

UCLA

UCLA Electronic Theses and Dissertations

Title

Advancing Statistical Rigor in Single-Cell and Spatial Omics Analysis Through In Silico Control Data

Permalink

<https://escholarship.org/uc/item/99c4j1s4>

Author

Yan, Guanao

Publication Date

2025

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA
Los Angeles

Advancing Statistical Rigor in Single-Cell and Spatial Omics Analysis
Through In Silico Control Data

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Statistics

by

Guanao Yan

2025

© Copyright by
Guanao Yan
2025

ABSTRACT OF THE DISSERTATION

Advancing Statistical Rigor in Single-Cell and Spatial Omics Analysis Through In Silico Control Data

by

Guanao Yan

Doctor of Philosophy in Statistics

University of California, Los Angeles, 2025

Professor Jingyi Li, Chair

Over the past decade, single-cell and spatial transcriptomics technologies have transformed our ability to study cellular diversity and tissue organization. These advances have led to the rapid development of computational methods for analyzing high-dimensional omics data. However, benchmarking these methods and ensuring their statistical rigor remain challenging, largely due to the absence of realistic synthetic data with ground truths and the conceptual ambiguity in defining key biological features such as spatially variable genes (SVGs). This dissertation addresses these gaps through two simulation frameworks and a comprehensive review that improve the statistical rigor and interpretability of tool development and evaluation.

My first project introduces scReadSim, a simulator designed to generate realistic synthetic data for single-cell RNA sequencing (scRNA-seq) and chromatin accessibility profiling (scATAC-seq). It produces simulated sequencing reads in standard formats by mimicking the characteristics of real datasets, while allowing users to specify key ground truths, such as transcript abundance for scRNA-seq and cell-type-specific open chromatin regions for scATAC-seq. scReadSim supports flexible simulation settings, including varying cell numbers and sequencing depths, and enables systematic benchmarking of preprocessing tools. Using scReadSim, we show that UMI-tools achieves higher accuracy in transcript quantification for scRNA-seq, while HMMRATAC and MACS3 perform best in peak calling for

scATAC-seq.

My second project presents scIsoSim, a simulator that generates single-cell RNA sequencing data with known isoform structures and their corresponding expression levels. In gene expression, a single gene can give rise to multiple isoforms—different versions of RNA transcripts—through a biological process called alternative splicing, where segments of RNA are included or excluded in various combinations. scIsoSim supports widely used experimental protocols, including Smart-seq2 and 10x Genomics 3' and 5' platforms, and captures realistic splicing patterns observed in real datasets. This tool enables systematic evaluation of computational methods for quantifying isoform expression and detecting alternative splicing events. Benchmarking results show that bulk RNA-seq tools, such as Salmon, perform accurately on Smart-seq2 data with high computational efficiency. In contrast, Scasa—the only existing tool for 10x 3' data—shows limited accuracy due to sparse data. Among splicing analysis tools, brie demonstrates better overall accuracy than outrigger but is less effective in detecting cell-specific splicing events.

My third project is a review of 34 state-of-the-art SVG detection methods for spatial transcriptomics data. The review introduces a new categorization framework that defines SVGs as overall, cell-type-specific, or spatial-domain-marker genes, based on their spatial expression patterns and analytic objectives. It summarizes the underlying assumptions and statistical hypothesis tests used by each method, and discusses trade-offs between power and specificity. The review also identifies limitations in existing benchmarks, such as inappropriate method comparisons and oversimplified simulation designs, and calls for category-specific benchmarking using well-annotated datasets and realistic simulators.

The dissertation of Guanao Yan is approved.

Michael J. Gandal

Sudipto Banerjee

Frederic P. Schoenberg

Jingyi Li, Committee Chair

University of California, Los Angeles

2025

To my parents.

TABLE OF CONTENTS

1	Introduction	1
1.1	scReadSim: a single-cell RNA-seq and ATAC-seq read simulator	2
1.2	scIsoSim: simulating single-cell RNA-seq reads with full-length isoform ground truths	2
1.3	Categorization of 34 computational methods to detect spatially variable genes from spatially resolved transcriptomics data	3
2	scReadSim: a single-cell RNA-seq and ATAC-seq read simulator	4
2.1	Introduction	4
2.2	Methods	6
2.2.1	The scReadSim method	6
2.2.2	Data analysis strategies	19
2.3	Results	29
2.3.1	scReadSim generates realistic synthetic scRNA-seq reads	29
2.3.2	scReadSim generates realistic synthetic scATAC-seq reads	30
2.3.3	scReadSim allows varying cell numbers and sequencing depths	32
2.3.4	Application 1: Benchmarking UMI deduplication tools	33
2.3.5	Application 2: Benchmarking scATAC-seq peak-calling tools	34
2.4	Discussion	35
2.5	Code and data availability	37
2.6	Acknowledgements	38
2.7	Figures	39
2.8	Supplementary materials	43

S2.8.1	Supplementary note	43
S2.8.2	Supplementary figures	45
S2.8.3	Supplementary tables	62
3	scIsoSim: simulating single-cell RNA-seq reads with full-length isoform ground truths	63
3.1	Introduction	63
3.2	Methods	66
3.2.1	The scIsoSim method	66
3.2.2	Data analysis strategies	71
3.3	Results	78
3.3.1	scIsoSim simulates realistic scRNA-seq reads with full-length isoform ground truths	78
3.3.2	Application 1: benchmarking bulk-level isoform quantification tools using synthetic Smart-seq2 scRNA-seq data	79
3.3.3	Application 2: evaluating Scasa for isoform quantification using synthetic 10x 3' scRNA-seq data	81
3.3.4	Application 3: benchmarking single-cell AS quantification tools	82
3.4	Discussion	83
3.5	Code and data availability	85
3.5.1	Software and code	85
3.5.2	Datasets	87
3.6	Acknowledgements	87
3.7	Figures	88
3.8	Supplementary materials	93
S3.8.1	Supplementary figures	93

S3.8.2	Supplementary tables	99
4	Categorization of 34 computational methods to detect spatially variable genes from spatially resolved transcriptomics data	101
4.1	Introduction	101
4.2	Methods for detecting overall SVGs	106
4.2.1	Euclidean-space-based methods	106
4.2.2	Graph-based methods	115
4.2.3	Relationship between Euclidean-space-based methods and graph-based methods	120
4.2.4	Applications of overall SVGs	122
4.3	Methods for detecting cell-type-specific SVGs	124
4.3.1	Applications of cell-type-specific SVGs	126
4.4	Methods for detecting spatial-domain-marker SVGs	127
4.4.1	Applications of spatial-domain-marker SVGs	128
4.5	Theoretical characterization of SVG detection methods that use frequentist hypothesis tests	129
4.5.1	Dependence tests	130
4.5.2	Regression tests	132
4.6	Discussion and future directions	138
4.6.1	Power, specificity, and scalability of SVG detection methods	138
4.6.2	Future direction 1: Accommodating differences in SRT technology and tissue types, and supporting the analysis of multi-sample SRT data	141
4.6.3	Future direction 2: Enhancing statistical rigor and method validation	143
4.6.4	Guidance for future benchmark studies	144
4.7	Acknowledgements	146

4.8	Figures	147
4.9	Tables	153
4.10	Supplementary materials	157
	S4.10.1Supplementary figures	157
	S4.10.2Supplementary tables	160
5	Summary and future directions	162
5.1	scReadSim: a single-cell RNA-seq and ATAC-seq read simulator	162
5.2	scIsoSim: simulating single-cell RNA-seq reads with full-length isoform ground truths	163
5.3	Categorization of 34 computational methods to detect spatially variable genes from spatially resolved transcriptomics data	164

LIST OF FIGURES

2.1	Workflows of scReadSim’s scRNA-seq (a) and scATAC-seq (b) read generation.	39
2.2	Benchmark of UMI deduplication tools (a–d) and peak calling tools (e–g) using scReadSim’s synthetic reads.	40
2.3	scReadSim outperforms existing scRNA-seq and scATAC-seq read simulators. .	41
2.4	scReadSim generates realistic scATAC-seq sequencing reads with ground-truth peaks.	42
2.5	Relationships between single-cell sequencing data analysis tools and simulators.	45
2.6	Workflow of scReadSim for generating scATAC-seq reads with user-designed open chromatin regions (i.e., ground-truth peaks).	46
2.7	The peak-mapping step in scReadSim’s data generation with user-designed open chromatin regions (i.e., ground-truth peaks).	47
2.8	Synthetic cells generated by SCAN-ATAC-Sim do not mimic real cells at the read-count level.	48
2.9	scReadSim’s synthetic data resemble real mouse 10x single-cell Multiome dataset (the RNA-seq modality only) [3] at the UMI-count level.	49
2.10	scReadSim’s synthetic data resemble real mouse 10x single-cell Multiome dataset (the RNA-seq modality only) [3] at the read-sequence level.	50
2.11	scReadSim’s synthetic data resemble real mouse 10x single-cell Multiome dataset (the ATAC-seq modality only) [3] at the read-count level.	51
2.12	scReadSim’s synthetic data resemble real sci-ATAC-seq data [5] at the read-count level.	52
2.13	scReadSim’s synthetic data resemble real mouse 10x single-cell Multiome dataset (the ATAC-seq modality only) [3] at the read-sequence level.	53
2.14	scReadSim’s synthetic data resemble real sci-ATAC-seq data [5] at the read-sequence level.	54

2.15	IGV genome browser [4] visualization of scReadSim’s synthetic sci-ATAC-seq reads with user-designed ground-truth (g.t.) peaks (bottom track).	55
2.16	scReadSim’s synthetic data with user-designed open chromatin regions resemble real sci-ATAC-seq data [5] at the read-sequence level.	56
2.17	scReadSim allows varying cell number and sequencing depth as input parameters, with the corresponding synthetic data visualized in the IGV genome browser [4].	57
2.18	Benchmark of UMI deduplication tools using scReadSim’s synthetic scRNA-seq reads.	58
2.19	UMAP visualizations of the ground-truth UMI count matrix and each deduplication tool’s output UMI count matrix.	59
2.20	Benchmark of peak-calling tools (MACS3, HOMER, and HMMRATAC) using scReadSim’s synthetic scATAC-seq reads generated from user-designed ground-truth peaks (i.e., open chromatin regions).	60
2.21	Benchmark of peak-calling tools (MACS3, SEACR, HOMER, and HMMRATAC) using scReadSim’s synthetic scATAC-seq reads generated from user-designed ground-truth peaks (i.e., open chromatin regions).	61
3.1	scIsoSim simulates scRNA-seq with full-length isoform ground truths.	88
3.2	scIsoSim generates realistic isoform-level scRNA-seq data.	89
3.3	Benchmark isoform quantification tools using synthetic Smart-seq2 scRNA-seq data.	90
3.4	Benchmark isoform quantification tools using synthetic 10x 3’ scRNA-seq data. .	91
3.5	Benchmarking AS quantification tools using synthetic Smart-seq2 data.	92
3.6	Illustration of sequencing read distributions for three scRNA-seq protocols: 10x 5’ (paired-end), Smart-seq2, and 10x 3’.	93
3.7	Three major steps of scIsoSim for simulating scRNA-seq reads.	94

3.8	scIsoSim’s synthetic data resemble real 10x 3’ scRNA-seq dataset profiling human lung adenocarcinoma cells [1].	95
3.9	scIsoSim’s synthetic data resemble real 10x 5’ single-cell NSCLC tumor dataset [2].	96
3.10	Benchmarking workflow of scIsoSim for isoform quantification tools using Smart-seq2 scRNA-seq data.	97
3.11	Benchmarking workflow of scIsoSim for AS quantification tools using Smart-seq2 scRNA-seq data.	98
4.1	General analysis workflows of single-cell transcriptomic data and spatially resolved transcriptomic (SRT) data.	148
4.2	Conceptual visualization of three SVG categories and timeline of 34 SVG detection methods.	149
4.3	A hierarchical summary of 34 SVG detection methods.	150
4.4	Conceptual diagram for Section 4.5 “Theoretical characterization of SVG detection methods that use frequentist hypothesis tests.”	151
4.5	Synthetic spatial patterns oversimplify the spatial patterns observed in real SRT data.	152
4.6	Methodological traits of 34 SVG detection methods.	158
4.7	Characteristics of 23 frequentist-hypothesis-tests-based SVG detection methods.	159

LIST OF TABLES

S2.1	Comparison of scReadSim, minnow, and SCAN-ATAC-Sim	62
S3.1	Comparison of scRNA-seq protocols for isoform quantification	99
S3.2	Comparison of scRNA-seq Read Simulators and Procedures	100
4.1	Classification of 34 SVG detection methods along with statistical inference types	154
4.2	Definitions of concepts for SVGs and frequentist statistical tests	155
4.3	Mathematical notations and interpretations	156
S4.4	Summary of three existing benchmark studies for SVG detection methods ¹ . . .	161

ACKNOWLEDGMENTS

This dissertation grew out of countless conversations, patient guidance, and steady encouragement from many corners of my life.

My deepest gratitude goes to my advisor, Dr. Jingyi Jessica Li. Her rigorous thinking, high standards, and steady confidence in my ideas shaped every chapter of this work and showed me what generous, principled mentorship looks like. Beyond scholarship, she treats her students as friends; together we have shared not only academic milestones but also music, art, and laughter that have made this journey richer and more memorable. The clarity, curiosity, and kindness she embodies will continue to guide my own journey, inspiring me to approach research and mentorship with the same spirit she has shown me.

I am equally thankful to my doctoral committee members, Dr. Michael J. Gandal, Dr. Sudipto Banerjee, and Dr. Frederic P. Schoenberg, whose thoughtful questions, constructive critiques, and consistent support sharpened my analysis and broadened my perspective.

I thank all current and former members of the JSB Lab for the conversations, feedback, and suggestions that strengthened this dissertation. I am especially grateful to Dr. Xinzhou Ge, Dr. Tianyi Sun, and Zhiqian Zhai—your friendship and camaraderie enriched my experience in the lab. I also appreciate the collaboration of Dr. Dongyuan Song, Weijian Wang, and Zhiyin Liu on several chapters, the valuable contributions of Shuo Hua, and Dr. Chongzhi Zang’s constructive feedback on the first chapter. Finally, working with Dr. Mark D. Biggin taught me a great deal; his dedication to rigorous, principled research continues to motivate my own scientific journey.

Last, I owe heartfelt thanks to my parents, Linlin Wu and Xiaolou Yan, whose steadfast love and belief in me have been the foundation of this journey. I am also grateful to Hui Huang for the support and encouragement she offered throughout my doctoral studies; may the road ahead bring her every happiness and success. I also thank the friends who have stood by me, adding laughter, perspective, and steady companionship along the way.

VITA

2013–2017	B.S. in Mathematics and Applied Mathematics, Department of Mathematical Science, B.Ec in Economics, Department of Economics, Shandong University
2017–2020	M.Sc in Probability and Mathematical Statistics, School of Mathematical Sciences, Zhejiang University
2020–2025	Ph.D. in Statistics, Department of Statistics, University of California, Los Angeles

PUBLICATIONS

(The list includes only those works where I am the lead author during my Ph.D. studies.)

Yan, G., Hua, S. and Li, J.J. (2025). Categorization of 34 computational methods to detect spatially variable genes from spatially resolved transcriptomics data. *Nature Communications*, 16, 1141.

Yan, G., Li, J.J., and Biggin, M. (2024). Question-Score Identity Detection (Q-SID): A statistical algorithm to detect collusion groups with error quantification from exam question scores. *arXiv*, 2407.07420.

Yan, G., Song, D. and Li, J.J. (2023). scReadSim: a single-cell RNA-seq and ATAC-seq read simulator. *Nature Communications*, 14(1), 7428

CHAPTER 1

Introduction

Single-cell and spatial transcriptomic technologies have rapidly advanced our ability to study cellular heterogeneity and spatial organization at unprecedented resolution. Technologies such as single-cell RNA sequencing (scRNA-seq) [1] and single-cell Assay for Transposase-Accessible Chromatin using sequencing (scATAC-seq) have empowered researchers to quantify gene expression and chromatin accessibility at the level of individual cells, uncovering diverse cellular states, differentiation trajectories, and regulatory programs [2]. More recently, spatially resolved transcriptomics (SRT) has expanded this frontier by preserving the spatial context of gene expression, enabling systematic investigation into the tissue microenvironment, spatial domain structure, and cell-cell interactions [3, 4]. Together, these technologies have generated a vast amount of high-dimensional data, which in turn has led to the development of thousands of computational tools [5, 6] to analyze, interpret, and visualize such data.

Despite this proliferation of computational methods, two major challenges persist. First, due to the complexity of single-cell and spatial omics data, there is a lack of gold-standard ground truth datasets, making it difficult to objectively evaluate or compare the performance of different tools. Second, many existing analytical pipelines adopt procedures that lack statistical rigor, either relying on heuristic rules or making assumptions that are difficult to verify in real data applications. These issues raise critical questions about reproducibility, interpretability, and the reliability of downstream biological conclusions. To address these limitations, *in silico* synthetic data with known ground truths can serve as positive or negative controls for benchmarking and validation purposes.

This dissertation addresses the above challenges by proposing two simulation frameworks

and one review study. The first two projects focus on generating synthetic single-cell sequencing data to enable rigorous benchmarking of computational tools. The third project provides a comprehensive conceptual framework and classification of spatially variable gene (SVG) detection methods, clarifying a key task in spatial transcriptomics analysis.

1.1 scReadSim: a single-cell RNA-seq and ATAC-seq read simulator

The first part of the dissertation, described in Chapter 2, introduces scReadSim, a read-level simulator for both scRNA-seq and scATAC-seq data. Unlike existing simulators that only generate count matrices [7–12], scReadSim produces synthetic sequencing reads by learning from real data and outputs FASTQ or BAM files with associated ground truths. For scRNA-seq, scReadSim simulates unique molecular identifiers (UMIs), allowing benchmarking of UMI deduplication tools such as UMI-tools [13], STARsolo [14], CellRanger [15], and Alevin [16]. For scATAC-seq, scReadSim provides ground-truth chromatin accessibility peaks, supporting evaluation of peak-calling algorithms such as MACS3 [17], HOMER [18], HMMRATAC [19], and SEACR [20]. Through realistic data generation and flexible user controls (e.g., sequencing depth, cell number), scReadSim addresses a critical gap in benchmarking tools that operate directly on read-level data.

1.2 scIsoSim: simulating single-cell RNA-seq reads with full-length isoform ground truths

The second part of the dissertation, detailed in Chapter 3, presents scIsoSim, a simulator designed to generate realistic scRNA-seq reads with predefined isoform structures and splicing mechanisms. Accurate isoform quantification and alternative splicing (AS) analysis at the single-cell level is challenged by protocol-specific biases and a lack of ground truth [21–31]. Existing simulation approaches adapted from bulk RNA-seq data (e.g., RSEM [21], Kallisto + Splatter + Polyester [7, 22, 32]) fail to support droplet-based protocols or flex-

ible isoform specification [33]. scIsoSim overcomes these limitations by supporting major scRNA-seq protocols (Smart-seq2 [34], 10x 3' [15], and 10x 5') and simulating reads that reflect protocol-specific biases. The synthetic datasets generated by scIsoSim allow benchmarking of both bulk-derived and single-cell-specific tools for isoform quantification and AS analysis. Using scIsoSim, we evaluated tools such as Salmon, Kallisto, Terminus, Scasa [35], Brie [36], and Outrigger [37], revealing protocol-specific performance differences and areas for improvement.

1.3 Categorization of 34 computational methods to detect spatially variable genes from spatially resolved transcriptomics data

The third part of the dissertation, described in Chapter 4, conducts a conceptual and methodological review of spatially variable gene (SVG) detection methods for spatial transcriptomics. While several benchmark studies have numerically compared methods [38–40], they often fail to clarify the underlying definitions and assumptions used across different approaches. This review introduces a novel three-category framework: overall SVGs, cell-type-specific SVGs, and spatial-domain-marker SVGs. Each category reflects a different biological purpose and statistical hypothesis, guiding tool selection based on study goals. We review 34 peer-reviewed methods and summarize their statistical paradigms, methodological properties, and downstream applications. By synthesizing insights from prior benchmarks and organizing methods according to their intended applications, this review aims to guide both practitioners and developers toward more interpretable and statistically grounded analyses.

In summary, my doctoral research focuses on improving statistical rigor in single-cell and spatial omics data analysis through the use of *in silico* control data. This dissertation introduces two simulation frameworks and a comprehensive conceptual review, detailed in Chapters 2, 3, and 4.

CHAPTER 2

scReadSim: a single-cell RNA-seq and ATAC-seq read simulator

2.1 Introduction

The development of single-cell sequencing technologies has enabled the characterization of genomic, epigenomic, and transcriptomic features of individual cells [2]. More than a thousand computational tools have been developed to analyze single-cell sequencing data [5], necessitating third-party benchmarking of computational tools. Realistic simulators are essential for fair benchmarking, because they can generate synthetic data that mimic real data and contain ground truths for evaluating computational tools.

Although many simulators have been developed for the two most popular single-cell sequencing technologies—single-cell RNA sequencing (scRNA-seq) and single-cell Assay for Transposase-Accessible Chromatin using sequencing (scATAC-seq), most simulators do not generate sequencing reads. Instead, they only simulate read counts in genes [7–11] or genomic regions [12], and we refer to them as “count simulators.” (See [41] for a comprehensive review of scRNA-seq count simulators.) As such, these count simulators cannot be used to benchmark read-level bioinformatics tools that process reads stored in a FASTQ or BAM file (Supplementary Fig. 2.5). Exemplary read-level tools include the unique molecular identifier (UMI) deduplication tools for scRNA-seq data [13–16] and the peak-calling tools for scATAC-seq data [17–20]. Benchmarking these tools requires simulators that can generate synthetic sequencing reads, and we refer to such simulators as “read simulators.”

For scRNA-seq, minnow [42] and Dropify [43] are the only two read simulators to our

knowledge. Only minnow has a publicly available software package, which inputs a UMI count matrix and a gene annotation file but does not learn from real scRNA-seq reads. Moreover, minnow generates scRNA-seq reads only from annotated genes, so it cannot resemble real scRNA-seq data in intergenic regions that might be transcribed.

For scATAC-seq, SCAN-ATAC-Sim [44] is the only read simulator to our knowledge. However, similar to minnow, SCAN-ATAC-Sim does not learn from real scATAC-seq reads. Instead, it generates scATAC-seq reads from bulk ATAC-seq reads under simplistic assumptions: treating genomic regions (peaks called from bulk ATAC-seq reads) as independent, with a maximum of 2 reads allowed in each genomic region per cell, irrespective of the region’s length. Moreover, SCAN-ATAC-Sim does not provide synthetic read sequences with quality scores in a FASTQ or BAM file. Rather, it generates a BED file that only contains read start and end positions, thereby limiting direct benchmarking of read-level tools that typically require input of a FASTQ or BAM file. Additionally, we found that installing the minnow and SCAN-ATAC-Sim software packages in C++ was non-trivial due to the specific operating system and compiler requirements.

Motivated by the limitations of existing read simulators (a detailed comparison of scReadSim with the existing read simulators is in Supplementary Table 1), we developed scReadSim as a scRNA-seq and scATAC-seq read simulator that generates realistic synthetic data by mimicking real data. scReadSim inputs a scRNA-seq or scATAC-seq BAM file and outputs synthetic reads in a FASTQ or BAM file (Fig. 2.1a–b). We verified that scReadSim generates synthetic scRNA-seq and scATAC-seq data that resemble real data at both the read-sequence and read-count levels. Moreover, scReadSim provides ground truths, including unique molecular identifier (UMI) counts for scRNA-seq and open chromatin regions for scATAC-seq. In particular, scReadSim enables users to specify cell-type-specific open chromatin regions for scATAC-seq read generation, and it also allows users to vary the cell number and sequencing depth of synthetic data.

In this work, we use two exemplary applications to highlight scReadSim’s utility as a benchmarking tool. First, for scRNA-seq, scReadSim mimics real data by first generating realistic UMI counts and then simulating reads. Hence, the synthetic UMI count matrix—an

intermediate output of scReadSim—serves as the ground truth for benchmarking scRNA-seq UMI deduplication tools such as UMI-tools [13], cellranger [15], STARsolo [14] and Alevin [16], which all process reads (in a FASTQ file) into a UMI count matrix. Our benchmarking results indicate that UMI-tools achieves the best accuracy, while Alevin and STARsolo are most computationally efficient. Second, for scATAC-seq, scReadSim provides ground-truth peaks and non-peaks by learning from user-specified trustworthy peaks and non-peaks or by calling trustworthy peaks and non-peaks from real data. The ground-truth peaks provided by scReadSim allow benchmarking peak-calling tools such as MACS3 [17], HOMER [18], HMMRATAC [19], and SEACR [20]. Our benchmarking results show that HMMRATAC and MACS3 are the top performers.

2.2 Methods

Supplementary Note (Section S2.8.1) lists the software packages and their versions used in scReadSim and the data analyses in this study.

2.2.1 The scReadSim method

The input of scReadSim is a BAM file storing the aligned sequencing reads from scRNA-seq or scATAC-seq. As an overview, scReadSim consists of three major steps: genome segmentation, synthetic count generation, and synthetic read generation.

The first step segments the genome so that each genomic region (segment) is defined as a feature. Specifically, for scRNA-seq, scReadSim requires users to input a gene annotation file to segregate the genome into genes and inter-genes; for scATAC-seq, scReadSim needs trustworthy peaks and non-peaks, which may be user-specified or not. If not user-specified, scReadSim provides two options; see Section 2.2.1.2 for detail. Based on the trustworthy peaks and non-peaks, scReadSim defines the complementary genomic regions as gray areas. Next, scReadSim summarizes scRNA-seq reads in the input BAM file into a gene-by-cell UMI count matrix and an inter-gene-by-cell UMI count matrix, or scReadSim summarizes scATAC-seq reads in the input BAM file into a peak-by-cell count matrix and a non-peak-

by-cell count matrix. We use a general term “feature-by-cell count matrix” to refer to each count matrix in the following text.

In the second step—synthetic count generation, `scReadSim` trains the count simulator `scDesign2` [10] (if the cells belong to distinct clusters; otherwise, `scDesign3` [45] can be used if the cells follow continuous trajectories) on the feature-by-cell count matrices to generate the corresponding synthetic count matrices. For scATAC-seq, `scReadSim` converts gray areas into non-peaks before generating the corresponding synthetic count matrix. In summary, `scReadSim` generates two synthetic count matrices—gene-by-cell and inter-gene-by-cell matrices—for scRNA-seq; `scReadSim` generates three synthetic count matrices—peak-by-cell, non-peak-by-cell, and non-peak-by-cell (converted from gray areas) matrices—for scATAC-seq.

The third step generates synthetic reads based on the synthetic count matrices, the input BAM file, and the reference genome (alternatively, for scRNA-seq, a collapsed transcriptome can be used to generate exon-only reads). The synthetic reads are outputted in a FASTQ or BAM file. Fig. 2.1a–b illustrate the `scReadSim` workflows for scRNA-seq and scATAC-seq, respectively.

2.2.1.1 `scReadSim` for scRNA-seq

The procedure below is for simulating data from UMI-based scRNA-seq technologies such as 10x Chromium [46] and Drop-seq [47]. For non-UMI-based scRNA-seq technologies such as Smart-seq2 [34], the simulation procedure follows Section 2.2.1.3 with user-designed open chromatin regions replaced by genes.

1. Pre-processing real scRNA-seq data to obtain UMI count matrices

To pre-process real scRNA-seq data for training, `scReadSim` requires a BAM file (containing scRNA-seq reads in cells) and a gene annotation file (in GTF format). Based on the gene coordinates in the annotation file, `scReadSim` segregates the reference genome into two sets of features: genes and inter-genes. Specifically, to account for the overlapping genes in the annotation file, `scReadSim` first aggregates overlapping

genes into non-overlapping regions (called “supergenes”) to avoid double counting of reads from overlapping genes. Then scReadSim refers to non-overlapping genes and supergenes as “genes” and defines “inter-genes” as the complementary regions. (We acknowledge that this simple strategy, though avoiding double counting of reads, may obscure the read coverage differences of overlapping genes. As a future direction, by extending scReadSim to an isoform-level read simulator, we can solve this problem by providing the ground-truth isoforms and their abundance.) Given the defined genes and inter-genes, scReadSim counts the number of UMIs whose reads overlap with each gene in every cell to construct a gene-by-cell UMI count matrix, and similarly, an inter-gene-by-cell UMI count matrix. The two matrices contain the same cells in the same column order.

2. Synthetic UMI count matrix generation

To generate the gene- and inter-gene-by-cell UMI count matrices for synthetic cells, scReadSim trains the count simulator scDesign2 [10] on the pre-processed gene- and inter-gene-by-cell UMI count matrices from real data, respectively. Note that scReadSim chooses scDesign2 because scDesign2 can generate realistic synthetic counts by capturing gene-gene correlations [41]; however, scReadSim is flexible to accommodate any scRNA-seq count simulator.

Specifically, scDesign2 involves two steps: (1) model fitting on the real data count matrix and (2) generation of a synthetic count matrix by sampling synthetic cells from the fitted model. Hence, scReadSim uses scDesign2 to generate synthetic gene-by-cell and inter-gene-by-cell count matrices from the real gene-by-cell and inter-gene-by-cell count matrices, respectively. Since scDesign2 requires cells to be in discrete cell types, scReadSim addresses this requirement by pre-clustering the cells using the gene-by-cell count matrix and Louvain clustering (in the Seurat R package [48]) if cells do not have pre-defined cell type labels. Note that scReadSim can also use the updated count simulator scDesign3 [45] to generate the synthetic count matrices if cells are from continuous trajectories instead of discrete cell types.

In summary, scReadSim generates the synthetic gene-by-cell and inter-gene-by-cell UMI count matrices in this step.

3. Synthetic read generation

To generate the scRNA-seq reads for synthetic cells, scReadSim uses:

- the synthetic gene-by-cell and inter-gene-by-cell UMI count matrices,
- the coordinates of the genes and inter-genes,
- the input BAM file,
- the reference genome,
- the user-specified read length (optional; default 90 nt).

In UMI-based scRNA-seq data, every paired-end read has only one end (referred to as the read 2) containing the RNA sequence information, while the other end (referred to as the read 1) contains the cell barcode and UMI information [42, 49]. The generation of these two ends is described below (Fig. 2.1a).

(a) Generation of the read 2 (RNA sequence)

For simplicity, we use a “read” to refer to the read 2. For each feature (gene or inter-gene) i in one synthetic cell (as an example), scReadSim (1) extracts the corresponding UMI count u_i (i.e., the number of synthetic UMIs) from the synthetic feature-by-cell UMI count matrices; (2) summarizes the read count of each UMI from the real reads that overlap with this feature i and obtains the empirical distribution $\hat{F}_i$ of the read count per UMI; (3) samples the number of synthetic reads c_{ij} for each synthetic UMI $j = 1, \dots, u_i$ according to the empirical distribution $\hat{F}_i$; and (4) for the j -th synthetic UMI,

- i. samples (with replacement) a real UMI from the real UMIs with read counts equal to c_{ij} ;
- ii. samples (with replacement) c_{ij} real reads from the reads belonging to the sampled real UMI;

- iii. generates c_{ij} synthetic reads from the c_{ij} real reads, given the read length and the reference genome;
- iv. assigns the c_{ij} synthetic reads to the j -th synthetic UMI.

Specifically, in (4), `scReadSim` converts the c_{ij} real reads' 5' positions in the reference genome into the c_{ij} synthetic reads' 5' positions (after adding random shifts, detailed in the next paragraph); then `scReadSim` finds the c_{ij} synthetic reads' 3' positions in the reference genome based on the read length (e.g., if a synthetic read has the 5' position x and the read length l , then its 3' position is $x + l - 1$). Next, given every synthetic read's 5' and 3' positions, and the strand information (preserved from the real reads), `scReadSim` extracts the synthetic read sequence from the reference genome sequence, obtaining a total of c_{ij} read sequences.

Regarding the random shifts mentioned in the above paragraph, `scReadSim` adds a random shift (sampled uniformly from -4 to 4 nt) to each real read's 5' position to obtain a synthetic read's 5' position. The purpose of doing so is to resolve duplicate 5' positions due to the sampling with replacement. To make the synthetic reads more realistic, `scReadSim` also allows the introduction of substitution errors, as described in Section 2.2.1.8.

Alternatively, in (4), `scReadSim` enables users to generate scRNA-seq synthetic reads from the transcriptome. To do so, `scReadSim` requires a collapsed transcriptome FASTA file. This collapsed transcriptome can be generated from a reference genome and a gene annotation GTF/GFF file by (i) combining all transcripts of a gene into a collapsed transcript with the software `cgat` [50]; (ii) generating the transcriptome FASTA file based on the reference genome and the collapsed transcript annotations using the software `gffread` [51]. For each gene i 's synthetic UMI $j = 1, \dots, u_i$, `scReadSim` uniformly samples the c_{ij} synthetic reads' 5' positions from $[0, l_i]$, where l_i is gene i 's collapsed transcript length; then `scReadSim` finds the c_{ij} synthetic reads' 3' positions in the collapsed transcript based on the read length. Next, given every synthetic read's 5' and 3' positions in the collapsed

transcript, scReadSim extracts the read sequence from the collapsed transcript sequence, resulting in a total of c_{ij} read sequences.

(b) Generation of the read 1 (cell barcode and UMI)

The read 1 is a concatenated sequence string of a UMI and a cell barcode. Accordingly, scReadSim concatenates a randomly generated cell barcode and a UMI to form a synthetic read 1, which will then be paired with a synthetic read 2 generated above. Below we describe the generation of a cell barcode and a UMI for each synthetic read 1, whose length is specified as 26 nt (including a 16-nt cell barcode and a 10-nt UMI).

Specifically, for each synthetic cell, scReadSim generates a 16-nt cell barcode by randomly sampling A, C, G, and T with replacement for 16 times and assigns this cell barcode to all synthetic reads belonging to the synthetic cell. To generate a UMI for each feature (gene or inter-gene) i in each synthetic cell, scReadSim (1) extracts the corresponding UMI count u_i (i.e., the number of UMIs) from the synthetic feature-by-cell UMI count matrices; (2) generates u_i 10-nt UMIs, with each UMI created by randomly sampling A, C, G, and T with replacement for 10 times; and (3) assigns the u_i UMIs to the synthetic read 2's based on the read-UMI relationship generated in (a). Note that the lengths of cell barcodes and UMIs can be user-specified (default lengths are 16 and 10 nt, respectively).

In summary, scReadSim outputs paired-end synthetic reads in two FASTQ files for read 1 and read 2 separately. Regarding the Phred quality scores in the FASTQ files, the read 2 FASTQ file uses the quality score 37 for reference sequences and the quality score 24 for erroneous substitutions, while the read 1 FASTQ file uses the quality score 37 for all sequences. Optionally, scReadSim can output a BAM file, which contains the mapped reads from the two FASTQ files to the reference genome.

2.2.1.2 scReadSim for scATAC-seq

1. Pre-processing

To pre-process real scATAC-seq data for training, scReadSim requires a BAM file (containing scATAC-seq reads in cells) and user-specified trustworthy peaks and non-peaks about the input BAM file. Alternatively, if users do not specify trustworthy peaks and non-peaks, scReadSim provides users with two options for generating trustworthy peaks and non-peaks. Option One is that scReadSim first deploys the peak-calling tool MACS3 [17] to identify the trustworthy peaks using a stringent rule (by setting the q-value threshold to 0.01) and the trustworthy non-peaks using another stringent rule (the non-peaks are complementary to the peaks called under the q-value threshold of 0.1). Option Two allows users to input a set of possible peaks from public databases, for instance, the ENCODE cCREs (candidate cis-regulatory elements) [52] summarized from various DNA accessibility assays and ChIP-seq experiments. This input set of possible peaks does not depend on the input BAM file, similar to the input gene annotation scReadSim uses to generate scRNA-seq reads. To define the trustworthy peaks and non-peaks under this Option Two, scReadSim selects the input possible peaks and non-peaks (defined as the regions complementary to the input possible peaks) that overlap with the identified peaks and non-peaks by MACS3 as in Option One. The definition of overlapping is if an input possible peak (or non-peak) has half of its length covered by one or more MACS3-identified peaks (or non-peaks); then such an input possible peak (or non-peak) would be considered trustworthy.

Based on the trustworthy peaks and non-peaks, scReadSim defines the “gray areas” as the genomic regions complementary to the peaks and non-peaks, and such gray areas represent the chromatin regions that cannot be confidently classified as peaks or non-peaks. In summary, scReadSim segments the reference genome into three sets of features: peaks, non-peaks, and gray areas (Fig. 2.1b). Then scReadSim counts the number of reads overlapping each peak in every cell to construct a peak-by-cell count matrix. scReadSim also generates a non-peak-by-cell count matrix similarly. The two matrices contain the same cells in the same column order.

2. Synthetic count matrix generation

In scATAC-seq read generation, scReadSim provides ground-truth peaks and non-peaks for benchmarking purposes by learning from user-specified trustworthy peaks and non-peaks or calling trustworthy peaks and non-peaks from real data. To generate the peak-by-cell and non-peak-by-cell count matrices for synthetic cells, scReadSim trains the count simulator scDesign2 on the pre-processed peak-by-cell and non-peak-by-cell count matrices from real data. Next, scReadSim converts the gray areas into non-peaks (so that the peaks can be regarded as “ground-truth peaks”) and constructs a synthetic count matrix based on the gray areas’ lengths, trustworthy non-peaks’ lengths, and the already-generated synthetic non-peak-by-cell count matrix.

Specifically, scReadSim generates gray areas’ synthetic read counts based on the trustworthy non-peaks’ synthetic read counts so that the gray areas’ synthetic read coverage mimics the non-peaks’ synthetic read coverage. For each gray area,

- (a) scReadSim first finds the closest non-peak on the 5’ of the gray area and extracts the non-peak’s synthetic count vector (the non-peak’s corresponding row) from the synthetic non-peak-by-cell count matrix;
- (b) scReadSim then randomly masks (sets to 0) the count entries of the non-peak’s synthetic count vector with a probability equal to $\max\left(1 - \frac{\text{length of gray area}}{\text{length of non-peak}}, 0\right)$ and assigns the masked count vector as the gray area’s synthetic count vector.

This random masking essentially scales the non-peak’s synthetic read coverage according to the ratio of the gray area’s length and the non-peak’s length (when the gray area is shorter than the non-peak) so that the gray area’s synthetic read coverage mimics that of its nearby non-peak. When the gray area is longer than the non-peak, the gray area’s read counts are set to the non-peak’s read counts, so the gray area has a lower read coverage and becomes more like a non-peak.

3. Synthetic read generation

To generate the scATAC-seq reads for synthetic cells, scReadSim uses

- the synthetic peak-by-cell, non-peak-by-cell count matrices,

- the coordinates of the peaks and non-peaks,
- the input BAM file,
- the reference genome,
- the user-specified read length (optional; default 50 nt).

As scATAC-seq usually generates paired-end reads, scReadSim accordingly generates synthetic paired-end reads. For clarity, we use a “read” to denote one end in a paired-end read. For each peak or non-peak in each synthetic cell, scReadSim (1) extracts the corresponding count c (i.e., the number of synthetic reads) from the synthetic feature-by-cell (a feature refers to a peak or non-peak) count matrices; (2) samples (with replacement) $\lceil c/2 \rceil$ (rounding $c/2$ to the least integer greater than or equal to $c/2$) real reads that overlap this feature in the BAM file; (3) finds the mates of (i.e., real reads in pairs with) the $\lceil c/2 \rceil$ sampled real reads from the BAM file; and (4) generates $\lceil c/2 \rceil$ synthetic read pairs based on the $\lceil c/2 \rceil$ real read pairs, the read length, and the reference genome.

Specifically, in (4), scReadSim follows a similar procedure to the generation of synthetic read 2 for scRNA-seq. The only difference is the accommodation of paired-end reads. In the random shifting step to avoid duplicate reads, scReadSim adds the same random shift to a pair of reads’ 5’ positions to keep the fragment size unchanged after the shift. Typically, a scATAC-seq BAM file stores cell barcodes as additional information. To generate a cell barcode (e.g., 16-nt long) for each synthetic cell, scReadSim randomly samples A, C, G, and T with replacement 16 times, and assigns this cell barcode to all synthetic reads belonging to the synthetic cell.

2.2.1.3 scReadSim for scATAC-seq with user-designed open chromatin regions

1. Pre-processing

Optionally, scReadSim allows users to design ground-truth open chromatin regions and then generates synthetic scATAC-seq reads accordingly. When users take this option,

scReadSim requires users to input the BAM file, the trustworthy peaks and non-peaks of the BAM file, and the user-designed ground-truth peaks (Supplementary Fig. 2.6). Given the user-designed ground-truth peaks, scReadSim specifies the complementary regions to the ground-truth peaks as the ground-truth non-peaks. In summary, scReadSim defines two sets of peaks and non-peaks: (1) the set of “trustworthy peaks and non-peaks” based on the user-specified (or scReadSim-defined) trustworthy peaks and non-peaks and (2) the set of “ground-truth peaks and non-peaks” based on the user-designed ground-truth peaks.

For the “trustworthy peaks and non-peaks” feature set, scReadSim counts the number of reads overlapping each trustworthy peak or non-peak in every cell to construct a trustworthy-peak-by-cell count matrix and a trustworthy-non-peak-by-cell count matrix. The two matrices contain the same cells in the same column order.

For the “ground-truth peaks and non-peaks” feature set, since the specified ground-truth peaks may not correspond to any real peaks in the BAM file, scReadSim constructs the ground-truth peaks’ counts by mapping the trustworthy peaks to the “most similar” ground-truth peaks in terms of region length. Below are the detailed construction steps.

- (a) For every ground-truth peak, scReadSim defines a corresponding trustworthy peak as follows. First, scReadSim finds the ground-truth peak’s 50 most similar trustworthy peaks (in the same chromosome) in terms of lengths (Supplementary Fig. 2.7a). Second, scReadSim chooses the trustworthy peak with the largest ratio of read count to peak length.
- (b) To construct the ground-truth-peak-by-cell count matrix, for every ground-truth peak, scReadSim adds the corresponding trustworthy peak’s row in the trustworthy-peak-by-cell count matrix to the ground-truth-peak-by-cell count matrix (Supplementary Fig. 2.7b). This constructed ground-truth-peak-by-cell count matrix has the same column order as the trustworthy-peak-by-cell count matrix.

The ground-truth-non-peak-by-cell matrix is constructed similarly, where every ground-

truth non-peak is mapped to a trustworthy non-peak. Note that for each ground-truth non-peak, scReadSim first finds the 50 most similar trustworthy non-peaks in terms of lengths, and then it chooses the trustworthy non-peak with the smallest ratio of read count to peak length.

The ground-truth-peak-by-cell and ground-truth-non-peak-by-cell count matrices, as well as the maps from the ground-truth features to the trustworthy features, will be used in the next steps.

2. Synthetic count matrix generation

Similar to the synthetic count matrix generation for scATAC-seq without user-designed ground-truth peaks (step “2. Synthetic count matrix generation” in Section 2.2.1.2), scReadSim trains scDesign2 on the ground-truth-peak-by-cell and ground-truth-non-peak-by-cell count matrices (constructed from step “1. Pre-processing”) to generate the ground-truth-peak-by-cell and ground-truth-non-peak-by-cell count matrices for synthetic cells.

3. Synthetic read generation

To generate synthetic scATAC-seq reads from the synthetic ground-truth-peak-by-cell and ground-truth-non-peak-by-cell count matrices, for each ground-truth feature (peak or non-peak) in every synthetic cell, scReadSim (1) extracts the corresponding count c (i.e., the number of synthetic reads) from the synthetic ground-truth-feature-by-cell count matrices; (2) samples (with replacement) $\lceil c/2 \rceil$ real reads in the BAM file that overlap the ground-truth feature’s corresponding trustworthy feature; (3) obtain the mates of the $\lceil c/2 \rceil$ sampled real reads; and (4) generates $\lceil c/2 \rceil$ synthetic read pairs based on the $\lceil c/2 \rceil$ real read pairs, the read length, the signed distance between the ground-truth feature and its corresponding real feature, and the reference genome.

Specifically, in (4), scReadSim converts the c real reads’ 5’ positions in the reference genome to the c synthetic reads’ 5’ positions after two adjustments: first, shifting real reads from the trustworthy feature to the ground-truth feature based on the signed distance between the two features; second, adding a random shift to each real read

pair’s 5’ positions. In greater detail, in the first adjustment step, assuming that the signed distance between the trustworthy feature and the ground-truth feature is $+d$, if a real read has a 5’ position x , then the synthetic read’s 5’ position is $x + d$. This adjustment ensures that the synthetic reads are located in the ground-truth feature as how the real reads are located in the trustworthy feature. The second adjustment (random shift addition) is to avoid duplicate reads due to sampling with replacement. Specifically, scReadSim adds the same random shift to a pair of reads’ 5’ positions to keep the fragment size unchanged after the shift.

2.2.1.4 scReadSim for single-cell multi-omics

The multi-modality of scReadSim enables it to simulate single-cell multi-omics (simultaneously measured scRNA-seq and scATAC-seq modalities) sequencing reads. In order to do so, scReadSim requires users to input a single-cell multi-omics dataset and the corresponding reference genome. Following the procedures for simulating individual scRNA-seq and scATAC-seq modalities described above, scReadSim first processes the two modalities to construct real count matrices, including scRNA-seq gene-by-cell and inter-gene-by-cell UMI count matrices and scATAC-seq peak-by-cell and non-peak-by-cell read count matrices. Then, given these count matrices, scReadSim implements scDesign3 [45], a single-cell multi-omics count simulator, to simulate synthetic cells’ UMI and read counts for both modalities. Finally, based on the synthetic counts, scReadSim generates synthetic reads for both modalities in the FASTQ format, along with a list of synthetic cell barcodes. As a result, the synthetic single-cell multi-omics sequencing reads would contain synthetic cells with both modalities and ground truths, including UMI abundance for the scRNA-seq modality and open chromatin regions for the scATAC-seq modality.

2.2.1.5 scReadSim for multiple samples/conditions

scReadSim provides a module for users to simulate single-cell sequencing reads from multiple samples or conditions. It requires users to input multiple BAM files corresponding to

multiple samples or conditions, as well as the corresponding reference genome file. Then, scReadSim trains a model for each input BAM file and simulates synthetic reads separately for each sample or condition. Specifically for scATAC-seq, to provide comparable ground-truth peaks across the multiple synthetic samples, scReadSim requires users to input a set of possible peaks, such as the ENCODE cCREs (candidate cis-regulatory elements) [52]. This pre-specified peak set provides scReadSim with a consistent segmentation of the reference genome, and scReadSim’s ground-truth peaks will be defined based on this segmentation.

2.2.1.6 Doublet/multiplet detection

In single-cell sequencing technologies that use droplets or wells (called reaction volumes) to isolate single cells, doublets/multiplets exist when more than one cell is captured by the reaction volume and mistaken as a single cell. Doublets/multiplets would bias downstream data analysis [53–55]. As scReadSim mimics real data, it requires doublets/multiplets to be removed from real data so its synthetic data would not contain doublets/multiplets. To remove doublets/multiplets from real data, scReadSim uses scDblFinder [56] as the doublet detection method based on the benchmark study [55]. For scRNA-seq data, scDblFinder is applied to the real gene-by-cell UMI count matrix to identify the columns (“cells”) that likely correspond to doublets/multiplets; then, the identified columns would be removed from both the real gene-by-cell and inter-gene-by-cell UMI count matrices. For scATAC-seq data, scDblFinder is applied to the real peak-by-cell read count matrix to identify the columns (“cells”) that likely correspond to doublets/multiplets; then, the identified columns would be removed from the real peak-by-cell and non-peak-by-cell read count matrices. Finally, after removing the doublets/multiplets from real data, scReadSim is trained on the post-removal real count matrices to generate synthetic data.

2.2.1.7 Output in a FASTQ or BAM format

To output the synthetic reads (excluding the UMI-based scRNA-seq reads 1, which contain cell barcodes and UMIs) in a FASTQ or BAM file, scReadSim first records the coordinates

of the synthetic reads in a BED file. Then scReadSim uses the bedtools [57] `getfasta` function to extract every synthetic read’s sequence from the reference genome and store the synthetic reads’ sequences in a FASTA file, which is then converted to a FASTQ file by the seqtk software [58]. For paired-end reads, scReadSim generates two FASTQ files: one for the read 1 and the other for the read 2. Users may choose FASTQ as the output format of scReadSim, or they may additionally output a BAM file by aligning the synthetic reads to the reference genome by the alignment software bowtie2 [59]. In addition, the list of synthetic cell barcodes is outputted for users’ reference, which could serve as a cell barcode white list for downstream analyses.

2.2.1.8 Substitution errors introduced into synthetic reads

To mimic the sequencing error rates observed in the actual Illumina sequencing reads, scReadSim introduces substitution errors into the synthetic reads. Specifically, scReadSim first uses the software fgbio [60] to calculate every position i ’s average error rate (i.e., the probability that the position is sequenced as a wrong nucleotide, denoted by p_i) and three substitution error rates (i.e., the probabilities that the position is wrongly sequenced as the three other nucleotides; the sum of the three probabilities is p_i) based on all real reads. Then for each synthetic read, scReadSim decides if every base call, say at position i , is erroneous by sampling from Bernoulli (p_i). If the base call is decided to be erroneous (i.e., the Bernoulli sample is 1), a substitution nucleotide is randomly sampled from the other three nucleotides with probabilities proportional to the three substitution error rates at position i . Specifically, for paired-end sequencing technologies like scATAC-seq, scReadSim introduces substitution errors separately for the two mates in a pair.

2.2.2 Data analysis strategies

2.2.2.1 Count-level comparison between synthetic data and real data

To verify that scReadSim generates synthetic data that mimics real data at the read-count level, we compared synthetic and real gene-by-cell UMI count matrices in the following

aspects: summary statistics, correlations among top-expressed genes (or peaks), and two-dimensional (2D) visualizations. For scATAC-seq, in addition to these three aspects used in scRNA-seq, we compared the RPKM distributions between synthetic and real peak-by-cell read count matrices.

- **Summary statistics.** We utilized the following summary statistics: for every feature, we calculated the mean, variance, coefficient of variance, and zero proportion across cells; for every cell, we calculated the library size and zero proportion across features.
- **Feature-feature correlations.** To illustrate that scReadSim preserves the correlations among the features (genes for scRNA-seq; peaks for scATAC-seq), we selected the 100 top-expressed features from the real count matrix and used a heatmap to show the correlations among these features in the real and synthetic count matrices, respectively.
- **2D visualizations.** We utilized principal component analysis (PCA) and uniform manifold approximation and projection (UMAP) [61, 62] to obtain the 2D embeddings of cells in the combined real and synthetic count matrix: we first used PCA to obtain the 30 principal components (PCs) and then deployed UMAP to obtain the 2D embeddings. We labeled the cells with the dataset sources (synthetic cells generated by scReadSim or real cells). We also reported the median of the integration local inverse Simpson’s index (miLISI) proposed by [63] to quantify the mixing level of the synthetic cells and the real cells (see Section 2.2.2.8). The miLISI value (ranging from 1 to 2) is close to 2 if the synthetic cells and real cells are perfectly mixed.
- **RPKM.** We used the quantity RPKM (see Section 2.2.2.8) to summarize the read coverage in a genomic region. Specifically, we reported the RPKM values separately for the peaks and non-peaks.

2.2.2.2 Sequence-level comparison between synthetic scRNA-seq data and real scRNA-seq data

We deployed scReadSim onto the mouse 10x single-cell Multiome dataset [64] (the RNA-seq modality only) by inputting the reference genome, the BAM file, and the gene annotation GTF file (Fig. 2.1a). Then, scReadSim generated synthetic scRNA-seq reads in the FASTQ and BAM formats.

To show that scReadSim’s synthetic scRNA-seq data mimics real data at the read-sequence level, we compared the synthetic BAM file and real BAM file in terms of the k-mer spectrum, the error rate per base, and the genome browser visualization.

- The k-mer spectrum. The k-mer spectrum measures the occurrences of k-mers (i.e., substrings of length k) within the read sequences. We used the tool Jellyfish [65] with the default settings to obtain the 11-mer, 21-mer, 31-mer, and 41-mer spectra in the synthetic reads and real reads, respectively.
- Error rate per base. We calculated and compared the substitution error rate for every base in the synthetic reads and real reads, respectively. The substitution error rate per base within reads was obtained using the software fgbio [60] with the function `ErrorRateByReadPosition` by setting the option `collapse` as false.
- Genome browser visualization. We used the IGV genome browser [66] to visualize the real and synthetic BAM files. This visualization illustrates the pseudo-bulk-level similarity of read coverages between the two BAM files.

2.2.2.3 Sequence-level comparison between synthetic scATAC-seq data and real scATAC-seq data

We deployed scReadSim onto the sci-ATAC-seq and mouse 10x single-cell Multiome dataset [64] (the ATAC-seq modality only) by inputting the reference genome, the BAM file, and the trustworthy peaks and non-peaks (Fig. 2.1b). The trustworthy peaks were identified by MACS3 from the input BAM file with a stringent rule (by setting the q-value parameter as

`-q 0.01`). To obtain the trustworthy non-peaks, we first deployed MACS3 onto the input BAM file with a relaxed rule (by setting the q-value parameter as `-q 0.1`) to identify peaks. Then we took the inter-peaks as the trustworthy non-peaks. Lastly, scReadSim generated synthetic scATAC-seq reads in the FASTQ and BAM formats.

To verify that scReadSim simulates realistic scATAC-seq data at the read-sequence level, we compared the synthetic BAM file and real BAM file in terms of the k-mer spectrum, the error rate per base, the genome browser visualization (see Section 2.2.1.1), and, additionally, the fragment size distribution. We further verified the realistic nature of the synthetic reads by deploying the peak caller MACS3 onto both the synthetic BAM file and real BAM file and then evaluating the consistency between the two sets of called peaks.

- **Fragment size distribution.** For paired-end sequencing reads, the fragment size distribution is the distribution of the distance between the two ends of a read pair. We used the function `bamPEFragmentSize` of the software `deeptools` [67] with the default settings to obtain the fragment size distributions of the synthetic scATAC-seq reads and real scATAC-seq reads, respectively.
- **Called peaks.** To verify that the synthetic reads mimic the real reads, we deployed the peak-calling tool MACS3 with the same setting (by setting the q-value parameter as `-q 0.01`) onto the real and synthetic BAM files and compared the two sets of called peaks using a Venn diagram. Moreover, we calculated the receiver operating characteristic (ROC) curve and the precision-recall curve by treating the real-data peaks called at `-q 0.01` as the ground truths. Then we first applied MACS3 to the synthetic reads with a relaxed criterion to obtain a long list of synthetic-data peaks (by setting the q-value parameter as `-q 0.5`) and next used various q-value thresholds for further pruning of the synthetic-data peaks. To match the synthetic peaks to the real-data peaks, we defined a match if a synthetic-data peak overlaps with a real-data peak for at least half of the minimal length of real-data peaks. Hence, given a q-value threshold and the corresponding set of synthetic-data peaks, we calculated the true positive rate and false positive rate for the ROC curve, and the precision and recall for the precision-recall

curve.

