Title
Inference in high dimensions with applications to the analysis of single-cell transcriptomic and bacterial genetic data

Author
Roux de Bezieux, Hector

Publication Date
2021

Peer reviewed|Thesis/dissertation

Inference in high dimensions with applications to the analysis of single-cell transcriptomic and bacterial genetic data

by

Hector Roux de Bézieux

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Biostatistics

and the Designated Emphasis

in

Computational and Genomic Biology

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Sandrine Dudoit, Chair
Associate Professor Elizabeth Purdom
Professor Lin He

Spring 2021

Inference in high dimensions with applications to the analysis of single-cell transcriptomic and bacterial genetic data

Abstract

Inference in high dimensions with applications to the analysis of single-cell transcriptomic and bacterial genetic data

by

Hector Roux de Bézieux

Doctor of Philosophy in Biostatistics

and the Designated Emphasis in Computational and Genomic Biology

University of California, Berkeley

Professor Sandrine Dudoit, Chair

English version

Over the last two decades, technological improvements have led to a tremendous reduction in the cost and speed of DNA sequencing. This has opened the door to many new applications, including the quantification of transcriptomes at the resolution of single cells (scRNA-Seq), and the discovery of genetic features associated with phenotypes of interest, also known as genome-wide association study (GWAS).

scRNA-Seq has emerged in just 10 years as a major tool to investigate biological diversity. The ability to assess gene expression for individual cells enables the study of a range of biological processes at new levels of resolution. This can have many interesting applications, including the identification of novel cell types, defined by their unique transcriptomic signatures. Drawing from the clustering literature, many methods have been developed to group cells together based on their gene expression characteristics. However, all those algorithms require the tuning of hyper-parameters based on typically ad hoc recommendations. Moreover, the direct validation of the discovered cell types is generally difficult, if not impossible. The grouping of cells is also not unique for a given biological system, and there often exists a hierarchy of cell types, with ever-finer levels of resolutions. Because of all this, the discovery of reliable, replicable cell types remains a major challenge. In Chapter 2, we will delve deeper into this issue and introduce a new method called Dune that tackles the resolution-replicability trade-off in clustering.

scRNA-Seq data also enable the tracking of continuous developmental changes, without the need for arbitrary discretization that stemmed purely from the data collection protocol. This allows to investigate processes such as the cell cycle, the differentiation of stem cells

into different cell types, or the cellular response to a drug over time. In Chapter 3, we will investigate how to characterize patterns of gene expression along such developmental trajectories, to identify dynamic genes and drivers of differentiation, using the tradeSeq method. In Chapter 4, we will provide a general workflow called condiments for analyzing such dynamic systems in the presence of multiple conditions, such as treatment/control.

GWAS represent another field that has gained major attention following the emergence of cheaper high-throughput sequencing technologies. In human populations, the problem has been extensively studied, mainly in the context of diseases such as diabetes. However, GWAS can also be applied to bacterial genomes, especially in the context of antibiotic resistance. Some concepts from the human GWAS literature are applicable in bacteria. However, characteristics of bacterial genome mean that other concepts, such as that of a reference genome, are inappropriate and irrelevant. New methods need to be developed for this specific problem. In Chapter 5, we present a new subgraph enumeration method named CALDERA that leverages the structure of the data to provides more robust analyses and facilitate the interpretation of bacterial GWAS data.

Version en français

Au cours des deux dernières décennies, des avancées technologiques ont permis une réduction drastique du coût et du temps nécessaire pour séquencer l'ADN. Ce bouleversement a contribué au développement de nombreuses applications, dont la quantification du transcriptome pour chaque cellule individuelle, et la découverte d'éléments génétiques associés à un phénotype étudié, aussi connu sous le nom d'études d'association pangénomiques.

Depuis sa première itération en 2009, le séquençage de l'ARN en cellule unique s'est rapidement implanté comme un outil majeur pour découvrir les variations au sein d'un système biologique. Mesurer le niveau d'expression des gènes dans chaque cellule séparément permet une compréhension de nombreux processus biologiques à une résolution précédemment inaccessible. Cette résolution permet notamment de découvrir de nouveaux types cellulaires, caractérisés par une signature transcriptomique unique. Tirant partie d'une littérature importante sur la partition des données, de nombreuses méthodes ont été développées pour regrouper les cellules en fonction du niveau d'expression de leurs gènes. Cependant, tous ces algorithmes nécessitent des hyper-paramètres qui doivent être spécifiés, selon des recommandations souvent ad hoc. De plus, la validation de ces nouveaux types cellulaires reste difficile. Enfin, pour de nombreux systèmes biologiques, il existe une hiérarchie de types cellulaires, représentant des niveaux de résolutions de plus en plus fins, plutôt qu'une partition fixe et unique des cellules en groupes distincts et uniques. Pour toutes ces raisons, la découverte de types cellulaires fiables et réplicables est encore un vrai chantier. Dans le chapitre 2, nous explorerons plus en détail ce problème et nous présenterons une nouvelle méthode nommée Dune qui explore le compromis entre résolution et réplicabilité inhérent aux méthodes de partitions des données.

Grâce au séquençage de l'ARN en cellule unique, il est aussi possible de suivre de manière continue les processus de développement, sans avoir besoin de procéder à une discrétisation arbitraire qui découlerait uniquement du procédé de collection des données. Cela permet d'étudier correctement des phénomènes comme le cycle cellulaire, la différentiation de cellules souches en types cellulaires distincts, ou la réponse de cellule à un médicament au cours du temps. Dans le chapitre 3, nous montrerons comment décrire les profils d'expression des gènes le long de ces trajectoires de développement, pour identifier les gènes qui évoluent dynamiquement ou sont moteurs, avec notre méthode tradeSeq. Dans le chapitre 4, nous présenterons une procédure appelée condiments qui permet d'adapter les méthodes d'analyses de ce type de système dynamique en présence de conditions multiples, par exemple un traitement et son contrôle.

L'étude d'association pangénomique est un autre domaine qui a connu un regain d'attention important suite à l'émergence du séquençage d'ADN abordable et à haut débit. Chez l'humain, les recherches se sont concentrées principalement sur l'identification de prédispositions génétiques à des maladies, comme le diabète de type 2. Cependant, ce type d'études peut aussi concerner des populations bactériennes, notamment dans le contexte croissant de la

résistance microbienne aux antibiotiques. Si certaines méthodologies sont directement transposables, d'autres se heurtent aux particularités des génomes bactériens. En particulier, chez les bactéries et microbes, le concept d'un génome de référence est souvent insuffisant ou non pertinent. De nouvelles méthodes appropriées doivent donc être développées. Dans le chapitre 5, nous présenterons une méthode d'énumération de sous-graphes appelée CALDERA qui utilise la structure des données des génomes bactériens pour produire des analyses plus robustes et faciliter l'interprétation des résultats des études d'association pangénomiques bactériennes.

To my parents, who supported me during the Ph.D (and long before), even though they had little idea what I did

Contents

Contents ii

List of Figures iv

List of Tables vii

1 Introduction 1
- 1.1 Next-Generation Sequencing and its applications . . . 1
- 1.2 Analysis of single-cell transcriptomics data (scRNA-Seq) . . . 3
- 1.3 Bacterial Genome-Wide association Studies . . . 7

2 Improving replicability in cell type discovery 10
- 2.1 Introduction . . . 11
- 2.2 Results . . . 12
- 2.3 Discussion . . . 23
- 2.4 Methods . . . 24

3 Trajectory-based differential expression analysis 39
- 3.1 Introduction . . . 40
- 3.2 Results . . . 42
- 3.3 Discussion . . . 55
- 3.4 Methods . . . 56

4 Trajectory inference across multiple conditions 71
- 4.1 Introduction . . . 72
- 4.2 Results . . . 73
- 4.3 Discussion . . . 87
- 4.4 Methods . . . 88

5 Enumeration of Closed Connected Subgraphs 102
- 5.1 Introduction . . . 103
- 5.2 Background on significant subgraph detection using testability . . . 105
- 5.3 Speeding up the detection of all significant CCSs with `CALDERA` . . . 107

5.4 Examples 112
5.5 Experiments 116
5.6 Discussion 118
5.7 Proofs 118

Bibliography **126**

A Supplementary Figures for chapter 2 **144**

B Dune workflow on the baron dataset **149**

C Supplementary Figures and Tables for chapter 3 **167**

D Supplementary Figures for chapter 4 **202**

E Supplementary Figures for chapter 5 **208**

List of Figures

1.1 Schematics of the first steps of a scRNA-Seq analysis . . . 4
1.2 Example of trajectory inference . . . 5
1.3 Simple example of bacterial genome-wide association study data . . . 7

2.1 Measuring and improving the concordance between clusterings. . . . 14
2.2 Illustrating Dune on a dataset with three sets of cluster labels. . . . 16
2.3 Dune stops at a meaningful level. . . . 17
2.4 Comparison of methods. . . . 19
2.5 Dune robustness analysis on simulated datasets. . . . 22
2.6 Dune robustness analysis on real datasets. . . . 36

3.1 Simulation study results. . . . 45
3.2 Benchmark of computation time and memory usage. . . . 47
3.3 Mouse bone marrow case study [108]. . . . 49
3.4 Mouse olfactory epithelium case study [44]. . . . 53
3.5 Tests currently implemented in the tradeSeq package. . . . 59
3.6 Overview of tradeSeq functionality. . . . 64

4.1 Illustrating the first two steps of the condiments workflow with several scenarios 74
4.2 Simulation results. . . . 79
4.3 **TGFB** dataset [92]: Differential topology and differential progression. . . . 82
4.4 **TGFB** dataset [92]: Differential expression. . . . 84
4.5 **KRAS** dataset [175]: Differential topology, differential progression, and differential differentiation. . . . 85

5.1 Intuition for the effectiveness of the new bound . . . 113
5.2 Simple examples where the search in breadth-first is much more efficient that depth-first . . . 115
5.3 Results of CALDERA. . . . 116

A.1 Impact of Seurat's two main tuning parameters on the number of clusters. . . . 144
A.2 Simulated datasets. . . . 145
A.3 Resolution-replicability trade-off for the Pancreas datasets. . . . 146
A.4 Comparison of methods. . . . 147

A.5 Simulation results including the ARI-based version of Dune. 148

C.1 Mouse bone marrow dataset: The NB-GAM is robust to the number of knots k. 172
C.2 Mouse bone marrow dataset: Outlying dendritic cells and eosinophils in UMAP space for TI with Monocle 3. 173
C.3 Bifurcating simulation scenario: GPfates only recovers meaningful trajectories if the true pseudotime is provided as input. 174
C.4 Cyclic simulation scenario: Selecting the optimal number of knots k using the AIC. 175
C.5 Cyclic simulation scenario: Monocle 3 inferred trajectories for each of the 10 simulated datasets. 176
C.6 Cyclic simulation scenario: PCA plots with slingshot inferred trajectory and FDP-TPR performance curves for trajectory-based differential expression analysis for each of the 10 simulated datasets. Monocle 3 errored on three datasets. edgeR_assoc is the edgeR-based version of the `associationTest`. 177
C.7 Simulation study results, including edgeR-based `associationTest` for the cyclic scenario. 178
C.8 Bifurcating simulation scenario: Selecting the optimal number of knots k using the AIC. 179
C.9 Bifurcating simulation scenario: Monocle 2 inferred trajectories for each of the 10 the simulated datasets. 180
C.10 Bifurcating simulation scenario: GPfates inferred trajectories for each of the 10 simulated datasets. 181
C.11 Bifurcating simulation scenario: Mean FDR-TPR performance curves for trajectory-based differential expression analysis across all 10 simulated datasets. 182
C.12 Bifurcating simulation scenario: FDP-TPR performance curves for trajectory-based differential expression analysis for each of the 10 simulated datasets. . . . 183
C.13 Bifurcating simulation scenario: FDP-TPR performance curves for trajectory-based differential expression analysis based on the simulation ground truth. . . . 184
C.14 Multifurcating simulation scenario: GPfates inferred trajectory on one simulated dataset. 185
C.15 Multifurcating simulation scenario: Selecting the optimal number of knots k using the AIC. 185
C.16 Computational time benchmark for the various tests implemented in tradeSeq. . 186
C.17 Mouse bone marrow dataset: Selecting the optimal number of knots k using the AIC. 187
C.18 Mouse bone marrow dataset: tradeSeq recovers markers for the progenitor cell population. 188
C.19 Mouse bone marrow dataset: Gene set enrichment plots for the erythrocyte gene set from Graaf et al. [47], for three differential expression methods: tradeSeq, BEAM, and edgeR. 189
C.20 Mouse bone marrow dataset: Stability of gene clustering methods across bootstrap samples. 190

C.21 Olfactory epithelium dataset: Selecting the optimal number of knots k using the AIC. 191
C.22 Mouse olfactory epithelium dataset: Cell cycle genes in the neuronal lineage . . 192
C.23 Mouse olfactory epithelium dataset: Top six differentially expressed genes as identified by a ZINB analysis with tradeSeq `patternTest`. 193
C.24 Mouse olfactory epithelium dataset: Trajectory with knots. 194
C.25 Mouse olfactory epithelium dataset: UpSet plot for `earlyDETest` applied around the first branching point to each pair among the three lineages. 195
C.26 Adipocyte differentiation dataset: Selecting the optimal number of knots k using the AIC. 196
C.27 Adipoyte differentiaion dataset: Inferred trajectory. 197
C.28 Adipocyte differentiation dataset: Top markers for progenitor cell population. . 198
C.29 Adipocyte differentiation dataset: Genes upregulated in the adipocyte precursor stage and a single differentiated cell type. 199
C.30 Adipocyte differentiation dataset: Genes sporadically upregulated across the entire lineage for a single differentiated cell type. 200
C.31 Mouse bone marrow dataset: UpSet plot for `startVsEndTest`, for multiple assignments of cells to lineages. 201

D.1 Simulation example. 202
D.2 Results on the third type of dataset. 203
D.3 **TCDD** dataset [100]: Differential topology and differential progression. 204
D.4 **TCDD** dataset [100]: Differential expression. 205
D.5 **KRAS** dataset [175]: Differential differentiation. 206
D.6 **KRAS** dataset [175]: Differential expression. 207

E.1 Finite numbers of possible p-values (log scale) for a fixed value of $n1 = 50$ and $x_{\mathcal{S}} = 64$. 208
E.2 Minimum p-value as a function of $x_{\mathcal{S}}$ for fixed values of $n_1 = 25$ and $n = 100$. . 209

List of Tables

2.1 Contingency table for two partitions $\mathbf{P}_1$ and $\mathbf{P}_2$. 25
2.2 Simulation parameters. 33
2.3 Parameters for pre-processing the datasets. 35

3.1 Overview of simulated datasets. 67

4.1 Summary of all case study datasets. 81
4.2 Summary of all simulated datasets. 99

5.1 Association table in community j for subgraph $\mathcal{S}$, used for the CMH test. 106
5.2 Order of exploration of the elements of $\mathcal{C}$ while exploring depth-first 114
5.3 Order of exploration of the elements of $\mathcal{C}$ while exploring breadth-first 114

C.1 Mouse olfactory epithelium dataset, `startVsEndTest` procedure 168
C.2 Mouse olfactory epithelium dataset, `patternTest` procedure. 169
C.3 Mouse olfactory epithelium dataset, `diffEndTest` procedure. 170
C.4 Mouse olfactory epithelium dataset, `earlyDETest` procedure. 171

Acknowledgments

First and foremost, I would like to thank my amazing adviser, Sandrine Dudoit. I would not only say that she made me a better scientist but that she made me a scientist entirely. She taught me intellectual rigor in a way that build upon the simple formality of my earlier school years. Above and beyond, she is a great person to work with, or just talk to about anything.

All throughout my thesis, I have been supported by many mentors. Kelly Street and Koen Van den Berge's names will pop up regularly throughout the next chapters, and most of my publications, and it clearly reflects how essential they have been in creating and advancing the projects. From the first meetings in a small window-less room in Berkeley to zoom call across three countries and presenting a workshop from a moving car and 6 time zones away, we do share a lot of memories!

Part of my doctorate has been supported by Pendulum Therapeutics, and the work presented in chapter 5 is the main example of this. I would like to deeply thank Fanny Perraudeau, who has been my mentor and supervisor over there. She was the first person to ever make truly think about doing a PhD (little regret there) and I believe we both learned a lot from working together.

Laurent Jacob lead (and is still leading) one of the most fun project of my PhD. He was always responsive, and ready to discuss and debates my most far-fetched ideas, and to accept to change an entire paper 5 days before the deadline because I had had a breakthrough at 3am on a Sunday.

Outside those mentors, I had the chance to work with people from many different labs. I'd like to thank Martin Kinisu, Lin He, Ding Yi , Zhenyu Xuan and everyone at the He Lab; Rebecca Barter, John Ngai, Hsinjun Chou, Boying Gong, Elizabeth Purdom, Davide Risso from the brainstat group; Phillipe Boileau, Courtney Schiffman and Partow Imani from the Dudoit Lab; Nima Hejazi, Suzanne Default, Alejandra Benitez, Ivana Malenica, Kevin Benac and all the people from the biostat department; James Bullard, Michael Souza, Paul McMurdie, Christian Sieber and Orville Kolterman and many more from Pendulum Therapeutics.

Five years in the Bay, 10,000 kms from France, means that I received vital support from housemates, friends and family. I'd like to mention in particular Titouan Jehl, Geoffroy Negiar and Romain Lopez for continuous discussions about a range of subjects so wide it would require another thesis to list them all, but also the people I had the chance to live with at the International House, at Ward, Church and California Street, and the friends from everywhere. From grading homework between beers to wrapping up a simulation from the parking lot of a skiing resort, life as a grad student is forever a mix of work and life.

Finally, I'd like to thank my family for the support, both while I was far away, and during the lockdown when I was in France and they had to adjust to my California-based work-schedule. Covid stopped some of them from visiting but I hope to be able to show them the bay some day.

Chapter 1

Introduction

This introduction provides background, context and motivation for the consecutive chapters. The rise of next generation sequencing will be explored and discussed, and its application to both single-cell transcriptomic profiling and genome-wide association studies will be explained. Examples of relevant applications and discoveries that stem from those fields as well as the statistical challenges that are inherent to the type of data collected will be presented.

1.1 Next-Generation Sequencing and its applications

Next-Generation Sequencing

Next-Generation Sequencing methods (NGS) encapsulate a series of technological advances that have occurred since the turn of the century, which have combined to produce a staggering effect. The Human Genome Project, which relied on the Sanger sequencing method [128], i.e., first-generation sequencing, to sequence the entirety of a human genome, cost $100M to finish. Nowadays, modern technologies allow for the sequencing of massive amounts of DNA reads in parallel: the price of sequencing of a human genome had dropped to $20,000 by 2010 and is now around $1,000, a 10^5-fold decrease in the span of two decades [169]. This drastic reduction in cost impacted many aspects of genomics and genetics. Here, we will focus on two: single-cell transcriptomics and genome-wide association studies for bacteria and metagenomes.

Single-cell transcriptomics

Transcriptomics can be defined broadly as the measurement of messenger RNA (mRNA) in tissues and cells, and the downstream analyses that can be conducted using those measurements. Measuring the abundance of mRNA molecules provides insight into gene regulation mechanisms and details about underlying biology [87]. NGS technologies enabled a new type of assays measuring mRNA expression, known as RNA-sequencing (RNA-Seq) [165]. Those assays provided a major improvement over previous micro-array assays [114, 101, 129]:

they can profile the whole transcriptome at once, instead of a set of pre-specified genes, and provide for example accurate quantification of entire human transcriptome. Methods varied between technological platforms [99, 9]. The most common platforms for datasets used here are Illumina [99] and 10X Genomics [183]. Details of how those methods work are outside the scope of this thesis but both companies provide very educational videos on their respective websites.

The first micro-arrays and RNA-Seq assays focused on RNA abundance for large pools of cells such as entire tissues. This bypassed the issue of isolating cells one-by-one and provided valuable insight into tissue level dynamics [171]. However, some biological settings require profiling at the single-cell level, e.g., the study of embryogenesis, where the number of available cells is limited. In 2009, Tang et al. [146] sequenced the transcriptomic profile of five individual mouse cells, and then 34 one year later [147]. A decade later, datasets with over 2M cells have been reported [27, 26, 120]. This explosion in size is once again linked to technological advances, both in micro-fluidic and in mRNA isolation and amplification [73].

Genome-Wide Association Studies

Very generally, Genome-Wide Association Studies (GWAS) encompass a field that looks for genetic features associated with a phenotype of interest. The advent of NGS has allowed one to sequence entire genomes at scale, instead of being limited to only querying a few genes of interest, and assess the genetic diversity of the population. In humans, this whole-genome approach has often focused on single-nucleotide polymorphisms (SNPs) as the main driver of genetic diversity [28]. Visscher et al. [163] offer a review of GWAS successes in humans, listing the discovery of drug candidates for Type 2 diabetes, rheumatoid arthritis, or LDL cholesterol.

Methods tailored for human data have been heavily developed and validated. However, they are often not appropriate for bacterial sequencing data. Indeed, many species are not well-known and lack a reference genome that can be used to then define polymorphisms [51]. Moreover, genetic variations are not restricted to SNPs but can also represent higher level changes such as absence / presence of entire genes that are accessory but can impact fitness under certain conditions [135, 31]. Jaillard et al. [63] show that in *Pseudomonas aeruginosa*, over half of genetic features linked to antibiotic resistance are found in the accessory genome, i.e., the part that is not present in all species. This latter question is especially pressing, given the major increase in antibiotic resistance over the last decade [66, 170].

1.2 Analysis of single-cell transcriptomics data (scRNA-Seq)

Data structure

After pre-processing [107, 19, 183] the output from sequencing machines, the data format that we will use in the following chapters follows a common structure. We represent the gene expression data for n cells from a species whose genome contains G genes as a count matrix $\mathbf{Y}$ with G rows and n columns (Fig 1.1a). The value $y_{g,i}$ of the g^{th} row and i^{th} column is an integer, generally describing the number of reads from the sequencing platform that are assigned to gene g in cell i.

Identification of cell types

In some datasets, the type of each single cell is evident at the sample collection stage. For example, in an embryogenesis dataset [36], the type of each sample can be observed by counting the number of cells in each embryo. However, in most settings, cells are collected in a batch which represent a mix of cell types. This can be the case when an entire organ is collected and sequenced at the single-cell level, for example the liver [10, 130], the brain [149, 176] or the olfactory system [44, 46]. There, cells come from a mixture of various cell types.

The most common way, by far, to identify cell types has been to rely on clustering methods [70, 39]. Broadly defined, clustering aims to partition a dataset in groups of samples such that samples within the same group are more similar to each other than to samples in other groups. Clustering methods are varied but, in the scRNA-Seq context, they usually rely on a low-dimensional representation of the data, which is an n by d matrix $\mathbf{X}$ such that $d \ll G$, but which preserves distances between cells. Note that, in $\mathbf{Y}$, columns correspond to cells, while for $\mathbf{X}$, rows correspond to cells. For example, in Figure 1.1b, we display the data from Deng et al. [36] with $d = 2$, using Principal Components Analysis (PCA). The 258 cells are colored according to their known label. Values of $d = 2$ or $d = 3$ are frequently used for visualisation (Figure 1.1b-c and Figure 1.2a-b) but higher values of d can be used when performing clustering. Different loss functions and search spaces yield different values of $\mathbf{X}$, with non-linear stochastic methods such as t-distributed stochastic neighbor embedding (t-SNE) [159, 160, 75] and uniform manifold approximation and projection (UMAP) [93, 11] more recently replacing PCA [145].

Depending on the mathematical definition of similarity between two samples and two groups, clustering algorithms can yield different results. In the context of cell-type discovery, clustering is used to identify subsets of cells that are defined by a common and separate transcriptomic signature. Clustering methods are widely used in the field. Indeed, using a curated database, Svensson, da Veiga Beltrame, and Pachter [145] show that around 90% of studies using this type of assays performed a clustering step. An example of the result of clustering using the algorithm from Tasic et al. [149] on the 123,155 mouse brain cells of Yao et al. [176] is also displayed in Figure 1.1c. Although the reduced dimensions coordinates

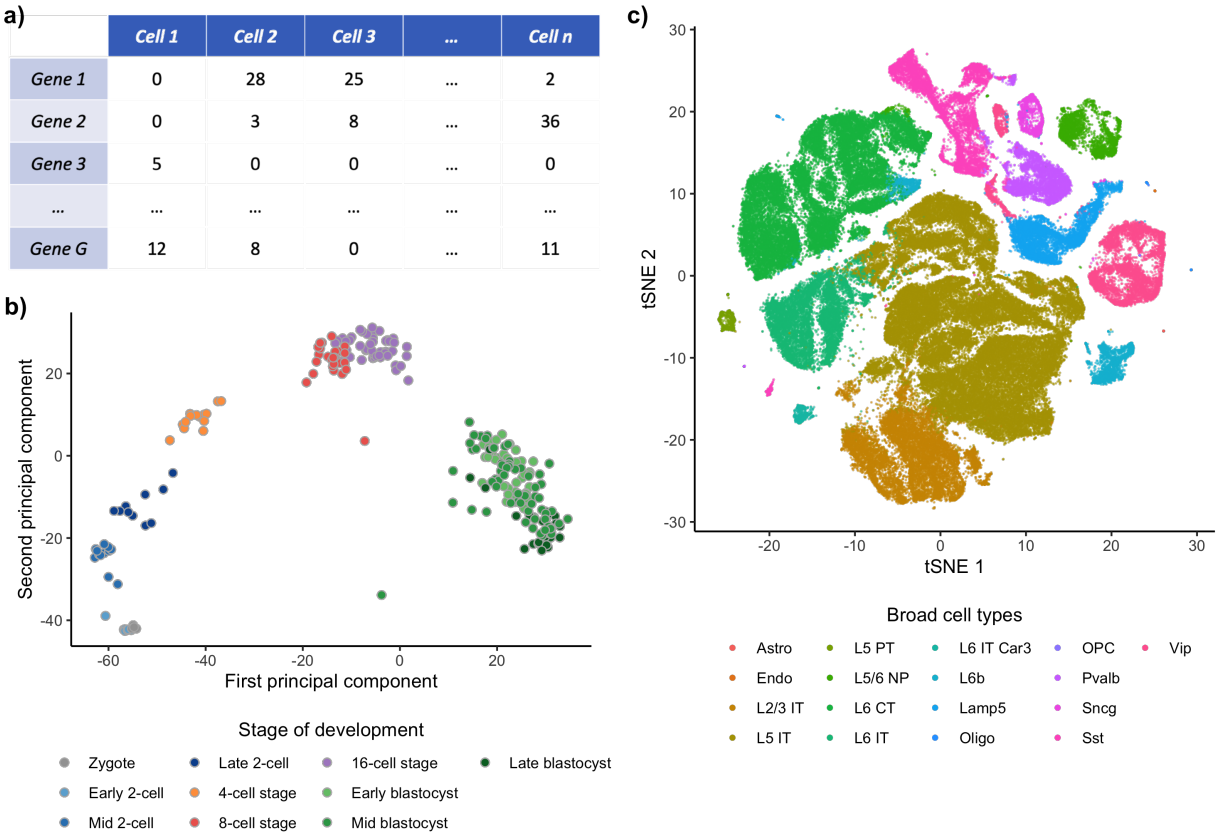


Figure 1.1: *Schematics of the first steps of a scRNA-Seq analysis.* The initial data takes the form of a count matrix with the cells as columns and the genes as rows **(a)**. Using the count matrix, the samples are projected on a reduced-dimensional space, which can be used for visualization and downstream analysis. For example, in **(b)**, the 258 embryo cells from Deng et al. [36] are displayed using the first two principal components. The reduced dimensions can also be used as input to clustering algorithms. For example, in **(c)**, the 123,155 mouse brain cells [176] are clustered [149] and displayed using t-SNE.

shown here are not used for clustering, they are consistent with the results: in that reduced space, cells that are close together are more likely to be clustered (i.e. colored) together.

Dynamic systems

Starting in 2014, Trapnell et al. [154] proposed a new framework to study dynamic biological systems. Clustering usually bins cells into discrete groups, which is usually not appropriate in developmental settings, where the underlying biology is more of a continuum than a

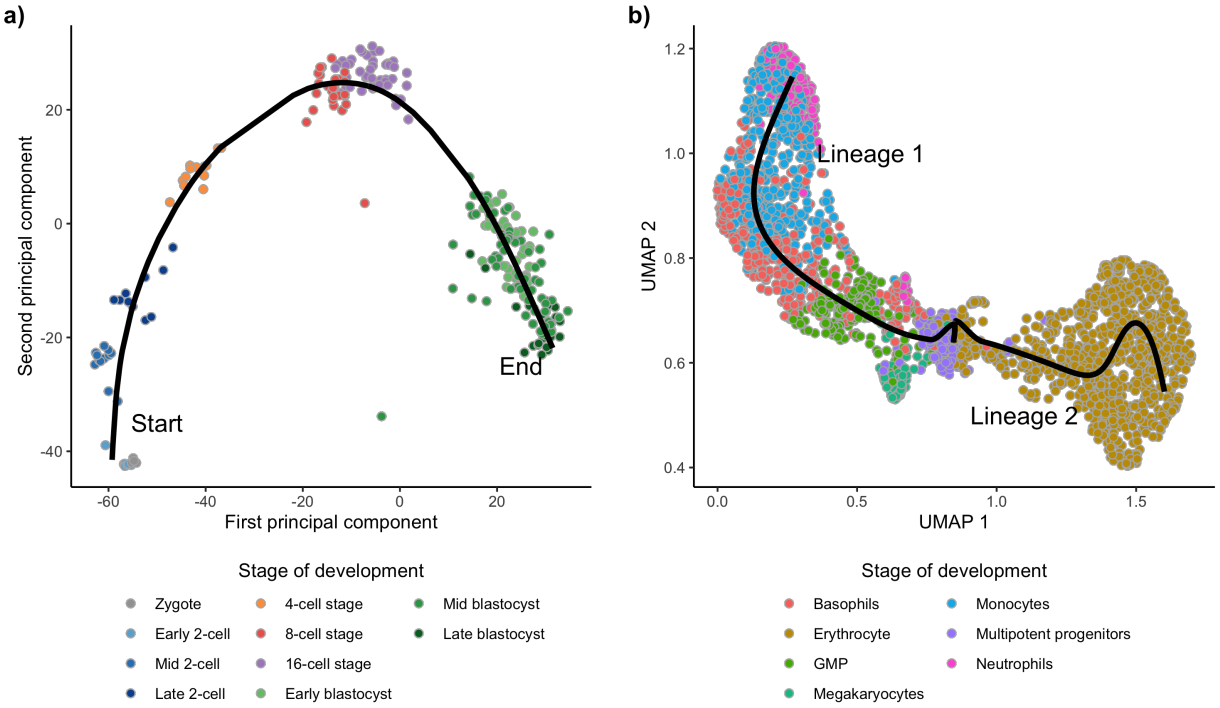


Figure 1.2: *Example of trajectory inference.* We show two datasets, projected in a reduced two-dimensional space. Using slingshot [141], we also infer a trajectory and plot it (black curve). **(a)** The 258 embryo cells from Deng et al. [36] are displayed using the first two principal components and colored by stage of development. Then, a single-lineage trajectory is inferred, starting from the Zygote stage and showing progression until late blastocyst. **(b)** The 2,660 cells from Paul et al. [108] are displayed using UMAP [11, 93] and colored using the cluster labels from the original publication. A branching trajectory with two lineages is inferred, starting from the multipotent progenitor cells. Lineage 1 tracks development of cells into neutrophils while Lineage 2 focuses on erythrocytes.

succession of discrete and separate cell states. Leveraging this key insight, they proposed Monocle, the first method to infer a trajectory on scRNA-Seq datasets. Since then, more than 50 methods have been proposed for trajectory inference (TI) [126].

In Figure 1.2a, we show the output of trajectory inference on the 258 mouse embryo cells from Deng et al. [36] used previously. Here, the trajectory is displayed as a smooth continuous curve that follows the known biological developmental process, although that information was not provided to the TI algorithm (here, we used slingshot [141]). The progression of each cell along that process can be measured alongside the curve, from start to end, and is referred to pseudotime. In this setting, each cell is associated with one pseudotime value.

This example represents the simplest possible developmental setting. It contains a single lineage, with a clear start and end (in the observed dataset).

To illustrate a more complex trajectory, we rely on a dataset of 2,660 bone marrow cells from Paul et al. [108]. In Figure 1.2b, we show those cells in reduced dimension (using UMAP [11, 93]). The developmental process starts in the multipotent progenitor cluster, in purple, around the middle of the plot. Cells then differentiate into two possible paths called lineages. Along the first path (going to the top left), cells differentiate as GMP then basophils, monocytes, and finally neutrophils. Along the second path (toward bottom right), cells differentiate as erythrocytes. In this trajectory with two lineages, each cell is given two pseudotime values, representing how far they have differentiated along each lineage. Indeed, especially at the beginning of the trajectory, cells cannot be definitively assigned to one lineage or another. Instead, each cell is also assigned two weights, representing how close they are to a given lineage. More details on this will be given in Chapters 3 and 4.

More generally, trajectory inference methods take as input the count matrix $\mathbf{Y}$ (or a reduced-dimensional representation $\mathbf{X}$) and return, for each cell, a set of observational weights and associated pseudotimes. In the general setting, the trajectory, which consists of a set of lineages, can also be a circle, when representing, for example, the cell cycle. There is also no requirements that all lineages start at the same place. For example, Cao et al. [27] study organo-genesis by collecting cells from mouse embryos after at least 9.5 days of gestation. At that stage, the initial pluripotent cell stage that gives rise to the different organs is not observed anymore: each set of lineages for a given organ will have a separate root.

Organisation of the next chapters

Chapters 2, 3, and 4 have either been published in refereed journals or released in preprint format. They have been adapted for this dissertation and are structured in similar fashion: an introduction that motivates the problem and provides background, a results section that provides an upper-level description of the method developed, as well as its performances on both synthetic and real datasets, a discussion outlining the conclusions, limitations, and possible future directions of the work, and a methods section that provides greater details and mathematical rigor for the method being developed.

Chapter 2 dives deeper into recent challenges in the clustering of scRNA-Seq data, especially the issue of finding cell types that are replicable between datasets, and presents a method called Dune that focuses on this issue. Chapter 3 focuses on downstream analysis following trajectory inference and the issue of finding genes with dynamic expression associated with the trajectory. It presents tradeSeq, a statistical method to estimate smooth gene expression profiles and perform differential expression analysis between and within-lineages to answer various biologically-relevant questions. Chapter 4 takes another look at the full trajectory inference question in a setting where the dynamic system is studied under different conditions, such as a control and treatment, and present a framework named condiments to conduct analyzes in such a setting.

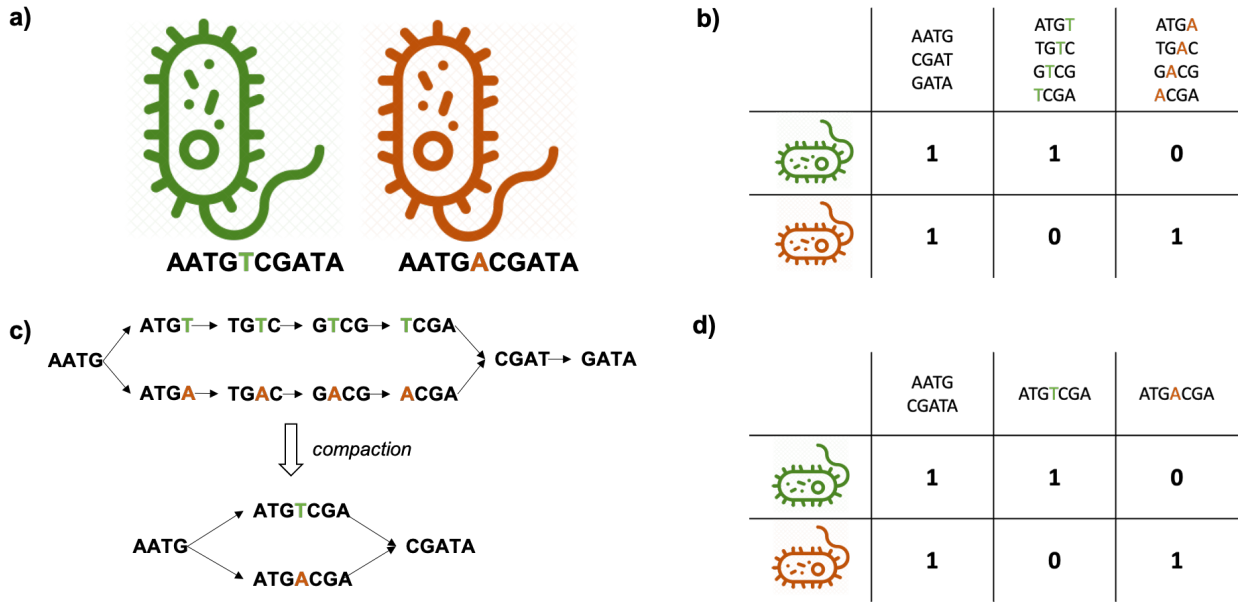


Figure 1.3: *Simple example of bacterial genome-wide association study data.* **(a)** We consider just two samples. The first bacterium is sensitive to antibiotic ($y_1 = 0$) while the second is resistant ($y_2 = 1$). Each sample is associated with just one read of length 10. The base that differs between the two is highlighted in color. **(b)** Each read can be broken-down into 4-mers and the table of presence/absence of all 4-mers in the two groups is shown. **(c)** The De Buijn Graph of all the 4-mers is constructed and then compacted to reduce linear sequences to larger nodes called unitigs. **(d)** The table of presence/absence of all unitigs in the two groups is shown.

Supplementary figures and tables for each chapter can be found in appendices but they are also properly referenced and introduced when relevant in the main chapters. A full workflow using the R implementation of Dune can also be found in an appendix.

1.3 Bacterial Genome-Wide association Studies

Input data

We will restrict ourselves to the study of binary phenotypes. We can assign to each of the n samples a binary phenotype value y_i that takes on values in $\{0, 1\}$. For example, in the setting of antibiotic resistance, we consider that each sample can be either sensitive or resistant to a given antibiotic. Although a continuous phenotype p_i can be binarized by choosing a cutoff value c such that $y_i = \mathbb{1}_{p_i \geq c}$, this can create artifacts (see Sugiyama and

Borgwardt [143] for examples). We therefore restrict ourselves to settings where the two groups are clearly distinct.

For each sample, we also have a set of reads from sequencing. This consists of a set of strings from the four-letter alphabet A, T, C, and G. The size of each string and the number of string are random and depend mostly on sequencing characteristics such as sequencing depth and technology.

As a simple example, we show in Figure 1.3a two samples. The first bacterium (in green) is sensitive to an antibiotic, i.e., its phenotype $y_1 = 0$. Associated with that bacterium, we have one unique read `AATGTCGATA`. Likewise, the second bacterium is resistant (in orange) to the antibiotic ($y_2 = 1$) and has one associated read `AATGACGATA`. The two reads differ only by one letter at the 4^{th} position.

Representing genetic diversity

As mentioned above, representing genetic diversity by comparison to a reference genome is mostly inappropriate for bacterial samples. Other methods have tried to identify genes in each sample and build a presence / absence table using a selected set of genes [40, 63] or all identified genes [104]. However, they rely on the correctness of gene identification algorithms and cannot identify variants outside the coding regions, such as regions involved in transcriptional regulation [14].

More recently, other methods have focuses on the study of k-mers [135, 40]. k-mers are defined as all the substrings of length k present in the reads. By enumeration all the k-mers in all the samples, it is then possible to build the vector of presence / absence of each k-mer among all n samples. In Figure 1.3b, we see all the k-mers associated with all three possible vectors of presence / absence ($\{(1,1),(1,0),(0,1)\}$). The vectors of presence / absence can then be tested for association with the phenotype of interest. Such approaches offer several advantages. They are very fast and scale extremely well, both to increasing sequencing depth and number of samples. However, the short size of the k-mers lead both to redundancy of the information and to a lack of interpretability of the results [116, 113].

To circumvent those issues, Jaillard et al. [62] proposed focusing on the De Bruijn Graph [35]. This graph is built using the k-mers as nodes and drawing an edge between two nodes if the k-mers are found consecutively in the samples. The De Bruijn graph of our example can be found in Figure 1.3c. If a set of nodes in the graph form a linear sequence (i.e., all the nodes only have two edges), then they can be compacted into one new node called a unitig. This creates the compacted De Bruijn Graph, as shown in Figure 1.3c. De Bruijn Graphs have a long history as a relevant data structure for genome assembly [180, 110]. As evidenced in Jaillard et al. [62], De Bruijn graphs provide a more compact representation of the data. Unitigs are often much longer than k-mers, providing easier interpretation without any loss of information (Figure 1.3d). Unitigs also maintain genetic context: variation in a promoter region can be identified as such by mapping neighbouring nodes to a gene database.

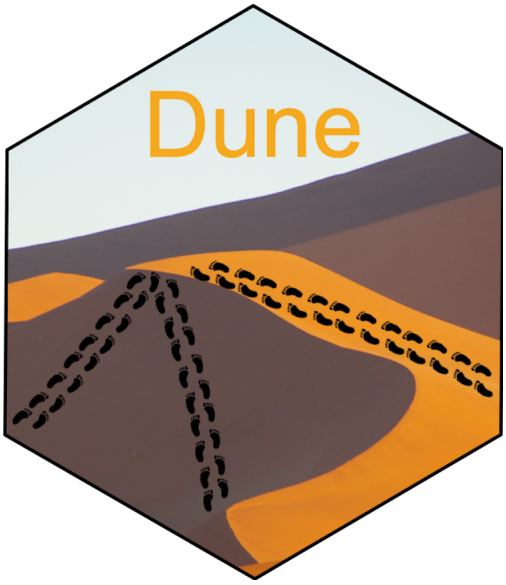
However, focusing only on the individual node limit the type of signal that can be detected. For example, consider a gene associated with the phenotype. Non-coding genetic

variation in that gene will mean that it is is represented by more than one node, thereby diluting the signal. This is even more true for larger structures such as plasmids. In chapter 5, we will expend on the DBGWAS method of Jaillard et al. [62], by presenting CALDERA, an enumeration and testing scheme that focuses not only on nodes of the De Bruijn Graph, but also on subgraphs of that graph and can therefore identify large genetic features associated with the phenotype of interest.

Chapter 2

Improving replicability in cell type discovery

As discussed in chapter 1, an essential part of scRNA-Seq analysis pipeline is cell type discovery and identification. In this chapter, we will focus on a crucial aspect of this step: assessing and improving the replicability of the results from clustering algorithms. A version of this work has been released in preprint format [125] [1].



[1] I would link to deeply thank my collaborators. Kelly Street and Stephan Fischer were involved in conducting the study, which was also designed with supervision from John Ngai, Elizabeth Purdom, Davide Risso, Jesse Gillis and Sandrine Dudoit. Koen Van den Berge and Rebecca Chance also provided crucial feedback on the manuscript

2.1 Introduction

Improvements in single-cell transcriptome sequencing (scRNA-Seq) over the last decade have allowed the characterization of gene expression in collections of thousands to hundreds of thousands of cells. As datasets have grown in size by several orders of magnitude, cell type identification remains a primary step in data analysis [145]. We will focus here on the task of unsupervised clustering, which can be broadly defined as partitioning observations into clusters based on a set of features, without using any prior knowledge on the groupings. In the scRNA-Seq context, clustering aims to identify groups of cells that are defined by a unique and consistent transcriptomic signature. Such groups of cells can represent either transient features, such as cellular states, or more permanent features, such as cellular types.

