer J Clin 71:209–249
12. Liu J, Lichtenberg T, Hoadley KA et al (2018) An integrated TCGA pan-cancer clinical data resource to drive high-quality survival outcome analytics. Cell 173:400–416
13. Arnold M, Rutherford MJ, Bardot A et al (2019) Progress in cancer survival, mortality, and incidence in seven high-income countries 1995-2014 (ICBP SURVMARK-2): a population-based study. Lancet Oncol 20: 1493–1505

14. Xin J, Gu D, Chen S et al (2023) SUMMER: a Mendelian randomization interactive server to systematically evaluate the causal effects of risk factors and circulating biomarkers on pan-cancer survival. Nucleic Acids Res 51: D1160–D1167
15. Prince C, Mitchell RE, Richardson TG (2021) Integrative multiomics analysis highlights immune-cell regulatory mechanisms and shared genetic architecture for 14 immune-associated diseases and cancer outcomes. Am J Hum Genet 108:2259–2270
16. Lyon MS, Andrews SJ, Elsworth B et al (2021) The variant call format provides efficient and robust storage of GWAS summary statistics. Genome Biol 22:32
17. Hemani G, Zheng J, Elsworth B et al (2018) The MR-Base platform supports systematic causal inference across the human phenome. elife 7:e34408
18. Vosa U, Claringbould A, Westra HJ et al (2021) Large-scale cis- and trans-eQTL analyses identify thousands of genetic loci and polygenic scores that regulate blood gene expression. Nat Genet 53:1300–1310
19. Hannon E, Gorrie-Stone TJ, Smart MC et al (2018) Leveraging DNA-methylation quantitative-trait loci to characterize the relationship between methylomic variation, gene expression, and complex traits. Am J Hum Genet 103:654–665
20. Sudlow C, Gallacher J, Allen N et al (2015) UK biobank: an open access resource for identifying the causes of a wide range of complex diseases of middle and old age. PLoS Med 12: e1001779
21. Zhu M, Wang T, Huang Y et al (2021) Genetic risk for overall cancer and the benefit of adherence to a healthy lifestyle. Cancer Res 81: 4618–4627
22. Bycroft C, Freeman C, Petkova D et al (2018) The UK Biobank resource with deep phenotyping and genomic data. Nature 562:203–209
23. Burgess S, Thompson SG (2017) Interpreting findings from Mendelian randomization using the MR-Egger method. Eur J Epidemiol 32: 377–389
24. Zhu Z, Zhang F, Hu H et al (2016) Integration of summary data from GWAS and eQTL studies predicts complex trait gene targets. Nat Genet 48:481–487
25. Wu Y, Zeng J, Zhang F et al (2018) Integrative analysis of omics summary data reveals putative mechanisms underlying complex traits. Nat Commun 9:918

Chapter 18

Predicting Tumor Antigens Using the LENS Workflow Through RAFT

Steven P. Vensko II, Dante Bortone, and Benjamin G. Vincent

Abstract

Tumor-specific and tumor-associated antigens presented on the tumor cell surface by MHC molecules are enticing targets for personalized vaccination and T cell receptor-engineered T cell (TCR-T) therapy. Accurately predicting suitable tumor antigens is a considerable challenge and requires flexibility in both computational tools and experimental methods. Here we describe our framework for reproducible bioinformatics, RAFT, as well as our highly modular neoantigen prediction workflow, LENS. We provide step-by-step instructions for installation, running, and modifying LENS to suit different purposes.

Key words Tumor antigen, Neoantigen, Immuno-oncology, Tumor-associated antigen, Neoantigen workflow, Bioinformatics

1 Introduction

Cancer immunotherapy involves the treatment of tumors through targeted control and elimination by a patient's immune system. Vaccines designed against tumor-specific neoantigens have shown success in eliciting tumor antigen-specific T cell responses in some patient groups and tumor types [10, 23]. Nevertheless, the vast majority of trials result in minimal clinical improvement for patients [12]. This lack of beneficial outcomes underscores many of the field's difficult and unanswered questions, such as those related to immunogenicity. Despite these hurdles, advancements in assays, sequencing technologies, and applications of machine learning suggest the efficacy of personalized neoantigen vaccines should vastly improve in the coming years.

Alexander Krasnitz and Pascal Belleau (eds.), *Cancer Bioinformatics*, Methods in Molecular Biology, vol. 2932,
https://doi.org/10.1007/978-1-0716-4566-6_18,

2 Tumor Antigens

Personalized tumor vaccines may target both tumor-specific and tumor-associated antigens [7]. Tumor-specific antigens are generated by disruptive genomic events occurring within the tumor, but not the adjacent normal tissues. These events include somatic mutations (single-nucleotide variants and insertions/deletions), tumor-specific transcriptional splice variants, and gene fusions, among others. Tumor-associated antigens, on the other hand, may exist in normal tissues but typically are not expressed or are heavily tumor-biased in their expression. These antigens include endogenous retroviruses and cancer-testis antigens. Endogenous retroviruses are ancient viruses that are abundant in the genome (up to 8% of the genome's content) [8]. Endogenous retroviruses are largely degraded and nonfunctional, but some retain sufficient coding sequences for transcription and translation. They have been observed in normal tissues but are largely downregulated [11, 16]. Cancer-testis antigens (CTAs) are produced by transcripts that are almost exclusively expressed in immune-privileged testis and embryonal tissue [25]. CTA expression in immune-privileged tissue suggests they may be recognized by the patient's immune system as non-self which may trigger an antitumor immune response. Tumors may have disrupted chromatin states throughout their genome allowing endogenous retroviruses and cancer-testis antigens to be aberrantly expressed.

3 Role of Bioinformatics in Personalized Neoantigen Vaccines

The ability to address challenges within the field of immuno-oncology is heavily reliant upon bioinformatics tools and workflows. The advent of relatively inexpensive short-read exome and RNA sequencing resulted in the explosive growth of data available across a variety of tumor types. These data, combined with bioinformatics tools, can be used to predict tumor-specific and tumor-associated antigens.