2.2.2.4 Benchmark of UMI deduplication tools

We deployed scReadSim with the default setting (generating scRNA-seq reads from the genome instead of the collapsed transcriptome; see Section 2.2.1.1) to the mouse 10x single-cell Multiome dataset [64] (the RNA-seq modality only) to generate the synthetic scRNA-seq data. Then we used scReadSim’s synthetic scRNA-seq reads as the input data and scReadSim’s gene-by-cell UMI count matrix as the ground truth to benchmark four UMI deduplication tools: UMI-tools, STARsolo, and cellranger (gene-level), as well as Alevin (transcript-level). We applied each deduplication tool to scReadSim’s synthetic scRNA-seq reads to obtain a deduplicated UMI count matrix. All four deduplication methods only count exonic reads (cellranger was implemented by setting the argument `include-introns` as false). Since these deduplication tools may output different numbers of cells, we only considered the common cells outputted by all the tools for a fair comparison. We also removed the genes with zero expression levels in all cells in any deduplicated UMI count matrices or the ground-truth UMI count matrix. We compared the deduplicated UMI count matrices with the ground-truth count matrix in the following three aspects.

1. We compared the UMI count matrices using the distributions of six summary statistics, including four gene-level statistics (mean, variance, coefficient of variance, and zero proportion) and two cell-level statistics (zero proportion and cell library size). The results are in Fig. 2.2b.
2. We calculated associations between the ground-truth UMI count matrix and each deduplicated UMI count matrix, gene-wise and cell-wise (Fig. 2.2c). The association measures include the Pearson correlation and Kendall’s tau.
3. We used PCA and UMAP for dimension reduction and visualization to examine whether the deduplicated UMI count matrices preserve the 2D cell embeddings of the ground-truth UMI count matrix. Cells are colored by the cell clusters outputted by scReadSim (the clusters are from the real data used to train scReadSim). Specifically, we

first log-transformed the three UMI count matrices. Then we performed PCA on the ground-truth matrix to obtain the top 30 PCs and the associated 30 loading vectors. Next, we applied UMAP to the top 30 PCs to find the 2D cell embeddings. For the deduplicated matrices, we first projected their cells to the same 30-dimensional PC space by left multiplying the 30-by- p loading matrix (with rows as the top 30 loading vectors; p is the number of genes); second, we projected the deduplicated matrices' cells from the PC space to the same UMAP space using the `predict()` function in the R package `umap`. We computed the Euclidean distance between each synthetic cell's two UMAP coordinates, calculated based on the cell's ground-truth UMI counts and deduplicated UMI counts, respectively. The mean value of the Euclidean distances of all synthetic cells is displayed for each UMI deduplication tool: a smaller value indicates that the deduplicated UMI count matrix better agrees with the ground truth UMI count matrix in the UMAP visualization. The results are in Fig. 2.2d.

In addition, we measured the time complexity by deploying each deduplication tool to scReadSim's synthetic data with varying cell numbers (Fig. 2.2a): 1219 ($0.25\times$), 2438 ($0.5\times$), 4877 ($1\times$, the original cell number), 9754 ($2\times$), 19508 ($4\times$); and varying sequencing depth $1.2M$ ($0.25\times$), $2.5M$ ($0.5\times$), $5.0M$ ($1\times$, the original sequencing depth), $10.1M$ ($2\times$), $20.1M$ ($4\times$). The analysis was implemented on a server with the 256x Intel® Xeon Phi™ CPU 7210 at 1.30GHz.

2.2.2.5 Benchmark of peak-calling tools

To generate synthetic scATAC-seq reads with specified ground-truth peaks (i.e., open chromatin regions), we provided scReadSim with three inputs: the mouse sci-ATAC-seq BAM file [68], the trustworthy peaks and non-peaks called from the BAM file by MACS3, and a list of specified ground-truth peaks (Supplementary Fig. 2.6). In detail, we specified each ground-truth peak as $[TSS, TSS + LEN - 1]$, where TSS is a gene's transcription start site, and the peak length LEN is uniformly sampled from 250 bp to 550 bp. If any ground-truth peaks were overlapping, we merged them into one peak so that the final ground-truth peaks

were non-overlapping. Taking the three inputs, scReadSim generated synthetic scATAC-seq reads in FASTQ and BAM files (see Section 2.2.1.3).

To benchmark four peak-calling tools—MACS3, HOMER, SEACR, and HMMRATAC, we deployed these tools with their default settings onto the synthetic scATAC-seq BAM file. Specifically, MACS3 was run under the `-BAMPE` mode with the q-value threshold 0.05; HOMER was run under the `-region` mode with the parameter `-minDist 150`; SEACR was run under the `-stringent` mode with the control threshold 0.5; HMMRATAC was run under the default setting. Then we compared the peaks called by each tool with the ground-truth peaks used to generate the synthetic data. Our comparison consisted of the following three aspects:

- First, we compared the distributions of RPKM values of the called peaks and the ground-truth peaks, and we calculated the mean difference between the two distributions. We also compared the distributions of RPKM values of the called non-peaks and the ground-truth non-peaks. The results are in Fig. 2.2e.
- Second, treating the ground-truth peaks as the truths, we plotted the ROC curve (Fig. 2.2f) and precision-recall curve (Fig. 2.2g) for each tool by applying varying thresholds to the called peaks. We considered a called peak as correct if it overlapped at least 225 bp of any ground-truth peak, where 225 bp is half the length of the shortest ground-truth peak.
- Third, we directly compared the four sets of peaks called by the four tools using the Venn diagram and the upset plot using the Python package Intervene [69]. The results are in Supplementary Figs. 2.20–2.21.

2.2.2.6 Comparison of scReadSim with minnow

We first deployed scReadSim to the mouse 10x single-cell Multiome dataset [64] (the RNA modality only) to generate synthetic reads in a BAM file. Since minnow does not accept reads but only a gene-by-cell UMI count matrix as input, we inputted into minnow the syn-

thetic gene-by-cell UMI count matrix (generated by scReadSim as an intermediate step) and kept minnow’s parameters as default. The synthetic reads minnow generated were outputted in two FASTQ files (for the two ends of paired-end reads), and we aligned them to the same reference genome scReadSim used. After binning the reference genome (chromosome 1) into 1,000-bp windows, we plotted the read coverage per window for the real data, scReadSim’s synthetic data, and minnow’s synthetic data (Fig. 2.3a). We further calculated two correlations (Spearman correlation and Pearson correlation) of read coverage between the real and synthetic data across the windows. We also provided a closer view of the read coverage (in the region chr1:86,425,858–86,443,378) using the IGV genome browser (Fig. 2.3a inset).

2.2.2.7 Comparison of scReadSim with SCAN-ATAC-Sim

We used the sci-ATAC-seq dataset [68] to train scReadSim and SCAN-ATAC-Sim. Then we compared the two simulators’ performance in preserving the peak counts in real data.

For scReadSim, we inputted the real dataset’s BAM file, the reference genome, and the trustworthy peaks (called at the q-value threshold 0.01) and non-peaks (complementary to the peaks called at the q-value threshold 0.1) identified by MACS3 from the BAM file. For each cell type, we used scReadSim to generate the same number of synthetic cells as the number of real cells. Moreover, we recorded the trustworthy peaks.

Since SCAN-ATAC-Sim requires per-cell-type input, we focused on the four largest cell types containing the most cells (predefined in [68], including Hematopoietic progenitors, Erythroblasts, Monocytes, and Immature B cells). Hence, we only examined the real reads and scReadSim’s synthetic reads for the four cell types.

For SCAN-ATAC-Sim to generate scATAC-seq data for the four cell types, it requires the four cell types’ pseudo-bulk ATAC-seq BAM files and their corresponding peak lists as input. Hence, we first extracted the four cell types’ reads from the real data BAM file to construct the four corresponding per-cell-type BAM files. Then we used MACS3 to call peaks from each per-cell-type BAM file. Finally, we inputted the four per-cell-type peak lists along with the four per-cell-type BAM files into SCAN-ATAC-Sim. For SCAN-ATAC-Sim’s

parameters, we specified the cell number of each cell type to the real cell number, which is also scReadSim’s synthetic cell number, of that cell type. For the other parameters, we used the default values suggested in the SCAN-ATAC-Sim paper [44]. SCAN-ATAC-Sim does not output synthetic read sequences but only a BED file containing synthetic reads’ starting and ending positions, along with the reads’ cell IDs.

To evaluate scReadSim and SCAN-ATAC-Sim’s performance at the peak-count level, we constructed scReadSim and SCAN-ATAC-Sim’s peak-by-cell read count matrices (one for each simulator) from scReadSim’s output BAM file and SCAN-ATAC-Sim’s output BED file, by counting each simulator’s synthetic reads that overlap with the trustworthy peaks in the synthetic cells. We also constructed a real count matrix in the same way, so the three count matrices have the same dimensions.

In three aspects, we compared the three peak-by-cell count matrices (real, scReadSim, and SCAN-ATAC-Sim). First, we calculated the distributions of six summary statistics based on the entries of each count matrix: peak-wise mean, variance, coefficient of variance, and zero proportion across cells, as well as cell-wise library size and zero proportion across peaks. We further performed the two-sample Kolmogorov–Smirnov (KS) test to compare each summary statistic’s distributions in the real count matrix and a synthetic count matrix (scReadSim or SCAN-ATAC-Sim) to measure the resemblance of the synthetic count matrix to the real count matrix (a smaller KS statistic value indicates greater similarity). The results are in Supplementary Fig. 2.8a.

Second, we obtained the 2D UMAP cell embeddings of the real and synthetic count matrices using the following steps. The results are in Fig. 2.3b.

1. We transformed the count matrices by applying the Term Frequency–Inverse Document Frequency (TF-IDF) transformation followed by the logarithmic transformation.
2. We performed PCA on the real pre-processed matrix to obtain the top 30 PCs and the associated 30 loading vectors. Next, we applied UMAP to the top 30 PCs to find the 2D cell embeddings.

3. For the synthetic pre-processed matrices, we projected their cells to the same 30-dimensional PC space by left multiplying the 30-by- p loading matrix (with rows as the top 30 loading vectors; p is the number of peaks). Then we projected the synthetic cells from the PC space to the same UMAP space using the `predict()` function in the R package `umap`. We labeled the cells with their cell types. We reported the miLSI (proposed by [63]) to quantify the degree of mixing between the synthetic cells and real cells (see Section 2.2.2.8). The miLSI value (ranging from 1 to 2) will be close to 2 if the synthetic cells and real cells are perfectly mixed.

Third, we visualized how scReadSim’s and SCAN-ATAC-Sim’s synthetic cells mix with real cells in the 2D UMAP embeddings (Supplementary Fig. 2.8b) using the following steps.

1. We combined the real cells and scReadSim’s synthetic cells by concatenating the real peak-by-cell pre-processed matrix and scReadSim’s synthetic peak-by-cell pre-processed matrix (from TF-IDF followed by the logarithmic transformation) horizontally as a peak-by-mixed-cell matrix. Similarly, we combined the real cells and SCAN-ATAC-Sim’s synthetic cells into a concatenated peak-by-mixed-cell matrix.
2. We projected the two groups of mixed cells to the previously used 30-dimensional PC space by left multiplying the 30-by- p loading matrix (with rows as the top 30 loading vectors; p is the number of peaks) obtained previously from the real peak-by-cell pre-processed matrix.
3. We projected the two groups of mixed cells from the PC space to the previously used UMAP space using the `predict()` function in the R package `umap`.

2.2.2.8 Evaluation metrics

- Reads Per Kilobase Million (RPKM). The RPKM value of a feature i is defined as

$$\text{RPKM} = \frac{n_i \times 1,000,000}{N \times l_i}, \quad (1)$$

where n_i represents the read count for feature i , $N = \sum_i n_i$ represents the total read count and l_i is the length of feature i in kilobase.

- **miLISI.** The integration local inverse Simpson’s Index (iLISI) [63] for each cell measures the effective number of datasets in a neighborhood of the cell. A cell surrounded by neighboring cells from a single dataset has an iLISI value equal to 1; otherwise, if the cell has a neighborhood with an equal number of cells from two datasets, the iLISI value is 2, indicating a perfect mixing of the two datasets in the cell’s neighborhood. Hence, the median iLISI (miLISI) of all cells measures the median mixing level of the two datasets across all cells’ neighborhoods, and a miLISI close to 2 means that the two datasets have a close-to-perfect mixing.

2.3 Results

2.3.1 scReadSim generates realistic synthetic scRNA-seq reads

To evaluate scReadSim’s ability to generate realistic scRNA-seq reads, we trained scReadSim on the RNA-seq modality of a mouse 10x single-cell Multiome dataset [64] and compared scReadSim’s synthetic data with the real data at two levels: UMI counts and read sequences. We also compared scReadSim with minnow, the only scRNA-seq read simulator with a publicly available software package.

At the UMI-count level, scReadSim uses the realistic count simulator scDesign2 [10] or scDesign3 [45] to generate UMI counts (Fig. 2.1a), and both scDesign2 and scDesign3 have been verified to generate more realistic UMI counts under more comprehensive settings compared with other count simulators. Hence, we only conducted a confirmation study by demonstrating that scReadSim’s synthetic UMI counts resemble real UMI counts in three aspects: (1) the distributions of six summary statistics, including four gene-level statistics (mean, variance, coefficient of variance, and zero proportion) and two cell-level statistics (zero proportion and cell library size) (Supplementary Fig. 2.9a); (2) correlations among the top expressed genes (Supplementary Fig. 2.9b); and (3) cells’ UMAP embeddings (Supple-

mentary Fig. 2.9c).

At the read-sequence level, we evaluated the resemblance of scReadSim’s synthetic reads to real reads in three aspects. First, scReadSim’s synthetic reads preserve the k-mer spectra, which describe the distribution of k-mers’ frequencies (i.e., numbers of occurrences) in reads. Supplementary Fig. 2.10a shows similar k-mer spectra in synthetic reads and real reads. Second, scReadSim can introduce substitution errors to imitate those observed in real reads. Supplementary Fig. 2.10b shows that scReadSim’s synthetic reads mimic real reads in terms of the substitution error rate per base. Third, after read alignment, the pseudo-bulk read coverage of scReadSim’s synthetic reads mimics that of real reads (Spearman correlation = 0.930 in Fig. 2.3a; Supplementary Fig. 2.10c). In contrast, the pseudo-bulk read coverage of minnow’s synthetic reads does not mimic real data (Spearman correlation = 0.065 in Fig. 2.3a), an expected result as minnow does not learn from real scRNA-seq reads.

2.3.2 scReadSim generates realistic synthetic scATAC-seq reads

For scATAC-seq, scReadSim provides ground-truth peaks and non-peaks for benchmarking purposes, by learning from user-specified or scReadSim-identified (under stringent criteria) peaks and non-peaks that are deemed trustworthy in real data (Fig. 2.1b). First, scReadSim sets trustworthy peaks and non-peaks as ground-truth peaks and non-peaks, respectively. Then regarding the gray areas (i.e., regions complementary to trustworthy peaks and non-peaks), whose read coverages are between those of trustworthy peaks and trustworthy non-peaks in real data, scReadSim converts the gray areas to ground-truth non-peaks.

We verified that scReadSim generates realistic synthetic scATAC-seq reads at two levels: read counts and read sequences, by mimicking two real datasets: the ATAC-seq modality of a mouse 10x single-cell Multiome dataset [64] and a sci-ATAC-seq dataset [68]. We also compared scReadSim with SCAN-ATAC-Sim, the only scATAC-seq read simulator available.

At the read-count level, we first confirmed that scReadSim’s synthetic peak-by-cell and non-peak-by-cell count matrices mimic the real count matrices in four aspects: (1) the distributions of six summary statistics, including four peak-level statistics (mean, variance,

coefficient of variance, and zero proportion) and two cell-level statistics (zero proportion and cell library size) (Supplementary Figs. 2.11a and 2.12a); (2) the distributions of normalized read counts (in Reads Per Kilobase Million (RPKM)) of peaks and non-peaks (Supplementary Figs. 2.11b and 2.12b); (3) correlations among the top open peaks (Supplementary Figs. 2.11c and 2.12c); and (4) cells' UMAP embeddings (Supplementary Figs. 2.11d and 2.12d). Second, we found that scReadSim outperforms SCAN-ATAC-Sim in mimicking the sci-ATAC-seq data in both the distributions of the six summary statistics (Supplementary Fig. 2.8a) and the cells' UMAP embeddings (Fig. 2.3b and Supplementary Fig. 2.8b), an expected result as SCAN-ATAC-Sim does not learn from real scATAC-seq data.

At the read-sequence level, we evaluated the resemblance of scReadSim's synthetic scATAC-seq reads to real reads in five aspects. First, after read alignment, scReadSim's synthetic read coverage mimics the real read coverage in the trustworthy peaks and non-peaks at the pseudo-bulk level (Fig. 2.3c and Supplementary Fig. 2.13a); moreover, scReadSim's synthetic read coverage in the ground-truth non-peaks converted from gray areas mimics the real read coverage in the nearby trustworthy non-peaks. We further verified that scReadSim preserves the cell-type-specific read coverage in real data (Fig. 2.4a). Second, scReadSim's synthetic reads mimic real reads in terms of k-mer spectra (Supplementary Figs. 2.13b and 2.14a). Third, scReadSim's synthetic reads preserve the fragment size distribution of real reads (Fig. 2.4b and Supplementary Fig. 2.14b); it is known that the fragment size distribution results from the pattern of chromatin accessibility and thus meaningful [70]. Fourth, scReadSim's synthetic reads maintain the substitution error rate per base in real reads (Fig. 2.4c and Supplementary Fig. 2.14c). Fifth, MACS3, a state-of-the-art peak calling tool, obtains consistent peak calling results from scReadSim's synthetic reads and real reads (Fig. 2.4d, Supplementary Figs. 2.13c and 2.14d-e; a reasonable result is that the consistency is higher for the ATAC-seq modality in the mouse 10x single-cell Multiome dataset, which is less sparse than the sci-ATAC-seq dataset).

Beyond mimicking real data, scReadSim also allows users to design ground-truth peaks and non-peaks for synthetic scATAC-seq data generation at the cell-type level (Fig. 2.4e) or even with an arbitrary design (Supplementary Fig. 2.15). We show that scReadSim's

synthetic scATAC-seq data with user-designed ground-truth peaks and non-peaks preserve the real read-sequence characteristics and mimic the trustworthy peaks and non-peaks in real data (Supplementary Fig. 2.16). In summary, scReadSim can provide reliable ground truths for benchmarking computational tools that process scATAC-seq reads.

2.3.3 scReadSim allows varying cell numbers and sequencing depths

scReadSim flexibly allows users to vary cell numbers and sequencing depths to generate synthetic data. For UMI-based scRNA-seq data, scReadSim defines the baseline sequencing depth as the number of UMIs in the input real data; for non-UMI-based scRNA-seq data and scATAC-seq data, scReadSim defines the baseline sequencing depth as the number of sequencing reads (with duplicated reads resulted from polymerase chain reaction (PCR) removed) in the input real data.

For demonstration purposes, we deployed scReadSim to the RNA-seq modality of a mouse 10x single-cell Multiome dataset [64] and varied the cell number or the sequencing depth (Supplementary Fig. 2.17). Supplementary Fig. 2.17a shows the synthetic datasets scReadSim generated with a fixed sequencing depth but increasing cell numbers 1,219 ($0.25\times$), 2,438 ($0.5\times$), 4,877 ($1\times$, the original cell number), 9,754 ($2\times$), and 19,508 ($4\times$). For a fair visual comparison, we downsampled every synthetic dataset to 1,219 cells ($0.25\times$) in Supplementary Fig. 2.17a and observed that the read coverage of the pooled 1,219 synthetic cells (pseudo-bulk) decreases as the synthetic cell number increases, an expected phenomenon reflecting the trade-off between the cell number and the per-cell sequencing depth given a fixed sequencing depth.

Supplementary Fig. 2.17b shows the synthetic datasets scReadSim generated with a fixed cell number but increasing sequencing depths 1.2M ($0.25\times$), 2.5M ($0.5\times$), 5.0M ($1\times$, the original sequencing depth), 10.1M ($2\times$), and 20.1M ($4\times$). As expected, the read coverage of the pooled cells (pseudo-bulk) increases as the sequencing depth increases.

Note that although scReadSim allows users to specify the sequencing depth arbitrarily, users must consider the biological limits so that the specified sequencing depth is reasonable

(e.g., the total number of mRNA transcripts is the upper bound on the number of UMIs).

2.3.4 Application 1: Benchmarking UMI deduplication tools

For UMI-based scRNA-seq data, UMI deduplication tools have been developed to quantify gene expression levels from scRNA-seq reads, some of which may come from the same RNA molecule. UMI deduplication tools input scRNA-seq reads (containing UMIs and cell barcodes) and output a gene-by-cell UMI count matrix. scReadSim can be used for benchmarking UMI deduplication tools because it provides the ground-truth UMI count matrix in its synthetic scRNA-seq read generation process. Hence, we deployed scReadSim to the mouse 10x single-cell Multiome dataset [64] (the RNA-seq modality only) and used the synthetic reads to benchmark four state-of-the-art UMI deduplication tools: cellranger [15], STARsolo [14], UMI-tools [13], and Alevin [16]. Since Alevin is a transcript-level deduplication tool but scReadSim only allows gene-level ground-truth UMI counts in its current version (V.1.4.1), we will focus on cellranger, STARsolo, and UMI-tools in the following result comparison (Fig. 2.2a–d); the Alevin results are in Supplementary Figs. 2.18 and 2.19. Our benchmark study shows that UMI-tools achieves better accuracy than STARsolo and cellranger, while STARsolo is the most computationally efficient among the three tools.

First, in terms of computational efficiency, STARsolo runs faster than UMI-tools and cellranger on all synthetic datasets (with varying cell numbers and sequencing depths) (Fig. 2.2a). As expected, all three tools take longer to run when the sequencing depth increases, and their running time is unaffected by the cell number when the sequencing depth is fixed.

Second, the UMI count matrix output by UMI-tools agrees better with the ground-truth UMI count matrix in terms of (1) the distributions of six summary statistics, including four gene-level statistics (mean, variance, coefficient of variance, and zero proportion) and two cell-level statistics (zero proportion and cell library size) (Fig. 2.2b); and (2) Pearson correlations and Kendall’s tau correlations between the two matrices, both gene-wise (the correlation across cells per gene) and cell-wise (the correlation across genes per cell)

(Fig. 2.2c).

Third, UMAP visualization shows that, UMI-tools outputs a UMI count matrix that is most similar to the ground-truth UMI count matrix (evidenced by the smallest average Euclidean distance between synthetic cells' two 2D UMAP coordinate vectors, calculated based on the ground-truth UMI counts and deduplicated UMI counts, respectively) and best preserves the separation among cell clusters (Fig. 2.2d).

2.3.5 Application 2: Benchmarking scATAC-seq peak-calling tools

For benchmarking scATAC-seq peak-calling tools, scReadSim provides ground-truth peaks and non-peaks by learning from user-specified trustworthy peaks and non-peaks or calling trustworthy peaks and non-peaks from real data.

Here we used scReadSim's synthetic scATAC-seq data (pseudo bulk), which contain designed ground-truth peaks and non-peaks and mimic the sci-ATAC-seq dataset [68], to benchmark four popular peak-calling tools: (1) MACS3, designed for identifying transcript factor binding sites from ChIP-seq data; (2) SEACR, designed for CUT&RUN data; (3) HOMER, designed for ChIP-seq data but also widely used for ATAC-seq data; and (4) HMMRATAC, designed for ATAC-seq data.

Our benchmark results indicate that MACS3 and HMMRATAC have the best overall performance among the four peak-calling tools. In particular, the peaks called by MACS3 best resemble the ground-truth peaks in terms of read coverage, while HMMRATAC achieves the best balance between precision and recall.

First, we plotted the RPKM distributions in ground-truth peaks and non-peaks, as well as in the identified peaks and non-peaks by each peak-calling tool (Fig. 2.2e). The distributions indicate that MACS3 identifies the peaks that are the most similar to the ground-truth peaks, followed by HMMRATAC, HOMER, and SEACR. Specifically, the peaks called by HMMRATAC and SEACR are much longer than those by MACS3 and HOMER: the average peak lengths are 1543.0 bp and 1746.9 bp for HMMRATAC and SEACR, in contrast to 490.4 bp and 431.6 bp for MACS3 and HOMER. Possibly due to the design for CUT&RUN data,

SEACR identifies the genomic regions with non-zero read coverages as peaks. However, the sci-ATAC-seq pseudo-bulk data used here contain many trustworthy non-peaks with non-zero read coverages, so most of the peaks called by SEACR contain so few reads that these peaks' RPKMs are close to 0.

Second, all four tools detect most of the ground-truth peaks, under the assumption that a ground-truth peak is considered as correctly identified if a minimum length of the peak (i.e., half of the length of the shortest peak) is overlapped by an identified peak (Fig. 2.2f and Supplementary Figs. 2.20–2.21). Under the default settings of the four tools, the recall rates (i.e., the percentage of ground-truth peaks identified correctly) are all high: 86.8% for MACS3, 83.9% for SEACR, 92.4% for HOMER, and 84.4% for HMMRATAC.

Third, HMMRATAC achieves the highest precision rate of 90.3% (i.e., the percentage of identified peaks that are correct), followed by 76.5% for MACS3, 56.3% for HOMER, and 7.8% for SEACR. SEACR's low precision rate is possibly due to its design, leading to a vast number of identified peaks (34,834 peaks vs. 3,678 peaks of MACS3, 5,314 peaks of HOMER, and 3,029 peaks of HMMRATAC) (Fig. 2.2g). Combining the precision and recall rates, the F1 scores indicate that HMMRATAC (87.3%) outperforms MACS3 (81.3%), HOMER (70.0%), and SEACR (14.3%).

2.4 Discussion

scReadSim, a single-cell RNA-seq and ATAC-seq read simulator, generates realistic synthetic reads by mimicking real reads while providing ground truths including UMI count matrix for scRNA-seq and user-designed open chromatin regions for scATAC-seq. For both modalities, scReadSim allows the cell number and sequencing depth to vary. Although the current version of scReadSim uses scDesign2 [10] as the internal count simulator (assuming that cells are from discrete cell types), scReadSim is easily adaptive to the new count simulator scDesign3 [45] that allows cells to come from continuous trajectories or have spatial locations, with conditions and batches, and can generate single-cell multi-omics data. Moreover, scReadSim can be used for evaluating the robustness of read-level bioinformatics tools

because scReadSim allows the generation of sequencing reads with varying levels of noise injected (e.g., sequencing errors, read positions, and read coverages).

One limitation of the current simulation framework is the potential for overfitting during model fitting. Specifically, several empirical distributions—such as the distribution of read starting positions—are estimated directly from the input dataset of raw sequencing reads. Because these distributions guide the simulation, the synthetic data may closely reflect the specific characteristics of the input dataset. As a result, tools benchmarked using this data may perform well under similar conditions but fail to generalize to datasets with different technical or biological properties. To partially address this, scReadSim supports the injection of noise—such as sequencing errors, variable read positions, and non-uniform coverage—to introduce variation and stress-test tool performance beyond the exact characteristics of the training data. Nonetheless, we acknowledge this trade-off and recommend interpreting benchmarking results with appropriate caution.

Additionally, the current version of scReadSim (V.1.4.1) generates scRNA-seq reads with gene-level ground truths (UMI counts) only, not isoform-level ground truths including isoform abundance and alternative splicing events. Therefore, we do not recommend using the current version of scReadSim to benchmark computational tools that study isoforms or alternative splicing events. In addition, the default implementation of scReadSim may generate intronic synthetic reads when extracting sequences from the reference genome. Therefore, for benchmarking scenarios where intronic and exonic reads might make a difference, such as implementing cellranger (v7.0.0) under the default mode (counting intronic reads), we recommend using the transcriptome mode of scReadSim to generate exonic reads by extracting sequences from a collapsed transcriptome, compiled from genes’ annotated transcripts [13, 71]. As a future direction, we will generalize scReadSim to generate scRNA-seq short and long reads with ground-truth RNA isoform abundance, a functionality unavailable in existing scRNA-seq simulators.

To generate multiple samples and conditions, the current version of scReadSim requires users to input one BAM file per sample or condition. As a future direction, we will leverage the count simulator scDesign3 [45], which allows for adding sample and condition effects,

to generate synthetic reads for multiple samples and conditions when the input is only one BAM file, by introducing sample and condition effects from realistic distributions learned from public data.

2.5 Code and data availability

The scReadSim Python package is available at <https://github.com/JSB-UCLA/scReadSim>. The comprehensive tutorials are available at <http://screadsim.readthedocs.io/>. In the tutorials, we described the input and output formats, model parameters, and exemplary datasets for each functionality of scReadSim. The source code for reproducing the results are available at: https://github.com/Dominic7227/scReadSim_source [72].

The 10x Genomics single-cell multiome dataset includes ATAC and gene expression data from the embryonic mouse brain tissue [64]. The raw BAM file (ATAC-seq modality) was downloaded from https://cf.10xgenomics.com/samples/cell-arc/2.0.0/e18_mouse_brain_fresh_5k/e18_mouse_brain_fresh_5k_atac_possorted_bam.bam. The raw BAM file (RNA-seq modality) was downloaded from https://s3-us-west-2.amazonaws.com/10x.files/samples/cell-arc/2.0.0/e18_mouse_brain_fresh_5k/e18_mouse_brain_fresh_5k_gex_possorted_bam.bam. The reference genome file (assembly version GRCm38.p4 release M10) was downloaded from GENCODE (https://www.gencodegenes.org/mouse/release_M10.html). The sci-ATAC-seq dataset measures the chromatin accessibility in 17 samples spanning 13 tissues in 8-week old mice [68]. The raw BAM file was downloaded from http://krishna.gs.washington.edu/content/members/mouse_ATAC_atlas_website/bams/BoneMarrow_62016.bam. Data from the tissue “Bone Marrow 62016” were used for analysis. The reference genome file (assembly version NCBIM37 release M1) was downloaded from GENCODE (https://www.gencodegenes.org/mouse/release_M1.html).

2.6 Acknowledgements

This chapter is based on my joint work with Dr. Dongyuan Song, and my Ph.D. advisor Dr. Jingyi Jessica Li. We appreciate the comments and feedback from Qingyang Wang, Dr. Xinzhou Ge, Zhiqian Zhai, and other members of the Junction of Statistics and Biology at UCLA (<http://jsb.ucla.edu>). We also thank Dr. Chongzhi Zang for his helpful suggestions.

2.7 Figures

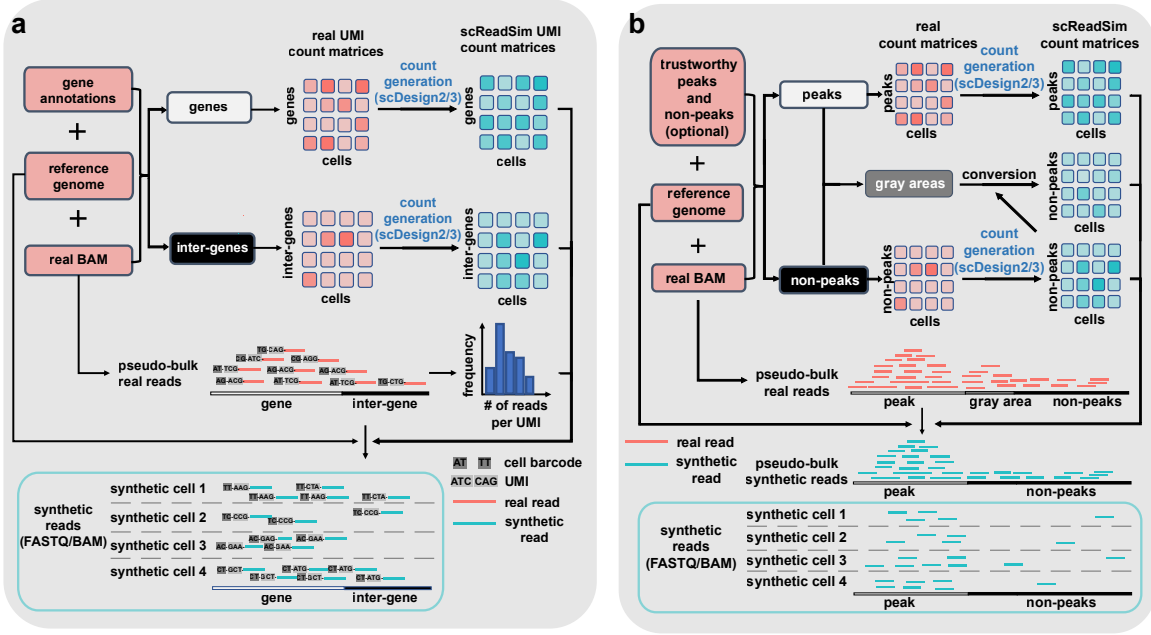


Figure 2.1: Workflows of scReadSim’s scRNA-seq (a) and scATAC-seq (b) read generation.

a, The required input includes a real scRNA-seq dataset’s BAM file, the corresponding reference genome, and a gene annotation GTF file. Based on the input, scReadSim segregates the reference genome into genes and inter-genes (i.e., intergenic regions). Based on the genes and inter-genes, scReadSim summarizes scRNA-seq reads in the input BAM file into a gene-by-cell UMI count matrix and an inter-gene-by-cell UMI count matrix. Then scReadSim trains the count simulator scDesign2 [10] (if the cells belong to distinct clusters; otherwise, scDesign3 [45] can be used if the cells follow continuous trajectories) on the two UMI count matrices to generate the corresponding synthetic UMI count matrices (“ground truths” for benchmarking UMI deduplication tools). Last, scReadSim generates synthetic scRNA-seq reads based on the synthetic UMI count matrices, the input BAM file, and the reference genome. The synthetic reads are outputted in a FASTQ or BAM file. **b**, The input includes a real scATAC-seq dataset’s BAM file, the corresponding reference genome, and optionally, users’ trustworthy peaks and non-peaks in the input BAM file; if users do not input trustworthy peaks and non-peaks, scReadSim provides two options; see Section 2.2.1.2 for detail. Based on the trustworthy peaks and non-peaks, scReadSim defines the complementary genomic regions as gray areas and summarizes scATAC-seq reads in the input BAM file into a peak-by-cell count matrix and a non-peak-by-cell count matrix. Next, scReadSim trains the count simulator scDesign2 [10] (or scDesign3 [45]) on the two count matrices to generate the corresponding synthetic count matrices for the peaks and non-peaks. Further, scReadSim converts the gray areas into non-peaks (so that the peaks can be regarded as “ground-truth peaks”) and constructs a synthetic count matrix based on the gray areas’ lengths and the already-generated synthetic non-peak-by-cell count matrix. Last, scReadSim generates synthetic reads based on the three synthetic count matrices, the input BAM file, and the reference genome. The synthetic reads are outputted in a FASTQ or BAM file.

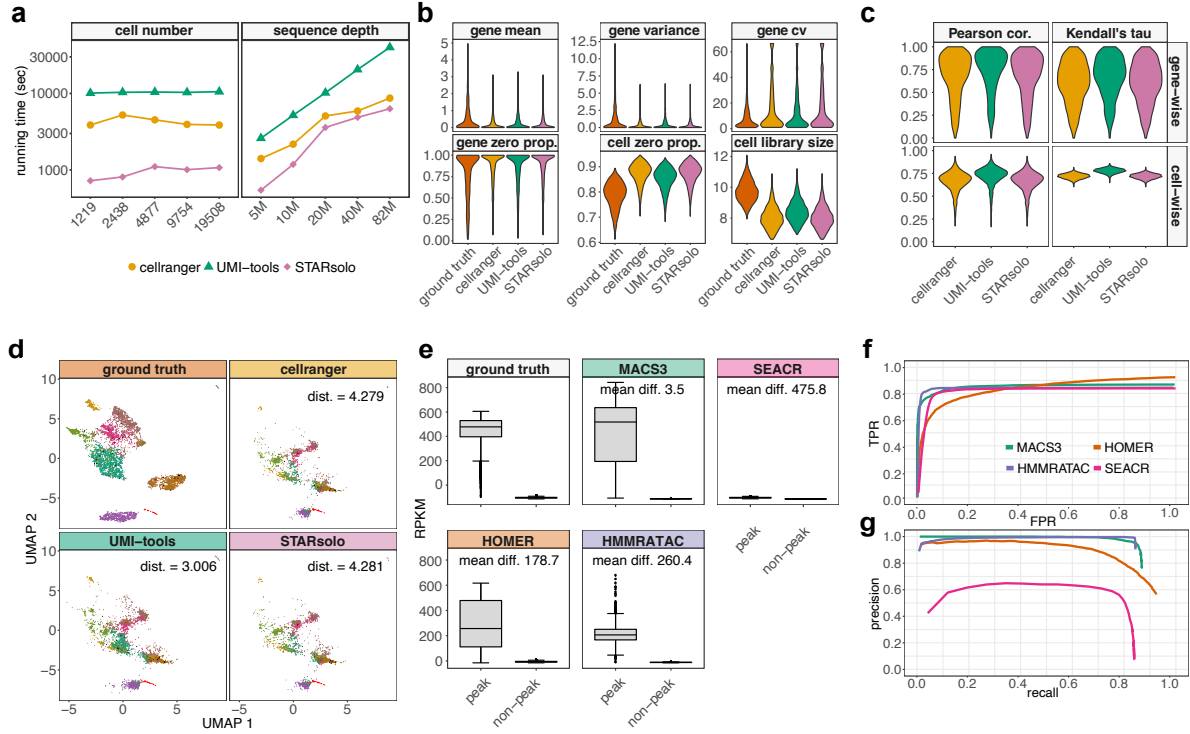


Figure 2.2: Benchmark of UMI deduplication tools (a–d) and peak calling tools (e–g) using scReadSim’s synthetic reads.

a, Time usage of deduplication tools on synthetic datasets with varying cell numbers (at a fixed sequencing depth) or varying sequencing depths (at a fixed cell number). The y-axis indicates the time lapse (in seconds), and the x-axis shows the number of synthetic cells (left) or the total number of UMIs (sequencing depth, right). **b**, Distributions of summary statistics of the UMI count matrices (ground truth, cellranger’s output, UMI-tools’ output, and STARsolo’s output) at the gene level (mean, variance, coefficient of variance (cv), and zero proportion) and the cell level (zero proportion and library size). **c**, Cell-wise and gene-wise correlations (Pearson correlation and Kendall’s tau) between the ground-truth UMI count matrix and each deduplication tool’s output UMI count matrix. **d**, UMAP visualizations of the ground-truth UMI count matrix and each deduplication tool’s output UMI count matrix. The mean value of the Euclidean distances of all synthetic cells is displayed for each UMI deduplication tool: a smaller value indicates that the deduplicated UMI count matrix better agrees with the ground-truth UMI count matrix in UMAP visualization. **e**, Distributions of RPKM values of peak and non-peak regions in the ground truth (specified in scReadSim) and each tool’s peak-calling result. The box center lines, bounds, and whiskers denote the medians, first and third quartiles, and minimum and maximum values within $1.5 \times$ the interquartile range of the box limits, respectively. Mean differences (mean diff.) of peak regions’ RPKM are calculated between ground truth and each method. The number of peaks and non-peaks are listed as follows: ground-truth peaks 2913 and non-peaks 2914, MACS3 peaks 3310 and non-peaks 3290, SEACR peaks 31350 and non-peaks 31339, HOMER peaks 4782 and non-peaks 4780, HMMRATAC peaks 2726 and non-peaks 2571. **f–g**, True positive rate (TPR) vs. false positive rate (FPR) curves (**f**) and precision vs. recall curves (**g**) using user-designed open chromatin regions as the ground-truth peaks.

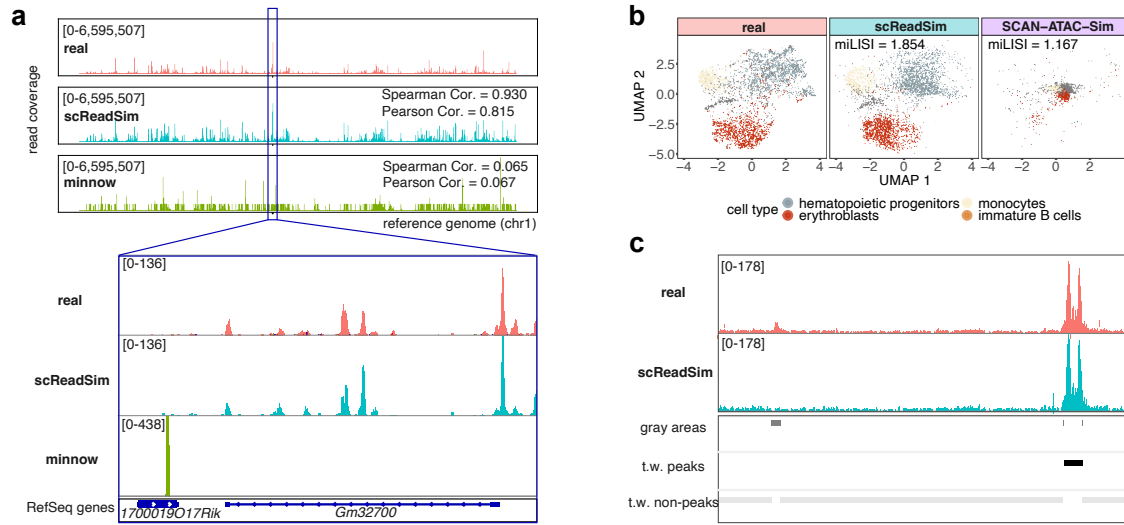


Figure 2.3: scReadSim outperforms existing scRNA-seq and scATAC-seq read simulators.

a, scReadSim outperforms the existing scRNA-seq read simulator minnow in preserving the read coverage in a mouse 10x single-cell Multiome dataset (the RNA-seq modality only) [64]. With chromosome 1 of the reference genome divided into consecutive, non-overlapping 1,000-bp windows, the x-axis represents the windows' positions along chromosome 1, and the y-axis indicates the number of the reads overlapping each window in each pseudo-bulk sample (real or synthetic, with all cells pooled). The track height is set to 6,595,507 for all three tracks. The Spearman correlation (Cor.) and Pearson Cor. measure the similarity of read coverage between real and synthetic data across the windows. Inset: a closer view of the region chr1:86,425,858–86,443,378 using the IGV genome browser [66]. Each coverage track displays the depth of the reads covered at each locus and indicates its track height 136 or 438 at the left corner. **b**, In UMAP visualizations, scReadSim outperforms the existing scATAC-seq read simulator SCAN-ATAC-Sim in mimicking real cells in the mouse 10x single-cell Multiome dataset (the ATAC-seq modality only) [64]. The miLISI measures the similarity of real and synthetic cells in the UMAP space: the miLISI value ranges between 1 and 2, with 2 indicating a perfect mixing of real and synthetic cells. **c**, scReadSim converts gray areas in real data to ground-truth non-peaks in synthetic data, maintaining trustworthy (t.w.) peaks and non-peaks in real data as ground-truth peaks and non-peaks, respectively, in synthetic data.

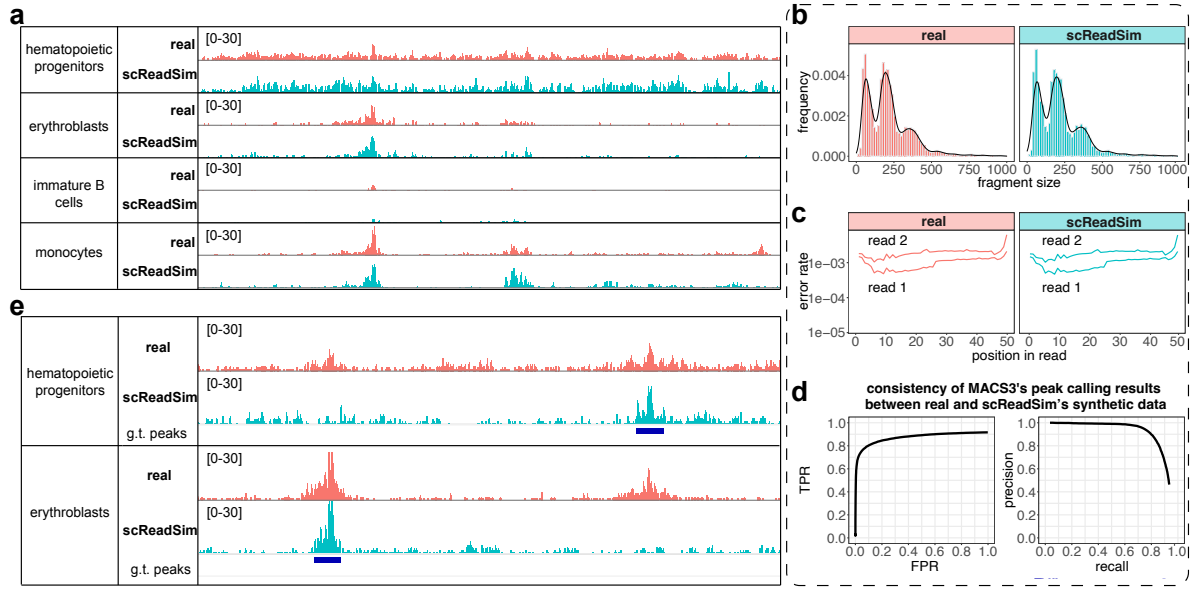


Figure 2.4: scReadSim generates realistic scATAC-seq sequencing reads with ground-truth peaks.

scReadSim's synthetic scATAC-seq data mimic the real sci-ATAC-seq dataset [68] in terms of the cell-type-specific read coverage (a), the fragment-size distribution (b), substitution error rate per base within a read (c), and the peaks called by MACS3 at the pseudo-bulk level (d). e, scReadSim enables user-designed, cell-type-specific ground-truth (g.t.) peaks for scATAC-seq read generation.

2.8 Supplementary materials

S2.8.1 Supplementary note

To run scReadSim, the following dependent software packages are needed. All of the software packages needed to run scReadSim are freely available.

- MACS3: Version 3.0.0a7 (<https://github.com/macs3-project/MACS>).
- samtools: Version 1.12 (with htlib 1.12) (<http://www.htlib.org/>).
- bedtools: Version 2.29.1 (<https://bedtools.readthedocs.io/en/latest/>).
- seqtk: Version 1.3-r117-dirty (<https://github.com/lh3/seqtk>).
- bowtie2: Version 2.3.4.1 (<http://bowtie-bio.sourceforge.net/bowtie2/index.shtml>).
- fgbio: Version 2.0.1 (<http://fulcrumgenomics.github.io/fgbio/>).
- Seurat: R package Seurat Version 4.0.6.
- scDblFinder: R package scDblFinder Version 1.14.0.
- scDesign2: R package scDesign2 Version 1.0.0.
- scDesign3: R package scDesign3 Version 0.99.5.

In addition, the following software packages were used for the analyses in this work.

Existing read simulators

- minnow: Version 0.1.0 (<https://github.com/COMBINE-lab/minnow>)
- SCAN-ATAC-Sim: <http://scan-atac-sim.gersteinlab.org/>

Dimension reduction methods

- UMAP: Function `umap()` from R package `umap` Version 0.2.7.0.
- PCA: Function `irlba()` from R package `irlba` Version 2.3.5.

UMI deduplication tools

- UMI-tools: Version 1.1.2 (<https://github.com/CGATOxford/UMI-tools>).
- STARsolo: Version 2.7.9a (<https://github.com/alexdobin/STAR>).
- Alevin: Software Alevin (integrated into the software salmon Version 1.8.0) (<https://salmon.readthedocs.io/en/latest/alevin.html>).
- cellranger: Version 7.0.0 (<https://support.10xgenomics.com/single-cell-gene-expression/software/pipelines/latest/what-is-cell-ranger>).

Peak-calling tools

- MACS3: Version 3.0.0a7 (<https://github.com/macs3-project/MACS>).
- SEACR: Version 1.3 (<https://github.com/FredHutch/SEACR>).
- HOMER: Software HOMER Version 4.11 (<http://homer.ucsd.edu/homer/index.html>).
- HMMRATAC: Version 1.2.10 (<https://github.com/LiuLabUB/HMMRATAC>).

Other tools

- Jellyfish: Version 2.3.0 (<https://github.com/gmarcais/Jellyfish>).
- Intervene: Python package Intervene Version 0.41.0 (<https://intervene.readthedocs.io/en/latest/install.html>).

S2.8.2 Supplementary figures

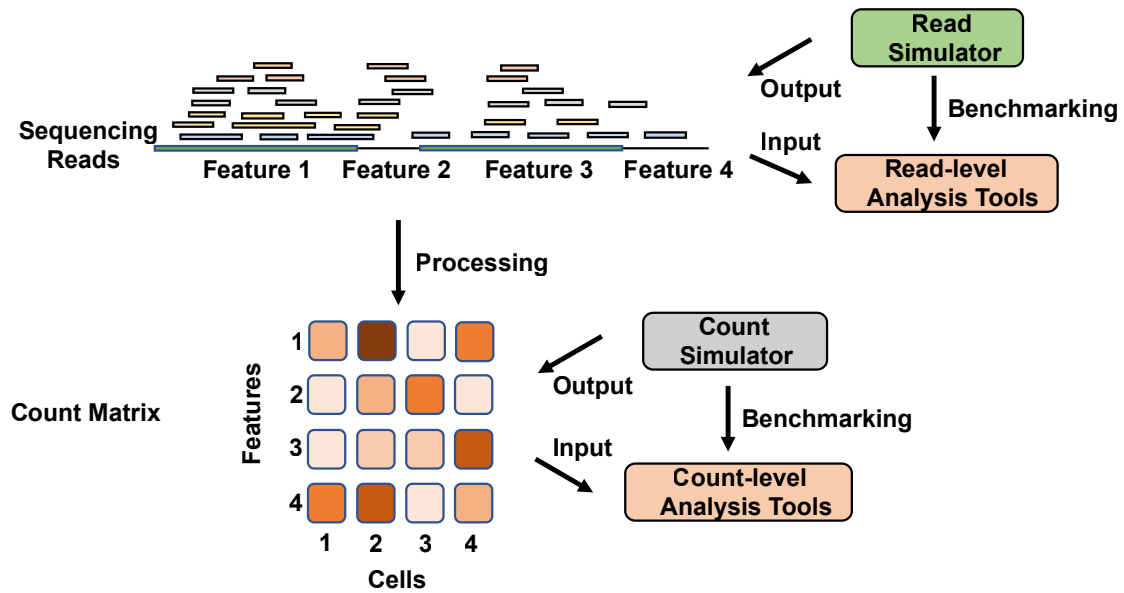


Figure 2.5: Relationships between single-cell sequencing data analysis tools and simulators.

Read-level analysis tools process sequencing reads directly, while count-level analysis tools input a processed sequencing read count matrix. To benchmark count-level analysis tools, researchers need count simulators that generate synthetic sequencing read counts. However, count simulators cannot provide ground truths for benchmarking read-level analysis tools.

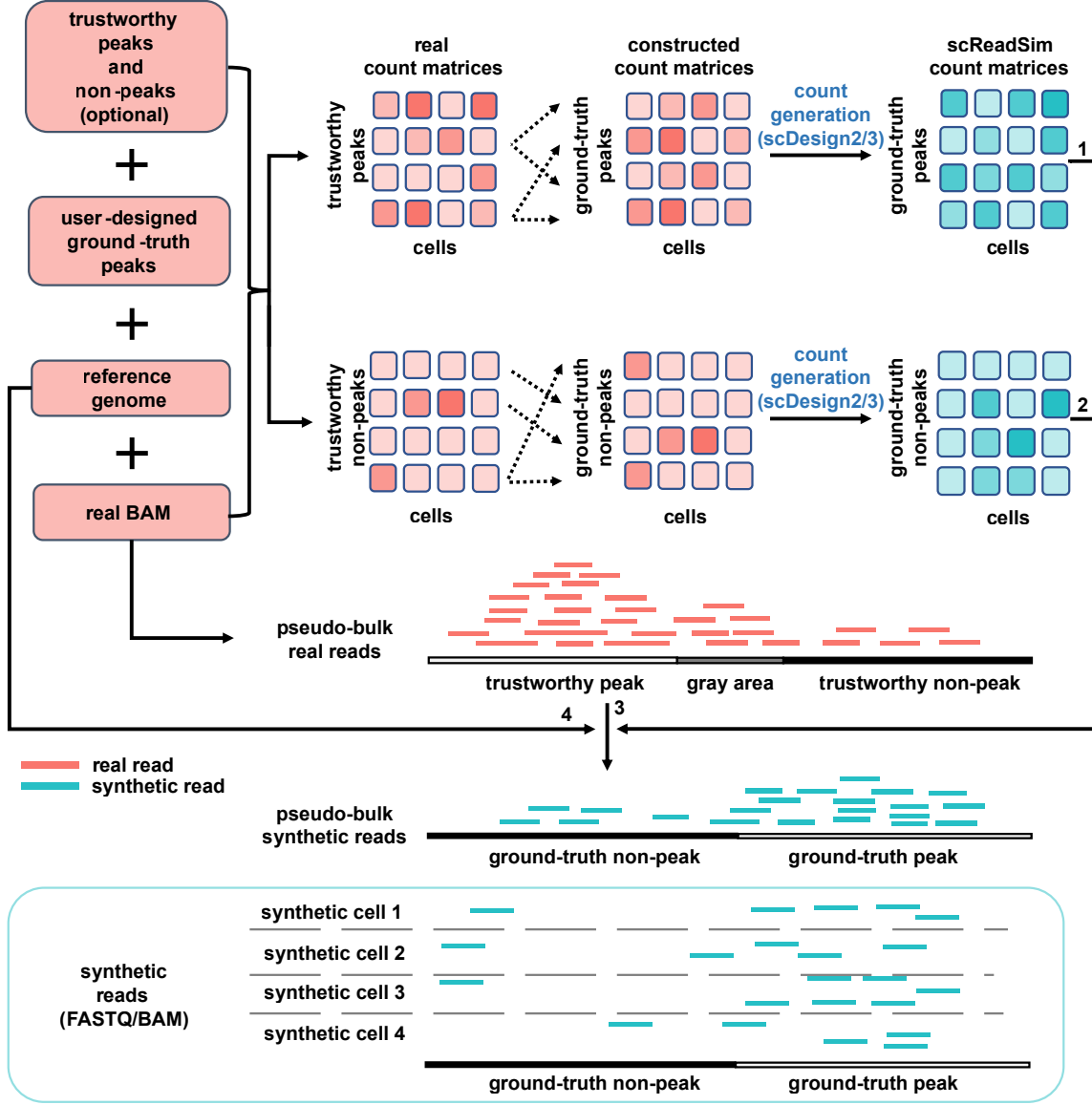


Figure 2.6: Workflow of scReadSim for generating scATAC-seq reads with user-designed open chromatin regions (i.e., ground-truth peaks).