Many clustering algorithms have been proposed for scRNA-Seq, most of which are adaptations from the clustering literature at large. Popular methods include SC3 [71], Seurat [142], and Monocle [27]. Duò, Robinson, and Soneson [39] offers a recent review of some scRNA-Seq clustering algorithms, identifying SC3 and Seurat as the best-performing methods across a wide range of benchmark settings. However, clustering remains a complex task. Kiselev, Andrews, and Hemberg [70] outline the various challenges – both biological and computational – of this step, including technical noise, biological heterogeneity, and the impact of tuning parameters (or hyper-parameters) for the clustering algorithms. While some methods, including SC3, provide a way of selecting the optimal values of their main tuning parameters, most do not, leaving the choice to the user. Consensus methods try to bypass this issue [121, 71], but they also rely on meta-parameters which can still have substantial impact on the results. Overall, replicating clustering results across datasets remains a difficult task. In this work, we declare clusters to be replicable if running the exact same clustering algorithm with the same tuning parameters on a related dataset yields similar clusters.

Additionally, the aforementioned clustering algorithms identify a pre-specified number of clusters either directly, as in k-means, or indirectly, through another tuning parameter. They implicitly assume that there is only one relevant level of clustering resolution, i.e., an optimal number of clusters, in the dataset. We argue that this is often not the case, since cell types usually have a hierarchy. For example, Tasic et al. [149] propose a tree structure for the mouse anterolateral motor (ALM) and primary visual (VISp) cortical areas. At the higher levels, cells can be clustered as neurons and non-neurons. Then, neurons can be further split into GABAergic and glutamatergic neurons and so on and so forth. This hierarchical structure means that the concept of an "optimal" number of clusters is not appropriate. Instead, many datasets can be better characterized by ever-finer levels of resolution. At the highest level, cells are grouped into broad clusters that are quite coarse, but are easily identifiable and very replicable across datasets. As the resolution increases, distinguishing between reproducible stable cell types and artifacts of the data becomes more challenging. Indeed, these clusters are more likely to reflect over-partitioning (cf. overfitting) of the data or the presence of transient states. This resolution-replicabilty trade-off is not obvious to quantify and is heavily dataset-dependent: it is not only influenced by the biological setting

under study and its complexity, but also highly dependent on technical properties of the data, such as sequencing depth and number of cells [145].

By far the most common method to establish a hierarchy for pre-defined clusters is agglomerative hierarchical clustering, a bottom-up method in which clusters are merged one-by-one until they are all merged into a single cluster. This procedure yields a tree structure linking clusters that are merged together. The tree can also be defined by merging clusters according to the fraction of differentially expressed (DE) genes between them [121, 149]. While several extensive benchmarks of clustering methods have been proposed [39, 45], these only focus on the resulting partitions rather than the full hierarchical structure; and mostly assume that the correct number of clusters is known. Zappia and Oshlack [177] propose a representation of clustering trees to visually describe hierarchies, but this type of analysis relies heavily on user-supervision.

Here, we present Dune, an ensemble method that aims to reconcile multiple clustering results and extract the common structure that they all capture. Dune relies on the assumption that, while different clustering algorithms run with different tuning parameters will naturally provide discrepant clusters, all good clustering methods should be able to identify a common higher-level clustering that is robust to the choice of tuning parameters. This represents a level of resolution that can be analyzed with high confidence given the biology and the dataset's characteristics. Dune identifies this common higher level of resolution shared by all methods without requiring any tuning by the user. Examining this level can both provide useful biological insight and help to compare various clustering methods.

In this manuscript, we first introduce the Dune algorithm. We then demonstrate that Dune outperforms agglomerative merging methods over a variety of simulation scenarios, as well as real scRNA-Seq and snRNA-Seq datasets from different sequencing platforms. We also discuss the value of Dune's stopping point and assess Dune's robustness and limitations.

2.2 Results

Scope of Dune

We wish to delineate at the outset the scope of Dune, i.e., its underlying assumptions, its required inputs, and how to interpret and use its outputs. In practice, researchers often try multiple clustering algorithms to explore different aspects of their data (e.g., resolution levels) and assess robustness of their clustering results. Dune's main assumption is that there is a common higher-level of clustering that should be identifiable by most decent clustering methods. Dune requires as input a set of clusterings, i.e., results from a variety of pre-processing steps, clustering algorithms, and associated tuning parameters applied to a given dataset, that all somewhat capture this higher level. It returns as output merged versions of each of these clusterings, obtained by producing hierarchies of clusters by merging clusters within each partition using information borrowed from the other partitions. As such, it is not a new clustering algorithm and it requires the user to make a number of subjective

choices about both its input and its output. In particular, the user needs to select the set of input clusterings. They also need to select which of its outputs, i.e., which of the merged clusterings, to retain for downstream analysis.

As demonstrated in this manuscript, what Dune accomplishes, however, is to (1) improve upon each of the input clusterings (according to a wide-range of metrics) and (2) lessen the impact of the choice of input clusterings and output clusterings on downstream analysis, by reducing the variability in the quality of the output compared to the input. In other words, the user is left to choose between improved clusterings and their choice is not as critical as if they were to select between the input clusterings.

The Dune algorithm

The Dune algorithm is a general framework that increases the agreement between different clusterings of the same dataset through iterative merging. It takes as input R sets of clustering results, generally produced from running R clustering algorithms (or the same algorithm with different tuning parameter values) on the same dataset. An example can be seen in Figure 2.1a, where a small subset of the AIBS snRNA-Smart dataset [176] (see the "Methods, Case Studies" section) is used to demonstrate some of the main concepts underlying Dune. The first row displays three examples of clusterings (i.e., sets of cluster labels) produced by three different clustering algorithms applied to the same dataset, reduced to two dimensions using t-distributed stochastic neighbor embedding (t-SNE) [160, 159, 75]. All three methods identify similar – but not identical – clusters. Indeed, the algorithms output partitions with different levels of resolution. For example, Monocle splits the bottom region (on the t-SNE plot) into two clusters, while the other two methods find three clusters. Likewise, Monocle and SC3 find two clusters in the top region, while Seurat only finds one. These differences can be summarized using confusion matrices (second row of Figure 2.1a), where the overlap between two clusters from any pair of clusterings is displayed both in terms of the number of cells in the intersection and by the Jaccard index (i.e., the cardinality of the intersection of the two clusters over the cardinality of their union; [59]). Rows and columns are ordered so as to maximize, as much as possible, the sum of the diagonal entries. Confusion matrices can be further summarized using the normalized mutual information (NMI). NMI is a commonly used measure for the agreement between two sets of clustering labels, see the "Methods, NMI" section for more details. As can be seen in the confusion matrices, SC3 and Seurat have the highest level of agreement. Indeed, this is also reflected in the fact that they have the highest NMI of any pair.

Dune merges clusters within each of the R partitions so that the R clustering results more closely match each other. An example of the merging is displayed in Figure 2.1b. Clusters 20 and 21 from SC3 are merged together, resulting in one larger cluster named 20. Doing so increases the agreement between SC3 and Monocle in the confusion matrix, as reflected by an increase in NMI from 0.7 to 0.73. This merge also improves the NMI between SC3 and Seurat (from 0.86 to 0.91) and hence increases the overall agreement between the three clusterings. This is the main idea behind Dune. Specifically, Dune performs an iterative

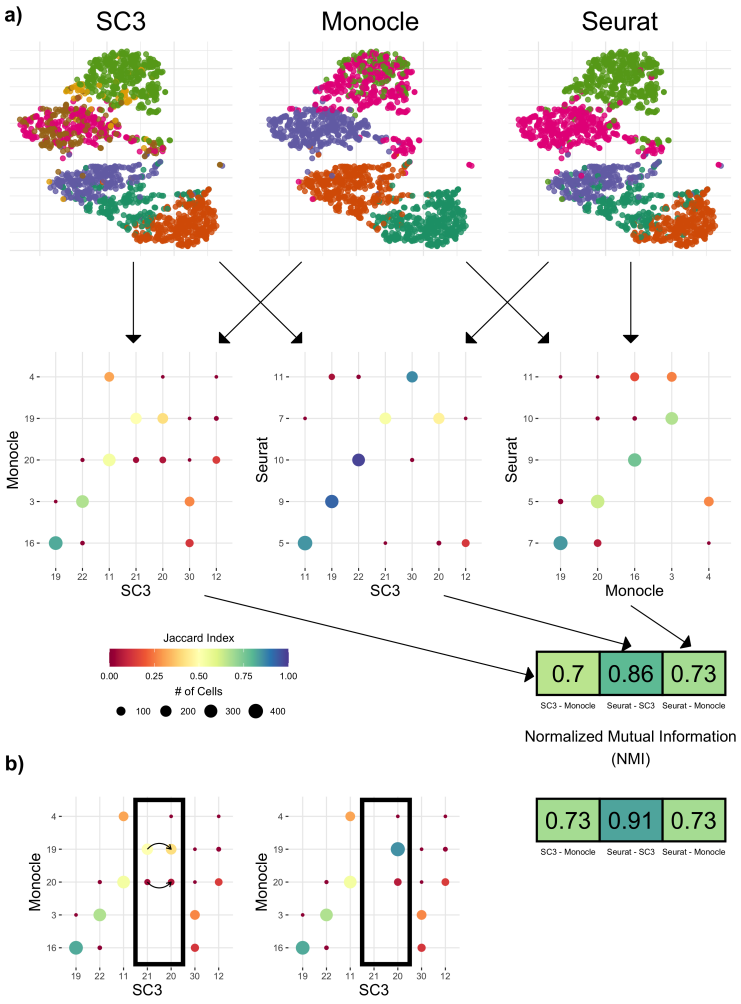


Figure 2.1: *Measuring and improving the concordance between clusterings.* We used a subset of the AIBS snRNA-Smart dataset [176] as an example. Panel **a.** SC3, Monocle, and Seurat were run on the dataset and their results are displayed using scatterx of the first two t-SNE components, where the color of the plotting symbol corresponds to the cluster label. Each pair of clusterings was then compared using a confusion matrix, resulting in three such matrices. For a pair of clusterings/partitions, a confusion matrix is a contingency table, where each entry corresponds to the number of observations in both a cluster from the first partition and a cluster from the second. The size of the dot represents the number of observations in both clusters and the color corresponds to the Jaccard index. Each confusion matrix produces one NMI value. Panel **b.** Merging Clusters 20 and 21 from SC3 into one cluster changes the confusion matrix and increases the NMI.

search where, at each iteration, it identifies the partition and pair of clusters within this partition that, when merged, most improve the average of the normalized mutual information over all pairs of clusterings ($\overline{\text{NMI}}$). Thus, the Dune algorithm can be viewed as an iterative algorithm for maximizing the average pairwise NMI of a collection of clustering results. A more formal definition of the algorithm is provided in the "Methods, Dune" section. Note that the NMI is only one of a variety of criteria that could be used to guide merging. The current implementation of Dune is flexible and allows for other measures. In particular, all benchmarks have also been conducted using the adjusted Rand index or ARI [118, 58], see Sections 2.4 and A.

We demonstrate how the Dune algorithm works in Figure 2.2, using the AIBS scRNA-Smart dataset, a scRNA-Seq dataset of 6,300 mouse brain cells described in the "Methods, Case Studies" section. For this example, we ran SC3, Seurat, and Monocle to obtain our initial clustering results for input into Dune ($R = 3$). Figure 2.2a displays the confusion matrix for a pair of clusterings (SC3 and Monocle) before any merging and Figure 2.2b displays a pseudocolor image of the matrix of all pairwise NMIs for the three clusterings before any merging. The agreement between the three methods is moderate. Indeed, the pairwise NMIs vary between 0.75 and 0.85 in Figure 2.2b. However, as can be seen in the confusion matrix, the clusterings do capture a shared underlying structure, which will serve as grounding for the Dune merging. Figure 2.2d shows the confusion matrix for the same two partitions as in 2.2a, after merging with Dune. We can see that we have, by design, fewer clusters in both partitions, but also that the concordance between the two partitions is greatly improved (as indicated by the color of the plotting symbols, which represents the Jaccard Index). This is further evidenced in Figure 2.2e, where the pairwise NMIs between the three partitions are displayed. The average NMI after all merging steps increased from 0.79 to 0.86. Figures 2.2c and 2.2f demonstrate the evolution of the average NMI and of the number of clusters per partition through the Dune merging process. At each step, we merge the pair of clusters that leads to the greatest increase in average NMI. Hence, at each step, the average NMI increases (Fig. 2.2c) and the number of clusters in one of the partitions decreases by one (Fig. 2.2f). The final partitions are achieved when the average NMI can no longer be improved.

We compare the performance of Dune to other methods of merging, referred to as Dist and DE. Both are hierarchical methods, that start by building a tree between the clusters. The Dist method then merges clusters in a bottom-up manner, starting with the two clusters that are closest in the tree and then iteratively until all clusters are merged. The second approach, DE, follows the method implemented in RSEC and merges clusters bottom-up based on the percentage of DE genes between clusters. It uses the limma package [122], where a gene is declared DE if its nominal false discovery rate (FDR) adjusted p-value is below 0.05 [13]. Pairs of clusters with less than a certain fraction of DE genes are merged. Increasing this threshold from 0 to 1 leads to an iterative merging procedure. More details on these two procedures can be found in the "Methods" section.

In the following sections, we evaluate Dune and compare it to the two hierarchical tree merging methods, using five simulated datasets and four real datasets. We use the simu-

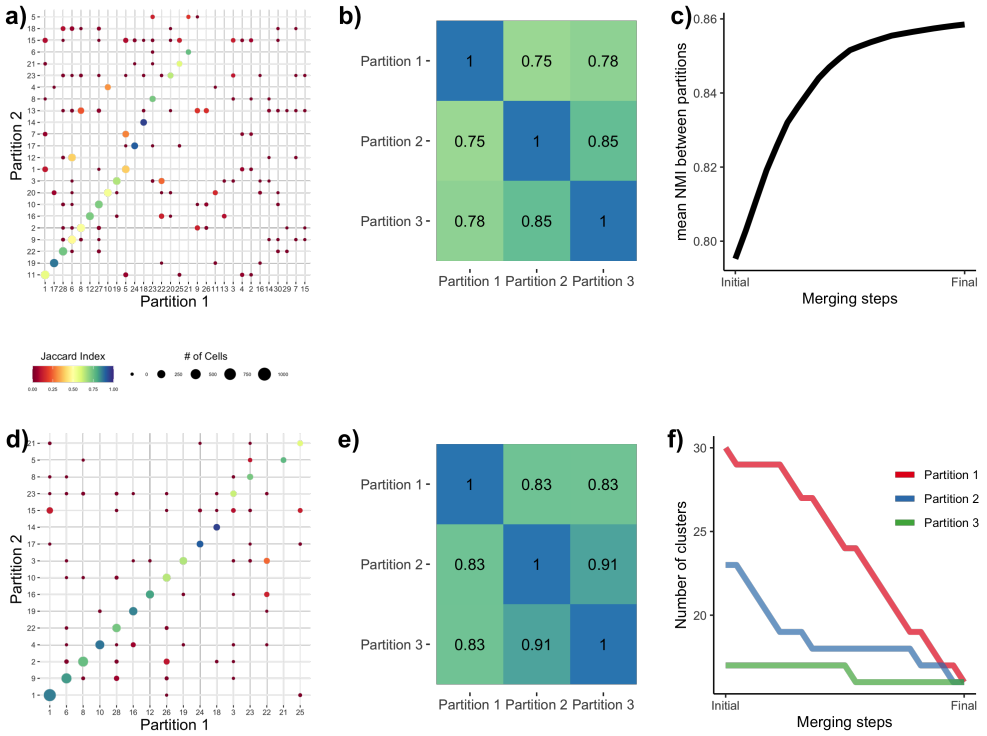


Figure 2.2: *Illustrating* Dune *on a dataset with three sets of cluster labels.* We used the AIBS scRNA-Smart dataset [176] as an example. Before any merging, the sets of cluster labels – or partitions – resulting from running SC3, Seurat, and Monocle have a moderate agreement. Panel **a** displays the confusion matrix between two of the partitions, where each entry corresponds to the number of observations in both a cluster from Partition 1 and a cluster from Partition 2. The confusion matrix shows that while many cells are similarly clustered in the two partitions, i.e., along the main diagonal, many others are not. This can be summarized by the NMI between Partitions 1 and 2. Panel **b** displays a pseudocolor image of the matrix of all pairwise NMIs between the three partitions. Panel **c** illustrates that the average NMI between partitions increases as pairs of clusters are merged when applying Dune. After running Dune, the confusion matrix in Panel **d** and the pairwise NMI matrix in Panel **e** both show that the partitions are indeed more similar. Panel **f** shows that, at each merging step, the number of clusters in one of the partitions is decreased by one, in Dune's greedy procedure to improve the average NMI by merging pairs of clusters.

lated datasets to investigate the value of Dune's stopping rule. Then, we demonstrate the superiority of Dune on real datasets over a wide range of metrics. Finally, we investigate the stability of the Dune algorithm to the clustering inputs and the sample size.

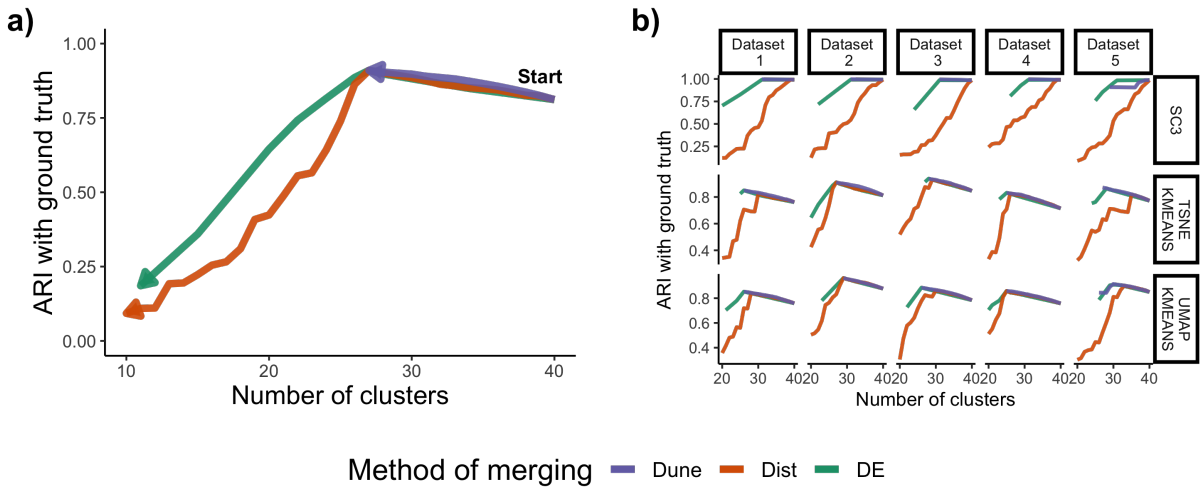


Figure 2.3: *Dune stops at a meaningful level.* Each of the three merging methods is applied to a simulated dataset and the ARI with the ground truth is tracked as the number of clusters decreases. Panel **a.** For Dataset 2 and `tSNE+kMeans`, Dune stops merging right where the concordance is maximal, while the other methods do not. Panel **b.** Over all clustering methods and datasets, Dune stops merging at one point, which always coincide with high agreement with the ground truth.

Dune has a natural stopping point

Unlike other merging methods, Dune provides a meaningful unsupervised stopping point: it merges clusters until no improvement in average NMI occurs, and then stops. Dune's stopping point identifies the level of resolution where all clustering algorithms are close to full agreement. By contrast, the two hierarchical merging methods continue to merge until there is only one cluster, which is not biologically meaningful or interesting.

We demonstrate the stopping rule on simulated data. Using Splatter [178], we generated five simulated datasets of various complexity (see "Methods, Simulations" section for more details), each with 30 clusters. These datasets are simpler than in real settings. As such, methods such as Seurat and Monocle are able to assign cells to the correct clusters with close-to-perfect accuracy over a wide range of simulation parameters. To allow for merging, we therefore relied on simpler methods, where we can specify the number of clusters and purposefully over-cluster. Following Duò, Robinson, and Soneson [39], we applied SC3 without the `sc3_estimate_k` function, as well as kMeans, using as input 3D representations of the data obtained by running either UMAP or t-SNE on the normalized counts. Cluster labels were then merged with Dune, DE, and Dist.

In Figure 2.3a, for example, we merged the clusters obtained from running tsne+kMeans on Dataset 2. For all three merging methods, as merging occurs, the resolution (i.e., num-

ber of clusters) decreases and the concordance with the ground truth increases at first, as measured with the adjusted Rand index (ARI,[118, 58]). Dune then stops when the agreement between cluster labels is at its peak. In this example, this coincides with the maximal agreement with the ground truth. On the other hand, DE and Dist keep on merging until there is only one cluster.

This result holds over all clustering methods and simulated datasets, as shown in Figure 2.3b. Note that the DE method is at an advantage here since it assumes the statistical model that is used to simulate the counts in Splatter. On the other hand, Dune does not assume such a model but performs on par with DE until Dune's stopping point in 14 out of 15 cases.

Dune outperforms other methods on real datasets

Datasets. Our evaluations rely on four datasets. Two are mouse brain datasets from the Allen Institute [176], one single-cell and one single-nucleus RNA-seq dataset, named **AIBS scRNA-Smart** and **AIBS snRNA-Smart**, respectively. Cluster labels for each of these datasets were computed using in-house iterative clustering performed by the authors [149]. The third is a human pancreas dataset [10], referred to as **Baron**, where cells are clustered using hierarchical clustering with a final manual merging step. The last is also a human pancreas dataset [130], referenced as **Segerstople**, where cells were assigned to manually defined clusters using prior biological knowledge. For each of these datasets, we ran $R = 3$ popular [145] clustering methods: SC3, Seurat, and Monocle, with various tuning parameters. The first two have been consistently ranked as some of the best performing clustering algorithms in benchmark studies [39, 45], while the last relies on the same clustering algorithm as Seurat, but with different pre-processing choices and parameter tuning.

Comparison with gold-standard clustering. To evaluate Dune, we first considered how well the resulting merged clusters compare to the published labels. At each merge (i.e., iteration), we computed the ARI between the gold standard and the merged clusters. This led to curves similar to those in Figure 2.3a. The entire ARI curve can be summarized by computing the area under it, referred to herein as the area under the curve (AUC), as depicted in Figure 2.4a.

Measuring clustering replicability across datasets. We then considered the replicability of the clusters found by Dune compared to the other two merging strategies. We measured replicability by evaluating whether the method finds similar clusters for multiple independent datasets – for example, datasets on the same biological system but from different labs or technologies. We considered pairwise comparisons of the clusterings for each of the two mouse brain datasets and for each of the two human pancreas datasets. To measure replicability, we relied on the MetaNeighbor algorithm from Crow et al. [33], which identifies replicable clusters between pairs of datasets (see "Methods, MetaNeighbor" for description).

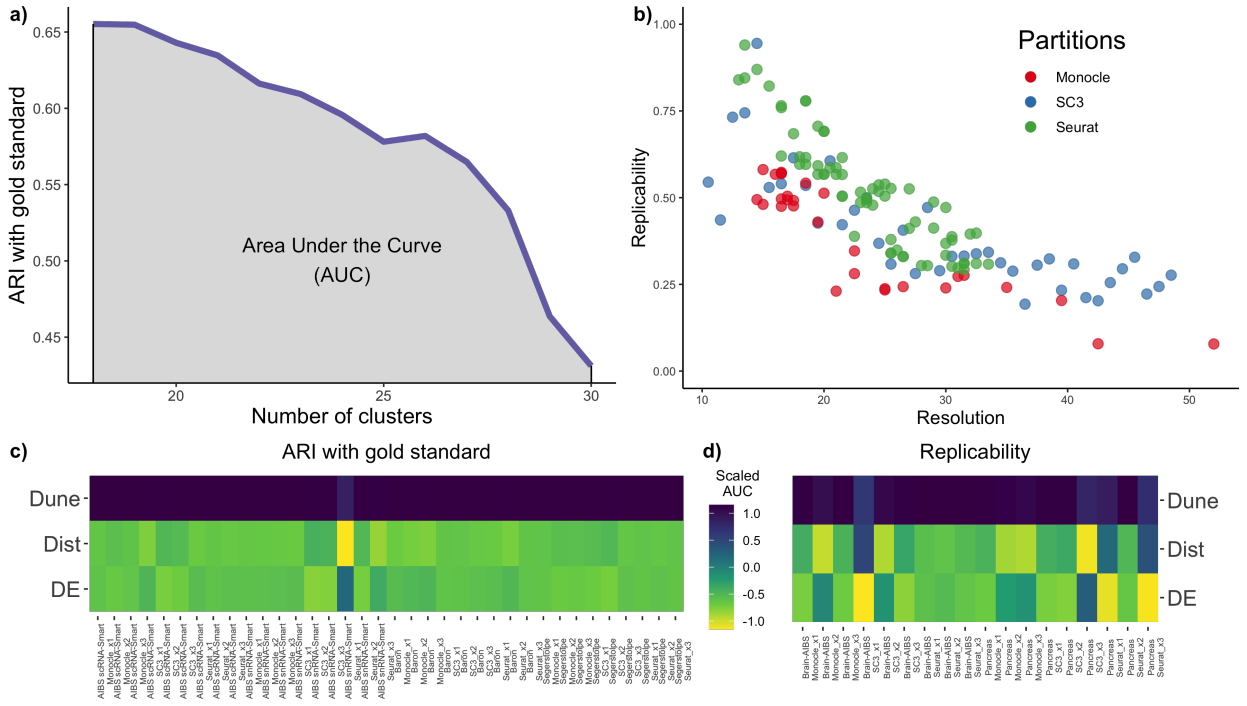


Figure 2.4: *Comparison of methods.* Panel **a.** SC3 was run on the **AIBS scRNA-Smart** dataset for $\theta_{sc3} = 0$ and merged with Dune (with $\theta_{Monocle} = 45$ and $\theta_{Seurat} = 1.2$, for Dune). The ARI with the labels from the original publication, treated as gold standard, was computed at each step of all three merging procedures. The area under this ARI curve was then computed. Panel **b.** SC3, Seurat, and Monocle were run on mouse brain datasets, for a wide range of tuning parameter values. Then, the MetaNeighbor method was used to find the clusters that are replicable between these two datasets and replicability was defined as the fraction of cells in replicable clusters. There is an apparent trade-off between resolution and replicability. Panel **c.** Repeating the procedure from **a.** for three clustering methods, each with three different values of their respective tuning parameter θ, and four datasets yields 36 comparisons of AUC. The resulting 36 AUC are displayed in the pseudocolor image, after being scaled to have a column mean of zero and column variance of 1. This was done to make AUC values comparable across datasets, clustering methods, and parameter values, since the AUARIC can have different scales across scenarios. **d.** Similarly, replicability is tracked as clusters are merged and AUC are computed. This yields 18 comparisons. The color legend is shared between both panels.

The replicability of a clustering was then defined as the fraction of cells in replicable clusters. We used this measure to compare Dune to other merging procedures.

Illustration of the trade-off between cluster resolution and replicability. Figure 2.4b displays replicability vs. resolution for a wide range of clustering results, where three clustering methods (SC3, Seurat, and Monocle) were run with a large grid of tuning parameter values, on the pair of mouse brain datasets. This clearly demonstrates the trade-off between replicability and resolution: as the number of clusters increases, the fraction of cells in replicable clusters decreases, regardless of the clustering method used. While the actual trade-off is specific to the biological context and the pair of datasets that are being considered, it should be stressed that a similar trade-off is clearly visible when applying the same type of analysis to the human pancreas datasets (Figure A.3). Note that although it might be tempting to use this figure to contrast and benchmark clustering methods, this would not be appropriate. Indeed, pre-processing steps were not identical between the three methods – as described in "Methods, Data analysis" – and, as such, no direct comparison is possible.

As pairs of clusters are merged, the resolution decreases, so a well-performing merging method is one that improves the replicability of the clusters. Therefore, a natural way to benchmark merging methods is to measure how and if replicability improves as the number of clusters is reduced. Similar to the comparison with the gold-standard datasets, an area under the replicability curve can be computed to compare all three merging methods.

Comparison of merging methods. While Dune stops merging when the average NMI can no longer be improved, the hierarchical merging procedures have no meaningful stopping point and continue merging until only one cluster is left, as mentioned before. To provide a reasonable stopping point, we stopped the other methods when merging no longer improves the NMI, similar to the requirement of Dune, which means we did not penalize these methods for not providing a natural stopping point. Note that this provides these methods with more information than they would otherwise have had and therefore biases the comparison in their favor.

Figure 2.4c shows the results of benchmarking the merging methods using the ARI with respect to the gold-standard labels, over a multiplicity of scenarios. Dune and the other merging methods rely on one or multiple clustering results – in this work, clusterings from SC3, Seurat, and Monocle. Because each of these three clustering methods has tuning parameters than can affect its performance, we ran each method on a grid of tuning parameter values for each of the four datasets, as described in the "Methods, Data analysis" section. The AUC for the three merging methods across these 36 scenarios are displayed in Figure 2.4c , with column-wise scaling to allow for easier display. Dune clearly outperforms the other two merging methods, ranking consistently first. For the replicability benchmark, since we considered pairs of datasets, the number of comparisons is halved. In Figure 2.4d, Dune outperformed the other two merging methods in all 18 comparisons. Given these results, we forgo further comparisons and focus on Dune in the remainder of the manuscript.

Dune increases the confidence of annotation

While cluster replicability is important in itself, producing robust and replicable clusters has other biologically meaningful applications, including cell type annotation. We investigated how Dune can be used to improve this task. Cell type annotation is a form of supervised classification, where labels learned on one dataset are used to annotate cells from another referred to as target dataset. Here, we relied on the annotation method of Stuart et al. [142], since it also scores the confidence of annotation with a value between 0 and 1, with higher values corresponding to a more confident cell type assignment. We could therefore compute how the average score among all cells of the target dataset evolved when using a clustering method before and after merging. Repeating this across all clustering methods, choices of tuning parameters, and reference datasets (more detail in the "Methods, cell type annotation" section) led to 36 scenarios. We found that merging with Dune consistently improved the confidence of the annotation: the average score increased by 20% for the mouse brain datasets and 10% for the pancreas datasets.

Empirical robustness of Dune

Dune is a semi-supervised method in the sense that it still requires users to select its input: which and how many clustering methods to use, and with which tuning parameters. While Dune does not entirely alleviate the need to make these choices, it provides a higher level of robustness and stability, compared to individual clustering methods. Using both the simulated and real datasets, we evaluated how much the output of Dune is impacted by upstream choices. As detailed below, Dune not only improves the overall quality of the individual clusterings, but importantly lessens the impact of the choice of input and output clusterings.

Robustness to sample size. Dune is very stable to downsampling. Decreasing the number of cells, either before clustering (Figure 2.5a) or after clustering but before Dune (Figure 2.6b), by up to 90% has little negative effect on the quality of the merged cluster labels. For example, on simulated data, the ARI with the ground truth on a dataset with $n = 500$ cells is never less than 97% of its value for a dataset of $n = 5,000$ cells, as shown in Figure 2.5a. Dune is also very stable to adding more clustering inputs. Using a variety of algorithms and associated tuning parameters as input to Dune on the simulation data, we can measure the impact of increasing the number of inputs from $R = 2$ to $R = 9$. For $R \geq 3$, increasing the number of of clustering inputs does not change the quality of the methods (Figure 2.5c). Note, however, that computational times are increased, as Dune scales as R^2 (Section 2.4). For this reason, we have found that in practice using $R = 3$ inputs works best. More details on these evaluations can be found in the "Methods, robustness" section and Section 2.4.

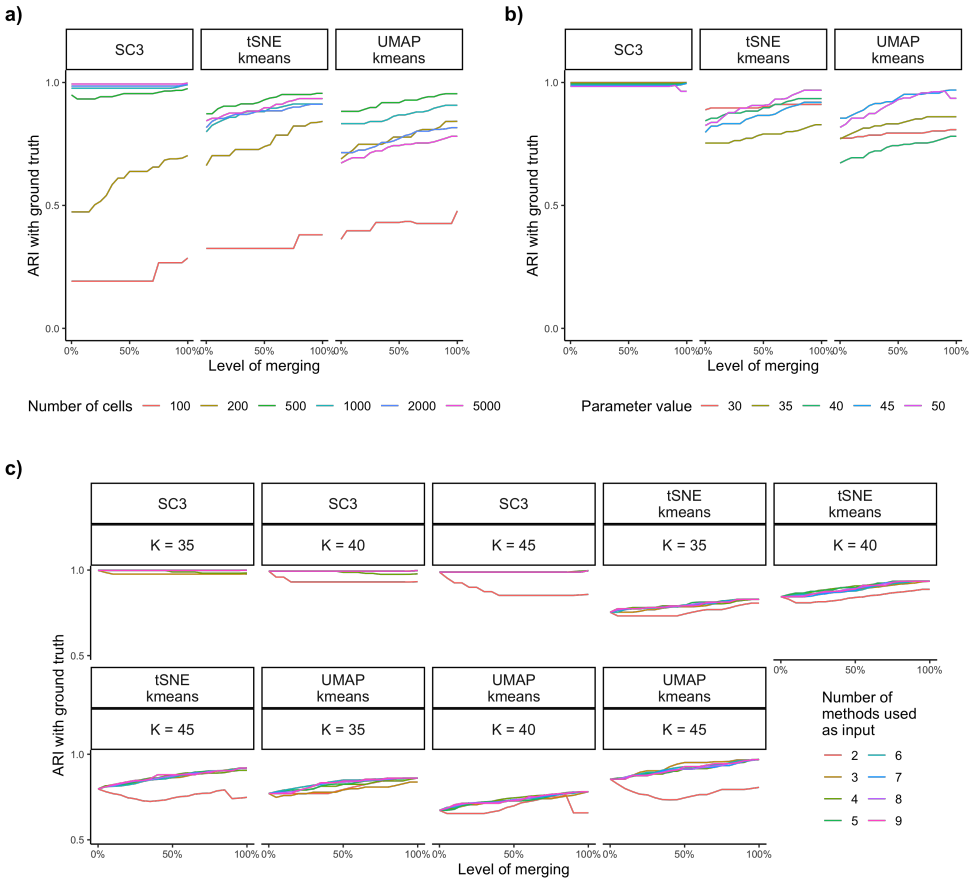


Figure 2.5: *Dune robustness analysis on simulated datasets.* 'Simple' simulated datasets with a DE proportion of .1 are generated. Clusterings from SC3, tSNE+kMeans, and UMAP+kMeans are used as inputs to Dune. Panel **a.** There are $R = 3$ inputs, $k = 40$ for each clustering method, and the number of cells n varies from 100 to 5,000. The quality of the final clusterings is stable to downsampling up to 90%. Panel **b.** Keeping $R = 3$ and $n = 500$, the tuning parameter is changed from $k = 30$ up to $k = 50$. Nearly all partitions are improved by Dune. If there is no over-partitioning ($k = 30$), Dune has very little impact on the quality of clusterings, i.e., there is little room for improvement. The initial ranking of methods is mostly left unchanged by Dune. Panel **c.** Keeping $n = 5,000$, the number of partitions used as input is changed, with partitions being randomly drawn from those described above. Once $R \geq 3$, adding more clusterings as input has a limited effect on the results of Dune.

Robustness to quality of input clusterings. Dune, however, relies on the quality of the input clusterings. As shown in Figure 2.5b-c, the ranking of clusterings before merging is

mostly conserved after merging. Using Dune to polish a poor-quality partition might improve it enough to outperform a high-quality one without merging, but the high-quality partition with Dune merging will nearly always produce better results. Finally, Dune relies on merging to identify a common level of resolution. If all input clusterings represent under-partitioning of the data, little merging will be done, as reflected in Figure 2.5b. As such, inputs to Dune should err on the side of over-partitioning to allow merging to be effective and this should be taken into consideration when selecting tuning parameters.

Selection of output clustering. Finally, Dune merges clusters to improve concordance between its input, but it does not select one more preferably. At this point, as is the case with any clustering workflow, user intervention is needed to select which set of cluster labels to use for downstream analysis. Dune ensures that this step is more stable and less critical by increasing concordance between methods. Indeed, we can assess the variability in quality before and after merging, as measured by the ARI with the ground truth. Over all 53 simulations conducted, the variance in quality after merging with Dune is on average a third of the original variance, and is increased only in one case, when $n = 100$.

Selecting the specific clustering output to retain is outside of the scope of Dune and other criteria need to be used. On simulated datasets, selecting the partition based on the average silhouette width leads to the best method 80% of the time, as measured by ARI or NMI with the ground truth, and never leads to the worst. Likewise, on the mouse brain datasets, when evaluating with either replicability or ARI with the gold standard, selecting a clustering based on the average silhouette width leads to the best method 75% of the time and the second best the remaining 25%. However, for the human pancreas dataset, the clustering with the highest average silhouette width has the lowest replicability and concordance with the gold-standard labels. Overall, there is no single metric that will work all the time. Visual inspection or relying on external biological insight, such as known marker genes, is often the best guide. To demonstrate this, we provide a full workflow, explaining how to use Dune in practice to improve fully off-the-shelf clustering results and illustrating how to select the final output.

2.3 Discussion

We have introduced Dune, a new ensemble method which aggregates clustering results from multiple algorithms. Dune improves upon each of the input clusterings over a variety of metrics, and in particular can correctly navigate the resolution-replicability trade-off in cluster analysis. In this regard, Dune outperforms more commonly used hierarchical merging methods. We stress that Dune is not a new clustering algorithm; instead, it relies on different clustering methods to identify the highest resolution at which cluster quality (i.e., replicability across datasets) remains high. In doing so, Dune identifies the commonalities of the input clusterings and uses this to improve each of these clusterings. lessens the impact of the choice of input clusterings and output clusterings on downstream analysis, by reducing

the variability in the quality of the output compared to the input. That is, the user is left to choose between improved clusterings but their choice is not as critical as if they were to select between the input clusterings. Furthermore, as a result of merging clusters, Dune provides a sensible hierarchy on the clusters based on their commonality across different methods. As we go up in this hierarchy, the number of clusters is reduced, but their replicability improves.

Dune automatically stops at a meaningful resolution level, where all clustering algorithms are in close agreement, while the other methods either keep merging until all clusters are merged into one or require user supervision to stop early. This feature helps users in identifying reliable structure in their scRNA, snRNA, or similar datasets. In contrast, the manual choice of a stopping point is difficult since, in practice, it is often impossible to measure replicability given the lack of a second appropriate dataset.

We focused on the normalized mutual information (NMI) to decide which clusters to merge. The current implementation of Dune also allows users select the ARI as merging criterion; other merging criteria could be implemented. Dune also allows for some cells to remain unclustered, such as currently implemented in RSEC [121]. Possible extensions include clustering methods that do not cluster all cells unambiguously, e.g., soft or fuzzy clustering methods which may assign some cells to multiple clusters based on weights. For now, using such methods as input to Dune would require forcing hard assignments of the cells to clusters (possibly to their nearest cluster). Extensions of the NMI to fuzzy clustering have been proposed [6] and could be evaluated.

This manuscript concerns the question of unsupervised clustering. Recent work in supervised clustering [179, 181, 38, 164] has proposed labeling cells in a new dataset by relying on information contained in other datasets or even cell atlases. In practice, these methods define marker genes for known cell types and build classifiers to assign new cells to these cell types. In particular, Garnett [112] allows a hierarchical clustering structure, but one that needs to be predefined, and scClassify [81] uses the HOPACH [77] algorithm to establish a hierarchy in the training dataset. Most of these algorithms can also identify new cell types not present in the reference. It is therefore possible to use Dune in a supervised clustering context, where one first identifies the cells that have known cell types and, if these do not provide information to help cluster the rest of the cells, one removes them and applies unsupervised clustering methods and Dune to the remaining cells.

While Dune has only been benchmarked on scRNA-Seq and snRNA-Seq datasets, it is a general framework that can be applied to any clustering setting.

2.4 Methods

Consider a – possibly high-dimensional – dataset of n observations, $\mathbf{X} = \{x_1, \ldots, x_n\}$, where $x_i \in R^J$, $i = 1, \ldots, n$. For instance, in scRNA-Seq, x_i corresponds to the J gene expression measures (i.e., normalized read counts) of cell i. Represent the results of any (non-fuzzy) clustering method as a partition, $\mathbf{P}$, which splits the set of n observations into k disjoint subsets or clusters, $\{\mathcal{C}_1, \ldots, \mathcal{C}_k\}$, where: **1)** $\mathcal{C}_i \cap \mathcal{C}_j = \emptyset$, $\forall i, j \in \{1, \ldots, k\}$, and **2)** $\cup_{i \in \{1,\ldots,k\}} \mathcal{C}_i = \mathbf{X}$.

Accordingly, a collection of R clustering results may be represented as multiple partitions, $\mathbf{P}_1, \ldots, \mathbf{P}_R$, with partition $\mathbf{P}_r$ containing k_r clusters, $r = 1, \ldots, R$. For each observation x_i, denote by $c_{i,r} \in \{\mathcal{C}^r_1, \ldots, \mathcal{C}^r_{k_r}\}$ the cluster to which it belongs in partition $\mathbf{P}_r$.

The focus of the present manuscript is to develop a general approach to combine clusters within the different partitions, $\mathbf{P}_1, \ldots, \mathbf{P}_R$, in order to balance the trade-off between cluster resolution and replicability. In the remainder of this section, we first present the mutual information, a well-known measure of concordance between two partitions, and its normalized version. We also review popular clustering methods in the scRNA-Seq literature and alternative approaches to merge clusters. Finally, we formalize the two key notions of cluster resolution and cluster replicability.

Normalized mutual information

Consider two partitions of a dataset, $\mathbf{P}_1$ and $\mathbf{P}_2$. These specify a discrete joint distribution defined by the contingency table with $(i,j)^{\text{th}}$ entry $n_{i,j}$ defined as the number of observations both in cluster i of partition $\mathbf{P}_1$ and cluster j of partition $\mathbf{P}_2$ (Table 2.1). Examples of contingency tables between two partitions can be found in Figures 2.1a, 2.1b, 2.2a, and 2.2d.

Table 2.1: *Contingency table for two partitions* $\mathbf{P}_1$ *and* $\mathbf{P}_2$.