Specifically, standard sequence-based bioinformatics tools coupled with custom scripts can be used to predict tumor antigens that may potentially be presented on the tumor cell surface. Somatic single-nucleotide variants, insertions, and deletions can be detected and annotated using tools like MuTect2 and snpEff, respectively [2, 4]. Tumor RNA-sequencing data can be used to detect gene fusion events or, when combined with a tissue-matched normal sample, tumor-biased splice events and endogenous retroviral expression. The resulting predicted peptides and patient-specific HLA allele information can be used to predict peptide-MHC (pMHC) binding affinity, stability, and relative expression—some of the factors believed to be relevant to predicting a pMHC's potential immunogenicity [28].

Here, we present a detailed description and tutorial for running the LENS neoantigen workflow through RAFT, our in-house framework, and workflow manager built on Nextflow DSL2 [5]. First, we will cover RAFT's functionality and how it supports reproducible bioinformatics (and, in turn, LENS), and then we will describe running LENS.

4 RAFT

A common issue with bioinformatics analyses is difficulty in reproducing the results of a journal article. Many article authors will generally describe the process in which data were generated and the analyses were performed, but commonly omit critical details such as tool versions and parameters used. Our workflow manager, RAFT (Reproducible Analyses Framework and Tools), seeks to support absolute reproducibility within the field of sequencing-based bioinformatics while also providing an easy-to-use interface. The eventual goal for RAFT is to support the running of complex workflows by third parties with minimal difficulty. Reproducibility through RAFT is accomplished through several interoperating facets: (1) a project context, (2) module-based components, and (3) RAFT packages. Here, we describe these aspects of RAFT and how they support project transparency and reproducibility.

4.1 Project Context

Specifically, a project context entails an isolated environment in which an analysis is performed. The project's contents consist of reference files, sample-level input files, the output files generated, and the module code running each workflow step. This isolated environment ensures that every component needed to reproduce results is contained within the project.

4.2 Module-Based Components

Modules within RAFT define command-level process definitions, subworkflow definitions, resource allocation (CPU and memory), and the Docker image to use for running the processes and workflows. These modules exist locally within the project such that users can modify them as needed to change tool-level behaviors.

4.3 Project Packaging

RAFT projects contain end-to-end analyses and can, depending on the workflow, require hundreds of gigabytes to terabytes of storage. Unfortunately, it is rarely practical to store the entire project for long periods. We have included functionality within RAFT, the RAFT package, to address this issue. Specifically, the RAFT package (.rftpkg) is a tarfile consisting of (1) checksums of input, outputs, and references, (2) all of the module code used to run the analysis (including any user modifications), and (3) the RAFT commands used to generate and run the project. The tarfile containing these components is a fraction of the size of the original project and allows for the regeneration of the project at a later date if needed.

RAFT packages also serve as a mechanism to share an analysis with third parties as it is sufficient to rerun the entire analysis (assuming the third party has access to the input files).

More information on RAFT can be found at https://useraft.io.

4.4 Installation

RAFT installation can be performed with either pip (https://pypi.org/project/reproducible-analyses-framework-and-tools) or using Conda environments (https://anaconda.org/raft/reproducible-analyses-framework-and-tools). RAFT requires the following dependencies:

- Python 3.6
- python-wget 3.2 (https://pypi.org/project/wget/)
- gitpython 3.1.9 (https://github.com/gitpython-developers/GitPython)
- python-gitlab 4.2.0 (https://github.com/python-gitlab/python-gitlab)
- Nextflow 23.10.0 (https://nextflow.io/)
- Git 2.43.0 (https://git-scm.com/)
- OpenJDK 21.0.1 (https://openjdk.java.net)

4.4.1 Installation Through pip

RAFT installation through pip requires Java $>= 11$ to be installed by the user (for example, using sudo apt-get install openjdk11-jdk) to support Nextflow. Note that installation through pip does *not* include installation of Singularity/Apptainer and instead requires the user to install it separately (or use another supported tool, like Docker).

Installing RAFT Using pip

```
pip install --user reproducible-analyses-fra-
mework
          -and-tools
```

4.4.2 Installation Through Conda

The RAFT Conda package provides a full working environment that installs all required dependencies.

Installing RAFT Using Conda

```
conda install -c bioconda -c conda-forge -c
raft \
          reproducible-analyses-framework-and
           -tools
```

4.4.3 Setting up RAFT

RAFT must be installed within a directory that will house all projects and project-related artifacts (e.g., references, fastqs, RAFT packages, etc.). RAFT can be quickly installed using:

Setting up RAFT with Default Parameters

```
raft.py setup -d
```

Usage of the -d/–default flag will create all of the required directories in the working directory in which the command is run. Users desiring a more flexible, custom installation can instead run:

Setting up RAFT with Interactive Prompt

```
raft.py setup
```

This command initiates a prompt requesting user-provided paths for each directory required by RAFT. These directories will be created or will be symbolically linked (symlinked) to the RAFT installation directory if they already exist elsewhere on the file system.

4.4.4 Setting up Nextflow Profile

Nextflow supports a variety of executors to schedule and run its processes. For demonstration purposes, we will be running in the local environment using Singularity/Apptainer to run containers. The following configuration can be placed in ~/.nextflow/config to run RAFT:

Example Nextflow Profile

```
profiles {
  standard {
    process.executor = 'local'
    executor.queueSize = '99'
    executor.submitRateLimit = '25/2min'
    singularity.enabled = true
  }
}
```

5 LENS Workflow Demonstration

Now that RAFT and its dependencies are installed, users can run LENS in demonstration mode. Demonstration mode uses a predefined manifest and set of FASTQs, so users can see how the workflow works before providing their samples. Demonstrations are run using the RAFT command run-demo. The demonstration project's identifier defaults to demo-<WORKFLOW> (e.g., demo-lens) and will be available in ./raft/projects/demo-<WORKFLOW>.

The LENS demonstration can be run using:

Running LENS Demonstration

```
raft.py run-demo --workflow lens -mb v1.2-dev
```

The -mb Parameter
The -mb parameter specifies the **m**odule **b**ranch. The latest version of LENS at this time of publication is v1.2-dev, but users are encouraged to check for the latest version on the LENS wiki (see Subheading 12).