The input includes a BAM file, the corresponding reference genome, trustworthy peak and non-peak lists, and a ground-truth peak list that the synthetic scATAC-seq reads will be generated accordingly. Specifically, if users do not input trustworthy peaks and non-peaks, scReadSim provides two options; see Section 2.2.2.8 for detail. Given ground-truth peaks, scReadSim defines the inter-peaks as the ground-truth non-peaks. Based on trustworthy peaks and non-peaks, scReadSim summarizes scATAC-seq reads in the input BAM file into a trustworthy-peak-by-cell count matrix and a trustworthy-non-peak-by-cell count matrix. Next, scReadSim defines a mapping between trustworthy and ground-truth peaks and constructs a ground-truth-peak-by-cell count matrix by extracting the corresponding trustworthy-peak-by-cell count matrix's entries. Similarly, scReadSim constructs a ground-truth-non-peak-by-cell count matrix from the trustworthy-non-peak-by-cell count matrix. Further, scReadSim trains the count simulator scDesign2 [1] (if the cells belong to distinct clusters; otherwise, scDesign3 [2] can be used if the cells follow continuous trajectories) on the ground-truth-peak-by-cell and ground-truth-non-peak-by-cell count matrices to generate the corresponding synthetic count matrices for the ground-truth peaks and non-peaks. Last, scReadSim generates synthetic reads based on the synthetic count matrices (Input 1 and 2), the input BAM file (Input 3), and the reference genome (Input 4). The synthetic reads are outputted in the FASTQ or BAM format.

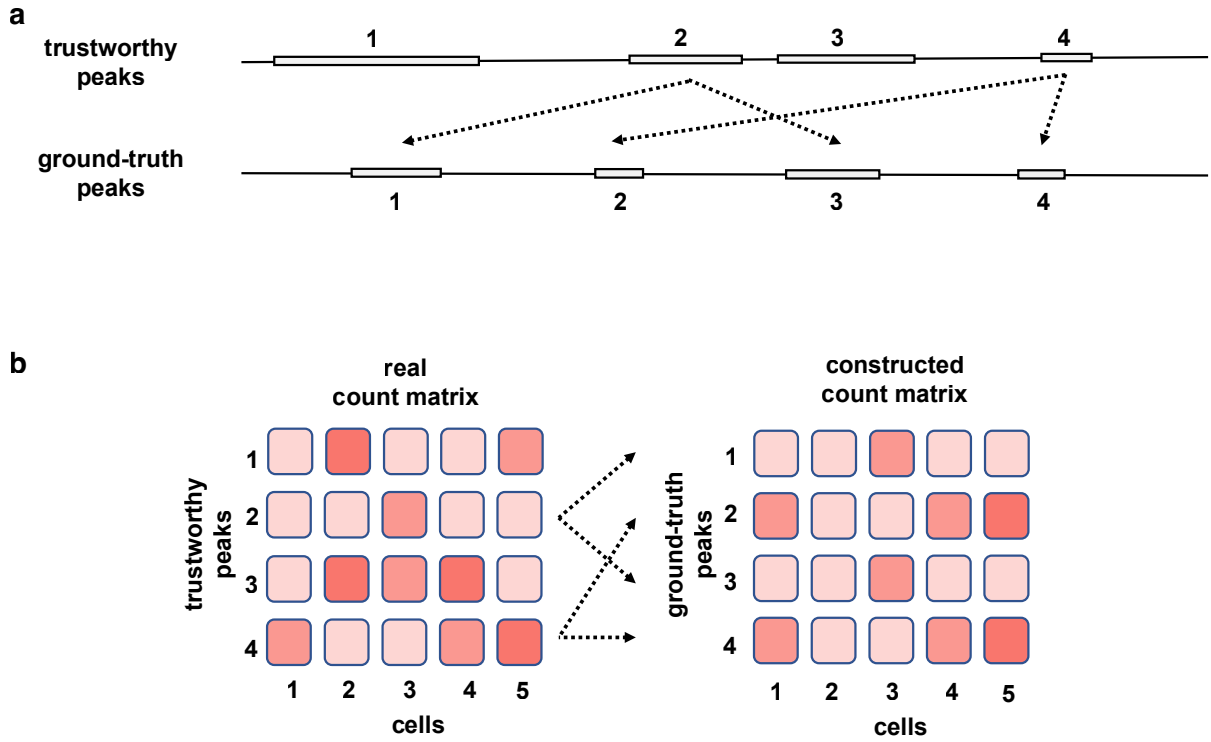


Figure 2.7: The peak-mapping step in scReadSim’s data generation with user-designed open chromatin regions (i.e., ground-truth peaks).

a, The mapping between trustworthy peaks and ground-truth peaks. For every ground-truth peak, scReadSim finds its most similar trustworthy peak (in the same chromosome) in terms of length. **b**, The constructed count matrix for ground-truth peaks. For every ground-truth peak, scReadSim adds the corresponding trustworthy peak’s row in the trustworthy peak-by-cell count matrix to the ground-truth peak-by-cell count matrix.

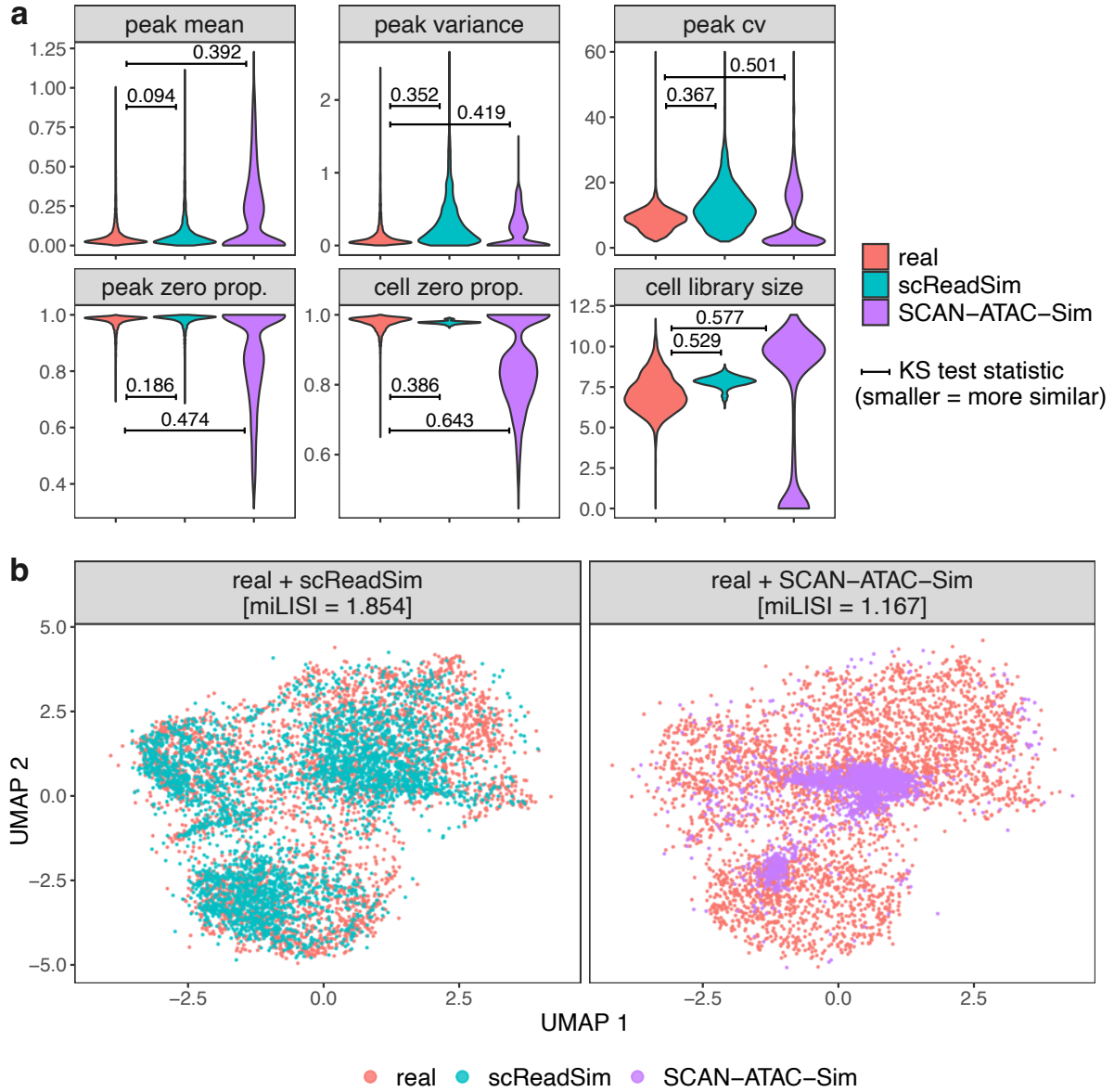


Figure 2.8: Synthetic cells generated by SCAN-ATAC-Sim do not mimic real cells at the read-count level.

a, Summary statistics of the count matrix at the peak level (mean, variance, coefficient of variance (cv), and zero proportion) and the cell level (zero proportion and cell library size). Kolmogorov–Smirnov tests (KS tests) are performed to compare the empirical distributions of summary statistics between the real and synthetic read count matrices. Smaller KS statistics indicate that the synthetic data mimic the real data better. **b**, UMAP visualization of the pooled real and synthetic cells. miLISI measures the similarity of real and synthetic cells in the UMAP space: the miLISI value ranges between 1 and 2, with 2 indicating a perfect mixing of real and synthetic cells.

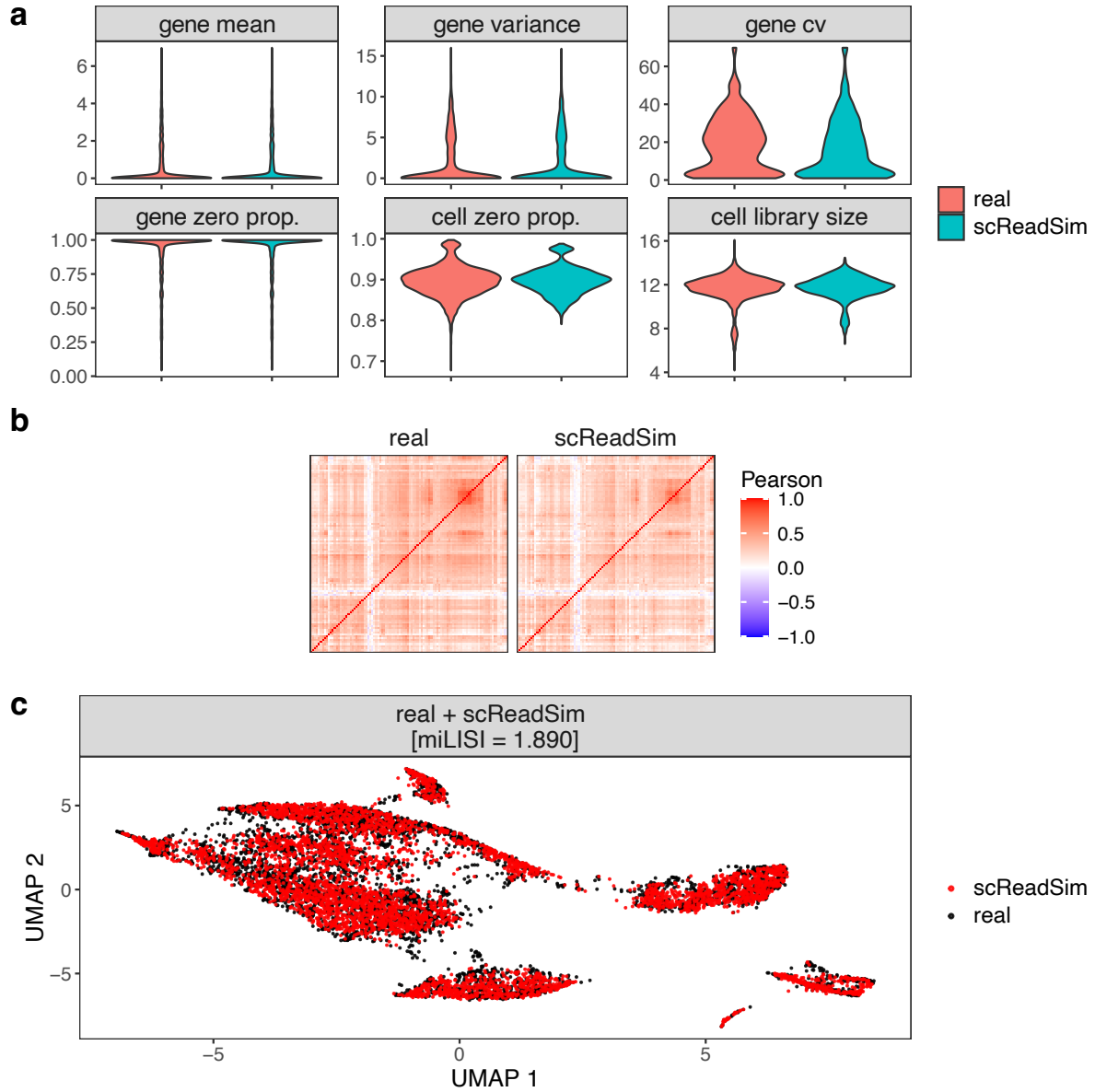


Figure 2.9: scReadSim's synthetic data resemble real mouse 10x single-cell Multiome dataset (the RNA-seq modality only) [3] at the UMI-count level.

a, Summary statistics of the UMI count matrix at the gene level (mean, variance, coefficient of variance (cv), and zero proportion) and the cell level (zero proportion and cell library size). **b**, Correlations among the 100 top-expressed genes in the synthetic and real count matrices. The top-expressed genes are defined based on the real count matrix. **c**, UMAP visualization of the pooled real and synthetic cells. miLISI measures the similarity of real and synthetic cells in the UMAP space: the miLISI value ranges between 1 and 2, with 2 indicating a perfect mixing of real and synthetic cells.

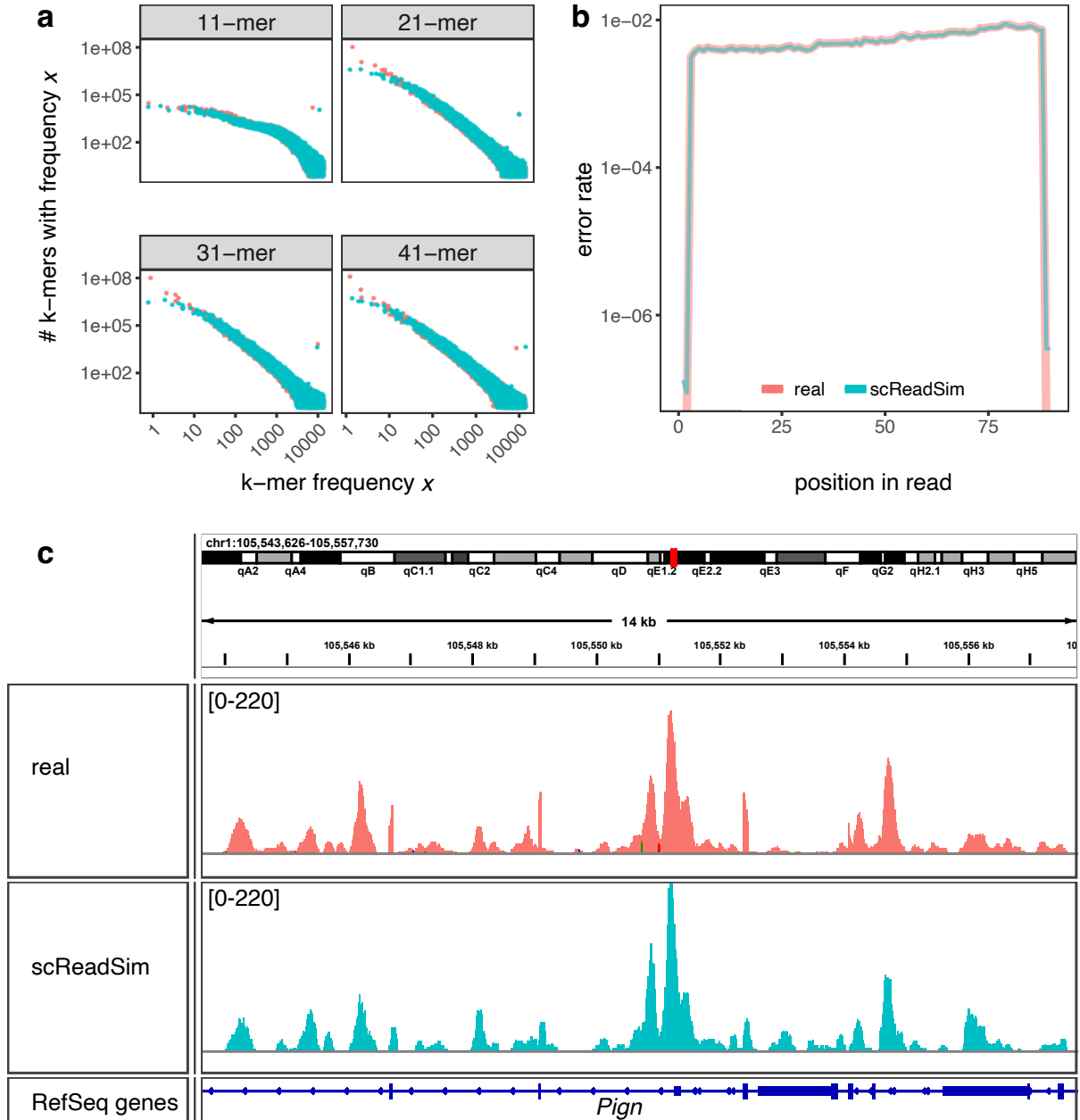


Figure 2.10: scReadSim's synthetic data resemble real mouse 10x single-cell Multiome dataset (the RNA-seq modality only) [3] at the read-sequence level.

a, Comparison of the k-mer spectra between scReadSim's synthetic data and real data. The x-axis refers to the occurrence frequency of a specific k-mer, and the y-axis represents the number of unique k-mers with this frequency. Both the x-axis and y-axis are on the $\log_{10}$ scale. **b**, Comparison of the error rate per base call within a read between scReadSim's synthetic data and real data. The x-axis represents the positions of bases within a read, and the y-axis refers to the substitution error rate in each position. **c**, Read coverage comparison of the real and synthetic BAM files in the IGV genome browser [4]. The genome browser's track height is set to 220 for both tracks.

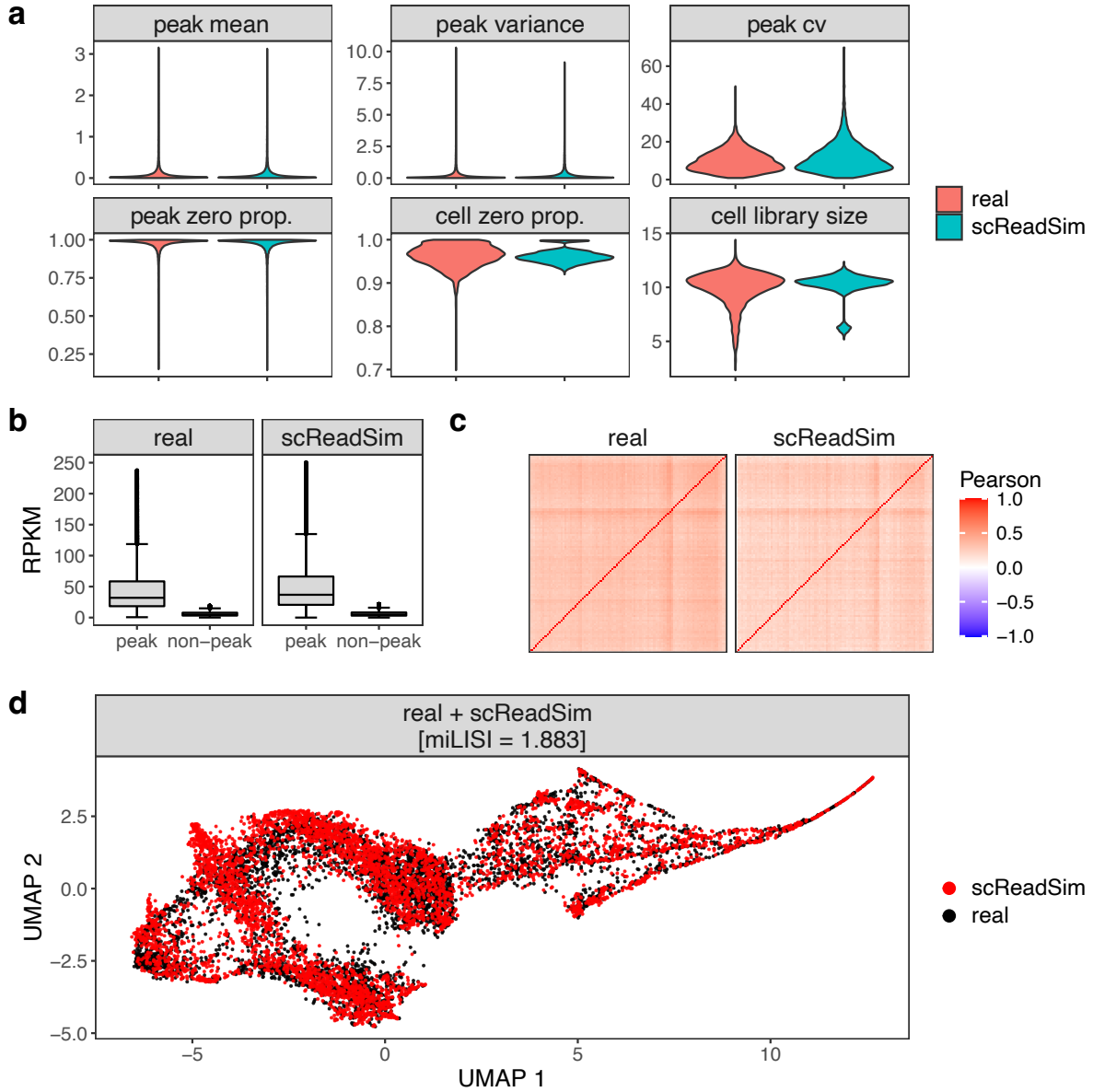


Figure 2.11: scReadSim's synthetic data resemble real mouse 10x single-cell Multiome dataset (the ATAC-seq modality only) [3] at the read-count level.

a, Summary statistics of the count matrix at the peak level (mean, variance, coefficient of variance (cv), and zero proportion) and the cell level (zero proportion and cell library size). **b**, Comparison of the RPKM (Reads Per Kilobase Million) value distributions in peak and non-peak regions between scReadSim's synthetic data and real data. Peaks ($n_{\text{peak}} = 13606$) and non-peaks ($n_{\text{non-peak}} = 24286$) are obtained using MACS3 from the real BAM file. **c**, Correlations among the 100 top-open peaks in the synthetic and real count matrices. The top-open peaks are defined based on the real count matrix. **d**, UMAP visualization of the pooled real and synthetic cells. miLISI measures the similarity of real and synthetic cells in the UMAP space: the miLISI value ranges between 1 and 2, with 2 indicating a perfect mixing of real and synthetic cells.

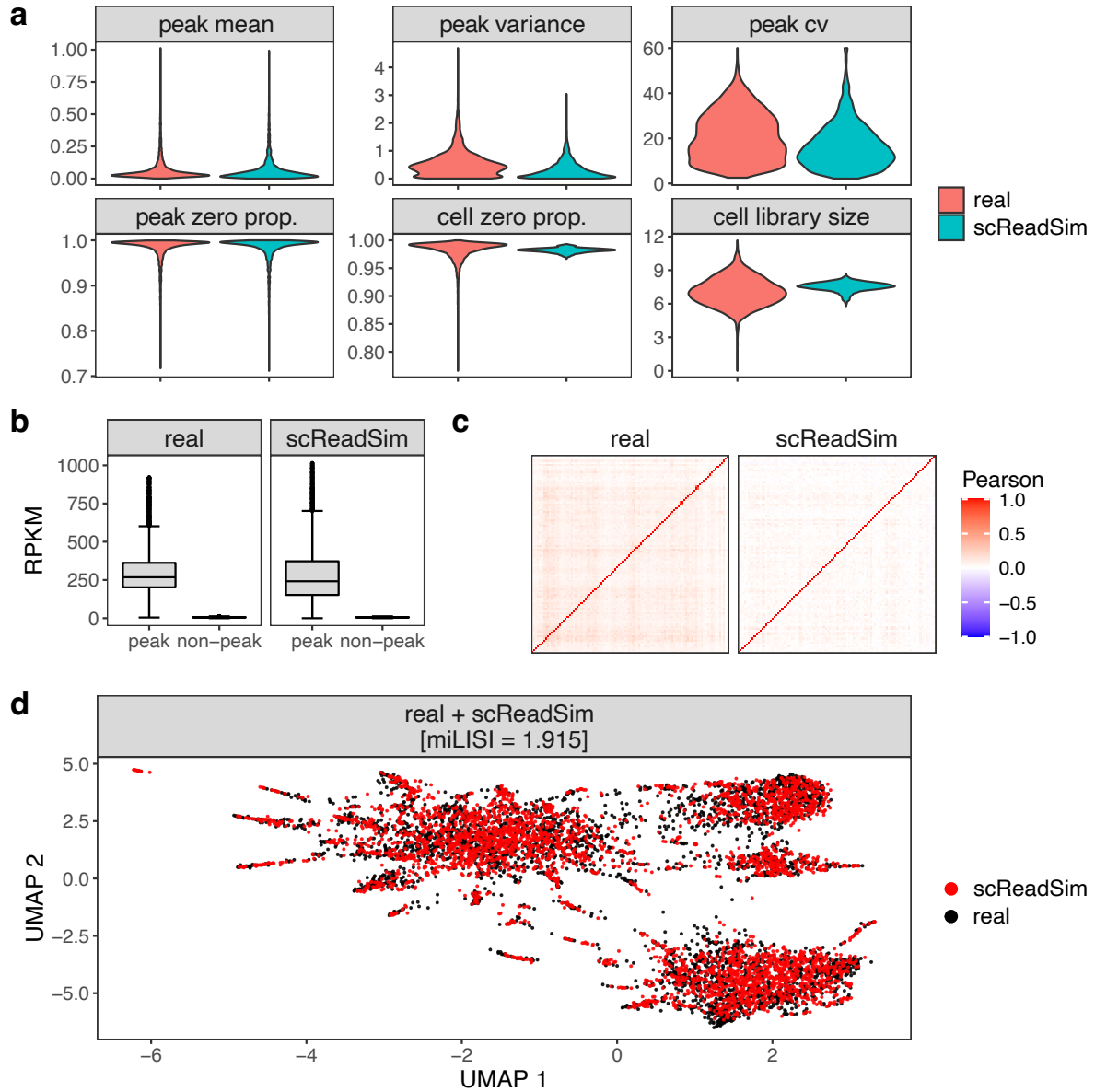


Figure 2.12: scReadSim's synthetic data resemble real sci-ATAC-seq data [5] at the read-count level.

a, Summary statistics of the count matrix at the peak level (mean, variance, coefficient of variance (cv), and zero proportion) and the cell level (zero proportion and cell library size). **b**, Comparison of the RPKM (Reads Per Kilobase Million) value distributions in peak and non-peak regions between scReadSim's synthetic data and real data. Peaks ($n_{\text{peak}} = 3644$) and non-peaks ($n_{\text{non-peak}} = 12816$) are obtained using MACS3 from the real BAM file. **c**, Correlations among the 100 top-open peaks in the synthetic and real count matrices. The top-open peaks are defined based on the real count matrix. **d**, UMAP visualization of the pooled real and synthetic cells. miLISI measures the similarity of real and synthetic cells in the UMAP space: the miLISI value ranges between 1 and 2, with 2 indicating a perfect mixing of real and synthetic cells.

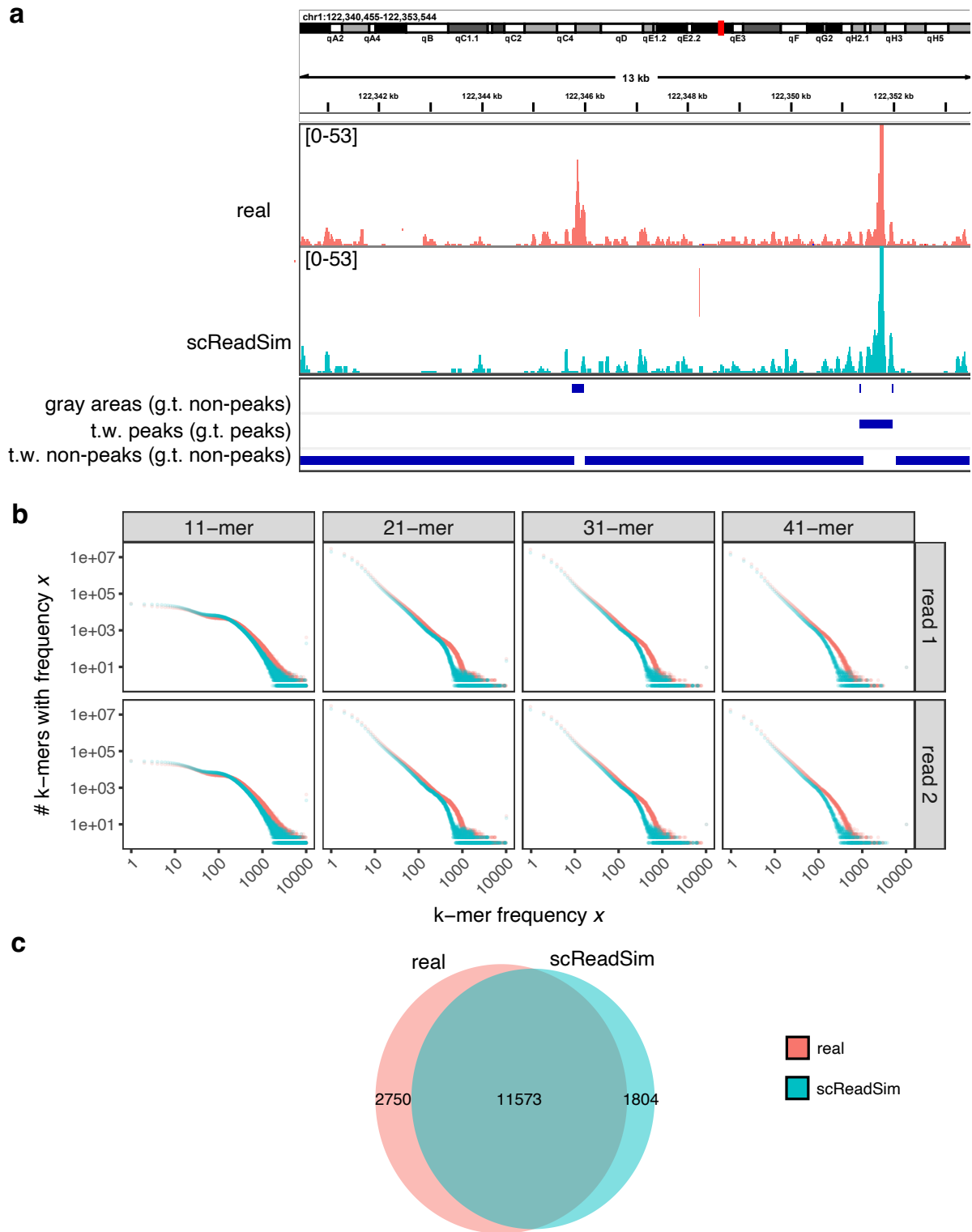


Figure 2.13: scReadSim's synthetic data resemble real mouse 10x single-cell Multiome dataset (the ATAC-seq modality only) [3] at the read-sequence level.

a, Read coverage comparison of the real and synthetic BAM files in the IGV genome browser [4]. The genome browser's track height is set to 53 for both tracks. Gray areas, trustworthy (t.w.) peaks, and trustworthy non-peaks are user-specified or scReadSim-identified from real data. scReadSim converts gray areas to ground-truth (g.t.) non-peaks, and it maintains trustworthy peaks and non-peaks as ground-truth peaks and non-peaks, respectively, in synthetic data. **b**, Comparison of the k-mer spectra between scReadSim's synthetic data and real data. The x-axis refers to the occurrence frequency of a specific k-mer, and the y-axis represents the number of unique k-mers with this frequency. Both the x-axis and y-axis are on the $\log_{10}$ scale. **c**, A Venn diagram of MACS3's called peaks from scReadSim's synthetic data and real data. A peak called from real data (referred to as a "real peak") is considered to overlap with a peak called from synthetic data (referred to as a "synthetic peak") if the real peak has at least 101 bp (half of the shortest real peak's length) overlapped by the synthetic peak.

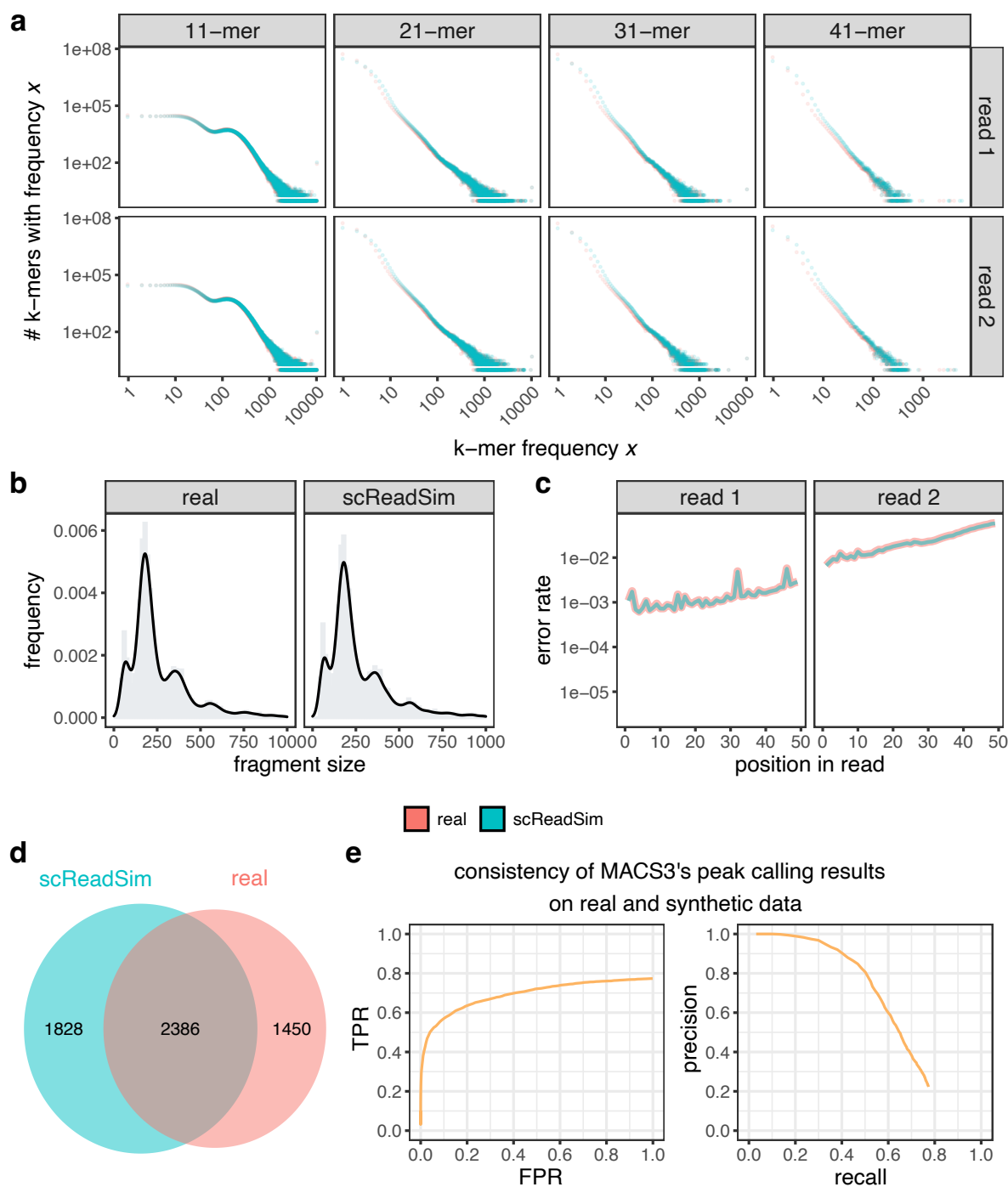


Figure 2.14: scReadSim's synthetic data resemble real sci-ATAC-seq data [5] at the read-sequence level.

a, Comparison of the k-mer spectra between scReadSim's synthetic data and real data. The x-axis refers to the occurrence frequency of a specific k-mer, and the y-axis represents the number of unique k-mers with this frequency. Both the x-axis and y-axis are on the $\log_{10}$ scale. **b**, Comparison of the fragment-size distributions between scReadSim's synthetic data and real data. **c**, Comparison of the error rate per base call within a read between scReadSim's synthetic data and real data. The x-axis represents the positions of bases within a read, and the y-axis refers to the substitution error rate in each position. **d**, A Venn diagram of MACS3's called peaks from scReadSim's synthetic data and real data. A peak called from real data (referred to as a "real peak") is considered to overlap with a peak called from synthetic data (referred to as a "synthetic peak") if the real peak has at least 135 bp (half of the shortest real peak's length) overlapped by the synthetic peak. **e**, True positive rate (TPR) vs. false positive rate (FPR) (left) and precision vs. recall (right) for evaluating the synthetic peaks in (d) by treating the real peaks in (d) as truths. The TPR, equivalent to the recall, is the proportion of real peaks that have at least 135 bp overlapped by any synthetic peak; the FPR is the proportion of real non-peaks (the regions complementary to the real peaks) that have at least 135 bp overlapped by any synthetic peak; the precision is the proportion of synthetic peaks that overlap at least 135 bp of any real peak.

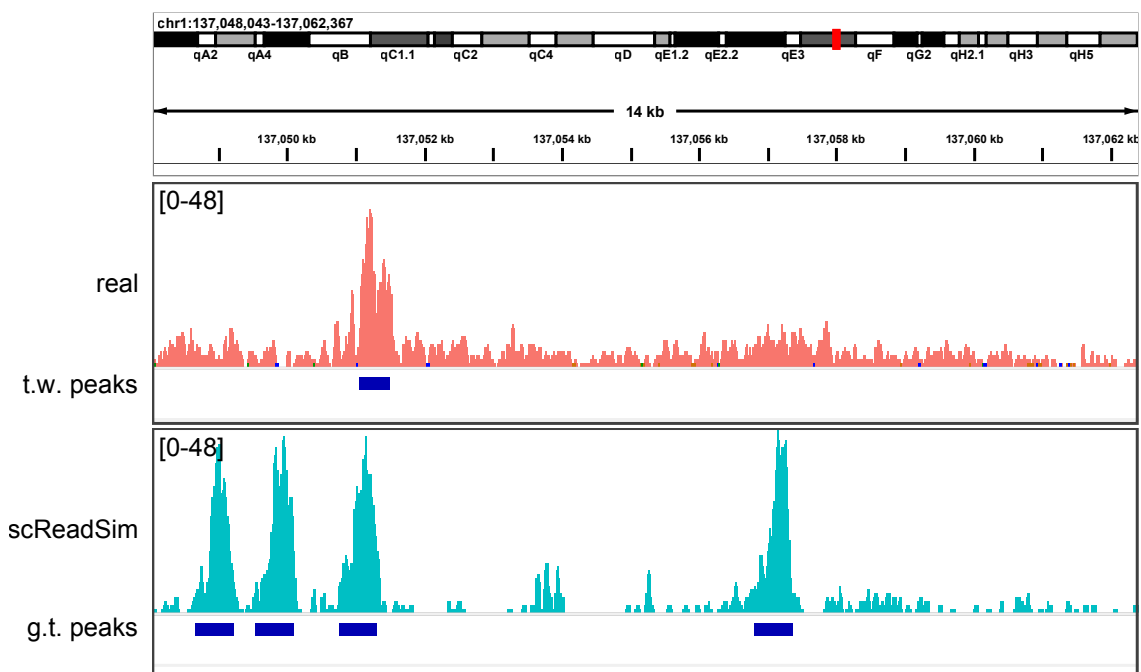


Figure 2.15: IGV genome browser [4] visualization of scReadSim's synthetic sci-ATAC-seq reads with user-designed ground-truth (g.t.) peaks (bottom track).

From the real sci-ATAC-seq reads (top track) [5], the trustworthy (t.w.) peaks are called by MACS3 and used to train scReadSim. The genome browser's track height is set to 48 for both tracks.

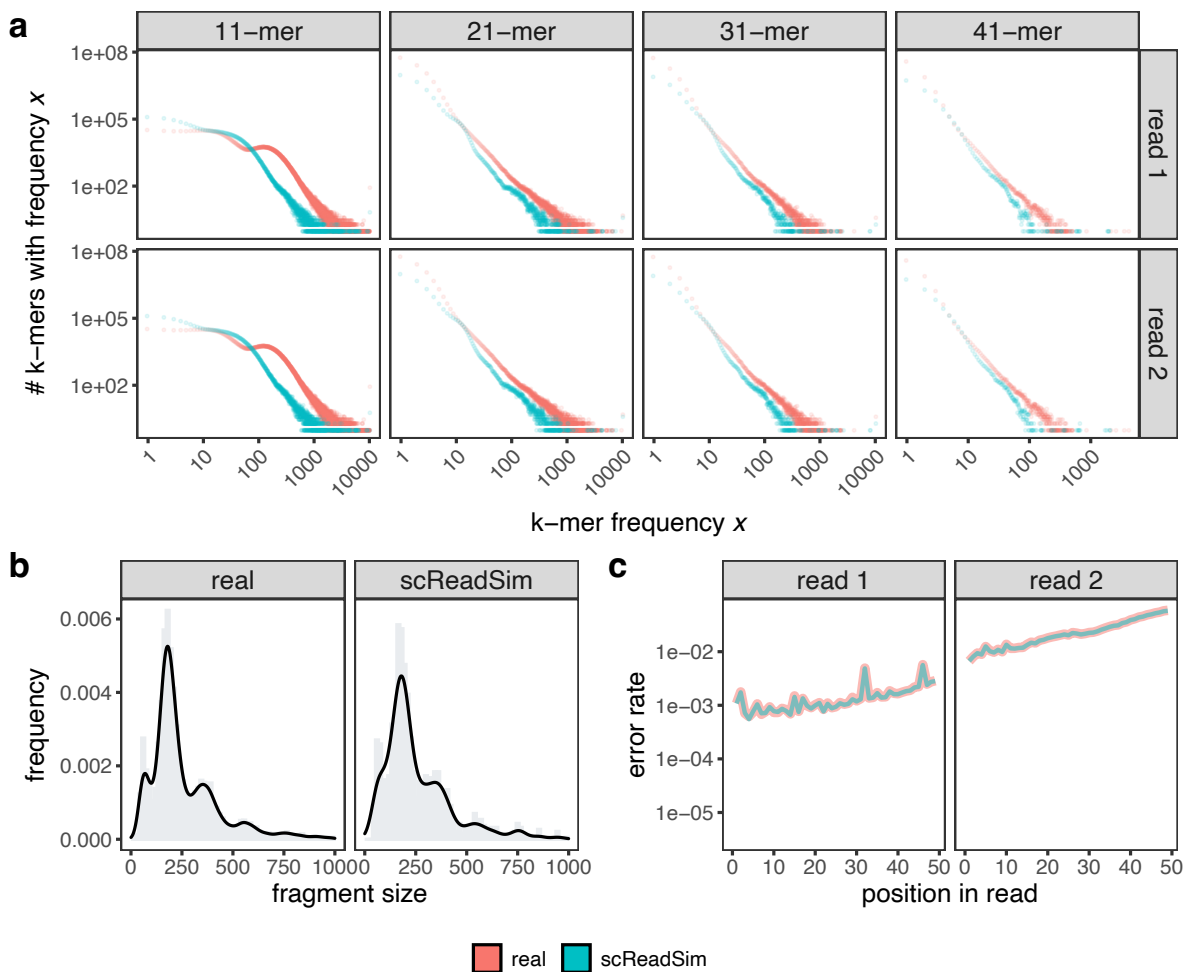


Figure 2.16: scReadSim's synthetic data with user-designed open chromatin regions resemble real sci-ATAC-seq data [5] at the read-sequence level.

a, Comparison of the k-mer spectra between scReadSim's synthetic data and real data. The x-axis refers to the occurrence frequency of a specific k-mer, and the y-axis represents the number of unique k-mers with this frequency. Both the x-axis and y-axis are on the $\log_{10}$ scale. The real reads and scReadSim's synthetic reads are expected to have different k-mer spectra because they correspond to different open chromatin regions. **b**, Comparison of the fragment-size distributions between scReadSim's synthetic data and real data. **c**, Comparison of the error rate per base call within a read between scReadSim's synthetic data and real data. The line plot's x-axis represents the positions of bases within a read, and the y-axis refers to the substitution error rate corresponding to each position.

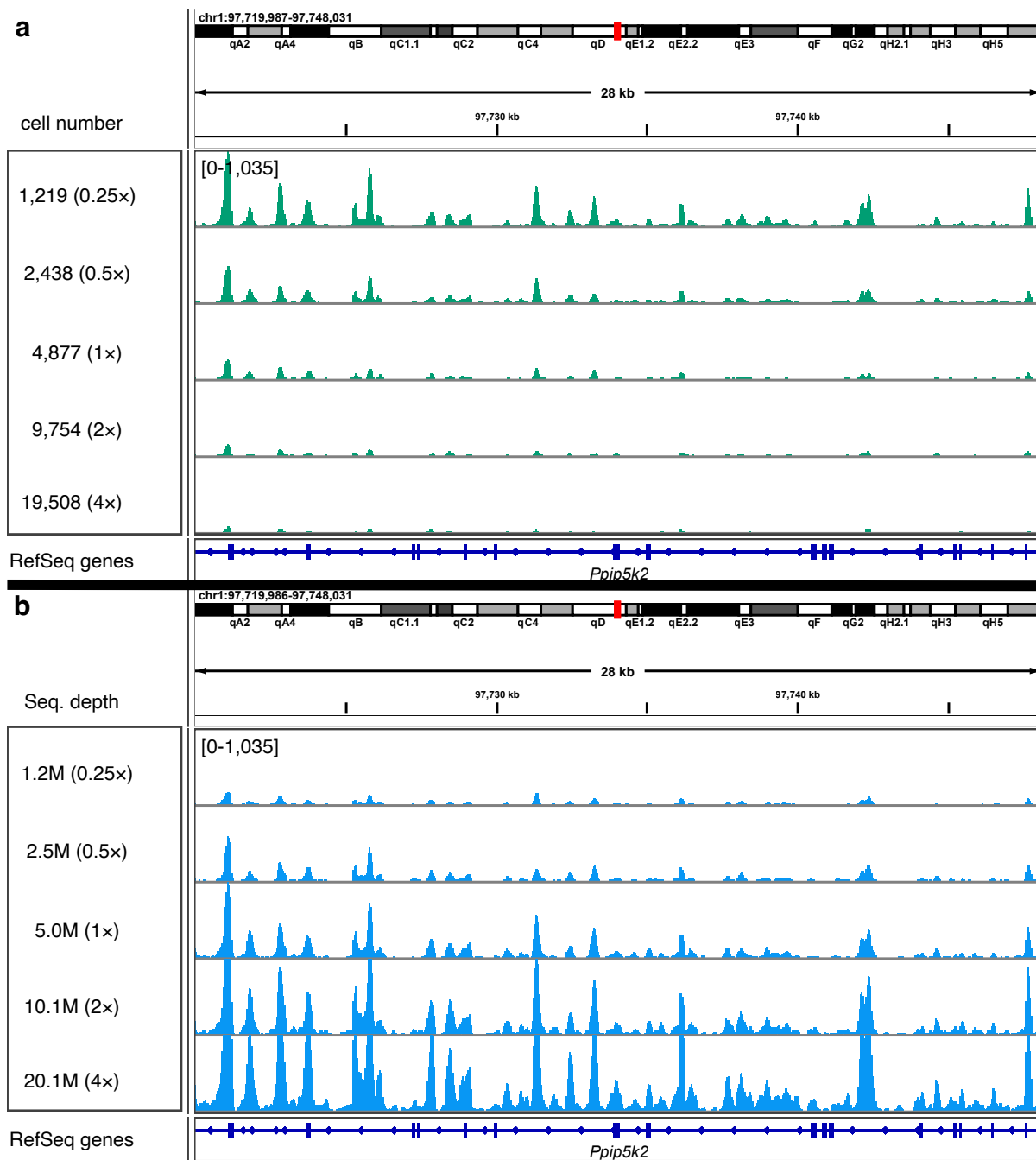


Figure 2.17: scReadSim allows varying cell number and sequencing depth as input parameters, with the corresponding synthetic data visualized in the IGV genome browser [4].

a, Varying numbers of synthetic cells. The track names indicate the multiplication factors: ‘1x’ specifies the cell number in the real data; ‘4x’ specifies 4 times the real cell number. All five synthetic datasets are downsampled to 1,219 cells for a fair comparison. **b**, Varying sequencing depths in synthetic data. The track names indicate the multiplication factors: ‘1x’ specifies the sequencing depth in the real data; ‘4x’ specifies 4 times the real sequencing depth. For better visualization, the genome browser’s track height is set to 1,035 for all 10 tracks.

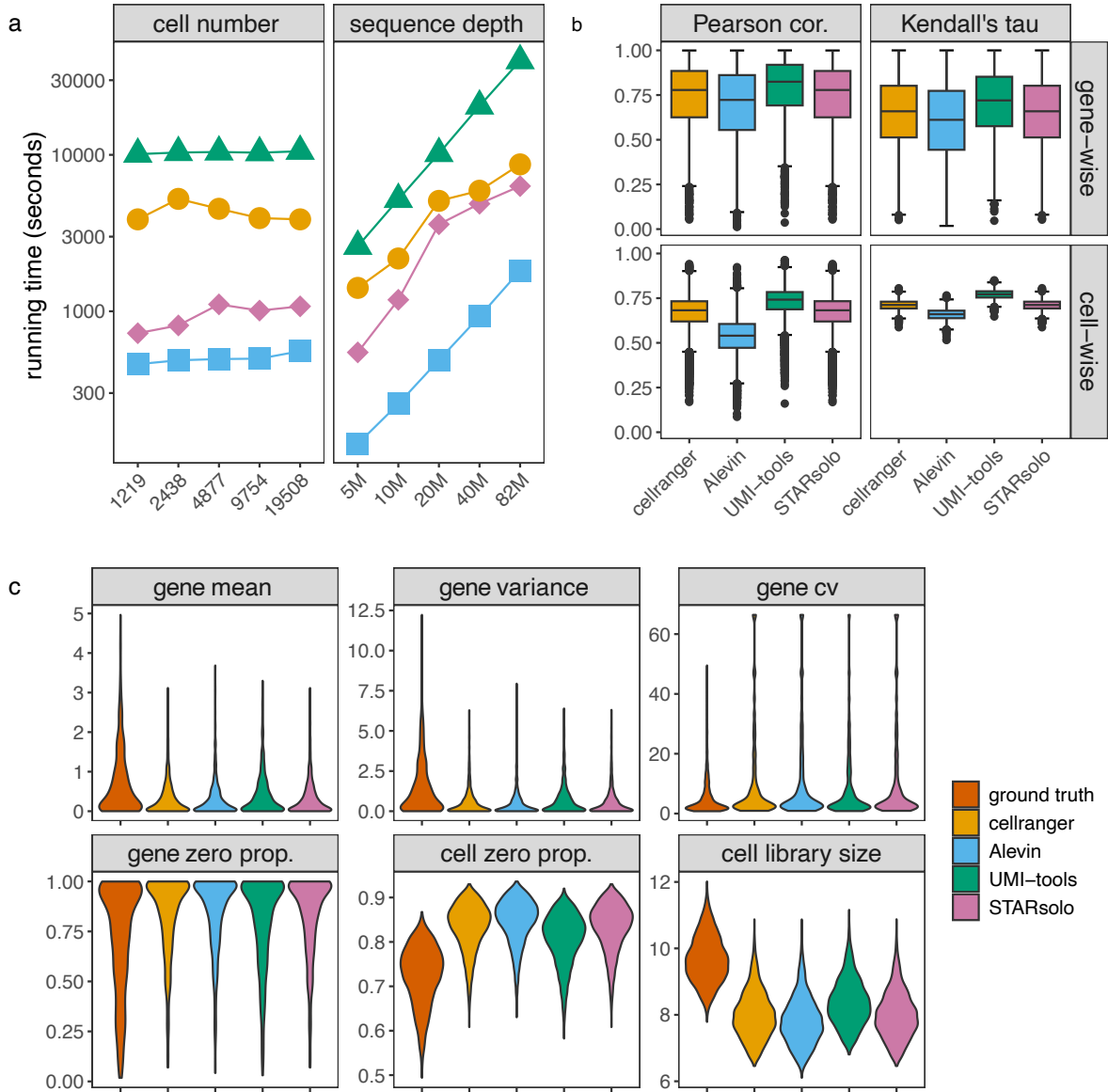


Figure 2.18: Benchmark of UMI deduplication tools using scReadSim's synthetic scRNA-seq reads.

The ground-truth UMI count matrix is the gene-by-cell UMI count matrix generated by scReadSim. Four deduplication tools are considered: cellranger, Alevin, UMI-tools, and STARsolo. **a**, Time usage of deduplication tools on synthetic datasets with varying cell numbers or sequencing depths. The y-axis indicates the time lapse (in seconds), and the x-axis reflects the number of synthetic cells or the total number of synthetic reads (sequencing depth). **b**, Cell-wise ($n_{\text{cell}} = 3256$) and gene-wise ($n_{\text{gene}} = 17652$) correlations (Pearson correlation and Kendall's tau) between the ground-truth UMI count matrix and each deduplication tool's output UMI count matrix. **c**, Summary statistics of the UMI count matrices at the gene level (mean, variance, coefficient of variance (cv), and zero proportion) and the cell level (zero proportion and cell library size).