	$\mathcal{C}^2_1$	$\mathcal{C}^2_2$	$\ldots$	$\mathcal{C}^2_{k_2}$	Sums
$\mathcal{C}^1_1$	$n_{1,1}$	$n_{1,2}$	$\ldots$	n_{1,k_2}	a_1
$\mathcal{C}^1_2$	$n_{2,1}$	$n_{2,2}$	$\ldots$	n_{2,k_2}	a_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
$\mathcal{C}^1_{k_1}$	$n_{k_1,1}$	$n_{k_1,2}$	$\ldots$	n_{k_1,k_2}	a_{k_1}
Sums	b_1	b_2	$\ldots$	b_{k_2}	

In general, the mutual information [118] of two random variables measures their concordance. For partitions $\mathbf{P}_1$ and $\mathbf{P}_2$, the mutual information is defined as

$$\begin{aligned}\mathbf{I}(\mathbf{P}_1, \mathbf{P}_2) =& \mathbf{H}(\mathbf{P}_1) - \mathbf{H}(\mathbf{P}_1|\mathbf{P}_2)\\ =& \mathbf{H}(\mathbf{P}_2) - \mathbf{H}(\mathbf{P}_2|\mathbf{P}_1),\end{aligned}$$

where $\mathbf{H}$ is the entropy function. In terms of the contingency table notation, the mutual information is computed as

$$\mathbf{I}(\mathbf{P}_1, \mathbf{P}_2) = -\sum_{i=1}^{k_1} a_i \log(a_i) + \sum_{i=1}^{k_1}\sum_{j=1}^{k_2} n_{i,j} \log\left(\frac{n_{i,j}}{b_j}\right). \tag{2.1}$$

The mutual information can be normalized to yield a value between zero and one. While several variants exist, we selected the following definition for the normalized mutual information (NMI)

$$\text{NMI}(\mathbf{P}_1, \mathbf{P}_2) = \frac{2 \times \mathbf{I}(\mathbf{P}_1, \mathbf{P}_2)}{\mathbf{H}(\mathbf{P}_1) + \mathbf{H}(\mathbf{P}_2)}. \tag{2.2}$$

For R partitions, the overall level of concordance can be quantified by the average NMI for all possible pairs of partitions

$$\overline{\text{NMI}}(\mathbf{P}_1, \ldots, \mathbf{P}_R) = \frac{1}{\binom{R}{2}} \sum_{\{(r,s) \in \{1,\ldots,R\} | r<s\}} \text{NMI}(\mathbf{P}_r, \mathbf{P}_s). \tag{2.3}$$

Note that, in the case of $R = 2$ partitions, this is simply the NMI between the two partitions. If one considers the matrix of pairwise NMIs between partitions, such as displayed in Figures 2.2b and e, then the average NMI is defined as the mean of the upper(or lower)-triangular matrix.

Dune with NMI merging

Given R partitions (possibly the result of different clustering algorithms or different tuning parameter values for the same clustering algorithm, or both), $\mathbf{P}_1, \ldots, \mathbf{P}_R$, with $\mathbf{P}_r$ containing k_r clusters, $r = 1, \ldots, R$, Dune seeks to improve the overall agreement among these, as measured by the average NMI, through an iterative process of merging clusters within partitions.

Specifically, Dune searches over each partition $\mathbf{P}_r$ and over each of $\binom{k_r}{2}$ pairs of clusters in $\mathbf{P}_r$ for the pair which produces the largest improvement in NMI when merged, i.e.,

$$(r^*, i^*, j^*) := \underset{\substack{r \in \{1,\ldots,R\} \\ i,j \in \{1,\ldots,k_r\}}}{\arg\max} \sum_{\{s \in \{1,\ldots,R\} | s \neq r\}} \text{NMI}(\mathbf{P}_r^{i \cup j}, \mathbf{P}_s) - \text{NMI}(\mathbf{P}_r, \mathbf{P}_s), \tag{2.4}$$

where $\mathbf{P}_r^{i \cup j}$ is the partition created by merging clusters $\mathcal{C}_i^r$ and $\mathcal{C}_j^r$ in partition $\mathbf{P}_r$, i.e.,

$$\begin{aligned} \mathbf{P}_r^{i \cup j} :=& \mathbf{P}_r \backslash \{\mathcal{C}_i^r, \mathcal{C}_j^r\} \cup \{\mathcal{C}_i^r \cup \mathcal{C}_j^r\} \\ =& \{\mathcal{C}_1^r, \ldots, \mathcal{C}_{i-1}^r, \mathcal{C}_{i+1}^r, \ldots, \mathcal{C}_{j-1}^r, \mathcal{C}_{j+1}^r, \ldots, \mathcal{C}_{k_r}^r, \mathcal{C}_i^r \cup \mathcal{C}_j^r\}. \end{aligned}$$

Dune amounts to a greedy algorithm for maximizing the average NMI, $\overline{\text{NMI}}$. At each step, we find the pair of clusters that, when merged, lead to the greatest improvement in $\overline{\text{NMI}}$. Once we have identified this pair of clusters, we update the collection of partitions: $\{\mathbf{P}_1, \ldots, \mathbf{P}_R\} \to \{\mathbf{P}_1, \ldots, \mathbf{P}_{r^*}^{i^* \cup j^*}, \ldots, \mathbf{P}_R\}$. We continue iterating until no beneficial merge can be identified, that is, we stop updating when

$$\max_{r,i,j} \sum_{s \neq r} \text{NMI}(\mathbf{P}_r^{i \cup j}, \mathbf{P}_s) - \text{NMI}(\mathbf{P}_r, \mathbf{P}_s) < 0.$$

This greedy approach means that each update step is constrained to merging a single pair of clusters from a single partition. As such, we never merge three clusters together in one iteration or two pairs of clusters in the same or in separate partitions. This ensures that, in our applications, we do not converge to the naive optimal solution of merging all clusters, which does represent a full agreement between the partitions but is of no practical interest. We discuss in greater detail Dune's greedy search strategy in the Supplementary Material (Section 2.4).

While Dune provides a natural stopping point for merging, it is also possible to stop earlier in the merging process, by tuning the merging parameter m_{Dune}, which is defined as the fraction of NMI improvement over the total NMI improvement. For example, $m_{\text{Dune}} = .5$ means that Dune returns the merged partitions that have a mean NMI halfway between the mean NMI of the original partitions and the mean NMI of the final ones.

Dune's search strategy

Computational scalability using contingency tables. Computing contingency tables for n pairs of labels scales as $O(n)$, since each observation must be assigned to a cell of the table. For large datasets, computing contingency tables for all pairs of cluster labels over all possible merges can be a very slow process. However, when merging two clusters, the contingency table can be easily updated by summing the appropriate two rows (or columns). Therefore, updating the contingency table scales as at most $O(K)$ and computing the NMI scales as less than $O(K^2)$. Since we have fewer than $\binom{K}{2} = O(K^2)$ possible merges, identifying the best merge at each step is at most $O(R \times K^4)$. There are at most $R \times K$ merges. So, overall, Dune scales as $O(R^2 \times K^5)$.

On the other hand, if the contingency table were recomputed at each step, or if the merging criterion did not rely on contingency tables but scaled as $O(n)$, the algorithm would scale as $O(R^2 \times (n + K^2) \times K^3)$. In practice, $K \ll n$. Indeed, Svensson, da Veiga Beltrame, and Pachter [145] find that, to a first approximation, $K = O(\log(n))$. Thus, using the trick of merging based on contingency tables, we go from $O(R^2 \times n(\log(n)^3)$ to $O(R^2 \times (\log(n)^5)$. For large datasets with $n = O(10^6)$, this translates in practice to a 100-fold acceleration. Using merging criteria based on contingency tables makes Dune scalable to such large datasets and hence justifies our choice of the ARI or NMI.

Computational scalability using a greedy search. Dune only tries to merge two clusters at a time in just one of the partitions. This is an obviously greedy approach. To understand why such an approach is necessary, let us rephrase the problem Dune is trying to solve: Given a set of R clustering labels, maximize the average NMI by merging any set of clusters in one or more partitions. The general solution to this problem is to merge all clusters for all partitions. This leads to an average NMI of 1, but this degenerate solution is of no practical interest.

To eliminate this issue, an option would be add a penalization term Λ that increases as more merging occurs, such that the function to maximize becomes $\overline{\text{NMI}}(\mathbf{P}_1, \ldots, \mathbf{P}_R) -$

$\Lambda(k_1, \ldots, k_R)$. This becomes a discrete optimization problem and we will show that the naive enumeration of all possible solutions is not feasible.

Solving by enumeration requires identifying all possible merges of any combination of clusters in all partitions at once. The number of possible merges in a partition P_r is the number of possible ways to groups clusters together. This can be redefined as the total number of possible ways to partition the set of clusters, that is, the Bell number $B(k_r)$ [12]. Then, since we have R partitions, we have $\Pi_{r=1}^{R} B(k_r)$ possible merging combinations. Computing the change in NMI resulting from such a merge still scales as $O(K^2)$, relying on the trick from the previous paragraph. Overall, switching from a greedy to a full-enumeration search means scaling from $O(R^2 \times K^5)$ to $O(K^2 \times \prod_{r=1}^{R} B(k_r))$. We can use the following inequality to have a lower bound on the growth rate of the Bell number: $\forall k, B(k+1) \geq (\frac{k}{2})^{k/4}$. Therefore, the growth of the Bell number is more than exponentially larger than k^3. Fully enumerating all possibilities is not feasible in practice and the solution of the penalized problem is not easily found. Since, in practice, the greedy approach of Dune finds an appropriate balance between improving clustering concordance and merging all clusters in reasonable time, we select this approach for our implementation.

Dune merging with ARI

The Rand index [118] measures the concordance between two partitions, $\mathbf{P}_1$ and $\mathbf{P}_2$. Denote by $a = |\{(x_i, x_j) \in \mathbf{X}^2 | (c_{i,1} = c_{j,1}) \& (c_{i,2} = c_{j,2})\}|$ the number of pairs of observations that are in the same cluster for both partitions $\mathbf{P}_1$ and $\mathbf{P}_2$ and by $b = |\{(x_i, x_j) \in \mathbf{X}^2 | (c_{i,1} \neq c_{j,1}) \& (c_{i,2} \neq c_{j,2})\}|$ the number of pairs of observations that are in different clusters for both partitions $\mathbf{P}_1$ and $\mathbf{P}_2$. The Rand index is then the ratio of $a + b$ over the total number of pairs of observations

$$\mathrm{RI}(\mathbf{P}_1, \mathbf{P}_2) = \frac{a+b}{\binom{n}{2}} \in [0, 1]. \tag{2.5}$$

Thus, intuitively, the Rand index is the proportion of pairs of observations for which the two partitions are in agreement.

However, the Rand index does not account for the fact that a pair of observations might be in the same (different) cluster(s) in the two partitions purely by chance. The adjusted Rand index (ARI) [58] adjusts for the level of concordance expected by chance, yielding a value between -1 and $+1$. Specifically, considering $\mathbf{P}$ a fixed partition and R a random permutation of $\mathbf{P}$, then $\mathbb{E}[\mathrm{ARI}(\mathbf{P}, R)] = 0$, where the expected value is over all cluster permutations (i.e., permutations of the cluster assignments of the observations, while keeping the number of clusters and the sizes of the clusters fixed). Negative values indicate less than the expected level of concordance and positive values indicate more than the expected level of concordance. The ARI relies on the contingency table of two partitions $\mathbf{P}_1$ and $\mathbf{P}_2$, with the $(i, j)^{\text{th}}$ entry $n_{i,j}$ defined as the number of observations both in cluster i of partition $\mathbf{P}_1$ and cluster j of partition $\mathbf{P}_2$ (Table 2.1). Given the contingency table notation, the adjusted

Rand index is defined as

$$\mathrm{ARI}(\mathbf{P}_1, \mathbf{P}_2) = \frac{\sum_{i,j}\binom{n_{i,j}}{2} - \frac{1}{\binom{n}{2}}\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}}{\frac{1}{2}\left(\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}\right) - \frac{1}{\binom{n}{2}}\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}}. \tag{2.6}$$

For R partitions, the level of concordance can be quantified by the average ARI for all possible pairs of partitions

$$\overline{\mathrm{ARI}}(\mathbf{P}_1, \ldots, \mathbf{P}_R) = \frac{1}{\binom{R}{2}} \sum_{\{(r,s)\in\{1,\ldots,R\}|r<s\}} \mathrm{ARI}(\mathbf{P}_r, \mathbf{P}_s). \tag{2.7}$$

Note that, in the case of $R = 2$ partitions, this is simply the ARI between the two partitions. If one considers the matrix of pairwise ARIs between partitions, such as displayed in Figures 2.2b and e, then the average ARI is defined as the mean of the upper(or lower)-triangular matrix.

Software implementation and run time

The Dune algorithm is implemented in an open-source R package released through the Bioconductor Project (https://bioconductor.org/packages/release/bioc/html/Dune.html). It is implemented in a fully-parallel and efficient manner. Run time for a large dataset of $\sim 100,000$ cells, with 3 partitions, is under 15 minutes with 10 CPUs. The package also contains plotting functions, which were used to create many of the figures in the present paper and provide options to create GIFs and track the evolution of the average NMI or confusion matrices over the merging steps.

Clustering algorithms for scRNA-Seq data

Any combination of clustering algorithms and associated tuning parameters, applied to an appropriate dataset, can produce a set of partitions that can be used as input to Dune. However, as our work was motivated by the classification of cells based on transcriptomic signatures, we will focus on this particular setting to benchmark Dune.

In the descriptions below, we use the notation from the original publications to describe the tuning parameters of each clustering method; the same notation may therefore correspond to different parameters depending on the algorithm.

SC3 [71] is a consensus clustering method that involves performing k-means clustering on different dimensionality reductions of the input dataset. A hierarchical clustering method is then applied to the resulting consensus matrix. The main tuning parameter is the number of clusters k, which is used both in k-means and to cut the hierarchical clustering tree. The method provides an estimate of the optimal value of this parameter, k_0, based on the number of eigenvalues of the centered and scaled distance matrix that are significantly different from 0 (see Kiselev et al. [71] for more details). For large datasets, there exists a hybrid version

of the algorithm, where the full SC3 clustering method is run on only a fraction of the cells to identify the clusters and the rest of the cells are assigned to the clusters using a support vector machine (SVM) algorithm.

Seurat's clustering algorithm ($SEURAT, RRID : SCR\_007322$) has evolved over the different versions of the software; here, we focus on version 3 [142] (we specifically use version 3.1.1). The algorithm first reduces the dimension of the data by selecting the first p principal components (PCs) and then computes a k-nearest neighbor (k-NN) graph. After refining the graph, it groups cells using, as default, the Louvain algorithm [15]. The two main tuning parameters are the number of neighbors k used to build the k-NN graph and the resolution parameter for the Louvain algorithm.

Monocle's clustering algorithm has also changed and we focus on version 3 [27] (implemented in the Monocle3 package, although we keep the name Monocle for simplicity; we specifically use version 0.1.3). Monocle's clustering algorithm is similar to the one implemented in Seurat, with a few differences. After initial dimensionality reduction based on principal component analysis (PCA), Monocle performs another dimensionality reduction step using uniform manifold approximation and projection (UMAP) [11, 93] and relies on that representation to build the k-NN graph. It then clusters cells using, by default, the Leiden algorithm [153].

Resampling-based sequential ensemble clustering (RSEC [121]) is a consensus method over user-supplied clustering algorithms and their associated tuning parameters. In order to improve the stability and tightness of the clusters, it also provides the option to perform clustering on subsamples of the observations, as well as sequential clustering. However, in this paper, we mainly use RSEC for its final step of hierarchical merging, see section "Existing methods to merge clusters".

Tuning parameters

For each method, we only tune the main parameter. For Seurat, however, there are two main tuning parameters. The k parameter controls the number of neighbors used to build the k-NN graph, while the resolution parameter defines the neighborhood in the Louvain clustering algorithm. In practice, the k parameter has much less impact than the resolution parameter (see Figure A.1). Moreover, depending on the value of the resolution, increasing k either increases or decreases the final number of clusters. Accordingly, we only consider changing the resolution parameter.

For ease and generality of notation, we will denote each method's main tuning parameter by θ and define θ such that increasing θ increases the number of clusters. Thus, for the methods described above, $\theta_{SC3} = k$, $\theta_{Seurat} =$ Resolution, and $\theta_{Monocle} = -k$. Each combination $\Theta = \{\theta_{SC3}, \theta_{Seurat}, \theta_{Monocle}\}$ of the three parameters defines a set of partitions that serves as input for Dune.

Existing cluster merging methods

Once a set of clusters has been identified, one can build a hierarchical tree for these clusters and then merge clusters that are similar. This involves specifying a measure of distance or similarity between individual observations (i.e., cells) as well as between clusters. It should be noted that the distance used to build the tree of clusters need not be the same as the distance used to merge clusters.

For scRNA-Seq datasets, commonly-used between-cell distance measures include the Euclidean distance and one minus the Spearman correlation coefficient. Between-cluster distances include classical linkage measures used in hierarchical clustering, e.g., maximum / minimum / average of all pairwise distances between observations in two clusters or distance between the cluster averages or medoids. For scRNA-Seq, another sensible between-cluster distance measure is the proportion of differentially expressed (DE) genes between clusters [121, 149]. A detailed discussion of such measures is out of the scope of this manuscript[69].

Here, we consider two possible merging approaches. In both cases, we compute the cluster medoids (median of observations within the cluster) based on the log-transformed count matrix (adding 1 to avoid taking the log of zero). We then build a hierarchical tree of clusters using the Euclidean distance between the cluster medoids. The first merging approach directly uses this tree to decide how to merge clusters. Specifically, clusters are merged bottom-up, starting with the two clusters that are closest in the tree and then iteratively until all clusters are merged. The parameter $m_{Dist} = n_{merges}$, the number of merges (between 0 and the initial number of clusters minus one), controls the amount of merging. The second approach follows the method implemented in RSEC. It computes the percentage of DE genes between clusters, using the limma package [122] ($LIMMA, RRID : SCR\_010943$), where a gene is declared DE if its nominal FDR adjusted p-value is below 0.05 [13]. The main tunable parameter is $m_{DE} = \alpha \in [0, 1]$, the threshold for the percentage of DE genes below which we merge. We name these two methods Dist and DE, respectively.

Cluster replicability using MetaNeighbor

We quantify the replicability of clusters across datasets by applying a modified version of unsupervised MetaNeighbor [33] ($MetaNeighbor, RRID : SCR\_016727$). MetaNeighbor requires as input two unnormalized datasets, two set of cluster labels, and a set of highly variable genes. One of the datasets is treated as a test dataset, where all cluster labels are hidden, the other dataset is treated as a training dataset, whose labels are propagated to the test dataset through a cell-cell similarity network.

To identify replicating clusters, we computed replicability scores using the `MetaNeighborUS` function with parameters "one-vs-best=TRUE" and "symmetric_output=FALSE". Briefly, Each pair of clusters (one in the training dataset, the other in the test dataset) receives a score based on how well the training cluster predicts the labels from the test cluster. Each cluster from the training dataset is then assigned a unique best matching cluster in the test set: the one it can predict the best. Finally, we compute a final score for each training set

cluster by reducing the test set to the two best matching clusters using the original score. Therefore, the final score measures how well a training cluster predicts a specific test cluster compared to its closest neighbour. Then the role of the test and training datasets are reversed. Each cluster from the two dataset is therefore assigned a target in the other dataset, and an associated score.

A cluster is considered replicable if there is a cluster in the other dataset such that the clusters are reciprocal best hits with score > 0.6 both ways. See Crow et al. [33] for more details and benchmark of this method. Finally, the **replicability score of a clustering** is defined as the fraction of cells contained in replicable clusters. More specifically, for a comparison of two datasets, we enumerate replicable clusters in each dataset, then deduce the number of cells that are in replicable clusters, sum this number across datasets, and divide by the total number of cells.

We used MetaNeighbor's `variableGenes` procedure to select genes that are highly variable across all datasets. For computational cost reasons, the `variableGenes` procedure was applied to a random subset of 50,000 cells for datasets exceeding that size. However, the full datasets were used for the rest of the analysis. In the end, we obtained a set of 541 highly variable genes for the brain datasets and $2,147$ genes for the pancreas datasets.

Simulation studies

Simulation study design. To generate simulated datasets with known ground truth, we relied on the Splater package [178]. Datasets of $n = 5,000$ cells and $J = 10^4$ genes were generated with 30 cell types. No batch effects were added, given that benchmarking of normalization procedures was of no interest for our purpose. The average proportion of differentially expressed genes between clusters, `DE`, was tuned between datasets. We generated two types of datasets. 'Simple' datasets had balanced numbers of cells per cluster, i.e., each cluster had $5,000/30 \approx 166$ cells, and the DE proportion (one-versus-all) was the same for every cluster. 'Hard' datasets had unbalanced designs, i.e., cells were randomly assigned to each cluster, and the DE proportion was sampled from a uniform distribution $\mathcal{U}[.75 * \texttt{DE}, 1.25 * \texttt{DE}]$.

Simulations parameters for each dataset. Therefore, each dataset is defined by the `DE` parameter and a label 'hard' or 'simple'. Datasets with the same parameters are however only identical if the random seed is set to an identical value.

Data analysis. Outlier genes and cells were removed and the data were normalized using the Seurat workflow. UMAP 2D plots of each dataset can be found in the supplementary figures A.2a-e. k-means was run with $k = 40$ on either UMAP or t-SNE reduced dimensions. SC3 was also run with $k = 40$. Then, clusters were merged using either Dune (with ARI or NMI as merging criteria), the DE, or Dist methods.

Table 2.2: *Simulation parameters.*

	DE	Type
Dataset 1	.1	'Simple'
Dataset 2	.1	'Simple'
Dataset 3	.05	'Simple'
Dataset 4	.1	'Hard'
Dataset 5	.05	'Hard'

Case studies

AIBS Smart mouse brain datasets

We used the two AIBS Smart datasets produced as part of the Brain Initiative Cell Census Network (BICCN: $RRID:SCR\_015820$) and described in Yao et al. [176], one corresponds to single-cell sequencing (*Zeng sn SSv4* `https://assets.nemoarchive.org/dat-k7p82j4`) and the other to single-nucleus sequencing (*Zeng sc SSv4* `https://assets.nemoarchive.org/dat-55mowp9`). We use the original publication's subclass labels as gold-standard cluster labels for these datasets. The datasets can be downloaded from the Neuroscience Multi-omics Archive ($RRID:SCR\_002001$; `nemoarchive.org`). More details on the parent dataset (`https://assets.nemoarchive.org/dat-ch1nqb7`) and data access can be found in Yao et al. [176].

Human pancreas datasets

We focus on two datasets from [10] (8,568 cells) and [130] (3,514 cells), which we name **Baron** and **Segerstople**, respectively. Both datasets were downloaded from https://hemberg-lab.github.io/scRNA.seq.datasets/ on October 1st, 2018. We use the clusters from the original publications as gold-standard cluster labels.

Data analysis

Except when otherwise specified, all methods and algorithms were run with default parameters or, if no available default, with the parameters recommended in the vignette or tutorial.

Pre-processing. Count matrices were filtered to remove lowly-expressed genes with fewer than i reads in j cells. See Table 2.3 for values of i and j for each dataset.

As indicated below, we follow different normalization strategies before running Seurat and Monocle in order to obtain more diverse clustering results. This is appropriate, as the goal of the manuscript is not to compare different clustering methods, but rather different merging methods for given clustering results. The merging methods that Dune is compared

to rely on only one clustering input; we therefore seek to benchmark merging methods using a variety of clustering inputs.

Seurat. Following the tutorial, we run `FindVariableFeatures` and `ScaleData` to normalize the data. Counts are log-transformed (adding 1 to avoid taking the log of zero) and normalized by sequencing depth. For the two pancreas datasets, batches are also normalized for using the `scaleData` function. Following principal component analysis, `FindNeighbors` and `FindClusters` are run for a number of neighbors k in $\{30, 50, 70\}$ and resolution θ from 0.3 to 2.5 in increments of 0.1.

SC3. The algorithm is run on a dataset normalized as above with the Seurat pipeline. The optimal value of k, k_0, is computed using the `sc3_estimate_k` function. The parameter θ is transformed to be $\theta_{SC3} = k - k_0$. SC3 is then run for values of θ ranging from -15 to $+15$.

Monocle. zinbwave [121] is first used for normalization and dimensionality reduction on the filtered count data. For the two pancreas datasets, batches are included as model covariates. We select K, the number of reduced dimensions, based on a visual representation for each dataset, see Table 2.3. This first step of dimensionality reduction is followed by another using UMAP [93] with two dimensions. The resulting two-dimensional representation is then used to build the k-NN graph, with k ranging from 10 to 150 in increments of 10.

Dune. For a given set of values for $\Theta = \{\theta_{SC3}, \theta_{Seurat}, \theta_{Monocle}\}$, we get three sets of cluster labels that we can use as input to Dune.

Building the hierarchical tree. The output of each clustering method is used as input to RSEC's `makeDendogram` function. Then, we either cut the tree using R's `cutree` function (for the Dist merging method) or RSEC's `mergeClusters` function (for the DE merging method).

Cell type annotation. Each dataset is normalized as described in the Seurat paragraph. For each pair of datasets (mouse brain or human pancreas), one dataset is used as reference and the other as target for which cells are to be labeled. The reference dataset is labeled using the cluster labels either before or after merging with Dune, for all values of θ described above. Each cell in the target dataset is assigned a label and a score using the Seurat `TransferData` function. The average score across all cells is used to evaluate the quality of the annotation.

Producing "bad" clusters. For each value of the tuning parameters Θ, on the pancreas datasets, we add fully random inputs to Dune. That is, we create "bad" clusterings by randomly assigning each cell a number (or cluster label) between 1 and $(k_{SC3} + k_{Monocle} + k_{Seurat})/3$, where k denotes the number of clusters for a particular clustering algorithm. Since cells are assigned randomly, the size of the clusters will vary, but all clusters have the

same expected size. To account for the stochastic nature of this procedure, we repeat this 10 times.

Downsampling. Downsampling the number of cells at the beginning of the analysis pipeline would affect both the quality of the input clusterings and the quality of the merging with Dune. As such, to test only the stability of Dune to the number of cells, we downsample the cells just before running Dune, that is, the clustering algorithms are run on the full dataset but only a subset of the dataset is used to decide which clusters to merge and in which order. Afterwards, cells that are not in the subsample are assigned to the merged clusters based on their original cluster labels. That is, if Clusters 1 and 2 are merged, all cells that were originally in Cluster 1 or Cluster 2, even those not selected in the downsampling and not used as input to Dune, are assigned to the merged cluster.

Most of the code was run using xsede [152].

Table 2.3: *Parameters for pre-processing the datasets.* Each dataset is filtered such that we keep all genes with a least i reads in j samples. Then, zinbwave is run with K dimensions.

Dataset	i	j	K
AIBS scRNA-Smart	50	50	30
AIBS snRNA-Smart	50	50	14
Baron	5	5	10
Segerstople	5	5	20

Pre-processing parameters

Robustness analysis

Robustness on simulated datasets

Simulation settings. All simulations presented in the robustness analysis are for scenarios of type 'Simple' with a DE proportion of .1, just as Datasets 1 and 2 of the simulation benchmark.

Downsampling. We generated datasets of $n \in \{100, 200, 500, 1,000, 2,000, 5,000\}$ cells. On these datasets, we used SC3, tSNE+kMeans, and UMAP+kMeans (all with $k = 40$) as input to Dune. Then, the ARI with the ground truth was computed as merging occurs.

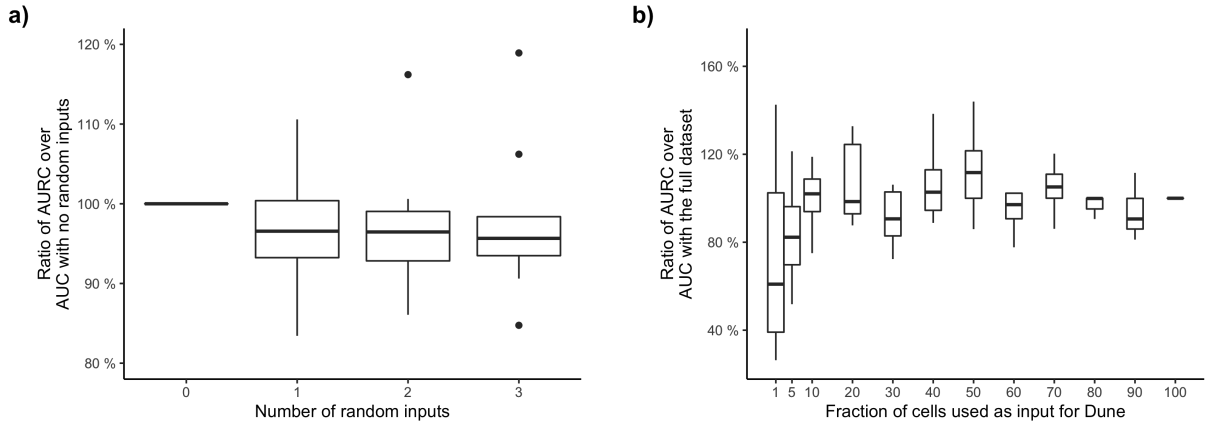


Figure 2.6: *Dune robustness analysis on real datasets.* Panel **a.** Adding an increasing number of random clustering inputs to Dune impacts only slightly the resolution-replicability area under the curve when merging the other correct clusters. Panel **b.** Likewise, Dune is stable to decreasing the number of input cells, as low as 10% of the original sample size.

Varying tuning parameters. For the dataset with $n = 5,000$ cells generated above, we used SC3, tSNE+kMeans, and UMAP+kMeans as input to Dune. The tuning parameter was identical for all methods and varied between $k = 39$ and $k = 50$. Then, the ARI with the ground truth was computed as merging occurs.

Changing the input clusterings. For the dataset with $n = 5,000$ cells generated above, we used SC3, tSNE+kMeans, and UMAP+kMeans, with $k \in \{35, 40, 45\}$. We randomly sampled a set of R (between 2 and 9) clustering inputs among the 9 methods. This process was repeated at least 5 times for each value of R, or until all possible combinations were selected if $\binom{9}{R} < 5$. Then, the ARI with the ground truth was computed as merging occurs. To facilitate presentation, we take the average over all repetitions for a given clustering method with a given tuning parameter value and given value of R.

Robustness on real datasets

Robustness to poor clustering inputs. Since Dune takes as input the results from clustering algorithms, its results depend on the quality of the clusterings produced by these algorithms. In general, Dune will not be able to produce good clusters when merging only clusters that capture no underlying biological signal. However, we showed that Dune is robust to a mix of "good" clustering inputs and "bad" clustering inputs. We used as "good" inputs the results of SC3, Seurat, and Monocle that are assumed to capture some common signal, and as "bad" inputs fully random clusters that will not have any commonality (see

the "Methods, Data analysis" section). Then, the replicability of the "good" clusterings was measured as merging happened and the AURC was computed and compared to the AURC when there were no "bad" inputs. As more and more "bad" clusters were added (Figure 2.5b), Dune still improved the replicability of the "good" clusters as it merged them, even when half of the clusters used as inputs were random. Hence, Dune can recover from very poor clustering inputs.

Robustness to sample size. We investigated how Dune handles datasets with an ever-smaller number of cells. To simulate such datasets, we downsampled the two pancreas datasets. Downsampling could affect both the quality of input clusters and the merging procedure of Dune. To disentangle these two effects, we downsampled the two human pancreas datasets after running SC3, Seurat, and Monocle, but before running Dune. We then measured how and whether merging still improved the cluster replicability by computing the AURC and constrasting it to its value without downsampling (see the "Methods, Data analysis" section for more details).

When the datasets were downsampled to between 90% and 10% of the original number of cells, Dune still correctly navigated the trade-off between resolution and replicability (Fig. 2.5c). Only when fewer than 10% of the cells were used (which amounts to datasets of fewer than 200 cells) did Dune's capacity to improve clustering replicability worsened noticeably. This demonstrates that the method is very stable to the number of cells.

Data availability

The Pancreas datasets were downloaded from the Hemberg group website, https://hemberg-lab.github.io/scRNA.seq.datasets/human/pancreas/, on October 1^{st}, 2018. The AIBS datasets can be obtained from the Neuroscience Multi-omics Archive ($RRID : SCR\_002001$; nemoarchive.org), *Zeng sn SSv4* at https://assets.nemoarchive.org/dat-k7p82j4 and *Zeng sc SSv4* at https://assets.nemoarchive.org/dat-55mowp9.

Code availability

The results from this paper can be reproduced using code from the following GitHub repository: https://github.com/HectorRDB/Dune_Paper. The Dune method is implemented in an open-source R package released through the Bioconductor Project (http://www.bioconductor.org/packages/release/bioc/html/Dune.html).

Notation. Following the "Methods" section, we consider a – possibly high-dimensional – dataset of n observations, $\mathbf{X} = \{x_1, \dots, x_n\}$, where $x_i \in R^J$, $i = 1, \dots, n$. For instance, in scRNA-Seq, x_i corresponds to the J gene expression measures (i.e., normalized read counts) of cell i. Represent the results of any (non-fuzzy) clustering method as a partition, $\mathbf{P}$, which splits the set of n observations into k disjoint subsets or clusters, $\{\mathcal{C}_1, \dots, \mathcal{C}_k\}$, where: **1)**

$\mathcal{C}_i \cap \mathcal{C}_j = \emptyset$, $\forall i, j \in \{1, \ldots, k\}$, and **2)** $\cup_{i \in \{1,\ldots,k\}} \mathcal{C}_i = \mathbf{X}$. Accordingly, a collection of R clustering results may be represented as multiple partitions, $\mathbf{P}_1, \ldots, \mathbf{P}_R$, with partition $\mathbf{P}_r$ containing k_r clusters, $r = 1, \ldots, R$. For each observation x_i, denote by $c_{i,r} \in \{\mathcal{C}_1^r, \ldots, \mathcal{C}_{k_r}^r\}$ the cluster to which it belongs in partition $\mathbf{P}_r$. We also let $K = \max_r\{k_r\}$, i.e., the largest number of clusters among all partitions.

Chapter 3

Trajectory-based differential expression analysis

The importance of trajectory inference was discussed in chapter 1. In this chapter, we will focus on how to conduct gene differential expression downstream of trajectory inference. A version of this work has been published at Nature Communications [158] [1].

[1] I would link to deeply thank my collaborators. Koen Van den Berge spearheaded this project, with deep involvement by Kelly Street, under supervision by Sandrine Dudoit and Lieven Clement. Wouter Saelens, Robrecht Cannoodt and Yvan Saeys provided crucial help designing the simulation studies

3.1 Introduction

Single-cell RNA sequencing (scRNA-seq) has revolutionized modern biology by allowing researchers to profile transcript abundance at the resolution of an individual cell. This has opened new avenues to study cellular pathways during the cell cycle, cell type differentiation, or cellular activation. Indeed, scRNA-seq can provide a snapshot of the transcriptome of thousands of single cells in a cell population, which are each at distinct points of the dynamic process under study. This wealth of transcriptional information, however, presents many data analysis challenges. Until recently, statistical and computational efforts have focused mostly on trajectory inference (TI) methods, which aim to first allocate cells to lineages and then order them based on pseudotimes within these lineages. A wide range of TI methods have been proposed; 45 of which are extensively benchmarked in Saelens et al. [126]. Note that we use the term *trajectory* to refer to the collection of *lineages* for the process under study.

Most TI methods share a common workflow: dimensionality reduction followed by inference of lineages and pseudotimes in the reduced-dimensional space [24]. In that reduced-dimensional space, a cell's pseudotime for a given lineage is the distance, along the lineage, between the cell and the origin of the lineage. As such, while pseudotime can be interpreted as an increasing function of true chronological time, there is no guarantee that the two follow a linear relationship. While early methods were limited to inferring trajectories comprised of a single linear lineage, recent developments have allowed the inference of trajectories that might bifurcate multiple times and consist of several smooth lineages, or that might have cyclic patterns [141, 84, 115]. These advances in TI methods enable researchers to study dynamic biological processes, such as complex differentiation patterns from a progenitor population to multiple differentiated cellular states [23, 56], and have the promise to provide transcriptome-wide insights into these processes.

Unfortunately, statistical inference methods are lacking to identify genes associated with lineage differentiation and to unravel how their corresponding transcriptional profiles are driving the dynamic processes under study. Indeed, differential expression (DE) analysis of individual genes along lineages is often performed on discrete groups of cells in the developmental pathway, e.g., by comparing clusters of cells along the trajectory or clusters of differentiated cell types. Such discrete DE approaches do not exploit the continuous expression resolution that can be obtained from the pseudotemporal ordering of cells along lineages provided by TI methods. Moreover, comparing cell clusters within or between lineages can obscure interpretation: it is often unclear which clusters should be compared, how to properly combine the results of several pairwise cluster comparisons, or how to account for the fact that not all these comparisons are independent of each other. Inevitably, the number of cluster comparisons also increases rapidly with the number of lineages of interest, leading to multiple testing issues at the gene level [157] and further decreasing the reproducibility of scRNA-seq DE results.

A number of methods have been developed for the analysis of bulk RNA-seq time-series data, which can exploit the continuous resolution of samples assayed at different times [102, 127, 5]. Nueda, Tarazona, and Conesa [102] requires multiple observations for each time-

point and estimates the mean expression for each time-point. However, in scRNA-seq, cells are never at the exact same pseudotime value. Other approaches assume a piecewise linear [5] or polynomial [127] relationship of mean expression with time, which provides insufficient flexibility to model the complex relationship between gene expression and pseudotime observed in scRNA-seq datasets. Often, these methods are also restricted to the estimation of only one or two functions for each gene.

A few methods have been been published with the aim of improving trajectory-based differential expression analysis by modeling gene expression as a smooth function of pseudotime along lineages. Monocle [154] tests whether gene expression is associated with pseudotime by fitting additive models of gene expression as a function of pseudotime. However, the method can only handle a single lineage. A similar approach has been adopted by TSCAN [64]. GPfates [84] relies on a mixture of overlapping Gaussian processes [79], where each component of the mixture model represents a different lineage. For each gene, the method tests whether a model with a bifurcation significantly increases the likelihood of the data as compared to a model without a bifurcation, essentially testing whether gene expression is differentially associated with the two lineages. Similarly, the BEAM approach in Monocle 2 [115] allows users to test whether differences in gene expression are associated with particular branching events on the trajectory. These trajectory-based methods improve upon discrete cluster-based approaches by: (1) exploiting the continuous expression resolution along the trajectory and (2) comparing lineages using a single test based on entire gene expression profiles. However, both GPfates and Monocle 2 lack interpretability, as they cannot pinpoint the regions of the gene expression profiles that are responsible for the differences in expression between lineages. Moreover, the GPfates model is restricted to trajectories consisting of just one bifurcation, essentially precluding its application to biological systems with more than two lineages (i.e., a multifurcation or more than one bifurcation). BEAM is restricted to the few dimensionality reduction methods that are implemented in the Monocle 2 software, namely, independent component analysis (ICA) and DDRTree [115]. Hence, novel methods to infer differences in gene expression patterns within or between transcriptional lineages with complex branching patterns are vital to further advance the field.

In this manuscript, we introduce tradeSeq, a method and software package for **tra**jectory-based **d**ifferential **e**xpression analysis for **seq**uencing data. tradeSeq provides a flexible framework that can be used downstream of any dimensionality reduction and TI method. Unlike previously proposed approaches, tradeSeq provides several tests that each identify a distinct type of differential expression pattern along a lineage or between lineages, leading to clear interpretation of the results. In practice, tradeSeq infers smooth functions for the gene expression measures along pseudotime for each lineage using generalized additive models and tests biologically meaningful hypotheses based on parameters of these smoothers. By allowing cell-level weights for each individual count in the gene-by-cell expression matrix, tradeSeq can handle zero inflation, which is essential for dealing with dropouts in full-length scRNA-seq protocols [156].

As it is agnostic to the dimensionality reduction and TI methodology, the approach scales from simple to complex trajectories with multiple bifurcations: tradeSeq only requires the

original expression count matrix of the individual cells, estimated pseudotimes, and a hard or soft assignment (weights) of the cells to the lineages to infer the lineage-specific smoothers. For within-lineage differential expression, tradeSeq provides both global tests to screen for genes with overall DE along a lineage, as well as specific tests to pinpoint relevant variation in gene expression profiles within the lineage. Likewise, for between-lineage comparisons, tradeSeq provides both global tests to compare expression patterns between entire lineages (useful for initial screening of interesting genes), as well as specific tests that allow researchers to pinpoint relevant differences in expression profiles between lineages. If multiple hypotheses are assessed for each gene, one can use our stageR package [157] to conduct an omnibus test (e.g., there are no differences in expression profiles across multiple lineages) prior to *post hoc* tests that identify the relevant specific differences (e.g., all pairwise comparisons between lineages). We benchmark our method against current state-of-the-art methods using simulated datasets (with cyclic, bifurcating, and multifurcating trajectories) and demonstrate its functionality and versatility on four real datasets. These case studies highlight the enhanced interpretability of tradeSeq's results, which lead to improved understanding of the underlying biology.

3.2 Results

In this Section, we first evaluate tradeSeq on simulated datasets with trajectories that span different topologies. Next, we demonstrate how tradeSeq can also be applied to bulk RNA-seq time course data, and how the method improves biological interpretation of trajectory inference results by applying it to three real datasets, a MARS-Seq dataset for mouse bone marrow [108], a SMART-Seq dataset for the mouse olfactory epithelium [44] and a 10X Genomics dataset on adipocyte differentiation [95].

Simulation study

To benchmark relevant differential expression methods, we generated multiple datasets, spanning three distinct trajectory topologies, using the independently developed dynverse toolbox [126]. We first generated 10 datasets (see Figure 3.1a for a representative dataset) corresponding to a single lineage contributing to a cyclic trajectory. Next, we considered a bifurcating topology, where a common lineage bifurcates into two differentiating lineages, and likewise generated 10 datasets (see Figure 3.1b for a representative dataset). Finally, we considered a multifurcating topology (Figure 3.1c). Note that the simulated datasets are relatively "clean", as reflected by the high sensitivity and specificity of most methods. In particular, cells are approximately uniformly distributed along each lineage and often balanced between lineages. The datasets are, however, still useful to provide a relative ranking of the methods.

We demonstrate the versatility of tradeSeq by using it downstream of three trajectory inference methods, slingshot [141], Monocle 2 [115], and GPfates [84], which will be denoted

by tradeSeq_slingshot, tradeSeq_Monocle2, and tradeSeq_Gpfates, respectively. However, we find that GPfates fails to recover the expected trajectory topology if run in an unsupervised way (Supplementary Figure C.3a). Feeding the true pseudotimes as input to GPfates may, however, result in meaningful trajectories (Supplementary Figure C.3b). We therefore adopt this approach in the simulation study, but note that this may provide an *a priori* competitive advantage to GPfates over other TI methods and that this would be impossible for real datasets.

Existing frameworks for differential expression analysis are not modular, in the sense that the DE method is tied to the TI method implemented in the same software package. Because of this, the comparison of DE methods is confounded with the quality of the upstream trajectory inference. We therefore also evaluate all trajectory-based DE methods by using the simulation ground truth as input for the DE analysis, which avoids such a confounding. GPfates was left out of this comparison, since we were not able to input the simulation ground truth to the method.

Within-lineage DE. First, we look for genes whose expression is associated with pseudotime for datasets with a cyclic topology (e.g., Figure 3.1a). We compare the `associationTest` of a tradeSeq_slingshot analysis to the Moran's I test implemented in Monocle 3. We apply tradeSeq using 5 knots, as determined using the AIC (Supplementary Figure C.4). We only consider Monocle 3 because it is the only method that provides a test to assess the association between gene expression and pseudotime within a single lineage. For each TI method, we use the default/recommended dimensionality reduction method, which is PCA for slingshot and UMAP for Monocle 3.