RAFT will then download them if necessary and load the required inputs into the project (truncated for brevity):

LENS Demonstration Output

```
Initializing project default-demo...
Pulling off-the-shelf workflow...
This may take some time due to module fetching.
Loading manifest...
Couldn't find lens.demo.manifest in RAFT meta-
data
       directory.
Loading references...
Couldn't find Homo_sapiens_assembly38.fasta.
       Downloading...
Couldn't find gencode.v44.annotation.gtf.gz.
       Downloading...
Loading fastqs...
Couldn't find SRR8668635_1.fastq.gz. Download-
```

(continued)

```
ing...
Couldn't find SRR8668635_2.fastq.gz. Download-
ing...
Building workflow...
Connecting subworkflows...
Populating default parameters...
Running workflow...
```

Outputs from the demonstration are available for review in ./raft/projects/demo-<WORKFLOW>/outputs. Output file descriptions can be found in Subheadings 8 and 9.

6 Running LENS with User-Provided Manifest

6.1 Loading Reference and FASTQ Files

RAFT expects reference files, FASTQ files, and, in most cases, metadata/manifest files to be available within specific directories (./raft/references, ./raft/fastqs, and ./raft/metadata, respectively) when running a project's workflow. Many of RAFT's off-the-shelf workflows, such as LENS, will automatically download any references required for running. They will also automatically copy the user-provided metadata file to the correct location. Users, however, must copy or symlink their sample-level FASTQs to the ./raft/fastqs directory.

LENS currently supports single-end and paired-end Illumina sequencing by synthesis (SBS) reads. For optimal performance, each patient should have three samples including: (1) normal DNA, (2) tumor DNA, and (3) tumor RNA. Users can also provide only a tumor RNA sample for each patient but will not receive SNV and InDel neoantigen predictions. A single-reference tissue-matched (to the tumor type) normal RNA sample is also required. LENS has been extensively tested on whole exome sequencing data (WES or WXS) but should also work with whole genome sequencing (WGS) data.

6.2 Generating and Checking the RAFT Manifest

The relationship between samples and patients is defined within the RAFT manifest. The samples required for a RAFT workflow vary depending on the workflow. For example, an RNA quantification workflow may use one or multiple RNA-sequencing samples per patient. Other workflows, like somatic variant calling, will require both a normal sample and a tumor sample for each patient. For optimal performance, LENS requires patient-specific normal DNA samples, tumor DNA samples, and tumor RNA samples as well as a tissue-specific control normal sample to be shared among patients. RAFT manifests may contain one or more patients. Many computer

Table 1
Required RAFT manifest columns

Column	Description	Allowed values
File_Prefix	Base name of FASTQ files	Free text
Patient_Name	Name for collection of samples	Free text
Normal	Is the sample normal or abnormal (tumor)?	(TRUE, FALSE)
Sequencing_Method	Sequencing protocol for sample	(RNA-Seq, WES, WXS, WGS)
Dataset	Name for collection of patients	Any string
Run_Name	Name for the specific sample	Free text (see note below)

clusters will allow for multiple patients to be run in parallel to reduce run time.

A RAFT manifest must have at least the columns defined in Table 1. Columns can be in any order, and other columns containing non-RAFT metadata are also allowed.

The general hierarchy of organization within RAFT follows:

$$Sample \in Patient \in Dataset.$$

In other words, samples belong to patients (patients can have multiple samples) and patients belong to datasets (datasets can have multiple patients).

Run_Names are instrumental in guiding samples through the LENS workflow. A sample's Run_Name should have a two-letter prefix that describes the type of sample followed by an arbitrary unique identifier. The first letter of the prefix is either **a** (for abnormal) or **n** (for normal). The second letter is either **r** (for RNA) or **d** (for DNA). For example, a sample with an ar- prefix is an abnormal (tumor) RNA sample, while a sample with a nd- prefix is a normal DNA sample.

Each line in the manifest after the header corresponds to a sample and provides the necessary data for running the workflow. The samples described within the manifest may, in some cases, be effectively independent (as in, the workflow does not attempt to pair samples from a patient), but in other cases, users must be careful that samples are properly labeled. For example, somatic variant calling generally requires a normal DNA sample and a tumor DNA sample. For RAFT to properly pair these samples together, they must have the correct sample prefix (nd- for the DNA tumor sample and nd- for the DNA normal sample) and be paired with the patient (Patient_Name field) and dataset (Dataset field). Consider the following example:

Example Manifest

```
Patient_Name Run_Name Dataset File_Prefix Se-
quencing
      _Method Normal
Pt01 ad-Pt01-03A AML      9f7f7    WES    FALSE
Pt01 nd-Pt01-11A AML      8e74a    WES    TRUE
Pt01 ar-Pt01-03A AML     cdb288   RNA-Seq FALSE
CTRL nr-CTRL      TCGA_LAML CD34-032U RNA-Seq
TRUE
```

Note that both the tumor DNA sample (ad-Pt01-03A) and the normal DNA (nd-Pt01-11A) sample belong to the same patient (Pt01) and the same dataset (AML).

User-provided manifests can be sanity-checked by using:

Sanity Checking Manifests

```
raft.py check-manifest -m /path/to/manifest.
tsv
```

Users can then provide the manifest to RAFT's run-ots command to run the workflow with their manifest.

6.3 Running LENS Workflow with Default Parameters

Running LENS with a user-provided manifest and default parameters can be accomplished with:

Running an off-the-Shelf Workflow

```
raft.py run-ots \
  --project-id my_lens_project \
  --workflow lens \
  -mb v1.2-dev \
  --metadata /path/to/manifest.tsv
```

The -mb Parameter

The -mb parameter specifies the **m**odule **b**ranch. The latest version of LENS at this time of publication is v1.2-dev, but users are encouraged to check for the latest version on the LENS wiki (see Subheading 12).

This command will create a new project in the ./raft/projects directory named my_lens_project. Note that project names are free text, but RAFT's developers highly encourage users to name their projects using meaningful nomenclature (e.g., < DATASET > - < WORKFLOW > - < DATE >). A meaningful naming convention will allow users to better organize their projects and prevent potential confusion.

Note that the relevant FASTQs must be present within the RAFT FASTQs directory (./raft/fastqs) for the workflow to be completed successfully. Information regarding accessing and interpreting LENS output files can be found in Subheading 8 and 9.