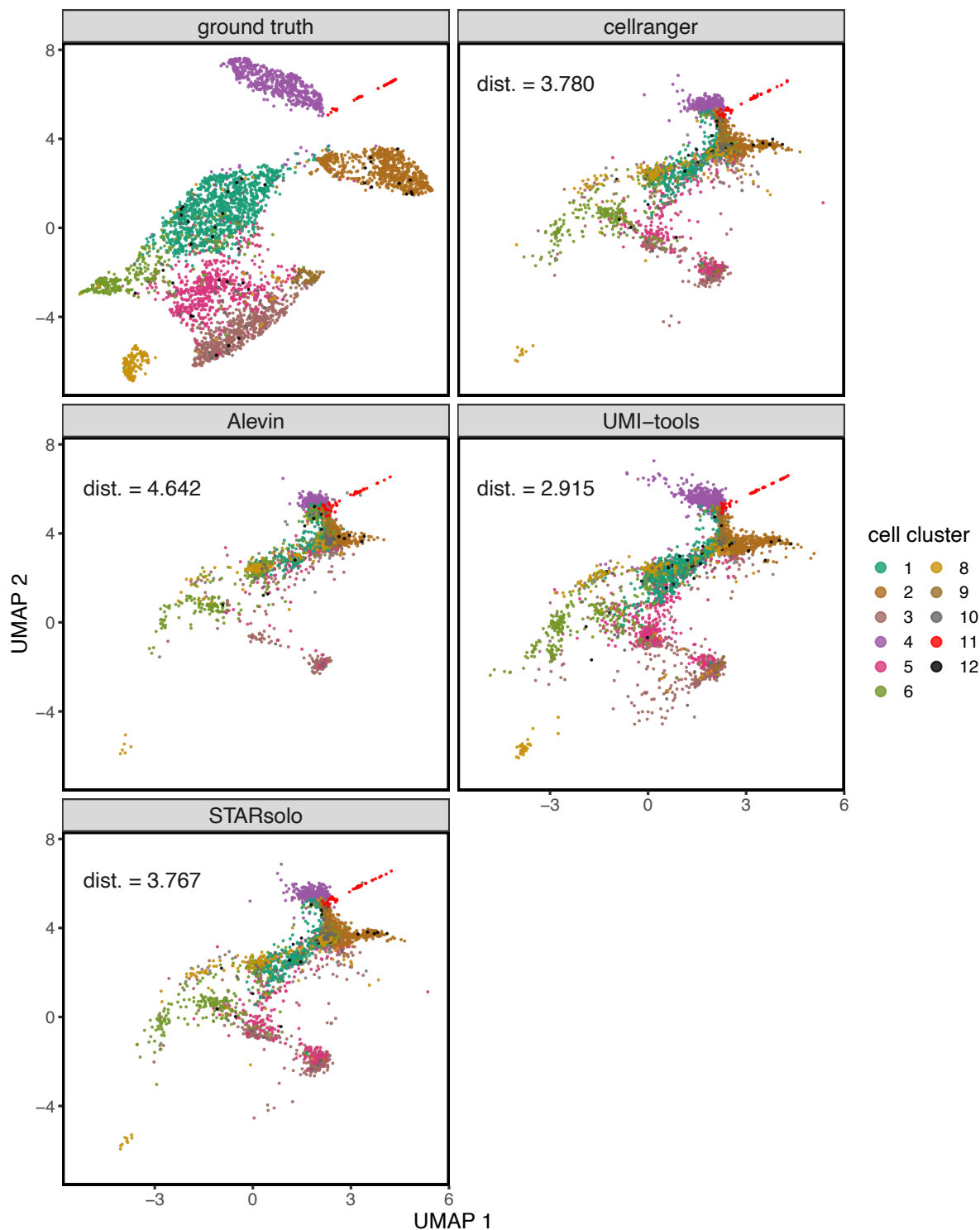
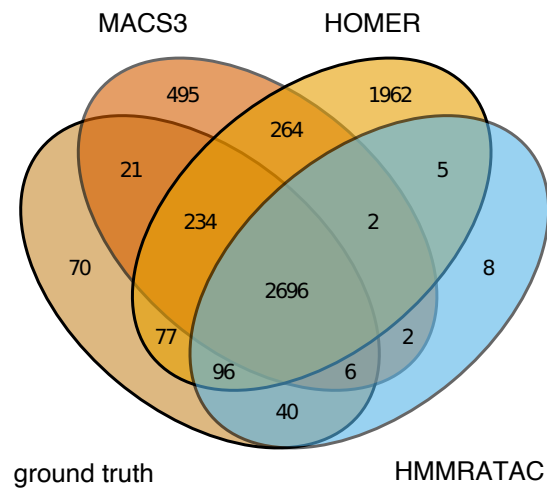


Figure 2.19: UMAP visualizations of the ground-truth UMI count matrix and each deduplication tool's output UMI count matrix.

Cells are colored by the cell clusters outputted by scReadSim (the clusters are from the real data used to train scReadSim). The synthetic cells' deduplicated UMI counts are projected to the UMAP space defined by the same cells' ground-truth UMI counts. We computed the Euclidean distance between each synthetic cell's two UMAP coordinates, calculated based on the cell's ground-truth UMI counts and deduplicated UMI counts, respectively. The mean value of the Euclidean distances of all synthetic cells is displayed for each UMI deduplication tool: a smaller value indicates that the deduplicated UMI count matrix better agrees with the ground truth UMI count matrix in UMAP visualization.

a



b

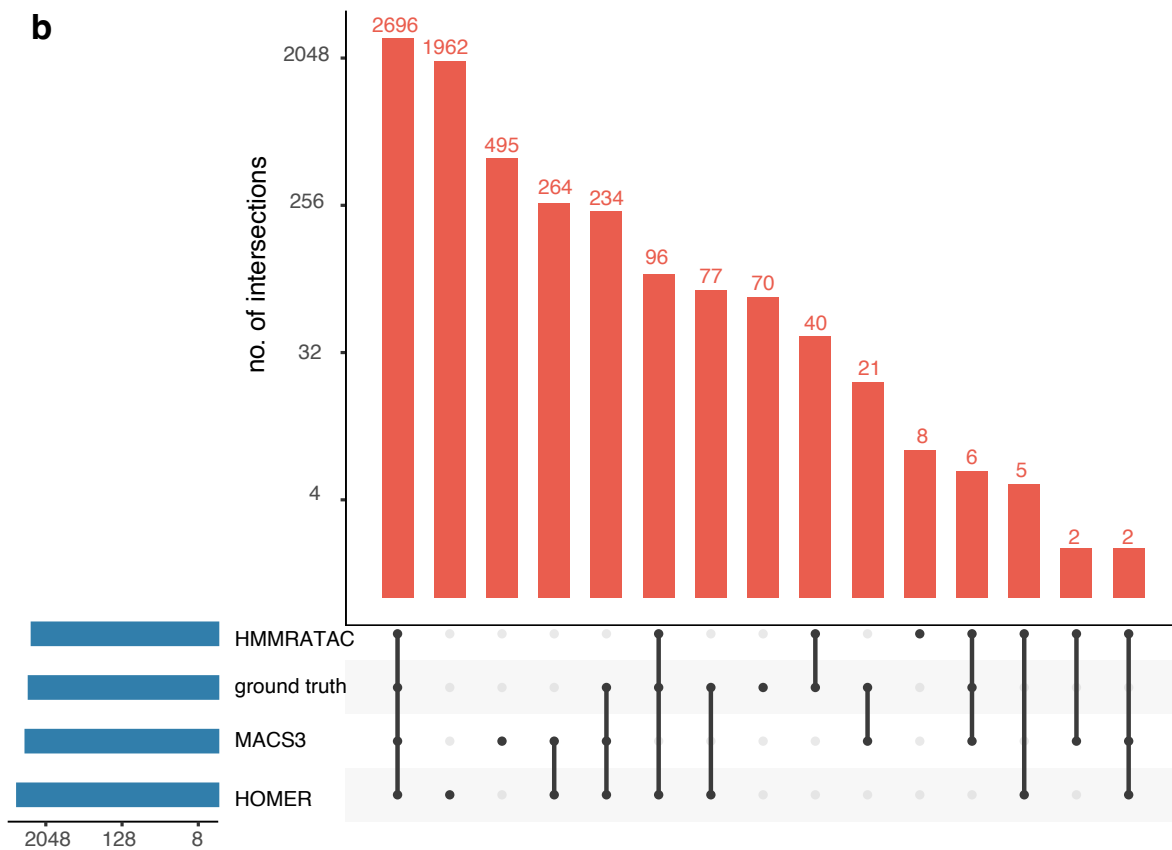


Figure 2.20: Benchmark of peak-calling tools (MACS3, HOMER, and HMMRATAC) using scReadSim’s synthetic scATAC-seq reads generated from user-designed ground-truth peaks (i.e., open chromatin regions).

The Venn diagram (a) and upset plot (b) of the user-designed open chromatin regions and the peaks called by MACS3, HOMER, and HMMRATAC. See Supplementary Fig. 2.21 for the comparison results that include the peak-calling tool SEACR, whose called peaks differ drastically from the ground truth and the peaks called by MACS, HOMER, and HMMRATAC.

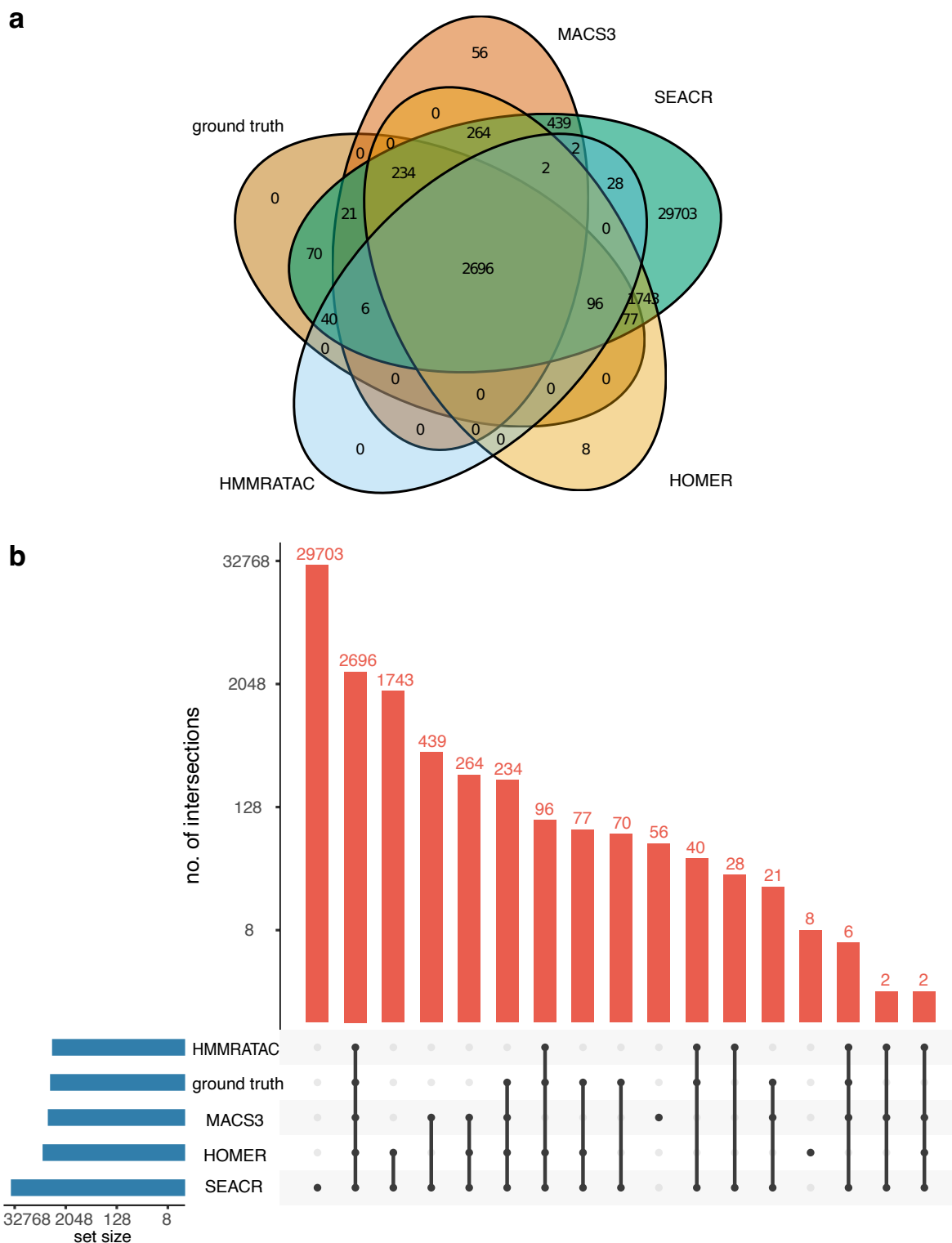


Figure 2.21: Benchmark of peak-calling tools (MACS3, SEACR, HOMER, and HMMRATAC) using scReadSim's synthetic scATAC-seq reads generated from user-designed ground-truth peaks (i.e., open chromatin regions).

The Venn diagram (a) and upset plot (b) of the user-designed open chromatin regions and the peaks called by MACS3, SEACR, HOMER, and HMMRATAC.

S2.8.3 Supplementary tables

Table S2.1: Comparison of scReadSim, minnow, and SCAN-ATAC-Sim

Tools Property	scReadSim	minnow	SCAN-ATAC-Sim
Modality	scRNA-seq scATAC-seq	scRNA-seq	scATAC-seq
Input	scRNA-seq reads or scATAC-seq reads gene annotations	UMI count matrix gene annotations	ATAC-seq reads peak lists
Output	FASTQ or BAM	FASTQ	BED
Ground truths	UMI counts open chromatin re- gions	UMI counts	N/A
Read-sequence preservation ¹	✓	✗ ²	✗
Read-count preservation ³	✓	✓	✗
Read-coverage preservation ⁴	✓	✗	✗ ⁵
Whole-genome read simulation	✓	✗	✓

¹: “Read-sequence preservation” means whether the simulator preserves the characteristics of real read sequences, such as k-mer spectra and sequencing errors.

²: Since minnow does not input scRNA-seq reads, it uses its internal, pre-trained model to generate sequencing errors.

³: “Read-count preservation” means whether the simulator generates synthetic cells that mimic real cells regarding gene or peak counts.

⁴: “Read-coverage preservation” means whether the simulator’s synthetic reads have a coverage profile mimicking that of real reads at the pseudo-bulk level.

⁵: SCAN-ATAC-Sim only inputs pseudo-bulk ATAC-seq reads and thus does not preserve read coverage at the single-cell level.

CHAPTER 3

scIsoSim: simulating single-cell RNA-seq reads with full-length isoform ground truths

3.1 Introduction

RNA-Seq has revolutionized transcriptomic research, providing comprehensive and quantitative insights into the transcriptome. However, accurately identifying and quantifying isoforms and AS events remains challenging due to the high structural similarity among isoforms and the complexity of splicing patterns. To address these issues, numerous computational tools have been developed for isoform quantification [21–26, 73] and AS analysis [27–31]. Benchmark studies have evaluated the performance of these tools, showing comparable levels of accuracy for isoform quantification [74–76] and varying strengths for AS detection [77–79]. Nevertheless, bulk RNA-Seq cannot capture cell-to-cell variability in isoform expression and splicing patterns, which is essential for studying cellular heterogeneity [80] and understanding its broader implications for biological processes, such as development [81], and disease progression [82].

The advancement of single-cell RNA-seq (scRNA-seq) has enabled researchers to study isoforms and AS at the single-cell level. Among the most commonly used next-generation sequencing (NGS) technologies [83], plate-based protocols such as Smart-seq2 [34] and CEL-seq [84, 85] generate full-length transcript coverage, allowing for the resolution of isoforms that differ at any position along the transcript. In contrast, droplet-based platforms such as the 10x Genomics Chromium system [15] profile transcripts only at one end—either the 3' or 5'—leading to end-biased sequencing data. Specifically, 10x 3' protocols capture polyadenylated transcripts at their 3' ends, enabling the distinction of isoforms only if they differ in

their terminal exons or polyadenylation sites. 10x 5' protocols, on the other hand, target the 5' ends, allowing resolution of transcription start site (TSS) variation but limited coverage of downstream splicing events. These protocol-specific biases constrain the scope of isoform analysis and require tailored computational approaches to interpret the resulting data accurately (Supplementary Fig. 3.6, Table S3.1).

In practice, isoform quantification in scRNA-seq data has been approached in different ways. For full-length Smart-seq2 data, bulk-level tools such as RSEM [21] or kallisto [22] are often applied to individual cells. For 3'-biased 10x data, tools like Scasa [35] have been developed to account for cases where isoforms are indistinguishable due to shared 3' ends, grouping them into "paralogs." In parallel, tools specifically developed for AS analysis in scRNA-seq, such as brie [86], outrigger [37], and brie2 [36], have been introduced to infer splicing variation across cells and conditions. Despite these methodological advances, it remains unclear whether current practices—especially the adaptation of bulk-level quantifiers or specialized heuristics for droplet-based protocols—are valid. The primary limitation is the lack of experimental ground truths: we rarely know the true isoform composition and abundance in each cell.

Answering such questions requires simulators specifically designed for scRNA-seq that provide ground truths for benchmarking isoform quantification tools and AS analysis pipelines. Previously, we developed scReadSim, a sequencing read simulator capable of generating realistic scRNA-seq reads with gene abundance as ground truths. However, scReadSim lacks isoform-level information, i.e., genetic splicing mechanism, making it unsuitable for benchmarking at the isoform resolution (Table S3.2). To the best of our knowledge, no existing standalone open-source simulators are capable of this functionality [33].

In a prior benchmark study evaluating the performance of bulk RNA-seq isoform quantification tools on simulated plate-based scRNA-seq data (e.g., SMARTer and SMART-seq2) [33], two distinct approaches were used to simulate single-cell RNA-seq reads (Table S3.2). The first approach employed RSEM [21], a bulk RNA-seq isoform quantification tool that also supports RNA-seq read simulation. To adapt RSEM for scRNA-seq data, individual cells were treated as bulk samples, and RSEM was trained on each cell separately. During

simulation, RSEM provided ground truth for isoform expression by recording the origin of each simulated read within the transcriptome. The second approach combined Kallisto [22], Splatter [7], and Polyester [32]. Kallisto was first applied to real scRNA-seq data to generate an isoform-by-cell abundance matrix, which was then processed by Splatter to simulate a synthetic abundance matrix. Finally, Polyester used this synthetic matrix to generate RNA-seq reads for each synthetic cell.

Despite their utility, these simulation schemes have notable limitations. First, neither approach supports droplet-based scRNA-seq technologies, as they lack the functionality to simulate cell barcodes or unique molecular identifiers (UMIs). Second, both approaches are inflexible regarding isoform composition, preserving isoform-level transcript abundance from input data without allowing modifications to exon-intron combinations or isoform proportions. Third, Splatter fails to preserve gene-gene correlations [10] and loses gene identity during simulation [33], resulting in less realistic synthetic data due to the reassignment of gene names to Polyester in the subsequent step. Finally, RSEM does not allow users to control key parameters, such as the number of cells or sequencing depth, limiting its ability to customize synthetic datasets for specific experimental needs.

To address this gap, we developed scIsoSim (Fig. 3.1, Supplementary Fig. 3.7), the first simulator capable of generating realistic scRNA-seq reads from isoforms with predefined structures and proportions. scIsoSim supports various NGS technologies, including Smart-seq2 and 10x 3' and 5' protocols, and mimics read sequences and splicing events observed in real scRNA-seq data. In this work, we demonstrate the utility of scIsoSim as a benchmarking tool through three key applications. First, we investigated whether bulk-level isoform quantification tools can be applied to scRNA-seq data. Using scIsoSim to simulate Smart-seq2 scRNA-seq data, we benchmarked four popular tools: Kallisto, Salmon, RSEM, and Terminus. Our results show that for the Smart-seq2 scRNA-seq protocol, all bulk-level tools deliver accurate and robust performance, with Salmon demonstrating the best computational efficiency. Second, we tested the reliability of Scasa, the only scRNA-seq-specific isoform quantification tool designed for 10x 3' protocol. By simulating a 10x 3' scRNA-seq dataset, we found that Scasa achieved limited accuracy, likely due to the sparsity of single

cell data. Finally, we assessed the performance of single-cell AS event quantification computational tools, outrigger and brie, using synthetic Smart-seq2 scRNA-seq data. The results show that while brie outperformed outrigger in overall accuracy, it does not always capture cell-specific AS events.

3.2 Methods

3.2.1 The scIsoSim method

The input of scIsoSim contains a BAM file storing the aligned scRNA-seq reads, a gene annotation file, an isoform proportion file, and a reference genome file (Fig. 3.1). As an overview, scIsoSim contains three major steps (Supplementary Fig. 3.7): synthetic gene abundance matrix generation, isoform ground truth specification, and synthetic read generation.

In the first step—synthetic gene abundance matrix generation—scIsoSim processes input reads from the BAM file to construct a gene-by-cell abundance matrix for genes listed in the input gene annotation file. For plate-based NGS technologies such as Smart-seq2, scIsoSim counts the number of reads overlapping each gene in every cell based on gene coordinates. For droplet-based sequencing technologies like 10x Chromium, which use UMIs to distinguish transcript molecules, scIsoSim instead counts UMIs, producing a gene-by-cell UMI count matrix. To generate synthetic gene-by-cell abundance matrices, scIsoSim then employs scDesign2 [10] or scDesign3 [87]. scDesign2 is particularly designed for scenarios where cells are grouped into distinct clusters. For cells that follow continuous trajectories or originate from different conditions, scDesign3 can be used instead to create the corresponding synthetic gene abundance matrices.

Additionally, scIsoSim extracts empirical reference distributions from the input data, including the number of reads per UMI $\hat{F}_{\text{UMI}}$ (for UMI-based protocols), gene body read coverage $\hat{F}_x$, and fragment length distributions $\hat{F}_L$ (for Smart-seq2), which are used in downstream read simulation unless specified otherwise by the user.

Specifically, the read coverage distribution $\hat{F}_x$ describes the distribution of sequencing

reads along the gene body. `scIsoSim` implements the `geneBody_coverage` function from RSeQC [88], which normalizes each transcript to 100 bp, calculates the number of reads covering each normalized position, and generates a coverage profile across the gene body. Optionally, `scIsoSim` provides pre-estimated distributions derived from publicly available datasets: a mouse hematopoietic single-cell dataset [89] for Smart-seq2, a human lung adenocarcinoma dataset [90] for 10x 3', and a human NSCLC tumor dataset [91] for 10x 5'.

The fragment length distribution $\hat{F}_L$ characterizes the cDNA fragment size distribution in Smart-seq2 data, typically ranging from 200 to 600 base pairs (bp) [34]. To estimate this distribution, `scIsoSim` implements `bamPEFragmentSize` from `deepTools` [67]. Users can either extract $\hat{F}_L$ from their own data or use pre-estimated distributions estimated from the mouse Smart-seq2 hematopoietic cell dataset.

In the second step—isoform ground truth specification, `scIsoSim` requires two inputs: a gene annotation file that lists the isoforms to be simulated and a corresponding proportion file (optional, detailed in the following Section 3.2.1.3) that records the relative abundance of each isoform within its gene at the cell type level. Using the synthetic gene-by-cell abundance matrix from Step 1 and the specified isoform proportion, `scIsoSim` generates a cell-type-specific synthetic isoform-by-cell abundance matrix. Specifically, for each gene in a synthetic cell, `scIsoSim` samples the gene's total abundance in that cell from a multinomial distribution, using the isoform proportions associated with the gene as the mean parameters.

In the third step—synthetic read generation, synthetic reads are generated using the empirical distributions from Step 1, the synthetic isoform-by-cell abundance matrix from the Step 2, the gene annotation file that lists the specified isoforms and the reference genome. The resulting synthetic reads are output in FASTQ format, with an option to convert to BAM format upon alignment. To preserve the unique characteristics of different sequencing protocols, the procedures for generating reads are tailored for Smart-seq2, 10x Genomics Chromium 3', and 10x Genomics Chromium 5' protocols. The protocol-specific details are as follows:

3.2.1.1 Synthetic read generation for Smart-seq2 scRNA-seq protocols

The following procedure describes the simulation of data for the Smart-seq2 scRNA-seq protocol [89]. This approach is also applicable to other plate-based scRNA-seq technologies, such as CEL-seq [84] and MARS-seq [1], which share similar read formats, including paired-end reads, the absence of UMIs, and full-length transcript coverage. To generate synthetic reads for the Smart-seq2 protocol, scIsoSim requires several inputs, including

- the synthetic isoform-by-cell abundance matrix based on read counts,
- the input BAM file,
- the reference genome,
- the read coverage distribution within gene body $\hat{F}_x$ (optional),
- the fragment length distribution $\hat{F}_L$ (optional),
- the read length r (optional; default 75 bp).

To generate synthetic reads, scIsoSim first generates a reference transcriptome file that records the transcripts' sequencing information for all isoforms based on the reference genome and gene annotation file. Then for each isoform in one synthetic cell, scIsoSim (1) extracts the corresponding count c (i.e., the number of synthetic reads) from the synthetic isoform-by-cell abundance matrix; (2) samples $\lceil c/2 \rceil$ read pairs (rounding $c/2$ to the least integer greater than or equal to $c/2$); (3) samples a fragment length l from the fragment length distribution $\hat{F}_L$ specified previously to represent the distance between paired-end reads for each read pair in $\lceil c/2 \rceil$; (4) samples fragment start position x (relative to reference transcriptome) from the read coverage distribution $\hat{F}_x$ for each read pair in $\lceil c/2 \rceil$; (5) generates $\lceil c/2 \rceil$ synthetic read pairs based on the fragment starting positions, the fragment length, the read length, and the reference transcriptome.

Note that Smart-seq2 paired-end reads are not strand-specific in the experimental data, meaning that the reads can originate from either strand of the gene. we assume that read

1 spans the range $x \rightarrow x + r$ on the sense strand (same strand as the reference transcript), while read 2 spans the range $x + l - r \rightarrow x + l$ on the reverse complement strand. Using the 5' and 3' positions of each synthetic read, scIsoSim extracts the corresponding sequences from the reference transcriptome, generating a total of $2 \cdot \lceil c/2 \rceil$ read sequences.

3.2.1.2 Synthetic read generation for 10x Genomics Chromium 3' and 5' scRNA-seq protocols

The following procedure describes the simulation of data for 10x 3' and 5' scRNA-seq protocols [90, 91]. This approach is also applicable to other droplet-based scRNA-seq technologies, such as Drop-seq [47], which use microfluidic droplet systems to capture single cells, prepare libraries, and assign unique barcodes and UMIs for high-throughput analysis.

To generate synthetic reads for 10x 3' and 10x 5' protocols, scIsoSim requires:

- the synthetic isoform-by-cell abundance matrix based on UMI counts,
- the input BAM file,
- the reference genome,
- the read coverage distribution within gene body $\hat{F}_x$ (optional),
- the read length r for read 2 (optional; default 126 bp).

For most droplet-based scRNA-seq technologies that use UMIs to distinguish sequencing reads originating from individual RNA molecules, every paired-end read has only one read (referred to as the read 2) containing the RNA sequencing information, whereas the other read (referred to as the read 1) contains the cell barcode and UMI information. The generation of these two ends is described below.

To generate read 2, scIsoSim generates a reference transcriptome file that stores the isoforms' transcript sequencing information based on the reference genome and gene annotation file. Then for each isoform i in one synthetic cell, scIsoSim (1) extracts the corresponding UMI count u_i (i.e., the number of synthetic UMIs) from the synthetic isoform-by-cell

abundance matrix; (2) samples the number of synthetic reads c_{ij} for each synthetic UMI $j = 1, \dots, u_i$ according to the empirical distribution $\hat{F}_{\text{UMI}}$; and (3) for the j -th synthetic UMI,

1. sample c_{ij} read start positions (relative to reference transcriptome) from the read coverage distribution $\hat{F}_x$;
2. generates c_{ij} synthetic reads based on the sampled read start positions, the read length, and the reference transcriptome.

Specifically, in step 2, assume the 5' end position of a synthetic read is x . For the 10x 3' protocol, the synthetic read spans the sequence range $x \rightarrow x + r$ on the sense strand (same as the reference transcriptome). In contrast, for the 10x 5' protocol, the read also spans $x \rightarrow x + r$, but on the reverse complement strand. For paired-end sequencing in 5' protocols, scIsoSim simulates read 1 from the sense strand (e.g., spanning $x \rightarrow x + r$) and read 2 from the reverse complement strand (e.g., spanning $x + l - r \rightarrow x + l$), reflecting the strand-specific nature of the 10x 5' assay. Using the 5' and 3' positions of each synthetic read, scIsoSim extracts the corresponding sequences from the reference transcriptome, generating a total of c_{ij} read sequences.

To generate the read 1, scIsoSim concatenates the sequence string of a randomly generated UMI and a cell barcode, which will then be paired with a synthetic read 2 generated above. By default, each read 1 is 26 bp long, consisting of a 16 bp cell barcode and a 10-bp UMI.

1. For each synthetic cell, scIsoSim generates a unique 16 bp barcode by randomly sampling from A, C, G, and T with replacement. This barcode is then assigned to all reads linked to that cell.
2. To generate UMIs for each isoform i in one synthetic cell, scIsoSim (1) extracts the UMI count u_i from the synthetic isoform-by-cell abundance matrix; (2) generates u_i unique 10-bp UMIs, with each UMI created by randomly sampling A, C, G, and T with replacement for 10 times; (3) assigns the u_i UMIs to the synthetic reads 2 generated above by assigning each synthetic UMI j , $j = 1, \dots, u_i$ to c_{ij} reads 2.

3.2.1.3 Specification of ground-truth isoform proportion

scIsoSim requires an isoform proportion file that specifies the relative abundance of each isoform within its gene at the cell type level. By default, for a gene with m isoforms annotated in the input gene annotation file, scIsoSim samples isoform proportions from a Dirichlet distribution $\text{Dir}(\boldsymbol{\alpha})$, where the concentration parameters $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_m)$ are set to $\alpha_i = 2$ for all $i = 1, \dots, m$. Users can adjust the concentration parameters of the Dirichlet distribution through a function option provided by the scIsoSim software.

Alternatively, scIsoSim allows users to provide a custom isoform proportion file, either arbitrarily designed or derived from isoform quantification tools applied to real data to capture biologically relevant isoform distributions.

3.2.2 Data analysis strategies

3.2.2.1 Comparison between synthetic and real scRNA-seq data

We applied scIsoSim to the mouse Smart-seq2 hematopoietic cell dataset [89] (with two cell types CMP and LSK), the human 10x 3' single-cell lung adenocarcinoma dataset [90], and the 10x 5' single-cell NSCLC tumor dataset [91]. Using the reference genome, BAM file, gene annotation file, and cell-type-specific isoform proportions as ground truths, scIsoSim generated synthetic scRNA-seq reads in both FASTQ and BAM formats. The cell-type-specific isoform proportions were estimated at the pseudo-bulk level using Kallisto [22], pooling cells of the same cell type into a single pseudo-bulk sample for analysis.

We evaluated the realism of scIsoSim's synthetic data by comparing it to real scRNA-seq data at both the read sequence and splicing event levels using the following metrics:

1. k-mer spectrum. The k-mer spectrum measures the occurrences of k-mers (i.e., substrings of length k) within the read sequences. We used the tool Jellyfish [65] with the default settings to obtain the 11-mer, 21-mer, 31-mer, and 41-mer spectra in the synthetic reads and real reads, respectively (Fig. 3.2a for Smart-seq2, Supplementary Fig. 3.8a for the 10x 3' protocol, and Supplementary Fig. 3.9a for the 10x 5' protocol).

2. Per-gene read coverage. Read coverage within each gene was analyzed to compare synthetic data against real data. This per-gene analysis utilized the `geneBody_coverage` function in RSeQC [88] (Fig. 3.2b, Supplementary Figs. 3.8–3.9b).
3. Splicing event visualization. Splicing events were visualized using ggsashimi [92]. Sashimi plots displayed isoform structures, read alignments (as read densities), and splice junction reads (arcs connecting exons). The arc width represented the number of reads spanning the junction (Fig. 3.2c, Supplementary Figs. 3.8–3.9c).

At the count level, scIsoSim employs the same modeling framework as scReadSim [93], which is based on the peer-reviewed simulators scDesign2 [10] and scDesign3 [87]. The fidelity of this count-level modeling has been demonstrated in the all three studies.

3.2.2.2 Benchmark of isoform quantification tools using synthetic Smart-seq2 data

We used scIsoSim with default parameters to generate synthetic scRNA-seq data based on the mouse Smart-seq2 hematopoietic cell dataset [89] (with cell type CMP). The simulated scRNA-seq reads served as input, while scIsoSim’s isoform-by-cell abundance matrix was treated as the ground truth for evaluating the performance of four isoform quantification tools: Kallisto [22], Salmon [23] (in both alignment-free and -based modes), RSEM [21] (with STAR), and Terminus [24]. Each tool was applied to the synthetic scRNA-seq reads, producing an isoform-by-cell abundance matrix. Supplementary Fig. 3.10 illustrates the benchmarking workflow of scIsoSim for isoform quantification tools using Smart-seq2 scRNA-seq data.

To ensure consistency, only isoforms present in the ground truth were retained, and isoforms with zero abundance across all cells in both the estimated and ground truth matrices were removed. Additionally, cells with zero abundance for all isoforms in both matrices were excluded. The performance of the quantification tools was then systematically evaluated across seven key aspects.

- First, we calculated six summary statistics to compare the estimated abundance matrices against the ground truth. At the isoform level, we computed the mean, variance, coefficient of variation, and zero proportion. At the cell level, we calculated the zero proportion and cell library size (Fig. 3.3a).
- Second, we evaluated the correlations between the ground truth and each estimated abundance matrix evaluated at both the isoform and cell levels. Pearson correlation and Kendall’s tau were used as the association measures (Fig. 3.3b).
- Third, we compared the estimated logarithmic isoform abundances to the ground truth using scatter plots (Fig. 3.3c). Isoform abundances were measured in Transcripts Per Million (TPM), as defined in section Section 3.2.2.5. Before plotting, the matrices were vectorized. We also calculated Pearson and Spearman correlations as global similarity measures between the estimated and ground-truth isoform abundance. Using ground-truth isoform abundance ≥ 1 TPM as the true condition, we calculated the false positive proportion (FPP) and false negative proportion (FNP), defined as the proportions of false positives and false negatives, respectively, out of the total entries. Specifically, false positives are isoforms with an estimated abundance ≥ 1 TPM but a ground-truth abundance < 1 TPM, while false negatives are isoforms with ground-truth abundance ≥ 1 TPM but an estimated abundance < 1 TPM.
- Fourth, we examined the effect of isoform complexity on quantification accuracy by calculating the mean relative difference between estimated and ground-truth abundances, stratified by the number of isoforms per gene. This analysis assessed how performance changes as gene isoform complexity increases (Fig. 3.3d).
- Fifth, to assess the performance of quantification tools in differential expression (DE) testing, we simulated scRNA-seq reads for different cell types using scIsoSim, based on mouse Smart-seq2 data from CMP and LSK cells. We compared the false discovery rate (FDR) versus the sensitivity of detecting differentially expressed isoforms in the simulated data. Differential expression in isoform expression across cell types was tested using a two-sided t-test of $\log_2(\text{TPM} + 1)$, with p-values adjusted using the

Benjamini–Hochberg method. Sensitivity was calculated for FDR thresholds at 0.01 intervals (Fig. 3.3e).

- Sixth, we evaluated how the performance of the quantification tools was affected by increasing cell numbers while keeping the total sequencing depth fixed. Performance was measured as the mean absolute log fold change between the estimated and ground truth abundances at the isoform level (Fig. 3.3f).
- Seventh, the running time of each tool was evaluated under varying cell numbers, with all tools executed using 12 threads (Fig. 3.3g).

3.2.2.3 Benchmark of isoform quantification tools using synthetic 10x data

We applied scIsoSim to the human 10x 3' single-cell lung adenocarcinoma dataset [90] to generate synthetic 10x 3' protocol scRNA-seq data. Using the simulated scRNA-seq reads as input, and the isoform-by-cell abundance matrix (based on UMI counts) produced by scIsoSim as the ground truth, we evaluated the single-cell isoform quantification tool, Scasa [35], and three bulk isoform quantification tools, Kallisto, Salmon (alignment-based) and Terminus. Scasa's output may include isoform paralogs—highly similar isoforms that cannot be distinguished based on supporting reads. These paralogs were excluded from the evaluation. To ensure consistency, isoforms with zero abundance across all cells in both the estimated and ground truth matrices were removed. Additionally, cells with zero abundance for all isoforms in both matrices were excluded.

At the single-cell level, we evaluated Scasa's performance by comparing its estimated isoform-by-cell abundance matrix to the ground truth in three aspects. First, we calculated Spearman correlation and Kendall's tau as association measures to assess the correlations between the ground-truth isoform-by-cell abundance matrix and the Scasa-estimated matrix at both the isoform and cell levels (Fig. 3.4a). Second, we compared the two matrices using four summary statistics: mean, variance, and zero proportion at the isoform level, as well as zero proportion at the cell level (Fig. 3.4b). Third, we compared the two matrices' entries through scatter plot, where the x-axis represents the ground truth and the y-axis represents

Scasa’s estimates. Pearson and Spearman correlations are reported at the top of Fig. 3.4c.

At the pseudo-bulk level, we compared Scasa’s isoform quantification performance against bulk-level tools, including Kallisto, Salmon, and Terminus. The ground-truth isoform-by-cell abundance matrix was aggregated into pseudo-bulk-level isoform abundances as the ground truth for comparison. For Scasa, isoform abundances were estimated by summing the estimates across cells, while for bulk-level tools, single-cell data were pooled into a pseudo-bulk sample and processed directly. The evaluation was twofold: First, we assessed the accuracy of each method by calculating symmetric absolute proportion error (APE) scores for individual isoforms (Fig. 3.4d), where smaller APE values indicate more accurate isoform quantification. Second, we compared the estimated logarithmic isoform abundances to the ground truth using scatter plots (Fig. 3.4e) with Pearson and Spearman correlations indicated in the upper left corner. We also reported the FPP, defined as the proportion of isoforms quantified with nonzero abundance despite having no supporting reads in the synthetic data; and FNP, defined as the proportion of isoforms with supporting reads in the synthetic data but quantified as zero.

3.2.2.4 Benchmark of AS quantification tools

To generate synthetic scRNA-seq data with specified ground-truth exon-skipping AS events, we provided scIsoSim with three inputs: (1) the mouse Smart-seq2 BAM file [89] (CMP cells only), (2) a processed gene annotation file containing only genes with exon-skipping events, and (3) an isoform proportion file corresponding to the processed gene annotation. Using these inputs, scIsoSim generated synthetic scRNA-seq reads in FASTQ format. The FASTQ files were then split by single cells and aligned into BAM files, ensuring that each cell had a separate BAM file for downstream analysis.

The processed gene annotation file was downloaded from [36], where BRIE-kit [94] extracted genes with isoforms exclusively involving exon-skipping events. Each gene was limited to two isoforms: exon-inclusion and exon-exclusion. For the isoform proportion file, scIsoSim assigned ground-truth isoform proportions by randomly sampling from a gene-wise Dirich-

let distribution, $\text{Dir}(\alpha_1, \alpha_2)$, where concentration parameters α_i were drawn from a uniform distribution, $U(0, 1)$.

To benchmark AS quantification tools—outrigger [37] and brie [36]—we applied both tools with their default settings to synthetic scRNA-seq FASTQ files. Specifically, brie was initially deployed in its default configuration (`--interceptMode=None`), referred to as mode 0 in this study. Additionally, we tested brie in two alternative configurations: mode 1 (`--interceptMode=cell`) and mode 2 (`--interceptMode=gene`). These modes modify brie’s implementation by incorporating either a cell-wise (mode 1) or gene-wise (mode 2) offset to the prior distribution’s mean parameter, resulting in a gene-shared or cell-shared prior.

Both tools output cell-wise percent spliced-in (PSI) scores for each gene, representing the estimated proportion of exon-inclusion isoforms. These PSI score matrices (gene-by-cell) were then compared against the ground-truth exon-inclusion isoform proportion matrix generated by scIsoSim, which is also gene-by-cell (Supplementary Fig. 3.11). Specifically, for each gene in a cell, the ground-truth exon-inclusion proportion was calculated by dividing the abundance of the exon-inclusion isoform by the total abundance of all isoforms for that gene. In our comparison, we kept the genes that were identified with at least one cell that has a non-zero PSI by all tools/modes. Our evaluation consisted of four aspects:

- First, we calculated correlations between the ground-truth exon-inclusion isoform proportions and the PSI score matrices at both the gene and cell levels. Pearson correlation and Kendall’s tau were used as association measures (Fig. 3.5a).
- Second, for each gene, we computed summary statistics, including the mean, variance, coefficient of variation, and zero proportion (Fig. 3.5b).
- Third, we conducted two-sample t-tests for each gene to determine whether the mean PSI scores across all cells significantly differed from the ground-truth exon-inclusion proportion. We used volcano plots (Fig. 3.5c) to visualize the p-values with the x-axis representing the mean differences between the PSI scores and exon-inclusion isoform proportions for each gene, and the y-axis showing the $-\log_{10}(\text{p-value})$. The red dashed

line represents the 0.05 significance level. In addition, we calculated the proportion of genes rejecting the null hypothesis in the t-test for each method. A smaller rejection rate indicates a closer resemblance between the mean of the estimated PSI scores and the ground-truth exon-inclusion isoform proportions.

- Fourth, scatter plots (Fig. 3.5d) were generated to compare PSI scores with ground-truth exon-inclusion isoform proportions. Before plotting, the matrices were vectorized. Pearson and Spearman correlations were also computed as global similarity measures between the vectorized PSI scores and ground-truth proportions. Additionally, using non-zero ground-truth exon-inclusion isoform proportions as the true condition, we calculated the FPP—the proportion of genes with an estimated exon-inclusion isoform proportion greater than zero but a ground-truth value of zero—and the FNP—the proportion of genes with a ground-truth exon-inclusion proportion greater than zero but an estimated value of zero.

3.2.2.5 Benchmark metrics

Transcripts Per Million (TPM). The TPM value for isoform transcript i is defined as

$$\text{TPM}_i = \frac{q_i/l_i}{\sum_j (q_j/l_j)} * 10^6,$$

where q_i denotes the number of reads mapped to transcript i and l_i denotes the length of transcript i .

Mean relative difference. The mean relative difference score represents the average log-transformed relative difference between estimated and ground-truth TPMs for transcripts from genes with i isoforms, which is defined as:

$$\text{Mean relative difference} = \frac{1}{N} \sum_{j=1}^N \left| \log_2 \left(\frac{|\text{TPM}_{est,j} - \text{TPM}_j|}{\text{TPM}_j + 1} + 1 \right) \right|$$

where N is the total number of transcripts from genes with i isoforms, $\text{TPM}_{est,j}$ is the estimated TPM value for transcript j and TPM_j is the ground truth TPM value for transcript j . A pseudocount of 1 is added to the denominator and the outer +1 inside the log ensures the argument of the logarithm is at least 1.

Absolute proportion error (APE). The APE score for transcript i is defined as:

$$\text{APE}_i = \left| \frac{t_i - e_i}{t_i + e_i + 1} \right|$$

where t_i is the ground truth total abundance of transcript i and e_i is the total transcript abundance estimated by each method. A pseudocount of 1 is added to the denominator.

3.3 Results

3.3.1 scIsoSim simulates realistic scRNA-seq reads with full-length isoform ground truths

To validate scIsoSim’s ability to generate realistic scRNA-seq reads, we trained the simulator using datasets profiled with various NGS technologies, including a mouse Smart-seq2 haematopoietic cell dataset [89] (including two cell types: common myeloid progenitor (CMP) and LinSca-1c-Kit (LSK)), a human 10x 3’ single-cell lung adenocarcinoma dataset [90], and a human 10x 5’ single-cell NSCLC tumor dataset [91]. We compared scIsoSim’s synthetic data with real experimental datasets in three aspects.

First, scIsoSim’s synthetic reads preserve the k-mer spectra, describing the frequency distribution of k-mers in sequencing reads (Fig. 3.2a for Smart-seq2, Supplementary Fig. 3.8a for the 10x 3’ protocol, and Supplementary Fig. 3.9a for the 10x 5’ protocol). Second, scIsoSim maintains the distribution of read coverage across the gene body (Fig. 3.2b for Smart-seq2, Supplementary Fig. 3.8b for the 10x 3’ protocol, and Supplementary Fig. 3.9b for the 10x 5’ protocol). As expected, Smart-seq2 datasets exhibit a more uniform read

coverage compared to 10x 3' and 10x 5', which are biased toward transcript ends due to their protocol-specific designs. Third, scIsoSim preserves the cell-type-specific read coverage over the splice junction sites in the real data (Fig. 3.2c, Supplementary Figs. 3.8–3.9c).

Additionally, at the count level, scIsoSim incorporates the realistic sequencing read count simulators scDesign2 [10] and scDesign3 [87] to generate gene abundance matrices (see Section 3.2.1). These simulators have been validated to produce more realistic scRNA-seq gene abundance than other methods, effectively preserving features such as gene-gene correlations observed in real data.

3.3.2 Application 1: benchmarking bulk-level isoform quantification tools using synthetic Smart-seq2 scRNA-seq data

To evaluate whether bulk-level isoform quantification tools can be applied to scRNA-seq data, we trained scIsoSim using the mouse Smart-seq2 dataset [89] and simulated synthetic single-cell RNA-seq data with specified isoform structures and abundance (see Section 3.2.2.2). The specified isoform structure and abundance are treated as ground truth to benchmark four state-of-art tools: Kallisto [22], Salmon [23] (two modes: alignment-free and -based), RSEM [21] (with STAR), and Terminus [24]. Since Smart-seq2 does not incorporate unique molecular identifiers (UMIs) to distinguish reads from different transcript molecules, transcript abundance is reported in TPM (Transcripts Per Million, defined in Section 3.2.2.5), a length-normalized measure of sequencing read counts.

Our benchmark results show that for the Smart-seq2 protocol, bulk-level quantification tools applied to single cells can achieve accurate isoform abundance (i.e., isoform-specific transcript abundance) estimations, demonstrated with strong concordance to ground truth. We evaluated performance across four key aspects.

First, we assessed how well the estimated isoform-by-cell isoform abundance matrix from the tools align with the ground truth. We calculated six summary statistics: mean, variance, coefficient of variation, and zero proportion at the isoform level, as well as zero proportion and cell library size at the cell level (Fig. 3.3a). The results show that all four tools (five

implementations, as Salmon includes two modes) produce distributions of summary statistics that closely resemble the ground truth. Additionally, Pearson and Kendall’s tau correlations between the estimated and ground-truth isoform abundance matrices are strong at both the isoform and cell levels (Fig. 3.3b), confirming high concordance. The estimated isoform abundances also show strong correlations with the ground truth, with low false positive and false negative rates (Fig. 3.3c).

Second, to evaluate the impact of task complexity, we stratified genes by the number of isoforms per gene and compared the mean relative difference. As shown in Fig. 3.3d, all tools exhibit stable performance when genes have fewer than 40 isoforms, but the estimation error increases sharply for genes with more than 40 isoforms. Notably, RSEM shows the highest difference across all numbers of isoforms per gene, while other tools show similar performance.

Third, we evaluated the performance of the quantification tools in differential expression (DE) transcript testing using simulated data generated by scIsoSim, based on the same Smart-seq2 dataset but with two cell types: CMP cells and LSK cells. The relationship between false discovery rate (FDR) and sensitivity in detecting differentially expressed transcripts is shown in Fig. 3.3e. All tools demonstrated comparable performance, maintaining consistent sensitivity in identifying differentially expressed transcripts, except for Terminus, which exhibited slightly lower sensitivity, particularly at lower FDR thresholds.

Fourth, we evaluated the tools’ performance as cell numbers increased while maintaining a fixed total sequencing depth. Kallisto and Salmon (alignment-free) achieved the highest accuracy with stable performance, whereas RSEM with STAR and Terminus exhibited lower accuracy and greater variability (Fig. 3.3f). Additionally, Salmon (alignment-free), Terminus, and Kallisto executed computations faster than RSEM with STAR and Salmon (alignment-based), particularly as cell numbers increased (Fig. 3.3g). This efficiency is due to their use of pseudo-alignment, which omits full read-to-reference genome alignment, reducing computational overhead.

3.3.3 Application 2: evaluating Scasa for isoform quantification using synthetic 10x 3' scRNA-seq data

Isoform quantification with 10x scRNA-seq data presents unique challenges compared to Smart-seq2 due to the 3' or 5' end biases inherent in 10x protocols. These biases produce similar read profiles across isoforms, making it difficult to differentiate sequencing reads from distinct isoforms. Additionally, the large scale of 10x datasets, often encompassing thousands to tens of thousands of cells, makes applying bulk-level tools to individual cells computationally impractical.

Therefore, in this application, we evaluated Scasa, the only scRNA-seq-specific isoform quantification tool designed for the 10x 3' protocol, to assess its reliability. We used scIsoSim to generate synthetic 10x 3' scRNA-seq reads based on a human 10x 3' single-cell lung adenocarcinoma dataset [90], with specified isoform structures and corresponding transcript abundances serving as ground truth (see Section 3.2.2.3). Our study shows that Scasa's accuracy in isoform abundance estimation is limited, likely due to the sparsity of single-cell data.

At the single-cell level, we evaluated Scasa's estimated isoform-by-cell abundance matrix against the ground truth. The results indicate moderate agreement, as reflected by a wide range of Spearman and Kendall's tau correlations at the isoform level (correlation across cells for each isoform) and mid-level correlations at the cell level (correlation across isoforms within each cell) (Fig. 3.4a). While summary statistics—three at the isoform level (mean, variance, and zero proportion) and one at the cell level (zero proportion)—show resemblance to the ground truth (Fig. 3.4b), a more detailed examination using scatter plots (Fig. 3.4c) reveals restricted accuracy, supported by a Pearson correlation of 0.69 and a Spearman correlation of 0.62.

To investigate whether the sparsity of single-cell 10x data contributed to this result, we implemented bulk-level tools (Kallisto, Salmon, and Terminus) by treating single cells as a pseudo-bulk sample. For comparison, Scasa was run at the single-cell level, and its total isoform abundances were summed across cells to match the pseudo-bulk scale. At the pseudo-

bulk level, accuracy was assessed using the absolute proportion of error (APE) between the estimated and ground-truth isoform abundances, with lower APE scores indicating higher accuracy (defined in Section 3.2.2.5). Among the tools, Kallisto achieved the lowest APE scores, indicating the highest accuracy (Fig. 3.4d). Additionally, scatter plots comparing estimated and ground-truth isoform abundances for all isoforms show that Kallisto provides the most accurate estimates, while Scasa performs the worst, as evidenced by the lowest Pearson and Spearman correlations among the tools (Fig. 3.4e). These results indicate that Scasa’s limited accuracy is due to single-cell data sparsity, as pooling cells into a pseudo-bulk sample enhances signal strength and improves isoform quantification with 10x 3’ data. This highlights the need to address sparsity when studying isoforms using 10x 3’ data.

3.3.4 Application 3: benchmarking single-cell AS quantification tools

To benchmark AS analysis tools, scIsoSim was used to simulate scRNA-seq reads with ground-truth AS events and their corresponding transcript abundances. We generated synthetic scRNA-seq data by specifying ground-truth AS events and transcript abundances (see Section 3.2.2.4) to evaluate two widely used single-cell AS analysis tools: outrigger [37] and brie [36]. For brie, we implemented three modes, each based on different prior distribution assumptions.

Both tools processed the synthetic scRNA-seq reads and output cell-wise percent spliced-in (PSI) scores for each gene, representing the estimated proportion of exon-inclusion isoforms. These PSI score matrices (gene-by-cell) were then compared against the ground-truth exon-inclusion isoform proportion matrix specified through scIsoSim. In our implementation, we considered outrigger and brie in three modes: mode 0 (default), mode 1 (incorporating a cell-wise offset to adjust for differences between individual cells), and mode 2 (incorporating a gene-wise offset to account for variations between genes).

Our benchmarking results show that brie outperformed outrigger in overall accuracy and robustness when estimating exon-inclusion isoform proportions using scIsoSim synthetic data. However, brie mode 2 failed to capture cellular heterogeneity, as its cell-shared prior

distribution led to the lowest variance across cells. In contrast, outrigger’s lower accuracy was primarily due to an excess of zero values in its outputs, although its non-zero estimates were accurate.

First, the PSI score matrices output by brie agree better with the ground-truth exon-inclusion isoform proportion matrix in terms of (1) Pearson correlations and Kendall’s tau correlations between the two matrices, both gene-wise (the correlation across cells per gene) and cell-wise (the correlation across genes per cell) (Fig. 3.5a); and (2) the distributions of four gene-level summary statistics, including mean, variance, coefficient of variance, and zero proportion (Fig. 3.5b). Notably, brie mode 2 exhibits zero gene-wise variance (variance across cells) for all genes, meaning that it fails to capture the cellular heterogeneity when estimating the exon-inclusion isoform proportions. In contrast, outrigger’s performance is likely hindered by an excessive number of zeros in its output, as highlighted in the gene zero proportion panel of Fig. 3.5b.

Second, volcano plots indicate that brie mode 1 and 2 produce PSI score matrices that more closely resemble the ground truth, as shown by a lower proportion of genes rejecting the null hypothesis in the t-test, which compares the mean differences between ground-truth proportions and estimated PSI scores across all cells (Fig. 3.5c).

Third, scatter plots reveal that brie mode 2 produces a PSI score matrix most similar to the ground-truth exon-inclusion isoform proportion matrix (Fig. 3.5d), with a Spearman correlation of 0.86 and a Pearson correlation of 0.88 (calculated after vectorizing the matrices). In contrast, outrigger shows the lowest global similarity to the ground truth, likely due to an excessive proportion of zeros in its output, as indicated by a false negative proportion (FNP) of 0.74. However, for non-zero entries, outrigger’s PSI score matrix aligns closely with the diagonal line, reflecting accurate estimates of exon-inclusion isoform proportions.

3.4 Discussion

We present scIsoSim, a single-cell RNA-seq (scRNA-seq) simulator designed to generate realistic synthetic reads that mimic experimental data while uniquely providing ground-

truth isoform structures and their corresponding abundances—functionality not available in existing single-cell sequencing read simulators. These capabilities make scIsoSim a versatile benchmarking tool for evaluating bioinformatics methods that process raw scRNA-seq reads, including isoform quantification and alternative splicing (AS) analysis.

There are several key areas where scIsoSim can be extended or applied to further advance scRNA-seq research.

First, scIsoSim could be extended to simulate long-read single-cell RNA sequencing. While scIsoSim enables realistic simulation of NGS scRNA-seq reads with ground-truth AS mechanisms, NGS technologies have inherent limitations in isoform and AS studies due to RNA fragmentation during library preparation, preventing the capture of full-length transcripts. In contrast, third-generation sequencing enables direct profiling of full-length transcripts, making it more suitable for isoform analysis [95, 96]. As a future direction, scIsoSim could be extended to simulate single-cell long-read sequencing reads, providing a tool to benchmark computational methods for long-read scRNA-seq and to study isoform diversity at a cellular level.

Second, scIsoSim can be used to systematically assess the impact of sequencing protocols on isoform and AS analysis. Single-cell RNA-seq allows for the study of splicing regulation at the single-cell level, but differences in AS detection across sequencing protocols remain underexplored. Tian et al. [97] observed a striking difference between 10x 5' and 10x 3' protocols, where SpliZ detected 4.3 times more AS genes per cell in the AIDA cohort (10x 5') than in the OneK1K dataset (10x 3'). These differences likely stem from protocol-specific factors such as exon painting rather than sequencing depth. Although our study did not directly investigate these biases, scIsoSim provides a framework to systematically evaluate the impact of sequencing protocols on AS detection. By generating synthetic reads with identical isoform abundances for both 10x 3' and 10x 5' protocols, researchers could assess differences in AS detection using tools like SpliZ. Additionally, simulating Smart-seq2 reads as a positive control—which provides full-length transcript coverage—would allow direct comparison against ground truths to evaluate whether SpliZ accurately detects cell-type-specific AS events.