Monocle 3, however, often fails to reconstruct the cyclic topology and instead may fit a disconnected or branching trajectory (Supplementary Figure C.5). The Moran's I test still has reasonably high sensitivity, possibly because it relies on nearest neighbors in the reduced dimensional space and not on the inferred trajectory. tradeSeq downstream of slingshot provides superior performance to discover genes whose expression is associated with pseudotime (Figure 3.1d). We also compared both methods using the same dimensionality reduction input, by having slingshot infer trajectories in the UMAP space that is used by Monocle 3. The performance of tradeSeq was generally similar for both dimensionality reduction methods, except for 2 out of 10 datasets (Supplementary Figure C.6). In all datasets, tradeSeq had better performance than Monocle 3. Finally, we evaluate an edgeR-based `associationTest` through fitting the NB-GAMs with edgeR instead of with mgcv (method edgeR_assoc, see Methods for details), and note that its performance is similar to the tradeSeq `associationTest` (Supplementary Figure C.7). This could be expected because few basis functions were selected for this simulation setting. In applications that require a rich basis, however, the edgeR implementation will be prone to overfitting.

Between-lineage DE. For the bifurcating datasets (Figure 3.1b), we assess differential expression between lineages using the `diffEndTest` and the `patternTest` from tradeSeq,

downstream of TI methods slingshot, Monocle 2, and GPfates. We apply tradeSeq with 4 knots, as determined using the AIC (Supplementary Figure C.8). We compare these tests with available approaches for trajectory-based differential expression analysis, namely, BEAM (implemented in Monocle 2), GPfates, and ImpulseDE2. Furthermore, we compare against the discrete DE method edgeR, where we supervise the test to assess DE between the clusters at the true endpoints of each lineage, as derived through k-means clustering in PCA space. For each TI method, we use the default/recommended dimensionality reduction, which is PCA for slingshot, GPLVM for GPfates, and DDRTree for Monocle 2. For ImpulseDE2, we use the same input as for tradeSeq, i.e., derived by slingshot TI.

Monocle 2 and GPfates fail to detect the correct topology of the trajectory (i.e., a bifurcation) in, respectively, 3 and 4 out of the 10 datasets (Supplementary Figures C.9 and C.10). In addition, out of the remaining 7 datasets, Monocle 2 misplaces the bifurcation in 4 of them, causing the two simulated lineages to be merged into the same lineage and creating another incorrect lineage (Supplementary Figure C.9). This strongly obscures the DE testing results. slingshot, on the other hand, correctly identifies the topology and reconstructs the trajectory for all 10 datasets.

Figure 3.1e shows performance curves for the three datasets for which all methods are able to recover the true topology of the simulated trajectory. The tradeSeq `patternTest` has superior performance regardless of the TI method. Only edgeR achieves a similar performance. This is not surprising since the edgeR analysis is supervised to compare the true cell populations at the endpoints of the lineages. Interestingly, tradeSeq's `diffEndTest` based on the slingshot trajectory performs comparably to a supervised edgeR analysis. This is especially encouraging, since the `diffEndTest` is a smoother-based analog of discrete DE. For TI methods Monocle 2 and GPfates, `diffEndTest` performs poorly, which is not surprising since the endpoints are typically ill-defined or artificially extended in the inferred trajectories for those methods (Supplementary Figures C.9 and C.10). In general, BEAM, ImpulseDE2, and GPfates are outperformed by the other methods. Across all methods, tradeSeq_slingshot has the best performance. Finally, we recapitulate that the performance curves in Figure 3.1e do not provide a complete view of method performance, since 7 out of 10 datasets were not used because at least one method failed to recover the simulated trajectory. Supplementary Figure C.11 shows mean performance curves across all 10 datasets for all methods, which clearly demonstrates the superiority of tradeSeq as a DE method and of slingshot as an upstream TI method. The performance and trajectories for all 10 individual datasets are shown in Supplementary Figure C.12.

In order to avoid the comparison of DE methods being obscured by differences in the upstream dimensionality reduction and trajectory inference methods, we compared tradeSeq, BEAM, and ImpulseDE2 on the simulation ground truth. We fit the tradeSeq NB-GAM once with 3 knots, for comparability with the BEAM approach that also uses 3 knots, and once with 4 knots, which was found to be optimal according to the AIC (Supplementary Figure C.8). The tradeSeq `patternTest` is unaffected by the change in the number of knots and outperforms all other methods for differential expression analysis (Supplementary Figure C.13). The performance of the tradeSeq `diffEndTest` is somewhat sensitive to the number of knots,

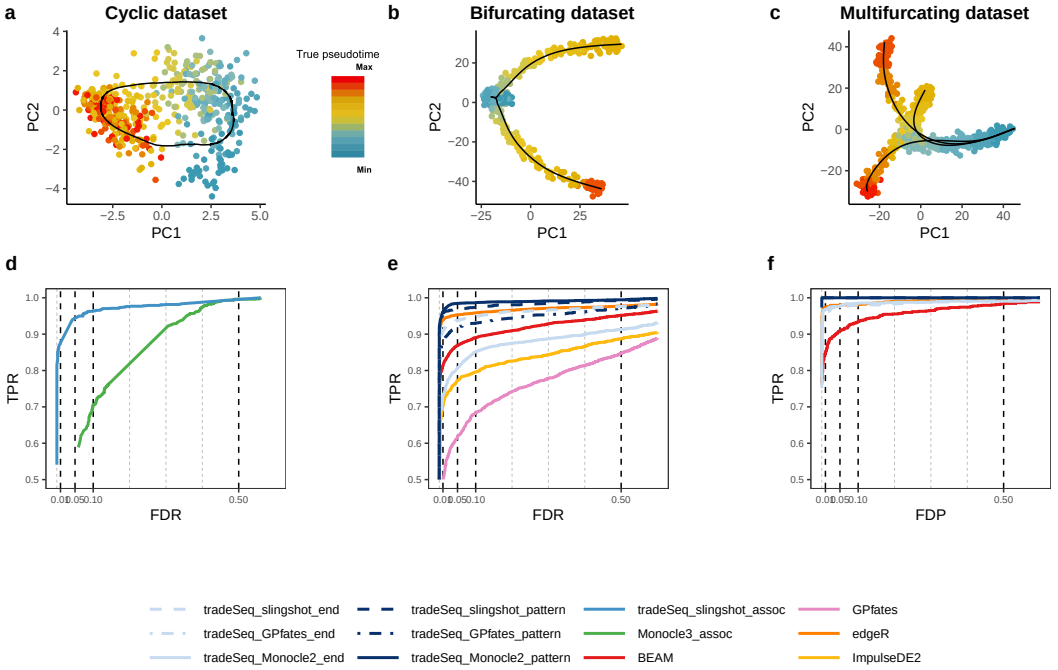


Figure 3.1: *Simulation study results.* PCA plots for the **(a)** cyclic, **(b)** bifurcating, and **(c)** multifurcating simulated trajectories. The plotting symbol for each cell is colored according to its true pseudotime; trajectories (in black) were inferred by princurve in (a) and slingshot in (b) and (c). **(d-f)** Scatterplot of the true positive rate (TPR) vs. the false discovery rate (FDR) or false discovery proportion (FDP) for various DE methods applied to the simulated datasets. Panel (d) displays the average performance curves of DE methods across seven out of 10 cyclic datasets for which all DE methods worked (Monocle 3 errored on three datasets). The `associationTest` from tradeSeq has superior performance for discovering genes whose expression is associated with pseudotime, as compared to Monocle 3. When investigating differential expression between lineages of a trajectory, the `patternTest` of tradeSeq consistently outperforms the `diffEndTest` across all three TI methods, since it is capable of comparing expression across entire lineages. Panel (e) displays the average performance curves across the three bifurcating datasets where all TI methods recovered the correct topology. Here, all tradeSeq `patternTest` workflows, tradeSeq_slingshot_end, and edgeR have similar performance and all are superior to BEAM, ImpulseDE2, and GPfates. Note that the performance of tradeSeq_Monocle2_end deteriorates as compared to tradeSeq_slingshot_end; the curve for tradeSeq_GPfates_end is not visible in this panel due to its low performance. For the multifurcating dataset of panel (f), tradeSeq_slingshot has the highest performance, closely followed by tradeSeq_Monocle2 and edgeR.

but still better than that of ImpulseDE2 and BEAM. Generally, ImpulseDE2 performs better than the BEAM approach.

For the multifurcating dataset, we forego a comparison with GPfates, since it is restricted to discovering only a single bifurcation (Supplementary Figure C.14). We fit tradeSeq with 3 knots, as determined using the AIC (Supplementary Figure C.15). The `patternTest` from tradeSeq_slingshot and tradeSeq_Monocle2 have highest performance, closely followed by edgeR and the `diffEndTest` for those respective TI methods (Figure 3.1f). BEAM was found to have the lowest performance.

Taken together, these results suggest that tradeSeq is a powerful and flexible procedure for assessing DE along and between lineages. Although tradeSeq is modular and can be used downstream of any TI method that provides pseudotime estimates, the choice of dimensionality reduction and TI method is crucial for the performance of the downstream analysis. The best performance was found for a tradeSeq_slingshot analysis, so we will mainly focus on slingshot as TI method for the real datasets.

Computation time and memory usage benchmark

To assess time and memory requirements, scRNA-seq datasets with a bifurcating trajectory were simulated using the same simulation framework as in the simulation study. Three datasets with 100, 1,000, and 10,000 cells were simulated (small, medium, and large datasets), each consisting of 5,000 genes. For BEAM and GPfates, only the fitting and DE testing part was assessed for each method, not the trajectory inference part. All methods were ran with default options. For tradeSeq, the `fitGAM` function was assessed with 4 knots, as determined in the simulation study. The different tests implemented in tradeSeq were benchmarked separately (Supplementary Figure C.16). Their running times are very small (always below 30 seconds), as compared to the `fitGAM` function, and do not increase for datasets with increasing numbers of cells.

To benchmark time requirements, the microbenchmark package was used, and each method was run 10, 2, and 2 times on respectively the small, medium, and large datasets. Variations in running times were very small (always under the minute), especially in comparison to between methods differences. Methods that reached the 4-hour mark without finishing were killed. This is the case for ImpulseDE2 on the datasets with 10^3 and 10^4 cells, and for GPfates on the largest dataset. Memory benchmark was assesses using the `Rprof` function; maximum memory usage was recorded.

Results are shown in Figure 3.2. ImpulseDE2 is by far the slowest, taking over 3.5 hours to finish on a small dataset of 100 cells. GPfates runs in about $30s$ on the small dataset but scales poorly. BEAM, edgeR and tradeSeq are quite fast and scale very well, even to large datasets, with BEAM scaling the best. In terms of memory requirements, all methods scale well to 10,000 cells. It should also be noted that tradeSeq, ImpulseDE2, and BEAM can utilize multiple cores but were benchmarked using only one core.

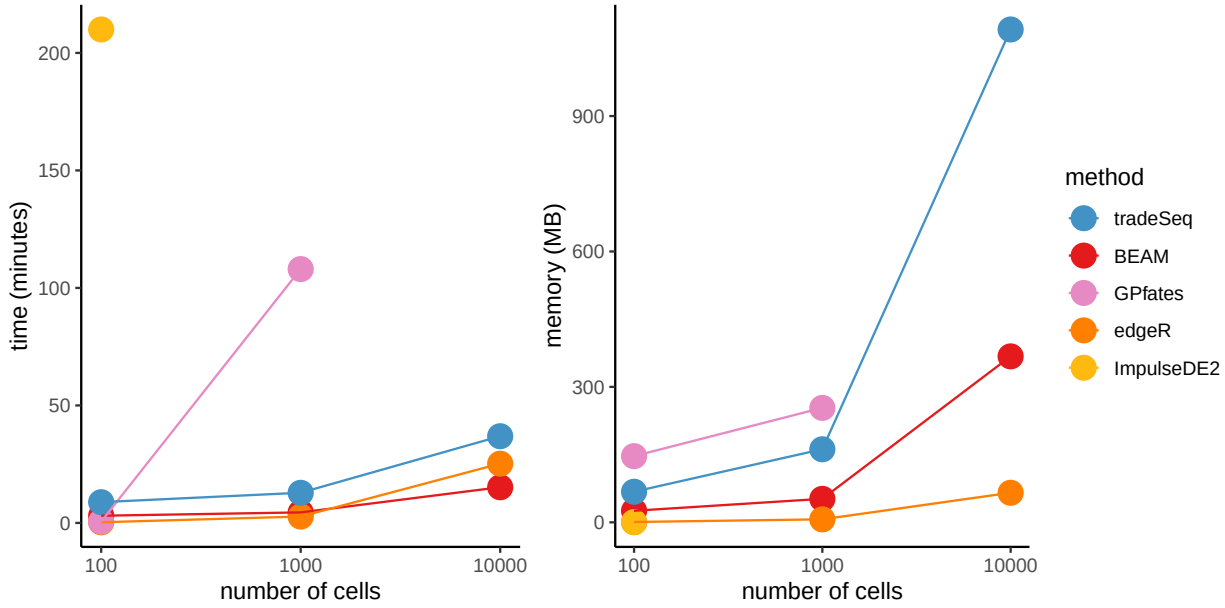


Figure 3.2: *Benchmark of computation time and memory usage.* Datasets with 100, 1,000 and 10,000 cells are simulated and each method is evaluated respectively 10, 2 and 2 times on each dataset to assess computation time and memory usage. The average across iterations is plotted for each method. Methods that went over a 4-hour mark were stopped and deemed taking too long.

Case studies

Bulk RNA-seq time-course dataset

While in this manuscript we focus on DE analysis downstream of TI, the applicability of tradeSeq extends beyond this setting. We demonstrate this by using tradeSeq on a bulk RNA-seq time-course study from Kiselev et al. [72], where we compare gene expression between wild type and PIK3CA H1047R cell lines upon stimulation of epidermal growth factor (EGF). Gene expression was measured for three replicates in each condition over over six time-points, ranging from 0 to 300 minutes post EGF stimulation. The original analysis in the manuscript assessed DE between the cell lines for each time-point separately using DESeq2, and found 7,486 DE genes at a 1% nominal FDR level (Benjamini, Yoav ; Hochberg [13] FDR-controlling procedure). We perform an analogous analysis using tradeSeq, by modeling gene expression measures as smooth functions of time and looking for differences in expression patterns with `patternTest`. This yields 7,184 DE genes at a 1% nominal FDR level. Around 89% of these genes overlap with the original DE list of 7,486 genes, demonstrating that the utility of tradeSeq goes beyond scRNA-seq applications.

Mouse bone marrow dataset

Paul et al. [108] study the evolution of gene expression for myeloid progenitors in mouse bone marrow. They construct a reference compendium of marker genes that are indicative of development from myeloid multipotent progenitors to erythrocytes and several types of leukocytes.

In order to compare our approach with BEAM, we are restricted to the dimensionality reduction procedures implemented in Monocle 2. We therefore first used ICA as dimensionality reduction method (Figure 3.3a) in the 'Discovering cell type markers' paragraph, but observed that this approach does not fully preserve the underlying biology. Indeed, a 2-dimensional visualization of the ICA dimensionality reduction shows that there is a seemingly large gap between the multipotent progenitors and the remaining cell types, and that a number of erythrocytes and granulocyte-macrophage progenitors (GMP) are misclassified as multipotent progenitors. In addition, megakaryocytes, which are thrombocyte progenitors and as such should not belong to any of the two lineages, seem to be split between the erythrocyte and leukocyte lineages. However, when applying UMAP dimensionality reduction (Figure 3.3b), these issues are resolved and the underlying biology seems better preserved than with ICA. In subsequent sections, we will therefore demonstrate the powerful interpretation of a tradeSeq_slingshot analysis based on UMAP dimensionality reduction. This additionally illustrates the flexibility of tradeSeq (and slingshot) to be applied downstream of any dimensionality reduction method.

In this case study, we apply tradeSeq with 6 knots, as found to be optimal by the AIC (Supplementary Figure C.17). We first identify marker genes for the progenitor and differentiated cell types in the 'Discovering cell type markers' paragraph. Next, we assess which genes behave differently along the two lineages in the 'Discovering progenitor population markers' paragraph. Finally, we demonstrate how one can group genes in clusters that share similar expression patterns in the 'Gene expression families' paragraph.

Discovering cell type markers. tradeSeq provides the flexibility to test several interesting and distinct hypotheses for this dataset, that cannot always be considered with other methods. For instance, we can find marker genes for the progenitor cell population vs. the differentiated leukocytes or erythrocytes with the `startVsEndTest` procedure (results shown in Supplementary Figure C.18). We can also discover marker genes for the differentiated cell types by comparing the differentiated leukocyte and erythrocyte cells themselves by contrasting the endpoints of the smoothers with the `diffEndTest` procedure. For the latter, tradeSeq finds $2,233$ significantly differentially expressed genes at a 5% nominal FDR level, while BEAM discovers 584 genes at a 5% nominal FDR level when testing whether the association between gene expression and pseudotime depends on the lineage (Benjamini, Yoav ; Hochberg [13] FDR-controlling procedure).

Since the identification of a larger set of DE genes does not necessarily imply more relevant biology, we select carefully constructed gene sets from Graaf et al. [47] to perform gene set enrichment analysis (GSEA) on blood cell types. As we are comparing erythrocytes

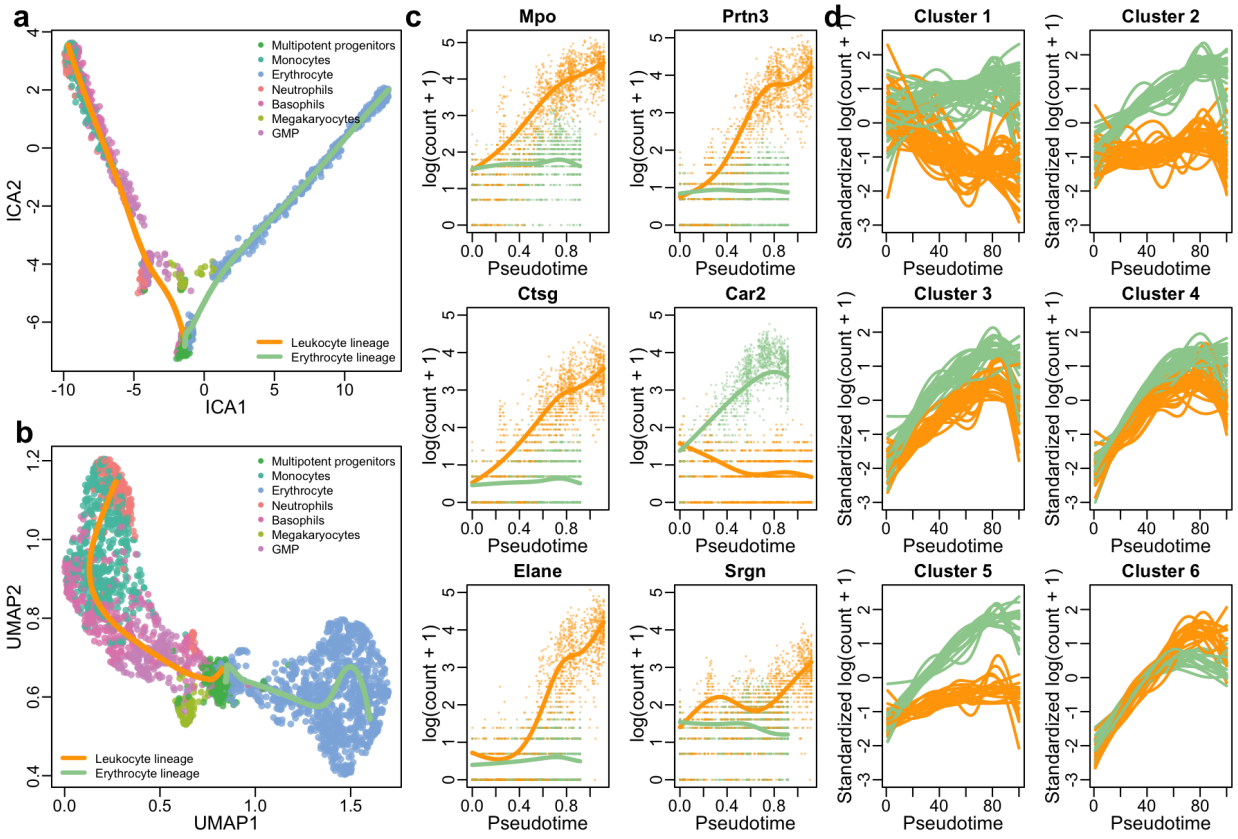


Figure 3.3: *Mouse bone marrow case study [108].* **(a)** Two-dimensional representation of a subset of the data using independent components analysis (ICA). The myeloid trajectory inferred by slingshot is displayed. **(b)** Two-dimensional representation of a subset of the data using UMAP. The myeloid trajectory inferred by slingshot is displayed. The UMAP dimensionality reduction method better captures the smooth differentiation process than ICA. **(c)** Estimated smoothers for the top six genes identified by the tradeSeq `patternTest` procedure on the trajectory from **(b)**. **(d)** Six clusters for the top 500 genes with different expression patterns between the two lineages (as identified by `patternTest` from tradeSeq).

with a mixture of leukocytes, we expect gene sets related to erythrocytes to be significant. Indeed, the erythrocyte gene set is the only one to be found significant by fgsea [132] for the tradeSeq analysis (FDR adjusted p-value $< .001$, with normalized enrichment score of 1.49), while no significant gene sets are found for the BEAM analysis (as reference, the FDR adjusted p-value for the erythrocyte gene set is 0.58). In this case, tradeSeq is therefore better able to recover a meaningful biological signal (Supplementary Figure C.19).

If one assumes that the cell type labels are known for all cells in the dataset, a cluster-based comparison is possible, where the different clusters correspond to the identified cell

types. We use edgeR [91] to assess differential expression between erythrocytes and neutrophils, since this comparison is most analogous to tradeSeq's `diffEndTest`. Only edgeR finds evidence for gene sets related to eosinophils and T-cells (FDR adjusted p-values of 0.042 and 0.049, respectively), however, the eosinophil cells were removed from this dataset prior to analysis (see Methods, subsection 'Case studies: Mouse bone marrow dataset'). The GSEA results for edgeR also provide less evidence for erythrocytes (FDR adjusted p-value = 0.043, normalized enrichment score=1.21) as compared to the tradeSeq analysis (Supplementary Figure C.19). None of the methods, however, recover evidence for the neutrophil cell types that are identified at the end of the lineage (Figure 3.3b; FDR adjusted p-values $p_{\text{tradeSeq}} = 0.99$, $p_{\text{BEAM}} = 0.86$, and $p_{\text{edgeR}} = 0.81$).
A tradeSeq analysis can thus provide relevant biological results without using the cell type labels. Moreover, while a cluster-based comparison can be powerful in some cases, many hypotheses are difficult to assess with discrete DE, as we demonstrate in the following paragraphs.

Discovering progenitor population markers. In addition to looking for markers at the differentiated cell type level, we could also look for markers of developing myeloid cells. tradeSeq can accommodate this by identifying genes with significantly different expression patterns between lineages. Remarkably, the top six genes (*Mpo*, *Prtn3*, *Ctsg*, *Car2*, *Elane*, and *Srgn*, Figure 3.3b) are all confirmed as biomarkers in the extensive analysis of the original manuscript of Paul et al. [108], confirming the relevant ranking of `patternTest` in tradeSeq. Indeed, *Prtn3* was found to be monocyte-specific, while *Mpo* and *Car2* discriminated between erythroid lineage progenitors and myeloid lineage progenitors. The cluster of genes *Elane*, *Prtn3*, and *Mpo* were the strongest markers for myeloid lineage progenitors and monocytes. In summary, all six top genes were labelled as "key genes" for hematopoiesis [108].

It might also be interesting to examine genes with significantly different expression patterns, that show little evidence for DE at the endpoints. We therefore select genes with both a high Wald test statistic (low p-value) for the `patternTest` and a low test statistic (high p-value) for the `diffEndTest`. Following the approach described in 'Case studies: Mouse bone marrow dataset' (Methods), we assign a score for each gene. Remarkably, within the top eight genes, four genes (*Erp29*, *Irf8*, *Psap*, and *ApoE*) were previously found to be major regulators of hematopoiesis. Indeed, *Irf8* has previously been identified as a major transcription factor involved in myeloid lineage commitment [76, 108]. *Erp29* and *Psap* are direct targets of the *Irf8* transcription factor [108, 136, 90], while *ApoE* regulates stem cell proliferation in atherosclerotic mice [98] and was also identified as a marker gene in the original manuscript of Paul et al. [108]. The remaining four genes include *Nedd4*, *Srrm2*, *Gatm*, and *Acin1*. Note that this analysis is not possible with any other method available, since these only test for global differential gene expression between lineages.

Gene expression families. Modeling gene expression in terms of smooth functions of pseudotime opens the door for additional downstream interpretation of results that are impossible with discrete DE methods, such as the clustering of genes based on their fitted expression patterns. In general, we found that RSEC clustering provides a more stable clustering than Partitioning around medoids (PAM) (Supplementary Figure C.20), the latter of which is also used by Monocle to cluster genes. For example, we can cluster the expression patterns for genes that were deemed significant by tradeSeq's `patternTest` (see Methods, section 'Clustering gene expression patterns'). This identifies gene families that have similar expression patterns within every lineage, and also similar fold-changes between the two lineages (Figure 3.3c shows six clusters). These gene sets can then be further screened for interesting patterns and validated by the biologist. Note that, for instance, the expression smoothers can be used to assess specific transient changes in expression during development, the signal for which might be diluted in cluster-based DE.

Mouse olfactory epithelium dataset

Fletcher et al. [44] study the development of horizontal basal cells (HBC) in the olfactory epithelium (OE) of mice. They activate the HBCs to be primed for development, which subsequently give rise to three different cell types: sustentacular cells, microvillous cells, and olfactory sensory neurons (Figure 3.4a,b). The olfactory sensory neurons are connected to the olfactory bulb for signal transduction of smell and the sustentacular cells are general supportive cells in the OE. The function of microvillous cells, however, is not well understood; while some cells have axons ranging to the olfactory bulb, potentially indicating a sensory neuron function, others lack a basal process or axon [54]. The samples from Fletcher et al. [44] were processed using the Fluidigm C1 system with SMART-Seq library preparation, hence we expect zero inflation to be present in this dataset. We therefore fit ZINB-GAMs to analyze the data using tradeSeq downstream of slingshot. Zero inflation weights are estimated with the ZINB-WaVE method [121], using the cluster labels and batch as covariates. We fit tradeSeq with 6 knots, as determined using the AIC (Supplementary Figure C.21). We were unable to fit a model for 0.8% of all 14,261 genes due to convergence issues of the ZINB-GAM. Note that, currently, no other trajectory-based DE method can account for zero inflation or provide the range of tests available in tradeSeq; hence, we forgo a comparison with other methods aside from a ZINB-edgeR analysis [156].
In this case study, we first consider differential expression within each lineage in the 'Within-lineage DE' paragraph, after which we assess differences between the three developmental lineages in the 'Between-lineage DE' paragraph.

Within-lineage DE. We first consider differential expression along the neuronal lineage (the orange lineage in Figure 3.4a). Using the `associationTest` implemented in tradeSeq, we recover 2,730 genes at a 5% nominal FDR level. Within the top DE genes, clear clusters of expression can be observed (Figure 3.4c), that are more active either at the beginning of the lineage, at specific locations along the lineage, or at the end of the lineage. Since

Fletcher et al. [44] observed that cells associated with the neuronal lineage undergo mitotic division during differentiation, we investigate whether we can recover the cell cycle biology using the `associationTest`. Indeed, many of the top genes are related to the cell cycle (Supplementary Figure C.22).

We also seek biological markers that differentiate the progenitor cells from the differentiated cell types in any of the three lineages using the `startVsEndTest` procedure as part of a global test (i.e., gene expression is compared between the start and end states for each lineage and the evidence is aggregated across the three lineages using a global test; see Methods) and then look for enriched gene sets for the top 250 genes. The results for the top 20 gene sets (Supplementary Table C.1) clearly reflect the biology of the experiment. The HBCs were primed for differentiation and the top gene sets include responses to (organic, external, and endogenous) stimuli. In addition, neurogenesis and tissue development are the first and third most significant gene sets, respectively, while the remaining list contains sets related to cell development, differentiation, and epithelium development, amongst others. Although the neuronal and microvillous cell lineages undergo mitotic division, it is worth noting that cell cycle related gene sets are absent from the `startVsEndTest` results, since this process occurs during differentiation, but not in the resting HBC or differentiated cell populations.

Between-lineage DE. We compare the three lineages by assessing differences in their expression patterns through stage-wise testing with the `patternTest` procedure (see Methods). At the screening stage, we first test whether any two lineages have significantly different expression patterns. The genes that pass the screening stage are then further assessed to discover which specific pair of lineages are deviating in their expression pattern. The screening stage identifies $3,275$ genes that have different expression patterns between any pair of lineages, at a 5% nominal FDR level (as reference, the top six genes are plotted in Supplementary Figure C.23). As could be expected, a large majority of the genes ($2,481$) are significant in the neuronal-sustentacular lineage comparison. However, remarkably, we discover more DE genes when comparing the microvillous and neuronal lineages ($2,149$ genes) than when comparing the microvillous and sustentacular lineages ($1,374$ genes), even though the microvillous lineage shares a longer path with the neuronal lineage. Out of all significant genes, 827 genes were identified in all three pairwise comparisons. Investigating the top 20 enriched gene sets based on the MSigDB database reveals that 12/20 of the top gene sets are related to the mitotic cell cycle (Supplementary Table C.2). This is reassuring, since only the neuronal and microvillous lineages go through the cell cycle, according to Fletcher et al. [44]. In addition, we find gene sets related to neurogenesis, referring to the development of olfactory sensory neurons. The functional interpretation of the results from the combined ZINB and tradeSeq analysis hence confirms the biology of the experiment and the battery of possible tests unlock a more detailed and meaningful interpretation of the results.

None of the previously developed trajectory-based methods for assessing differential expression between lineages can currently accommodate zero inflation. The only relevant comparison is between the `diffEndTest` procedure from tradeSeq and a discrete DE test

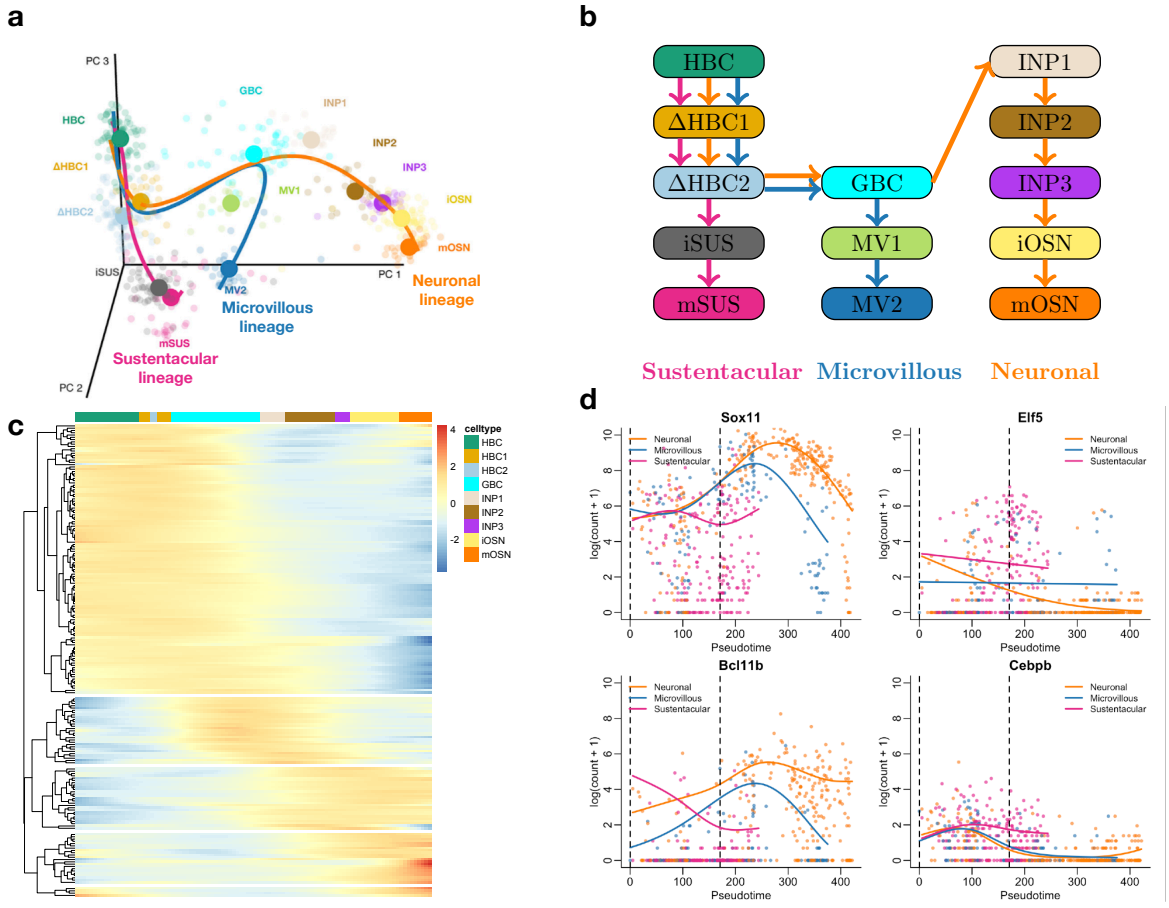


Figure 3.4: *Mouse olfactory epithelium case study [44].* **(a)** Three-dimensional PCA plot of the scRNA-seq data, where cells are colored according to their cluster membership as defined in the original paper (see Methods). The simultaneous principal curves for the lineages inferred by slingshot are displayed. **(b)** Schematic of the cell types and their ordering along the lineages. **(c)** Heatmap for the top 200 genes that are associated with the neuronal lineage, as identified with the `associationTest` procedure from tradeSeq. Five clear gene clusters can be identified, each with a different region of activity during the developmental process. **(d)** Four transcription factors involved in epithelial cell differentiation that are discovered by the `earlyDETest` between the pseudotimes of knots 1 and 3 (knots indicated with vertical dashed lines).

between the differentiated cell types using ZINB-edgeR as introduced in Van den Berge et al. [156]. For both methods, we use a global test to compare mean expression between all three differentiated cell types. While a ZINB-edgeR analysis discovers 1,984 genes, the ZINB-tradeSeq analysis discovers 3,719 genes, which include $\sim 86\%$ of the genes also discovered

by the ZINB-edgeR analysis. In order to assess the relevance of the extra 1,994 genes discovered with ZINB-tradeSeq, we perform GSEA on this gene set (Supplementary Table C.3). The top 20 significant gene sets contain relevant biological processes for the system under study, such as "regulation of multicellular organismal development", "positive regulation of biosynthetic process", and "tissue development".

We can also identify genes that drive the branching based on the `earlyDETest` applied around the first branching point, i.e., between knots 1 and 3 (see Supplementary Figure C.24 and Figure 3.4d). We apply stage-wise testing [157] (see Methods) to first assess any difference across the three lineages using a global test. We discover 2,083 genes to be DE between any of the three lineages at a 5% nominal FDR level. Among those 2,083 genes, we then discover 634 significant genes between the neuronal and microvillous lineages, 1,068 significant genes between the microvillous and sustentacular lineages, and 1,312 significant genes between the neuronal and sustentacular lineages (Supplementary Figure C.25). In total, 151 genes are significant in all pairwise comparisons (Supplementary Figure C.25), and these genes may be potentially important regulators of the transcriptional program involved in olfactory epithelium development. In these early stages of development, one could expect transcription factors to drive the differences between the three developmental lineages. Out of all 2,083 significant genes, we recover 84 transcription factors, as identified by TFcheckpoint [29]. Interestingly, aside from their general functionality as regulators of gene expression, the list of 84 transcription factors is enriched for gene sets related to epithelial cell differentiation, cell fate commitment, neuron differentiation, amongst other relevant gene sets (Supplementary Table C.4).

Adipocyte differentiation dataset

As final case study, we reanalyze a 10x Genomics scRNA-seq dataset from Merrick et al. [95], studying adipocyte differentiation from the developing sub-cutaneous inguinal white adipose tissue (iWAT) of 12-day-old mice. We use tradeSeq to fit NB-GAMs with 8 knots (Supplementary Figure C.26) based on the trajectory inferred by slingshot in 2D UMAP space. As in the original manuscript, the progenitor cells differentiate into two different cell populations (Supplementary Figure C.27). While we confirm *Dpp4+* and *Wnt2* as interstitial progenitor markers, we discover several other markers as top genes from our `startVsEndTest` procedure that are even more pronounced, e.g., *Pi16*, *Akr1c18*, *Fn1*, and *Fbn1* (Supplementary Figure C.28). In addition, we search for markers distinguishing between the two differentiated cell populations. Since these are relatively large heterogeneous groups of cells, `diffEndTest` is not representative for the entire set of cells. However, `earlyDETest` can be used to discover DE across their developmental range. This reveals several interesting patterns, such as genes upregulated in the adipocyte precursor stage and subsequently downregulated in only a single differentiated cell population (e.g., *Mgp* and *Meox2*; Supplementary Figure C.29), as well as genes that are sporadically highly expressed across the entire lineage for one of the two differentiated cell populations (e.g., *H19* and *Col14a1*; Supplementary Figure C.30)).

3.3 Discussion

We have proposed tradeSeq, a novel suite of tests for identifying dynamic temporal gene regulation using single-cell RNA-seq data. These tests allow researchers to investigate a range of hypotheses related to temporal gene expression, ranging from the general to the highly specific. Whereas previous methods only provide global tests of differential expression along or between lineages, tradeSeq offers a highly flexible framework that can be adapted to a single lineage, multiple lineages, or specific points or ranges along lineages. The flexibility provided by tradeSeq is crucial, as trajectory-based DE is often the final (or near final) step in a much longer analysis pipeline.

Our analyses are based on the NB-GAM of Equation (3.1) which conditions on cell pseudotimes and hence ignores the fact that pseudotimes are typically inferred random variables. We therefore expect some uncertainty in pseudotime values which may or may not be quantified by a particular TI method. Even when measures of pseudotime variability are available, neither tradeSeq nor other methods such as BEAM and GPFates currently make use of this information. Instead, all of these methods treat the pseudotimes as fixed and known. The BranchedGP method allows for uncertainty in the assignment of cells to lineages and relies on branching Gaussian processes to identify gene-specific branching dynamics [18]. However, it is computationally very intensive, with reported computation time of 2 minutes per gene on a dataset that has been subsampled to 467 cells [18]; we therefore did not consider this method in our evaluation.

While we generally assume that pseudotime values are on similar scales across lineages, this may not always be the case. Furthermore, Trapnell et al. [154] noted that any trajectory inference method can produce pseudotime values that are not necessarily reflective of true biological time. At best, pseudotime values represent some monotonic transformation of the true maturity of each cell. Therefore, some authors have proposed the use of dynamic time warping to align pseudotime values from different experiments on potentially different scales [1]. This approach can be beneficial in cases where, for example, one lineage is much longer or shorter than another. If a gene, in reality, has a similar pattern of expression along two such lineages, this pattern could, for instance, consume 75% of the shorter lineage, but only 25% of the longer lineage. As such, the gene could be called DE by the `patternTest` procedure. However, applying the same test after dynamic time warping may yield a negative result. Since tradeSeq only requires the estimated pseudotimes as input, which could be warped or not, it is compatible with any form of warping between lineages. We urge users to carefully consider whether pseudotime values across lineages are comparable and, if not, consider such warping strategies before comparing patterns of expression with tradeSeq.

Moving forward, it may be possible to fit ZINB-GAMs in a single step by numerically maximizing the ZINB-GAM likelihood. This could improve upon the two-step approach that we have taken in this paper, where (i) posterior probabilities of zero inflation are first estimated using ZINB-WaVE and (ii) subsequently used to unlock the NB-GAM for DE analysis in the presence of excess zeros.

In this manuscript, we have demonstrated tradeSeq on several scRNA-seq datasets. How-

ever, the tests that we provide downstream of the `fitGAM` function are applicable beyond this setting. Indeed, the framework may also be applicable to, e.g., downstream analysis of chromatin accessibility trajectories in scATAC-seq datasets (e.g., Chen et al. [30]) or bulk RNA-seq time-course studies; we have demonstrated the latter in the Results.

While we propose a number of tests based on the NB-GAM, it is important to realize that users may also implement their own statistical tests related to their specific hypotheses of interest. For example, it may be of interest to investigate whether the speed or acceleration in transcription varies significantly along or between lineages. This can be assessed in the tradeSeq framework using first or second derivatives of the linear predictor in Equation (3.1), respectively. The derivatives are linear combinations of the parameters in the basis function expansion, i.e., $\sum_{k=1}^{K} \beta_{glk} b'_k(t)$, where the derivatives of the basis functions $b'_k(t)$ often have a closed-form expression (e.g., cubic splines) or otherwise can be approximated using finite differencing (e.g., thin-plate splines) [172]. Genes that significantly increase (decrease) in their rate of expression along a lineage can then, for example, be discovered by testing whether the first derivative is significantly higher (lower) than zero. We therefore welcome contributions of new tests to the GitHub repository of the package.

Single-cell RNA-seq tends to produce noisy data requiring long analysis pipelines in order to glean biological insight. While "all-in-one" tools that simplify this analysis may be attractive from a user's standpoint, they are not guaranteed to offer the best methods for each individual step. We therefore propose a more modular approach that expands upon previous work and opens up new classes of questions to be asked and hypotheses to be tested.

3.4 Methods

In this Section, we first present a negative binomial generalized additive model for expression measures along a trajectory. Building on this model, we then describe a general and flexible framework for identifying genes that are differentially expressed either within or between lineages of a given trajectory.

Negative binomial generalized additive models

We build on the generalized additive model (GAM) methodology to model gene expression profiles as non-linear functions of pseudotime for the different lineages in a complex trajectory. In our GAM framework, each lineage is represented by a separate cubic smoothing spline, i.e., a linear combination of cubic basis functions of pseudotime. The flexibility of GAM also allows us to easily adjust for other covariates or confounders such as treatment and batch. The discrete nature and the over-dispersion of read counts is addressed by modeling the expression measures Y_{gi}, for a given gene $g \in \{1, \dots, G\}$ across cells $i \in \{1, \dots, n\}$, using a negative binomial (NB) distribution with cell and gene-specific means μ_{gi} and gene-specific dispersion parameters ϕ_g. Hence, we propose the following gene-wise negative binomial gen-

eralized additive model (NB-GAM)

$$\begin{cases} Y_{gi} & \sim & NB(\mu_{gi}, \phi_g) \\ \log(\mu_{gi}) & = & \eta_{gi} \\ \eta_{gi} & = & \sum_{l=1}^{L} s_{gl}(T_{li}) Z_{li} + \mathbf{U}_i \boldsymbol{\alpha}_g + \log(N_i), \end{cases} \tag{3.1}$$

where the mean μ_{gi} of the NB distribution is linked to the additive predictor η_{gi} using a logarithmic link function. The gene-wise additive predictor consists of lineage-specific smoothing splines s_{gl}, that are functions of pseudotime T_{li}, for lineages $l \in \{1, \ldots, L\}$. The binary matrix $\mathbf{Z} = (Z_{li} \in \{0, 1\} : l \in \{1, \ldots, L\}, i \in \{1, \ldots, n\})$ assigns every cell to a particular lineage based on user-supplied weights (e.g., from slingshot [141] or GPfates [84], see details in Supplementary Methods). We let $\mathcal{L}_l = \{i : Z_{li} = 1\}$ denote the set of cells assigned to lineage l. In addition, we allow the inclusion of p known cell-level covariates (e.g., batch, age, or gender), represented by an $n \times p$ matrix $\mathbf{U}$, with i^{th} row $\mathbf{U}_i$ corresponding to the i^{th} cell, and regression parameters $\boldsymbol{\alpha}_g$ of dimension $p \times 1$. Differences in sequencing depth or capture efficiency between cells are accounted for by cell-specific offsets N_i.