7 Modifying LENS Modules, Parameters, and References

RAFT's off-the-shelf workflows run with sensible default parameters, but it is possible to modify workflow- to tool-level parameters to suit specific situations. In this case, users should run the run-ots command with the --setup-only parameter:

Setting up (but not Running) an off-the-Shelf Workflow

```
raft.py run-ots \
  --project-id my_lens_project \
  --workflow lens \
  -mb v1.2-dev
  --metadata /path/to/user-generated-manifest.
tsv \
  --setup-only
```

This command will run the same steps as the run-ots command, except it will not execute Nextflow. This provides an opportunity to modify the workflow before execution.

7.1 Workflow Modifications

Users can drastically alter workflow behaviors by modifying tool versions, tool containers, and tool-level parameters. They can also swap tools (e.g., swapping salmon with kallisto for RNA quantification) or reference files. Each section below describes the steps required to modify the workflow.

7.1.1 Parameter Modifications

Tool-level and workflow parameters can be tweaked by users through modification of the main.nf file (found in ./raft/projects/ < PROJECT_ID > /workflow/main.nf). Lines within the main.nf file that start with "params." are parameters that can be

user-modified. The parameter names, while long, describe what aspect of the workflow is being controlled by the parameter. Parameter names are $ delimited with each segment providing an increasingly narrow scope of the parameter's application. For example:

Parameter for DNA Alignment Tool

```
params.lens$alignment$manifest_to_dna_alns
$aln_tool
```

This parameter is a part of the lens module which is utilizing the alignment module's manifest_to_dna_alns subworkflow which requires the specific parameter aln_tool.

Using the above logic, we can see the following parameter assigns the RNA alignment step to use the tool STAR [6]:

Setting RNA Alignment Tool to star

```
params.lens$alignment$manifest_to_rna_alns
$aln_tool
     = "star"
```

This allows us to specify the tool to be used but does not specify the parameters provided to that tool. Providing tool-specific parameters is shown of the next line:

Setting Parameters for Running star

```
params.lens$alignment$manifest_to_rna_alns
$aln_
     tool_parameters = \
"[star': --quantMode TranscriptomeSAM --out-
SAMtype
     BAM \
SortedByCoordinate --twopassMode Basic --out-
SAMunmapped
     Within']"
```

Tool-level parameters are defined using a key:value strategy similar to a Python dictionary. In the above example, 'star' is the

key, and '–quantMode TranscriptomeSAM –outSAMtype BAM SortedByCoordinate –twopassMode Basic –outSAMunmapped Within' is the value. Note that this parameter set is *only* passed to star in the context of running the manifest_to_rna_alns workflow and will not affect other instances of star being called.

7.1.2 Tool, Container, and Resource Modifications

Users may also change the tool or the tool container used within a process. Recall the previous section in which star is being used for aligning short-read RNA-sequencing data to a reference. If users instead wanted to use a different tool such as bbmap [3], then they would change

Specifying STAR for RNA Alignment

```
params.lens$alignment$manifest_to_rna_alns
$aln_tool
      = "star"
```

to

Specifying BBMap for RNA Alignment

```
params.lens$alignment$manifest_to_rna_alns
$aln_tool
      = "bbmap"
```

Likewise, they would also change the line

Specifying STAR Alignment Parameters

```
params.lens$alignment$manifest_to_rna_alns
$aln_tool
      _parameters = \
"[star': --quantMode TranscriptomeSAM --out-
SAMtype
      BAM \
SortedByCoordinate --twopassMode Basic --
      outSAMunmapped Within']"
```

to

Specifying BBMap Alignment Parameters

```
params.lens$alignment$manifest_to_rna_alns
$aln_tool
     _parameters = \
"[bbmap': ']"
```

while replacing '' with any parameters they would like passed to bbmap.

Note that only tools currently supported by RAFT may be utilized. A set of supported tools for each workflow, as well as information on creating modules to support new tools and workflows, can be found on the RAFT and LENS wikis (see Subheading 12).

Finally, multiple tools can be specified for some subworkflows. For example, somatic variant calling can be performed with a single tool:

Specifying a Single Somatic Variant Caller

```
params.somatic$alns_to_som_vars$som_var_cal-
ler
     = "strelka2"
```

or users may specify multiple variant callers [2, 13, 15]:

Specifying Multiple Somatic Variant Callers

```
params.somatic$alns_to_som_vars$som_var_cal-
ler = \
"strelka2,mutect2,varscan2"
```

Note that parameters can be specified on a tool basis using the following:

Applying Parameters to Multiple Somatic Variant Callers

```
params.lens$somatic$alns_to_som_vars$som_-
```

(continued)

```
var_caller
     _parameters = \
"['gatk_filter_mutect_calls_suffix': '.
gfilt', \
'varscan2': '--output-vcf']"
```

Different workflows have differing mechanisms for handling situations in which multiple tools are specified for a workflow, but that topic is beyond the scope of this chapter.

Specifying different containers for Nextflow processes allows users to change the version of a tool being run within the workflow. Containers are defined on a per-process level within the nextflow.config file (located at ./raft/projects/<PROJECT_ID>/workflow/nextflow.config). Viewing the nextflow.config file reveals the containers used for several tools:

Specifying Docker Images for Tools

```
...
withLabel: star_container {
  container = docker://mgibio/star:2.7.0f'
}
...
```

Here, we see that processes with the label star_container will be run with mgibio's star:2.7.0f Docker image. To determine which processes in the workflow have the star_container label, users can run:

Determining Processes Using Specific Docker Image Label

```
$ grep -n star_container projects/<PROJEC-
T_ID>/
     workflow/*/*nf
projects/<PROJECT_ID>/workflow/star/star.
nf:20: \
  label star_container'
projects/<PROJECT_ID>/workflow/star/star.
nf:75:\
  label star_container'
```

which directs users to lines 20 and 75 of star.nf.

Users can change the version of star used within the workflow by changing the Docker image specified within nextflow.config. For example:

Changing Docker Image for Specific Tool

```
...
withLabel: star_container {
  container = docker://quay.io/biocontainers/
star
       :2.7.11a'
}
...
```

7.1.3 Reference Modifications

Another crucial consideration when running a complex bioinformatics workflow is the set of reference files used. Users may want to use a reference file more relevant to their specific application, or they may want to update a reference file to a newer version. Reference modifications can be performed using a three-step process. Specifically, users must (1) download the new reference from the external source, (2) load the reference into the project using the load-reference mode of RAFT (e.g., raft.py load-reference -p mylens-project -f reference-name), and (3) modify the appropriate line within their project's main.nf file to reflect the updated reference.