Third, scIsoSim can be extended to model dropout effects and evaluate robustness in isoform quantification. Dropout events, where lowly expressed isoforms fail to be detected, introduce biases in isoform quantification and AS analysis. By incorporating transcript-specific expression variability and technical noise, scIsoSim could simulate controlled dropout effects, allowing researchers to assess the extent to which dropout impacts the detection of cell-type-specific isoform expression. Additionally, modeling different levels of sequencing noise could facilitate the development of noise-correction strategies, helping improve the accuracy of isoform quantification methods under varying sequencing depths and data quality conditions.

Fourth, scIsoSim can be expanded to simulate non-coding RNAs (ncRNAs) and regulatory elements. Currently, scIsoSim focuses on protein-coding isoforms, but extending its capabilities to model ncRNAs, such as long non-coding RNAs (lncRNAs), circular RNAs (circRNAs), and microRNAs (miRNAs), would enable benchmarking computational tools designed for ncRNA detection, quantification, and functional annotation. Since ncRNAs play critical roles in gene regulation and AS, their integration into scIsoSim would allow the study of interactions between coding and non-coding transcripts in single cells. Furthermore, incorporating transcriptional regulatory elements, such as enhancer RNAs and promoter-associated RNAs, could provide insights into the coordination between transcriptional regulation and isoform expression dynamics across different cell types and conditions.

Overall, we envision scIsoSim not just as a simulator, but as a flexible framework that empowers the computational biology community by enabling rigorous benchmarking, supporting method development, and promoting reproducibility in single-cell transcriptomics research.

3.5 Code and data availability

3.5.1 Software and code

The scIsoSim Python package is available at <https://github.com/JSB-UCLA/scIsoSim>.

In addition, the following software packages were used for the analyses in this work.

Isoform quantification tools

- **Kallisto**: Version 0.46.2 (<https://github.com/pachterlab/kallisto>).
- **Salmon**: Version 1.10.1 (<https://github.com/COMBINE-lab/salmon>).
- **STAR**: Version 2.7.10b (<https://github.com/alexdobin/STAR>).
- **RSEM**: Version 1.3.1 (<https://github.com/deweylab/RSEM>).
- **Terminus**: Version 0.1.0 (<https://github.com/COMBINE-lab/terminus>).
- **Scasa**: Version 1.0.1 (<https://github.com/eudoraleer/scasa>).

Alternative splicing analysis tools

- **Outrigger**: Version 1.1.1 (<https://yeolab.github.io/outrigger/>).
- **Brie**: Version 2.2.2 (<https://brie.readthedocs.io/en/latest/index.html>).

Other tools

- **scReadSim**: Version 1.4.1 (<https://github.com/JSB-UCLA/scReadSim>).
- **Jellyfish**: Version 2.3.0 (<https://github.com/gmarcais/Jellyfish>).
- **Intervene**: Python package Intervene Version 0.41.0 (<https://intervene.readthedocs.io/en/latest/install.html>).
- **RSeQC**: Version 5.0.2 (<https://rseqc.sourceforge.net>).
- **bamPEFragmentSize**: Version 3.3.2 (<https://deeptools.readthedocs.io/en/develop/content/tools/bamPEFragmentSize.html>).
- **ggsashimi**: Version 1.1.5 (<https://github.com/guigolab/ggsashimi>).

3.5.2 Datasets

- **Mouse Smart-seq2 haematopoietic cell dataset:** The Smart-seq2 dataset (GEO accession number GSE70245) measures the gene expression of haematopoietic cells in bone marrow [89]. The raw FASTQ files were downloaded from <https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE70245>. Data from two cell types, common myeloid progenitor and LinSca-1c-Kit, were used for analysis. The reference genome file (assembly version GRCm38.p6) was downloaded from GENCODE (https://www.gencodegenes.org/mouse/release_M23.html).
- **Human 10x 3' single-cell lung adenocarcinoma dataset:** The 10x Genomics Chromium 3' scRNA-seq dataset (GEO accession number GSE126906) measures the gene expression data from 5 human lung adenocarcinoma cell lines H2228, H1975, A549, H838 and HCC827 [90]. The raw FASTQ files were downloaded from <https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE126906>. The reference genome file (assembly version GRCh38.p13) was downloaded from GENCODE (https://www.gencodegenes.org/human/release_38.html).
- **Human 10x 5' single-cell NSCLC tumor dataset:** The 10x Genomics Chromium 5' dataset measures the gene expression data from a fresh surgical resection of a squamous non-small cell lung carcinoma (NSCLC) tumor [91]. The raw FASTQ and BAM files were downloaded from <https://www.10xgenomics.com/datasets/nsclc-tumor-5-gene-expression-1-standard-2-2-0>. The same reference genome (assembly version GRCh38.p13) used for the 10x 3' data was used in the analysis.

3.6 Acknowledgements

This chapter is based on my joint work with Weijian Wang, Zhiyin Liu, and my Ph.D. advisor Dr. Jingyi Jessica Li. We appreciate the comments and feedback from the members of the Junction of Statistics and Biology at UCLA (<http://jsb.ucla.edu>).

3.7 Figures

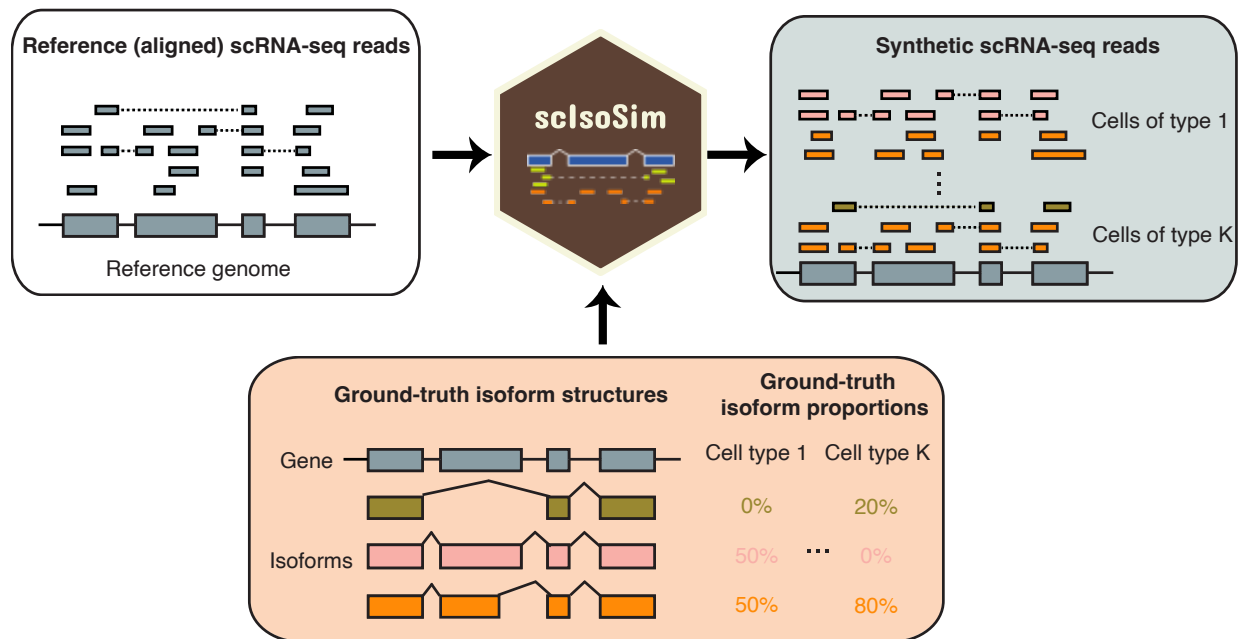


Figure 3.1: scIsoSim simulates scRNA-seq with full-length isoform ground truths.

Supporting multiple next-generation sequencing technologies, including Smart-seq2, 10x 3' and 5', scIsoSim uses aligned experimental scRNA-seq reads (reference data) as input and generates synthetic scRNA-seq data with predefined cell-type-specific isoform structures and proportions.

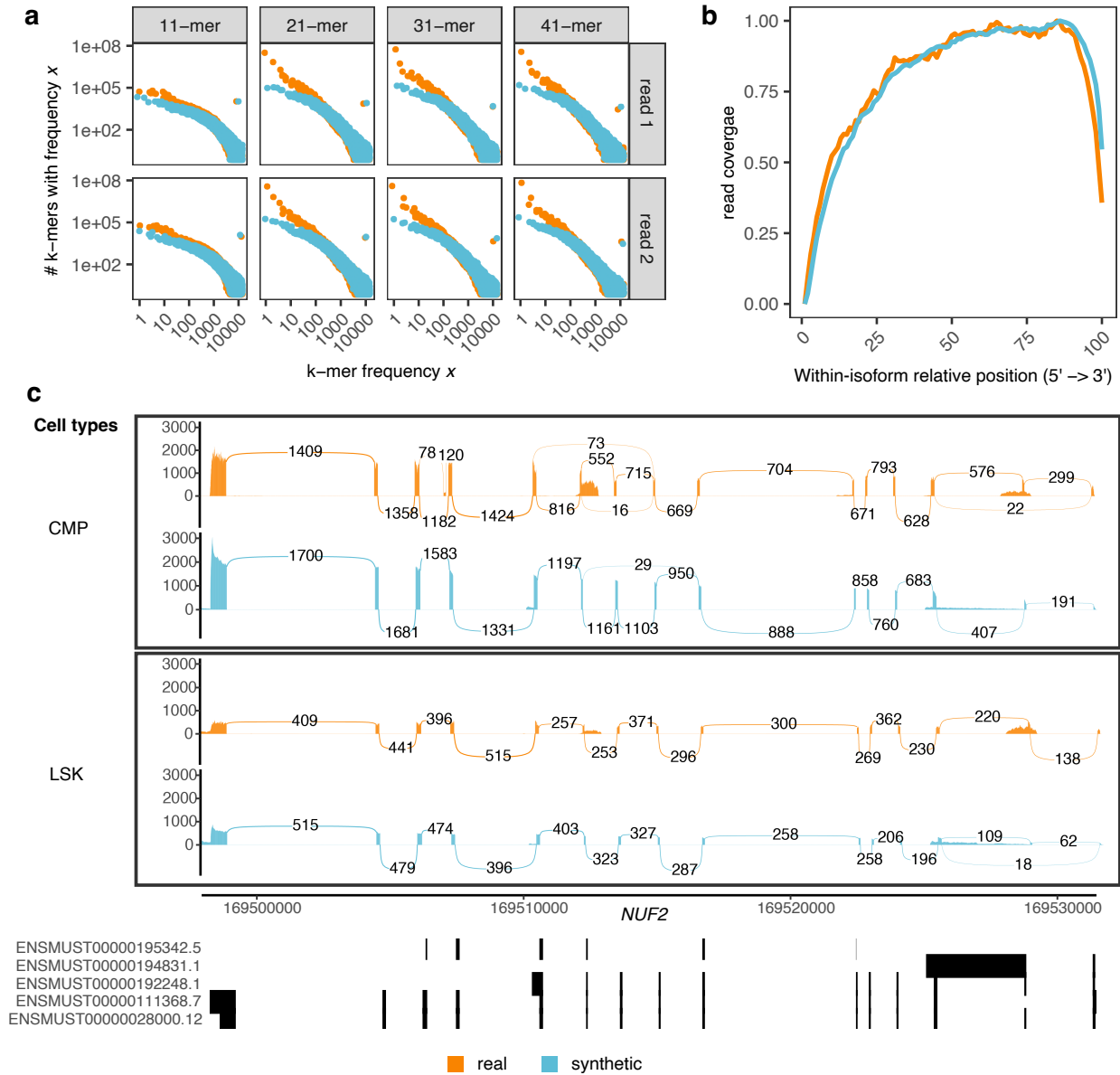


Figure 3.2: scIsoSim generates realistic isoform-level scRNA-seq data.

a, Comparison of k-mer spectra between scIsoSim's synthetic data and real data, using the mouse Smart-seq2 scRNA-seq dataset (CMP cells only). The x-axis shows the occurrence frequency of specific k-mers, while the y-axis represents the number of unique k-mers with each frequency, both displayed on a log10 scale. Synthetic read 1 and read 2 are presented in the first and second rows, respectively. **b**, Comparison of read coverage within isoform between scIsoSim's synthetic data and real data for the CMP cells. The x-axis represents the within-isoform relative position from 5' to 3', and the y-axis displays the min-max normalized read coverage. **c**, Preservation of cell-type-specific read coverage by scIsoSim. A Sashimi plot compares synthetic and real data across two cell types: CMP and Lin-Sca1+c-Kit+ (LSK). The x-axis indicates genomic regions, while the y-axis represents read coverage. Different isoform structures are color-coded at the bottom of the plot, with ground-truth isoform proportions shown at the top. Arcs connecting exons illustrate splice junction reads, with arc width proportional to the number of reads spanning the junction, and each arc labeled with the splice junction read coverage.

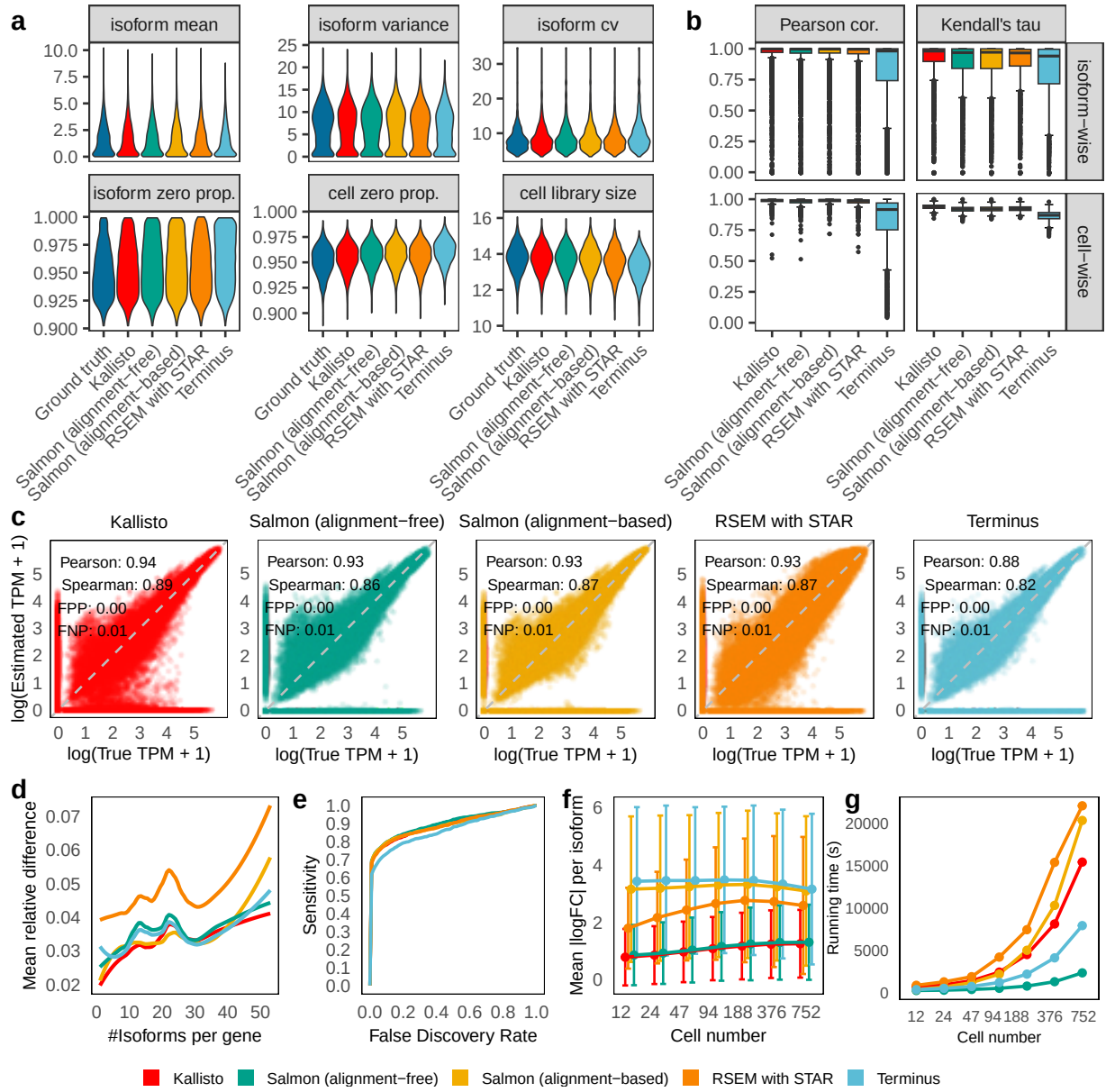


Figure 3.3: Benchmark isoform quantification tools using synthetic Smart-seq2 scRNA-seq data.

a, Summary statistics distributions of isoform-by-cell abundance matrices (ground truth and outputs from Kallisto, Salmon (two modes: alignment-free and -based), RSEM (STAR mode), and Terminus) at the isoform and cell levels. Isoform-level metrics include mean, variance, coefficient of variation (cv), and zero proportion. Cell-level metrics include zero proportion and library size. **b**, Correlations between the ground-truth and estimated abundance matrices from isoform quantification tools at both the isoform and cell levels. Pearson correlation and Kendall's tau were used as the association measures. **c**, Comparison of ground-truth isoform abundance against estimated abundance (measured in TPM) across all quantification tools. Pearson correlation, Spearman correlation, the false positive proportion (FPP, proportion of isoforms with non-zero abundance estimations but zero ground-truth abundance), and the false negative proportion (FNP, proportion of isoforms with zero abundance estimations but non-zero ground-truth abundance) are displayed in the top left corner. **d**, Relationship between the number of isoforms per gene and the mean relative difference between estimated and ground-truth isoform abundances. **e**, The false discovery rate (FDR) versus sensitivity of detecting differentially expressed transcripts by all quantification tools on scIsosim simulated data from mouse Smart-seq2 for CMP cells and LSK cells. **f**, Performance of quantification tools under varying cell numbers, measured as the mean absolute log fold change between the estimated and ground truth abundances at the isoform level. The bolded cell number (94) represents the number of real cells in the dataset used to train scIsosim. **g** Running time of quantification tools under varying cell numbers.

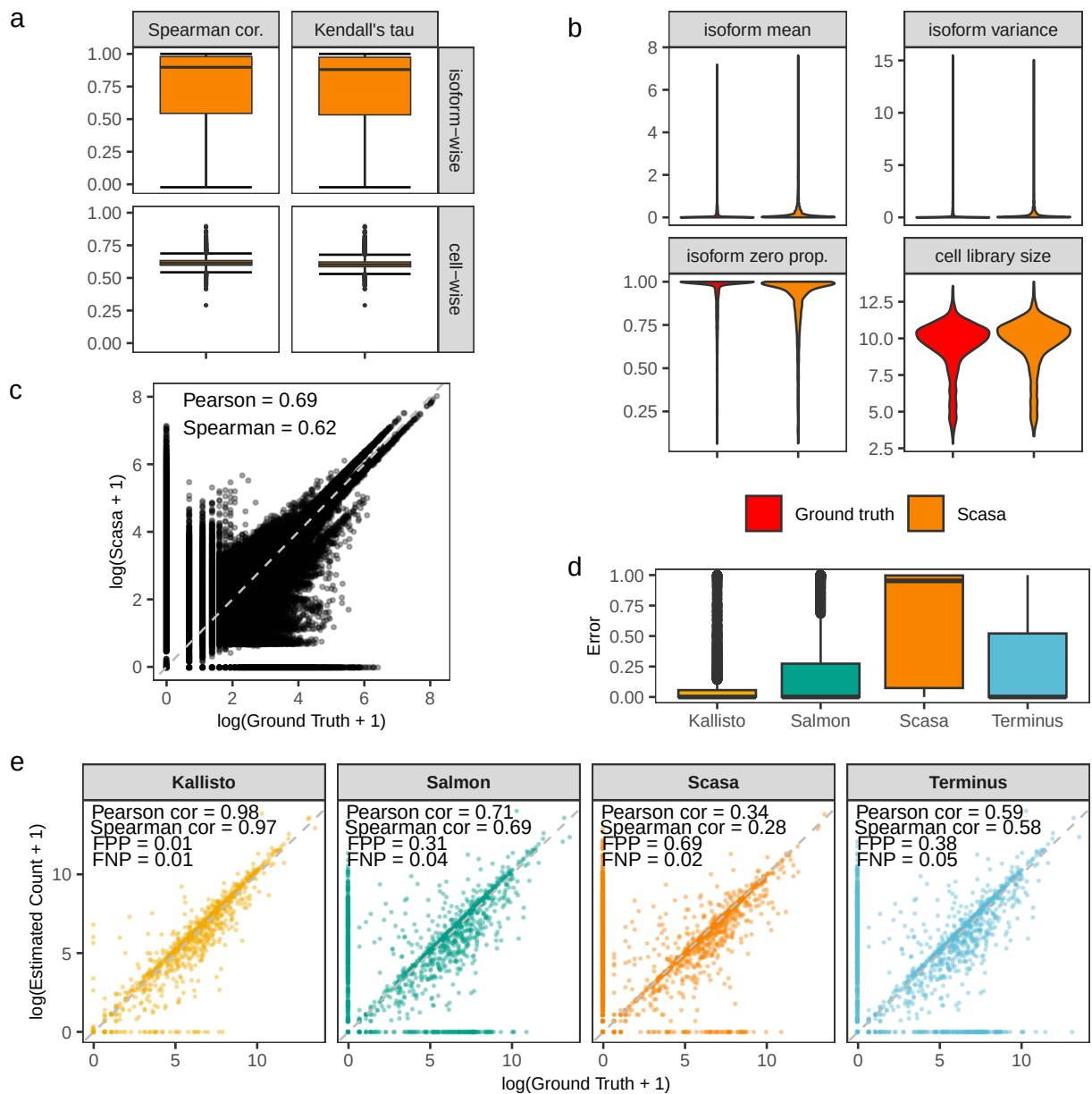


Figure 3.4: Benchmark isoform quantification tools using synthetic 10x 3' scRNA-seq data.

a, Correlations between the ground-truth and Scasa-estimated isoform-by-cell abundance matrices at both the isoform and cell levels. Spearman correlation and Kendall's tau were used as the association measures. **b**, Summary statistics distributions of the isoform-by-cell abundance matrices at the isoform and cell levels. Isoform-level metrics include mean, variance, coefficient of variation (cv), and zero proportion. Cell-level metrics include zero proportion and library size. **c**, Scatter plot of the ground-truth isoform-by-cell abundance matrix (x-axis) against Scasa-estimated abundance matrix (y-axis) with Spearman and Pearson correlation displayed at the top. **d**, Boxplots of errors measured in absolute proportion error (APE) scores of Kallisto, Salmon, Scasa, and Terminus. Smaller APE scores indicate higher accuracy. **e**, Pseudo-bulk-level isoform quantification comparing bulk-level tools and Scasa. All tools' estimation of isoform abundances are plotted against the ground truth values. The Spearman and Pearson correlation, FPP (proportions of isoforms with non-zero abundance estimations but zero ground-truth abundance) and FNP (proportions of isoforms with zero abundance estimations but non-zero ground-truth abundance) are shown in the upper left corner.

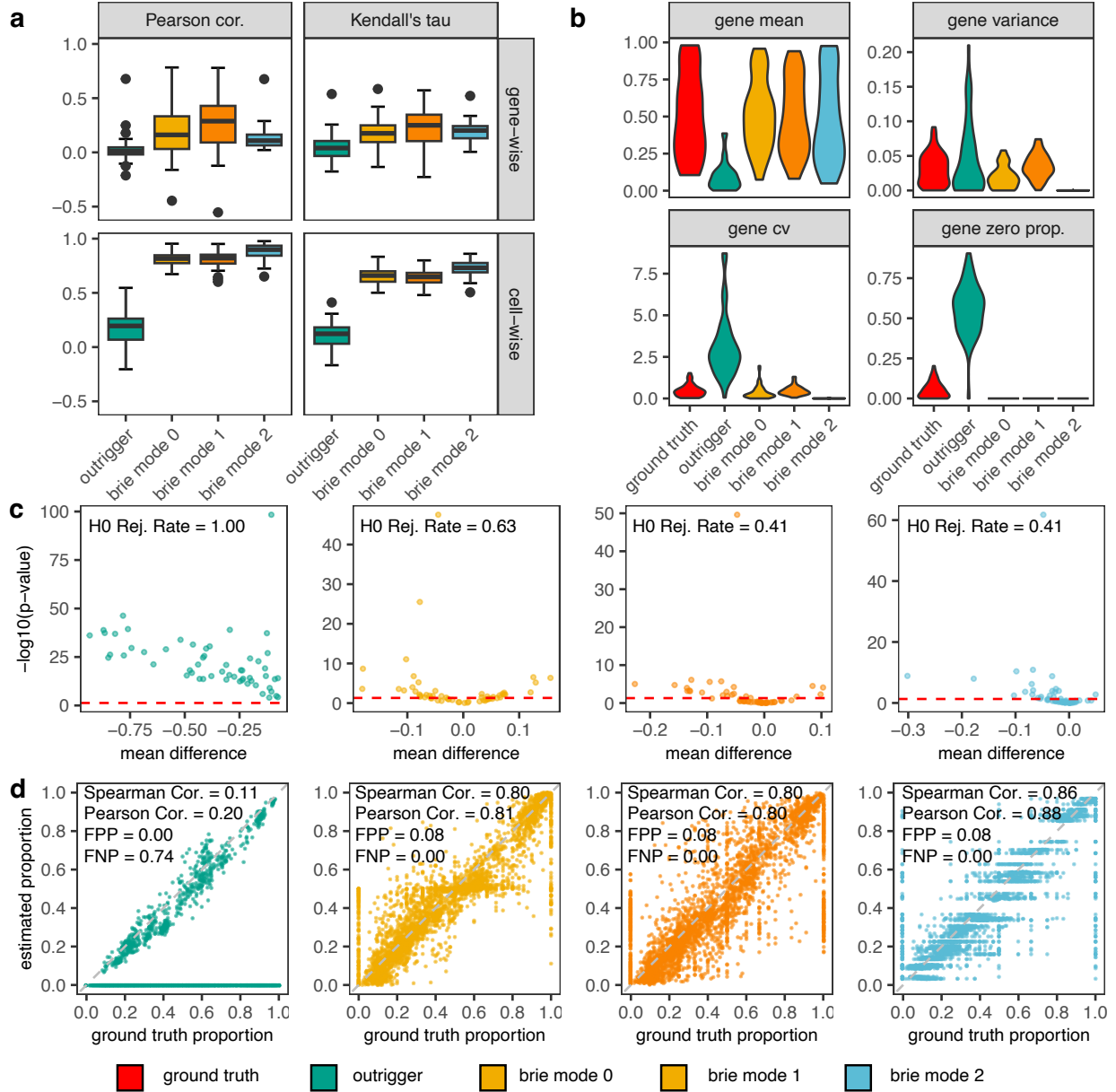


Figure 3.5: Benchmarking AS quantification tools using synthetic Smart-seq2 data.

a, Correlations between the ground-truth exon-inclusion isoform proportion matrix and the percent spliced-in (PSI) score matrices output by benchmarking tools at both the gene and cell levels. Pearson correlation and Kendall's tau were used as the association measures. **b**, Summary statistics distributions of exon-inclusion isoform proportion matrices (ground truth and the output from benchmarking tools) at the gene level. For each gene, summary statistics including mean, variance, coefficient of variation (cv), and zero proportion are reported. **c**, Volcano plots showing the two-sample t-test p-values for PSI scores estimated by the benchmarking tools compared to the ground-truth exon-inclusion isoform proportions at the gene level. The x-axis represents the mean differences between the PSI scores and exon-inclusion isoform proportions, and the y-axis shows $-\log_{10}(\text{p-value})$. The red dashed line represents the 0.05 significance level. The proportion of genes rejecting the null hypothesis in the t-test (H0 Rej. Rate) is displayed in the top left corner for each method. A smaller rejection rate indicates a closer resemblance between the mean of the estimated PSI scores and the ground-truth exon-inclusion isoform proportions. **d**, Scatter plot comparing the PSI scores output by benchmarking tools against the ground-truth proportions of the exon-inclusion isoforms in the synthetic data. Pearson correlation, Spearman correlation, the proportion of false positives (genes with non-zero estimations of exon-inclusion isoform proportion but zero ground-truth proportion, defined as FPP) and false negatives (genes with zero estimations of exon-inclusion isoform proportion but non-zero ground-truth proportion, defined as FNP) are displayed in the top left corner.

3.8 Supplementary materials

S3.8.1 Supplementary figures

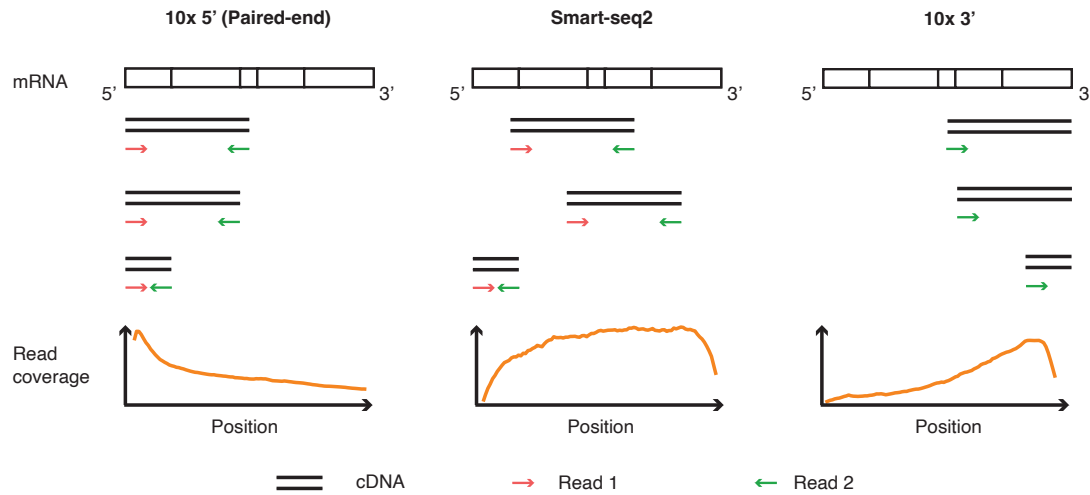


Figure 3.6: Illustration of sequencing read distributions for three scRNA-seq protocols: 10x 5' (paired-end), Smart-seq2, and 10x 3'.

The 10x 5' and 10x 3' protocols generate sequencing reads concentrated at the respective ends of mRNA molecules, while Smart-seq2 produces a relatively uniform coverage across the entire transcript.

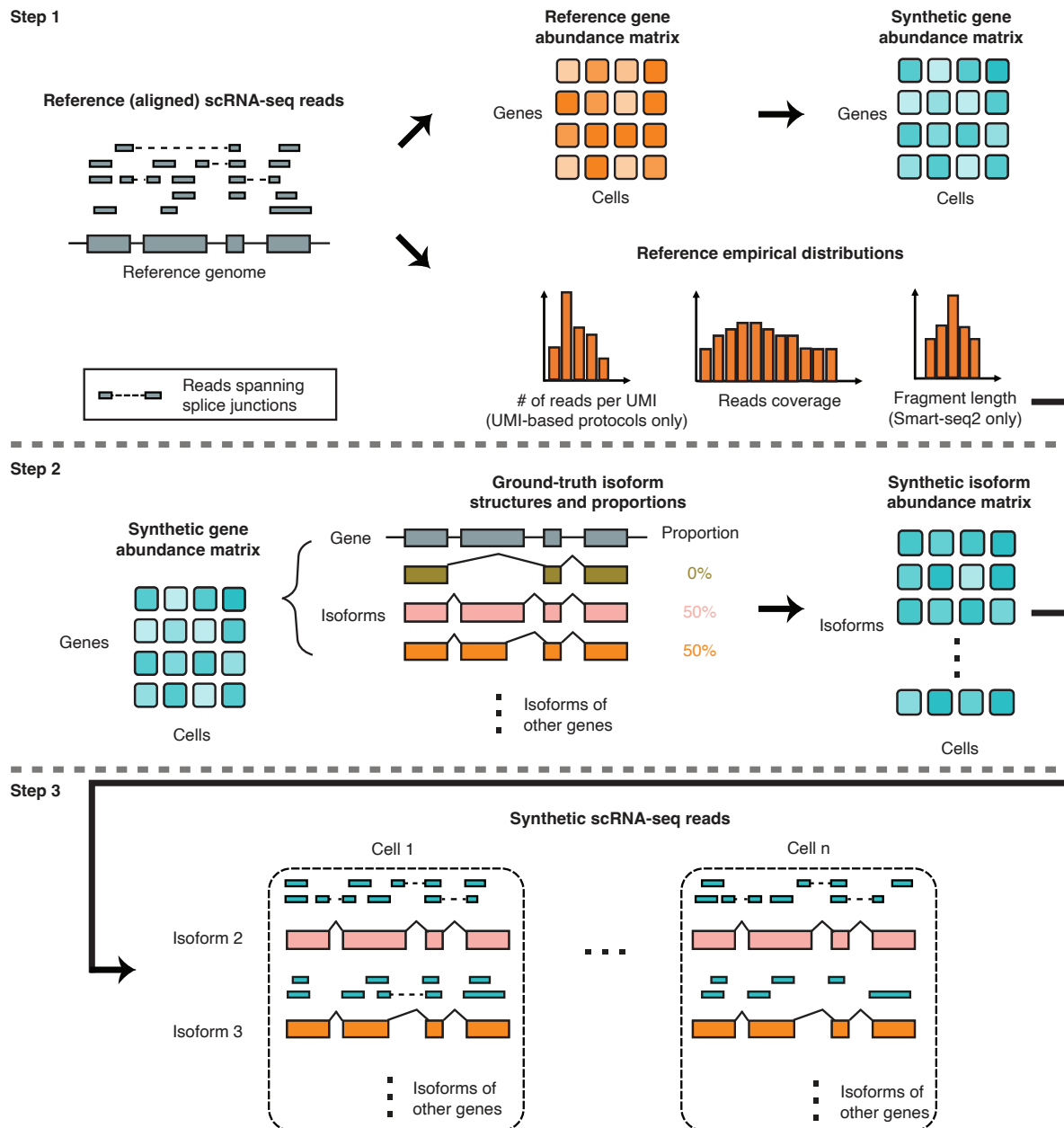


Figure 3.7: Three major steps of scIsoSim for simulating scRNA-seq reads.

Step 1 generates synthetic gene-level abundance matrices and extracts reference empirical distributions, including the number of reads per UMI (for UMI-based protocols), read coverage along the gene body, and fragment length distributions (for Smart-seq2), from the input reference scRNA-seq reads. Step 2 defines the isoform structures and specifies their abundances (as proportions summing to one for each gene), constructing a synthetic isoform-by-cell abundance matrix. Step 3 generates synthetic scRNA-seq reads by combining the empirical distributions from Step 1 and the isoform-by-cell abundances from Step 2.

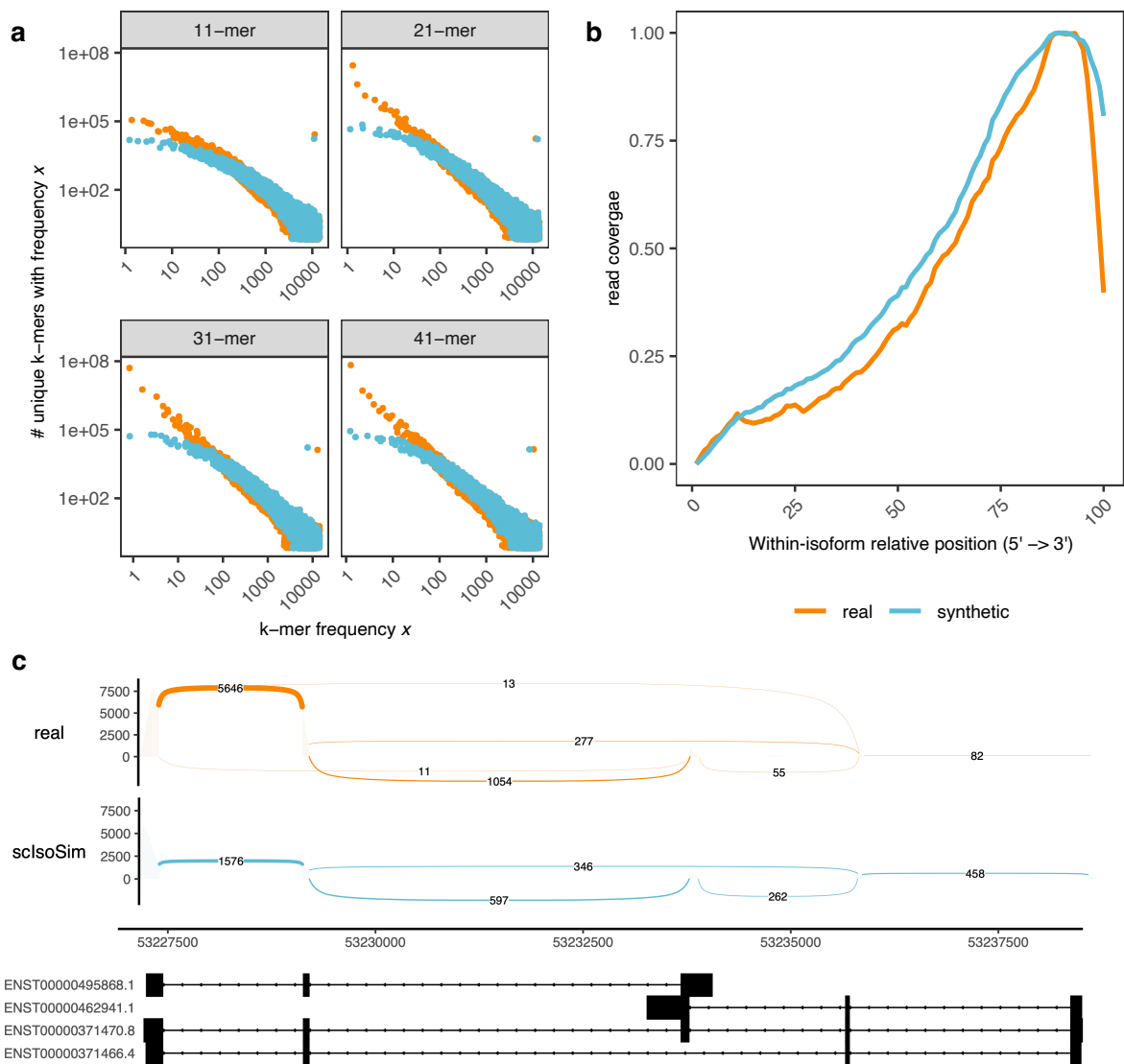


Figure 3.8: scIsoSim's synthetic data resemble real 10x 3' scRNA-seq dataset profiling human lung adenocarcinoma cells [1].

a, Comparison of the k-mer spectra between scReadSim's synthetic data and real data. The x-axis refers to the occurrence frequency of a specific k-mer, and the y-axis represents the number of unique k-mers with this frequency. Both the x-axis and y-axis are in the $\log_{10}$ scale. **b**, Comparison of read coverage within isoforms between scIsoSim's synthetic data and real data. The x-axis represents the within-isoform relative position from 5' to 3', and the y-axis displays the min-max normalized read coverage. **c**, Preservation of splice junction read coverage by scIsoSim through a sashimi plot. The x-axis indicates genomic regions, while the y-axis represents per-base read density. Different isoform structures are displayed at the bottom of the plot. Arcs connecting exons illustrate splice junction reads, with arc width proportional to the number of reads spanning the junction, and each arc labeled with the splice junction read coverage.

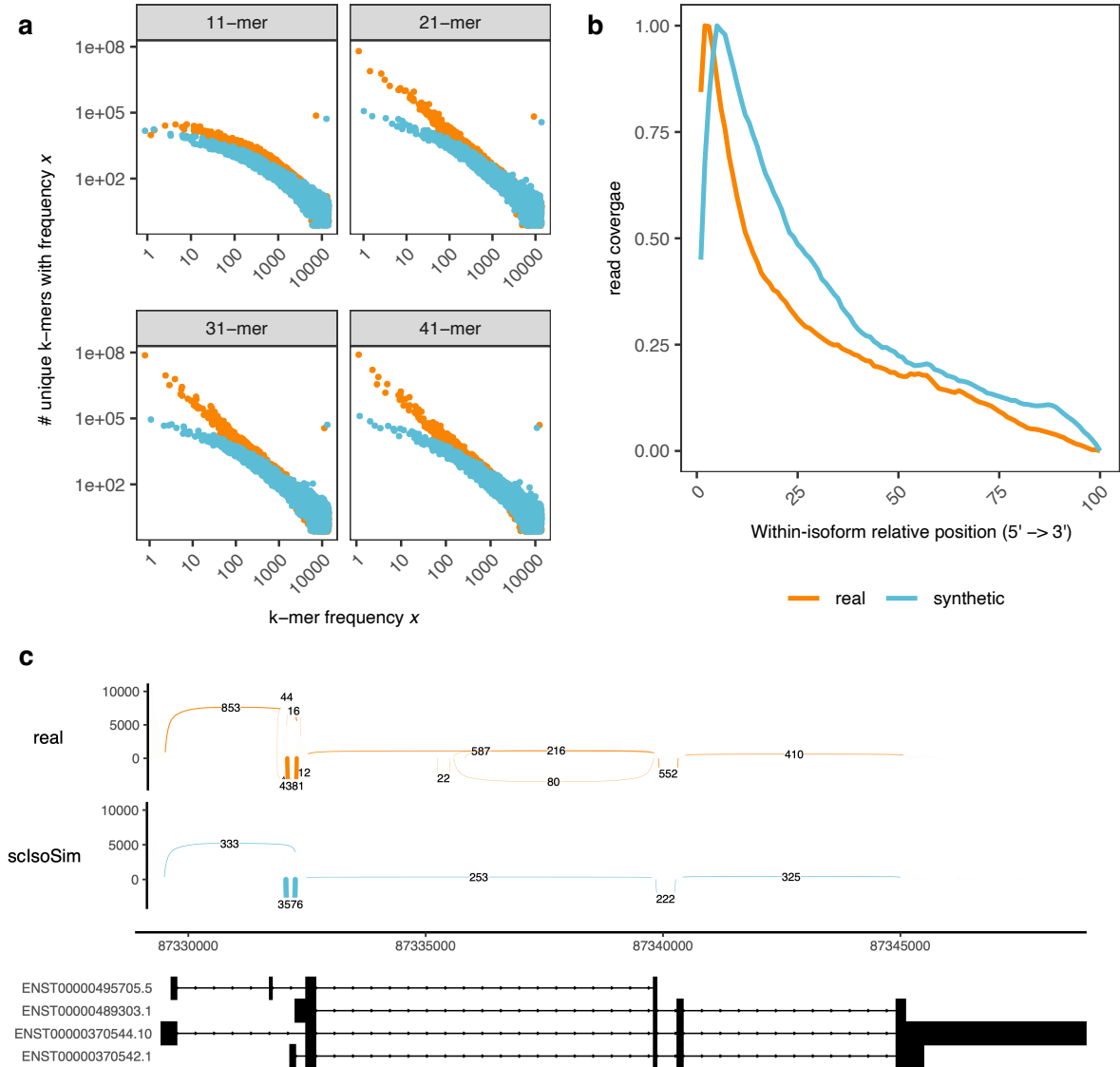


Figure 3.9: scIsoSim's synthetic data resemble real 10x 5' single-cell NSCLC tumor dataset [2].

a, Comparison of the k-mer spectra between scReadSim's synthetic data and real data. The x-axis refers to the occurrence frequency of a specific k-mer, and the y-axis represents the number of unique k-mers with this frequency. Both the x-axis and y-axis are in the $\log_{10}$ scale. **b**, Comparison of read coverage within isoforms between scIsoSim's synthetic data and real data. The x-axis represents the within-isoform relative position from 5' to 3', and the y-axis displays the min-max normalized read coverage. **c**, Preservation of splice junction read coverage by scIsoSim through a sashimi plot. The x-axis indicates genomic regions, while the y-axis represents per-base read density. Different isoform structures are displayed at the bottom of the plot. Arcs connecting exons illustrate splice junction reads, with arc width proportional to the number of reads spanning the junction, and each arc labeled with the splice junction read coverage.

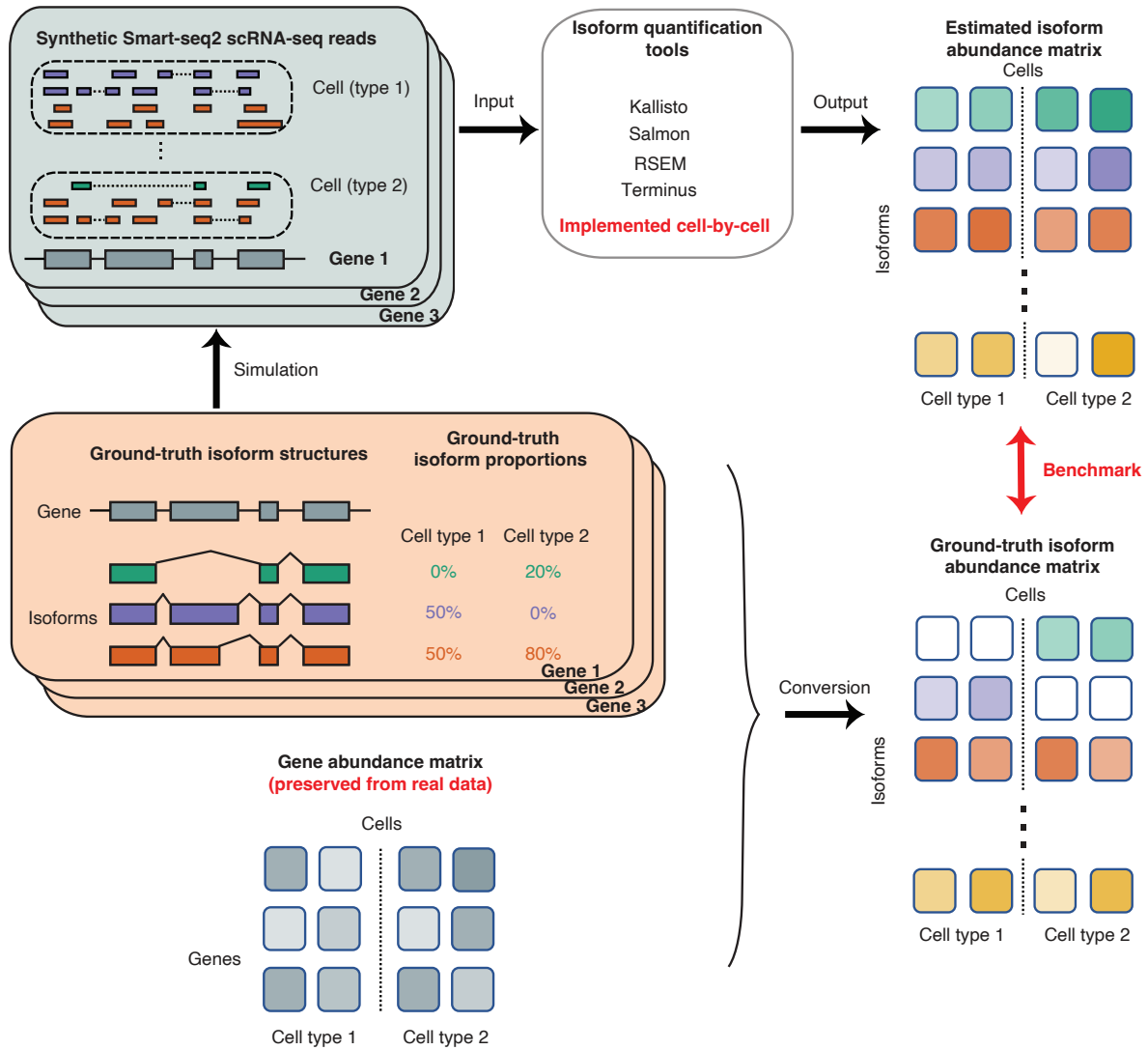


Figure 3.10: Benchmarking workflow of scIsoSim for isoform quantification tools using Smart-seq2 scRNA-seq data.

The workflow benchmarks isoform quantification tools by supplying them with synthetic Smart-seq2 scRNA-seq reads generated by scIsoSim. Each tool produces an isoform-by-cell abundance matrix, which is compared against the ground-truth matrix. scIsoSim derives ground-truth isoform abundances from user-specified isoform structures and proportions, combined with gene-level abundance obtained from real data.

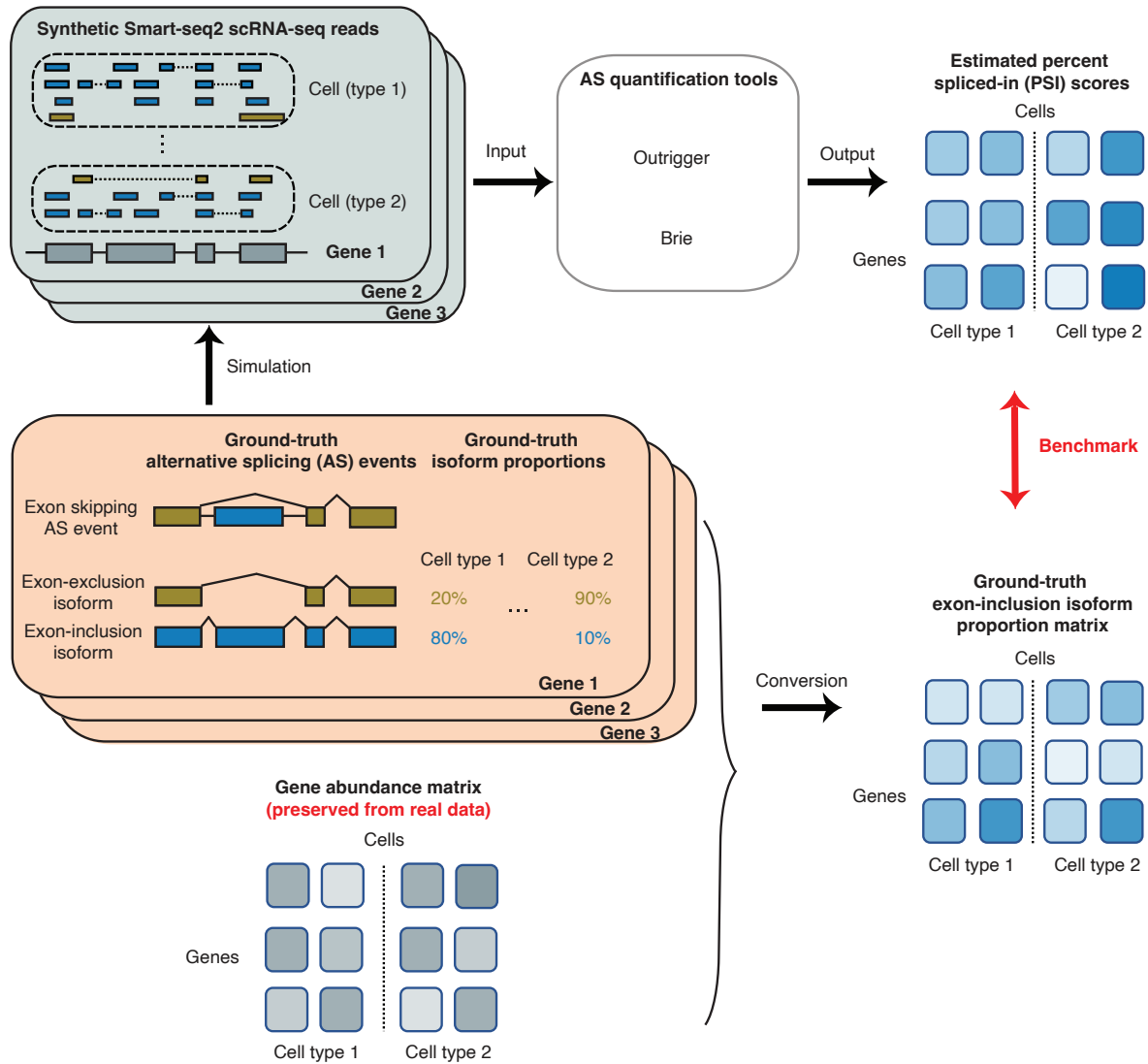


Figure 3.11: Benchmarking workflow of scIsoSim for AS quantification tools using Smart-seq2 scRNA-seq data.

The workflow benchmarks isoform quantification tools by supplying them with synthetic Smart-seq2 scRNA-seq reads generated by scIsoSim. Each tool produces a gene-by-cell percent spliced-in (PSI) score matrix, which is compared against the ground-truth gene-by-cell exon-inclusion isoform proportion matrix. scIsoSim derives ground-truth exon-inclusion isoform proportion from user-specified isoform structures and proportions, combined with gene-level abundance obtained from real data.

S3.8.2 Supplementary tables

Table S3.1: Comparison of scRNA-seq protocols for isoform quantification

	Smart-seq2	10x Chromium 3'	10x Chromium 5'
Cell Throughput	Low-to-medium	High	High
Sequencing Depth per Cell	High	Low	Low
Coverage Region	Entire transcript	3' end	5' end
Distinguishable Isoforms	All isoforms	Only those with distinct 3' exons	Only those with distinct 5' ends
Isoform Quantification Tools	Bulk tools used per cell	Scasa	No major tool
Other Tasks	Alternative splicing	Alternative polyadenylation	Alternative TSS ¹

¹: TSS refers to transcription starting site.

Table S3.2: Comparison of scRNA-seq Read Simulators and Procedures

	scIsoSim	scReadSim	RSEM¹	Kallisto + Splatter + Polyester
Input	scRNA-seq reads	scRNA-seq reads	RNA-seq reads	scRNA-seq reads
Output	FASTQ BAM	FASTQ BAM	FASTQ	FASTA
NGS technologies	Droplet-based Plate-based	Droplet-based Plate-based	Plate-based	Plate-based
Gene abundance preservation ²	✓	✓	✓	✗ ³
Ground-truth isoform composition	✓	✗	✓	✓
Cell number flexibility ⁴	✓	✓	✗	✗
Seq. depth ⁵ flexibility	✓	✓	✗	✗
Isoform composition flexibility	✓	✗	✗	✗
Standalone software	✓	✓	✗ ⁶	✗

¹: RSEM was designed to simulate bulk-level RNA-seq reads. We incorporated it in this table since [3] implemented it for Smart-seq2 scRNA-seq simulation by modeling real cells individually and then simulating synthetic data.

²: “Gene abundance preservation” means whether the simulator generates synthetic cells that mimic real cells regarding genes’ transcript abundance.