The smoothing spline s_{gl}, for a given gene g and lineage l, can be represented as a linear combination of K cubic basis functions,

$$s_{gl}(t) = \sum_{k=1}^{K} b_k(t) \beta_{glk},$$

where the cubic basis functions $b_k(t)$ are enforced to be the same for all genes and lineages. Our default computational implementation sets $K = 6$. Thus, for each gene and each lineage in the trajectory, we estimate $K = 6$ regression coefficients β_{glk}. The number of parameters in the gene-wise model is $L \times K + p + 1$, which is typically much lower than the number of cells n in the dataset.

The NB-GAM is fitted gene by gene using the `fitGAM` function from the tradeSeq package, which relies on the mgcv package in R. We build upon recent developments in mgcv that allow the joint estimation of the NB regression parameters in μ_{gi} and dispersion parameter ϕ_g [174]. In order to control the smoothness of the spline, the coefficients β_{glk} are shrunken by substracting a penalty $\lambda_g \boldsymbol{\beta}_g^T \mathbf{S} \boldsymbol{\beta}_g$ from the log-likelihood function, where $\boldsymbol{\beta}_g$ denotes the concatenation of the L K-dimensional column vectors $\boldsymbol{\beta}_{gl}$ of lineage-specific smoother coefficients and $\mathbf{S}$ is an $(LK) \times (LK)$ diagonal matrix that indicates which coefficients in $\boldsymbol{\beta}_g$ are to be penalized. The magnitude of penalization is controlled by the smoothing parameter λ_g, which is selected using generalized cross-validation [172]. Note that we enforce identical basis functions between lineages, i.e., b_k does not depend on l, as well as identical smoothing parameter λ_g, in order to ensure that the smoothers are comparable across lineages.

Importantly, the model of Equation (3.1) can accommodate zero-inflated counts typical for full-length scRNA-seq protocols by using observation-level (i.e., cell-level) weights obtained, for instance, from the zero-inflated negative binomial (ZINB) approach of Van den Berge et al. [156] and Risso et al. [121].

Choosing an appropriate number of knots

Ideally, the number of knots K should be selected to reach an optimal bias-variance trade-off for the smoother, where one explains as much variability in the expression data as possible with only a few regression coefficients (see Supplementary Figure C.1). In practice, the number of knots K may be selected by evaluating the Akaike Information Criterion (AIC) using the `evaluateK` function implemented in tradeSeq. We have deliberately chosen the AIC as evaluation criterion, since the Bayesian Information Criterion (BIC) seemed to favor overly complex models (i.e., an excessively high number of knots). The knots are by default positioned according to the quantiles of the pseudotime values. For example, if a smoother is fit with 3 knots, then there will be a knot at the minimum, median, and maximum pseudotime values. The knots may be interpreted as relative markers of progress along the trajectory. However, it is important to realize that this might not necessarily linearly correlate with true chronological time.

Statistical inference

We propose a general and flexible testing framework for (linear combinations of) the parameters $\boldsymbol{\beta}_g$, which allows us to pinpoint specific types of differences in gene expression both within and between lineages; see Figure 3.6 for an overview. We first present the general approach and then detail the implementation and interpretation of specific DE tests.

All proposed DE procedures involve testing null hypotheses of the form $H_0 : \mathbf{C}^T\boldsymbol{\beta}_g = 0$ using Wald test statistics

$$W_g = \hat{\boldsymbol{\beta}}_g^T \mathbf{C}(\mathbf{C}^T\hat{\boldsymbol{\Sigma}}_{\hat{\boldsymbol{\beta}}_g}\mathbf{C})^{-1}\mathbf{C}^T\hat{\boldsymbol{\beta}}_g, \tag{3.2}$$

where $\hat{\boldsymbol{\beta}}_g$ denotes an estimator of $\boldsymbol{\beta}_g$, $\hat{\Sigma}_{\hat{\boldsymbol{\beta}}_g}$ represents an estimator of the covariance matrix $\boldsymbol{\Sigma}_{\hat{\boldsymbol{\beta}}_g}$ of $\hat{\boldsymbol{\beta}}_g$, and $\mathbf{C}$ is an $(LK) \times C$ matrix representing the C contrasts of interest for the DE test.

For each gene, we compute p-values based on the nominal chi-squared asymptotic null distribution of the Wald statistics (with degrees of freedom equal to the column-rank of $\mathbf{C}$). Rather than attaching strong probabilistic interpretations to the p-values (which, as in most RNA-seq applications, would involve a variety of hard-to-verify assumptions and would not necessarily add much value to the analysis), we view the p-values simply as useful numerical summaries for ranking the genes for further inspection. There are five tests currently implemented in the tradeSeq package, which are introduced in detail in the sections below. Figure 3.5 provides a visual overview of the scope of each test.

Within-lineage comparisons

`associationTest`. A relevant first question is whether gene expression is associated with pseudotime along a given lineage, i.e., whether the smoother is flat or varying along pseudotime. To address this question, the `associationTest` tests the null hypothesis that all

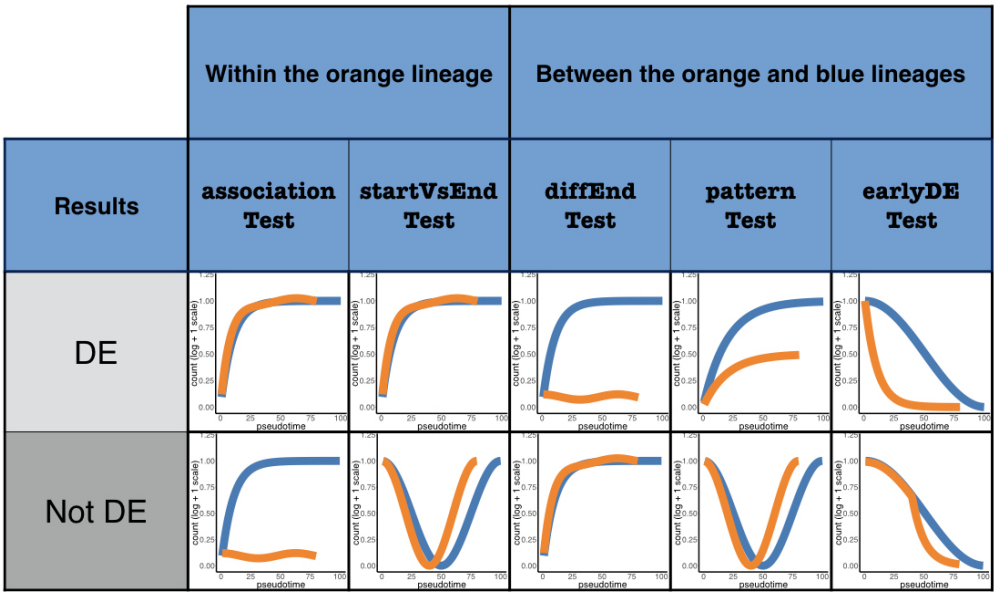


Figure 3.5: *Tests currently implemented in the* tradeSeq *package.* Each column corresponds to a test. Tests are broken down into two categories, depending on whether they concern a within-lineage comparison, i.e., properties of the orange curve, or a between-lineage comparison, i.e., contrasting the blue and orange curves. For each test, we have two toy examples of gene expression patterns. The top one corresponds to a differentially expressed gene according to the test, while the bottom one does not.

smoother coefficients within the lineage are equal, i.e., $H_0 : \beta_{glk} = \beta_{glk'}$ for all $k \neq k' \in \{1, \dots, K\}$. This null hypothesis can be encoded in several ways; here, we chose the contrast matrix $\mathbf{C}$ to be an $LK \times L(K-1)$ matrix, where each column corresponds to a contrast between two consecutive β_{glk} and $\beta_{gl(k+1)}$ and where we have $K-1$ contrasts per lineage for a total of $L(K-1)$ contrasts.

`startVsEndTest`. By default, the `startVsEndTest` compares mean expression at the progenitor state (i.e., the start of the lineage) to mean expression at the differentiated state (i.e., the end of the lineage). Specifically, $\mathbf{C}$ is an $(LK) \times L$ matrix, whose entry in row $k + (l-1)K$ and column l encodes the contrast for lineage l and knot k and is defined by $b_k\big(T_{l,max}\big) - b_k\big(T_{l,min}\big)$, where $T_{l,max} = \max_{\{i:i\in\mathcal{L}_l\}} T_{li}$ and $T_{l,min} = \min_{\{i:i\in\mathcal{L}_l\}} T_{li}$ denote, respectively, the maximum and minimum pseudotime across all cells assigned to lineage l. Other entries of $\mathbf{C}$ are set to zero. Therefore, the l^{th} element of the vector $\mathbf{C}^T\boldsymbol{\beta}_g$ is $\sum_{k=1}^{K}\big(b_k(T_{l,max}) - b_k(T_{l,min})\big)\boldsymbol{\beta}_{glk} = s_{gl}\big(T_{l,max}\big) - s_{gl}\big(T_{l,min}\big)$, which contrasts mean expression at the beginning and at the end of the lineage. Note that contrasting the start and

end points of a lineage is a special case of a more general capability of tradeSeq to compare the mean expression between any two regions of a given lineage. As such, this test can be considered a generalization of cluster-based discrete DE within a lineage (e.g., Risso et al. [121]).

Between-lineage comparisons

`diffEndTest`. The `diffEndTest` compares average expression at the differentiated states of multiple lineages, i.e., it compares the endpoints of different lineage-specific smoothers. It can be viewed as an analog of discrete DE for the differentiated cell types. The test is implemented using a Wald test statistic, as described above, where $\mathbf{C}$ is an $(LK) \times L(L-1)/2$ matrix. Each column of $\mathbf{C}$ encodes a pairwise contrast between the endpoints of two lineages, such that the corresponding element of $\mathbf{C}^T\boldsymbol{\beta}_g$ is $s_{gl_1}\big(T_{l_1,max}\big) - s_{gl_2}\big(T_{l_2,max}\big)$ for lineages l_1 and l_2.

`patternTest`. This test compares the expression patterns along pseudotime between lineages by contrasting a fixed set of equally-spaced pseudotimes ($M = 100$ by default). First selecting the pseudotimes and subsequently comparing their expression levels between lineages, allows for comparisons between smoothers of different lengths. Specifically, for lineage l, let P_{lm} denote the m^{th} equally-spaced pseudotime between $T_{l,\min}$ and $T_{l,\max}$. The contrast of M points corresponds to testing the null hypothesis that a gene has the same expression pattern along pseudotime across the lineages under comparison, while normalizing for the length of the lineages. The test is implemented using a Wald test statistic, as described above, where $\mathbf{C}$ is an $(LK) \times L(L-1)M/2$ matrix. Each column of $\mathbf{C}$ encodes a pairwise comparison between two pseudotimes of two different lineages, such that the corresponding element of $\mathbf{C}^T\boldsymbol{\beta}_g$ is $s_{gl_1}\big(P_{l_1m}\big) - s_{gl_2}\big(P_{l_2m}\big)$ for lineages l_1 and l_2 and $m \in \{1, \ldots, M\}$. The test is implemented through the eigendecomposition of the estimated variance-covariance matrix of the contrasts to avoid singularity problems [138] (see Supplementary Methods). It should be noted that this test is a general test, able to identify both differences in patterns of expression as well as genes with similar patterns but different mean expression across the pseudotime range. It is therefore most useful as a screening test to identify any form of differential expression between the lineages.

`earlyDETest`. The `earlyDETest` aims to identify genes that are differentiating around a branching of the trajectory. It is similar to the `patternTest`, in that it also compares the expression patterns along pseudotime between lineages by contrasting a fixed set of equally-spaced pseudotimes ($M = 100$ by default). However, instead of using points distributed from the beginning $T_{l,\min}$ to the end $T_{l,\max}$ of the lineages as in the `patternTest`, it relies on points over a shorter range of time. In the current implementation, this range is delimited by the pseudotimes of two user-specified knots. The knots should be chosen to enclose the branching event (or any event of interest) and do not need to be consecutive.

Global testing

While the statistical tests introduced above can assess DE within one lineage or between a pair of lineages, one may want to investigate multiple (i.e., more than two) lineages. For example, if a trajectory consists of three lineages, one may wish to test the global null hypothesis that, for each of the three lineages, there is no association between gene expression and pseudotime using the `associationTest`. The null hypothesis that would be tested can be expressed as $H_0 : \forall l$ and $\forall k \neq k'$, $\beta_{glk} = \beta_{glk'}$, i.e., within each of the three lineages, all K regression coefficients are equal. We refer to such a test as a "global test". The tradeSeq package provides functionality for global testing for each of the within and between-lineage tests described above. For within-lineage tests, the user can specify whether the test should be done for each lineage individually or at the global level (i.e., for all lineages). For between-lineage tests, the user can specify if a global test should be assessed or whether all pairwise comparisons should be performed.

Stage-wise testing

For the olfactory epithelium case study [44] detailed below, we apply stage-wise testing, as implemented in stageR [55, 157], to assess DE between lineages using multiple tests for each gene. Stage-wise testing aims to control the overall false discovery rate (OFDR) [55], i.e., the expected proportion of genes with at least one falsely rejected null hypothesis among all genes declared DE. In our case, the OFDR can be interpreted as a gene-level FDR [157]. Stage-wise testing is performed in two stages, a screening and a confirmation stage. At the screening stage, each gene is screened by performing a global test across all null hypotheses of interest, essentially testing whether at least one of these hypotheses can be rejected. At that stage, the FDR is controlled across genes at level α_I. At the confirmation stage, each specific hypothesis is assessed, but only for the genes that have passed the screening stage. For each gene, the family-wise error rate (FWER) is controlled across hypotheses at level $\alpha_{II} = \frac{R}{G}\alpha_I$, where R denotes the number of genes that had their global null hypothesis rejected at the screening stage and G the total number of genes assessed. Heller et al. [55] proved that this procedure controls the overall FDR at level α_I. It should be noted that, while the stage-wise testing paradigm theoretically controls the OFDR (given underlying assumptions are satisfied), the resulting p-values might still be too liberal since the same data are used for trajectory inference and differential expression. As mentioned before, we use p-values simply as numerical summaries for ranking the genes for further inspection.

Eigenvalue decomposition of $\hat{\mathbf{\Sigma}}_{\hat{\boldsymbol{\beta}}_g}$

Let $\mathbf{C}$ correspond to the $(LK) \times L(L-1)M/2$ matrix that defines the linear contrasts of interest for the `patternTest`, i.e., every column of $\mathbf{C}$ corresponds to the comparison of two points for a pair of lineages. Tests for the contrasts are performed using a Wald test statistic

defined as

$$W_g = \hat{\boldsymbol{\beta}}_g^T \mathbf{C} (\mathbf{C}^T \hat{\boldsymbol{\Sigma}}_{\hat{\boldsymbol{\beta}}_g} \mathbf{C})^{-1} \mathbf{C}^T \hat{\boldsymbol{\beta}}_g,$$

with $\hat{\boldsymbol{\Sigma}}_{\hat{\boldsymbol{\beta}}_g}$ the estimated variance-covariance matrix of the estimated smoother coefficients. Letting $\hat{\boldsymbol{\alpha}}_g = \hat{\boldsymbol{\beta}}_g^T \mathbf{C}$ and $\hat{\boldsymbol{\Sigma}}_{\hat{\boldsymbol{\alpha}}_g} = \mathbf{C}^T \hat{\boldsymbol{\Sigma}}_{\hat{\boldsymbol{\beta}}_g} \mathbf{C}$, we can rewrite the Wald statistic as

$$W_g \;=\; \hat{\boldsymbol{\alpha}}_g (\hat{\boldsymbol{\Sigma}}_{\hat{\boldsymbol{\alpha}}_g})^{-1} \hat{\boldsymbol{\alpha}}_g^T.$$

Taking the eigendecomposition of $\hat{\boldsymbol{\Sigma}}_{\hat{\boldsymbol{\alpha}}_g}$,

$$W_g \;=\; \hat{\boldsymbol{\alpha}}_g \hat{\mathbf{V}}_g^T \hat{\boldsymbol{\Lambda}}_g^{-1} \hat{\mathbf{V}}_g \hat{\boldsymbol{\alpha}}_g^T,$$

where $\hat{\mathbf{V}}_g$ is an $L(L-1)M/2 \times L(L-1)M/2$ matrix with columns corresponding to the $L(L-1)M/2$ eigenvectors of $\hat{\boldsymbol{\Sigma}}_{\hat{\boldsymbol{\alpha}}_g}$ and $\hat{\boldsymbol{\Lambda}}_g$ the $L(L-1)M/2 \times L(L-1)M/2$ diagonal matrix of eigenvalues $\lambda_i \in [0,1]$ of $\hat{\boldsymbol{\Sigma}}_{\hat{\boldsymbol{\alpha}}_g}$, in decreasing order. Note that, since $\hat{\boldsymbol{\Lambda}}_g$ is a diagonal matrix, we can simply invert its diagonal elements instead of inverting the full matrix. We determine the rank r of $\hat{\boldsymbol{\Sigma}}_{\hat{\boldsymbol{\alpha}}_g}$ by calculating the number of eigenvalues that are larger than $1e^{-8}\lambda_1$, with λ_1 corresponding to the largest eigenvalue. When the rank r of $\hat{\boldsymbol{\Sigma}}_{\hat{\boldsymbol{\alpha}}_g}$ is less than $L(L-1)M/2$ (i.e., the matrix is not of full rank), we only use the first r eigenvectors from $\hat{\mathbf{V}}_g$, associated with the largest r eigenvalues from $\hat{\boldsymbol{\Lambda}}_g$. This provides an efficient computation of the test statistic and avoids singularity problems with the estimated variance-covariance matrix of the contrasts [138].

Defining Z based on user-supplied weights

If one has user-supplied weights $\mathbf{W} = (W_{li} \in [0,1] : l \in \{1,\ldots,L\}, i \in \{1,\ldots,n\})$ for the assignment of cells to lineages, one can construct the binary matrix $\mathbf{Z}$ from $\mathbf{W}$ as follows.

First, note that the weights $\mathbf{W}$ may be defined differently depending on the TI method that was used to estimate them. For example, slingshot [141] defines weights based on the distance from a cell to a particular lineage; hence, the sum of the weights across all lineages for a particular cell may be greater than 1. As such, these weights cannot be interpreted as probabilities. GPfates [84], however, does return posterior probabilities that a cell i belongs to a particular lineage l, where $\sum_{l=1}^{L} W_{li} = 1$ for each i. We therefore first normalize the weights for each cell, such that, for normalized weights W_{li}^*, the sum across lineages equals one, i.e., $\sum_{l=1}^{L} W_{li}^* = 1$ for each cell i. Next, we assign each cell i to a lineage by sampling one observation from a Multinomial distribution with L groups and probabilities W_{li}^*. The lineage assignments are then encoded in the $L \times n$ matrix $\mathbf{Z}$, by setting all elements of the i^{th} column equal to zero except for a 1 in the row corresponding to the sampled lineage for cell i.

The multinomial sampling to assign each cell to a lineage may introduce variability in the results if the models are fit multiple times, due to differing cell allocations. This is especially so if there is a high uncertainty about the lineage allocation (e.g., a cell is equally likely to belong to each of two lineages), which typically occurs around the inception of a trajectory. While we ensure reproducibility by setting a seed in the software, the results may vary slightly over different seeds. To quantify this variability, we use the data of Paul et al. [108] and allocate cells to lineages using 10 different seeds. Since we expect the variability across different assignments to be largest at the inception of the lineage, we evaluate DE using tradeSeq's `startVsEndTest`. Using a global test across the two lineages, the number of DE genes at a 5% nominal FDR level varied between 1,990 and 2,049, with 1,739 DE genes shared across all 10 assignments (Supplementary Figure C.31). For each assignment, at least 93% of the top 1,000 DE genes are shared with the top 1,000 DE genes of any other assignment.

Clustering gene expression patterns

The NB-GAM can also be used to cluster genes according to their expression patterns, as shown in Figure 3.6. Specifically, for each gene, we extract a number of fitted values for each lineage (100 by default). We can then use resampling-based sequential ensemble clustering (RSEC), as implemented inclusterExperiment [121], to perform the clustering based on (the top principal components of) the standardized fitted values matrix (i.e., the fitted values are standardized to have zero mean and unit variance across cells for each gene). Importantly, we allow for any clustering algorithm that is built-in into clusterExperiment or chosen by the user to perform the clustering. This clustering approach is implemented in the tradeSeq package (`clusterExpressionPatterns` function) for downstream analysis facilitating the interpretation of DE genes.

Implementation

The above described fitting procedure, DE tests and clustering of expression patterns are implemented in the open-source R package tradeSeq, available through the Bioconductor Project (`http://www.bioconductor.org/packages/release/bioc/html/tradeSeq.html`). We provide an extensive vignette along with the package, as well as a cheat sheet describing the different types of DE patterns detected with each test.

Methods comparison

slingshot is a fast and robust method for TI that was shown to be among the top-performing methods in a recent large-scale benchmarking study [126]. Hence, we evaluate tradeSeq downstream of a slingshot analysis, which can work with any dimensionality reduction and clustering methods. slingshot builds a cluster-based minimum spanning tree (MST) to infer the global lineage topology and make an initial assignment of cells to lineages. This structure

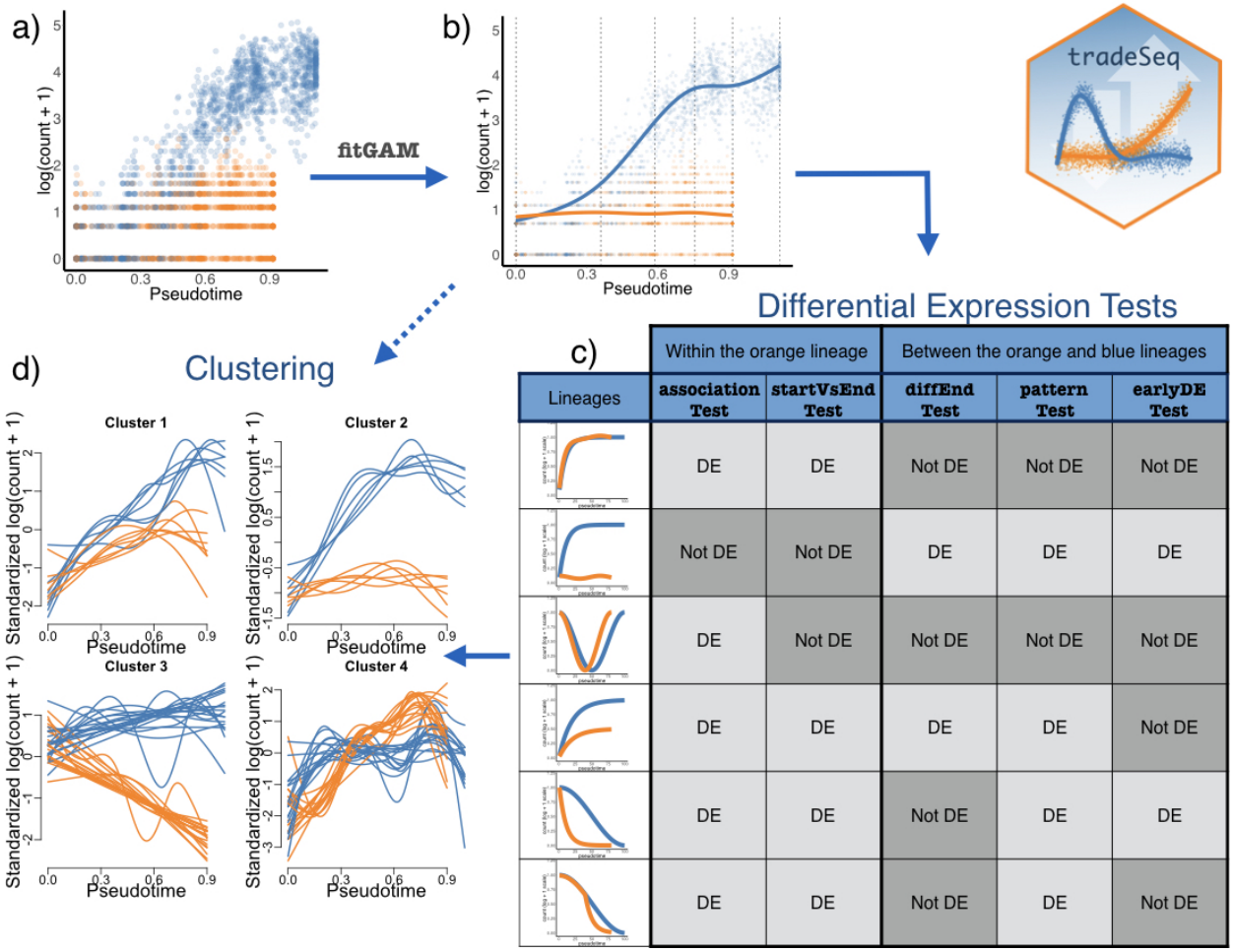


Figure 3.6: *Overview of tradeSeq functionality.* **(a)** A scatterplot of expression measures vs. pseudotimes for a single gene, where each lineage is represented by a different color (top left). **(b)** A NB-GAM is fitted using the `fitGAM` function. The locations of the knots for the splines are displayed with gray dashed vertical lines. **(c)** The NB-GAM can then be used to perform a variety of tests of differential expression within or between lineages. In the table, we assume that the `earlyDETest` is used to assess differences in expression patterns early on in the lineage, e.g., with option `knots = c(1, 2)`, meaning that we test for differential patterns between the first and second dashed grey lines from Panel (b). **(d)** Interesting genes can finally be clustered to display the different patterns.

is then smoothed by fitting simultaneous principal curves, which refine the assignment of cells to lineages. This process results in lineage-specific pseudotimes and weights of assignment for each cell.

GPfates [84] is a Python package that adopts Gaussian processes in reduced dimension to infer trajectories. Dimensionality reduction is performed using Gaussian process latent variable models (GPLVM) [78]. GPfates is able to identify bifurcation points and assess how well a bifurcation fits the expression pattern for every gene, i.e., whether the patterns of gene expression are different between the lineages. This allows us to compare a slingshot + tradeSeq analysis with a GPfates analysis. In addition, we also evaluate a tradeSeq analysis downstream of TI with GPfates, since GPfates also calculates posterior probabilities that each cell belongs to a particular lineage. We then compare the complete GPfates (TI and DE) analysis to a GPfates + tradeSeq analysis.

Monocle 2 [115] applies reverse graph embedding to infer trajectories and yields a principal graph that is allowed to branch. It provides a similar approach as tradeSeq with the branch expression analysis modeling (BEAM) method. It assumes a gene-wise negative binomial model for gene expression, where the mean is expressed in terms of lineage-dependent smooth functions of pseudotime, i.e.,

$$\log(\mu_{gi}) = \sum_{l=1}^{L} (\beta_{0gl} + s_{gl}(T_{li})). \tag{3.3}$$

In this model, the lineage-specific intercepts β_{0gl} account for mean differences in expression between lineages, while the lineage-specific smoothers $s_{gl}(t)$ model the expression change along pseudotime. To test for lineage-dependent expression, the full model is compared to a null model of the form

$$\log(\mu_{gi}) = \beta_{g0} + s_g(T_i)$$

using a likelihood ratio test. Thus, BEAM tests whether the smooth functions of gene expression along pseudotime are different between lineages. Importantly, the BEAM method is restricted to the dimensionality reduction methods that are implemented in Monocle 2, namely DDRTree [115] and Independent Components Analysis (ICA). Additionally, it only provides a screening test (like the `patternTest` in tradeSeq), as it only allows testing for any difference in expression profiles between lineages and does not specify the exact type of divergence.

An alpha release for Monocle 3 is available online (downloaded August 30, 2018 from the Monocle GitHub repository) which, unlike Monocle 2, performs uniform manifold approximation and projection [93] dimensionality reduction upstream of the trajectory inference. Additionally, Monocle 3 implements the Moran's I test to discover genes whose expression is significantly associated with pseudotime; a functionality that is unavailable in Monocle 2.

ImpulseDE2 [41] also assumes a gene-wise negative binomial model for the expression counts, where the mean is expressed as a weighted combination of two sigmoid functions. This model essentially allows the estimation of three "state-specific expression values", where

the transitions between the states are modeled with the two sigmoid functions. The DE method is not linked to any trajectory inference procedure since it assumes that the pseudotime for each cell is known. In this manuscript, we use ImpulseDE2 downstream of slingshot. Prior to the fitting, ImpulseDE2 relies on DESeq2 for normalization and estimation of the NB dispersion parameter. However, the DESeq2 procedure cannot handle datasets where each gene has at least one zero count, which is common in scRNA-seq. In such a scenario, we therefore "manually" estimate size factors and dispersion parameters using the DESeq2 `poscounts` normalization, which was developed to deal with this issue [94, 156].

edgeR [91] is a discrete differential expression method, where the groups under comparison must be defined *a priori*. It is therefore useful for assessing DE between, for example, annotated clusters or different treatment groups. For such comparisons, edgeR is a powerful method with high sensitivity. Note that, while edgeR was originally developed for group-based differential expression, it would be possible to incorporate the basis functions of the smoothers as continuous covariates in the model. However, no regularization would be performed on the estimation of the smoother regression coefficients, hence the model would be prone to overfitting. A similar approach was evaluated in Fischer, Theis, and Yosef [41], where DESeq2 [86] was used to fit splines by incorporating natural cubic basis functions in the linear predictor. Hence, while it is possible to fit smoothers by using edgeR, instead of mgcv, we emphasize that this would merely be an alternative approach to fitting the NB-GAMs we propose in our manuscript. Indeed, edgeR does not provide an implementation of the DE tests in tradeSeq. Only the `associationTest` is readily available in edgeR by testing whether all basis function parameters are equal to zero and the other tests would require a similar development as presented in tradeSeq.

Simulation study

The simulation study evaluates methods that (differentially) associate gene expression with pseudotime for three different trajectory topologies, i.e., a cyclic, a bifurcating, and a multifurcating trajectory. As independent evaluation, we use the extensive trajectory simulation framework dynverse that previously served for benchmarking trajectory inference methods in Saelens et al. [126]. Interested readers should refer to the original publication for details on the data simulation procedure. Dataset characteristics are listed in Table 3.1.

For each of the cyclic and bifurcating topologies, we generate and analyze 10 datasets. Since the multifurcating topology is very variable across simulations due to its flexible definition, its analysis requires substantial supervision. Therefore, we analyze only one representative multifurcating dataset.

Prior to trajectory inference, the simulated counts are normalized using full-quantile normalization [16, 21]. For TI with slingshot, we apply principal component analysis (PCA) dimensionality reduction to the normalized counts and k-means clustering in PCA space. For the bifurcating and multifurcating trajectories, the start and end clusters of the true trajectory are provided to slingshot to aid it in inferring the trajectory. For the edgeR analysis, we assess DE between the end clusters that are also provided to slingshot. The

Table 3.1: *Overview of simulated datasets.* Each dataset is simulated using one of the frameworks from the dynverse toolbox (dyngen or dyntoy), which are designed to simulate scRNA-seq data according to trajectory topologies. Every dataset can be characterized by the topology of the trajectory, as well as the number of cells and genes. Low-dimensional representations of representative datasets can be found in Figure 3.1. Note that the cyclic datasets have some variation in the numbers of genes and cells and in the amount of differential expression, which is inherent to the dyngen simulation framework.

	Cyclic dataset	**Bifurcating dataset**	**Multifurcating dataset**
Simulation framework	dyngen	dyntoy	dyntoy
Number of cells	$505 - 508$	500	750
Number of genes	$312 - 444$	$5,000$	5,000
% of DE genes	$42 - 47\%$	20%	20%
Number of lineages	1	2	3
Topology	Cyclic	Bifurcating	Multifurcating
Number of datasets	10	10	1

BEAM method can only test one bifurcation point at a time. For the multifurcating dataset, we therefore assessed both branching points separately and aggregated the p-values using Fisher's method [42]. For the tradeSeq and edgeR analyses of the multifurcating dataset, we perform global tests across all three lineages.

We assess performance based on scatterplots of the true positive rate (TPR) vs. the false discovery proportion (FDP), according to the following definitions

$$\begin{aligned} FDP &= \frac{FP}{\max(1, FP + TP)} \\ TPR &= \frac{TP}{TP + FN}, \end{aligned}$$

where FN, FP, and TP denote, respectively, the numbers of false negatives, false positives, and true positives. FDP-TPR curves are calculated and plotted with the Bioconductor R package iCOBRA [139].

Case studies

Bulk RNA-seq time-course dataset

As proof-of-principle case study, we analyze a bulk RNA-seq time-course dataset from Kiselev et al. [72] with tradeSeq. The data were downloaded from the GitHub repository at https://github.com/daniel-spies/rna-seq_tcComp, and the original differential ex-

pression results were downloaded from https://github.com/wikiselev/rnaseq.mcf10a/tree/master/data.

Mouse bone marrow dataset

We use as second case study the mouse haematopoiesis scRNA-seq dataset of Paul et al. [108]. Two small cell clusters corresponding to the dendritic and eosinophyl cell types were removed from the trajectory inference and downstream DE analysis, since these are outlying cell types that do not seem to belong to any particular lineage (Supplementary Figure C.2). We use the same dataset as the Monocle 3 vignette, which was prefiltered to contain genes with relatively high expression. After filtering, the dataset consists of $3,004$ genes and $2,660$ cells.

tradeSeq downstream of slingshot is compared to the BEAM approach from Monocle 2. Since BEAM is restricted to the dimensionality reduction methods implemented in the package, we use independent components analysis (ICA) for both slingshot and Monocle 2 in this comparison. For Monocle 2, we specify the argument `num_paths=2` to aid it in inferring two lineages.

Subsequently, we demonstrate a tradeSeq analysis downstream of slingshot by performing dimensionality reduction using UMAP [93], following the data processing pipeline described in the Monocle 3 vignette, since this better reflects the biology of the experiment.

In this case study, we show how one can perform multiple tests to identify genes with distinct types of behavior, specifically, genes that are deemed DE for one test (test 1) but not another (test 2). Let $W_g^{(\tau)}$ denote the test statistic for gene g in test $\tau \in \{1, 2\}$ and $\text{rk}_g^{(\tau)}$ denote the rank (in terms of ordering from low to high) of $W_g^{(\tau)}$ among all G test statistics associated with the G genes. Then, define a score for each gene g as $\text{score}_g = (\text{rk}_g^{(1)})^2 + (G - \text{rk}_g^{(2)})^2$. Genes with high scores are genes which are expected to be DE for test 1 but not DE for test 2 and *vice versa*. This is used to identify genes that are DE with the `patternTest` (test 1) but not the `diffEndTest` (test 2), i.e., genes that are transiently DE between lineages. Note that the procedure only provides a ranking of the genes and not an evaluation of statistical significance.

Mouse olfactory epithelium dataset

The olfactory epithelium (OE) dataset from Fletcher et al. [44] is our third case study. We use the lineages discovered in the original manuscript. Prior to the analysis, the dataset is filtered to retain genes with reasonably high expression, and we consider $14,261$ genes and 616 cells for downstream analysis. In brief, counts are normalized using full-quantile normalization [16, 21] followed by regression-based adjustment for quality control variables [44]. Dimensionality reduction is performed through PCA on the normalized log-transformed counts that are offset by 1 to avoid taking the log of zero, i.e., $\log(y + 1)$. Clustering is performed through k-means on the first 50 principal components by varying the number of clusters $k \in \{4, \ldots, 15\}$; stable clusters are derived using clusterExperiment [121], yielding a

final repertoire of 13 cell clusters. Next, slingshot is used to infer trajectories with the initial cluster chosen by known marker genes of horizontal basal cells (HBC), an adult stem cell population. A double bifurcation is discovered, with the first giving rise to sustentacular cells and two more lineages that split into microvillous cells and olfactory sensory neurons. The data were downloaded from GEO with accession number GSE95601.

Adipocyte differentiation dataset

As final case study, we reanalyze a 10x Genomics scRNA-seq dataset from Merrick et al. [95], studying adipocyte differentiation from the developing sub-cutaneous inguinal white adipose tissue (iWAT) of 12-day-old mice. The gene expression counts were downloaded from GEO with accession number GSE128889. We focus the analysis on the single cells collected from 12-day old mice. The raw dataset consists of $27,998$ genes and $11,423$ cells. We only retain genes with a count of at least 2 in at least 400 cells, and normalize the data using full quantile normalization [16]. Since not all cells in the dataset are involved in the adipocyte differentiation process, we first identify the relevant clusters of cells using the marker genes described in the original manuscript. We apply k-means clustering ($k = 10$) on the top 8 principal components of log-transformed counts. Using this clustering, we identify the relevant clusters based on the reported markers, and subsequently apply UMAP dimensionality reduction [11, 93] on the top 20 principal components for that subset of cells. The processed dataset consists of $2,851$ genes and $8,071$ cells. We use slingshot [141] for trajectory inference in 2-dimensional UMAP space.

We use tradeSeq to fit NB-GAMs with 8 knots (Supplementary Figure C.26) based on the trajectory inferred by slingshot in 2D UMAP space. As in the original manuscript, the progenitor cells differentiate into two different cell populations (Supplementary Figure C.27). While we confirm *Dpp4+* and *Wnt2* as interstitial progenitor markers, we discover several other markers as top genes from our `startVsEndTest` procedure that are even more pronounced, e.g., *Pi16*, *Akr1c18*, *Fn1*, and *Fbn1* (Supplementary Figure C.28). In addition, we search for markers distinguishing between the two differentiated cell populations. Since these are relatively large heterogeneous groups of cells, `diffEndTest` is not representative for the entire set of cells. However, `earlyDETest` can be used to discover DE across their developmental range. This reveals several interesting patterns, such as genes upregulated in the adipocyte precursor stage and subsequently downregulated in only a single differentiated cell population (e.g., *Mgp* and *Meox2*; Supplementary Figure C.29), as well as genes that are sporadically highly expressed across the entire lineage for one of the two differentiated cell populations (e.g., *H19* and *Col14a1*; Supplementary Figure C.30)).

Code availability

The code to reproduce the analyses, figures, and tables in the paper is available on GitHub at https://github.com/statOmics/tradeSeqPaper. The tradeSeq open-source R package is

available through the Bioconductor Project at http://www.bioconductor.org/packages/release/bioc/html/tradeSeq.html.

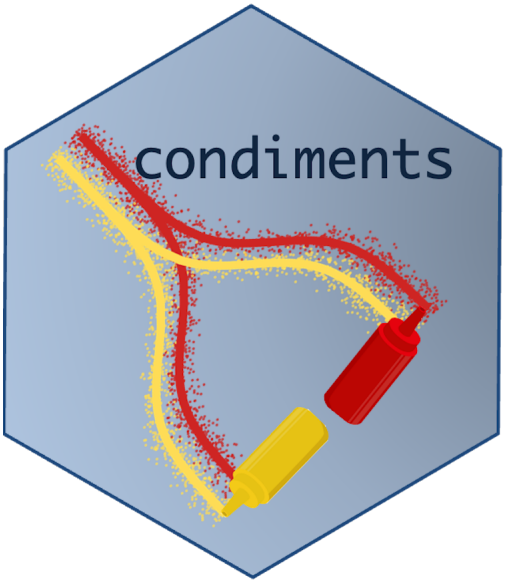
Data availability

The code to generate all simulated datasets is included in the GitHub repository of the paper at https://github.com/statOmics/tradeSeqPaper. The data for the mouse bone marrow case study were downloaded from http://trapnell-lab.gs.washington.edu/public_share/valid_subset_GSE72857_cds2.RDS. The raw data for the olfactory epithelium case study are available on GEO with accession number GSE95601.

Chapter 4

Trajectory inference across multiple conditions

The importance of trajectory inference was discussed in chapter 1. In this chapter, we will focus on how to perform a full trajectory inference analysis when the dynamic system is being studied under several conditions (e.g. treatment / control, or Wild-type / knockout). A version of this work has been released in preprint format [123] [1].



[1] I would link to deeply thank Koen Van den Berge and Kelly Street which were of crucial help in this project, as well as Sandrine Dudoit for her advises and supervision

4.1 Introduction

The emergence of RNA sequencing at the single-cell level (scRNA-Seq) has enabled a new degree of resolution in the study of cellular processes. The ability to consider biological processes as continuous phenomena instead of individual discrete stages has permitted a finer and more comprehensive understanding of dynamic processes such as embryogenesis and cellular differentiation. Trajectory inference was one of the first applications that leveraged this continuum [154] and a consequential number of methods have been proposed since then [141, 84, 65]. Saelens et al. [126] offer an extensive overview and comparison of such methods. Analysis of scRNA-Seq datasets using a curated database reveals that about half of all datasets were used for trajectory inference (TI) [145]. At its core, TI represents a dynamic process as a directed graph. Distinct paths along this graph are called lineages. Individual cells are then projected onto these lineages and the distance along each path is called pseudotime. In this setting, developmental processes are often represented in a tree structure, while cell cycles are represented as a loop. Following TI, other methods have been proposed to investigate differential expression (DE) along or between lineages, either as parts of TI methods [115, 84] or as separate modules that can be combined to create a full pipeline [158].

More recently, other methods have emerged to answer an orthogonal problem, focusing on systems under multiple conditions. This includes, for example, situations where a biological process is studied both under a normal (or control) condition and under an intervention such as a treatment [92, 100, 167] or a genetic modification [106]. In other instances, one may want to contrast healthy versus diseased [117] cells or even more than two conditions [8]. In such settings, one might look for differential abundance, i.e., cell population shifts between conditions. Initial analytical approaches ignored the continuous nature of biological processes and binned cells into discrete clusters before looking at differences in composition between clusters. Borrowing from the field of mass cytometry [88], milo [34], and DAseq [182] rely on low-dimensional representations of the observations and define data-driven local neighborhoods in which they test for differences in compositions. Each of these methods show clear improvements in performance over cluster-based methods, and provide a more principled approach that better reflects the nature of the system.

However, many studies with multiple conditions, if not most, actually involve processes that can be described by a trajectory. Utilizing this underlying biology could increase either the interpretability of the results or the ability to detect true and meaningful changes between conditions. In this manuscript, we present the condiments workflow, a general framework to analyze dynamic processes under multiple conditions that leverages the concept of a trajectory structure. condiments has a more specific focus than milo or DAseq, but it compensates for this by improving the quality of the differential abundance assessment and its biological interpretation. Our proposed analysis workflow is divided into three steps. In Step 1, condiments considers the trajectory inference question, assessing whether the dynamic process is fundamentally different between conditions, which we call *differential topology*. In Step 2, it tests for differential abundance of the different conditions along lineages and between

lineages, which we respectively call *differential progression* and *differential differentiation*. Lastly, in Step 3, it estimates gene expression profiles similarly to Van den Berge et al. [158] and tests whether gene expression patterns differ between conditions along lineages, therefore extending the scope of `differential expression`.

In this manuscript, we first present the condiments workflow, by detailing the underlying statistical model, and providing an explanation and intuition for each step. We then benchmark condiments against more general methods that test for differential abundance to showcase how leveraging the existence of a trajectory improves the assessment of differential abundance. Finally, we demonstrate the flexibility and improved interpretability of the condiments workflow in three case studies that span a variety of biological settings and topologies.