As an example, consider a scenario in which a user may want to upgrade their hg19 reference to an hg38 reference within LENS. This would require the user to download the updated reference:

Downloading External Reference

```
wget https://storage.googleapis.com/genomics-
public
    -data/\
     resources/broad/hg38/v0/Homo_sapiens_as-
sembly38
    .fasta
```

Then copy, move, or symlink that reference to their RAFT's /references directory:

Moving External Reference to RAFT References Directory

```
mv Homo_sapiens_assembly38.fasta /path/to/
raft/
  references
```

Next, the user must load the reference into their project. Here, the project identifier is my-lens-project.

Loading External Reference into a RAFT Project

```
raft.py load-reference -p my-lens-project \
                  -f Homo_sapiens_assembly38
                  .fasta
```

Note that *only* the file's name is needed and that the file does not need to have a specific path within the /references directory (as long as the file is contained within RAFT's references directory).

Finally, users must modify the appropriate lines within the project's main.nf file.

Original Genomic References within main.nf

```
params.lens$alignment$manifest_to_dna_alns
$alns_ref
 = \ "params.{ref_dir}/hg19.fa"
 ...
params.lens$alignment$manifest_to_rna_alns
$alns_ref
 = \ "params.{ref_dir}/hg19.fa"
```

can be changed to:

Updated Genomic References within main.nf

```
params.lens$alignment$manifest_to_dna_alns
$alns_ref = \
"params.{ref_dir}/Homo_sapiens_assembly38.
fasta"
```

(continued)

```
...
params.lens$alignment$manifest_to_rna_alns
$alns_ref = \
"params.{ref_dir}/Homo_sapiens_assembly38.
fasta"
```

In the case of LENS, both the RNA and DNA references must be modified as LENS allows users to use different references for each nucleic acid sequence type.

Users must be careful when updating some reference files due to inter-dependencies among reference files. For example, LENS, by default, utilizes both a GTF file for transcript annotations and a CTAT trinity reference for gene fusion detection through STAR-Fusion [9]. Both of these references are versioned and will not only have transcript-level information, but version-specific transcript information. As a result, users should ensure these references (as well as any other versioned references) are compatible with each other such that transcript and transcript *versions* within LENS are consistent among antigen sources.

7.2 Running a Modified Workflow

Users can run their modified workflow through RAFT using the following command:

Running a Modified RAFT Project

```
raft.py run-workflow -p my-lens-project
```

This command will create an augmented Nextflow command and execute it per the user's Nextflow configuration.

8 Examining RAFT Output Files

RAFT produces a variety of output files which are available for review in the project's /outputs directory. Patient-level and sample-level output files are generally published using a < DATASET > / < PATIENT_NAME > / < RUN_NAME > directory hierarchy. For example, the quant.sf output file from Salmon for Patient01's sample nr-123 in dataset MyDataset would be located in ./outputs/samples/MyDataset/Patient01/nr-123/salmon_quant [18].

Generally speaking, the Unix find command may be the easiest way to quickly find and analyze files from the outputs directory. For example, if a user wants to count the number of lines in each sample's MuTect2 VCF, then they can run the following line:

Using find to Interact with RAFT Output Files

```
find ../projects/<PROJECT_ID>/outputs -name "
    *mutect2*vcf" \
-exec wc -l {} \;
```

LENS-specific reports can be found in the /outputs/lens directory. These reports should be tab-separated text files (.tsv) and should be viewable within Microsoft Excel. Next, we will go over relevant columns from the report that may be useful for prioritizing tumor antigens.

9 Understanding LENS Output Reports

LENS generates a variety of reports depending on the user-specific parameters provided. These reports are subject to change as LENS development continues, but generally speaking, many columns are highly relevant to tumor antigen interpretation and are not expected to change. Here, we provide an overview of potentially useful columns from LENS reports. We begin with columns that help identify peptide-MHCs (pMHCs), then describe columns relevant for ranking and prioritizing pMHCs, and end with antigen source-specific columns used for further filtering.

9.1 pMHC Identifying Columns

Each line within the LENS report represents a unique peptide-MHC complex predicted by LENS. The columns relevant for describing a pMHC are **allele**, **peptide**, **identity**, and **antigen source**. The **allele** column contains the HLA allele from the patient that combines with the **peptide** to create the pMHC. The **identity** column contains a unique checksum that can be used to trace the genomic origin of the pMHC for debugging purposes. Finally, the **antigen source** column describes the type of pMHC being described. As of the time of this writing, antigen sources include SNV, INDEL, SPLICE, FUSION, CTA/SELF, VIRUS, and ERV.

9.2 pMHC Descriptive Columns

The factors associated with a suitably targetable tumor antigen remain a topic of debate within the literature [24]. Generally speaking, pMHCs with higher binding affinity, higher stability, higher relative abundance, higher clonality, and higher dissimilarity

from the self-proteome are thought to be most effective for targeting. LENS is capable of providing metrics for all of these factors so users can develop their own strategy for prioritizing tumor antigens. LENS also provides a priority score as a starting point for users.

Specifically, binding affinity is, by default, calculated by NetMHCpan and MHCflurry [17, 21]. NetMHCpan provides raw scores, percent rank values (relative to a set of random natural peptides), and binding affinity (measured in nanomolar). MHCflurry provides information on binding affinity but also includes both proteasomal processing and cell surface presentation scores.

Binding stability is calculated by NetMHCstabpan [20]. NetMHCstabpan provides a stability percent score, percent rank, and a binding half-life (in hours).

Dissimilarity and foreignness are defined by antigen.garnish [22]. Dissimilarity is a measure of how "non-self" peptides are relative to the human proteome, while foreignness is a measure of how similar a peptide is to known immunogenic peptides in the Immune Epitope Database (IEDB) [14]. LENS also calculates agretopicity by BLASTing peptides against the human proteome to discover the closest match and performing both NetMHCpan and MHCflurry binding affinity calculations against the match [1]. Specifically, the agretopicity value is defined as the ratio of mutant binding affinity to wild-type binding affinity ratio. Agretopicity remains a debated metric, but some evidence suggests high agretopicity peptides are rarely presented in wild-type form (and thus have a lower probability of central tolerance) which should boost the immunogenicity performance of the mutant form [22].