³: Splatter loses the genes’ identity when simulating the isoform-by-cell abundance matrix by modeling the real abundance matrix.

⁴: Flexibility means whether users have the flexibility to modify the synthetic data given the input real data.

⁵: Seq. depth refers to sequencing depth.

⁶: RSEM’s software was designed to simulate bulk-level RNA-seq reads and does not have a module for scRNA-seq read simulation.

CHAPTER 4

Categorization of 34 computational methods to detect spatially variable genes from spatially resolved transcriptomics data

4.1 Introduction

In multicellular organisms, cells work cooperatively in tissues and organs to fulfill biological functions. Measuring cells' spatial locations along with transcriptome profiles can help unravel collaborative cell organizations and molecular mechanisms in tissues and organs [3]. Spatially resolved transcriptomics (SRT) technologies have been rapidly evolving to enable high-throughput profiling of transcriptomes at spatial locations, which may represent single cells or groups of cells, in a tissue slice. SRT data provide unprecedented insights into genes' spatial expression patterns, tissue's cellular organizations, and cell-cell communications [4].

To date, major SRT technologies are either imaging-based or sequencing-based, and the two types of technologies have complementary advantages [3]. Imaging-based SRT technologies use fluorescence in situ hybridization (FISH) to measure the expression levels of selected 200–400 target genes at the single-cell or subcellular spatial resolution [3]. Examples of imaging-based SRT technologies include in situ sequencing (ISS) [98], sequential fluorescence in situ hybridization (seqFISH) [99–101], multiplexed error-robust fluorescence in situ hybridization (MERFISH) [102], STARmap [103], ExSeq [104], and 10x Xenium [105]. The high spatial resolution of these approaches makes them particularly useful for studies requiring precise localization of transcripts within tissue sections. For example, MERFISH is widely used to annotate cell types in subregions of the mouse brain [106, 107] and to

investigate the spatial interaction between specific cell types in the mouse somatosensory cortex [108]. However, imaging-based SRT technologies cannot provide transcriptome-wide gene expression levels.

In contrast, sequencing-based SRT technologies, such as Spatial Transcriptomics [100], 10x Visium [109], Slide-seq [110], and GeoMx [111], capture transcriptome-wide gene expression at a lower spatial resolution. Each spot’s diameter ranges between 10 μm and 100 μm , containing multiple cells of possibly different types [3]. Recently developed sequencing-based SRT technologies [112, 113] can achieve higher, even single-cell, spatial resolution, with spot diameters reduced to below 1 μm , despite the significantly higher cost. Sequencing-based SRT technologies enable transcriptome-wide analyses, such as identifying novel marker genes for tissue layers [114], and sequencing-read analysis, such as inferring RNA velocity from spliced and unspliced sequencing reads [115].

Although imaging-based and sequencing-based SRT technologies have different spatial resolutions, for simplicity, we use the same term “spots” to refer to the measured spatial locations for both technologies in this review. Hence, SRT data encompass two components: (1) an expression count matrix of p genes at n spots and (2) a spatial coordinate matrix containing the two-dimensional (2D) coordinates of n spots.

As with other types of high-throughput data where hundreds or thousands of genes are measured simultaneously, a crucial early step of SRT data analysis is the identification of informative genes. Prior to SRT data, single-cell RNA-seq data contain gene expression profiles of single cells without spatial information. In typical single-cell RNA-seq data analysis (Fig. 4.1), highly variable gene (HVG) detection methods are used to screen a proportion of genes (e.g., 10%–20%) with the largest variances (adjusting for cell library sizes in several methods) to reduce the dataset’s dimensionality (from p genes to a smaller number of HVGs) and remove the unimportant variations of many genes. The underlying assumption of HVG detection is that genes exhibiting significant expression variations across single cells are more likely to reflect biological variations rather than technical variations caused by sampling effects in sequencing. Analogous to single-cell RNA-seq data analysis, SRT data analysis typically includes an early step to detect *spatially variable genes (SVGs)*, which con-

ceptually generalize HVGs by including the spatial information (Fig. 4.1). Intuitively, SVGs are genes whose gene expression levels exhibit non-random, informative spatial patterns.

In single-cell RNA-seq data analysis (Fig. 4.1), common steps after HVG detection include cell clustering and differentially expressed gene (DEG) detection. The goal of cell clustering is to identify potential cell types, and subsequent DEG detection aims to find the genes that are significantly more highly expressed in each cluster. The resulting highly-expressed DEGs of each cell cluster are then used to annotate the cluster as a particular cell type (or subtype), serving as cell-type markers. Similarly, in SRT data analysis, following SVG detection, spots are often partitioned into spatial domains, where each domain contains proximal spots exhibiting similar gene expression profiles. Parallel to DEG detection between clusters in single-cell RNA-seq data analysis, DEGs can be identified between spatial domains, and the resulting highly-expressed DEGs of each spatial domain serve as spatial-domain markers.

Many computational methods have been developed to detect SVGs. However, the SVG definitions in these methods lack consensus, resulting in diverse meanings for the detected SVGs and making understanding difficult. Consequently, the inconsistent definitions of SVGs may lead to ambiguous usage of computational methods. Although 19 methods were reviewed previously [116], the review did not categorize SVG definitions but only focused on summarizing the methodologies, including the input data type (count data vs. normalized data), algorithm type (model-based vs. model-free), statistical paradigm (frequentist vs. Bayesian), the availability of false discovery rate (FDR) control, etc. Nor did the review discuss the biological implications and downstream applications of SVGs.

Motivated by this gap in understanding SVGs, here we review 34 peer-reviewed SVG detection methods (Fig. 4.2) and define three SVG categories: *overall SVGs*, *cell-type-specific SVGs*, and *spatial-domain-marker SVGs*. For the first time, our review unveils the biological significance underpinning the three categories of SVGs, summarizes the frequentist hypothesis tests implemented in 23 SVG detection methods, and discusses the limitations of existing methods and outlines future directions for improvement. Moreover, we construct a hierarchy to summarize the 34 SVG detection methods in terms of methodological differences in Fig. 4.3 and list the technical details in Table 4.1 and Supplementary Figs. 4.6–4.7. Addi-

tionally, to help readers better understand our review, we provide Table 4.2, which lists the definitions of the frequently used terminologies.

Methods for detecting the three SVG categories serve different purposes (Fig. 4.2a). First, the detection of overall SVGs screens informative genes for downstream analysis, including the identification of spatial domains and functional gene modules. Second, detecting cell-type-specific SVGs aims to reveal spatial variation within a cell type and help identify distinct cell subpopulations or states within cell types. Third, spatial-domain-marker SVG detection is used to find marker genes to annotate and interpret spatial domains already detected. These markers help understand the molecular mechanisms underlying spatial domains and assist in annotating tissue layers in other datasets.

The relationship among the three SVG categories depends on the detection methods, particularly the null and alternative hypotheses they employ. If an overall SVG detection method uses the null hypothesis that a non-SVG’s expression is independent of spatial location and the alternative hypothesis that any deviation from this independence indicates an SVG, then its SVGs should theoretically include both cell-type-specific SVGs and spatial-domain-marker SVGs. For example, DESpace [117] is a method that detects both overall SVGs and spatial-domain-marker SVGs, and its detected overall SVGs must be marker genes for some spatial domains. This inclusion relationship holds true except in extreme scenarios, such as when a gene exhibits opposite cell-type-specific spatial patterns that effectively cancel each other out. However, if an overall SVG detection method’s alternative hypothesis is defined for a specific spatial expression pattern, then its SVGs may not include some cell-type-specific SVGs or spatial-domain-marker SVGs. For readers with a statistical background seeking deeper insights, we provide a comprehensive discussion on the frequentist hypothesis tests implemented by 23 SVG detection methods in the Section 4.5. Other readers can skip this section and still grasp the core content of our review.

Although three benchmark studies were conducted to compare SVG detection methods [38–40], they have three major differences from our review. First, the benchmark studies performed numerical comparisons of SVG detection methods, while our review focuses on categorizing methods conceptually and methodologically. Second, the benchmark studies

did not categorize SVGs but focused on detecting the overall SVGs by our definition. For example, spaGCN, a method that detects spatial-domain-marker SVGs, does not perform well in a benchmark study [40], possibly because the study focused on detecting overall SVGs. This observation underscores the need to categorize SVGs. Third, our review has a more comprehensive coverage of SVG detection methods: 34 peer-reviewed methods (all methods to our knowledge) compared to 6 methods in Charitakis *et al.* [38], 7 methods in Chen *et al.* [39], and 14 methods (12 peer-reviewed) in Li *et al.* [40]. In summary, existing benchmark studies complement our review by providing numerical evidence to rank methods in detecting certain types of overall SVGs. Future benchmark studies are needed to compare methods within the three SVG categories and examine more types of null and alternative hypotheses.

To facilitate our discussion, we introduce the following mathematical notations. For each gene with expression levels (in counts or normalized) measured at n spatial spots, we use $y_i \in \mathbb{R}$ represent its expression level at spot $i = 1, \dots, n$. When gene expression levels are considered random variables, we use Y_i to emphasize the random nature. The vector notation is $\mathbf{Y} = (Y_1, \dots, Y_n)^\top$. For each spot $i = 1, \dots, n$, we denote its 2D spatial location as $\mathbf{s}_i = (s_{i1}, s_{i2})^\top \in \mathbb{R}^2$. The spatial coordinates of all n spots are represented by the matrix $\mathbf{s} = [\mathbf{s}_1, \dots, \mathbf{s}_n]^\top \in \mathbb{R}^{n \times 2}$, where each row corresponds to a spot. When spots are annotated with L spatial domain labels, the spatial-domain indicator vector for spot i is $\mathbf{d}_i = (d_{i1}, \dots, d_{iL})^\top \in \{0, 1\}^L$, with $\sum_{l=1}^L d_{il} = 1$; that is, $d_{il} = 1$ means that spot i belongs to spatial domain l . When spots are annotated with K cell type labels, the cell-type proportion vector for spot i is $\mathbf{c}_i = (c_{i1}, \dots, c_{iK})^\top \in [0, 1]^K$, where c_{ik} indicates the proportion of cell type k at spot i , with $\sum_{k=1}^K c_{ik} = 1$. In summary, each spot i has three covariate vectors: spatial location $\mathbf{s}_i$, spatial-domain indicator $\mathbf{d}_i$, and cell-type proportions $\mathbf{c}_i$. We summarize the mathematical notations in Table 4.3.

4.2 Methods for detecting overall SVGs

We define *overall SVGs* as the genes that exhibit non-random spatial expression patterns. This is the most general category of SVGs detected using only SRT data without incorporating external information such as spatial domains or cell types (See Fig. 4.2a and Table 4.2). Methods for detecting overall SVGs are generally classified into Euclidean-space-based and graph-based methods (Fig. 4.3). Euclidean-space-based methods analyze spots in a tissue slice by considering their locations in a 2D Euclidean space. In contrast, graph-based methods first construct a graph of spatial spots and then analyze this graph, focusing on the connections between spots rather than their Euclidean distances. Both approaches, regardless of the spatial representation they use (Euclidean space or graph), study how a gene's expression levels vary across spatial spots, examining the relationship between each gene and its spatial context.

4.2.1 Euclidean-space-based methods

Euclidean-space-based methods are further divided based on whether they use a kernel function to target a specific spatial pattern. Kernel-based methods utilize a kernel function to specify the covariances of spatial spots, enhancing their power to detect the targeted spatial patterns. In contrast, kernel-free methods do not use a kernel function and instead rely on other approaches to capture spatial patterns.

Kernel-based methods. Kernel-based methods use a pre-defined kernel function $K(\mathbf{s}_i, \mathbf{s}_j)$ to specify the covariance between spots i and j based on their spatial locations. This kernel function usually decays as the Euclidean distance between spots i and j increases. Define $\mathbf{K}(\mathbf{s}) = [K(\mathbf{s}_i, \mathbf{s}_j)]$ as the $n \times n$ matrix representing these covariances. The most commonly used kernel functions include the Gaussian kernel for detecting clustered or focal expression patterns, expressed as

$$K_G(\mathbf{s}_i, \mathbf{s}_j) = \exp \left(-\frac{\|\mathbf{s}_i - \mathbf{s}_j\|^2}{2\sigma^2} \right), \quad (4.1)$$

and the cosine kernel for detecting periodic expression patterns, denoted by

$$K_C(\mathbf{s}_i, \mathbf{s}_j) = \cos \left(2\pi \frac{\|\mathbf{s}_i - \mathbf{s}_j\|}{\phi} \right). \quad (4.2)$$

For a given gene, the covariance matrix of its expression vector $\mathbf{Y}$ (may be subject to transformation) can be decomposed into components, one of which depends on $\mathbf{K}(\mathbf{s})$. If the contribution of this term is significantly positive according to a hypothesis test, commonly known as the “variance component test,” the gene is detected as an overall SVG.

Following this general idea, ten kernel-based methods (Fig. 4.3) make different assumptions about the distribution of $\mathbf{Y}$ and the covariance matrix decomposition. Below, we briefly summarize these methods and highlight their similarities and differences.

SpatialDE [118] assumes that a gene’s normalized expression $\mathbf{Y} \in \mathbb{R}^n$ follows an n -dimensional Gaussian distribution, with a covariance matrix that includes a spatial covariance component involving $\mathbf{K}(\mathbf{s})$:

$$\mathbf{Y} \sim \text{MVN}(\mu, \sigma_s^2 \cdot \mathbf{K}(\mathbf{s}) + \delta \cdot \mathbf{I}),$$

where MVN represents a multivariate Gaussian distribution with a mean vector $\mu \in \mathbb{R}^n$ and a covariance matrix $\sigma_s^2 \cdot \mathbf{K}(\mathbf{s}) + \delta \cdot \mathbf{I}$. Here, $\mathbf{I}$ is an n -dimensional identity matrix, and σ_s^2 and δ are parameters representing the spatial covariance component and the error variance component, respectively. This model is a realization of a Gaussian process with the kernel function $K(\cdot, \cdot)$. To determine if a gene is an overall SVG, SpatialDE employs a likelihood ratio test with the null hypothesis $H_0 : \sigma_s^2 = 0$, comparing this model with a null model that does not include the spatial covariance component: $\mathbf{Y} \sim \text{MVN}(\mu, \sigma^2 \cdot \mathbf{I})$.

Using the same model as SpatialDE, **nnSVG** [119] improves computational efficiency by replacing the Gaussian process with the nearest-neighbor Gaussian process [120], providing a scalable approximation. This enhancement allows nnSVG’s computational complexity and runtime to scale linearly with n , the number of spatial spots, rather than cubically.

SOMDE [121] also follows the statistical model of SpatialDE but adds a data prepro-

cessing step for scalability. Specifically, SOMDE condenses the original spatial spots into fewer grid points and assigns each grid point a “meta-expression,” which aggregates a gene’s expression levels at the spots condensed into that grid point. After this preprocessing step, SOMDE uses the same approach as SpatialDE to detect whether the gene is an overall SVG, but it does so using the grid points instead of the original spots.

Compared to the first three methods, **SVCA** [122] modifies the covariance matrix decomposition by adding two additional components:

$$\mathbf{Y} \sim \text{MVN}(\mu, \mathbf{K}_{\text{int}} + \mathbf{K}_{\text{c-c}} + \sigma_s^2 \cdot \mathbf{K}(\mathbf{s}) + \sigma^2 \cdot \mathbf{I}),$$

where $\mathbf{K}_{\text{int}}$ denotes the intrinsic cell-state covariance, and $\mathbf{K}_{\text{c-c}}$ represents the cell-cell interaction covariance. Specifically, $\mathbf{K}_{\text{int}}$ is defined solely using gene expression data without spatial information, while $\mathbf{K}_{\text{c-c}}$ incorporates both gene expression data and Euclidean distances among spots.

Instead of using normalized gene expression data, **SPARK** [123], **GPcounts** [124], and **BOOST-GP** [125] directly model a gene’s expression count at spot i , $Y_i \in \mathbb{N}$, using Poisson, negative binomial (NB), and zero-inflated negative binomial (ZINB) distributions, respectively. For instance, **SPARK** assumes that

$$Y_i \stackrel{\text{ind}}{\sim} \text{Poisson}(\mu_i(\mathbf{s}_i)), \quad i = 1, 2, \dots, n.$$

For spot i , the Poisson mean parameter $\mu_i(\mathbf{s}_i)$ is a function of the spatial location $\mathbf{s}_i$, specified by

$$\log(\mu_i(\mathbf{s}_i)) = \mathbf{x}_i^\top \beta + b_i(\mathbf{s}_i) + \epsilon_i,$$

where $\mathbf{x}_i$ indicates spot i ’s covariates, and $b_i(\mathbf{s}_i)$ is the random intercept at spot i . The n random intercepts, $b_1(\mathbf{s}_1), \dots, b_n(\mathbf{s}_n)$, are assumed to follow a Gaussian process with the kernel function $K(\cdot, \cdot)$:

$$(b_1(\mathbf{s}_1), b_2(\mathbf{s}_2), \dots, b_n(\mathbf{s}_n))^\top \sim \text{MVN}(0, \sigma_s^2 \cdot \mathbf{K}(\mathbf{s})).$$

To decide whether the gene is an overall SVG, SPARK tests the null hypothesis $H_0 : \sigma_s^2 = 0$. Thus, SPARK shares the same variance component test idea with SpatialDE but uses a hierarchical model, incorporating a Gaussian process in the top layer and Poisson distributions in the bottom layer, to account for count data.

Unlike the previous seven methods that use parametric models, three methods—**SPARK-X** [126], **SMASH** [127], and **singlecellHaystack** [128]—employ non-parametric tests to determine whether a gene’s expression is independent of its spatial location.

SPARK-X [126] tests whether two $n \times n$ spot similarity matrices are independent to decide if a gene is an overall SVG. One similarity matrix is based on the gene’s expression levels at the n spots, defined as $\mathbf{E} = \mathbf{Y} (\mathbf{Y}^\top \mathbf{Y})^{-1} \mathbf{Y}^\top$. The other is based on the kernel-transformed spatial locations of the n spots, defined as $\mathbf{K} = \mathbf{s}' ((\mathbf{s}')^\top \mathbf{s}')^{-1} (\mathbf{s}')^\top$, where $\mathbf{s}'$ is an $n \times 2$ matrix whose i -th row $\mathbf{s}'_i = (s'_{i1}, s'_{i2})^\top$ encode the kernel-transformed spatial location of spot $i = 1, \dots, n$. Specifically, SPARK-X transforms the spatial location $\mathbf{s}_i = (s_{i1}, s_{i2})^\top$ of spot i using two kernel-based functions: a Gaussian transformation $s'_{il} = \exp\left(\frac{-s_{il}^2}{2\sigma_l^2}\right)$, $l = 1, 2$, to detect clustered or focal patterns, and a cosine transformation $s'_{il} = \cos\left(\frac{2\pi s_{il}}{\phi_l}\right)$, $l = 1, 2$, to detect periodic patterns, where σ_1 , σ_2 , ϕ_1 , and ϕ_2 are tuning parameters. Eventually, SPARK-X uses the Pearson correlation between the two similarity matrices, $\text{trace}(\mathbf{E}\mathbf{K})/n$, as the test statistic to decide if a gene is an overall SVG.

SMASH [127] generalizes SPARK-X by adding two alternatives to the spatial location similarity matrix $\mathbf{K}$. The two alternatives include the Gaussian kernel covariance matrix $\mathbf{K}_G = [K_G(\mathbf{s}_i, \mathbf{s}_j)]$ and the cosine kernel covariance matrix $\mathbf{K}_C = [K_C(\mathbf{s}_i, \mathbf{s}_j)]$, where $K_G(\cdot, \cdot)$ and $K_C(\cdot, \cdot)$ are the kernel functions defined in Equations (4.1) and (4.2), respectively. Then, SMASH uses the same test statistic as SPARK-X.

singlecellHaystack [128] is a test of independence between a gene’s expression level and its spatial location. It involves two pre-processing steps: first, the gene’s expression levels at various spots are binarized into two states: detected and undetected; second, the 2D Euclidean space of a tissue slice is divided into a grid along both axes, with intersection points defined as grid points, serving as coarse spatial coordinates. After pre-processing, the

method tests if the gene’s two expression states are randomly distributed across the grid points. If they are not, the gene is considered an overall SVG. `singlecellHaystack` uses a Gaussian kernel to define three distributions of grid points: a reference distribution based on all spatial spots, a conditional distribution based on spatial spots in the detected state, and another conditional distribution based on spatial spots in the undetected state. Technically, for a grid point with a 2D spatial location $\mathbf{s} = (s_1, s_2)^\top$, its reference distribution density is defined as $\sum_{i=1}^n \exp\left(-\frac{\|\mathbf{s}_i - \mathbf{s}\|^2}{2}\right)$, subject to normalization. The two conditional distribution densities are similarly defined, with the sum across all spots replaced by sums across the detected and undetected spots, respectively. The test statistic is defined as the sum of two Kullback-Leibler (KL) divergences, each representing the deviance of one conditional distribution from the reference distribution. Intuitively, the larger the test statistic, the more likely the gene is an overall SVG. Finally, the test statistic is converted to a p-value using a permutation test, which shuffles the gene’s expression levels across spots.

To summarize, all ten kernel-based methods require pre-specification of the kernel function $K(\cdot, \cdot)$ to target specific spatial patterns. Consequently, the statistical power of these methods depends on how well the kernel function captures the spatial expression patterns of biologically informative genes in the SRT data. Due to the limited choices and subjectivity of kernel functions, these methods may be insufficient for detecting complex spatial expression patterns, such as those found in cancer SRT data.

Kernel-free methods. Nine kernel-free methods employ diverse approaches to detect overall SVGs without relying on pre-specified kernel functions (Fig. 4.3). Instead, they utilize various statistical and computational techniques to capture spatial patterns. Below, we briefly summarize each method.

Trendsceek [129] tests if a gene’s expression level is dependent of the spatial location using a marked point process, which models the joint probability distribution of spatial locations $\mathbf{s}_i$, $i = 1, \dots, n$, as “points” and a given gene’s expression levels y_i , $i = 1, \dots, n$, as “marks.” If deemed dependent, the gene is detected as an overall SVG. For a pair of points $(\mathbf{s}_i, \mathbf{s}_j)$ and their corresponding marks (y_i, y_j) , Trendsceek parameterizes their joint probability density by the distance (called “radius”) $r_{ij} = \|\mathbf{s}_i - \mathbf{s}_j\|$ and the marks y_i, y_j

as $f(y_i, y_j, r_{ij})$. Formally, the dependence of the gene’s expression on spatial location is formulated as a conditional distribution of a pair of gene expression levels ($y_i = y_1, y_j = y_2$), given the radius $r_{ij} = r$:

$$M(y_1, y_2|r) = \frac{f(y_1, y_2, r)}{f(r)}.$$

This conditional distribution can then be summarized into four mark-segregation summary statistics as functions of the radius r , such as Stoyan’s mark-correlation function. To test if the gene is an overall SVG, Trendsceek implements a permutation test by sampling marks without replacement and randomly reassigning them to points. For each radius r , summary statistics are calculated and compared to the null distributions derived from the permutations to obtain p-values, which are then combined into a single p-value across all radius values. Note that Trendsceek has inspired the “markvariogram” approach used in the single-cell and SRT analysis toolkit Seurat [130] for SVG detection.

MULTILAYER [131], similar to singlecellHaystack, discretizes spatial spots for each gene based on the gene’s expression levels. The discretization procedure in MULTILAYER consists of three steps. First, each spot receives a $\log_2$ fold change, defined as the ratio of the gene’s normalized expression level at the spot to its average normalized expression level across all spots. The $\log_2$ fold changes with absolute values greater than 1 are truncated to 1 with the corresponding sign. Second, MULTILAYER applies agglomerative clustering to the spots based on the $\log_2$ fold change values, using a pre-specified distance threshold. Third, if a cluster contains contiguous spots with all positive $\log_2$ fold change values, MULTILAYER assigns the cluster a label of “1.” Finally, MULTILAYER uses the size of the largest cluster labeled as “1” as a summary statistic for the gene and ranks genes from high to low based on this statistic. Thus, if a gene has a large contiguous cluster labeled as “1,” it is considered to have a high-expression neighborhood and is regarded as an overall SVG. However, MULTILAYER does not provide statistical significance for its overall SVG ranking.

sepal [132] is a diffusion-process-based method that identifies a gene as an overall SVG if its spatial expression pattern deviates from randomness. To quantify this deviation, sepal simulates a diffusion process from the gene’s observed expression pattern until it converges

to randomness, using the convergence time as a measure. The longer the convergence time, the more likely the gene is an overall SVG. However, this method ranks genes but does not provide statistical significance for the detected overall SVGs. Specifically, *sepal* uses the diffusion equation based on Fick’s second law [133] to simulate the diffusion of the gene’s transcripts in the 2D Euclidean space of a tissue slice. Note that *sepal* is included in the SRT analysis toolkit *squidpy* [134] for SVG detection.

BSP [135] is built upon the intuition that an overall SVG should have its spatial expression pattern vary significantly at different spatial resolutions; otherwise, the pattern should remain consistent. To implement this, BSP follows a four-step procedure. First, for each spot, it defines “big” and “small” patches consisting of neighboring spots within pre-specified large and small radii, respectively, so each spot has one big patch and one small patch. Second, for each gene, BSP calculates the average expression levels within the big patches and the variance of these averages, then similarly calculates the variance for the small patches. Third, BSP uses the ratio of the big-patch variance to the small-patch variance as a statistic to summarize the change in the gene’s expression pattern with changing spatial resolution. Finally, BSP ranks genes by this statistic, identifying those with large values as overall SVGs. Although BSP attempts frequentist inference by defining a null distribution for its statistic, this null distribution is improperly defined as a distribution fitted to the statistic values of all genes, implying that all genes are non-SVGs. Consequently, a p-value threshold of 0.05 will always lead to 5% of genes being detected as overall SVGs, regardless of how many genes are truly overall SVGs. Therefore, we do not consider BSP to provide valid statistical inference in our review. While the ranking of genes by the BSP statistic could still be informative, the lack of valid statistical inference means that we do not have a threshold on the statistic to reflect a target FDR.

PROST [136] determines if a given gene is an overall SVG based on computer image segmentation. First, treating the gene’s spatial expression in a tissue slice as an image, PROST applies image segmentation techniques to detect multiple foreground regions, where the gene is considered expressed, and one background region, where the gene is treated as unexpressed. Second, PROST defines two statistics: (1) a significance factor summarizing

the overall elevation of the gene’s expression from the background region to the foreground regions, and (2) a separability factor measuring the overall homogeneity of the gene’s expression within each foreground region. Finally, PROST combines these two statistics into a single index. Based on this index, PROST ranks genes from high to low and selects the top-ranking genes as overall SVGs. Although PROST performs frequentist statistical tests, they are for Moran’s I [137] rather than its index. Therefore, we do not consider PROST to provide valid statistical inference in our review.

HEARTSVG [138] is a time-series-based method designed to detect spatially variable genes (SVGs) by first converting a gene’s expression levels in a 2D tissue slice, structured as a matrix with n_{row} rows and n_{col} columns, into four one-dimensional time series. Notably, HEARTSVG assumes that the tissue slice is square, which may impose a limitation. These four time series, termed “marginal expression series,” are generated through a process called “semi-pooling.” For example, if a tissue slice has $n_{\text{row}} = 120$ and $n_{\text{col}} = 60$, HEARTSVG defines the following four time series for each gene: (1) a per-row-average time series with 120 entries, each representing the average expression level within the corresponding row; (2) a within-row-window-average time series with $120 \times \frac{60}{\ln(120)} = 1440$ entries, where each entry is derived from averaging expression levels within a non-overlapping sliding window of size $\ln(120) = 5$ within each row; (3) a per-column-average time series with 60 entries, each representing the average expression level within the corresponding column; and (4) a within-column-window-average time series with $60 \times \frac{120}{\ln(60)} = 1800$ entries, where each entry is derived from averaging expression levels within a non-overlapping sliding window of size $\ln(60) = 4$ within each column. HEARTSVG then applies the Portmanteau test to each of these four time series to assess whether there are significant autocorrelations, which would suggest that the time series displays a trend or periodic pattern, potentially indicating an informative spatial pattern in the original 2D square. The four p-values, obtained from the Portmanteau tests on the four time series, are then combined into a single p-value using Stouffer’s method; that is, the gene should be considered an overall SVG if the combined p-value falls below a certain threshold. Finally, to determine the appropriate p-value threshold and account for multiple testing across all genes, HEARTSVG applies Holm’s method to

adjust the p-values and control the familywise error rate (FWER). It is important to note that HEARTSVG may have certain conceptual limitations, particularly the requirement for a square tissue slice and the seemingly arbitrary definitions of the four time series, especially regarding the chosen window sizes and the construction of the time series, given the many possible alternatives.

SPADE [139] leverages the hematoxylin-and-eosin (H&E) image accompanying the SRT data (e.g., 10x Visium) to detect overall SVGs. It defines a gene as an overall SVG if the gene’s expression levels can be predicted by H&E features, which are extracted by a pre-trained convolutional neural network and expected to contain critical spatial information, in a linear model. Specifically, SPADE first performs principal component analysis (PCA) on the 512 H&E features for dimensionality reduction. Then, it tests if the resulting principal components can predict the gene’s expression levels in a linear model. If so, the gene is defined as an overall SVG.

BOOST-MI [140] and **BOOST-HMI** [141] are Bayesian methods that detect overall SVGs based on the Ising model and its extension to the geostatistical mark interaction model, respectively. The Ising model, historically used to model ferromagnetism, represents each magnetic moment (or spin) in a material as a discrete variable that can take one of two values: +1 or -1 (interpreted as spin up or spin down). In ferromagnetic materials, spins tend to align with their neighbors to minimize the system’s energy. The Ising model exhibits a phase transition at a critical temperature, below which spins align, resulting in net magnetization corresponding to the ferromagnetic phase. Above this temperature, thermal fluctuations dominate, and spins are randomly oriented, corresponding to the paramagnetic phase. BOOST-MI applies the Ising model to SRT data by treating spatial spots as spins, with the positive and negative values corresponding to high and low expression levels of a gene after dichotomization. BOOST-MI detects a gene as an overall SVG if it finds the gene to be in a ”ferromagnetic phase.” BOOST-HMI extends this approach for imaging-based SRT data to accommodate the irregular spatial distribution of measured spots.

4.2.2 Graph-based methods

In manifold learning, when data originate from a non-linear manifold within Euclidean space, using a graph to represent the data effectively captures the intrinsic geometry and preserves the local structures of the manifold. Extending this concept to spatial spots on a non-linear manifold within the 2D Euclidean space of a tissue slice, graph-based SVG detection methods first construct a neighborhood graph by connecting nearby spots in Euclidean space. These methods then operate on the graph to detect overall SVGs. The way the graph is constructed is crucial, as it determines how well the graph represents the manifold, significantly impacting the performance of graph-based methods.

In this section, we introduce ten graph-based methods (Fig. 4.3), focusing on their graph construction approaches and subsequent operations on the graph for detecting overall SVGs.

Hotspot [142] aims to detect overall SVGs as the genes that exhibit high expression levels in local spot neighborhoods (i.e., “hotspots”). It first constructs a K -nearest neighbor (KNN) directed graph by treating each spot as a node and connecting it to its closest K spots in Euclidean distance. For a spot i connected to a spot j , with Euclidean distance denoted by d_{ij} , the directed edge from i to j is assigned a weight defined as $w_{ij} = e^{-d_{ij}^2/\sigma_i^2}$, where σ_i represents the bandwidth of spot i (defined as the distance from spot i to its $[K/3]$ -th neighbor). Next, for each gene, denoting its expression count at spot i by Y_i , Hotspot uses an autocorrelation statistic H to quantify the dependence of the gene’s expression level on the graph structure:

$$H = \sum_i \sum_{j \neq i} w_{ij} Y_i Y_j,$$

which is related to Moran’s I [137], a spatial autocorrelation measure that quantifies the degree to which similar expression levels are clustered together in space. Moran’s I is defined as

$$I = \frac{n}{\sum_i \sum_j w_{ij}} \frac{\sum_i \sum_j w_{ij} (Y_i - \bar{Y})(Y_j - \bar{Y})}{\sum_i (Y_i - \bar{Y})^2}, \quad (4.3)$$

where n is the number of spatial spots, and $\bar{Y}$ is the gene’s mean expression level across spots. Similar to I , a large H indicates that spots where the gene is highly expressed are clustered

in local neighborhoods, suggesting the gene is an overall SVG. To assess the statistical significance of H , Hotspot converts H to a z-statistic by subtracting the expectation of H under the null model, where all spots are assumed to be independent, and dividing by the standard deviation of H under the null model. Specifically, Hotspot assumes two null models where Y_i independently follows either an NB distribution or a Bernoulli distribution. The z-statistic is then assumed to follow the standard Gaussian distribution under the null hypothesis, and a one-sided p-value is computed.

HRG [143] differs from Hotspot in three key aspects. First, it replaces the KNN graph with a shared-nearest-neighbors (SNN) graph, where the edge weight w_{ij} between spots i and j is defined as the Jaccard index of their respective K -nearest neighbors. The Jaccard index is calculated as the size of the intersection of the two sets divided by the size of their union. Second, HRG modifies Y_i , the gene count at spot i , to a z-score $Z_i = \frac{Y_i - \bar{Y}}{\sqrt{\frac{1}{n} \sum_{j=1}^n (Y_j - \bar{Y})^2}}$, where $\bar{Y}$ is the mean gene count and n is the total number of spots. Third, unlike Hotspot, HRG does not provide statistical significance for its detected overall SVGs but instead ranks the genes based on H .

SVGbit [144] is also a spatial-autocorrelation-based method that ranks genes as overall SVGs based on the spatial autocorrelations of local neighborhoods. The method does not provide statistical significance for its rankings. For each gene, SVGbit uses three steps to calculate a summary statistic, called the “aggregation index,” which is used to rank genes. First, SVGbit identifies “hotspots” as spatial spots with large and statistically significant local Moran’s I values, using a z-test-based p-value thresholded at the 5% FDR. Specifically, for spot i , local Moran’s I is defined as

$$I_i = (Y_i - \bar{Y}) \sum_j w_{ij} (Y_j - \bar{Y}),$$

where the terms are defined similarly to Moran’s I (Equation (4.3)). A high I_i value indicates that spot i is part of a neighborhood of similar values (either high-high or low-low). Hence, spot i is treated as a hotspot if I_i is deemed significantly large. Second, for each hotspot, SVGbit considers its neighboring hotspots among the 6-nearest-neighbor spots and defines

the “local aggregation density” as the sum of these neighboring hotspots’ normalized log-transformed p-values (i.e., the log-transformed p-value of each neighboring hotspot divided by the sum of the log-transformed p-values of the 6 neighboring spots). Thus, a hotspot with more neighboring hotspots has a higher local aggregation density. Third, the gene’s aggregation index is computed as the average of all hotspots’ local aggregation densities. Intuitively, the overall SVGs with large aggregation indices are those whose expression patterns exhibit clustered hotspots.

SINFONIA [145] aims to detect overall SVGs as those exhibiting positive spatial autocorrelations, where similar expression levels are clustered in neighboring spots, either globally or locally. It begins by constructing a KNN directed graph, setting the edge weight from spot i to spot j (if connected) as $w_{ij} = 1 - \frac{d_{ij}}{\max_j d_{ij}}$, where d_{ij} is the Euclidean distance between spots i and j , and $\max_j d_{ij}$ is the maximum distance between spot i and its K nearest neighbors. Given this graph, SINFONIA calculates two measures of spatial autocorrelation for each gene: (1) Moran’s I [137], which provides a global measure of spatial autocorrelation, with values ranging from -1 to 1 , where a value close to 1 indicates strong positive spatial autocorrelation; (2) Geary’s C [146], which is more sensitive to local spatial differences, with values ranging from 0 to 2 , where a value close to 0 indicates strong positive spatial autocorrelation. Geary’s C is defined as

$$C = \frac{(n-1)}{2 \sum_i \sum_j w_{ij}} \frac{\sum_i \sum_j w_{ij} (Y_i - Y_j)^2}{\sum_i (Y_i - \bar{Y})^2},$$

where the terms are defined similarly to Moran’s I (Equation (4.3)). SINFONIA then ranks genes from high to low based on Moran’s I and from low to high based on Geary’s C. To identify overall SVGs, SINFONIA takes the union of the top J genes from both rank lists, where J is a pre-specified positive integer. However, SINFONIA does not provide statistical significance for the detected overall SVGs.

MERINGUE [147] also uses Moran’s I to detect overall SVGs but differs from previous methods in its graph construction approach. Instead of using a KNN graph, MERINGUE constructs a graph of spatial spots using Delaunay triangulation, making it suited for spatial

spots with non-uniform density, such as cells in MERFISH data. Delaunay triangulation [148] connects points in a 2D space to form triangles such that no point lies inside the circumcircle (the circle that passes through all three vertices) of any triangle. In simpler terms, it maximizes the minimum angle of the triangles, avoiding skinny triangles. Unlike KNN, Delaunay triangulation does not require selecting an arbitrary K parameter and automatically adapts to variations in point density, providing more connections in denser regions and fewer in sparser regions. Given the graph, MERINGUE assigns an edge weight between spots i and j as $w_{ij} = 1$ if the two spots are connected, and $w_{ij} = 0$ otherwise. To determine if a gene is an overall SVG, MERINGUE uses Moran’s I, transforms it to a z-statistic, and calculates a one-sided p-value based on the standard Gaussian distribution.

SpaGene [149], similar to previous methods that use Moran’s I, aims to detect overall SVGs as those highly expressed at neighboring spots. Starting from a KNN graph of spots, it uses an alternative approach to boost computational efficiency. For each gene, it dichotomizes spots, coding them as 0 or 1 corresponding to low or high expression levels of the gene. It then extracts the high-expression spots (i.e., the spots labeled as 1) from the graph and summarizes the distribution of these spots’ degrees (the number of edges connected to each spot) in the subgraph. Finally, it compares the degree distribution of these high-expression spots to that of the whole graph, and calculates the earth mover’s distance between the two distributions. A small distance indicates that the high-expression spots are densely connected to each other and rarely connected to other spots, so the gene is likely an overall SVG. Finally, SpaGene converts the distance to a p-value using a permutation test.

SpaGFT [150] detects overall SVGs by leveraging a graph signal processing approach. Starting from a KNN graph of spots, SpaGFT applies the graph Fourier transform (GFT) to decompose each gene’s expression levels on this graph into components of varying spatial frequencies, analogous to decomposing a signal into sine and cosine waves. While the components are common to all genes, each gene has unique coefficients associated with these components. Low-frequency components capture spatially organized patterns, while high-frequency components primarily represent noise. The classification of low- and high-frequency domains is based on the eigenvalues derived from the GFT, with thresholds determined at two in-

flexion points of the eigenvalue curve, identified using the Keedle algorithm [151]. For each gene, SpaGFT calculates a GFT score, which is a weighted sum of the gene’s coefficients for all components, where lower-frequency components are assigned larger weights. This GFT score reflects the gene’s overall concentration in low-frequency components. Genes with high GFT scores are considered to exhibit substantial spatial variation and are potential SVGs. To identify SVGs, SpaGFT employs two thresholds: a GFT score cutoff and a p-value cutoff. The GFT score cutoff is determined by ranking genes in descending order of their GFT scores and identifying the inflection point of the GFT score curve across all genes. The p-value calculated by SpaGFT for each gene is not derived from the GFT score; rather, it is based on the Wilcoxon rank-sum test comparing the coefficients of low-frequency and high-frequency components. However, since the ranking of SVGs does not depend on the p-value and the detection threshold is not solely based on the p-value, we do not consider SpaGFT to provide valid statistical inference in our review.

BinSpect, part of the Giotto toolbox [152], detects overall SVGs by examining whether the dichotomized spots (i.e., spots coded as 0 or 1 corresponding to low and high expression levels of a gene, respectively) are randomly distributed in a graph. Specifically, BinSpect constructs a graph of spots either as a KNN graph or by Delaunay triangulation. Then, for each gene, BinSpect follows three steps to determine if the gene is an overall SVG. First, it assigns a binary label to each spot by either thresholding at the top 30% expressed spots or using k -means clustering ($k = 2$). Second, it constructs a 2-by-2 contingency table by counting the edges that connect spots with labels (0,0), (0,1), (1,0), or (1,1). Finally, it applies Fisher’s exact test to the contingency table to decide whether spots with the same label tend to be connected, thereby detecting a gene as an overall SVG.

Note that, unlike Giotto, two other popular toolkits with SRT functionalities—Seurat and squidpy—implement different methods for detecting overall SVGs. Seurat [130] uses two approaches: Moran’s I and a mark-point-process-based method called “markvariogram” similar to Trendsceek [129]. On the other hand, squidpy [134] employs *sepal* [132], a diffusion-process-based method.

scGCO [153] is similar to BinSpect at a high level by testing whether a gene’s discretized

expression levels are dependent on spatial neighborhoods. However, scGCO approaches the problem in a more complex way. After constructing an undirected graph of spots using Delaunay triangulation, scGCO performs three steps. First, it fits a Gaussian mixture model to the gene’s log-transformed expression levels to discretize the gene’s expression into a number of levels. Second, using a graph cut algorithm, it fits a hidden Markov random field to the gene’s log-transformed expression levels on the spot graph to divide the spots into segments. Finally, it tests whether each discretized level is randomly distributed in each segment, using the homogeneous Poisson process as the underlying null model. In summary, a gene is considered an overall SVG if at least one of its discretized levels is non-randomly distributed in any segment.

RayleighSelection [154] is a theoretical approach that generalizes a graph to a simplicial complex [155], incorporating additional higher-dimensional topological information about spatial spots compared to a graph. The method first constructs the Vietoris-Rips complex, a type of simplicial complex, of spatial spots by considering balls of a fixed radius around each spot and forming edges, triangles, and higher-dimensional simplices based on the intersections of these balls. For each gene, it then computes the combinatorial Laplacian score, which measures how well the gene respects the local structure of the simplicial complex. If a gene has high expression levels in highly connected regions, it will have a low combinatorial Laplacian score, indicating a significant spatial pattern and suggesting it to be an overall SVG. Finally, RayleighSelection computes a p-value for the combinatorial Laplacian score using a permutation test.

4.2.3 Relationship between Euclidean-space-based methods and graph-based methods

When it is reasonable to assume data lie on a non-linear manifold in Euclidean space, using a graph to represent the data offers significant advantages. A graph can capture the intrinsic geometry of the manifold by representing data points as nodes and their similarity relationships as edges based on some similarity or distance measure. Graph-construction methods

like KNN (where each node’s neighborhood is defined by a fixed number of neighbors, K) or ϵ -neighborhoods (where each node’s neighborhood is defined by a fixed radius, ϵ) ensure that each node is connected to its nearest neighbors, preserving the local relationships among nodes. The shortest-path distance between two nodes in the graph approximates their “geodesic distance” on the manifold, reflecting the data’s intrinsic structure more accurately than the Euclidean distance. Additionally, graph-based methods are more robust to the curse of dimensionality: in high-dimensional spaces, where Euclidean distances often lose meaning, a graph can effectively capture relationships among data points.

However, in the context of SVG detection, using Euclidean distance to represent distances between spots might be more appropriate. Biological relationships between spots inherently follow Euclidean distances in a physical 2D Euclidean space in tissue. Moreover, since Euclidean distances are based on 2D spatial locations rather than high-dimensional gene expression data, the curse of dimensionality is not an issue.

It remains an open question how to construct a graph to accurately represent a non-linear manifold of spatial spots. The choice of graph-construction method, such as KNN or Delaunay triangulation, is crucial. KNN connects each spot to its K nearest neighbors, preserving the local structure of the data. In contrast, Delaunay triangulation handles the non-uniform spatial distribution of spots in SRT data, such as MERFISH or Slide-seq data. Delaunay triangulation connects spots to form triangles such that no point is inside the circumcircle of any triangle, resulting in a graph that more naturally respects the non-uniform distribution of spots and preserves local density variations. However, Delaunay triangulation is more computationally expensive and more sensitive to outliers than KNN. Future studies could explore whether the computational cost of using Delaunay triangulation for imaging-based SRT data is justified, and how this decision may vary depending on the tissue structure—whether the tissue is structured (such as in the brain) or unstructured (such as in tumors). Additionally, research could focus on improving the scalability and robustness of Delaunay triangulation in these contexts.

Additionally, the definition of edge weights in the graph is another important consideration. Edges can be directional or undirectional, depending on whether the relationship

between spots has a direction (e.g., one cell type influencing another cell type’s gene expression). The edge weights can take multiple forms. For example, weights can be determined by a Gaussian kernel, where weights decay exponentially with the Euclidean distance between spots, emphasizing closer spots. Alternatively, weights can decrease linearly with distance. These choices affect how the graph captures the manifold structure of data and, consequently, how effectively it can be used to detect spatial patterns of gene expression that are biologically relevant.

4.2.4 Applications of overall SVGs

The detection of overall SVGs mainly serves as a pre-processing step in SRT data analysis. While the definition of overall SVGs may vary across methods, the common objective is to select informative genes for downstream analysis. A common downstream analysis is to identify *spatial domains* (also referred to as *spatial communities*) by partitioning a tissue slice into regions so that spots have similar expression profiles of overall SVGs in each region.

Identifying spatial domains can help uncover tissue layers where morphological architecture is less defined. For example, SPADE [139], an overall SVG detection method, uses its detected overall SVGs to identify spatial domains as substructures in cortical layers and amygdala in a 10x Visium mouse brain immunofluorescence dataset. Another use of spatial domain identification is to reveal gene expression profiles underlying tissue structures. For example, scGCO [156], another overall SVG detection method, identifies spatial domains from a mouse breast cancer biopsy dataset sequenced by Spatial Transcriptomics [157], and the identified domains align with annotated tissue structures including invasive ductal cancer, ductal cancer, and normal tissue. This alignment confirms that these tissue structures have distinct gene expression profiles, so the SRT data can potentially lead to the discovery of new marker genes for these tissue structures, a task tackled by spatial-domain-marker SVG detection.

A common approach to identifying spatial domains is clustering spatial spots using overall SVGs’ expression levels and spot locations. For example, graph-based clustering (e.g.,

Louvain clustering) can be applied after spatial spots are connected into a graph based on their spatial proximity and gene expression levels; then, the identified clusters contain spatial spots in proximity and exhibiting similar gene expression levels. For example, using Louvain clustering, the overall SVGs detected by SPADE [139] and SINFONIA [145] resulted in spatial domains in well-structured brain tissues in the mouse olfactory bulb Spatial Transcriptomics dataset [157], as well as in the mouse hippocampus seqFISH [100] and Slide-seq [158] datasets.

The spatial domains identified from overall SVGs can be validated, directly or indirectly, in three ways. First, spatial domains can be compared with tissue layers annotated by pathologists from the H&E image accompanying the SRT data (e.g., 10x Visium). For example, in the SpaGCN [159] method paper, three annotated layers of the primary pancreatic cancer were compared to the spatial domains identified from a Spatial Transcriptomics dataset [160]. Second, spatial domains can be annotated with cell-type labels in external transcriptomic data to verify if distinct domains have different cell-type compositions. For example, in a study of the human dorsolateral prefrontal cortex 10x Visium dataset [114], researchers annotated the identified spatial domains as cortical layers based on the cytoarchitecture and marker genes obtained from external large-scale single nucleus RNA-seq datasets. Third, for well-structured tissue types like the brain, spatial domains can be verified by transferring tissue layer labels from external public annotations. For instance, in the SINFONIA [145] method paper, annotations from Allen Brain Atlas [161] and Mouse Brain Gene Expression Atlas (<http://mousebrain.org/>) were transferred to several SRT datasets, including the mouse brain coronal section 10x Visium dataset, the mouse hippocampus Slide-seqV2 dataset [158], and the mouse olfactory bulb STEREO-seq dataset [112].

Although the detection of overall SVGs may serve as a feature screening step before spatial domain identification to circumvent the curse of dimensionality (i.e., dealing with too many genes as features), it is important to note that this screening step may not be necessary for spatial domain detection methods that employ alternative strategies to reduce feature dimensions. For example, BayesSpace [162] and StLearn [163] implement spatial domain identification using lower-dimensional representations of spatial spots based on selected

HVGs.

Besides spatial domain identification, another downstream analysis is the unsupervised identification of spatial gene modules among the detected overall SVGs. Each module is a cluster of overall SVGs with similar spatial expression patterns, representing a common spatial pattern shared by these genes (also referred to as a co-expression pattern). Compared to the spatial expression patterns of individual overall SVGs, the patterns of gene modules are less noisy and can provide clearer insights into the molecular architecture underlying tissue structures—specifically, which genes co-determine a tissue structure in a synchronized manner. For example, in downstream applications of MULTILAYER [131], MERINGUE [164], and Binspect in Giotto [152], the detected overall SVGs were grouped into modules using hierarchical clustering based on a constructed gene co-expression matrix or a gene-gene similarity graph.

It is worth noting that the identification of spatial gene modules can precede and facilitate the discovery of non-global spatial domains that are specific to one or more gene modules. For example, among the overall SVGs detected by MULTILAYER [131], the genes *ACTA2* and *ELN* were found to be spatially co-expressed, leading to the identification of five tissue substructures in human heart tissue. Similarly, spatial gene modules derived from the overall SVGs detected by MERINGUE [164] enabled the detection of functional spatial regions in mouse brain tissue. Additionally, spatial gene modules identified from the overall SVGs detected by Binspect in Giotto [152] successfully recovered known anatomical structures in mouse kidney tissue.

4.3 Methods for detecting cell-type-specific SVGs

In both sequencing- and imaging-based SRT data, a gene’s expression variance across spots can result from the gene’s different expression levels in distinct cell types. Given that cell types are typically distributed non-uniformly within a tissue slice, neglecting the cell type information of spatial spots may hinder the discovery of genes whose expression patterns reflect informative spatial variation beyond the variation attributed to cell-type composition

[165, 166]. Thus, we define *cell-type-specific SVGs* as the genes that exhibit non-random spatial expression patterns within a cell type. These genes are detected using both SRT data and external cell-type annotations for the spatial spots (See Fig. 4.2a and Table 4.2). Unlike the methods for detecting overall SVGs that do not consider cell type annotations, the three methods for detecting cell-type-specific SVGs—**CTSV** [165], **C-SIDE** [167], and **spVC** [168]—begin by annotating the cell types of the spatial spots in SRT data and then use a regression framework to identify SVGs within cell types by examining the interaction effects between cell types and spatial locations (see Section 4.5.2 for a detailed discussion on the general framework).

CTSV assumes a ZINB distribution for a given gene’s expression count Y_i at spot i :

$$Y_i \stackrel{\text{ind}}{\sim} \text{ZINB}(\mu_i(\mathbf{s}_i), \theta),$$

where $\mu_i(\mathbf{s}_i)$ is the mean expression function of the spatial location $\mathbf{s}_i = (s_{i1}, s_{i2})^\top$, and θ indicates the other parameters (dispersion and zero-inflated probability) necessary to describe the distribution. To determine if the gene is a cell-type-specific SVG, CTSV includes K cell types’ cell-type-level mean functions of $\mathbf{s}_i$: $\eta_k(\mathbf{s}_i)$, $k = 1, \dots, K$, and assumes that

$$\begin{aligned} \log \mu_i(\mathbf{s}_i) &= \log \ell_i + \sum_{k=1}^K \eta_k(\mathbf{s}_i) w_{ik}, \\ \eta_k(\mathbf{s}_i) &= \beta_{k0} + \beta_{k1} b_1(s_{i1}) + \beta_{k2} b_2(s_{i2}). \end{aligned}$$

where ℓ_i denotes spot i ’s library size (i.e., total expression count), and the weights w_{ik} , $k = 1, \dots, K$, are pre-estimated cell type proportions of spot i obtained by spatial deconvolution methods such as SPOTlight [169] and RCTD [166], and $b_1(s_{i1})$ and $b_2(s_{i2})$ are two non-cell-type-specific, non-parametric, univariate functions of the spatial coordinates in the two dimensions. CTSV then tests if β_{k1} and β_{k2} are both zero. If not, the gene is considered an SVG specific to cell type k .

C-SIDE is similar to CTSV but has three major differences. First, C-SIDE assumes a Poisson distribution for Y_i . Second, C-SIDE assumes a different relationship between $\mu_i(\mathbf{s}_i)$

and $\eta_k(\mathbf{s}_i)$, $k = 1, \dots, K$. Third, C-SIDE uses a different non-parametric function form for each $\eta_k(\mathbf{s}_i)$.

$$\begin{aligned} Y_i &\stackrel{\text{ind}}{\sim} \text{Poisson}(\mu_i(\mathbf{s}_i)) , \\ \log(\mu_i(\mathbf{s}_i)) &= \gamma_0 + \log \ell_i + \log \left(\sum_{k=1}^K \eta_k(\mathbf{s}_i) w_{ik} \right) + \epsilon_i , \\ \log(\eta_k(\mathbf{s}_i)) &= \beta_{k0} + \sum_{\ell=1}^L \beta_{k\ell} b_\ell(\mathbf{s}_i) , \end{aligned}$$

where the common terms are defined similarly to CTSV, ϵ_i is a zero-mean random-effect term, and $\sum_{\ell=1}^L \beta_{k\ell} b_\ell(\mathbf{s}_i)$ is a cell-type-specific, bivariate smooth-spline function of $\mathbf{s}_i$ consisting of L basis functions. C-SIDE then tests if $\beta_{k1}, \dots, \beta_{kL}$ are all zero. If not, the gene is considered an SVG specific to cell type k .

spVC is similar to C-SIDE but assumes a different relationship between $\mu_i(\mathbf{s}_i)$ and $\eta_k(\mathbf{s}_i)$, $k = 1, \dots, K$.

$$\log(\mu_i(\mathbf{s}_i)) = \gamma_0 + \log \ell_i + \sum_{k=1}^K w_{ik} \beta_k + \sum_{k=1}^K \eta_k(\mathbf{s}_i) w_{ik} + \gamma(\mathbf{s}_i) ,$$

which considers cell types' non-spatial effects as $\sum_{k=1}^K w_{ik} \beta_k$ and a baseline spatial effect $\gamma(\mathbf{s}_i)$. Then, spVC implements a two-step testing procedure. First, it tests whether $\beta_1, \dots, \beta_K$ are all zero and whether $\gamma(\cdot) = 0$ in a reduced model without the cell-type-specific spatial effect term $\sum_{k=1}^K \eta_k(\mathbf{s}_i) w_{ik}$. If both null hypotheses are rejected, the second step is to test whether $\eta_k(\cdot) = 0$. If not, the gene is considered a SVG specific to cell type k .