4.2 Results

General model and workflow

Data structure and statistical model. We observe gene expression measures for J genes in n cells, resulting in a $J \times n$ count matrix $\mathbf{Y}$. For each cell i, we also know its condition label $c(i) \in \{1, \dots, C\}$ (e.g.,"treatment" or "control", "knock-out" or "wild-type"). We assume that, for each condition c, there is an underlying developmental structure $\mathcal{T}_c$, or trajectory, that possesses a set of L_c lineages.

For a given cell i with condition $c(i)$, its position along the developmental path $\mathcal{T}_{c(i)}$ is defined by a vector of $L_{c(i)}$ pseudotimes $\mathbf{T}_i$ and a unit-norm vector of $L_{c(i)}$ weights $\mathbf{W}_i$ ($||\mathbf{W}_i||_1 = 1$) (i.e., there is one pseudotime and one weight per lineage), with

$$\mathbf{T}_i \sim G_{c(i)} \text{ and } \mathbf{W}_i \sim H_{c(i)}. \tag{4.1}$$

The cumulative distribution functions (CDF) G_c and H_c are condition-specific and we make limited assumptions on their properties (see the method section for details). The pseudotime values represent how far a cell has progressed along each lineage, while the weights represent how likely it is that a cell belongs to each lineage. The gene expression model will be described below. Using this notation, we can properly define a trajectory inference (TI) method as a function that takes as input $\mathbf{Y}$ – and potentially other arguments – and returns estimates of L_c, $\mathbf{T}$, $\mathbf{W}$, and eventually $\mathcal{T}_c$.

Step 1 - Differential Topology: Should we fit a common trajectory? The first question to ask in our workflow is: Should we fit a common trajectory to all cells regardless of their condition? Or are the developmental trajectories too dissimilar between conditions? To demonstrate what this means, consider two extremes. For a dataset that consists of a mix of bone marrow stem cells and epithelial stem cells, using tissue as our condition, it is obvious that the developmental trajectories of the two conditions are not identical and should be estimated separately. On the other hand, if we consider a dataset where only a

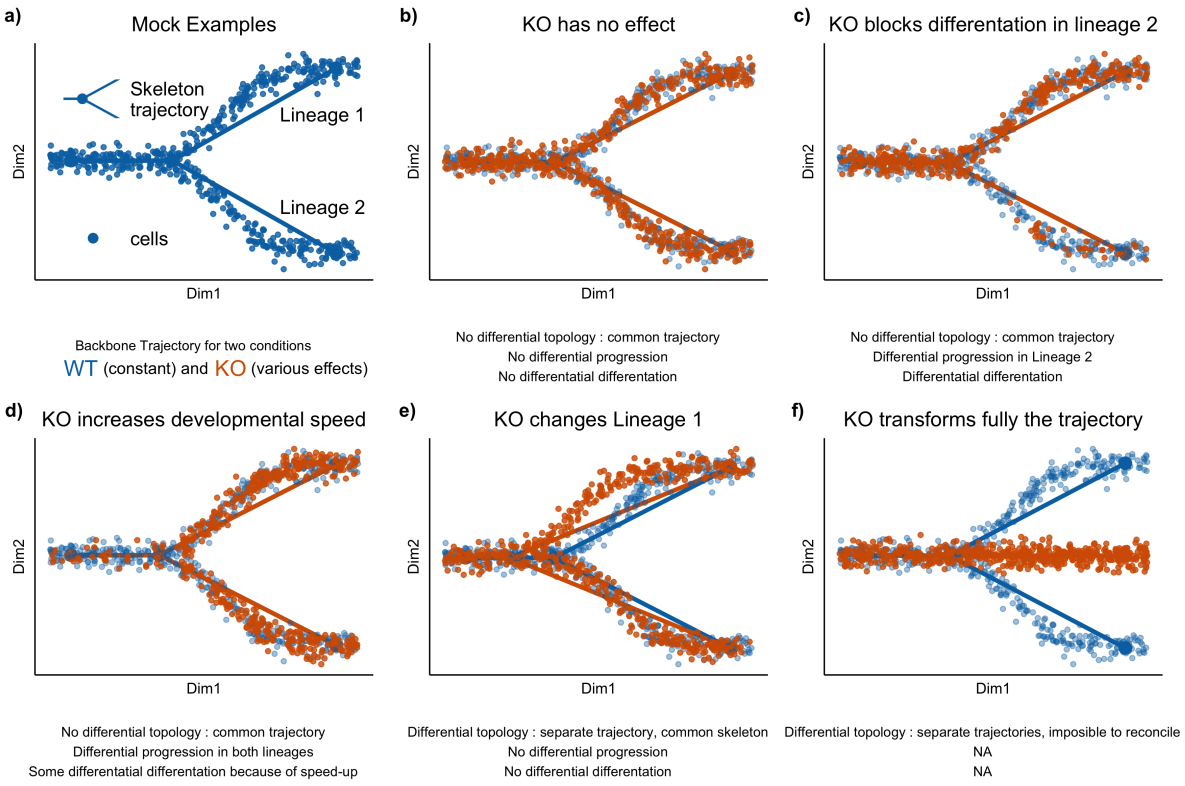


Figure 4.1: *Illustrating the first two steps of the* condiments *workflow with several scenarios* **(a.)** The examples are all built on a similar wild-type backbone, i.e., two lineages that slowly diverge in the absence of knocking out. Cells either originate from a wild-type (WT, blue) or a knock-out (KO, orange) condition. In **(b.)**, the knock-out has no effect, all three tests fail to reject their null hypothesis. In **(c.)**, the knock-out partly blocks differentiation along Lineage 2, meaning that fewer cells develop along that lineage. In this case, while the `topologyTest` fails to reject the null, we have both differential progression along Lineage 2 and differential differentiation. In **(d.)**, the knock-out speeds development, so there are more orange cells toward the end of both lineages. This leads to both differential progression and differentiation. In **(e.)**, the knock-out modifies the intermediate stage for Lineage 1 and changes where the lineages bifurcates; based on the `topologyTest`, we fit one trajectory per condition. However, the skeleton structure is unchanged, so there is a mapping between the two trajectories and we can still test for differential progression and differentiation. In both cases, we fail to reject the null. Finally, in **(f.)**, the knock-out fully disrupts the developmental process: all cells in the knock-out condition progress along a new lineage. Here, we fit separate trajectories and these cannot be reconciled easily, so we cannot proceed to Steps 2 and 3.

few genes are differentially expressed between conditions, the impact on the developmental process will be minimal and it is sensible to estimate a single common trajectory.

Indeed, we favor fitting a common trajectory for several reasons. Firstly, fitting a common trajectory is a more stable procedure since more cells are used to infer the trajectory. Secondly, our workflow still provides a way to test for differences between conditions along and between lineages even if a common trajectory is inferred. In particular, fitting a common trajectory between conditions does not require that cells of distinct conditions differentiate similarly along that trajectory. Finally, fitting different trajectories greatly complicates downstream analyses since we may need to map between distinct developmental structures before comparing them (i.e., each lineage in the first trajectory must match exactly one lineage in the second trajectory). Therefore, our workflow recommends fitting a common trajectory if the differences between conditions are *small enough*.

To quantify what *small enough* is, we rely on two approaches. The first is a qualitative diagnostic tool called *imbalance score*. It requires as input a reduced-dimensional representation $\mathbf{X}$ of the data $\mathbf{Y}$ and the condition labels. Each cell is assigned a score that measures the imbalance between the local and global distributions of condition labels. Similarly to Burkhardt et al. [22] and Dann et al. [34], the neighborhood of a cell is defined using a k-nearest neighbor graph on $\mathbf{X}$, which allows the method to scale very well to large values of n. Cell-level scores are then locally scaled using smoothers in the reduced-dimensional space (see the Methods section).

However, visual representation of the scores may not always be enough to decide whether or not to fit a common trajectory in less obvious cases. Therefore, we introduce a more principled approach, the `topologyTest`. This test assesses whether we can reject the following null hypothesis:

$$H_0 : \forall (c_1, c_2) \in \{1, \dots, C\}^2, \mathcal{T}_{c_1} = \mathcal{T}_{c_2}. \tag{4.2}$$

Under the null, the trajectory is common among all conditions and can therefore be estimated using all cells. Therefore, an estimation of the pseudotime vectors done by inferring a trajectory for each condition should be equivalent to the same procedure after permuting the condition labels. This is what is done for the `topologyTest`. A set of pseudotime vectors is estimated with the true condition labels. Another set is generated using permuted labels. Under the null, these two distributions should be equal. We can therefore test hypothesis (4.2) by testing for the equality in distributions of pseudotime using a variety of statistical tests (see the Methods section for details). Since we want to favor fitting a common trajectory and we only want to discover cases that are not only statistically significant but also biologically relevant, the tests typically include a minimum magnitude requirement for considering the difference between distributions to be significant (similar to a minimum log-fold-change for assessing DE). More details and practical implementation considerations are discussed in the Methods section.

In practice, the `topologyTest` requires maintaining a mapping between each of the trajectories, both between conditions and between permutations (see the Methods section where we define a mapping precisely). Trajectory inference remains a semi-supervised task, that

generally cannot be fully automated. In particular, the number of estimated lineages might change between different permutations for a given condition, precluding a mapping. As such, the `topologyTest` is only compatible with certain TI methods that allow for the specification of an underlying skeleton structure [141, 65], where the adjacency matrix can be pre-specified, as well (optionally) start and/or end states.

In the examples from Fig 4.1, the skeleton of the trajectory is represented by a series of nodes and edges. In examples 4.1b-d, the knock-out has no impact on this skeleton compared to the wild-type. In example 4.1e, the knock-out (KO) modifies the skeleton, in that the locations of the nodes change. However, the adjacency matrix does not change and the two skeletons represent isomorphic graphs: the skeleton structure is preserved.

For some TI methods [141, 65], it is possible to specify and preserve this skeleton structure. This means that the mapping of lineages can be done automatically. The `topologyTest` utilises this, and is thus restricted to such TI methods. This common skeleton structure can also be used if the null of the `topologyTest` is rejected. The availability of a mapping between lineages means that the next steps of the workflow can be conducted as if we had failed to reject the null hypothesis, as done in Fig 4.1e. The third case study will also present an example of this.

Even if the null is rejected by the `topologyTest` and separate trajectories must be fitted for each condition, a common skeleton structure can still be used to map between trajectories. This mapping means that the next steps of the workflow can be conducted as if we had failed to reject the null hypothesis, as done in Fig 4.1e. The third case study will also present an example of this. In cases where no common skeleton structure exists, such as Fig 4.1f, no automatic mapping exists. Differential abundance can be assessed but requires a manual mapping. Differential expression can still be conducted as well.

Step 2 - Differential abundance: What are the global differences between conditions? The second step of the workflow focuses on differences between conditions at the trajectory level. It requires either a common trajectory, or multiple trajectories and a mapping. We can then ask whether cells from different conditions behave similarly as they progress along the trajectory. To facilitate the interpretation of the results, we break this into two separate questions. Note that, at this step and the next, we are no longer limited to specific TI methods. Moreover, the mapping can be partial. In that case, Step 2 will be restricted to the parts (or subgraphs) of the trajectories that are mappable. See the Methods section for proper definitions of mapping and partial mapping.

Step 2a: Differential Progression. Although the topology might be common, cells might progress at different rates along the lineages for different conditions. For example, a treatment might limit the differentiation potential of the cells compared to the control, or instead speed it up. In the first case, one would expect to have more cells at the early stages and fewer at the terminal state, when comparing treatment and control. Using our

statistical framework, testing for differential progression amounts to testing:

$$H_0 : \forall (c_1, c_2) \in \{1, \dots, C\}^2, G_{c_1} = G_{c_2}. \tag{4.3}$$

This test can also be conducted at the individual-lineage level. If we denote by G_{lc} the l^{th} component of the distribution function G_c, we can test for differential progression along lineage l by considering the null hypothesis:

$$H_0 : \forall (c_1, c_2) \in \{1, \dots, C\}^2, G_{lc_1} = G_{lc_2}. \tag{4.4}$$

We can assess either or both null hypotheses in the `progressionTest`, which relies on non-parametric tests to compare two or more distributions, e.g., the Kolmogorov-Smirnov test [137] or the classifier test [85]. More details and practical implementation considerations are discussed in the Methods Section.

Step 2b: Differential Differentiation. Although the topology might be common, cells might also differentiate in varying proportions between the lineages for different conditions. For example, an intervention might lead to preferential differentiation along one lineage over another, compared to the control condition; or might alter survival rates of differentiated cells between two end states. In both cases, the weight distribution will be different between the control and treatment. Assessing differential differentiation at the global level amounts to testing, in our statistical framework, the null hypothesis

$$H_0 : \forall (c_1, c_2) \in \{1, \dots, C\}^2, H_{c_1} = H_{c_2}. \tag{4.5}$$

This test can also be conducted for a single pair of lineages (l, l'):

$$H_0 : \forall (c_1, c_2) \in \{1, \dots, C\}^2, [H_{lc_1}, H_{l'c_1}] = [H_{lc_2}, H_{l'c_2}]. \tag{4.6}$$

The above null hypotheses can again be tested by relying on non-parametric test statistics. We also discuss specific details and practical implementation in the Methods section.

The `progressionTest` and `differentiationTest` are quite linked since the functions G_c and H_c are correlated and will therefore often return similar results. However, they do answer somewhat different questions. In particular, looking at single-lineage (`progressionTest`) and lineage-pair (`differentiationTest`) test statistics will allow for a better understanding of the global differences between conditions. Differential differentiation does not necessarily imply differential progression and vice versa.

Step 3 - Differential Expression: Which genes have different expression patterns between conditions? Steps 1 and 2 focus on differences at a global level (i.e., aggregated over all genes) and will detect large changes between conditions. However, such major changes are ultimately driven by underlying differences in gene expression patterns. Furthermore, even in the absence of global differences, conditions might still have some more

subtle impact at the gene level. In the third step, we therefore compare gene expression patterns between conditions for each of the lineages. Step 3 is even more general than Step 2, in that it can be used without mapping between trajectories, i.e., some or all lineages could be condition-specific.

Following the `tradeSeq` manuscript by Van den Berge et al. [158], we consider a general and flexible model for gene expression, where the gene expression measure Y_{ji} for gene j in cell i is modeled with the negative binomial generalized additive model (NB-GAM) described in Equation (4.13). We extend the `tradeSeq` model by additionally estimating condition-specific average gene expression profiles for each gene. We therefore rely on lineage-specific, gene-specific, and condition-specific smoothers, s_{jlc}. With this notation, we can introduce the `conditionTest`, which, for a given gene j, tests the null hypothesis that these smoothers are identical across conditions:

$$H_0 : s_{jlc_1} = s_{jlc_2},\ \forall (c_1, c_2), \forall l. \tag{4.7}$$

As in tradeSeq, we rely on the Wald test to test H_0 in terms of the smoothers' regression coefficients. We can also use the fitted smoothers to visualize gene expression along lineages between conditions or cluster genes according to their expression patterns.

Simulations

We generate multiple trajectories using the simulation framework provided by Cannoodt et al. [25]. Within this framework, it is possible to knock out a specific gene. Here, we knock out a master regulator that drives differentiation into the second lineage. The strength of this knock-out can be controlled via a multiplier parameter m. If $m = 0$, the knock-out is total. If $0 < m < 1$, we have partial knock-out. If $m > 1$, the master regulator is over-expressed and cells differentiate much faster along the second lineage.

Three types of datasets are generated: Simple branching trajectories (two lineages, e.g., Fig. 4.2a) of $3,500$ cells, with equal parts wild-type and knock-out; trajectories with two consecutive branchings (and thus three lineages, e.g., Fig. 4.2b) of $3,500$ cells, with equal parts wild-type and knock-out; and branching trajectories (two lineages) of $5,000$ cells with three conditions, wild-type, knock-out with multiplier m, and induction with multiplier $1/m$ (Fig. 4.2c).

The simulation framework cannot, however, generate distinct trajectories for the different conditions, so we start the condiments workflow at Step 2, downstream of slingshot. We compare the `progressionTest` and `differentiationTest` from condiments to methods that also do not rely on clustering, but instead take into account the continuum of differentiation. milo [34] and DAseq[182] both define local neighborhoods using k-nearest neighbors graphs and look at differences of proportions in these neighborhood to test for differential abundance. These methods returns multiple tests per dataset (i.e., one per neighborhood), so we adjustfor multiple hypothesis testing using the Benjamini-Hochberg procedure [13]. By applying milo, DAseq, and condiments on the simulated datasets, we can compare the results of the tests

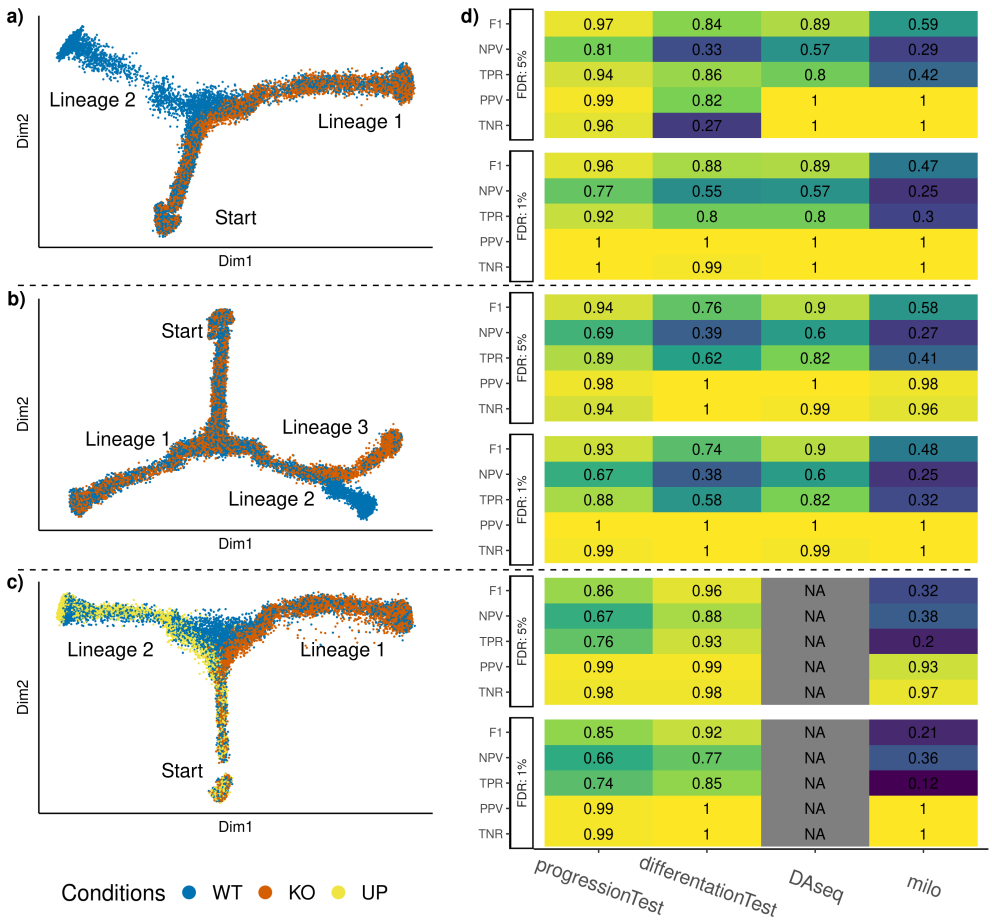


Figure 4.2: *Simulation results.* Three types of datasets are generated, with respectively two, three, and two lineages, and two, two, and three conditions. Reduced-dimensional representations of these datasets, for a multiplier value of $m = .5$, are presented in **(a.)**, **(b.)**, and **(c.)**, respectively. After generating multiple versions of the datasets for a range of values of m, we compare the performance of the `progressionTest` and `differentiationTest` with that of DAseq[182] and milo[34], when controlling the false discovery rate at nominal levels 1% and 5% using the Benjamini-Hochberg [13] procedure. In **(d.)**, each cell represents the performance measure associated with one test on one dataset for one nominal FDR level. Cells are also colored according to the performance. Overall, with two conditions, the `progressionTest` ranks first, followed by DAseq and the `differentiationTest`. With three conditions, the `differentiationTest` ranks first. DAseq is limited to two conditions. Exact simulation parameters and metrics are specified in the Methods section.

versus the values of m: We count a true positive when a test rejects the null and $m \neq 1$, and a true negative if the test fails to reject the null and $m = 1$.

We compare the methods' ability to detect correct differences between conditions using five metrics: The true negative rate (TNR), positive predictive value (PPV), true positive rate (TPR), negative predictive value (NPV), and F1-score, when controlling the FDR at two nominal levels of 1% and 5%. More details on the simulation scenarios and metrics can be found in the Methods section. Results are displayed in Fig. 4.2d.

On all simulations, all methods display excellent results for the TNR and PPV (except for the `differentiationTest` with level 1% on the branching dataset). However, the performances for the TPR (power), NPV, and F1-rate vary quite widely. On the two types of datasets with two conditions, the ranking is uniform over all metrics and levels: `progressionTest`, DAseq, `differentiationTest`, and milo. On the third simulation setting with three conditions, we cannot benchmark DAseq since its testing framework is restricted to two conditions. Here, also, the ranking is uniform but the `differentiationTest` outperforms the `progressionTest`. Looking more closely at the results, we can see (Fig D.2) that this mostly stems from increased power for the `differentiationTest` when m is close to 1.

Overall, the tests from the condiments workflow offer a flexible approach that can handle various scenarios and still outperform competitors.

Case studies

We consider three real datasets as case studies for the application of the condiments workflow. Table 4.1 gives an overview of these datasets and summary results. These case studies aim to demonstrate the versatility and usefulness of the condiments workflow, as well as showcase how to interpret and use the tests in practice.

TGFB dataset

McFaline-Figueroa et al. [92] studied the epithelial-to-mesenchymal transition (EMT), where cells migrate from the epithelium (inner part of the tissue culture dish) to the mesenchyme (outer part of the tissue culture dish) during development. The developmental process therefore is both temporal and spatial. As cells differentiate, gene expression changes. Moreover, the authors studied this system under two settings: a mock (control) condition and a condition under activation of transforming growth factor β (TGFB).

After pre-processing, normalization, and integration (see details in the supplementary methods), we have a dataset of $9,268$ cells, of which $5,207$ are mock and $4,241$ are TGFB-activated. The dataset is represented in reduced dimension using UMAP[11] (Fig. 4.3a). Adding the spatial label of the cells (Fig. 4.3b) shows that the reduced-dimensional representation of the gene expression data captures the differentiation process.

We can then run the condiments workflow. The imbalance score of each cell is computed and displayed in Fig. 4.3c. Although some regions do display strong imbalance, there is no specific pattern along the developmental path. This is confirmed when we run the

Table 4.1: *Summary of all case study datasets.* We report the name, number of cells n, number of conditions C, number of lineages L of each dataset, as well as the p-value resulting from testing for differential topology, progression and differentiation and the number of differentially expressed genes between conditions according to the `conditionTest`

Dataset	n	C	L	`topology`	`progression`	`differentiation`	`DE`
TGFB[92]	$9,268$	2	1	0.38	$\leq 2.2 \times 10^{-16}$	*NA*	$1,993$
TCDD[100]	$9,951$	2	1	0.07	$\leq 2.2 \times 10^{-16}$	*NA*	$2,144$
KRAS[175]	$10,177$	3	3	$\leq 2.2 \times 10^{-16}$	$\leq 2.2 \times 10^{-16}$	$\leq 2.2 \times 10^{-16}$	363

`topologyTest`. The nominal p-value of the associated test is 0.38. We clearly fail to reject the null hypothesis and we consequently fit a common trajectory to both conditions using slingshot with the spatial labels as clusters. This single-lineage trajectory is shown in Fig. 4.3d.

Next, we can ask whether the TGFB treatment impacts the differentiation speed. The developmental stage of each cell is estimated using its pseudotime. Plotting the per-condition kernel density estimates of pseudotimes in Fig. 4.3e reveals a strong treatment effect. The pseudotime distribution for the mock cells is trimodal, likely reflecting initial, intermediary, and terminal states. However, the first mode is not present in the TGFB condition, and the second is skewed towards higher pseudotime values. This is very consistent with the fact that the treatment is a growth factor that would increase differentiation, as shown in the original publication. Testing for equality of the two distributions with the `progressionTest` confirms the visual interpretation. The nominal p-value associated with the test is smaller than 2.2×10^{-16} and we reject the null that the distributions are identical. Since the trajectory is limited to one lineage, there is no possible differential differentiation between pairs of lineages.

Then, we proceed to identifying genes whose expression patterns differ between the mock and TGFB conditions. After gene filtering, we fit smoothers to $10,549$ genes, relying on the model described in Equation (4.13). We test whether the smoothers are significantly different between conditions using the `conditionTest`. Testing against a log-fold-change threshold of 2, we find $1,993$ genes that are dynamically differentially expressed between the two conditions when controlling the false discovery rate (FDR) at a nominal level of 5%. Fig. 4.4a and b show the two genes with the highest Wald test statistic. The first gene, *LAMC2*, was also found to be differentially expressed in the original publication and has been shown to regulate EMT [109]. The second gene, *TGFBI* or TGFB-induced gene, is not surprising, and was also labelled as differentially expressed in the original publication. In contrast, the gene that is deemed the least differentially expressed exhibits no differences between the smoothers (Fig. 4.4c.). Looking at all $1,993$ DE genes, we can cluster and display their expression patterns along the lineage for both conditions (Fig. 4.4) and identify several groups of genes that have different patterns between the two conditions.

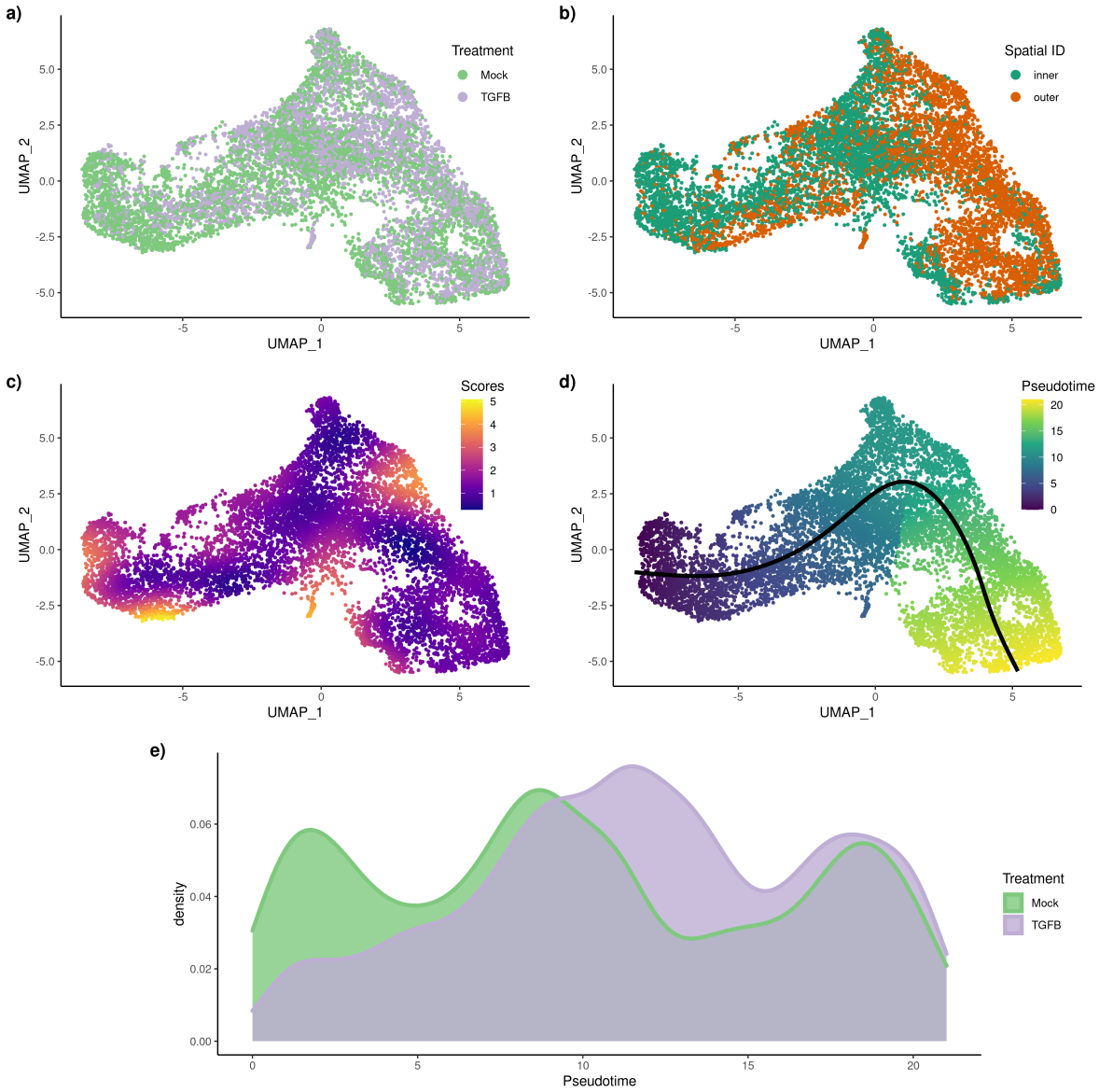


Figure 4.3: ***TGFB*** *dataset: Differential topology and differential progression.* After normalization and projection on a reduced-dimensional space (using UMAP), the cells can be colored either by treatment label **(a.)** or spatial origin **(b.)**. Using the treatment label and the reduced-dimensional coordinates, an imbalance score is computed and displayed **(c.)**. The `topologyTest` fails to reject the null hypothesis of no differential topology and a common trajectory is therefore fitted **(d.)**. However, there is differential progression between conditions: the pseudotime distributions along the trajectory are not identical **(e.)** between conditions and we reject the null using the `progressionTest`.

Finally, we perform a gene set enrichment analysis on all the genes that are differentially expressed between the conditions. The full results are available in Supplementary Table S1. Top annotations include gene sets involved in cell motility, adhesion, and morphogenesis, which are consistent with the expected biology.

TCDD dataset

Nault et al. [100] collected a dataset of 16,015 single nuclei to assess the hepatic effects of 2,3,7,8-tetrachlorodibenzo-p-dioxin or TCDD. In particular, they focused on the effect of TCDD on the 9,951 hepatocytes cells along the central-portal axis. This dataset is not a developmental dataset per se but still exhibits continuous changes along a spatial axis, demonstrating the versatility of the trajectory inference framework in general, and of the condiments workflow in particular.

Fig. D.3a shows a reduced-dimensional representation of the dataset, with cells labelled according to treatment/control condition, while Fig. D.3b shows the same plot colored by cell type, as derived by the authors of the original publication. The cells are aligned in a continuum, from central to mid-central and then mid-portal and portal. The imbalance score shows some spatial pattern (Fig. D.3c). However, the nominal p-value associated with the `topologyTest` is .07. We therefore fail to reject the null and we infer a common trajectory using slingshot on the spatial clusters. This results in a single-lineage trajectory that respects the ordering of the spatial clusters (Fig. D.3d). Note that, since the trajectory reflects a spatial continuum rather than a temporal one, the start of the trajectory is arbitrary. However, inverting the start and end clusters amounts to an affine transformation of the pseudotimes for all the cells. Step 2 and 3 are fully invariant to this transformation, so we can pick the Central cluster as the start of the trajectory.

The densities of the treatment and control pseudotime distributions differ greatly visually (Fig. D.3e), with the TCDD density heavily skewed toward the start of the trajectory. Indeed, the `progressionTest` has a nominal p-value $\leq 2.2 \times 10^{-16}$. This coincides with the finding of the original publication which highlighted the periportal hepatotoxicity of TCDD.

The ability of the `progressionTest` to correctly find large-scale changes in the spatial distribution of cells between conditions underscores why we favor fitting a common trajectory. Indeed, the p-value of the `topologyTest` in Step 1 is rather small and would have been below .05 if we had not conducted a test against a threshold. However, testing against a threshold and thus fitting a common trajectory does not stop the workflow from finding large-scale differences between conditions in Step 2 and results in a more stable estimate of the trajectory.

After gene filtering, we test 8,027 genes for spatial differential expression between conditions and we find 2,114 DE genes when controlling the FDR at a nominal level of 5%. The genes with the largest, second largest, and smallest test statistics are displayed in Fig.D.4a-c. Similarly to Nault et al. [100], we obtain a list of zonal genes from Halpern et al. [52]. The proportion of zonal genes among the DE genes is twice their proportion among non-DE genes.

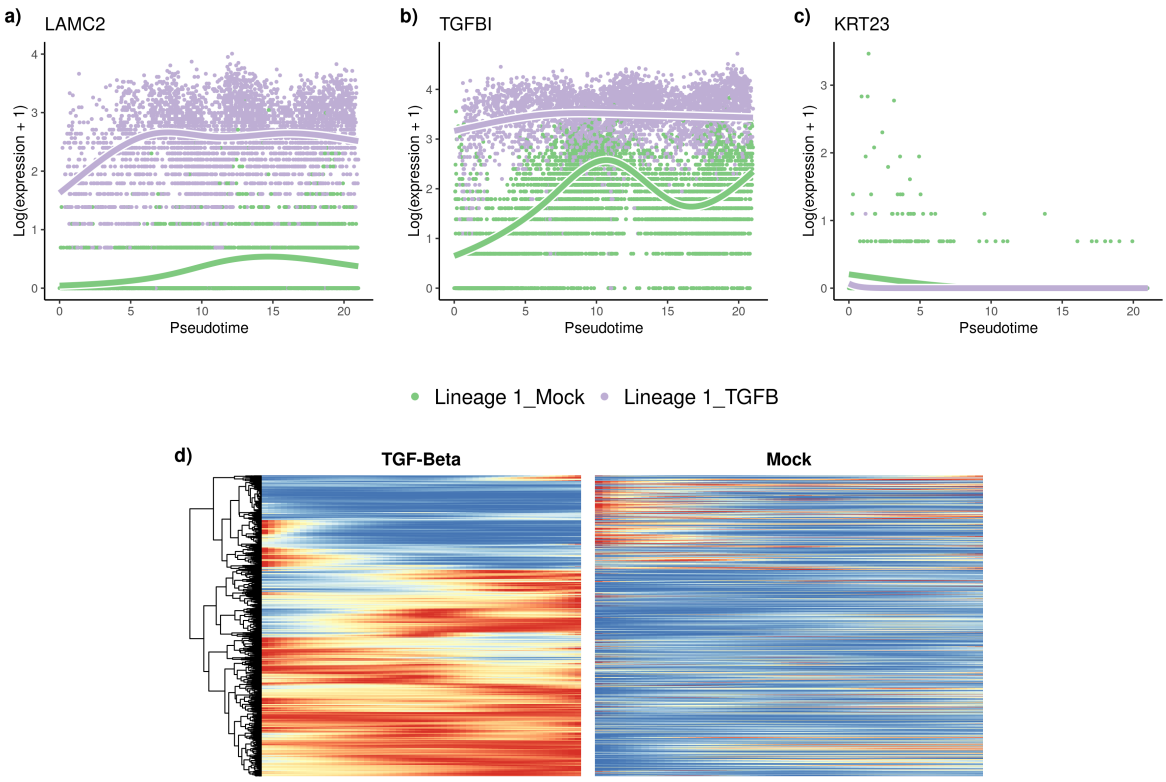


Figure 4.4: ***TGFB*** *dataset: Differential expression.* The tradeSeq gene expression model is fitted using the trajectory inferred by slingshot. Differential expression between conditions is assessed using the `conditionTest` and genes are ranked according to the test statistics. The genes with the highest **(a.)**, second highest **(b.)**, and smallest **(c.)** test statistics are displayed. After adjusting the p-values to control the FDR at a nominal level of 5%, we display genes for both conditions using a pseudocolor image **(d.)** after scaling each gene to a $[0, 1]$ range.

KRAS dataset

Xue et al. [175] studied the impact of KRAS(G12C) inhibitors at the single-cell level on three models of KRAS(G12C) lung cancers. Specifically, they examined how various cell populations react to these inhibitors and how some cells can return in proliferation mode shortly after the end of the treatment. Here, we want to investigate how the three cancer models (H358, H2122, and SW1573) differ in their response to the KRAS(G12C) inhibitors.

We use the reduced-dimensional representation from the original paper to display the $10,177$ cells from the various types (Fig 4.5a). Using the cancer type labels and the reduced-dimensional coordinates, an imbalance score can be computed (Fig 4.5b); some regions clearly show an imbalance. This is further confirmed by the `topologyTest`, with p-value smaller

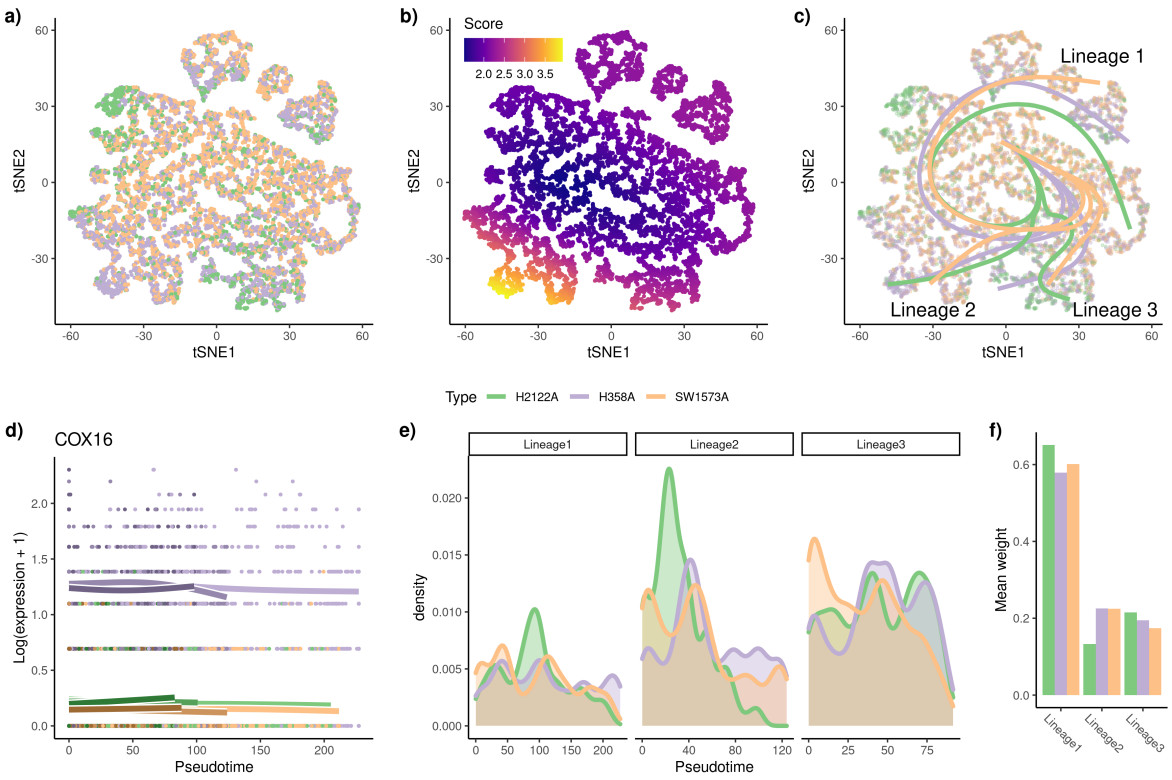


Figure 4.5: ***KRAS*** *dataset: Differential topology, differential progression, and differential differentiation.* Using the reduced-dimensional representation of the original publication (t-SNE), the cells can be colored by cancer type (a.). Using the cancer type label and the reduced-dimensional coordinates, an imbalance score is computed and displayed **(b.)**. The `topologyTest` rejects the null hypothesis of a common trajectory, we thus fit one trajectory per condition **(c.)**. However, the skeleton graphs have the same structure **(d.)**, so we can progress to the next steps in the `condiments` workflow. There is differential progression **(d.)** and we indeed reject the null of identical pseudotime distributions along the trajectory using the `progressionTest`. Similarly, there is differential differentiation **(e.)** and we reject the null of identical weight distributions along the trajectory using the `differentiationTest`. Here, we summarize the distributions by looking at the average weight for each lineage in each condition, which already shows some clear differences.

than 2.2×10^{-16}. We therefore do not fit a common trajectory to all cancer types (Fig 4.5c).

Note that this does not necessarily imply that the trajectory of reaction to the KRAS(G12C) inhibitors is different between cancer types. Indeed, this may also reflect strong batch effects between conditions, which the normalization scheme was unable to fully remove when integrating the three cancer types in one common reduced-dimensional representation. Thus, it

is not really possible to draw a biological conclusion at this first step. However, this does mean that a separate trajectory should be fitted to each condition.

Here, the trajectories, although different, are similar enough that we can still use an underlying common skeleton (Fig 4.5d). Indeed, we keep the tree structure derived by computing the minimum spanning tree (MST) on the clusters using all cells. This way, it is possible to derive a one-to-one mapping between the lineages of the three trajectories and we respect the assumptions detailed in Section 4.4 that are necessary for the `progressionTest` and `differentiationTest`.

Using this common mapping, we can then proceed to the `progressionTest`. At the global trajectory level, the nominal p-value is smaller than 2.2×10^{-16}, showing clear differential progression. At the lineage level, all three lineages show strong differential progression, with p-values of 2.2×10^{-16}, 1.2×10^{-12}, and 1.2×10^{-14}, respectively. The density plots for the pseudotime distributions at the single-lineage level (Fig 4.5e) indicate that the differential progression is driven by a group of cells from cancer type H2122A. This matches the top left part of the reduced-dimensional plot, the region where cells exit the initial inhibition stage to enter the reactivation stage. The second lineage also shows a difference between H2122A and the two other models. The pseudotime distribution is heavily skewed toward earlier points in that model compared to the other two. Lineage 2 represents differential progression to a drug-induced state. In Lineage 3, it is the SW1573A model that displays more differential progression.

The `differentiationTest` also has a p-value smaller than 2.2×10^{-16}. Although all pairwise comparisons are significant, the test statistics are much higher for the Lineage 2 vs. 1 and Lineage 2 vs. 3 comparisons. This again suggests that one model differentiates less into the drug-induced path, compared to the other two. Since the weights have to sum to 1, the 3-dimensional distribution can be fully summarized by any two components. Fig D.5 shows clear differences in distributions but visually interpreting different 2D distributions is still challenging. A simpler way to compare the distributions is to look at the average weight in each condition for each lineage (Fig 4.5f). This ignores the correlation between lineages but still allows for some interpretation. We can see in particular that Lineages 1 and 3 have greater weights for H2122A than for the other two conditions, which is consistent with the different pairwise statistics.

With the mapped trajectories, we can also perform gene-level analysis using the `conditionTest`. When comparing genes across all lineages and conditions, we find 363 differentially expressed genes when controlling the FDR at nominal level 5%. We show the genes with the highest, second highest, and smallest test statistics in Fig. D.6a-c. Displaying these global patterns across all three lineages and all three conditions makes it hard to interpret. We therefore focus on the first (and longest) lineage. In that lineage, we find 366 DE genes and we show their expression patterns along Lineage 1 in all three cancer models in Fig. D.6d.

4.3 Discussion

In this manuscript, we have introduced condiments, a full workflow to analyze dynamic systems under multiple conditions. By separating the analysis into several steps, condiments offers a flexible framework with increased interpretability. Indeed, we follow a natural progression through a top-down approach, by first studying overall differences in trajectories with the `topologyTest`, then differences in abundances at the trajectory level with the `progressionTest` and `differentiationTest`, and finally gene-level differences in expression with the `conditionTest`.