Tumor antigens are quantified by calculating the reads supporting the peptide's coding sequence from the tumor RNA BAM. Quantification algorithms are defined within the LENS manuscript [27].

Cancer cell fraction (CCF) is provided by LENS through means of somatic variant calling and CNVKit copy number alteration analysis [26]. Cancer cell fraction measures the clonality of a predicted tumor antigen such that tumor antigens with higher cancer cell fractions are present in more of the tumor's cells and may be better targets for therapy. LENS currently only supports cancer cell fraction calculations for SNVs and InDels.

Finally, LENS also provides a simple prioritization score as the **priority_score** column. This score takes relative antigen read support, binding affinity, and cancer cell fraction into account. Specifically, binding affinity values are transformed using:

$$\frac{abs(\chi - 1000)}{1000}.$$

Tumor antigen read support is log-transformed and then normalized by dividing each quantification value by the maximum observed count such that quantification values range from $[0, 1]$. Cancer cell fraction is unmodified as it is already bound between $[0, 1]$. The priority score is then calculated as

$$S = pMHC_{BA} * pMHC_{RS} * pMHC_{CCF},$$

where $pMHC_{BA}$ is the transformed binding affinity, $pMHC_{RS}$ is the log-transformed and normalized read support, and $pMHC_{CCF}$ is the estimated cancer cell fraction. We have also developed an alternate metric, **priority_score_no_ccf**, which is calculated without the cancer cell fraction value. This alternate metric serves as an antigen source agnostic value that should be available for all peptides.

9.3 SNV- and InDel-Specific Columns

SNV and InDel tumor antigens are generated from somatic mutations occurring within canonical transcripts which result in altered protein content. As such, LENS provides several useful columns that users can use to further investigate predicted SNV and InDel peptides. More information can be found in Table 2.

9.4 Fusion-Specific Columns

Gene fusions are a subset of structural variants that can produce novel, non-self-sequences. LENS provides several columns relevant to understanding the gene fusions detected within a patient. More information can be found in Table 3.

9.5 Splice-Specific Columns

Tumor-specific splice variants arise from noncanonical splicing during RNA maturation. These events can result in a variety of scenarios including intron skipping and the inclusion of cryptic exons. LENS provides the columns **gene** with the name of the gene harboring the splice variant as well as **tumor_splice** which contains the tumor-specific splice coordinates.

Table 2
LENS SNV and InDel output columns

Column	Description
gene_name	Gene name
transcript_id	Transcript identifier
mut_aa_pos	Mutation position within predicted peptide
mut_aa_range	Range of mutant positions within predicted peptide
variant_coords	Variant genomic position (chromosome:position)
variant_position_in_cds	Variant position within transcript coding sequence

Table 3
LENS fusion output columns

Column	Description
fusion_type	Fusion type (in-frame or frameshift)
fusion_left_breakpoint	Genomic coordinates of left breakpoint
fusion_left_gene	Gene name on the left of breakpoint
fusion_left_transcript	Transcript identifier on the left of breakpoint
fusion_right_breakpoint	Genomic coordinates of right breakpoint
fusion_right_gene	Gene name on the right of breakpoint
fusion_right_transcript	Transcript identifier on the right of breakpoint
fusion_annotation	Fusion annotation informative, if available
variant_position_in_cds	Variant position within transcript coding sequence

Table 4
LENS ERV output columns

Column	Description
erv_geve_annot	gEVE database identifier
erv_hervq_region	hERVQuant associated region, if any
erv_mtec_exp_status	True if expressed in medullary thymic endothelial cells, false otherwise
erv_norm_exp_statu	True if expressed in normal tissues, false otherwise
erv_hervq_region_proteins_list	List of ERV proteins present in hERVQuant region
erv_hervq_erv_uniq_proteins_counts	Count of unique ERV proteins present in hERVQuant region
erv_hervq_region_avg_exp_corr	Average correlated expression among ERV proteins present in hERVQuant region
erv_normed_erv_orf_confidence_score	Normalized confidence score between 0.0 and 1.0—closer to 1.0 is better

9.6 ERV-Specific Columns

LENS utilizes the gEVE database for defining the coordinates of ERVs for each species. This database relies upon computational prediction for ERV annotation and is, as a result, prone to false positive entries. LENS provides both identifying columns (**erv_geve_annot**), several descriptive columns, as well as scoring column. More information can be found in Table 4.

Table 5
LENS CTA/self-antigen output columns

Column	Description
gene_name	Gene name
transcript_id	Transcript identifier
mtec_tpm	Expression (TPM) of transcript in medullary thymic endothelial cells
mtec_num_reads	Expression (number of reads) of transcript in medullary thymic endothelial cells
gene_detectable_normal_tissues	List of normal tissues transcript is detectable in (Human Protein Atlas [19])
gene_main_subcellular_location	Subcellular location of gene product

9.7 Cancer-Testis Antigens- and Self-Antigens-Specific Columns

Cancer-testis antigens are derived from transcripts physiologically expressed and translated within the testis; however, they can become aberrantly expressed within tumors under some conditions. The testis is an immune-privileged tissue, and thus the presentation of peptides derived from these transcripts on the cell surface by MHC molecules has the potential to prompt an immune response. More information can be found in Table 5.

10 Generating a RAFT Package for Analysis Storage

A RAFT package is a minimal, yet sufficient, collection of metadata from a RAFT project used for regenerating projects. Users can create a RAFT package after their LENS project is completed using the command:

Generating a RAFT Package

```
raft.py package-project -p <PROJECT_ID> \
                    -o <rft_pkg_name>
```

The resulting RAFT package (.rftpkg) will be available within the project's /rftpkgs directory (./raft/projects/<PROJECT_ID>/rftpkgs). This file can be used for regenerating the project in the future.

11 Loading a Project from a RAFT Package

Loading a RAFT project from a RAFT package can be performed using:

Generating a RAFT Project from a rftpkg File

```
raft.py load-project -p <PROJECT_ID> -r /path/
to
    /rftpkg
```

Note that the Project Identifier can be different than the project identifier originally used with the project. Users loading a project must have the relevant reference, metadata, and fastqs in their global RAFT directory (./raft/references, ./raft/metadata, and ./raft/fastqs, respectively). The project can be run using the raft.py run-workflow like a standard RAFT project.