4.3.1 Applications of cell-type-specific SVGs

Although the publications of the three cell-type-specific SVG detection methods [165, 167, 168] did not specifically outline applications for cell-type-specific SVGs, we envision potential applications that extend the two major applications of overall SVGs—identifying spatial domains and spatial gene modules—to a more granular, cell-type level. First, cell-type-specific SVGs can be used to identify and characterize distinct cell subpopulations or states

within particular cell types. Second, cell-type-specific SVGs can enable the identification of gene modules within specific cell types. These applications have the potential to provide new molecular insights into tissues with complex structures and diverse cell types, such as cancer and brain tissue. For example, in cancer research, cell-type-specific SVGs can help identify distinct states of cancer cells and immune cells, leading to a better understanding of tumor heterogeneity and the tumor microenvironment. Additionally, cell-type-specific SVGs can facilitate the identification of cell-type-specific gene modules that could be crucial for developing targeted therapies and personalized treatment strategies. Similarly, in brain tissue, cell-type-specific SVGs can identify diverse neuronal subpopulations that may correspond to functional brain regions. Furthermore, cell-type-specific SVGs can enable the discovery of cell-type-specific gene modules that may regulate neuronal activity and connectivity, potentially advancing our understanding of brain development and neurodegenerative diseases.

4.4 Methods for detecting spatial-domain-marker SVGs

We define *spatial-domain-marker SVGs* as the genes that exhibit significantly higher expression in a spatial domain compared to other domains. These genes are detected using SRT data and spatial domains, which are usually detected from the same SRT data (See Fig. 4.2a and Table 4.2). Unlike previous SVG detection methods, the two methods for detecting spatial-domain-marker SVGs—**SpaGCN** [159] and **DESpace** [117]—first partition spatial spots into more than one spatial domain. Then, they implement hypothesis tests to assess a gene’s mean expression differences between these spatial domains. A gene is defined as a marker SVG of a spatial domain if shows significantly higher expression in that domain than in other domains.

SpaGCN identifies spatial domains using a pre-trained graph convolutional network applied to SRT data and the paired H&E image. For each gene, it performs Wilcoxon rank-sum tests on normalized expression levels between each domain and the neighboring spots. If the gene is found to have significantly higher expression in a domain, it is considered a

marker SVG of that domain.

DESpace first implements existing spatial clustering methods, such as BayesSpace [162] and StLearn [170], to SRT data to identify spatial clusters as spatial domains. DESpace then offers two modes for detecting SVGs: the first mode identifies spatial-domain-marker SVGs, while the second mode detects overall SVGs. In the first mode, DESpace uses an NB generalized linear model to assess if the spatial domains have a significant effect on a gene’s expression. If a significant effect is found, the gene is identified as a spatial-domain-marker SVG, associated with the domain where it has significantly higher or lower expression compared to other domains. In the second mode, DESpace tests the null hypothesis that all spatial domains have the same effect on a gene’s expression. If this hypothesis is rejected, indicating that the gene exhibits differences among spatial domains, the gene is detected as an overall SVG. Therefore, the spatial-domain-marker SVGs detected by DESpace must be included among the overall SVGs it identifies. Conversely, the overall SVGs detected by DESpace must serve as marker genes for some spatial domains.

4.4.1 Applications of spatial-domain-marker SVGs

Spatial-domain-marker SVGs have two applications for characterizing predefined spatial domains in a tissue slice, particularly when these domains align with tissue structures. First, spatial-domain-marker SVGs can help elucidate the underlying molecular mechanisms of tissue structures. For example, SpaGCN [159] detected spatial-domain-marker SVGs *KRT17* and *MMP11* in a pancreatic cancer region, which align well with pancreatic cancer biology. *KRT17* functions as a tumor promoter and regulates proliferation in pancreatic cancer, while *MMP11* is a prognostic biomarker for the disease. Second, spatial-domain-marker SVGs can assist in annotating tissue structures in other SRT datasets. For instance, the spatial-domain-marker SVGs identified by SpaGCN [159] from a human dorsolateral prefrontal cortex slice (No. 151673 in the LIBD (Lieber Institute for Brain Development) SRT dataset) were used to annotate the tissue layers in a different brain slice (No. 151507).

4.5 Theoretical characterization of SVG detection methods that use frequentist hypothesis tests

Among the 34 SVG detection methods, 23 of them implement statistical hypothesis tests to detect SVGs using *frequentist inference* (i.e., defining a test statistic, deriving the test statistic’s null distribution, and converting the test statistic value to a p-value). For these frequentist hypothesis-testing-based methods, their null and alternative hypotheses direct their SVG detection. In general, methods that use different null hypotheses are not directly comparable. The reason is that a gene may satisfy one null hypothesis (and is considered a true non-SVG) but not the other null hypothesis (and is defined as a true SVG). Hence, we would like to clarify and categorize the null hypotheses used in the 23 methods to deepen our understanding of these methods’ conceptual similarities and differences.

Based on the types of null hypotheses, in Table 4.1 we summarize the hypothesis tests used in the 23 methods into three types: dependence tests, regression fixed-effect tests, and regression random-effect tests (also known as variance component tests). So far, dependence tests and regression random-effect tests have only been used for overall SVG detection, while regression fixed-effect tests have been used to detect all three categories of SVGs. For each method, besides the test type, we also list the test statistic and null distribution in Supplementary Fig. 4.7. Moreover, we define the three test types in Table 4.2 and include a conceptual diagram in Fig. 4.4 that illustrates the relationships between the three test types and the three SVG categories.

Among the three types of hypothesis tests for SVG detection (Fig. 4.4), dependence tests have the most general null hypothesis: a gene’s expression level is independent of the spatial location. In contrast, the two types of regression tests rely on specific assumptions for a regression model, which has a gene’s expression level as the response variable and the spatial location as the predictor (also known as the covariate or explanatory variable). The co-existence of dependence tests and model-specific regression tests is common in the statistics literature [171]. In general, dependence tests, thanks to their general independence null hypothesis, can capture SVGs with more diverse patterns but can be less powerful for

detecting SVGs of specific patterns, compared to the regression tests. In contrast, relying on specific model assumptions, regression tests are more powerful for discovering the SVGs satisfying the assumptions but, meanwhile, more prone to false discoveries when the model assumptions do not hold [172]. Moreover, an advantage of regression tests is that they can more easily incorporate other covariates for adjustment compared to dependence tests. Additionally, their test statistics typically have theoretical null distributions, making p-value calculation straightforward and efficient. It remains an open question to benchmark the three types of tests regarding robustness to model misspecification and the trade-off between robustness and power. Additionally, it is important to identify which assumptions are more reasonable for SRT data generated from various tissues and by different technologies.

4.5.1 Dependence tests

For a given gene, the most general hypothesis test for SVG detection is to decide whether the gene’s expression level Y is independent of the spatial location $\mathbf{S}$, i.e., the null hypothesis is

$$H_0 : Y \perp \mathbf{S}.$$

In this formulation (Fig. 4.4), we assume that $(y_1, \mathbf{s}_1), \dots, (y_n, \mathbf{s}_n)$ are independently sampled from the joint distribution of $(Y, \mathbf{S})$, where $(y_i, \mathbf{s}_i)$ indicates spot i ’s expression level of the given gene and spatial location. When the independent null hypothesis H_0 does not hold, the gene is a *true overall SVG*.

Out of the 23 frequentist hypothesis-testing-based SVG detection methods, 11 methods adopt the dependent test formulation, constructing a test statistic to summarize the dependence between a gene’s expression level and the spatial location. The 11 methods are SPARK-X, SMASH, singlecellHaystack, Trendsseek, HEARTSVG, Hotspot, MERINGUE, SpaGene, BinSpect, scGCO, and RayleighSelection. We categorize the test statistics used in the 11 methods into two types: test statistics with theoretical null distributions (SPARK-X, SMASH, Hotspot, HEARTSVG, MERINGUE, BinSpect, and scGCO) and test statis-

tics without theoretical null distributions (Trendsceek, singlecellHaystack, RayleighSelection, and SpaGene).

Among the seven methods that use test statistics with theoretical null distributions, SPARK-X is a representative method [126]. The SPARK-X test statistic is defined to capture the “agreement” between two spot similarity matrices (referred to as “covariance matrices” by the SPARK-X authors), one defined based on a gene’s expression levels at the n spots and the other based on the spatial locations of the n spots (subject to a transformation of the locations before defining the similarity between spots, so that the similarity matrix reflects a specific spatial pattern such as the Gaussian or cosine kernel). Specifically, the test statistic is defined as $(1/n)$ times the trace of the product of the two similarity matrices, which is essentially the Pearson correlation between the two vectorized matrices. The rationale is that under the independence null hypothesis, the test statistic should be small and follow a theoretical, asymptotic null distribution as a mixture chi-square distribution.

Four methods use test statistics that lack theoretical null distributions and therefore employ a permutation strategy. They generate the null distribution of a test statistic by permuting a gene’s expression levels across the n spots, thus removing any dependence between the gene’s expression level and the spatial location. The permutation procedure is a general solution for obtaining the null distribution when the theoretical null distribution is difficult to derive, but it is computationally intensive due to the need for repetitive permutations to generate the null distribution. For example, singlecellHaystack [128] defines a test statistic based on the KL divergence of the conditional density of spots’ spatial locations, which is conditional on a gene’s dichotomized expression level (on or off), from the unconditional density of spots’ spatial locations. Specifically, both densities are defined for each spatial grid, which consists of many spots and is pre-defined by the singlecellHaystack algorithm. Then, the test statistic is a summation of the KL-divergence-like statistics across the spatial grids. As this test statistic is complex and has no theoretical null distribution, the permutation is used to generate the null distribution.

4.5.2 Regression tests

There are two types of regression tests (Fig. 4.4): fixed-effect tests, where the effect of the spatial location is assumed to be fixed, and random-effect tests, which assume the effect of the spatial location as random. To explain these two types of tests, we start with a *linear mixed model* for a given gene

$$Y_i = \beta_0 + \mathbf{x}_i^\top \beta + \mathbf{z}_i^\top \gamma + \epsilon_i, \quad (4.4)$$

where the response variable Y_i is the gene's expression level at spot i , $\mathbf{x}_i \in \mathbb{R}^p$ indicates the fixed-effect covariates of spot i , $\mathbf{z}_i \in \mathbb{R}^q$ denotes the random-effect covariates of spot i , and ϵ_i is the random measurement error at spot i with $\mathbb{E}[\epsilon_i] = 0$. In the model parameters, β_0 is the (fixed) intercept, $\beta \in \mathbb{R}^p$ indicates the fixed effects, and $\gamma \in \mathbb{R}^q$ denotes the random effects with zero means $\mathbb{E}[\gamma] = 0$ and the covariance matrix $(\gamma) = \Sigma \in \mathbb{R}^{q \times q}$. In this linear mixed model, independence is assumed between γ and $\epsilon = (\epsilon_1, \dots, \epsilon_n)^\top$ and among $\epsilon_1, \dots, \epsilon_n$.

Fixed-effect tests examine whether some or all of the fixed-effect covariates $\mathbf{x}_i$ contribute to the mean of the response variable $\mathbb{E}[Y_i]$. If all fixed-effect covariates make no contribution, then $\mathbb{E}[Y_i|\mathbf{x}_i] = \mathbb{E}[Y_i]$. The null hypothesis

$$H_0 : \beta = \mathbf{0},$$

implies $\mathbb{E}[Y_i|\mathbf{x}_i] = \mathbb{E}[Y_i]$, $i = 1, \dots, n$.

Random-effect tests examine whether the random-effect covariates contribute to the variance of the response variable (Y_i), focusing on the decomposition

$$(Y_i) = (\mathbb{E}[Y_i|\mathbf{z}_i]) + \mathbb{E}[(Y_i|\mathbf{z}_i)] = \mathbf{z}_i^\top \Sigma \mathbf{z}_i + (\epsilon_i) \quad (4.5)$$

and testing if the contribution of the random-effect covariates ($\mathbb{E}[Y_i|\mathbf{z}_i]$) is zero. The null hypothesis

$$H_0 : \Sigma = \mathbf{0}$$

implies $(\mathbb{E}[Y_i|\mathbf{z}_i]) = 0$, $i = 1, \dots, n$.

To generalize the linear mixed model (Equation (4.4)) for data where the response variable Y_i is not continuous (e.g., binary or count), we can decompose the model into a *random structure*, where each Y_i independently follows a distribution with mean $\mu_i = \mathbb{E}[Y_i]$, and a *systematic structure* that links μ_i to the fixed-effect covariates $\mathbf{x}_i$. Specifically, for the model (Equation (4.4)), if we assume that $\gamma \sim \text{MVN}(\mathbf{0}, \Sigma)$ and $\epsilon_i \sim N(0, \sigma^2)$, then the *random structure* is

$$Y_i \stackrel{\text{ind}}{\sim} N(\mu_i, \mathbf{z}_i^\top \Sigma \mathbf{z}_i + \sigma^2),$$

and the *systematic structure* is

$$\mu_i = \beta_0 + \mathbf{x}_i^\top \beta.$$

The generalization of the linear mixed model can occur in both the random and systematic structures. For simplicity, we only focus on the fixed-effect covariates $\mathbf{x}_i$ for the generalization and omit the random-effect covariates $\mathbf{z}_i$. If we change the random structure from the Gaussian distribution to another exponential-family distribution, we have

$$Y_i \stackrel{\text{ind}}{\sim} \text{Exponential Family}(\mu_i, \phi), \quad (4.6)$$

where ϕ is a nuisance parameter not of primary interest. An example of exponential-family distribution is the Poisson distribution $Y_i \sim \text{Poisson}(\mu_i)$. According to the random structure change, the systematic structure can be written in general as

$$g(\mu_i) = \beta_0 + \mathbf{x}_i^\top \beta, \quad (4.7)$$

where the function $g(\cdot)$ is referred to as the *link function* and specified based on the distribution in the random structure. We refer to the model specified by Equations (4.6)–(4.7) as a *generalized linear model*. If we further generalize the systematic structure so that the

effects of the p covariates in $\mathbf{x}_i = (x_{i1}, \dots, x_{ip})^\top$ on μ_i is non-linear and additive:

$$g(\mu_i) = \beta_0 + \sum_{j=1}^p f_j(x_{ij}), \quad (4.8)$$

we referred to the model specified by Equations (4.6) and (4.8) as a *generalized additive model*.

For regression models with different complexities (roughly speaking, different numbers of parameters to estimate), model selection centers on the bias-variance trade-off. An oversimplified model misses key data characteristics, leading to biased parameter estimates, though with low variance. In contrast, an overly complex model wastes parameters on unimportant details, increasing variance in parameter estimates. As a trade-off, a reasonable model appropriately addresses the data characteristics without overfitting the data noise, thereby being more powerful and robust.

For SVG detection, the effect of the spatial location $\mathbf{s}_i$ on the given gene's expression level Y_i can be formulated as either fixed or random. In the following two subsections, we will discuss the SVG detection methods that adopt each formulation.

Regression fixed-effect tests. Among the 23 frequentist hypothesis-testing-based SVG detection methods, six methods use the regression fixed-effect test formulation: SPADE, C-SIDE, CTSV, spVC, SpaGCN, and DESpace. Notably, unlike dependence tests and regression random-effect tests, only regression fixed-effect tests cover all three SVG categories: overall SVGs, cell-type-specific SVGs, and spatial-domain-marker SVGs. Methods within each category use the same fixed-effect covariates $\mathbf{x}_i$ for spot i , while methods of different categories use different $\mathbf{x}_i$'s. Below, we introduce the six methods in the three categories.

Among the six methods, SPADE [139] is the only method that detects overall SVGs. For a given gene, it uses a linear model without random-effect covariates:

$$\mu_i = \beta_0 + \mathbf{x}_i(\mathbf{s})^\top \beta,$$

where the fixed-effect covariates $\mathbf{x}_i(\mathbf{s})$ are defined as some processed features from the n

spots' spatial locations $\mathbf{s}$ and an H&E image. Specifically, the processed features are based on 512 features from a pre-trained convolutional neural network applied to an H&E image. The gene is a *true overall SVG* if $\beta \neq \mathbf{0}$. SPADE uses the R package limma [173] to fit the linear model and tests if each component of β is zero using a t-test.

For cell-type-specific SVG detection, the fixed-effect covariates of spot i (i.e., $\mathbf{x}_i$) include the library size ℓ_i (i.e., spot i 's total expression count), the cell type $\mathbf{c}_i = (c_{i1}, \dots, c_{iK})^\top \in [0, 1]^K$, and the spatial location $\mathbf{s}_i \in \mathbb{R}^2$ (see Section ?? for definitions). The considered fixed effects include the marginal effects of $\mathbf{c}_i$ and $\mathbf{s}_i$ and their interactive effects. Hence, in a generalized additive model formulation for a given gene, the systematic structure is

$$g(\mu_i) = \beta_0 + \log \ell_i + \sum_{k=1}^K c_{ik} \beta_k + f_0(\mathbf{s}_i) + \sum_{k=1}^K c_{ik} f_k(\mathbf{s}_i), \quad (4.9)$$

where β_0 is the overall intercept, $\log \ell_i$ is the intercept effect of spot i 's library size, β_k indicates the cell-type-specific spatial-invariant effect for cell type k , $f_0(\mathbf{s}_i)$ is the overall spatial effect at spot i , and $f_k(\mathbf{s}_i)$ is the cell-type-specific spatial effect for cell type k at spot i . For identifiability, constraints are needed for $\beta = (\beta_1, \dots, \beta_K)^\top$ and $f_0(\cdot), f_1(\cdot), \dots, f_K(\cdot)$. The gene is a *true cell-type-specific SVG* of cell type k if $f_k(\cdot) \neq 0$.

C-SIDE, CTSV, and spCV are three methods for cell-type-specific SVG detection. As an example method, spVC [168] assumes a gene's expression count Y_i follows a Poisson distribution (i.e., the random structure specified in Equation (4.6)) with the systematic structure (4.9). To decide whether the gene is a cell-type-specific SVG, spVC implements a two-step procedure for sequential hypothesis tests. First, it considers a reduced model without interactive effects between $\mathbf{c}_i$ and $\mathbf{s}_i$, so the systematic structure becomes

$$g(\mu_i) = \beta_0 + \log \ell_i + \sum_{k=1}^K c_{ik} \beta_k + f_0(\mathbf{s}_i).$$

Then it tests two null hypotheses: $H_0 : \beta = (\beta_1, \dots, \beta_K)^\top = \mathbf{0}$ and $H_0 : f_0(\cdot) = 0$ using the likelihood ratio test and the Wald test, respectively. If both null hypotheses are rejected, it proceeds to the second step. Second, it considers the full model with interactive effects, with

the systematic structure (4.9). It tests if any of the interactive effects $f_1(\cdot), \dots, f_K(\cdot)$ are 0 using the Wald test. Similar to spVC, CTSV [165] assumes a generalized additive model. However, CTSV has a different random structure and testing procedure. Specifically, CTSV assumes the random structure to be that Y_i follows a ZINB distribution, and the systematic structure is written as

$$g(\mu_i) = \beta_0 + \log \ell_i + \sum_{k=1}^K c_{ik} \beta_k + \sum_{k=1}^K c_{ik} f_k(\mathbf{s}_i), \quad (4.10)$$

where $f_k(\mathbf{s}_i)$ is further assumed to be additive $f_k(\mathbf{s}_i) = f_{k1}(s_{i1}) + f_{k2}(s_{i2})$. Unlike spVC's two-step testing procedure, CTSV directly tests if any of the interactive effects $f_1(\cdot), \dots, f_K(\cdot)$ are 0 using the Wald test. Same as spVC, C-SIDE [167] assumes the random structure to be Poisson. However, it has a different systematic structure from those of spVC and CTSV, allowing for random effects and additional non-linear transformations. C-SIDE uses a two-sided z-test.

For spatial-domain-marker SVG detection, the fixed-effect covariates of spot i (i.e., $\mathbf{x}_i$) include the library size ℓ_i and the spatial domain, indicated by $\mathbf{d}_i = (d_{i1}, \dots, d_{iL}) \in \{0, 1\}^L$, with $\sum_{l=1}^L d_{il} = 1$ (that is, $d_{il} = 1$ if spot i belongs to domain l). Then the systematic structure becomes

$$g(\mu_i) = \beta_0 + \log \ell_i + \sum_{l=1}^L d_{il} \beta_l, \quad (4.11)$$

where β_l indicates the effect of spatial domain l on $g(\mu_i)$, i.e., the transformed gene i 's true expression level. For identifiability, constraints are needed for $\beta = (\beta_1, \dots, \beta_L)^\top$, e.g., $\beta_1 = 0$. The gene is a *true spatial-domain-marker SVG* (i.e., a DEG across spatial domains) if $\beta \neq \mathbf{0}$.

SpaGCN and DESpace are two methods for spatial-domain-marker SVG detection. Following the systematic structure specified in Equation (4.11), DESpace [117] assumes the gene expression Y_i follows an NB distribution and tests the null hypothesis $H_0 : \beta = \mathbf{0}$ using the likelihood ratio test statistic, which follows an asymptotic chi-square distribution under the null hypothesis. For SpaGCN [159], although it does not explicitly adopt a regression

framework but instead uses the Wilcoxon rank-sum test on normalized gene expression levels (with the library size effect removed) to decide if a gene is significantly more highly expressed in one spatial domain compared to its neighboring spots, the Wilcoxon rank-sum is a score test statistic of the proportional odds ordinal logistic regression model. Hence, the test in SpaGCN can broadly be considered to follow the systematic structure (4.11).

Regression random-effect tests. Unlike the nine methods that adopt dependence tests and the six methods that use regression fixed-effect tests, the last six of the 23 frequentist hypothesis-testing-based methods use regression random-effect tests: SpatialDE, nnSVG, SOMDE, SVCA, SPARK, and GPcounts.

All the six methods generalize the linear mixed model (Equation (4.4)) for each gene, by encoding the spatial location $\mathbf{s}_i$ of spot i as the random-effect covariate $\mathbf{z}_i$. With n spots, $\mathbf{z}_i = (z_{i1}, \dots, z_{in})^\top \in \{0, 1\}^n$ is a binary indicator vector with only one 1 indicating spot i . Without loss of generality, we assume that $z_{ii} = 1$. Then, the corresponding random-effect vector $\gamma = (\gamma_1, \dots, \gamma_n)^\top \in \mathbb{R}^n$ has γ_i indicating the random effect of $\mathbf{s}_i$. The covariance matrix of $\gamma_1, \dots, \gamma_n$, (γ) , is assumed to depend on the spatial proximity of $\mathbf{s}_1, \dots, \mathbf{s}_n$. Accordingly, γ can be explicitly written as $\gamma(\mathbf{s})$. When the null hypothesis $H_0 : (\gamma(\mathbf{s})) = \mathbf{0}$ does not hold, the gene is a *true overall SVG*.

As an early and exemplar method, for each gene SpatialDE [118] assumes a linear random-effect model that only contains $\mathbf{z}_i$ as the random-effect covariates:

$$Y_i = \beta_0 + \mathbf{z}_i^\top \gamma(\mathbf{s}) + \epsilon_i, \quad (4.12)$$

where the random errors $\epsilon_1, \dots, \epsilon_n$ independently follow a Gaussian distribution $N(0, \delta)$, and the random effects $\gamma(\mathbf{s})$ jointly follow a multivariate Gaussian distribution $\text{MVN}(\mathbf{0}, \sigma_s^2 \cdot \mathbf{K}(\mathbf{s}))$, with $\mathbf{K}(\mathbf{s}) = [K(\mathbf{s}_i, \mathbf{s}_j)]_{n \times n}$ specified by a kernel function $K(\cdot, \cdot)$ applied to the n spatial locations. This model is essentially a Gaussian process. The null hypothesis then becomes $H_0 : \sigma_s^2 = 0$.

Two methods, nnSVG [119] and SOMDE [121], also use the Gaussian process model as in SpatialDE but adopt more efficient computational algorithms. The other three methods ex-

tend the Gaussian process model. Specifically, SVCA [122] considers two additional variance terms to accommodate other covariates, such as cell-cell interactions. Instead of assuming that Y_i follows a Gaussian distribution, SPARK [123] and GPcounts [124] assume Poisson and NB distributions, respectively, and generalize the linear mixed effect model (4.12) to generalized linear mixed effect models.

All six methods use the likelihood ratio test with an asymptotic chi-square distribution under the null hypothesis.

4.6 Discussion and future directions

Below, we discuss the comparative advantages and trade-offs of existing SVG detection methods in terms of detection power, specificity, and scalability. Further, we outline two future directions for improving SVG detection methods: (1) accommodating differences among various SRT technologies and tissue types, as well as supporting multi-sample SRT data and (2) enhancing statistical rigor and method validation. Last, we summarize the limitations of existing benchmark studies and provide guidance for future benchmark studies.

4.6.1 Power, specificity, and scalability of SVG detection methods

Among the methods for detecting overall SVGs, the Euclidean-space-based, kernel-based methods define more specific spatial patterns for overall SVGs than the Euclidean-space-based, kernel-free methods and the graph-based methods. The statistical power of these kernel-based methods depends on the alignment between the pre-defined kernels' spatial patterns and the biologically relevant spatial patterns in SRT data. This alignment is often questionable because biologically relevant patterns are rarely as regular as the pre-defined kernel patterns. As a result, kernel-based methods may lose statistical power when this alignment is poor. To address this, some kernel-based methods, such as SPARK [123], incorporate multiple pre-defined kernels to capture a broader diversity of spatial patterns. Despite the potential loss of overall power, kernel-based methods are more specific and powerful for discovering overall SVGs whose spatial expression patterns align with the kernels compared

to methods that do not use kernels. In short, there is a trade-off between overall power and specificity: methods targeting specific spatial patterns are less powerful at discovering other patterns, leading to a loss of overall power, but more powerful at discovering the targeted patterns with higher specificity.

Four types of genes that exhibit interesting but non-global expression patterns can be easily missed by methods that detect overall SVGs. First, some genes of interest may only be highly expressed in small regions of interest (ROIs) and can be overlooked by methods that do not distinguish small ROIs. For such genes, methods that detect spatial-domain-marker SVGs, such as SpaGCN [159], might be more appropriate because these methods can identify small ROIs as spatial domains in the first step before identifying marker genes in the second step. Second, there are genes that exhibit spatial expression patterns within spatial domains. These genes, referred to as *spatial-domain-specific SVGs*, may be a subset of spatial-domain-marker SVGs because, although marker SVGs are highly expressed within a spatial domain, they may not exhibit informative spatial variation within that domain. While no existing methods are specifically designed to detect these spatial-domain-specific SVGs, methods for detecting overall SVGs can likely be applied to specific spatial domains to discover these genes. Identifying spatial-domain-specific SVGs may help uncover spatial subdomains, capturing the internal variation within spatial domains. An alternative approach is to perform clustering within a spatial domain and then identify the resulting subdomain marker genes. However, these genes are not conceptually equivalent to those identified by an overall SVG detection method within the spatial domain, reflecting the same conceptual difference as that between overall SVGs and spatial-domain-marker SVGs. Third, cell-type-specific SVGs might also be easily missed by methods that detect overall SVGs if the cell types of interest have small proportions. To address this, methods for detecting cell-type-specific SVGs have been developed, including CTSV [165], C-SIDE [167], and spVC [168]. These methods rely on regression models, whose goodness-of-fit to SRT data remains to be explored. Ensuring a good fit is essential to avoid spurious discoveries. Fourth, some genes may exhibit sharp expression changes at tissue layer boundaries, which are too local to be detected by methods looking for overall SVGs. Belayer [174] aims to detect such genes by

examining gene expression change rates (gradients) in the 2D space. As a future direction, adding the accompanying H&E image can help refine tissue boundaries, ensuring that the identified genes are interpretable. Moreover, if users have prior knowledge on some interesting genes that should be detected as SVGs, how to incorporate this knowledge into SVG detection to improve the specificity remains an open question.

The scalability of an SVG detection method is determined by the computational time of two steps: calculating a summary statistic for each gene and converting the summary statistic to a p-value based on a null distribution. The second step is only included in methods that provide statistical significance for the detected SVGs in a frequentist manner. In the first step of summary statistic calculation, computational time is measured in terms of n , the number of spatial spots. For example, fitting a Gaussian process takes $O(n^3)$ time in SpatialDE [118] and SPARK [123], but this time is reduced to $O(n)$ by using the nearest-neighbor Gaussian process approximation in nnSVG [119]. By changing the modeling framework from a Gaussian process to a Pearson correlation between two similarity matrices, SPARK-X [126] also achieves a fast computational time of $O(n)$. In the second step of p-value calculation, computational time is fast if the summary statistic is a test statistic with a closed-form theoretical null distribution, as is the case for most methods that use regression tests. However, methods that use test statistics require permutation to calculate the null distribution, making them computationally intensive. To improve the scalability of a method, considerations can be put into expediting both steps. The first step can be accelerated using approximation algorithms. The number of permutations in the second step can be reduced through approaches such as p-value-free FDR control [175], parametric smoothing of the permutation-based null distribution [176], and adaptive strategies that use a large number of permutations only for potentially small p-values [177].

4.6.2 Future direction 1: Accommodating differences in SRT technology and tissue types, and supporting the analysis of multi-sample SRT data

There are two key differences among SRT technologies, yet most current SVG detection methods do not account for these differences in the pre-processing and modeling of SRT data. Below, we introduce these differences and discuss their potential impact on SVG detection.

1. Spatial resolution. Imaging-based SRT measures per transcript’s spatial localization at a single-cell or subcellular resolution, whereas sequencing-based SRT captures data at a multicellular level with relatively coarse resolution. In imaging-based SRT data, detected SVGs might result from the irregular distribution of cell types, such as cancer cells spreading across tumor tissue or mixed cell types at the boundaries of tissue layers. In contrast, the coarse resolution of sequencing-based SRT likely results in smaller variance of gene expression levels across spatial spots compared to imaging-based SRT data. Consequently, the spatial resolution differences between the two types of SRT technologies necessitate different interpretations for the detected SVGs. However, most existing SVG detection methods do not distinguish between the SRT technologies but use the same approach to identify SVGs, indicating a need for improvement.

2. Positional randomness of spatial spots. Different SRT technologies use various strategies to record the positions of spatial spots, resulting in differences in positional randomness. For instance, technologies like Spatial Transcriptomics and 10x Visium capture transcripts in predefined rectilinear or hexagonal grids on micro-slides, leading to structured spatial spots that lack positional randomness. In contrast, technologies like Slide-seq, MERFISH, and SeqFISH capture transcripts wherever they are located, without using predefined grids, leading to unstructured spatial spots with positional randomness. This difference in spatial positional randomness necessitates different modeling strategies for spatial variance. For example, sepal [132] designs distinct spatial modeling frameworks—rectilinear, hexagonal, and unstructured spots—for Spatial Transcriptomics, 10x Visium, and Slide-seq, respectively. MERINGUE [164] and several other graph-based methods use Delaunay triangulation

to account for the nonuniform density of unstructured spots. BOOST-HMI [141] uses the mark interaction model, an extension of the Ising model used in BOOST-MI, to address the irregular distribution of spatial spots for imaging-based SRT data. However, the majority of SVG detection methods have not considered the positional randomness differences among SRT technologies, potentially leading to biased detection of SVGs.

Moreover, the type of tissue plays a crucial role in SVG detection. Whether a tissue is morphologically structured or not should be considered when defining SVGs to ensure they are biologically meaningful. In well-structured tissues, such as the main olfactory bulb or cerebellum, meaningful SVGs are expected to exhibit regular expression patterns that align with the tissue’s morphological structures. Therefore, kernel-based SVG detection methods, which identify SVGs based on regular kernel patterns, may be particularly relevant in these cases. In contrast, for tissues without clear morphological structures, such as tumors or diseased tissues, meaningful SVGs may display irregular patterns. In other words, the expression patterns of SVGs can differ significantly between well-structured and unstructured tissues. Consequently, SVG detection should account for the morphological characteristics of the tissue. Among existing methods, SpaGCN [159] follows this rationale by defining spatial domains through the joint analysis of SRT data and H&E images, encouraging the identified spatial domains to reflect the underlying tissue structures. The SVGs identified by SpaGCN then serve as spatial-domain markers that explain these spatial domains. However, it remains an open question whether incorporating H&E images into overall SVG detection is necessary, especially if overall SVGs are primarily used as pre-screened features for further analysis and computational efficiency is a primary concern. For cell-type-specific SVG detection, H&E images have not yet been incorporated into existing methods. This raises an interesting future research question: how do cell types interact with morphological structures?

Another open question and future direction is the detection of SVGs from multi-sample SRT data, such as consecutive slices of a single tissue or tissue slices from different patients. Existing methods typically detect SVGs from a single tissue slice; however, intuitively, SVGs should be meaningful features that can be compared across multiple tissue slices. The challenge, therefore, is how to effectively identify SVGs from multi-sample SRT data to

enable downstream multi-sample comparisons (e.g., normal vs. diseased tissue). In multi-sample SRT data of well-structured tissues, a common strategy is to align the tissue slices into a common 2D coordinate system [178–181]. This alignment facilitates the detection of SVGs. However, for unstructured tissues (e.g., tumors), alignment may not be feasible or reasonable, necessitating new strategies for SVG detection that can summarize the common, meaningful spatial variations in multi-sample SRT data of unstructured tissues.

4.6.3 Future direction 2: Enhancing statistical rigor and method validation

Similar to single-cell RNA-seq data analysis [182–184], SRT data analysis faces the ”double-dipping” challenge: the same data are analyzed more than once, and the final statistical tests rely on variables inferred from the same data in previous steps, leading to a ”confirmation bias”. This issue is prominent in spatial-domain-marker SVG detection, where the genes involved in identifying spatial domains would inherently be identified as spatial-domain markers in a subsequent step, even if they do not exhibit significant expression changes between spatial domains. Therefore, strategies are needed to remove false-positive discoveries resulting from double-dipping, such as ClusterDE [184].

Moreover, as many SVG detection methods use complex algorithms with implicit assumptions, interpretable ways for sanity checks on the detected SVGs are essential. For example, using *in silico* negative control SRT data can help users test SVG detection methods and identify spurious discoveries. Additionally, fast visualization tools will assist users in interpreting the top detected SVGs, providing a clearer understanding of the results and enhancing the reliability of the findings.

No method can be optimal in every aspect, making method choice for users’ goals crucial. Users should select methods that capture the spatial patterns of interest, ensuring the method’s strengths align with their research needs. Consequently, the diversity of methods is indispensable to cater to various user needs, and method validation and benchmarking are necessary for users to choose the appropriate methods. However, benchmarking is challenging due to the lack of well-annotated datasets with SVG ground truths. As a result, methods

often justify their effectiveness indirectly, which may or may not reflect the biological questions users seek answers for. Early methods like SpatialDE and Trendsceek validate their detected SVGs by visual inspection. Another prevalent strategy is to use synthetic datasets with artificial spatial patterns, such as hotspots, streaks, and rings (Fig. 4.5a), to evaluate methods’ detection power. However, these artificial patterns are oversimplified and might not be biologically relevant (Fig. 4.5b). Hence, benchmarking needs well-annotated SRT datasets from diverse tissues (e.g., structured tissues like the brain and unstructured tissues like tumors) and realistic SRT data simulators [87, 185], not only for comprehensive method validation but also for the potential development of supervised methods. Future research is needed to enhance the realism, comprehensiveness, and scalability of SRT data simulators.

4.6.4 Guidance for future benchmark studies

To date, three benchmark studies have been conducted to evaluate the performance of selected SVG detection methods [38–40]. These studies highlight the complexity of the SVG detection task and the variability in the performance of existing methods. In Supplementary Table S4.4, we summarize the three benchmark studies, detailing their choices of methods, evaluation metrics, and conclusions.

A limitation of the existing benchmark studies is their lack of consideration for the conceptual differences among SVG detection methods (in our categorization: overall SVGs, cell-type-specific SVGs, and spatial-domain-marker SVGs). This limitation can lead to unnecessary comparisons that are essentially like comparing apples to oranges. For instance, two benchmark studies [38, 40] compared overall SVG detection methods (such as SpatialDE and SPARK) with spatial-domain-marker SVG detection methods (such as SpaGCN), using the number of SVGs detected as the evaluation metric. However, these two types of methods have distinct goals: overall SVG detection methods are intended to serve as a preprocessing step for feature selection and are inherently more likely to detect a greater number of genes as SVGs, while spatial-domain-marker SVG detection methods are specific to pre-defined spatial domains and tend to identify fewer genes as spatial domain markers. Conceptually,

the number of detected genes is not a fair metric for comparing these two types of methods. Indeed, this anticipated result was confirmed in both benchmark studies. Had this conceptual categorization been recognized before the benchmark studies were conducted, computational resources could have been conserved by avoiding such comparisons.

A second limitation of the existing benchmark studies is that, although they compared SVG detection methods using diverse methodologies, their conclusions did not adequately summarize or emphasize how different core methodologies (e.g., graph-based or kernel-based approaches) affect the methods’ performance within the benchmarks. This limitation means that the impact of fundamental methodological choices on the effectiveness of SVG detection remains underexplored, potentially limiting the insights that could guide future method development.

A third limitation of the existing benchmark studies is their varied selection of SRT datasets and benchmarking scenarios, leading to evaluation results and conclusions that are not fully comparable. Using limited benchmarking scenarios may favor methods that perform well under specific conditions while overlooking their robustness across a broader range of spatial expression patterns. Consequently, without a consensus on benchmark design, the conclusions drawn from these studies may not provide a comprehensive assessment of the strengths and weaknesses of the methods being compared.

In light of the three identified limitations in existing benchmark studies, we offer three recommendations for future benchmark studies, guided by our conceptual categorization of SVG detection methods. Recognizing that the definitions of SVGs differ among the three conceptual categories (overall SVGs, cell-type-specific SVGs, and spatial-domain-marker SVGs), it is both unfair and uninformative to compare methods across these categories. Therefore, our first recommendation is that future benchmark studies should evaluate methods within each conceptual category, providing more accurate and insightful assessments of SVG detection methods, and better guiding researchers in selecting the most appropriate methods for their specific needs.

Our second recommendation is to benchmark distinct methodologies aimed at achieving

the same detection goal. This approach is valuable for method developers, as it provides a clearer understanding of the sensitivity and specificity of different methodologies, helping to avoid reinventing the wheel or encountering unknown pitfalls in future method development. For example, in the context of overall SVG detection, it would be insightful to compare the performance of dependence tests (with a general independence null hypothesis) against regression tests (with a model-based null hypothesis). Another question worth investigating is whether graph-based methods using Delaunay triangulation, such as MERINGUE and Binspect, can better capture cellular adjacency compared to KNN-based methods like SpaGene and Hotspot in tissues where cells are non-uniformly arranged.

Our third recommendation is to include SRT datasets from diverse tissue types and technologies, along with comprehensive and carefully designed simulations, to ensure broad coverage of informative spatial expression patterns. While this practice is challenging, it is essential to avoid benchmark conclusions that may biasedly favor certain types of SVG detection methods. For instance, if SRT data is simulated based on a specific kernel pattern, benchmark results will inherently favor SVG methods tailored to that kernel. Similarly, if the simulation design includes only a limited range of spatial expression patterns, methods using dependence tests—which operate under a general independence null hypothesis and can detect SVGs with diverse patterns—may be unfairly disfavored. Looking forward, it is essential for the community to collaboratively curate a benchmark database to ensure fair benchmarking and foster future method development.

4.7 Acknowledgements

This chapter is based on my joint work with Shuo Harper Hua, and my Ph.D. advisor Dr. Jingyi Jessica Li. We appreciate the comments and feedback from the members of the Junction of Statistics and Biology at UCLA (<http://jsb.ucla.edu>).

4.8 Figures

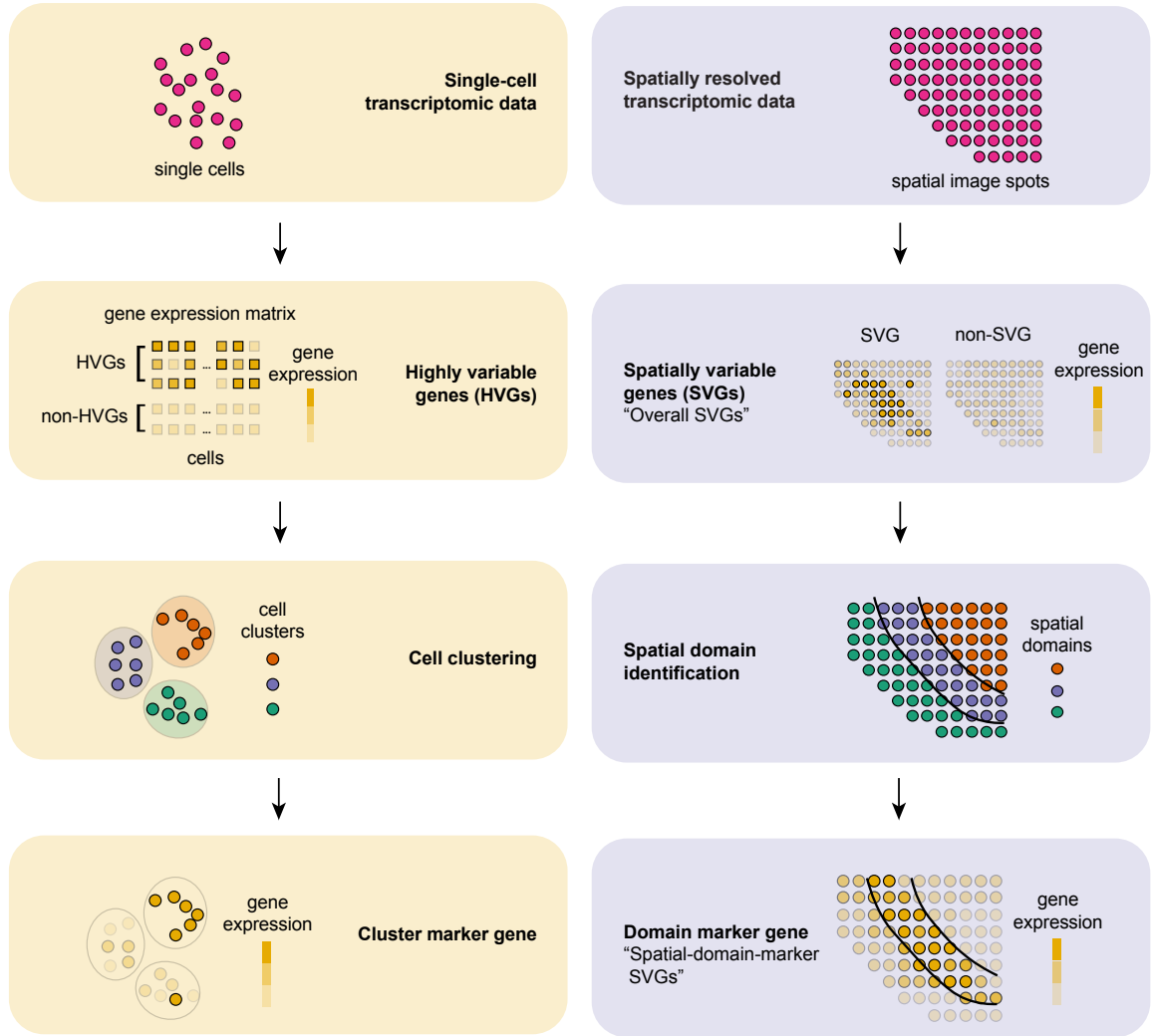


Figure 4.1: General analysis workflows of single-cell transcriptomic data and spatially resolved transcriptomic (SRT) data.

The left column shows a general analysis workflow for single-cell transcriptomic data with steps including highly variable gene (HVG) detection, cell clustering, and cluster-marker gene identification. The right column illustrates a workflow for analyzing SRT data with steps including spatially variable gene (SVG) detection, spatial domain identification, and domain-marker gene identification.

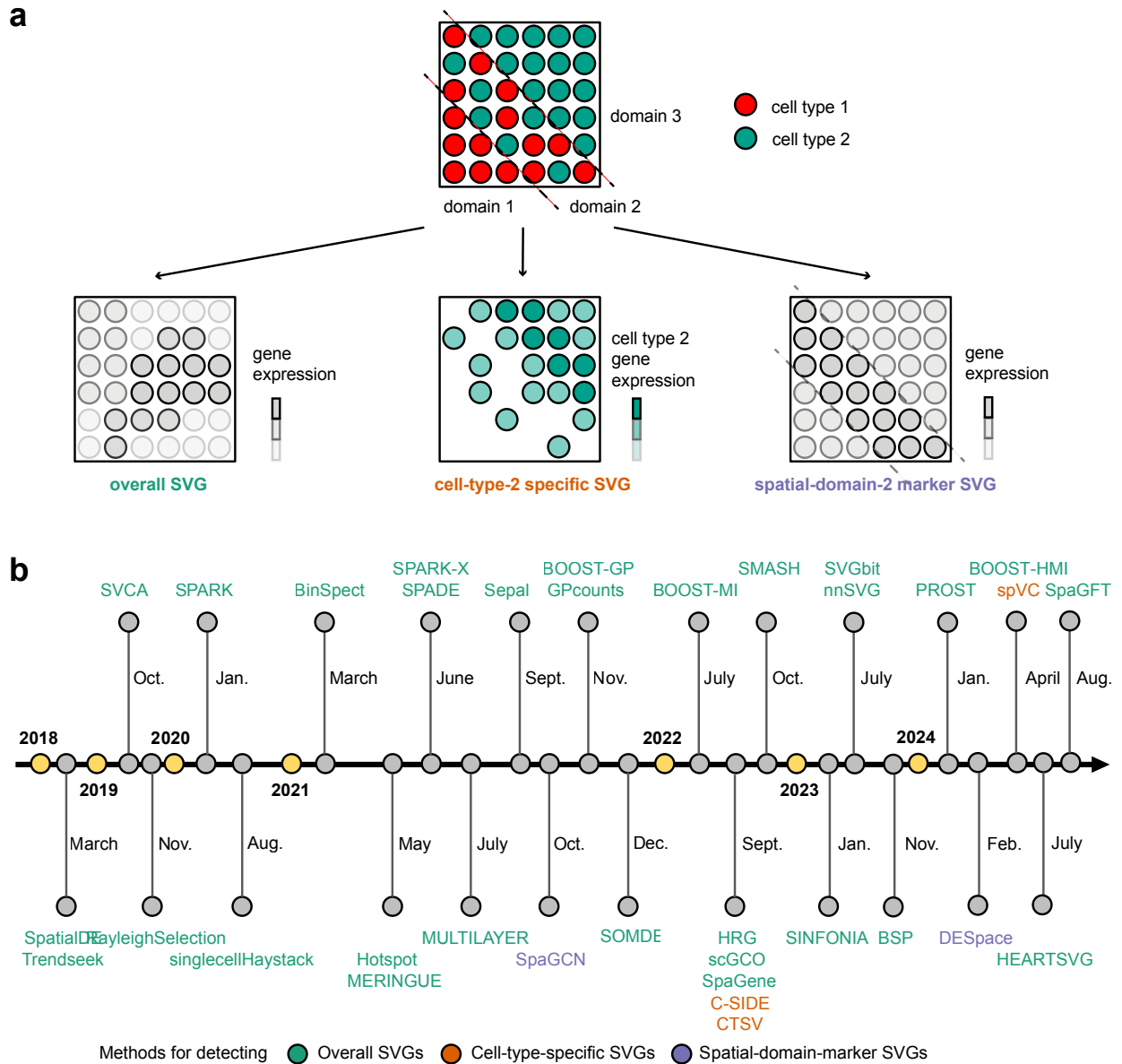


Figure 4.2: Conceptual visualization of three SVG categories and timeline of 34 SVG detection methods.

a, Conceptual visualization of three SVG categories: overall SVGs, cell-type-specific SVGs, and spatial-domain-marker SVGs. The top row shows a tissue slice with two cell types and three spatial domains. From left to right, exemplar genes with colors representing the expression levels are shown for an overall SVG, a cell-type-specific SVG, and a spatial-domain-marker SVG, respectively. **b**, Publication timeline of 34 SVG detection methods. Colors represent three SVG categories: overall SVGs (green), cell-type-specific SVGs (red), and spatial-domain-marker SVGs (purple).

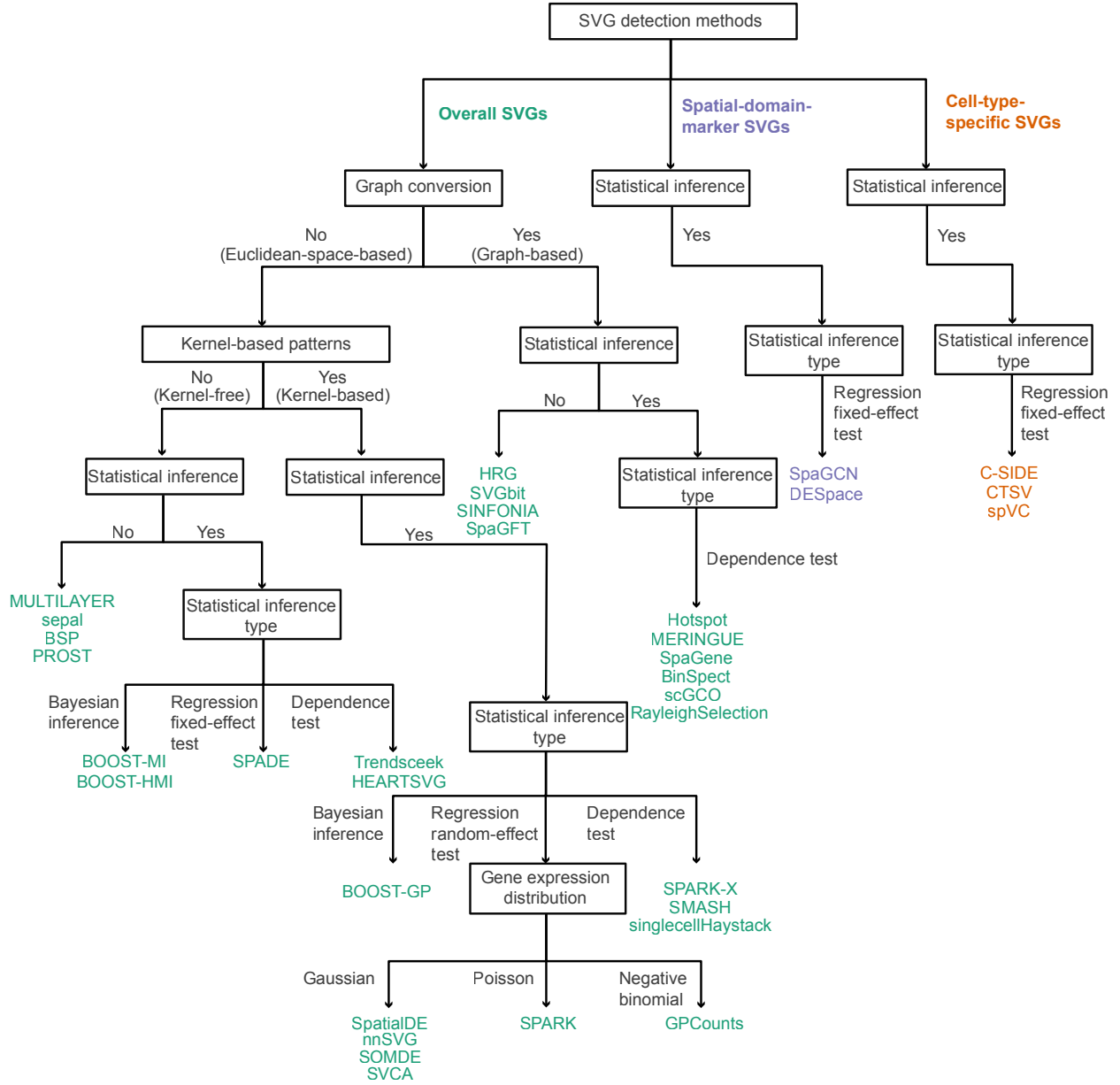


Figure 4.3: A hierarchical summary of 34 SVG detection methods.

The hierarchical summary considers the methodological characteristics, including graph conversion, kernel-based patterns, availability of statistical inference, statistical inference types, and gene expression distributions. Colors represent three SVG categories: overall SVGs (green), cell-type-specific SVGs (red), and spatial-domain-marker SVGs (purple).

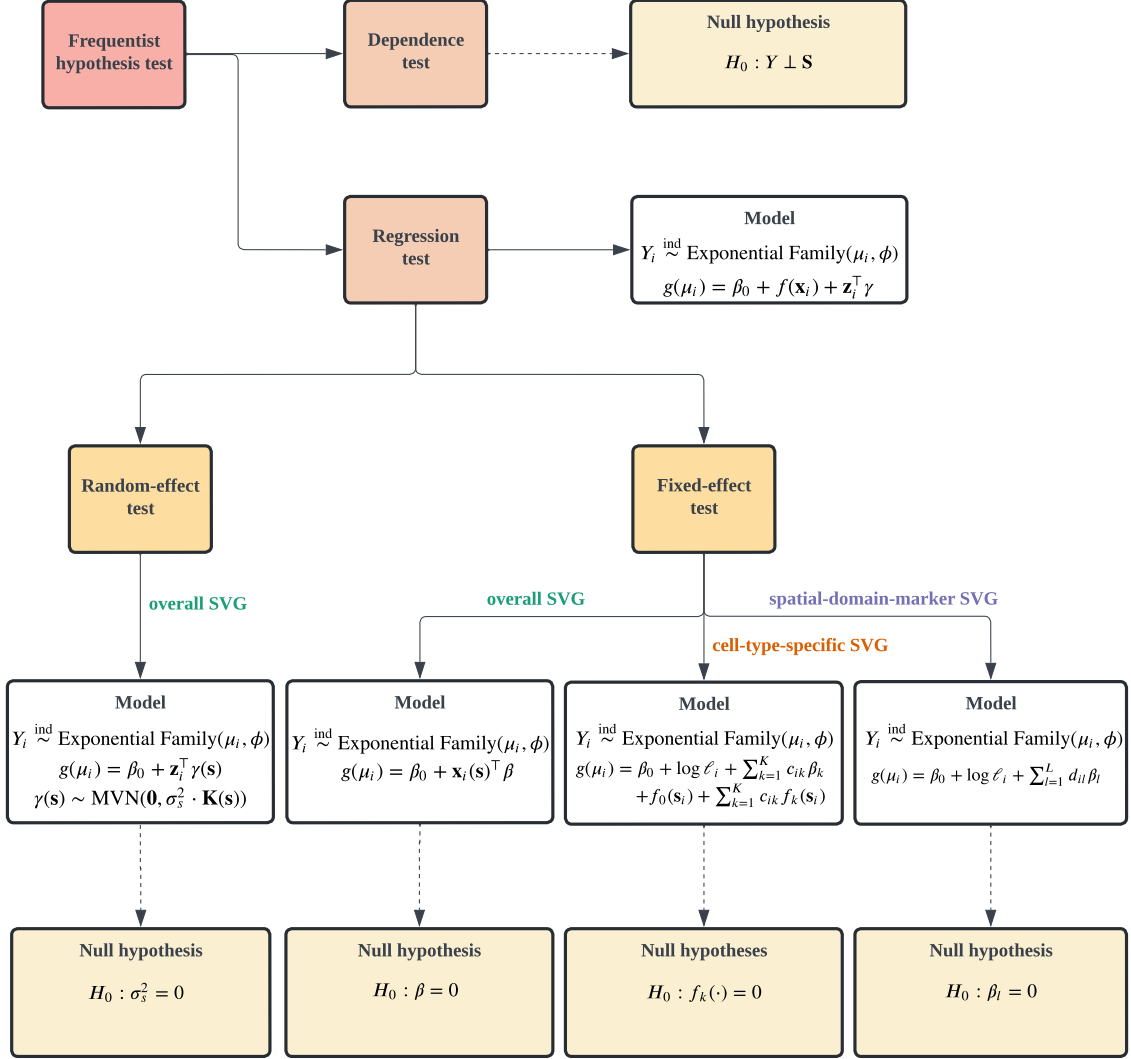


Figure 4.4: Conceptual diagram for Section 4.5 “Theoretical characterization of SVG detection methods that use frequentist hypothesis tests.”