As demonstrated in the simulation studies, taking into account the dynamic nature of systems via the trajectory representation enables condiments to better detect true changes between conditions. The flexibility offered by our implementation, which provides multiple tests for non-parametric comparisons of distributions, also allows us to investigate a wide array of scenarios. This is evident in the three case studies presented in the manuscript. Indeed, in the first case study we have a developmental system under treatment and control conditions, while in the second case study the continuum does not represent a developmental process but spatial separation. In the third case study, the conditions do not reflect different treatments but instead different cancer models. This shows that condiments can be used to analyze a wide range of datasets.

Often, the different conditions also represent different batches. Indeed, some interventions cannot be delivered on a cell-by-cell basis and this creates unavoidable confounding between batches and conditions. Normalization and integration of the datasets must therefore be done without eliminating the underlying biological signal. This balance can be hard to strike, as discussed in Zhao et al. [182]. Proper experimental design – such as having several batches per condition – or limiting batch effects as much as possible – for example, sequencing a mix of conditions together – can help lessen this impact. Still, some amount of confounding is sometimes inherent to the nature of the problem under study.

The tests used in the workflow (e.g., Kolmogorov-Smirnov test) assume that the pseudotime and weight vector are known and independent observations for each cell. However, this is not the case: they are estimated using TI methods which use all samples to infer the trajectory, and each estimate inherently has some uncertainty. Here, we ignore this dependence, as is the case in other differential abundance methods, which assume that the reduced-dimensional coordinates are observed independent random variables even thought they are being estimated using the full dataset. We stress that, rather than attaching strong probabilistic interpretations to p-values (which, as in most RNA-seq applications, would involve a variety of hard-to-verify assumptions and would not necessarily add much value to the analysis), we view the p-values produced by the condiments workflow as useful numerical summaries for guiding the decision to fit a common trajectory or condition-specific trajectories and for exploring trajectories across conditions and identifying genes for further inspection.

Splitting the data into two groups, where the first is used to estimate the trajectory and the second is used for pseudotime and weight estimation could, in theory, alleviate the

dependence issue, at the cost of smaller sample sizes. However, this would ignore the fact that, in practice, users perform exploratory steps using the full data before performing the final integration, dimensionality reduction, and trajectory inference. Moreover, results on simulations show that all methods considered keep excellent control of the false discovery rate despite the violation of the independence assumptions. This issue of "double-dipping" therefore seems to have a limited impact.

The two issues raised in the previous paragraphs highlight the need for independent benchmarking. Simulation frameworks such as dyngen [25] are crucial. They also need to be complemented by real-world case studies, which will become easier as more and more datasets that study dynamic systems under multiple conditions are being published. condiments has thus been developed to be a general and flexible workflow that will be of use to researchers asking complex and ever-changing questions.

4.4 Methods

Tests for equality of distributions

General setting

Consider a set of n i.i.d. observations, $\mathbf{X}$, with $\mathbf{X}_i \sim \mathbf{P}_1$, and a second set of m i.i.d. observations, $\mathbf{Y}$, with $\mathbf{Y}_j \sim \mathbf{P}_2$, independent from $\mathbf{X}$. For example, in our setting, $\mathbf{X}$ and $\mathbf{Y}$ may represent estimated pseudotimes for cells from two different conditions. We limit ourselves to the case where $\mathbf{X}$ and $\mathbf{Y}$ are random vectors of the same dimension d.

The general goal is to test the null hypothesis that $\mathbf{X}$ and $\mathbf{Y}$ have the same distribution, i.e., $H_0 : \mathbf{P}_1 = \mathbf{P}_2$.

Univariate case: The weighted Kolmogorov-Smirnov test

The two-sample Kolmogorov-Smirnov test. Consider the case where $\mathbf{X}_i$ and $\mathbf{Y}_j$ are scalar random variables (i.e., $d = 1$). The associated empirical cumulative distribution functions (ECDFs) are denoted, respectively, by $\mathbf{F}_{1,n}$ and $\mathbf{F}_{2,m}$. The univariate case occurs, for example, when there is only one lineage in the trajectory(ies), so that the pseudotime estimates are scalars.

In this setting, one can test H_0 using the standard Kolmogorov-Smirnov test [137], with test statistic defined as:

$$\begin{aligned}\mathbf{D}_{n,m} &\equiv \sup_x \left|\mathbf{F}_{1,n}(x) - \mathbf{F}_{2,m}(x)\right| \\ &= \sup_{x \in \mathbf{X} \cup \mathbf{Y}} \left|\mathbf{F}_{1,n}(x) - \mathbf{F}_{2,m}(x)\right|.\end{aligned}$$

The rejection region at nominal level α is

$$\left[\sqrt{-\frac{1}{2} \times \log \frac{\alpha}{2} \times \frac{n+m}{n \times m}}, \infty\right).$$

That is, we reject the null hypothesis at the α-level if and only if $\mathbf{D}_{n,m} \geq \sqrt{-1/2 \times \log \alpha/2 \times \frac{n+m}{n \times m}}$.

The two-sample weighted Kolmogorov-Smirnov test. Consider a more general setting where we have weights $w_{1,i} \in [0,1]$ and $w_{2,j} \in [0,1]$ for each of the observations. In trajectory inference, the weights may denote the probability that a cell belongs to a particular lineage in the trajectory. Following Monahan [97], we modify the Kolmogorov-Smirnov test in two ways. Firstly, the empirical cumulative distribution functions are modified to account for the weights

$$\begin{aligned}\mathbf{F}_{1,n}(x) =& \frac{1}{\sum_{i=1}^{n} w_{1,i}} \sum_{i=1}^{n} w_{1,i} \times \mathcal{I}_{(-\infty,x]}(\mathbf{X}_i) \\ \mathbf{F}_{2,m}(x) =& \frac{1}{\sum_{j=1}^{m} w_{2,j}} \sum_{j=1}^{m} w_{2,j} \times \mathcal{I}_{(-\infty,x]}(\mathbf{Y}_j).\end{aligned}$$

Secondly, the definition of $\mathbf{D}_{n,m}$ is unchanged, but the significance threshold is updated, that is, the rejection region is

$$\left[\sqrt{-\frac{1}{2} \times \log \frac{\alpha}{2} \times \frac{n'+m'}{n' \times m'}}, \infty\right),$$

where

$$n' = \frac{\left(\sum_{i=1}^{n} w_{1,i}\right)^2}{\sum_{i=1}^{n} w_{1,i}^2} \text{ and } m' = \frac{\left(\sum_{j=1}^{m} w_{2,j}\right)^2}{\sum_{j=1}^{m} w_{2,j}^2}.$$

Multivariate case: The classifier test

Concept. Suppose that we have a classifier $\delta(\cdot)$, which could be, for example, a multinomial regression or SVM classifier. This classifier is a function from the support of $\mathbf{X}$ and $\mathbf{Y}$ into $\{1,2\}$. The data are first split into a learning and a test set, such that the test set contains n_{test} observations, equally-drawn from each population, i.e., there are $n_{\text{test}}/2$ observations $\mathbf{X}^{(\text{test})}$ from $\mathbf{X}$ and $n_{\text{test}}/2$ observations $\mathbf{Y}^{(\text{test})}$ from $\mathbf{Y}$. Next, the classifier is trained on the learning set. We denote by $Acc \equiv |\{i : \delta(\mathbf{X}_i^{(\text{test})}) = 1\} \bigcup \{j : \delta(\mathbf{Y}_j^{(\text{test})}) = 2\}|$ the number of correct assignations made by the classifier on the test set.

If $n = m$, under the null hypothesis of identical distributions, no classifier will be able to perform better on the test set than a random assignment would, i.e., where the predicted label is a $Bernoulli(1/2)$ random variable. Therefore, testing the equality of the distributions of $\mathbf{X}$ and $\mathbf{Y}$ can be formulated as testing

$$H_0 : \mathbb{E}[Acc] = \frac{n_{\text{test}}}{2} \text{ vs. } H_1 : \mathbb{E}[Acc] > \frac{n_{\text{test}}}{2}.$$

Under the null hypothesis, the distribution of Acc is:

$$Acc \sim_{H_0} Binom(n_{\text{test}}, 1/2).$$

As detailed in Lopez-Paz and Oquab [85], one can use the classifier to devise a test that will guarantee the control of the Type 1 error rate.

The classifier test in practice. In practice, we make no assumptions about the way in which the distributions we want to compare might differ, which means the classifier needs to be quite flexible. Following Lopez-Paz and Oquab [85], we chose to use either a k-nearest neighbor classifier (k-NN) or a random forests classifier [20], since such classifiers are fast and flexible. Hyper-paramters are chosen through cross-validation on the learning set. To avoid issues with class imbalance, we downsample the distribution with the largest number of samples first so that each distribution has the same number of observations. That is, we have $n' = \min(m, n)$ observations in each condition (or class). A fraction (by default 30%, user-defined) is kept as test data, so that $n_{\text{test}} = .3 \times n'$. We then train the classifier on the learning data, and select the tuning parameters through cross-validation on that learning set. Finally, we predict the labels on the test set and compute the accuracy of the classifier on that test set. This yields our classifier test statistic.

Power of the classifier test. It is interesting to note that the classifier test is valid no matter the classifier chosen. However, the choice of classifier will have obvious impact on the power of the test.

Multivariate case: Other methods

Although we have found that the classifier test performs best in practice, there are many methods that test for the equality of two multivariate distributions. We have implemented a few such methods in condiments, in case users would like to try them: The two-sample kernel test [48] and the permutation test relying on the Wasserstein distance. These methods were found to be less efficient in initial benchmarking, but are implemented in case users want to use them.

Multivariate case: The two-sample kernel test

Mean maximum discrepancy. The two-sample kernel test was defined by Gretton et al. [48] and relies on the mean maximum discrepancy (MMD). Considering a kernel function

$$\begin{aligned} k : \mathbb{R}^d \times \mathbb{R}^d &\to \mathbb{R} \\ (x, y) &\mapsto k(x, y) \end{aligned}$$

the MMD is then defined as

$$MMD^2(\mathbf{P}_1, \mathbf{P}_2, k) \equiv \mathbb{E}_{\mathbf{P}_1, \mathbf{P}_1}[k(X, X')] + \mathbb{E}_{y\mathbf{P}_2, \mathbf{P}_2}[k(Y, Y')] - 2\mathbb{E}_{\mathbf{P}_1, \mathbf{P}_2}[k(X, Y)].$$

For a properly defined kernel, we have $MMD^2(\mathbf{P}_1, \mathbf{P}_2, k) = 0$ i.i.f. $\mathbf{P}_1 = \mathbf{P}_2$.

Unbiased statistic. Following Gretton et al. [48], we define the unbiased MMD statistic:

$$MMD_u^2(\mathbf{X}, \mathbf{Y}) \equiv \frac{1}{n(n-1)} \sum_{i=1}^{n} \sum_{j \neq i}^{n} k(\mathbf{X}_i, \mathbf{X}_j) + \frac{1}{m(m-1)} \sum_{i=1}^{m} \sum_{j \neq i}^{m} k(\mathbf{Y}_i, \mathbf{Y}_j) - \frac{2}{mn} \sum_{i=1}^{n} \sum_{j=1}^{m} k(\mathbf{X}_i, \mathbf{Y}_j).$$

Linear statistic for faster computations. While the MMD^2 offers fast convergence, it can be burdensome to compute when m and n get large. Gretton et al. [48] propose a linear statistic in the case $m = n$. We can extend this in the general setting by just sampling a fixed fraction of the terms of each sum. This lowers kernel computation costs drastically.

Null distribution of the statistic. For some kernels, the MMD_u^2 follows some theoretical inequalities under the null that allows one to define rejection regions. However, this is not always the case. Therefore, in practice, we instead rely on permutations to compute a null distribution for the test statistic. Under the null, $\mathbf{X}_i$ and $\mathbf{Y}_j$ are from the same distribution so they can be swapped in the sums. We can therefore generate an empirical distribution and use it to define rejection regions.

Multivariate case: Optimal transport

We consider the Wasserstein distance [166, 37], also known as earth's mover distance, between the two distributions, estimated using the samples $\mathbf{X}$ and $\mathbf{Y}$. We can generate a null distribution for this metric by permuting observations in the combined $\mathbf{X}$ and $\mathbf{Y}$ datasets, thereby obtaining a valid test for $H_0 : \mathbf{P}_1 = \mathbf{P}_2$. This works in any number of dimensions, but is limited to the two-sample case.

Extending the setting by considering more than two conditions

Consider $C \geq 2$ sets of samples, such that, for $c \in \{1, \ldots, C\}$, we have n_c i.i.d. observations $\mathbf{X}^{(c)}$ with $\mathbf{X}_i^{(c)} \sim \mathbf{P}_c$. We want to test the null hypothesis:

$$H_0 : \mathbf{P}_{c_1} = \mathbf{P}_{c_2}, \ \forall c_1, c_2 \in \{1, \ldots, C\} \text{ and } c_1 \neq c_2.$$

While extensions of the Kolmogorov-Smirnov test [68] and the two-sample kernel test [7] have been proposed, we choose to focus only on the framework that is most easily extended to C conditions, namely, the classifier test. Indeed, the C-condition classifier test requires choosing a multiple-class classifier instead of a binary classifier (which is the case for the k-NN classifier and random forests), selecting n_{test}/C observations for each class in the test set, and testing:

$$H_0 : \mathbf{E}[Acc] = \frac{n_{\text{test}}}{C} \text{ vs. } H_1 : \mathbf{E}[Acc] > \frac{n_{\text{test}}}{C}.$$

Under the null distributions, the distribution of Acc is:

$$Acc \sim_{H_0} Binom(n_{\text{test}}, 1/C).$$

Extending the setting by considering an effect size

Effect size for the Kolmogorov-Smirnov test. The null hypothesis of the (weighted) Kolmogorov-Smirnov test is $H_0 : P_1 = P_2$. We can modify this null hypothesis by considering an effect size threshold t, such that $H_0(t) : \sup_x \left|P_1(x) - P_2(x)\right| \leq t$. The test statistic is then modified as:

$$\mathbf{D'}_{n,m} \equiv \max(\mathbf{D}_{n,m} - t, 0)$$

and the remainder of the testing procedure is left unchanged.

Effect size for the classifier test. Similarly, the null and alternative hypotheses of the classifier test can be modified to test against an effect size threshold t as follows

$$H_0 : Acc \leq \frac{n_{\text{test}}}{C} + t \text{ vs. } H_1 : Acc > \frac{n_{\text{test}}}{C} + t.$$

General statistical model for the trajectories

Consider a set of condition labels $c \in \{1, \ldots, C\}$ (e.g.,"treatment" or "control", "knock-out" or "wild-type"). For each condition, there is a given topology/trajectory $\mathcal{T}_c$ that underlies the developmental process. This topology is generally in the form of a tree, with a starting state which then differentiates along one or more lineages; but one can also have a circular graph, e.g., for the cell cycle. In general, a trajectory is defined as a directed graph.

We denote by L_c the number of unique paths – or lineages – in the trajectory $\mathcal{T}_c$ and by $\mathcal{C}_c$ the set of cells that belong to condition c. For example, for a tree structure, paths go from the root node (stem cell type) to the leaf nodes (differentiated cell type). For a cell cycle, any node can be be used as the start. A cell i from condition c_i is characterized by the following features:

$$\begin{aligned} \mathbf{T}_i &\sim G_{c_i} && : \text{ A vector of pseudotimes, one per lineage of } \mathcal{T}_{c_i} \\ \mathbf{W}_i &\sim H_{c_i} && : \text{ A vector of weights, one per lineage of } \mathcal{T}_{c_i}, \text{ s.t. } ||\mathbf{W}_i||_1 = 1. \end{aligned}$$

Note that the distribution functions are condition-specific. We further make the following assumptions:

- All G_c and H_c distributions are continuous;
- The support of all G_c is bounded in $\mathbb{R}^{L_c}$;
- The support of all H_c is $[0, 1]^{L_c}$.

The gene expression model will be discussed below, in the differential expression section.

Trajectory inference. Many algorithms have been developed to estimate lineages from single-cell data [126]. Most algorithms provide a binary indicator of lineage assignment, that is, the $\mathbf{W}_i$ vectors are composed of 0s and 1s, so that a cell either belongs to a lineage or it does not (note that when cells fall along a lineage prior to a branching event, this vector may include multiple 1s, violating our constraint that the $\mathbf{W}_i$ have unit norm. In such cases, we normalize the weights to sum to 1).

Mapping between trajectories. Many of the tests that we introduce below assume that the cells from different conditions follow "similar" trajectories. In practice, this means that we either have a common trajectory for all conditions or that there is a possible manual mapping from one lineage to another. The term "mapping" is more rigorously defined as follows.

Definition 1 ***The trajectories*** $\{\mathcal{T}_c : c \in \{1, \dots, C\}\}$ ***have a mapping if and only if*** $\forall (c_1, c_2) \in \{1, \dots, C\}^2$***,*** $\mathcal{T}_{c_1}$ ***and*** $\mathcal{T}_{c_2}$ ***are isomorphic.***

If there is a mapping, this implies in particular that the number of lineages L_c per trajectory $\mathcal{T}_c$ is the same across all conditions c and we call this this value L. Since a graph is always isomorphic with itself, a common trajectory is a special case of a situation where there is a mapping.

Definition 2 ***The trajectories*** $\{\mathcal{T}_c : c \in \{1, \dots, C\}\}$ ***have a partial mapping if and only if*** $\forall (c_1, c_2) \in \{1, \dots, C\}^2$***, there is a subgraph*** $\mathcal{T}'_{c_1} \subset \mathcal{T}_{c_1}$ ***and a subgraph*** $\mathcal{T}'_{c_2} \subset \mathcal{T}_{c_2}$ ***that are isomorphic.***

Essentially, this means that the size of the changes induced by the various conditions do not disturb the topology of the original trajectory *too much*. The above mathematical definitions aim to formalize what *too much* is. Indeed, if the conditions lead to very drastic changes, it will be quite obvious that the trajectories are different and comparing them will mostly be either non-informative or will not require a complex framework. We aim to build a test that retains reasonable power in more subtle cases.

Differential topology

Imbalance score. Consider a set of n cells, with associated condition labels $c_i \in \{1, \dots, C\}$ and coordinate vectors $\mathbf{X}_i$ in d dimensions, usually corresponding to a reduced-dimensional representation of the expression data obtained via PCA or UMAP[11, 93].

Let $\mathbf{p} = \{p_c\}_{c \in \{1,\dots,C\}}$ denote the "global" distribution of cell conditions, where p_c is the overall proportion of cells with label c in the sample of size n. The imbalance score of a cell reflects the deviation of the "local" distribution of conditions in a neighborhood of the cell compared to the global distribution $\mathbf{p}$. Specifically, for each cell i, we compute its k-nearest neighbor graph using the Euclidean distance in the reduced-dimensional space. We

therefore have a set of k neighbors and a set of associated neighbor condition labels $c_{i,\kappa}$ for $\kappa \in \{1, \dots, k\}$. We then assign to the cell a z-score, based on the multinomial test statistic $P(\{c_{i,\kappa}\}_{\kappa \in \{1,\dots,k\}}, \mathbf{p})$, as defined in Section 4.4. Finally, we smooth the z-scores in the reduced-dimensional space by fitting s cubic splines for each dimension. The fitted values for each of the cells are the imbalance scores. Thus, the imbalance scores rely on two user-defined parameters, k and s. We set default values of 10 for both parameters. However, since this is meant to be an exploratory tool, we encourage users try different values for these parameters and observe the changes to better understand their data.

General setting for the `topologyTest`. The imbalance score only provides a qualitative visual inspection of local imbalances in the distribution of cell conditions. However, we need a more global and formal way to test for differences in topology between condition-specific trajectories. That is, we wish to test the null hypothesis

$$H_0 : \mathcal{T}_{c_1} = \mathcal{T}_{c_2}, \ \forall (c_1, c_2) \in \{1, \dots, C\}^2. \tag{4.8}$$

In practice, in order to test H_0, we have a set of cells i with condition labels c_i. We can estimates the pseudotimes of each cell when fitting a trajectory for each condition. We then want to compare this distribution of pseudotimes to a null distribution. To generate this null distribution, we use permutations in the following manner

a) Estimate $\mathbf{T}_i$ for all i by inferring one trajectory per condition, using any trajectory inference method.

b) Randomly permute the condition labels c_i to obtain new labels c'_i, re-estimate $\mathbf{T'}_i$ for each i.

c) Repeat the permutation r times (by default, $r = 100$).

Under the null hypothesis, the n $\mathbf{T}_i$ should therefore be drawn from the same distribution as the $r \times n$ $\mathbf{T'}_i$. We can test this using the weighted Kolmogorov-Smirnov test (if $L = 1$), the kernel two-sample test (if $C = 2$), or the classifier test (any C). This is the `topologyTest`.

The aforementioned tests require that the samples be independent between the two distributions under comparison. However, here, the two distributions correspond to different pseudotime estimates for the same cells so the samples are not independent between distributions. Even within distributions, the independence assumption is violated: the pseudotimes are estimated using trajectory inference methods that rely on all samples. Moreover, within the $\mathbf{T'}_i$, we have r pseudotime estimates of each cell.

The first two violations of the assumptions are hard to avoid and are further addressed in the discussion section. However, we can eliminate the third one by simply taking the average $\mathbf{T'}_i$ for each cell. We then compare two distributions each with n samples. Both options (with and without averaging) are implemented in the condiments R package, but the default is the average.

Furthermore, rather than attaching strong probabilistic interpretations to p-values (which, as in most RNA-seq applications, would involve a variety of hard-to-verify assumptions and would not necessarily add much value to the analysis), we view the p-values produced by the condiments workflow simply as useful numerical summaries for exploring trajectories across conditions and identifying genes for further inspection.

Running the `topologyTest` in practice. Under the null, there should exist a mapping between trajectories, both within conditions and between permutations. However, in practice, most trajectory inference methods will be too unstable to allow for automatic mapping between the runs. Indeed, they might find a different number of lineages for some runs. Moreover, even if the number of lineages and graph structure remained the same across all permutations, mapping between permutations would break even more the independence assumption since the condition labels would need to be used.

Therefore, for now, the `topologyTest` test is limited to two trajectory inference methods, slingshot [141] and TSCAN [65], where a set graph structure can be prespecified. Both methods rely on constructing a minimum spanning tree (MST) on the centroids of input clusters in a reduced-dimensional space to model branching lineages. In TSCAN, a cell's pseudotime along a lineage is determined by its projection onto a particular branch of the tree, and its weight of assignment is determined by its distance from the branch. slingshot additionally fits simultaneous principal curves. A cell's pseudotime along a lineage is determined by its projection onto a particular curve and its weight of assignment is determined by its distance from the curve. We therefore construct the MST on the full dataset (i.e., using all the cells regardless of their condition label), based on user-defined cluster labels. Then, we keep the same graph structure as input to either TI method: the nodes are the centers of the clusters, but restrained to cells of a given condition. This way, the path and graph structure are preserved. Note however, that there no guarantee that the graph remains the MST when it is used for TI on a subset of cells.

Testing for differential progression

The differential progression test requires that a (partial) mapping exists between trajectories. If the mapping is only partial, we restrict ourselves to the mappable parts of the trajectories (i.e., subgraphs).

Testing for differential progression for a single lineage. For a given lineage l, we want to test the null hypothesis that the pseudotimes along the lineage are identically distributed between conditions, which we call *identical progression*. Following the above notation, we want to test that the l^{th} components G_{lc} of the distribution functions G_c are identical across conditions

$$H_0 : G_{lc_1} = G_{lc_2},\ \forall (c_1, c_2). \tag{4.9}$$

Testing for global differential progression. We can also test for global differences across all lineages, that is,

$$H_0 : G_{c_1} = G_{c_2}, \ \forall (c_1, c_2). \tag{4.10}$$

Possible tests. If $C = 2$, all tests introduced in Section 4.4 can be used to test the hypothesis in Equation (4.9). If $C > 2$, we need to rely on the classifier test.

If $L = 1$, the hypotheses in Equations (4.9) and (4.10) are identical. However, for $L > 1$, the functions G_c are not univariate distributions.

Using the Kolmogorov-Smirnov test in the $L > 1$ setting. For $L > 1$, we can use lineage-level weights as observational weights for each individual lineage, which is an appealing property. Two settings are possible.

- Test the null hypothesis in Equation (4.9) for each lineage using the Kolmogorov-Smirnov test and perform a global test using the classifier test or the kernel two-sample test.
- Test the null hypothesis in Equation (4.9) for each lineages using the Kolmogorov-Smirnov test and combine the p-values p_l for each lineage l using Stouffer's Z-score method [140], where each lineage is associated with observational weights $W_l = \sum_{i=1}^{n} \mathbf{W}_i[l]$. The nominal p-value associated with the global test is then

$$p_{glob} \equiv \frac{\sum_{l=1}^{L} W_l p_l}{\sqrt{\sum_{i=1}^{L} W_l^2}}.$$

Note that the second setting violates the assumption of Stouffer's Z-score method, since the p-values are not i.i.d. However, this violation does not seem to matter in practice and this test outperforms others so we set it as default.

Testing for differential differentiation

The differential progression test requires that a (partial) mapping exists between trajectories. If the mapping is only partial, we restrict ourselves to the mappable parts of the trajectories.

Testing for differential differentiation for a single pair of lineages. For a given pair of lineages l, l', we want to test the null hypothesis that the cells differentiate between l and l' in the same way between all conditions, which we call *identical differentiation*. Following the above notation, we want to test that the l^{th} and l'^{th} components of the distibution function H_c are the same

$$H_0 : \forall (c_1, c_2), [H_{lc_1}, H_{l'c_1}] = [H_{lc_2}, H_{l'c_2}]. \tag{4.11}$$

Testing for global differential differentiation. We can also test for a global difference across all pairs of lineages, that is,

$$H_0 : \forall (c_1, c_2), H_{c_1} = H_{c_2} \tag{4.12}$$

Possible tests. Since all variables are multivariate, we cannot use the Kolmogorov-Smirnov test. By default, this test relies on the classifier test with random forest as a classifier.

Testing for differential expression

Notation. The gene expression model does not require a mapping or even a partial mapping. Indeed, it can work as well with a common trajectory, different trajectories, or even a mix where some lineages can be mapped between the trajectories for various conditions and others cannot. To reflect this, we consider all L_{tot} lineages together. We introduce a new weight for each cell

$$\mathbf{Z}_i = \{Z_{ilc}\}_{l\in\{1,\dots,L_{tot}\},c\in\{1,\dots,C\}} \text{ s.t. } \begin{cases} Z_{ilc} = 0, & \text{if } i \neq \mathcal{C}_c \text{ or } l \notin \mathcal{T}_{c_i} \\ \{Z_{ilc_i}\}_{l\in\{1,\dots,C\}} \sim \mathcal{M}(\mathbf{W_i}), & \text{otherwise} \end{cases},$$

where $\mathcal{M}(\mathbf{W_i})$ is a binary (or one-hot) encoding representation of a multinomial distribution with proportions $\mathbf{W}_i$ as in tradeSeq.

Likewise, we modify the pseudotime vector to have length L_{tot} such that

$$T_{li} = \begin{cases} 0, \text{ if } l \notin \mathcal{T}_{c_i} \\ \mathbf{T}_i[l], \text{ otherwise} \end{cases}.$$

Gene expression model. We adapt the model from Van den Berge et al. [158] to allow for condition-specific expression. For a given gene j, the expression measure Y_{ji} for that gene in cell i can be modeled thus:

$$\begin{cases} Y_{ji} & \sim & NB(\mu_{ji}, \phi_j) \\ \log(\mu_{ji}) & = & \eta_{ji} \\ \eta_{ji} & = & \sum_{l=1}^{L_{tot}} \sum_{c=1}^{C} s_{jlc}(T_{li}) Z_{ilc} + \mathbf{U}_i \boldsymbol{\alpha}_j + \log(N_i) \end{cases}, \tag{4.13}$$

where the mean μ_{ji} of the negative binomial distribution is linked to the additive predictor η_{ji} using a logarithmic link function. The $\mathbf{U}$ matrix represents an additional design matrix, representing, for example, a batch effect.

The model relies on lineage-specific, gene-specific, and condition-specific smoothers s_{jlc}, which are linear combinations of K cubic basis functions, $s_{jlc}(t) = \sum_{k=1}^{K} b_k(t)\beta_{jlck}$.

Testing for differential Expression. With this notation, we can introduce the `conditionTest`, which, for a given gene j, tests the null hypothesis that the smoothers are identical across conditions:

$$H_0 : \forall(c_1, c_2), \forall k, \forall l, \beta_{jlc_1k} = \beta_{jlc_2k}. \tag{4.14}$$

We fit the model using the mgcv package [173] and test the null hypothesis using a Wald test for each gene. Note that, although the gene expression model can be fitted without any mapping, the `conditionTest` only exists for lineages with at least a mapping for two conditions.

Simulation study

Simulating datasets

The simulation study relies on the dyngen framework of Cannoodt et al. [25] and all datasets are simulated as follows. 1/ A common trajectory is generated, with an underlying gene network that drives the differentiation along the trajectory. 2/ A set of N_{WT} cells belonging to the wild-type condition (i.e., with no modification of the gene network) is generated. 3/ One master regulator that drives differentiation into one of the lineages is impacted, by multiplying the wild-type expression rate of that gene by a factor m. If $m = 1$, there is no effect; if $m > 1$, the gene is over-expressed; and if $m < 1$, the gene is under-expressed, with $m = 0$ amounting to a total knock-out. 4/ A set of $N_{KO} = N_{WT}$ cells is generated using the common trajectory with the modified gene network. 5/ A common reduced-dimensional representation is computed.

We generate three types of datasets, over a range of values of m: a simple trajectory with $L = 2$ lineages and $C = 2$ conditions (WT and KO) named $\mathcal{T}_1$; a trajectory with two consecutive branchings with $L = 3$ lineages and $C = 2$ conditions (WT and KO) named $\mathcal{T}_2$; and a simple trajectory with $L = 2$ lineages and $C = 3$ conditions (WT, KO, and UP) named $\mathcal{T}_3$. For the latter case, Steps 3-4/ are repeated twice, with values of m for KO and $1/m$ for UP.

For $\mathcal{T}_1$ and $\mathcal{T}_2$, we use values of $m \in \{.5, .8, .9, .95, 1, 1/.95, 1/.9, 1/.8, 1/.5\}$, such that at the extremes the KO cells fully ignore some lineages. Values of .95 and 1/.95 represent the closest to no condition effect ($m = 1$), where the effect was still picked out by some tests. For $\mathcal{T}_3$, since the simulation is symmetrical in m, we pick $m \in \{.5, .8, .9, .95, 1\}$. We have one large dataset per value of m and per trajectory type. We use those large datasets to generate smaller ones of size n, by sampling 10% of the cells from each condition 50 times and applying the various tests on the smaller datasets. The reason for first generating a large dataset and then smaller ones by subsampling instead of generating small ones straightaway are computational: the generation of the datasets is time-consuming and the part that scales with N_{WT} can be parallelized. Hence, it is almost as fast to generate a large dataset than a small one with dyngen. We pick $N_{WT} = 20,000$ (for the large dataset) and thus $n = 2,000$.

Since we generate many datasets with true effect ($m \neq 1$) but only one null dataset, the size of N_{WT} for $m = 1$ is doubled to $40,000$. To be comparable, the fraction of cells

sampled is decreased to 5% so that $n = 2,000$ and we perform 100 subsampling. Table 4.2 recapitulates all this.

Table 4.2: *Summary of all simulated datasets.* We report the name, number of cells n_{WT} for values of $m \neq 1$ and $m = 1$, number of conditions C, number of lineages L, impacted master regulator, and figure numbers for the associated gene network and an example of low-dimensional representation.

Dataset	N_{WT}		n	L	C	Impacted Regulator	Gene Network Figure	Reduced Dimension Representation
	$m \neq 1$	$m = 1$						
$\mathcal{T}_1$	20,000	40,000	2,000	2	2	B3	Fig D.1a	Fig.4.2a
$\mathcal{T}_2$	20,000	40,000	2,000	3	2	D2	Fig D.1b	Fig.4.2b
$\mathcal{T}_3$	20,000	40,000	2,000	2	3	B3	Fig D.1a	Fig.4.2c

Measuring the performance of the tests on the simulated datasets

To run the condiments workflow, we first estimate the trajectories using slingshot with the clusters provided by dyngen. Then, we run the `progressionTest` and the `differentiationTest` with default arguments.

We compare condiments to two other methods. milo [34] and DAseq[182] both look at differences in proportions within local neighborhoods, using k-nearest neighbor graphs to define this locality. Then, milo uses a negative binomial GLM to compare counts for each neighborhood, while DAseq uses a logistic classifier test. Therefore, both methods test for differential abundance in multiple regions. To account for multiple testing, we adjust the p-values using the Benjamini, Yoav ; Hochberg [13] FDR-controlling procedure.

We select two adjusted p-value cutoffs, .01 and .05, which amount to controlling the FDR at nominal level 1% and 5%, respectively. For a given cutoff c and a given dataset, we can look at the results of each test on all simulated datasets for all values of m. For each test, the number of true positives (TP) is the number of simulated datasets where $m \neq 1$ and the adjusted p-value is smaller than c, the number of true negatives (TN) is the number of simulated datasets where $m = 1$ and the adjusted p-value is larger than c, the number of false positives (FP) is the number of simulated datasets where $m = 1$ and the adjusted p-value is smaller than c, and the number of false negatives (FN) is the number of subsampled datasets where $m \neq 1$ and the adjusted p-value is larger than c. We then examine 5 metrics built on these four variables:

$$\text{True Negative Rate (TNR)} = \frac{TN}{TN + FP}$$
$$\text{True Positive Rate (TPR)} = \frac{TP}{TP + FN}$$
$$\text{Positive Predictive Value (PPV)} = \frac{TP}{TP + FP}$$
$$\text{Negative Predictive Value (NPV)} = \frac{TN}{TN + FN}$$
$$\text{F1-score} = 2\frac{PPV \times TPR}{PPV + TPR}.$$

Case studies: Processing.

TGFB. The two conditions are normalized separately using SCTransform [50] and then integrated using Seurat [142]. The reduced-dimensional representation is computed using UMAP [11] on the top 50 principal components (PC). The imbalance score is computed with parameters $k = 20$ and $smooth = 40$. The trajectory is estimated using slingshot. The `topologyTest` is run with 100 permutations with the Kolmogorov-Smirnov test and default threshold of .01. The `progressionTest` is run with defaults. All genes with at least 2 reads in 15 cells are kept. The smoothers are fitted for each gene using 7 knots as recommended by the `evaluateK` function. Gene set enrichment analysis is done using the fgsea [74] package on the GO Biological Process ontology sets.

TCCD. The dataset is first filtered using the cell type assignments from the original publication to only retains cells labelled as hepatocytes. The count matrix is scaled using Seurat [142] and reduced-dimensional coordinates are computed using UMAP [11] on the top 30 PCs. The imbalance score is computed with default k and $smooth = 5$. The trajectory is estimated using slingshot. The `topologyTest` is run with 100 permutations with the Kolmogorov-Smirnov test and default threshold of .01. The `progressionTest` is run with defaults. All genes with at least 2 reads in 15 cells are kept; all genes with at least 3 reads in 10 cells are kept. The smoothers are fitted for each gene using 7 knots as recommended by the `evaluateK` function.

KRAS. The reduced-dimensional coordinates were obtained from the original publication. The imbalance score is run with defaults and the `topologyTest` is run with 100 permutations with the classifier test and default threshold of .01. The trajectories are estimated using slingshot with parameters $reweight = FALSE$ and $reassign = FALSE$. The `progressionTest` and `differentiationTest` are run with defaults. All genes with at least 5 reads in 10 cells are kept. The smoothers are fitted for each gene using 6 knots as recommended by the `evaluateK` function.

Mutinomial test

We consider a set of categories arbitrarily numbered from 1 to C. Additionally, we consider a null distribution $\mathbf{C}_0$, defined on 1 to C by a vector of probabilities $\mathbf{p} = \{p_c\}_{c=1}^{C}$. Then, given a set of n i.i.d. realizations $(c(1), \ldots, c(n))$ of a random variable $\mathbf{C}$, we can test the null hypothesis $H_0 : \mathbf{C} \sim \mathbf{C}_0$ or, equivalently, $H_0 : \mathbf{P}(\mathbf{C} = c) = p_c, \forall c \in \{1, \ldots, C\}$. Under the null, $\mathbf{P}(c_i) = p_{c_i}$ and the associated p-value of the multinomial test can be defined as:

$$P(x, \mathbf{p}) = \sum_{y \in \{1,\ldots C\}^n : \mathbf{P}_{H_0}(y) \leq \mathbf{P}_{H_0}(x)} \mathbf{P}_{H_0}(y).$$

It verifies: $\forall \alpha \in [0 : 1], \mathbf{P}_{H_0}(P(x, \mathbf{p}) \leq \alpha) \leq \alpha$.

Data and code availability

The results from this chapter can be fully reproduced by following along the vignettes at https://hectorrdb.github.io/condimentsPaper. These also contain the code needed to recreate the datasets used for the simulations, as well as processed versions of all three datasets used in the case studies, augmented by metadata and functions to recreate the processed versions, using raw counts obtained from GEO (**TGFB** dataset: GSE114687, **TCDD** dataset: GSE148339, **KRAS** dataset: GSE137912).

The condiments workflow is available as an R package from Github (https://github.com/HectorRDB/condiments) and will be made available through the Bioconductor Project.

All the methods to test for equality of two (or k) distributions have been put together for use by others in an R package called Ecume, available through CRAN and that can be explored at https://hectorrdb.github.io/Ecume.

Chapter 5

Enumeration of Closed Connected Subgraphs

As discussed in chapter 1, in bacterial genome-wide association studies, the sequences can be correctly represented using a compacted De Bruijn Graph that reflects the granularity of genetic variation. Here, we present `CALDERA`, an enumeration algorithm that can find all significant closed connected subgraphs at scale. The following sections will further motivated the problem in a more general context, provide background on existing methods. We will then introduce the `CALDERA` methods, show results on both real and simulated data, and discuss those. Finally, we provide the proofs for the theorems used in this chapter. A shorter version of this work was presented in poster format at the Machine Learning for Computational Biology (MLCB) conference in November 2020 [124][1].

[1] I would link to deeply thank my collaborators. Laurent Jacob was central both conducting and supervising the entirety of this project. Arnaud Mary provided essential ideas and rigour for the design of the method. Fanny Perraudeau and Sandrine Dudoit also provided feedback and support

5.1 Introduction

Networks are pervasive in molecular biology, where they are used to represent, for instance, gene regulations or interactions between proteins or metabolic pathways. They are also a major opportunity for statistical analysis, as many applications involve few samples and many descriptors, leading to high-dimensional problems. Exploiting the domain structure encoded in the network allows to both reduce the dimension and enforce more interpretable solutions in a variety of contexts ranging from classification of expression profiles [119, 61, 89] to the detection of disrupted pathways [60, 162, 161, 4]. In many such cases, the resulting formal problem is to test the association between an outcome and a large number of covariates defined from all connected subgraphs of the full network. This network connects binary covariates describing the data, and for every connected subgraph, a new covariate is created by taking a disjunction or a conjunction over its vertices, leading to a new data representation that embeds information encoded in the network. In genome-wide association studies (GWAS), for example, one seeks genes or positions along the genome whose mutation is associated with a phenotype of interest—e.g., a human disease or crop yield—in a dataset for which both the phenotype and all mutation statuses are observed. Given a network encoding known metabolic pathways, one may be interested in finding subnetworks such that mutating any of the genes involved in the corresponding set of reactions affects the phenotype [134]. As different mutations along the same pathway can lead to the same effect, the subnetwork is a more relevant unit for GWAS than the individual gene. Similarly, if the network is a simple chain representing a linear genome, this approach detects genome segments whose mutation is associated to the phenotype [82, 83].

Testing all subgraphs seems doomed for two reasons: (1) their number grows exponentially with the number of nodes in the network, making the task computationally intractable for most cases of interest, and (2) adjusting for multiple testing over this very large number of tests leaves little to no power to detect associations. Both problems have been addressed in comparable situations by exploiting the concept of testability introduced in [148]. The underlying idea of Tarone's testing procedure is that many of the tested covariates can be discarded by inspecting quantities that do not affect the test statistic. In the context of subgraphs, the procedure is also well suited to pruning strategies because most testability criterions verify a monotonicity property, and all subgraphs including a non-testable one can be safely ignored.

Most existing procedures are restricted to a particular type of graph. [151, 96] considered all possible interactions in a set of covariates, which amounts to considering a complete graph. [105] extended this procedure to tests that can control for a categorical feature, an essential point in GWAS, where confounders such as population structure can lead to many false discoveries. Llinares-López et al. [82, 83] introduced efficient methods to test the association between a phenotype and mutations anywhere in a genomic interval, corresponding to a linear graph. [144] extended this idea to frequent subgraph mining, a related task where the goal is to detect subgraphs whose presence in a larger graph is associated with a property of this graph. Finally, [134] described an algorithm to test all closed connected subgraphs

(CCS), i.e., connected subgraphs such that adding any neighbor does not affect the created covariate. Their algorithm combined the testability-based procedure `LAMP` of [151] with COIN [133], an enumeration method for CCSs. While no experiment was provided in [134] we found—using an improved version of LAMP [96, 82]—that this combination could find all significant CCSs in graphs with up to 20000 nodes in a day. A more scalable method is however warranted to make CCSs testing amenable to important modern applications such as bacterial GWAS, that involve millions of nodes [62]. Compared to linear and complete graphs, a major additional challenge for general graphs is enumeration and its interplay with pruning. Here, we introduce `CALDERA`, a scalable testing procedure for CCSs in general graphs. We rely crucially on a novel enumeration scheme that leads to more pruning than `COIN` when combined with Tarone's procedure.

Our contributions are the following: We introduce a novel, provably complete and non-redundant enumeration scheme for CCSs. We also improve an existing pruning criterion for the Cochran-Mantel-Haenszel test. We show that combining these contributions with Tarone's testability-based procedure makes it possible to find all significant CCSs in a large graph, making it suited to applications such as bacterial GWAS, a critical and contemporary problem for human health. We provide—in the Supplementary material—the first implementation of a procedure finding all significant CCSs.

Notation and goal We consider a set of n samples, $(x_i, y_i, c_i)_{i=1}^n$, where $x_i \in \{0,1\}^p$ are p binary covariates describing sample i, $y_i \in \{0,1\}$ denotes a binary phenotype, and $c_i \in \{1,\dots,J\}$ assigns sample i to one population among J. We denote n_1 and n_2 the number of samples such that $y_i = 0$ and 1 respectively. Furthermore, we consider an undirected unweighted connected graph $\mathcal{G} = (\mathcal{V}, E)$, where $\mathcal{V} = \{v_1, \dots, v_p\}$ and each vertex $v_j \in \mathcal{V}$ is associated with one of the p binary covariates represented in x. We denote by $\mathcal{I}(v_j) = \{i : x_i^j = 1\}$. For $i \in [1:n]$, we note $\mathbb{V}_i = \{v \in \mathcal{V} : i \in \mathcal{I}(v)\}$. For any connected subgraph $\mathcal{S} = (\mathcal{V}', E')$, such that $\mathcal{V}' \subseteq \mathcal{V}$ and $E' \subseteq E$, we let $\mathcal{I}(\mathcal{S}) = \bigcap_{v \in S} \mathcal{I}(v)$. The set of all connected subgraphs of $\mathcal{G}$ is denoted by $\mathcal{A}$. Of note, this framework addresses both disjunctions and conjunctions, as the latter can simply be obtained by replacing each x_i by its complement. We now properly define the notion of closed connected subgraph. The validity of this operation is proved in Supplementary 5.7.