12 Further Help

More information about running RAFT can be found at https://useraft.io, and more information on LENS can be found at https://uselens.io. Interactive help is also available on our Slack server: https://tinyurl.com/raft-slack. Nextflow experience can help with debugging workflow errors that may occur. We encourage users to learn more about Nextflow using free training provided by its developers at https://training.nextflow.io/.

References

1. Altschul SF, Madden TL, Schäffer AA, Zhang J, Zhang Z, Miller W, Lipman DJ (1997) Gapped blast and PSI-blast: a new generation of protein database search programs. Nucleic Acids Res 25(17):3389–3402
2. Benjamin D, Sato T, Cibulskis K, Getz G, Stewart C, Lichtenstein L (2019) Calling somatic SNVs and Indels with Mutect2. BioRxiv 861054
3. Bushnell B (2014) BBMap: a fast, accurate, splice-aware aligner. Tech. rep., Lawrence Berkeley National Lab.(LBNL), Berkeley, CA (United States) (2014)
4. Cingolani P, Platts A, Coon M, Nguyen T, Wang L, Land S, Lu X, Ruden D (2012) A program for annotating and predicting the effects of single nucleotide polymorphisms, SnpEff: SNPs in the genome of Drosophila melanogaster strain w1118; iso-2; iso-3. Fly 6(2):80–92
5. Di Tommaso P, Chatzou M, Floden EW, Barja PP, Palumbo E, Notredame C (2017) Nextflow enables reproducible computational workflows. Nat Biotechnol 35(4):316–319
6. Dobin A, Davis CA, Schlesinger F, Drenkow J, Zaleski C, Jha S, Batut P, Chaisson M, Gingeras TR (2013) Star: ultrafast universal RNA-seq aligner. Bioinformatics 29(1):15–21
7. Feola S, Chiaro J, Martins B, Cerullo V (2020) Uncovering the tumor antigen landscape: what to know about the discovery process. Cancers 12(6):1660
8. Griffiths DJ (2001) Endogenous retroviruses in the human genome sequence. Genome Biol 2(6):reviews1017–1

9. Haas BJ, Dobin A, Stransky N, Li B, Yang X, Tickle T, Bankapur A, Ganote C, Doak TG, Pochet N et al (2017) Star-fusion: fast and accurate fusion transcript detection from RNA-seq. BioRxiv 120295
10. Hodi FS, Kluger H, Sznol M, Carvajal R, Lawrence D, Atkins M, Powderly J, Sharfman W, Puzanov I, Smith D et al (2016) Abstract ct001: durable, long-term survival in previously treated patients with advanced melanoma (MEL) who received nivolumab (NIVO) monotherapy in a phase I trial. Cancer Res 76(14_Supplement):CT001–CT001
11. Jo JO, Kang YJ, Ock MS, Song KS, Jeong MJ, Jeong SJ, Choi YH, Ko EJ, Leem SH, Kim S et al (2016) Expression profiles of HERV-K Env protein in normal and cancerous tissues. Genes Genom 38:91–107
12. Katsikis PD, Ishii KJ, Schliehe C (2024) Challenges in developing personalized neoantigen cancer vaccines. Nat Rev Immunol 24(3): 213–227
13. Kim S, Scheffler K, Halpern AL, Bekritsky MA, Noh E, Källberg M, Chen X, Kim Y, Beyter D, Krusche P et al (2018) Strelka2: fast and accurate calling of germline and somatic variants. Nat Methods 15(8):591–594
14. Kim Y, Ponomarenko J, Zhu Z, Tamang D, Wang P, Greenbaum J, Lundegaard C, Sette A, Lund O, Bourne PE et al (2012) Immune epitope database analysis resource. Nucleic Acids Res 40(W1):W525–W530
15. Koboldt DC, Zhang Q, Larson DE, Shen D, McLellan MD, Lin L, Miller CA, Mardis ER, Ding L, Wilson RK (2012) VarScan 2: somatic mutation and copy number alteration discovery in cancer by exome sequencing. Genome Res 22(3):568–576
16. Lavie L, Kitova M, Maldener E, Meese E, Mayer J (2005) CpG methylation directly regulates transcriptional activity of the human endogenous retrovirus family HERV-K (HML-2). J Virol 79(2):876–883
17. O'Donnell TJ, Rubinsteyn A, Laserson U (2020) MHCflurry 2.0: improved pan-allele prediction of MHC class I-presented peptides by incorporating antigen processing. Cell Syst 11(1):42–48
18. Patro R, Duggal G, Love MI, Irizarry RA, Kingsford C (2017) Salmon provides fast and bias-aware quantification of transcript expression. Nat Methods 14(4):417–419
19. Pontén F, Jirström K, Uhlen M (2008) The human protein atlas—a tool for pathology. J Pathol A J Pathol Soc Great Br Ireland 216(4):387–393
20. Rasmussen M, Fenoy E., Harndahl M, Kristensen AB, Nielsen IK, Nielsen M, Buus S (2016) Pan-specific prediction of peptide–MHC class I complex stability, a correlate of t cell immunogenicity. J Immunol 197(4):1517–1524
21. Reynisson B, Alvarez B, Paul S, Peters B, Nielsen M (2020) NetMHCpan-4.1 and netMHCIIpan-4.0: improved predictions of MHC antigen presentation by concurrent motif deconvolution and integration of MS MHC eluted ligand data. Nucleic Acids Res 48(W1):W449–W454
22. Richman LP, Vonderheide RH, Rech AJ (2019) Neoantigen dissimilarity to the self-proteome predicts immunogenicity and response to immune checkpoint blockade. Cell Syst 9(4):375–382
23. Rojas LA, Sethna Z, Soares KC, Olcese C, Pang N, Patterson E, Lihm J, Ceglia N, Guasp P, Chu A et al (2023) Personalized RNA neoantigen vaccines stimulate t cells in pancreatic cancer. Nature 618(7963):144–150
24. Schmidt J, Smith AR, Magnin M, Racle J, Devlin JR, Bobisse S, Cesbron J, Bonnet V, Carmona SJ, Huber F et al (2021) Prediction of neo-epitope immunogenicity reveals TCR recognition determinants and provides insight into immunoediting. Cell Rep Med 2(2): 100194
25. Suri A (2006) Cancer testis antigens–their importance in immunotherapy and in the early detection of cancer. Expert Opin Biol Therapy 6(4):379–389
26. Talevich E, Shain AH, Botton T, Bastian BC (2016) CNVkit: genome-wide copy number detection and visualization from targeted DNA sequencing. PLoS Comput Biol 12(4): e1004873
27. Vensko SP, Olsen K, Bortone D, Smith CC, Chai S, Beckabir W, Fini M, Jadi O, Rubinsteyn A, Vincent BG (2023) Lens: landscape of effective neoantigens software. Bioinformatics 39(6):btad322
28. Wells DK, van Buuren MM, Dang KK, Hubbard-Lucey VM, Sheehan KC, Campbell KM, Lamb A, Ward JP, Sidney J, Blazquez AB et al (2020) Key parameters of tumor epitope immunogenicity revealed through a consortium approach improve neoantigen prediction. Cell 183(3):818–834