This diagram illustrates the logical relationships among the three types of statistical tests (dependence test, regression fixed-effect test, and regression random-effect test) used by the 23 SVG detection methods that rely on frequentist hypothesis tests. The diagram also introduces the general form of statistical models upon which regression-based tests are performed and the corresponding null hypotheses for detecting SVGs.

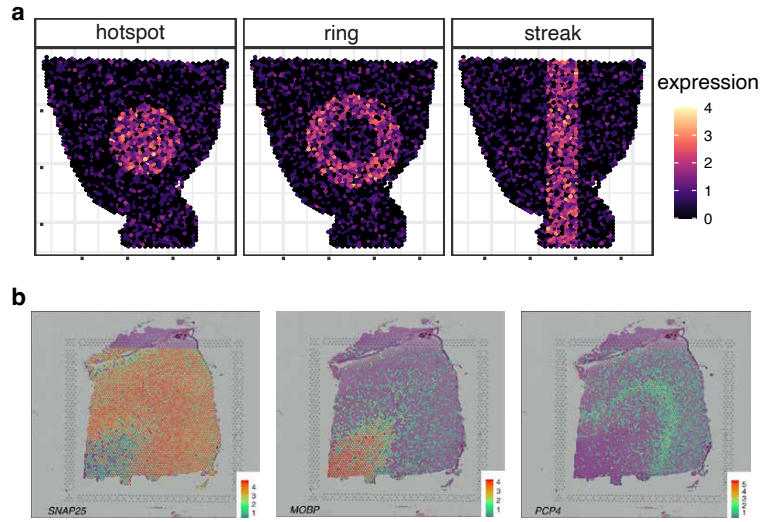


Figure 4.5: Synthetic spatial patterns oversimplify the spatial patterns observed in real SRT data.

a, Representative spatial patterns used in synthetic SRT data for evaluating SVG detection methods such as Trendsceek [129], SpatialDE [118], and others. **b**, Spatial patterns shown in the 10x Visium dataset [114] profiling a dorsal lateral prefrontal cortex sample. The color indicates the log transformed expression for genes *SNAP25*, *MOBP*, and *PCP4*. With permission from Springer Nature.

4.9 Tables

Table 4.1: Classification of 34 SVG detection methods along with statistical inference types

Inference types \ Categories		Overall SVGs	Cell-type-specific SVGs	Spatial-domain-marker SVGs
Frequentist inference	Dependence tests	SPARK-X SMASH singlecellHaystack Trendsceek HEARTSVG Hotspot MERINGUE SpaGene BinSpect scGCO RayleighSelection		
	Regression fixed-effect tests	SPADE	CTSV C-SIDE spVC	SpaGCN DESpace
	Regression random-effect tests	SpatialDE nnSVG SOMDE SVCA SPARK GPcounts		
Bayesian inference		BOOST-GP BOOST-MI BOOST-HMI		
No statistical inference		MULTILAYER sepal BSP ¹ PROST ² HRG SVGbit SINFONIA SpaGFT ³		

¹ Although BSP attempts to perform frequentist inference by defining a null distribution for its test statistic, the null distribution is improperly defined as a distribution fitted to the test statistic values of all genes, implying that all genes are non-SVGs. Hence, we do not label BSP as a method with statistical inference in the table.

² PROST performs frequentist statistical tests for Moran’s I, instead of the PROST index it uses to rank genes as SVGs.

³ SpaGFT does not perform frequentist statistical tests for the GFT scores it uses to rank genes as SVGs.

Table 4.2: Definitions of concepts for SVGs and frequentist statistical tests

	Concept	Definition
SVGs	Overall SVGs	Genes that exhibit non-random spatial expression patterns. This is the most general category of SVGs detected using only SRT data without incorporating external information such as spatial domains or cell types.
	Cell-type-specific SVGs	Genes that exhibit non-random spatial expression patterns within a cell type. These genes are detected using both SRT data and external cell-type annotations for the spatial spots.
	Spatial-domain-marker SVGs	Genes that exhibit significantly higher expression in a spatial domain compared to other domains. These genes are detected using SRT data and spatial domains, which are usually detected from the same SRT data.
Frequentist Tests	Dependent Test	Examines the dependence between a gene's expression level and the spatial location.
	Fixed-effect Tests	Examines whether some or all of the fixed-effect covariates $\mathbf{x}_i$ contribute to the mean of the response variable, i.e., a gene's expression.
	Random-effect Tests	Examines whether the random-effect covariates $\mathbf{z}_i$ contribute to the variance of the response variable, i.e., a gene's expression.

Table 4.3: Mathematical notations and interpretations

Symbol	Interpretation
$n \in \mathbb{Z}^+$	Number of spatial spots
$y_i \in \mathbb{R}$	Expression level of a gene at spot i
$Y_i \in \mathbb{R}$	Random variable representing the expression level of a gene at spot i
$\mathbf{Y} = (Y_1, \dots, Y_n)^\top \in \mathbb{R}^n$	Random vector of a gene's expression levels at all n spatial spots
$\mathbf{s}_i = (s_{i1}, s_{i2})^\top \in \mathbb{R}^2$	2D spatial location of spot i
$\mathbf{s} = [\mathbf{s}_1, \dots, \mathbf{s}_n]^\top \in \mathbb{R}^{n \times 2}$	Matrix of spatial coordinates for all n spots
$\mathbf{d}_i = (d_{i1}, \dots, d_{iL})^\top \in \{0, 1\}^L$	Spatial-domain indicator vector for spot i
$\mathbf{c}_i = (c_{i1}, \dots, c_{iK})^\top \in [0, 1]^K$	Cell-type proportion vector for spot i
$\mathbf{x}_i \in \mathbb{R}^p$	Fixed-effect covariates of spot i
$\mathbf{z}_i \in \mathbb{R}^q$	Random-effect covariates of spot i
$\epsilon_i \in \mathbb{R}$	Random measurement error at spot i
$\boldsymbol{\epsilon} = (\epsilon_1, \dots, \epsilon_n)^\top \in \mathbb{R}^n$	Vector of random measurement errors with assumed independence
$\beta_0 \in \mathbb{R}$	(Fixed) intercept in the linear mixed model
$\boldsymbol{\beta} \in \mathbb{R}^p$	Fixed effects in the linear mixed model
$\boldsymbol{\gamma} \in \mathbb{R}^q$	Random effects in the linear mixed model
$\boldsymbol{\Sigma} \in \mathbb{R}^{q \times q}$	Covariance matrix of random effects $\boldsymbol{\gamma}$

4.10 Supplementary materials

S4.10.1 Supplementary figures

Method	Category	Statistical framework			Count modeling	Statistical inference (frequentist or Bayesian hypothesis tests)
		Graph-based	Euclidean-space-based			
			Kernel-based	Kernel-free		
SpatialDE	Overall SVG		✓			✓ (Frequentist)
nnSVG	Overall SVG		✓			✓ (Frequentist)
SOMDE	Overall SVG		✓			✓ (Frequentist)
SVCA	Overall SVG		✓			✓ (Frequentist)
SPARK	Overall SVG		✓		✓	✓ (Frequentist)
GPcounts	Overall SVG		✓		✓	✓ (Frequentist)
BOOST-GP	Overall SVG		✓		✓	✓ (Bayesian)
SPARK-X	Overall SVG		✓			✓ (Frequentist)
SMASH	Overall SVG		✓			✓ (Frequentist)
singlecellHaystack	Overall SVG		✓			✓ (Frequentist)
Trendsceek	Overall SVG			✓		✓ (Frequentist)
MULTILAYER	Overall SVG			✓		
sepal	Overall SVG			✓		
BSP	Overall SVG			✓		
PROST	Overall SVG			✓		
HEARTSVG	Overall SVG			✓	✓	✓ (Frequentist)
SPADE	Overall SVG			✓		✓ (Frequentist)
BOOST-MI	Overall SVG			✓		✓ (Bayesian)
BOOST-HMI	Overall SVG			✓	✓	✓ (Bayesian)
Hotspot	Overall SVG	✓				✓ (Frequentist)
HRG	Overall SVG	✓				
SVGbit	Overall SVG	✓				
SINFONIA	Overall SVG	✓				
MERINGUE	Overall SVG	✓				✓ (Frequentist)
SpaGene	Overall SVG	✓				✓ (Frequentist)
SpaGFT	Overall SVG	✓				✓ (Frequentist)
BinSpect	Overall SVG	✓				
scGCO	Overall SVG	✓				✓ (Frequentist)
RayleighSelection	Overall SVG	✓				✓ (Frequentist)
CTSV	Cell-type-specific SVG			✓	✓	✓ (Frequentist)
C-SIDE	Cell-type-specific SVG			✓	✓	✓ (Frequentist)
spVC	Cell-type-specific SVG			✓	✓	✓ (Frequentist)
SpaGCN	Spatial-domain-marker SVG			✓		✓ (Frequentist)
DESpace	Spatial-domain-marker SVG			✓	✓	✓ (Frequentist)

Figure 4.6: Methodological traits of 34 SVG detection methods.

Method	Gene expression modeling	Spatial location modeling	Hypothesis test type	Null hypothesis	Test statistics	Null distribution	
SpatialDE	Gaussian distribution	Gaussian process	Regression random-effect test	Spatial covariance has no contribution to gene expression variance	LRT statistic	Chi-square	
nnSVG							
SOMDE							
SVCA							
SPARK	Poisson / Gaussian distribution						
GPcounts	Negative binomial distribution						
SPARK-X	Spot-spot similarity		Dependence test	Spatial location and gene expression are independent	Pearson correlation of spot-spot location similarity matrix and gene expression similarity matrix	Chi-square	
SMASH					KL divergence	Permutation	
singlecellHaystack	Dichotomization	Nonparametric distribution					
Trendsceek	Marked point process						Mark-segregation summary statistics
HEARTSVG	Four 1D time series converted from a 2D slice by row or column averaging			No autocorrelation in all four time series	Portmanteau test statistic (per time series)	Chi-square	
SPADE	Gaussian distribution	Feature extracted by CNN	Regression fixed-effect test	Spatial location has no effect on gene expression	t-statistic	Student's t-distribution	
Hotspot	Autocorrelation statistic			A gene's expression counts at spots independently follow negative binomial or Bernoulli distribution	z-score	Standard normal	
MERINGUE				Moran's I			A gene's expression levels at spots have no spatial autocorrelations
SpaGene	Dichotomization	Node degree distribution in a KNN graph		Spatial location and gene expression are independent	Earth mover's distance	Permutation	
BinSpect		Edges in a graph			Fisher's exact test statistic	Hypergeometric distribution	
scGCO	Gaussian mixture model (GMM) & Hidden Markov random field (HMRF)	Hidden Markov random field			Proportion of spots with a particular level (from GMM) within a spatial segment (from HMRF)	Homogeneous Poisson process	
RayleighSelection	Combinatorial Laplacian score for a simplicial complex				Combinatorial Laplacian score	Permutation	
CTSV	Zero-inflated negative binomial	Fixed-effect covariate (spatial location) in a generalized additive model		Regression fixed-effect test	Spatial location has no effect on cell-type-specific gene expression	Wald statistic	Chi-square
C-SIDE	Poisson					z-score	Standard normal
spVC	Poisson		LRT statistic and Wald statistic			Chi-square	
SpaGCN	Graph convolutional network		No differential gene expression between spatial domains		Wilcoxon rank sum	Standard normal	
DESpace	Negative binomial	Fixed-effect covariate (spatial domain) in a generalized linear model			LRT statistic	Chi-square	

Figure 4.7: Characteristics of 23 frequentist-hypothesis-tests-based SVG detection methods.

The 23 SVG detection methods using frequentist statistical hypothesis tests (Supplementary Fig. 4.6) are included in this table.

S4.10.2 Supplementary tables

Table S4.4: Summary of three existing benchmark studies for SVG detection methods¹

	Charitakis <i>et al.</i> , 2023 [1]	Chen <i>et al.</i> , 2024 [2]	Li <i>et al.</i> , 2023 [3]
SVG Detection Methods	1. SpatialDE 2. SPARK-X 3. Squidpy 4. Seurat (“markvariogram”) 5. SpaGCN 6. scGCO	1. SpatialDE 2. SPARK-X 3. SOMDE 4. Giotto 5. nnSVG 6. MERINGUE 7. Seurat (“moransi”)	1. Moran’s I 2. Spanve 3. scGCO 4. SpaGCN 5. SpaGFT 6. Sepal 7. SpatialDE 8. SpatialDE2 9. SPARK 10. SPARK-X 11. BOOST-GP 12. GPcounts 13. nnSVG 14. SOMDE
Evaluating Metrics	1. Number of SVGs 2. Concordance of SVG rankings 3. Gene ontology enrichment analysis 4. Sensitivity and specificity	1. Concordance of SVG rankings 2. Dependency of SVG statistics on gene expression levels 3. Reproducibility with down-sampling 4. Sensitivity and specificity 5. Impact on spatial domain detection 6. Computational time and memory usage	1. AUPRC 2. Sensitivity and specificity 3. Statistical calibration 4. Impact on spatial domain detection 5. Computational time and memory usage 6. Applicability to other spatial modalities
Conclusions	1. Discrepancies in SVG detection 2. Need for more simulation tools and benchmark workflows	1. Moderate-to-high concordance in SVG rankings, variability in SVGs 2. SVG detection depends on gene expression 3. Affected by gene and spatial spot numbers 4. Estimated FDRs differ from true FDRs 5. 900–1300 SVGs optimize spatial clustering 6. SPARK-X and SOMDE are fastest and most memory-efficient	1. Moran’s I: competitive in AUPRC and calibration 2. No method excels in both sensitivity and specificity 3. SVGs outperform HVGs in spatial domain identification 4. SOMDE: most efficient in memory and speed 5. Need specialized methods for spatial chromatin accessibility data

¹ FDR (false discovery rate); AUPRC (area under the precision-recall curve).

CHAPTER 5

Summary and future directions

This dissertation focuses on improving statistical rigor in single-cell and spatial transcriptomics data analysis through the development of simulation frameworks and a conceptual review. It introduces two computational tools—scReadSim and scIsoSim—for generating synthetic sequencing reads with known ground truths, and a review of methods for detecting spatially variable genes (SVGs) from spatial transcriptomics (SRT) data. Together, these efforts support the evaluation, development, and comparison of computational methods in the field.

5.1 scReadSim: a single-cell RNA-seq and ATAC-seq read simulator

Chapter 2 presented scReadSim, a simulator designed to generate realistic single-cell RNA-seq (scRNA-seq) and ATAC-seq (scATAC-seq) reads. By mimicking the structure and complexity of real sequencing data, scReadSim enables robust benchmarking of read-level bioinformatics tools while providing ground-truth gene-level outputs, such as UMI count matrices for scRNA-seq and user-defined open chromatin regions for scATAC-seq. The simulator also allows users to vary parameters like the number of cells and sequencing depth.

Although the current version of scReadSim focuses on gene-level ground truths and assumes discrete cell types, it is readily extensible to newer count simulation frameworks such as scDesign3, which supports continuous cell trajectories, spatial coordinates, and multiple experimental conditions. A key limitation of scReadSim is its inability to generate isoform-level reads or simulate alternative splicing events, which limits its utility for benchmarking

isoform quantification or splicing analysis tools. This limitation motivated the development of scIsoSim, a dedicated simulator designed to address isoform- and splicing-level benchmarking needs. Future work on scReadSim will aim to incorporate long-read simulation, and the ability to simulate multi-sample data from a single input using distributions learned from public datasets.

5.2 scIsoSim: simulating single-cell RNA-seq reads with full-length isoform ground truths

Chapter 3 introduces scIsoSim, the first single-cell simulator capable of generating next-generation sequencing (NGS) reads with predefined isoform structures and abundances. This feature fills a critical gap in existing simulation tools and enables systematic evaluation of isoform quantification and alternative splicing (AS) detection methods in scRNA-seq. scIsoSim supports major protocols, including Smart-seq2 and 10x Genomics 3' and 5' platforms, and allows researchers to simulate protocol-specific biases and assess tool performance under realistic conditions.

While scIsoSim is currently designed for short-read data, it can be extended to model long-read single-cell RNA sequencing, which enables direct observation of full-length transcripts and is increasingly used in isoform studies. Another application of scIsoSim is to study how sequencing protocols influence AS detection outcomes. For example, researchers can simulate identical isoform compositions across 10x 3', 10x 5', and Smart-seq2 protocols to compare the detection performance of tools such as SpliZ. Moreover, future development of scIsoSim could incorporate dropout modeling and technical noise to assess method robustness across varying data quality and coverage. Expanding its capacity to simulate non-coding RNAs and regulatory transcripts would further broaden its applications, supporting the study of transcriptome complexity and gene regulation in single-cell settings.

5.3 Categorization of 34 computational methods to detect spatially variable genes from spatially resolved transcriptomics data

Chapter 4 presents a review of 34 published methods for detecting spatially variable genes from spatial transcriptomics data. The methods are organized into three categories based on the definition of SVGs: overall SVGs, cell-type-specific SVGs, and spatial-domain-marker SVGs. This classification clarifies the biological goals and methodological assumptions of each group. The review discusses trade-offs across methods in terms of statistical power, specificity, and computational efficiency. For example, kernel-based approaches offer high specificity for predefined spatial patterns but may lose power when those patterns do not match the actual spatial structure in the data. Graph-based and model-free methods tend to be more flexible but can be harder to interpret. The review also identifies types of spatial patterns—such as small expression regions and sharp boundaries—that are often missed by existing methods. Given the increasing diversity of SRT technologies and tissue types, future methods will need to account for differences in spatial resolution, spot arrangement, and morphological structure, and should be able to support analyses across multiple samples or experimental conditions.

To strengthen statistical validity, future research should address common issues such as double use of data when detecting spatial domains and identifying marker genes. Approaches like ClusterDE may offer guidance for addressing this problem. In addition, using simulated negative control datasets and effective visualization tools will help prevent false discoveries and improve interpretation. Benchmarking remains critical for evaluating method performance, but current studies often lack a clear structure and compare methods with different goals. Future benchmarks should focus on comparing methods within the same SVG category and evaluate core methodological differences, such as kernel-based versus graph-based designs. Including a variety of datasets and simulation scenarios will improve fairness and support more appropriate method selection.

Bibliography

- [1] Diego Adhemar Jaitin et al. “Massively parallel single-cell RNA-seq for marker-free decomposition of tissues into cell types”. In: *Science* 343.6172 (2014), pp. 776–779.
- [2] Yukie Kashima et al. “Single-Cell Sequencing Techniques from Individual to Multi-omics Analyses”. In: *Experimental & Molecular Medicine* 52.9 (2020), pp. 1419–1427.
- [3] SK Longo et al. “Integrating single-cell and spatial transcriptomics to elucidate intercellular tissue dynamics.” In: *Nat Rev Genet* 22 (Oct. 2021), pp. 627–644.
- [4] Vivien Marx. “Method of the Year: spatially resolved transcriptomics”. In: *Nature Methods* 18.1 (2021), pp. 9–14.
- [5] Luke Zappia and Fabian J Theis. “Over 1000 tools reveal trends in the single-cell RNA-seq analysis landscape”. In: *Genome Biology* 22.1 (2021), pp. 1–18.
- [6] Mirjana Efremova and Sarah A Teichmann. “Computational methods for single-cell omics across modalities”. In: *Nature Methods* 17.1 (2020), pp. 14–17.
- [7] Luke Zappia, Belinda Phipson, and Alicia Oshlack. “Splatter: simulation of single-cell RNA sequencing data”. In: *Genome Biology* 18.1 (2017), pp. 1–15.
- [8] Davide Risso et al. “A general and flexible method for signal extraction from single-cell RNA-seq data”. In: *Nature Communications* 9.1 (2018), pp. 1–17.
- [9] Wei Vivian Li and Jingyi Jessica Li. “A statistical simulator scDesign for rational scRNA-seq experimental design”. In: *Bioinformatics* 35.14 (2019), pp. i41–i50.
- [10] Tianyi Sun et al. “scDesign2: a transparent simulator that generates high-fidelity single-cell gene expression count data with gene correlations captured”. In: *Genome Biology* 22.1 (2021), pp. 1–37.
- [11] Jinjin Tian, Jiebiao Wang, and Kathryn Roeder. “ESCO: Single Cell Expression Simulation Incorporating Gene Co-Expression”. In: *Bioinformatics* 37.16 (2021), pp. 2374–2381.

- [12] Zeinab Navidi, Lin Zhang, and Bo Wang. “simATAC: a single-cell ATAC-seq simulation framework”. In: *Genome Biology* 22.1 (2021), pp. 1–16.
- [13] Tom Smith, Andreas Heger, and Ian Sudbery. “UMI-tools: Modeling Sequencing Errors in Unique Molecular Identifiers to Improve Quantification Accuracy”. In: *Genome Research* 27.3 (2017), pp. 491–499.
- [14] Benjamin Kaminow, Dinar Yunusov, and Alexander Dobin. “STARsolo: accurate, fast and versatile mapping/quantification of single-cell and single-nucleus RNA-seq data”. In: *bioRxiv* (2021), pp. 2021–05.
- [15] Grace XY Zheng et al. “Massively parallel digital transcriptional profiling of single cells”. In: *Nature Communications* 8.1 (2017), pp. 1–12.
- [16] Avi Srivastava et al. “Alevin efficiently estimates accurate gene abundances from dscRNA-seq data”. In: *Genome Biology* 20.1 (2019), pp. 1–16.
- [17] Yong Zhang et al. “Model-based analysis of ChIP-Seq (MACS)”. In: *Genome Biology* 9.9 (2008), pp. 1–9.
- [18] Sven Heinz et al. “Simple combinations of lineage-determining transcription factors prime cis-regulatory elements required for macrophage and B cell identities”. In: *Molecular cell* 38.4 (2010), pp. 576–589.
- [19] Evan D Tarbell and Tao Liu. “HMMRATAC: a Hidden Markov Modeler for ATAC-seq”. In: *Nucleic Acids Research* 47.16 (2019), e91–e91.
- [20] Michael P Meers, Dan Tenenbaum, and Steven Henikoff. “Peak calling by Sparse Enrichment Analysis for CUT&RUN chromatin profiling”. In: *Epigenetics & chromatin* 12.1 (2019), pp. 1–11.
- [21] Bo Li and Colin N Dewey. “RSEM: accurate transcript quantification from RNA-Seq data with or without a reference genome”. In: *BMC Bioinformatics* 12 (2011), pp. 1–16.
- [22] Nicolas L Bray et al. “Near-optimal probabilistic RNA-seq quantification”. In: *Nature Biotechnology* 34.5 (2016), pp. 525–527.

- [23] Rob Patro et al. “Salmon provides fast and bias-aware quantification of transcript expression”. In: *Nature Methods* 14.4 (2017), pp. 417–419.
- [24] Hirak Sarkar et al. “Terminus enables the discovery of data-driven, robust transcript groups from RNA-seq data”. In: *Bioinformatics* 36.Supplement_1 (2020), pp. i102–i110.
- [25] Cole Trapnell et al. “Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation”. In: *Nature Biotechnology* 28.5 (2010), pp. 511–515.
- [26] Adam Roberts and Lior Pachter. “Streaming fragment assignment for real-time analysis of sequencing experiments”. In: *Nature Methods* 10.1 (2013), pp. 71–73.
- [27] Shihao Shen et al. “rMATS: robust and flexible detection of differential alternative splicing from replicate RNA-Seq data”. In: *Proceedings of the National Academy of Sciences* 111.51 (2014), E5593–E5601.
- [28] Simon Anders, Alejandro Reyes, and Wolfgang Huber. “Detecting differential usage of exons from RNA-seq data”. In: *Nature Precedings* (2012), pp. 1–1.
- [29] Yarden Katz et al. “Analysis and design of RNA sequencing experiments for identifying isoform regulation”. In: *Nature Methods* 7.12 (2010), pp. 1009–1015.
- [30] Jorge Vaquero-Garcia et al. “A new view of transcriptome complexity and regulation through the lens of local splicing variations”. In: *elife* 5 (2016), e11752.
- [31] Yang I Li et al. “Annotation-free quantification of RNA splicing using LeafCutter”. In: *Nature genetics* 50.1 (2018), pp. 151–158.
- [32] Alyssa C Frazee et al. “Polyester: simulating RNA-seq datasets with differential transcript expression”. In: *Bioinformatics* 31.17 (2015), pp. 2778–2784.
- [33] Jennifer Westoby et al. “Simulation-based benchmarking of isoform quantification in single-cell RNA-seq”. In: *Genome Biology* 19 (2018), pp. 1–14.
- [34] Simone Picelli et al. “Full-length RNA-seq from single cells using Smart-seq2”. In: *Nature Protocols* 9.1 (2014), pp. 171–181.

- [35] Lu Pan et al. “Isoform-level quantification for single-cell RNA sequencing”. In: *Bioinformatics* 38.5 (2022), pp. 1287–1294.
- [36] Yuanhua Huang and Guido Sanguinetti. “BRIE2: computational identification of splicing phenotypes from single-cell transcriptomic experiments”. In: *Genome Biology* 22.1 (2021), p. 251.
- [37] Yan Song et al. “Single-cell alternative splicing analysis with expedition reveals splicing dynamics during neuron differentiation”. In: *Molecular cell* 67.1 (2017), pp. 148–161.
- [38] Natalie Charitakis et al. “Disparities in spatially variable gene calling highlight the need for benchmarking spatial transcriptomics methods”. In: *Genome Biology* 24.1 (2023), p. 209.
- [39] Carissa Chen, Hani Jieun Kim, and Pengyi Yang. “Evaluating spatially variable gene detection methods for spatial transcriptomics data”. In: *Genome Biology* 25.1 (2024), p. 18.
- [40] Zhijian Li et al. “Benchmarking computational methods to identify spatially variable genes and peaks”. In: *bioRxiv* (2023), pp. 2023–12.
- [41] Helena L Crowell et al. “Built on sand: the shaky foundations of simulating single-cell RNA sequencing data”. In: *bioRxiv* (2021).
- [42] Hirak Sarkar, Avi Srivastava, and Rob Patro. “Minnow: A Principled Framework for Rapid Simulation of dscRNA-seq Data at the Read Level”. In: *Bioinformatics* 35.14 (2019), pp. i136–i144.
- [43] Wenshan Li and Wei Wang. “Read simulator for single cell RNA sequencing”. In: *UCLA Electronic Theses and Dissertations* (2019).
- [44] Zhanlin Chen et al. “SCAN-ATAC-Sim: a scalable and efficient method for simulating single-cell ATAC-seq data from bulk-tissue experiments”. In: *Bioinformatics* 37.12 (2021), pp. 1756–1758.

- [45] Dongyuan Song et al. “scDesign3 generates realistic in silico data for multimodal single-cell and spatial omics”. In: *Nature Biotechnology* (2023), pp. 1–6.
- [46] Grace X. Y. Zheng et al. “Massively Parallel Digital Transcriptional Profiling of Single Cells”. In: *Nature Communications* 8.1 (2017), p. 14049.
- [47] Evan Z Macosko et al. “Highly parallel genome-wide expression profiling of individual cells using nanoliter droplets”. In: *Cell* 161.5 (2015), pp. 1202–1214.
- [48] Yuhan Hao et al. “Integrated analysis of multimodal single-cell data”. In: *Cell* 184.13 (2021), pp. 3573–3587.
- [49] Robert Salomon et al. “Droplet-Based Single Cell RNAseq Tools: A Practical Guide”. In: *Lab on a Chip* 19.10 (2019), pp. 1706–1727.
- [50] David Sims et al. “CGAT: computational genomics analysis toolkit”. In: *Bioinformatics* 30.9 (2014), pp. 1290–1291.
- [51] Geo Pertea and Mihaela Pertea. “GFF utilities: GffRead and GffCompare”. In: *F1000Research* 9 (2020).
- [52] Jill E Moore et al. “Expanded encyclopaedias of DNA elements in the human and mouse genomes”. In: *Nature* 583.7818 (2020), pp. 699–710.
- [53] Pierre-Luc Germain, Anthony Sonrel, and Mark D Robinson. “pipeComp, a general framework for the evaluation of computational pipelines, reveals performant single cell RNA-seq preprocessing tools”. In: *Genome Biology* 21.1 (2020), pp. 1–28.
- [54] Nan Miles Xi and Jingyi Jessica Li. “Benchmarking computational doublet-detection methods for single-cell RNA sequencing data”. In: *Cell systems* 12.2 (2021), pp. 176–194.
- [55] Nan Miles Xi and Jingyi Jessica Li. “Protocol for executing and benchmarking eight computational doublet-detection methods in single-cell RNA sequencing data analysis”. In: *STAR protocols* 2.3 (2021), p. 100699.
- [56] Pierre-Luc Germain et al. “Doublet identification in single-cell sequencing data using scDblFinder”. In: *F1000Research* 10 (2021).

- [57] Aaron R Quinlan and Ira M Hall. “BEDTools: a flexible suite of utilities for comparing genomic features”. In: *Bioinformatics* 26.6 (2010), pp. 841–842.
- [58] Wei Shen et al. “SeqKit: a cross-platform and ultrafast toolkit for FASTA/Q file manipulation”. In: *PloS one* 11.10 (2016), e0163962.
- [59] Ben Langmead and Steven L Salzberg. “Fast gapped-read alignment with Bowtie 2”. In: *Nature Methods* 9.4 (2012), pp. 357–359.
- [60] Tim Fennell and Nils Homer. *fgbio*. <https://github.com/fulcrumgenomics/fgbio>. 2019.
- [61] Leland McInnes, John Healy, and James Melville. “Umap: Uniform manifold approximation and projection for dimension reduction”. In: *arXiv preprint arXiv:1802.03426* (2018).
- [62] Etienne Becht et al. “Dimensionality reduction for visualizing single-cell data using UMAP”. In: *Nature Biotechnology* 37.1 (2019), pp. 38–44.
- [63] Ilya Korsunsky et al. “Fast, sensitive and accurate integration of single-cell data with Harmony”. In: *Nature Methods* 16.12 (2019), pp. 1289–1296.
- [64] 10xGenomics. *Fresh Embryonic E18 Mouse Brain (5k), Single Cell Multiome ATAC + Gene Expression Dataset by Cell Ranger ARC 2.0.0*. <https://www.10xgenomics.com/resources/datasets/fresh-embryonic-e-18-mouse-brain-5-k-1-standard-2-0-0>. 2019.
- [65] G Marcais and C Kingsford. “Jellyfish: A fast k-mer counter”. In: *Tutorialis e Manuais* 1 (2012), pp. 1–8.
- [66] James T Robinson et al. “Integrative genomics viewer”. In: *Nature Biotechnology* 29.1 (2011), pp. 24–26.
- [67] Fidel Ramírez et al. “deepTools2: a next generation web server for deep-sequencing data analysis”. In: *Nucleic Acids Research* 44.W1 (2016), W160–W165.
- [68] Darren A Cusanovich et al. “A single-cell atlas of in vivo mammalian chromatin accessibility”. In: *Cell* 174.5 (2018), pp. 1309–1324.

- [69] Tiziano Flati et al. “A gene expression atlas for different kinds of stress in the mouse brain”. In: *Scientific Data* 7.1 (2020), pp. 1–18.
- [70] Feng Yan et al. “From reads to insight: a hitchhiker’s guide to ATAC-seq data analysis”. In: *Genome Biology* 21 (2020), pp. 1–16.
- [71] Nadia M Davidson, Anthony DK Hawkins, and Alicia Oshlack. “SuperTranscripts: a data driven reference for analysis and visualisation of transcriptomes”. In: *Genome Biology* 18.1 (2017), pp. 1–10.
- [72] Guanao Yan and Jingyi Jessica Li. *scReadSim: a single-cell RNA-seq and ATAC-seq read simulator*. Version Version 1. Apr. 2023. DOI: [10.5281/zenodo.7824380](https://doi.org/10.5281/zenodo.7824380). URL: <https://doi.org/10.5281/zenodo.7824380>.
- [73] Rob Patro, Stephen M Mount, and Carl Kingsford. “Sailfish enables alignment-free isoform quantification from RNA-seq reads using lightweight algorithms”. In: *Nature Biotechnology* 32.5 (2014), pp. 462–464.
- [74] Pierre-Luc Germain et al. “RNAontheBENCH: computational and empirical resources for benchmarking RNAseq quantification and differential expression methods”. In: *Nucleic Acids Research* 44.11 (2016), pp. 5054–5067.
- [75] Mingxiang Teng et al. “A benchmark for RNA-seq quantification pipelines”. In: *Genome Biology* 17 (2016), pp. 1–12.
- [76] Chi Zhang et al. “Evaluation and comparison of computational tools for RNA-seq isoform quantification”. In: *BMC genomics* 18 (2017), pp. 1–11.
- [77] Minghao Jiang et al. “A comprehensive benchmarking of differential splicing tools for RNA-seq analysis at the event level”. In: *Briefings in Bioinformatics* 24.3 (2023), bbad121.
- [78] Arfa Mehmood et al. “Systematic evaluation of differential splicing tools for RNA-seq studies”. In: *Briefings in bioinformatics* 21.6 (2020), pp. 2052–2065.

- [79] Ruolin Liu, Ann E Loraine, and Julie A Dickerson. “Comparisons of computational methods for differential alternative splicing detection using RNA-seq in plant systems”. In: *BMC Bioinformatics* 15 (2014), pp. 1–16.
- [80] Ángeles Arzalluz-Luque and Ana Conesa. “Single-cell RNAseq for the study of isoforms—how is that possible?” In: *Genome Biology* 19 (2018), pp. 1–19.
- [81] Francisco E Baralle and Jimena Giudice. “Alternative splicing as a regulator of development and tissue identity”. In: *Nature reviews Molecular cell biology* 18.7 (2017), pp. 437–451.
- [82] Dunhui Li et al. “Neurodegenerative diseases: a hotbed for splicing defects and the potential therapies”. In: *Translational neurodegeneration* 10.1 (2021), p. 16.
- [83] Yue You et al. “Benchmarking UMI-based single-cell RNA-seq preprocessing workflows”. In: *Genome Biology* 22.1 (Dec. 2021), p. 339. ISSN: 1474-760X. DOI: [10.1186/s13059-021-02552-3](https://doi.org/10.1186/s13059-021-02552-3). URL: <https://doi.org/10.1186/s13059-021-02552-3> (visited on 01/07/2024).
- [84] Tamar Hashimshony et al. “CEL-Seq: single-cell RNA-Seq by multiplexed linear amplification”. In: *Cell Reports* 2.3 (2012), pp. 666–673.
- [85] Tamar Hashimshony et al. “CEL-Seq2: sensitive highly-multiplexed single-cell RNA-Seq”. In: *Genome Biology* 17 (2016), pp. 1–7.
- [86] Yuanhua Huang and Guido Sanguinetti. “BRIE: transcriptome-wide splicing quantification in single cells”. In: *Genome Biology* 18 (2017), pp. 1–11.
- [87] Dongyuan Song et al. “scDesign3 generates realistic in silico data for multimodal single-cell and spatial omics”. In: *Nature Biotechnology* 42.2 (2024), pp. 247–252.
- [88] Liguang Wang, Shengqin Wang, and Wei Li. “RSeQC: quality control of RNA-seq experiments”. In: *Bioinformatics* 28.16 (2012), pp. 2184–2185.
- [89] Andre Olsson et al. “Single-cell analysis of mixed-lineage states leading to a binary cell fate choice”. In: *Nature* 537.7622 (2016), pp. 698–702.

- [90] Luyi Tian et al. “Benchmarking single cell RNA-sequencing analysis pipelines using mixture control experiments”. In: *Nature Methods* 16.6 (2019), pp. 479–487.
- [91] 10x Genomics. *Human 10x 5’ Single-Cell NSCLC Tumor Dataset*. <https://www.10xgenomics.com/datasets/nsclc-tumor-5-gene-expression-1-standard-2-2-0>. 2018.
- [92] Diego Garrido-Martín et al. “ggsashimi: Sashimi plot revised for browser-and annotation-independent splicing visualization”. In: *PLoS Computational Biology* 14.8 (2018), e1006360.
- [93] Guanao Yan, Dongyuan Song, and Jingyi Jessica Li. “scReadSim: a single-cell RNA-seq and ATAC-seq read simulator”. In: *Nature Communications* 14.1 (2023), p. 7482.
- [94] Yuanhua Huang and Guido Sanguinetti. “Transcriptome-wide splicing quantification in single cells”. In: *bioRxiv* (2017), p. 098517.
- [95] Zhongxiao Li et al. “An isoform-resolution transcriptomic atlas of colorectal cancer from long-read single-cell sequencing”. In: *Cell Genomics* 4.9 (2024).
- [96] Yaqi Su et al. “Comprehensive assessment of mRNA isoform detection methods for long-read sequencing data”. In: *Nature Communications* 15.1 (2024), p. 3972.
- [97] Chi Tian et al. “Single-cell RNA sequencing of peripheral blood links cell-type-specific regulation of splicing to autoimmune and inflammatory diseases”. In: *Nature Genetics* 56.12 (2024), pp. 2739–2752.
- [98] R Ke et al. “In situ sequencing for RNA analysis in preserved tissue and cells.” In: *Nat Methods* 10 (Sept. 2013), pp. 857–60.
- [99] Eric Lubeck et al. “Single-cell in situ RNA profiling by sequential hybridization”. In: *Nature Methods* 11.4 (2014), pp. 360–361.
- [100] Sheel Shah et al. “In situ transcription profiling of single cells reveals spatial organization of cells in the mouse hippocampus”. In: *Neuron* 92.2 (2016), pp. 342–357.
- [101] Chee-Huat Linus Eng et al. “Transcriptome-scale super-resolved imaging in tissues by RNA seqFISH+”. In: *Nature* 568.7751 (2019), pp. 235–239.

- [102] KH Chen et al. *Spatially resolved, highly multiplexed RNA profiling in single cells*. *Science* 348: aaa6090. 2015.
- [103] Xiao Wang et al. “Three-dimensional intact-tissue sequencing of single-cell transcriptional states”. In: *Science* 361.6400 (2018), eaat5691.
- [104] Shahar Alon et al. “Expansion sequencing: Spatially precise in situ transcriptomics in intact biological systems”. In: *Science* 371.6528 (2021), eaax2656.
- [105] Amanda Janesick et al. “High resolution mapping of the tumor microenvironment using integrated single-cell, spatial and in situ analysis”. In: *Nature Communications* 14.1 (2023), p. 8353.
- [106] Meng Zhang et al. “Spatially resolved cell atlas of the mouse primary motor cortex by MERFISH”. In: *Nature* 598.7879 (2021), pp. 137–143.
- [107] Jeffrey R Moffitt et al. “Molecular, spatial, and functional single-cell profiling of the hypothalamic preoptic region”. In: *Science* 362.6416 (2018).
- [108] Jeffrey A Stogsdill et al. “Pyramidal neuron subtype diversity governs microglia states in the neocortex”. In: *Nature* 608.7924 (2022), pp. 750–756.
- [109] 10xGenomics. *Inside Visium spatial capture technology*. https://pages.10xgenomics.com/rs/446-PB0-704/images/10x_BR060_Inside_Visium_Spatial_Technology.pdf. 2019.
- [110] Samuel G Rodriques et al. “Slide-seq: A scalable technology for measuring genome-wide expression at high spatial resolution”. In: *Science* 363.6434 (2019), pp. 1463–1467.
- [111] Daniel R Zollinger et al. “GeoMx™ RNA assay: high multiplex, digital, spatial analysis of RNA in FFPE tissue”. In: *In Situ Hybridization Protocols*. Springer, 2020, pp. 331–345.
- [112] Ao Chen et al. “Spatiotemporal transcriptomic atlas of mouse organogenesis using DNA nanoball-patterned arrays”. In: *Cell* 185.10 (2022), pp. 1777–1792.

- [113] Chun-Seok Cho et al. “Microscopic examination of spatial transcriptome using SeqScope”. In: *Cell* 184.13 (2021), pp. 3559–3572.
- [114] Kristen R Maynard et al. “Transcriptome-scale spatial gene expression in the human dorsolateral prefrontal cortex”. In: *Nature Neuroscience* 24.3 (2021), pp. 425–436.
- [115] Tamim Abdelaal et al. “SIRV: Spatial inference of RNA velocity at the single-cell resolution”. In: *bioRxiv* (2021), pp. 2021–07.
- [116] Sikta Das Adhikari et al. “Recent advances in spatially variable gene detection in spatial transcriptomics”. In: *Computational and Structural Biotechnology Journal* (2024).
- [117] Peiying Cai, Mark D Robinson, and Simone Tiberi. “DESpace: spatially variable gene detection via differential expression testing of spatial clusters”. In: *Bioinformatics* 40.2 (2024), btae027.
- [118] Valentine Svensson, Sarah A Teichmann, and Oliver Stegle. “SpatialDE: identification of spatially variable genes”. In: *Nature Methods* 15.5 (2018), pp. 343–346.
- [119] Lukas M Weber et al. “nnSVG for the scalable identification of spatially variable genes using nearest-neighbor Gaussian processes”. In: *Nature Communications* 14.1 (2023), p. 4059.
- [120] Abhirup Datta et al. “Hierarchical nearest-neighbor Gaussian process models for large geostatistical datasets”. In: *Journal of the American Statistical Association* 111.514 (2016), pp. 800–812.
- [121] Minsheng Hao, Kui Hua, and Xuegong Zhang. “SOMDE: a scalable method for identifying spatially variable genes with self-organizing map”. In: *Bioinformatics* 37.23 (2021), pp. 4392–4398.
- [122] Damien Arno et al. “Modeling cell-cell interactions from spatial molecular data with spatial variance component analysis”. In: *Cell Reports* 29.1 (2019), pp. 202–211.
- [123] Shiquan Sun, Jiaqiang Zhu, and Xiang Zhou. “Statistical analysis of spatial expression patterns for spatially resolved transcriptomic studies”. In: *Nature Methods* 17.2 (2020), pp. 193–200.

- [124] Nuha BinTayyash et al. “Non-Parametric Modelling of Temporal and Spatial Counts Data from RNA-seq Experiments”. In: *Bioinformatics* 37.21 (2021), pp. 3788–3795.
- [125] Qiwei Li et al. “Bayesian Modeling of Spatial Molecular Profiling Data via Gaussian Process”. In: *Bioinformatics* 37.22 (2021), pp. 4129–4136.
- [126] Jiaqiang Zhu, Shiquan Sun, and Xiang Zhou. “SPARK-X: non-parametric modeling enables scalable and robust detection of spatial expression patterns for large spatial transcriptomic studies”. In: *Genome Biology* 22.1 (2021), pp. 1–25.
- [127] Souvik Seal, Benjamin G Bitler, and Debashis Ghosh. “SMASH: Scalable Method for Analyzing Spatial Heterogeneity of genes in spatial transcriptomics data”. In: *PLoS Genetics* 19.10 (2023), e1010983.
- [128] Alexis Vandenbon and Diego Diez. “A clustering-independent method for finding differentially expressed genes in single-cell transcriptome data”. In: *Nature Communications* 11.1 (2020), pp. 1–10.
- [129] Daniel Edsgård, Per Johnsson, and Rickard Sandberg. “Identification of spatial expression trends in single-cell gene expression data”. In: *Nature Methods* 15.5 (2018), pp. 339–342.
- [130] Tim Stuart et al. “Comprehensive Integration of Single-Cell Data”. In: *Cell* 177 (2019), pp. 1888–1902. DOI: [10.1016/j.cell.2019.05.031](https://doi.org/10.1016/j.cell.2019.05.031). URL: <https://doi.org/10.1016/j.cell.2019.05.031>.
- [131] Julien Moehlin et al. “Inferring biologically relevant molecular tissue substructures by agglomerative clustering of digitized spatial transcriptomes with multilayer”. In: *Cell Systems* 12.7 (2021), pp. 694–705.
- [132] Alma Andersson and Joakim Lundeberg. “sepal: identifying transcript profiles with spatial patterns by diffusion-based modeling”. In: *Bioinformatics* 37.17 (2021), pp. 2644–2650.
- [133] Leonid Vitalevich Kantorovich. “Approximate methods of higher analysis”. In: *Inter-science* (1958).

- [134] Giovanni Palla et al. “Squidpy: a scalable framework for spatial omics analysis”. In: *Nature Methods* 19.2 (2022), pp. 171–178.
- [135] Juexin Wang et al. “Dimension-agnostic and granularity-based spatially variable gene identification using BSP”. In: *Nature Communications* 14.1 (2023), p. 7367.
- [136] Yuchen Liang et al. “PROST: quantitative identification of spatially variable genes and domain detection in spatial transcriptomics”. In: *Nature Communications* 15.1 (2024), p. 600.
- [137] Patrick AP Moran. “Notes on continuous stochastic phenomena”. In: *Biometrika* 37.1/2 (1950), pp. 17–23.
- [138] Xin Yuan et al. “HEARTSVG: a fast and accurate method for identifying spatially variable genes in large-scale spatial transcriptomics”. In: *Nature Communications* 15.1 (2024), p. 5700.
- [139] Sungwoo Bae, Hongyoon Choi, and Dong Soo Lee. “Discovery of molecular features underlying the morphological landscape by integrating spatial transcriptomic data with deep features of tissue images”. In: *Nucleic Acids Research* 49.10 (2021), e55–e55.
- [140] Xi Jiang, Guanghua Xiao, and Qiwei Li. “A Bayesian modified Ising model for identifying spatially variable genes from spatial transcriptomics data”. In: *Statistics in Medicine* 41.23 (2022), pp. 4647–4665.
- [141] Jie Yang et al. “Bayesian hidden mark interaction model for detecting spatially variable genes in imaging-based spatially resolved transcriptomics data”. In: *Frontiers in Genetics* 15 (2024), p. 1356709.
- [142] David DeTomaso and Nir Yosef. “Hotspot identifies informative gene modules across modalities of single-cell genomics”. In: *Cell Systems* 12.5 (2021), pp. 446–456.
- [143] Yanhong Wu et al. “Highly Regional Genes: graph-based gene selection for single-cell RNA-seq data”. In: *Journal of Genetics and Genomics* (2022).

- [144] Yingzhou Hong et al. “The spatiotemporal dynamics of spatially variable genes in developing mouse brain revealed by a novel computational scheme”. In: *Cell Death Discovery* 9.1 (2023), p. 264.
- [145] Rui Jiang et al. “SINFONIA: Scalable Identification of Spatially Variable Genes for Deciphering Spatial Domains”. In: *Cells* 12.4 (2023), p. 604.
- [146] Robert C Geary. “The contiguity ratio and statistical mapping”. In: *The incorporated statistician* 5.3 (1954), pp. 115–146.
- [147] Brendan F Miller et al. “Characterizing spatial gene expression heterogeneity in spatially resolved single-cell transcriptomic data with nonuniform cellular densities”. In: *Genome Research* 31.10 (2021), pp. 1843–1855.
- [148] Franco P Preparata and Michael I Shamos. *Computational geometry: an introduction*. Springer Science & Business Media, 2012.
- [149] Qi Liu, Chih-Yuan Hsu, and Yu Shyr. “Scalable and model-free detection of spatial patterns and colocalization”. In: *Genome Research* 32.9 (2022), pp. 1736–1745.
- [150] Yuzhou Chang et al. “Graph Fourier transform for spatial omics representation and analyses of complex organs”. In: *Nature Communications* 15.1 (2024), p. 7467.
- [151] Ville Satopaa et al. “Finding a” kneedle” in a haystack: Detecting knee points in system behavior”. In: *2011 31st international conference on distributed computing systems workshops*. IEEE. 2011, pp. 166–171.
- [152] Ruben Dries et al. “Giotto: a toolbox for integrative analysis and visualization of spatial expression data”. In: *Genome Biology* 22.1 (2021), pp. 1–31.
- [153] Ke Zhang, Wanwan Feng, and Peng Wang. “Identification of spatially variable genes with graph cuts”. In: *Nature Communications* 13.1 (2022), pp. 1–15.
- [154] Kiya W Govek, Venkata S Yamajala, and Pablo G Camara. “Clustering-independent analysis of genomic data using spectral simplicial theory”. In: *PLoS Computational Biology* 15.11 (2019), e1007509.

- [155] Danijela Horak and Jürgen Jost. “Spectra of combinatorial Laplace operators on simplicial complexes”. In: *Advances in Mathematics* 244 (2013), pp. 303–336.
- [156] Ke Zhang, Wanwan Feng, and Peng Wang. “Identification of spatially variable genes with graph cuts”. In: *Nature Communications* 13.1 (2022), pp. 1–15.
- [157] Patrik L Ståhl et al. “Visualization and analysis of gene expression in tissue sections by spatial transcriptomics”. In: *Science* 353.6294 (2016), pp. 78–82.
- [158] Robert R Stickels et al. “Highly sensitive spatial transcriptomics at near-cellular resolution with Slide-seqV2”. In: *Nature Biotechnology* 39.3 (2021), pp. 313–319.
- [159] Jian Hu et al. “SpaGCN: Integrating gene expression, spatial location and histology to identify spatial domains and spatially variable genes by graph convolutional network”. In: *Nature Methods* 18.11 (2021), pp. 1342–1351.
- [160] Wei Guo et al. “Integrating microarray-based spatial transcriptomics and single-cell RNA-sequencing reveals tissue architecture in esophageal squamous cell carcinoma”. In: *EBioMedicine* 84 (2022).
- [161] Susan M Sunkin et al. “Allen Brain Atlas: an integrated spatio-temporal portal for exploring the central nervous system”. In: *Nucleic Acids Research* 41.D1 (2012), pp. D996–D1008.
- [162] Edward Zhao et al. “Spatial transcriptomics at subspot resolution with BayesSpace”. In: *Nature Biotechnology* 39.11 (2021), pp. 1375–1384.
- [163] Duy Pham et al. “stLearn: integrating spatial location, tissue morphology and gene expression to find cell types, cell-cell interactions and spatial trajectories within undissociated tissues”. In: *bioRxiv* (2020), pp. 2020–05.
- [164] Brendan F Miller et al. “Characterizing spatial gene expression heterogeneity in spatially resolved single-cell transcriptomic data with nonuniform cellular densities”. In: *Genome Research* 31.10 (2021), pp. 1843–1855.
- [165] Jinge Yu and Xiangyu Luo. “Identification of cell-type-specific spatially variable genes accounting for excess zeros”. In: *Bioinformatics* 38.17 (2022), pp. 4135–4144.

- [166] Dylan M Cable et al. “Robust decomposition of cell type mixtures in spatial transcriptomics”. In: *Nature Biotechnology* 40.4 (2022), pp. 517–526.
- [167] Dylan M Cable et al. “Cell type-specific inference of differential expression in spatial transcriptomics”. In: *Nature Methods* 19.9 (2022), pp. 1076–1087.
- [168] Shan Yu and Wei Vivian Li. “spVC for the detection and interpretation of spatial gene expression variation”. In: *Genome Biology* 25.1 (2024), p. 103.
- [169] Marc Elosua-Bayes et al. “SPOTlight: seeded NMF regression to deconvolute spatial transcriptomics spots with single-cell transcriptomes”. In: *Nucleic Acids Research* 49.9 (2021), e50–e50.
- [170] Duy Pham et al. “Robust mapping of spatiotemporal trajectories and cell–cell interactions in healthy and diseased tissues”. In: *Nature Communications* 14.1 (2023), p. 7739.
- [171] Jingyi Jessica Li et al. “Dissecting gene expression heterogeneity: generalized Pearson correlation squares and the K-lines clustering algorithm”. In: *Journal of the American Statistical Association* just-accepted (2024), pp. 1–22.
- [172] Yumei Li et al. “Exaggerated false positives by popular differential expression methods when analyzing human population samples”. In: *Genome Biology* 23.1 (2022), p. 79.
- [173] Matthew E Ritchie et al. “limma powers differential expression analyses for RNA-sequencing and microarray studies”. In: *Nucleic Acids Research* 43.7 (2015), e47–e47.
- [174] Cong Ma et al. “Belayr: Modeling discrete and continuous spatial variation in gene expression from spatially resolved transcriptomics”. In: *Cell Systems* 13.10 (2022), pp. 786–797.
- [175] Xinzhou Ge et al. “Clipper: p-value-free FDR control on high-throughput data from two conditions”. In: *Genome Biology* 22 (2021), pp. 1–29.
- [176] Dongyuan Song and Jingyi Jessica Li. “PseudotimeDE: inference of differential gene expression along cell pseudotime with well-calibrated p-values from single-cell RNA sequencing data”. In: *Genome Biology* 22.1 (2021), p. 124.

- [177] Simone Tiberi et al. “distinct: a novel approach to differential distribution analyses”. In: *The Annals of Applied Statistics* 17.2 (2023), pp. 1681–1700.
- [178] Kalen Clifton et al. “STalign: Alignment of spatial transcriptomics data using diffeomorphic metric mapping”. In: *Nature Communications* 14.1 (2023), p. 8123.
- [179] Andrew Jones et al. “Alignment of spatial genomics data using deep Gaussian processes”. In: *Nature Methods* 20.9 (2023), pp. 1379–1387.
- [180] Meng Zhang et al. “Molecularly defined and spatially resolved cell atlas of the whole mouse brain”. In: *Nature* 624.7991 (2023), pp. 343–354.
- [181] Stephan Preibisch, Stephan Saalfeld, and Pavel Tomancak. “Globally optimal stitching of tiled 3D microscopic image acquisitions”. In: *Bioinformatics* 25.11 (2009), pp. 1463–1465.
- [182] Jesse M Zhang, Govinda M Kamath, and N Tse David. “Valid post-clustering differential analysis for single-cell RNA-Seq”. In: *Cell Systems* 9.4 (2019), pp. 383–392.
- [183] Anna Neufeld et al. “Inference after latent variable estimation for single-cell RNA sequencing data”. In: *Biostatistics* 25.1 (2024), pp. 270–287.
- [184] Dongyuan Song et al. “ClusterDE: a post-clustering differential expression (DE) method robust to false-positive inflation caused by double dipping”. In: *Research Square* (2023).
- [185] Jiaqiang Zhu, Lulu Shang, and Xiang Zhou. “SRTsim: spatial pattern preserving simulations for spatially resolved transcriptomics”. In: *Genome Biology* 24.1 (2023), p. 39.