INDEX

A

Algorithm 106, 113, 114, 122, 134, 156, 159, 214, 221, 237, 260, 268, 284, 285, 337
Amazon Web Services (AWS) 52
Analysis
  flux balance 187, 188
  functional pathway inference 203–227
  principal component 145, 163, 280, 281
  signature 94, 127, 133
Antigen
  tumor 75, 76, 319–341
  tumor-associated 75, 76, 320
Attention mechanism 247–256

B

Bioinformatics
  cancer 2
Biological mutation signal 125, 126, 128, 132, 135
Biomarkers 23, 247, 303–316

C

Cancer
  bioinformatics 2
  genomics 1–17, 20, 23, 38–40, 48, 89, 247, 249
  immunology 77, 90
  informatics 47–72
  precision medicine 284
  survival 304, 306, 308–310, 313
Cell signaling 204
Cloud 3, 5, 40, 47, 49, 51, 145
Common Workflow Language (CWL) 48, 49, 51, 54, 56, 61, 71
Community connectivity 178
Complex genomic rearrangements 105–122
Continental populations 157
Copy number variation (CNV) 23, 106–108, 112–114, 120, 121, 140, 141, 153
Cordial 204–208, 212, 213
CRISPR-Cas9 ..... 208, 210, 226

D

Data
  structures 3, 10, 32–35, 39, 157, 205
  synthetic 159, 163, 267
Database 77, 86, 93–95, 154, 191, 204, 234, 239–244, 250, 251, 255, 273, 274, 280, 282, 285, 304, 307, 309, 311–316, 339
De-noising 125
Docker 38, 39, 48–54, 56, 57, 68, 69, 145, 296, 321, 322, 332, 333
Drug
  response prediction 273–286
  sensitivity prediction 259–261
  synergy prediction 273–275, 277–279, 282, 283, 285

E

Epigenomics 24, 248, 249
Epitopes 75–89

F

Formalin-fixed, paraffin-preserved (FFPE) samples 125, 126, 128, 130, 132, 133, 292–293
Functional pathway inference analysis (FPIA) 203–227

G

Gene
  dependency 204, 285
  driver 247–256
  expression 24, 50, 51, 65, 103, 137–139, 145–147, 177, 180, 184, 187–201, 204, 231–235, 239–244, 249, 250, 255, 260, 261, 266, 276–278, 280, 281, 304, 310, 313
  network 248, 249, 252, 256
  regulation 239–242, 244
Genetic ancestry 153–174
Genome instability 93, 105, 153

Alexander Krasnitz and Pascal Belleau (eds.), *Cancer Bioinformatics*, Methods in Molecular Biology, vol. 2932,
https://doi.org/10.1007/978-1-0716-4566-6,

Genotyping 144–146, 155, 308

I

Immuno-oncology 320
Immunotherapy 25, 76, 88, 150, 319

L

Learning
  deep 248, 259, 260, 273–286, 292, 293, 295, 297–299
  machine 25, 248, 260, 268, 274–276, 280, 284, 285, 292, 294, 295, 319

M

Mendelian randomization (MR) 303, 304, 306, 309, 310, 313
Metabolism 187, 188
Multi-omics data 33, 204, 248, 249, 254, 278
Mutational processes 125, 126, 135
Mutations 10, 12, 13, 22, 87, 94–97, 102, 103, 125–135, 153, 155, 247, 249, 254, 277, 280, 291–295, 297–299, 320, 338

N

Neoantigen
  workflow 321
Network
  convolutional neural 259–270, 295
  gene 248, 249, 252, 256
  graph attention 248, 249, 252–254
Non-B DNA
  Burden in Cancer (NBBC) 93–103
Noncoding RNA
  gene regulatory pairs (NGP) 231–244

O

Omics 248–250, 254, 277
Ontology 20–23
Open source software 2

P

Pathway 26, 150, 180, 195, 203, 204, 224, 234, 239, 242, 244, 249–252, 255, 279, 285
Probabilistic modeling 138, 292
Protein-protein interaction (PPI) 248–251, 279, 285
Pruning 178, 185

R

R (programming language) 2, 3, 5, 10, 14, 24, 33, 38–44, 106, 107, 118, 144, 156, 157, 178, 180, 188, 189, 194, 204–208, 232–234, 243, 244, 251, 307, 309
Reproducibility 3, 48, 49, 67, 68, 163, 321
Risk factors 303, 304, 306, 309, 313, 316
RNA
  interference (RNAi) 204, 208, 210
  sequencing
    bulk 25, 26, 188–195, 198
    single cell 50, 138, 196–201, 243

S

Sequencing
  next-generation 48, 292
  RNA
    bulk 25, 26, 188–195, 198
    single cell 50, 138, 196–201, 243
  single cell whole genome 138–144, 146–149
Single cell
  sequencing
    RNA 50, 138, 196–201, 243
    whole genome 138–144, 146–149
Software
  design 39
  open source 2
Somatic variant calling 291–296, 299, 325, 331, 337
Spatial transcriptomics 27–33, 177–185
Starfish (algorithm) 105–122
Structural variation 105, 144

T

The Cancer Genome Atlas (TCGA) 7–14, 102, 154, 155, 239, 241, 242, 247, 249, 273, 281
Transcriptomics
  spatial 27–33, 177–185
Trimming 178, 180–185
Tumor
  evolution 138
  microenvironment 25–27, 184, 187, 188, 196, 199, 201

W

Workflow
  neoantigen 321