Definition 3 *A connected subgraph $\mathcal{S} \in \mathcal{A}$ is closed if and only if there exists no edge $(v_1, v_2) \in E$ such that $v_1 \in \mathcal{S}$, $v_2 \notin \mathcal{S}$, and $\mathcal{I}(\mathcal{S} \bigcap \{v_2\}) = \mathcal{I}(\mathcal{S})$. We denote by $\mathcal{C} \subseteq \mathcal{A}$ the set of all closed connected subgraphs of $\mathcal{G}$.*

Assuming that $(x_i, y_i, c_i)_{i=1}^n$ are n i.i.d. realizations of random variables $\mathbf{X}, \mathbf{Y}$, and $\mathbf{C}$, our objective is to test null hypotheses of the form $H_0^{\mathcal{S}}(\mathbf{X}, \mathbf{Y}, \mathbf{C}) : \big(\mathcal{I}(S) \perp \mathbf{Y}\big)|\mathbf{C}$ for all $\mathcal{S} \in \mathcal{C}$, while controlling the family-wise error rate (FWER, i.e., the chance of at least one Type I error or false positive) at level α. Translated in the context of GWAS, we want to test the association between the pattern $\mathcal{I}(\mathcal{S})$ of each closed connected subgraph

$\mathcal{S}$ with the phenotype $\mathbf{Y}$, while controlling for the population structure $\mathbf{C}$. We denote $H_0(\mathcal{S}) = H_0^{\mathcal{S}}(\mathbf{X}, \mathbf{Y}, \mathbf{C})$ in the remainder of this manuscript, as $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{C}$ are common for all elements of $\mathcal{C}$.

5.2 Background on significant subgraph detection using testability

Here, we describe the important concept of minimal attainable p-value proposed by [148], and how it can be used to sharpen the threshold required to control the family-wise error rate compared to the Bonferroni correction, leading to more rejections. We also discuss how the same idea can be used in pruning strategies, leading to computationally efficient procedures, and how these ideas extend to test statistics that adjust for a categorical covariate.

Using minimal attainable p-values for a tighter FWER control

The Bonferroni correction [17] is a common procedure to control the FWER at a level α. A null hypothesis is rejected if its p-value is smaller than $\frac{\alpha}{N}$, where N is the total number of tested null hypotheses. In our context, the smallest p-value therefore needs to be smaller than $\frac{\alpha}{|\mathcal{C}|}$ for the procedure to reject at least one hypothesis. As described in Tarone [148], discrete tests admit a minimal attainable p-value $p^\star$, which can be used to control the FWER with a substantially smaller correction factor than N. Since the distribution only takes a finite number of values, this minimal p-value can be strictly larger than zero for some tests. For example, Fisher's exact test [43] relies on 2×2 contingency table, conditioning on the observed margins–in our case, for a given n, the number of samples with phenotype 1 and the number of samples such that $i \in \mathcal{I}(S)$. Under the null hypothesis, the two represented factors are independent. Given the margins, only a finite number of cell count partitions are possible and the p-value of Fisher's test can only take on a finite number of values, the smallest of which is strictly positive (Fig E.1). Importantly, this minimal attainable p-value $p^\star$ is entirely determined by the margins of the contingency table: given these margins, $p^\star$ is the minimum over a finite number of possible partitions, and is independent from the actual cell counts.

The Bonferroni correction is motivated by a simple union bound: the FWER is upper-bounded by the sum of the rejection probabilities over the N tested hypotheses. Since each of the individual tests is controlled at level $\frac{\alpha}{N}$, the sum is upper-bounded by α. However, rejection can only occur for the subset of m hypotheses such that $p^\star < \frac{\alpha}{N}$. Since $p^\star$ only depends on the data through margins and because inference is made conditional on these margins, this subset is deterministic and the FWER is actually controlled at level $\frac{m\alpha}{N} \leq \alpha$. This suggests that using a lower threshold than the Bonferroni $\frac{\alpha}{N}$ could still control the FWER at level α while rejecting more hypotheses. Defining $m(k)$ as the number of hypotheses such that $p^\star < \frac{\alpha}{k}$, the lowest threshold guaranteeing such a control is $\frac{\alpha}{k_0}$, where k_0 is the smallest k such that $m(k) \leq k$.

Variable	$i \in \mathcal{I}(\mathcal{S})$	$i \notin \mathcal{I}(\mathcal{S})$	Rows totals
$\mathbf{y_i = 1}$	$a_{\mathcal{S},j}$	$n_{1,j} - a_{\mathcal{S},j}$	$n_{1,j}$
$\mathbf{y_i = 0}$	$x_{\mathcal{S},j} - a_{\mathcal{S},j}$	$n_{2,j} - x_{\mathcal{S},j} + a_{\mathcal{S},j}$	$n_{2,j}$
Cols Totals	$x_{\mathcal{S},j}$	$n_j - x_{\mathcal{S},j}$	n_j

Table 5.1: Association table in community j for subgraph $\mathcal{S}$, used for the CMH test.

Using minimal attainable p-values to efficiently explore $\mathcal{C}$

Provided that enough CCSs have sufficiently large $p^\star$, Tarone's procedure could therefore address the loss of power incurred when exploring $\mathcal{C}$. However, naively finding k_0 requires to compute the minimal p-values for all $|\mathcal{C}|$ CCSs, leaving the computational problem unsolved. In practice, non-exhaustive strategies have been proposed to determine k_0. The most efficient one [82, 96] starts from $k = 1$ and increments a set $\mathcal{R}$ of *testable* hypotheses, *i.e.*, of elements with $p^\star < \frac{\alpha}{k}$. When $|\mathcal{R}|$ becomes larger than k, k is incremented to $|\mathcal{R}|$. All hypotheses that are not testable anymore under the new threshold—*i.e.*, such that $\frac{\alpha}{|\mathcal{R}|} \leq p^\star < \frac{\alpha}{k}$—are removed from $|\mathcal{R}|$, and the exploration continues until the point where all testable hypotheses are in $\mathcal{R}$ and $k = k_0$.

In addition to avoiding a full enumeration by stopping when enough testable hypotheses have been found, this search algorithm for k_0 is well suited to pruning strategies—a fact already used in [82, 96]. Let $p^\star(\mathcal{S})$ be the minimal p-value associated with $H_0(\mathcal{S})$ for a test at hand. Assuming that for some pairs of subgraphs $\mathcal{S}_1, \mathcal{S}_2$, $\mathcal{S}_1 \subseteq \mathcal{S}_2 \Rightarrow p^\star(\mathcal{S}_1) \leq p^\star(\mathcal{S}_2)$, we can stop exploring all subgraphs including $\mathcal{S}_1$ as soon as $\mathcal{S}_1$ itself is found non-testable. This monotonicity property is verified when using Fisher's exact test to test $H_0(\mathcal{S})$: provided that $|\mathcal{I}(\mathcal{S})| \geq \max(n_1, n_2)$, $p^\star$ is strictly increasing in $|\mathcal{I}|$, and adding nodes to $\mathcal{S}$ can only increase $|\mathcal{I}|$ (Figure E.2).

Controlling for a categorical covariate: the Cochran-Mantel-Haenszel (CMH) test

When testing for associations, controlling for confounders is essential to avoid spurious discoveries. This is particularly important in bacterial GWAS, where strong population structures can lead to large sets of clade-specific variants to be found associated with a phenotype. The CMH test can be used to test associations of two binary variables while controlling for a third categorical variable. It relies on J two-by-two association tables such as the one in Table 5.1, with $j \in \{1, \ldots, J\}$, $a_{\mathcal{S},j} = \big|\{i : y_i = 1, i \in \mathcal{I}(\mathcal{S}), c_i = j\}\big|$, $x_{\mathcal{S},j} = \big|\{i : i \in \mathcal{I}(\mathcal{S}), c_i = j\}\big|$ and $n_{1,j} = \big|\{i : y_i = 1, c_i = j\}\big|$.

Like Fisher's exact test, the CMH test is done conditional on all margins $(x_{\mathcal{S},j}, n_{1,j}, n_{2,j})_{j=1}^{J}$. Papaxanthos et al. [105] furthermore demonstrated that this minimal p-value could be computed in $O(J)$ (proof also in Supplementary 5.7) using the margins. However, the minimal p-

value of the CMH test does not verify the monotonicity property $\mathcal{S}_1 \subseteq \mathcal{S}_2 \Rightarrow p^\star(\mathcal{S}_1) \leq p^\star(\mathcal{S}_2)$ which is required to prune while exploring $\mathcal{C}$. Papaxanthos et al. [105] introduced the envelope, a lower bound on $p^\star(\mathcal{S})$, which verifies the monotonicity property. It can also be computed in $O(J \log(J))$ for all $\mathcal{S}$ such that, for all categories j, $x_{\mathcal{S},j} \geq \max(n_{1,j}, n_{2,j})$. This allows for a valid pruning strategy. The condition on $x_{\mathcal{S},j}$ is the CMH analogous of the $|\mathcal{I}(\mathcal{S})| \geq \max(n_1, n_2)$ condition of Fisher's test, and can decrease the number of prunable subgraphs as it must be verified for all J groups.

5.3 Speeding up the detection of all significant CCSs with CALDERA

We are now ready to present our contributions for scalable detection of significants elements in $\mathcal{C}$: an efficient exploration algorithm and an improved envelope for the CMH test, allowing for more pruning in the presence of imbalanced populations.

Critical properties for a fast, Tarone-aware enumeration of $\mathcal{C}$

The testing procedure described in Section 5.2 relies on an exploration of the set of hypotheses—in our setting, one for each element of $\mathcal{C}$. The scalability of the testing procedure is affected by both the computational behavior—speed and memory footprint–of the exploration scheme itself, and its ability to take advantage of the pruning opportunity offered by the Tarone procedure.

We exploit several factors to provide a fast exploration. First, we ensure that it is non-redundant, *i.e.*, that each element of $\mathcal{C}$ is enumerated exactly once. More precisely, we define a tree structure whose nodes are the elements of $\mathcal{C}$ and propose an algorithm to traverse this tree. Second, the tree is directly built over $\mathcal{C}$, as opposed to the set $\mathcal{A} \supset \mathcal{C}$ of connected subgraphs. The latter option, as proposed in [133] is more straightforward to define and to explore and would still induce a tree over $\mathcal{C}$, but would yield a much larger object and result in a more expensive traversal. Third, we propose an exploration scheme that does not to rely on a mechanism maintaining subgraph connectivity such as a block-cut tree [168]. Such a mechanism is efficient to build a tree over connected subgraphs but is costly to compute. Finally, we also minimize the reliance on itemtables to store visited elements, therefore limiting the memory footprint of our exploration. Scalability also depends on the ability of the exploration of $\mathcal{C}$ to exploit the pruning opportunity offered by the testing procedure. As highlighted by [96, 82], exploration should follow subgraph inclusion, *i.e.*, ensure that all subgraphs $\mathcal{S}'$ explored from a subgraph $\mathcal{S}$ are such that $\mathcal{S}' \supsetneq \mathcal{S}$. This way, the exploration can be stopped from any element that is found non-testable. In order to enforce this behavior, our tree over $\mathcal{S}$ must be such that the children of a node representing $\mathcal{S} \in \mathcal{C}$ always represent subgraphs $\mathcal{S}' \subsetneq \mathcal{S}$.

Haraguchi et al. [53] and Okuno et al. [103] define a tree on $\mathcal{C}$, but the root of the tree corresponds to the entire graph $\mathcal{G}$: the inclusion relationship along edges of the tree is

Algorithm 1 Children of $\mathcal{S}$

```
1: procedure CHILDREN(S, S_p, i, T)
2:     children← ∅
3:     for k, G in enumerate(EqGroups(S)) do
4:         v ← G[0]
5:         S' ← cl(S ⋃ {v})
6:         if i is NULL then
7:             if (S, S') verify (1-3) then
8:                 Add S' to siblings
9:                 Add Children(S', S, i_S', T = ∅) to children
10:            end if
11:        else if (S_p, S') verify (1-3) then
12:            if i_S' = i and {I ∈ T : I ⊂ I(S')} = ∅ then
13:                T' = T ⋃ {I_1(S'), ..., I_{k-1}(S')}
14:                Add Children(S', S_p, i_S', T') to children
15:            end if
16:        end if
17:    end for
18:    return children
19: end procedure
```

the opposite to the one we need, making their exploration unsuited to our problem. The `COIN/COPINE` algorithm described in Seki and Sese [131] and Sese, Seki, and Fukuzaki [133] builds a tree over the set of connected subgraphs, which induces a tree over $\mathcal{C}$. `COIN` maintains an itemtable to enforce a tree structure by avoiding the enumeration of the same element twice. This itemtable has an important memory footprint, and only guarantees a tree structure when exploring in depth first. Finally, the enumeration of connected subgraphs requires to maintain a list of articulation points along each explored branch, a costly operation. We now describe an exploration scheme that verifies the properties listed in this section.

Defining and exploring the tree over $\mathcal{C}$

In order to build a tree over $\mathcal{C}$ rooted on the empty CCS, we use a reverse search, introduced in [3]. Reverse search relies on a reduction operation, which takes one element of the set to be enumerated, and returns a unique, strictly smaller element of the same set. This operation necessarily defines a tree over the elements of the set, by ensuring a unique path between any element and the empty one—the root of the tree. This reduction operation defines the unique parent of every element in the tree. In order to traverse the tree from the root, one needs to inverse the reduction operation, *i.e.* in our setting, given a CCS $\mathcal{S}$ to recover all CCSs that lead to $\mathcal{S}$ by reduction. Here we introduce a reduction operation over $\mathcal{C}$, as well as its inversion. We consider the parent operation $\mathcal{P}$ given by Definition 4 for any element

of $\mathcal{C}$, and show that it defines a valid reduction as introduced above.

Definition 4 *For a subgraph $\mathcal{S} \in \mathcal{C}$, we denote $\mathcal{J}(\mathcal{S})_i = \bigcap_{v \in \mathcal{S}} \mathcal{I}(v)$.*

- *If $\mathcal{I}(\mathcal{S}) = \mathcal{J}(\mathcal{S})$, then the parent of $\mathcal{S}$, $\mathcal{P}(\mathcal{S})$ is $\emptyset$.*
- *Else we note $i_{\mathcal{S}} = \max(\mathcal{I}(\mathcal{S}) \setminus \mathcal{J}(\mathcal{S}))$. The parent $\mathcal{P}(\mathcal{S})$ of $\mathcal{S}$ is the connected subgraph of $\mathcal{S} \setminus \mathbb{V}_{i_{\mathcal{S}}}$ that contains $\max \mathcal{S} \setminus \mathbb{V}_{i_{\mathcal{S}}}$.*

Lemma 1 *The function $\mathcal{P}$ defines a valid reduction over $\mathcal{C}$.*

Note that we have $\mathcal{S} \supsetneq \mathcal{P}(\mathcal{S})$ for all $\mathcal{S}$ so this structure allows pruning. Lemma 2 then provides necessary and sufficient conditions for $\mathcal{S}' \in \mathcal{C}$ to be a child of $\mathcal{S} \in \mathcal{C}$:

Lemma 2 *For $\mathcal{S}, \mathcal{S}' \in \mathcal{C}$, $\mathcal{S} = \mathcal{P}(\mathcal{S}')$ if and only if the three following conditions are verified:*

(C1) $i_{\mathcal{S}'} \notin \mathcal{I}(\mathcal{S})$

(C2) $\max\{v' \in \mathcal{S}' \setminus \mathbb{V}_{i_{\mathcal{S}'}}\} = \max \mathcal{S}$

(C3) $\{v' \in \mathcal{S}' \setminus \mathbb{V}_{i_{\mathcal{S}'}} : v' \in Ne(\mathcal{S})\} = \emptyset$

Interestingly, the reduction itself is never used in the exploration, only its inverse. Besides, using (C1–3) in Lemma 2 to check whether $\mathcal{S} = \mathcal{P}(\mathcal{S}')$ for any $\mathcal{S}'$ does not require to identify the connected components of $\mathcal{S}' \setminus \mathbb{V}_{i_{\mathcal{S}'}}$, even though the reduction $\mathcal{P}$ itself does rely on these connected components. This property of the inverse reduction is critical for the scalability of `CALDERA`, as repeatedly identifying or maintaining these components would be very costly. It results from the fact that the reduction operation $\mathcal{P}$ does not maintain connectivity—it only retains one of the components obtained by removing nodes with $i_{\mathcal{S}}$. Doing so comes at a price: finding the children of $\mathcal{S}$ is not straightforward, as we must identify and reconnect all the connected components involved: Lemma 2 only provides a way to check if a candidate $\mathcal{S}'$ is a child of $\mathcal{S}$.

Algorithm 2 List significant closed connected subgraphs

1: **procedure** LIST_SIG_CLOSED_SUBGRAPHS($\mathcal{G}, \alpha$)
2: $Q \leftarrow Children(\emptyset, \emptyset$,NULL$, \emptyset)$
3: $\mathcal{R} \leftarrow \emptyset$
4: $k \leftarrow 1$
5: **while** $Q \neq \emptyset$ **do**
6: $\mathcal{S} \leftarrow$ `Dequeue`(Q)
7: **if** $p^\star(\mathcal{S}) \leq \alpha/k$ **then**
8: $\mathcal{R} \leftarrow \mathcal{R} \cup \{\mathcal{S}\}$
9: **end if**
10: **if** $\left|\mathcal{R}\right| > k$ **then**
11: $k \leftarrow k + 1$
12: $\mathcal{R} \leftarrow \{\mathcal{S} \in \mathcal{R} : p^\star(\mathcal{S}) \leq \alpha/k\}$
13: **end if**
14: **if** $\tilde{p}^\star(\mathcal{S}) \leq \alpha/k$ **then**
15: **for** $\mathcal{S}' \in$ `Children`$(\mathcal{S}, \mathcal{S}$,NULL$, \emptyset)$ **do**
16: `Enqueue`$(\mathcal{S}', Q)$
17: **end for**
18: **end if**
19: **end while**
20: Solutions$\leftarrow \emptyset$
21: **for** $\mathcal{S} \in \mathcal{R}$ **do**
22: **if** $p(\mathcal{S}) \leq \alpha/k$ **then**
23: Add $\mathcal{S}$ to Solutions
24: **end if**
25: **end for**
26: **return** Solutions
27: **end procedure**

For any subgraph $\mathcal{S}$, we further note $Ne(\mathcal{S}) = \{v \in \mathcal{G} \setminus \mathcal{S} : \exists v_1 \in \mathcal{S}, (v, v_1) \in E\}$ the set of neighbouring nodes of $\mathcal{S}$. We can partition $Ne(\mathcal{S})$ in equivalence groups of neighbours with regard to the pattern. An equivalence group $G_k(\mathcal{S}) \subset Ne(\mathcal{S})$ verifies: $v_1, v_2 \in G_k(\mathcal{S}) \implies \mathcal{I}(\mathcal{S} \bigcup \{v_1\}) = \mathcal{I}(\mathcal{S} \bigcup \{v_2\})$. We name $\mathcal{I}_k(\mathcal{S})$ the pattern of the equivalence group $G_k(\mathcal{S})$. With this notation, we define Algorithm 1. By Theorem 1, Algorithm 1 solves the problem of inverting the reduction, and therefore of building a tree structure on $\mathcal{C}$.

Theorem 1 *For any $\mathcal{S} \in \mathcal{C}$, Algorithm 1 returns the set $\{\mathcal{S}' \in \mathcal{C} : \mathcal{S} = \mathcal{P}(\mathcal{S}')\}$.*

Efficient implementation of `CALDERA`

In practice, we do not need to store the full table $\mathcal{T}$ in order to verify the second condition of Algorithm 1, Line 12. We rely on a concept from [155] to reduce memory footprint.

We consider a subgraph $\mathcal{S}'$ created following Algorithm 1, in the second case. We therefore have $\mathcal{S}, \mathcal{S}_p$ such that: $\mathcal{P}(\mathcal{S}') = \mathcal{P}(\mathcal{S}) = \mathcal{S}_p$ and $i_{\mathcal{S}'} = i_{\mathcal{S}}$. After creating $\mathcal{S}'$, we explore its children, with an itemtable $\mathcal{T}$. All elements of $\texttt{Children}(\mathcal{S}', \mathcal{S}_p, i_{\mathcal{S}'}, \mathcal{T})$ will have a pattern which includes $\mathcal{I}(\mathcal{S}')$. Moreover, by definition of the equivalence groups, we already know that $\{\mathcal{I} \in \mathcal{T} : \mathcal{I} \subset \mathcal{I}(\mathcal{S}')\} = \emptyset$. Therefore, when constructing $\mathcal{S}'' \in \texttt{Children}(\mathcal{S}', \mathcal{S}_p, i_{\mathcal{S}'}, \mathcal{T})$, only the elements in $\mathcal{I}(\mathcal{S}'') \setminus \mathcal{I}(\mathcal{S}')$ need to be considered.

We store $\mathcal{T}$ as a matrix of binary patterns. Therefore, some columns can be deleted without loss of information: in Line 13 of algorithm 1, we only keep the columns that are not in $\mathcal{I}(\mathcal{S})$. As the `Children` function is called recursively, the itemtable $\mathcal{T}$ will grow in the number of patterns saved (*i.e* number of rows) but the memory footprint of each pattern will be smaller (*i.e* fewer columns).

A breadth-first-search enumeration

We argue that exploring any tree structure on $\mathcal{C}$ in breadth first will often allow for more pruning than in depth first. At any level, even if the CCSs visited along a branch do increase k and therefore lower the testability threshold, all the other CCSs of the level will need to be visited regardless of their testability. By contrast, the increase of k gained by visiting all CCSs of the same level in the tree will lower the threshold α/k for all CCS at the next level, making more branches prunable. Section 5.4 provides illustrations of this phenomenon on simplified examples and we demonstrate this in section 5.5 on simulation and real-world data. A search in breadth is also easily parallelized since the computation of the minimal p-value, the envelope and the childrens of every CCS of a given level can be done in parallel, before increasing k and updating $\mathcal{R}$. By contrast, a parallelized search in depth-first would need to share and regularly update k and $\mathcal{R}$, which negates the advantages of parallelization.

Algorithm 2 explores $\mathcal{C}$ through a BFS traversal of the tree defined by the reduction $\mathcal{P}$, exploiting Algorithm 1 (L.15) to invert the reduction and using this exploration to apply the Tarone testing procedure described in Section 5.2 (L7-12, 14), before finally testing the testable CCS(L21-25).

Pruning more CCSs when controlling for an imbalanced categorical covariate

The envelope $\tilde{p}^\star(\mathcal{S}) = \min_{x' \geq x_{\mathcal{S}}} p^\star(\mathcal{S})$ introduced in Papaxanthos et al. [105] verifies the monotonicity for any subgraph $\mathcal{S}$ because $\mathcal{S}' \supseteq \mathcal{S} \Rightarrow x_{\mathcal{S}'} \geq x_{\mathcal{S}}$. However, the $O(J \log J)$ algorithm to compute this envelope only applies to the so called *potentially prunable* subgraphs which are such that $x_{\mathcal{S},j} \geq \max(n_{1,j}, n_{2,j})$ for all subgroups $j = 1, \ldots, J$ defined by the categorical covariate adjusted for by the CMH test. Pruning can therefore not be done from subgraphs for which at least one of the J groups has few occurrences of the corresponding covariate. This limitation arises in Lemma 2 of Papaxanthos et al. [105], which characterizes the argmin of the envelope of a subgraph $\mathcal{S}$. Lemma 3 lifts this restriction:

Lemma 3 *For any connected subgraph $\mathcal{S}$, the envelope $\tilde{p}^\star$ is attained for an optimum $x^*_{\mathcal{S}'}$ such that $x^*_{\mathcal{S}',j} \in \{\max(x_{\mathcal{S},j}, n_{1,j}), \max(x_{\mathcal{S},j}, n_{2,j}), n_j\}$.*

The proof is provided in the latter section 5.7. Lemma 3 exploits a cruder bound for groups that are not in the increasing regime of the minimal p-value. Accordingly, it recovers the Lemma 2 of Papaxanthos et al. [105] for potentially prunable subgraphs, while offering an additional pruning opportunity for the other ones. If a subgraph was not potentially prunable only because it was missing the $x_{\mathcal{S},j} \geq \max(n_{1,j}, n_{2,j})$ condition for one small group j, it may still be actually prunable since small groups of samples only affect the CMH test statistic marginally. On the other hand if the condition is not verified for a large group or several small ones, the resulting envelope will be very lose and will not allow for pruning in practice. We illustrate this phenomenon in the next section through an example.

Generating the simulated dataset

For a given value of n and p, we first generate n samples with phenotype $y_i \in \{0, 1\}$ such that $\mathbf{P}(y_i = 0) = 0.5$. Then, we generate p nodes. 10% of the nodes will be associated with the phenotype. For each node in the remaining 90%, we randomly generate 3 edges between this node and another in the 90%. The average degree is therefore 6. For those nodes v_j, the associated pattern $\mathcal{I}(v_j)$ is a random vector such that $\mathbf{P}(i \in \mathcal{I}(v_j)) = 0.5$.

Then, we generate the remaining 10% of the nodes associated with the phenotype. We first generate associated patterns $\mathcal{I}_{sig}$ such that $\mathbf{P}(i \in \mathcal{I}_{sig} | y_i = 1) = 0.95$ and $\mathbf{P}(i \in \mathcal{I}_{sig} | y_i = 0) = 0.05$. Then, those patterns are split into 10 significant nodes sig_j such that $\mathbf{P}(i \in \mathcal{I}(\text{sig}_j) | i \in \mathcal{I}_{sig}) = 0.9$ and $\mathcal{I}(\bigcup_{j \in [1...10]}) = \mathcal{I}_{sig}$.

5.4 Examples

Benefits of the new envelope

To demonstrate the situations where the new envelope introduced in 3is beneficial, and where it is not, we consider a situation where $n = 280, J = 2$. Then, we look at two cases: $n_1 = n_2 = 140$ and $n_1 = 13 \times n_2 = 260$. In both settings, we compute the envelope as defined in [105] and in our case. Then, for $\alpha = 10^{-8}$ (a value that we used in practice) and for all possible values of $\{x_{\mathcal{S},1}, x_{\mathcal{S},2}\}$, we consider whether we would prune (for $k = 1$) using the definition of the envelope from Papaxanthos et al. [105] or the extended new bound defined in this paper. The new bound nearly doubles the space of prunable subgraphs when there is a clear imbalance, as evidenced in Fig 5.1b, while it has no effect when the two populations are perfectly balanced, as in Fig 5.1a.

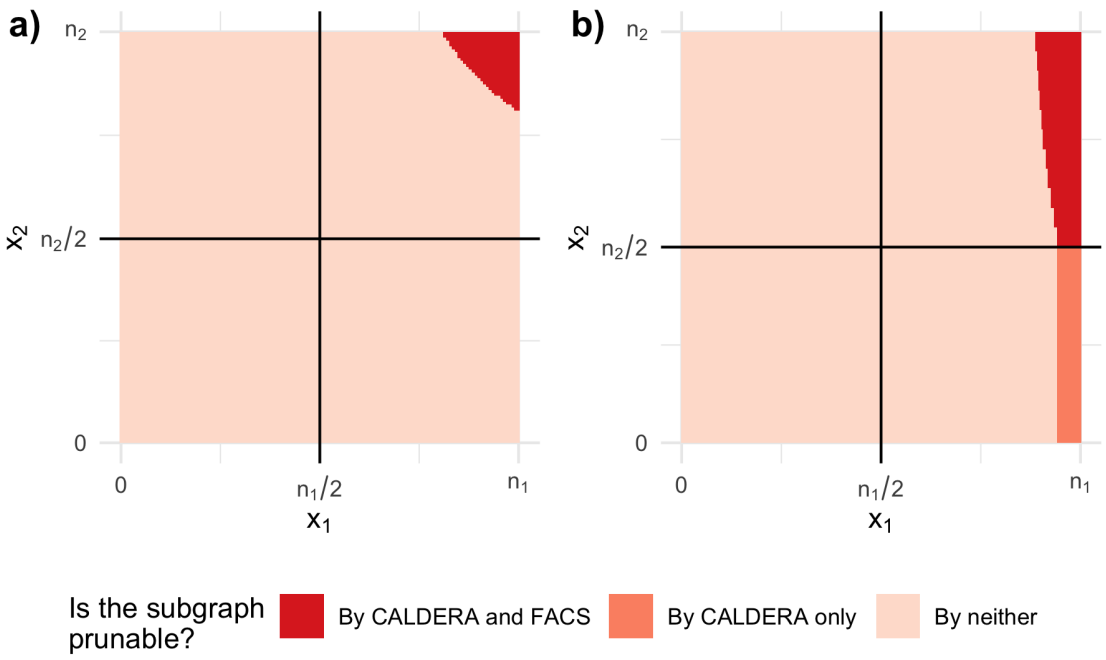


Figure 5.1: *Intuition for the effectiveness of the new bound* We consider the space of all possible patterns for $n = 280, J = 2$ and two cases: a) $n_1 = n_2 = 140$ and b) $n_1 = 260 = 13 \times n_2$. The phenotypes are well balanced in each population and $alpha = 10^{-8}$. The extended lower bound increases the number of prunable subgraphs when the populations are imbalanced.

Benefit of breadth-first search

First example: simplified scenario

We consider a very simple graph with $p = 3$ nodes, $J = 1$ population and $n = 12$ samples. The graph is displayed in Fig 5.2a. Using the reduction from `CALDERA`, we generate a tree structure on $\mathcal{C}$, displayed in Fig 5.2b.

Then we can explore this structure in depth-first or breadth-first, while pruning using $\alpha = 1$. The order resulting from an exploration in depth-first can be found in Table 5.2 and the order from the exploration in breadth-first can be found in Table 5.3. In this simple setting, exploring in breadth only visits 4 subgraphs while exploring in depth visits 7. This is because the BFS enumerates testable subgraphs more quickly, thereby increasing k and lowering the threshold, which means that the branch starting at $\{v_1\}$ is pruned earlier in the exploration.

Subgraph explored	Number of subgraphs explored	Value of the threshold	Testable subgraphs
$\{v_1\}$	1	.15	$\{\{v_1\}\}$
$\{v_1, v_2\}$	2	.15/2	$\{\{v_1\}, \{v_1, v_2\}\}$
$\{v_1, v_2, v_3\}$	3	.15/2	$\{\{v_1\}, \{v_1, v_2\}\}$
$\{v_1, v_3\}$	4	.15/3	$\emptyset$
$\{v_2\}$	5	.15/3	$\{\{v_2\}\}$
$\{v_2, v_3\}$	6	.15/3	$\{\{v_2\}, \{v_2, v_3\}\}$
$\{v_3\}$	7	.15/3	$\{\{v_2\}, \{v_2, v_3\}, \{v_3\}\}$

Table 5.2: Order of exploration of the elements of $\mathcal{C}$ while exploring depth-first

Subgraph explored	Number of subgraphs explored	Value of the threshold	Testable subgraphs
$\{v_1\}$	1	.15	$\{\{v_1\}\}$
$\{v_2\}$	2	.15/2	$\{\{v_1\}, \{v_2\}\}$
$\{v_3\}$	3	.15/3	$\{\{v_2\}, \{v_3\}\}$
$\{v_2, v_3\}$	4	.15/3	$\{\{v_2\}, \{v_2, v_3\}, \{v_3\}\}$

Table 5.3: Order of exploration of the elements of $\mathcal{C}$ while exploring breadth-first

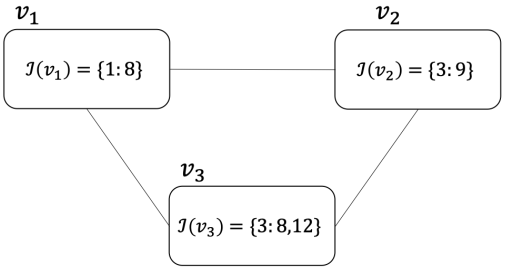
Second example: more general setting

We consider a very simple graph model where, for $v \in \mathcal{V}$ and $i \in \{1, \dots, n\}$, $i \in \mathcal{I}(v) \sim$ Binom(prop) and the patterns are independent across nodes . We have no population structure, which means that we consider Fisher's exact test. For a given level α, we want to compute $f(\alpha, \text{prop}) = \mathbb{P}(p^\star(\{v\}) > \alpha, \forall v \in \mathcal{V})$, that is the probability that no subgraph is testable at the first stage of our tree on $\mathcal{C}$.

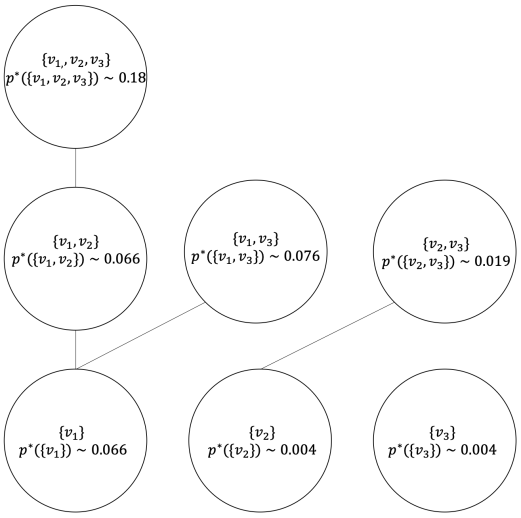
Since we consider Fisher's exact test, there is a bijection between $p^\star(\{v\})$ and $x_{\{v\}}$ so $p^\star(\{v\}) > \alpha \implies x_{\{v\}} \geq \sigma(\alpha)$. Moreover, $x_{\{v\}} \sim \mathcal{B}(\text{prop}, n)$, so $f(\alpha, \text{prop}) = 1 - \left(\mathbf{F}_{\mathcal{B}\backslash\backslash\mathbb{1}(\text{prop},n)}(\sigma_\alpha)\right)^p$ with $\mathbf{F}_{\mathcal{B}\backslash\backslash\mathbb{1}(\text{prop},n)}$ the cumulative distribution function of the binomial (prop, n). Since the nodes are independent, the distribution of $x_{\mathcal{S}}$ at any stage of the tree can be computed by recursion. We furthermore assume that the graph structure is such that the number of closed subgraphs is $s \times p$ at stage s.

In Fig 5.2c, we display the probability that any subgraph is 1-testable or prunable at stage s, for $s \in \{1, 2, 3\}$, $p = 100$ and $\alpha = 10^{-4}$.

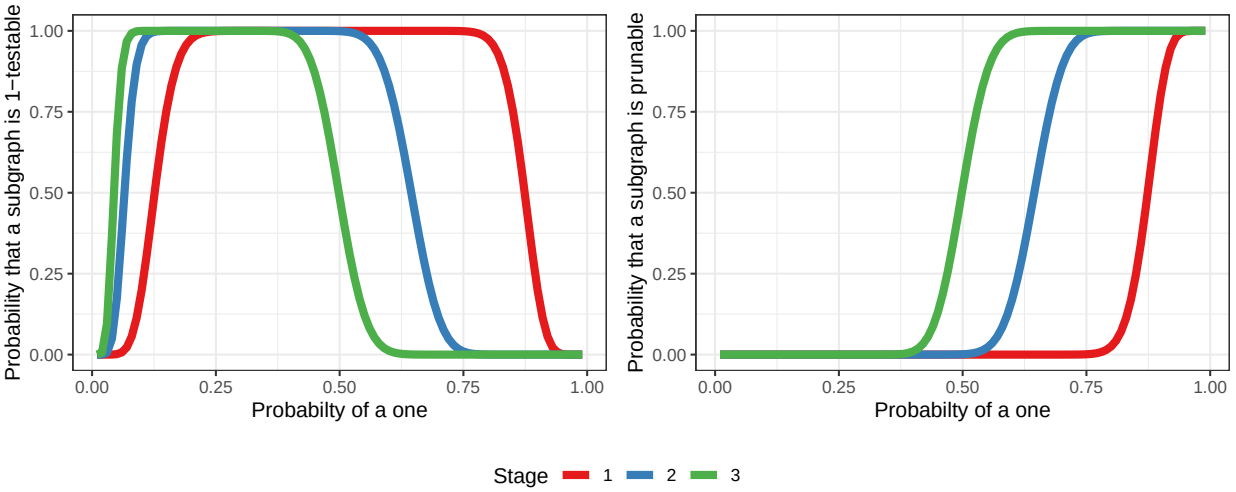
For most of the range of values, there is at least one testable subgraph in the first stage. So, by exploring in a BFS manner, we start the second stage with a much lower threshold (i.e., a much higher value of k) which leads to more pruning. For very low values of `prop`, there might be no testable subgraphs at the first stage but there will be at the second stage, which still justifies an exploration in depth. Note that for large p, we can see that there is no testable subgraph at the stages 2 and 3. That is because all such subgraphs have a pattern



(a) A simple graph example with $p = 3$, $J = 1$ and $N = 12$ which we explore using `CALDERA` with pruning, and $\alpha = .15$



(b) Order on elements of $\mathcal{C}$ from the graph in a), according to the reduction of definition 4



(c) $n = 100, p = 100, \alpha = 10^{-4}$. For a simple model described in 5.4 where $i \in \mathcal{I}(v) \sim \text{Binom}(\text{prop})$, we plot the probability that any subgraph is 1-testable or prunable as a function of prop

Figure 5.2: Simple examples where the search in breadth-first is much more efficient that depth-first

that is too large. While there may be not testable subgraphs, there are many prunable ones. In that case, an exploration in breadth-first or depth-first would be identical.

This example simplifies two aspects which have opposite effects. The first is that, in practice, the probability of $i \in \mathcal{I}(v)$ is of course not uniform across the graph. It is a distribution with much heavier tails which means that, even if the average number of 1

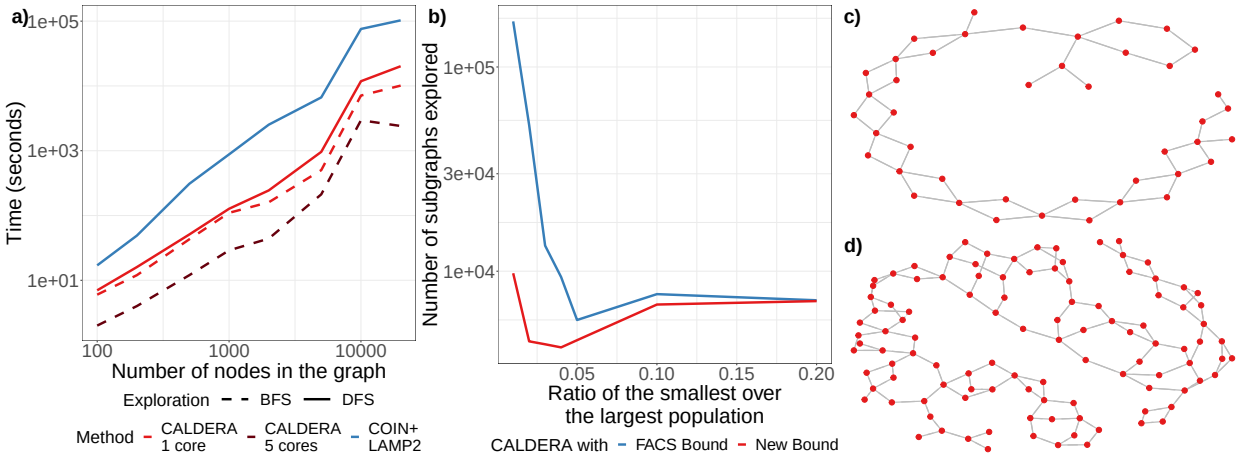


Figure 5.3: *Results of CALDERA.* **a.** Runtimes for `CALDERA` and `COIN+LAMP` on graphs with various values of p. In this setting, $n = 100$. **b.** Number of subgraphs explored when pruning using the envelope from `FACS` or the new envelope, depending on the ratio n_1/n_2. In this setting, $n = 200$ and $p = 3000$. **c, d.** The two most significant subgraph found when running `CALDERA` on *P. Aeruginosa*

might be small, it is still quite likely that at least one subgraph is testable. The second is that the patterns of neighbouring nodes are correlated. As such, the patterns cannot increase by as much between stages, which limits both the increase in testable pattern discovery, and the pruning.

5.5 Experiments

Speed benchmark on simulated data

Benefit of `CALDERA`'s exploration scheme We generate datasets with $n = 100$ samples represented by $p \in [100 : 200000]$ covariates, and a graph connecting these covariates, to test the speed of our algorithm. As a baseline, we include `COIN` with the improved `LAMP` algorithm of [96], which we denote `COIN+LAMP2`. For simplicity, we do not include a confounding covariate in the simulation model. The simulation framework was described in section 5.3. In addition to `COIN+LAMP2`, we benchmark 3 versions of `CALDERA`. The first one, closest to `COIN+LAMP2`, is the DFS implementation. The second one is the BFS implementation, where we modify the enumeration order of the elements of $\mathcal{C}$ to promote pruning. The last is a parallelized BFS implementation, using 5 cores.

The ranking in speed is uniform over all value of p, with `COIN+LAMP2` being by far the slowest, followed by the DFS and BFS implementation. On average, parallelizing with 5 cores offer a 3.2× speed-up compared to non-parallelized BFS version of `CALDERA`. The ratio

of runtime between CALDERA in its BFS version with 5 cores, and COIN+LAMP2 is on average 30 and is at least 8.5 (for $p = 100$). For $p = 20000$, COIN+LAMP2 takes more than a day to run while the best version of CALDERA took 40 minutes.

Benefit of CALDERA's lower-bound on runtime for imbalanced population We generate a dataset with $n = 200$ and $p = 3000$. Samples are furthermore assigned to one of two strongly imbalanced populations, such that $n_2/n_1 \ll 1$. For extreme ratios— below 0.02—the new lower bound allows much more pruning and enumerates an order of magnitude fewer elements of $\mathcal{C}$. Up to a ratio of 0.1, the new lower bound leads to a decrease of at least 10% in the number of explored subgraphs. Such an imbalance might seem extreme but is not uncommon in applications—the application in 5.5 involved three population with $n_3/n = 6\%$.

Network-guided GWAS on *A. thaliana* genomes

We now demonstrate how CALDERA performs on a medium-size GWAS dataset. We obtained over 6 millions SNPs and a "date to flowering" phenotype for $n = 936$ *A. thaliana* genomes from easyGWAS [49]. We also obtained 137 *A. thaliana* metabolic pathways from KEGG [67] using the KEGGrest [150] and DEGraph [60] R packages. The union of the pathways involved $p = 3150$ genes and the average degree of the resulting graph is ~ 21.2. We mapped each SNP to the closest gene using snpEff [32] and defined each gene to be mutated in a sample if it contained at least one mutation mapping to the gene. Runtime was under a minute using 8 cores. $k_0 = 70$ and 10 subgraphs are found to be significant. In particular, these subgraphs involve pathways ATH00260:*Glycine, serine and threonine metabolism* and 03013: *RNA transport*, which respectively contains the first and second most significant subgraphs, were previously linked to flower development [57, 111].

Bacterial GWAS

When dealing with bacterial genomes that are poorly suited to alignment and therefore to a description by SNPs, GWAS is often performed by detecting k-mers whose presence in a genome is associated with the phenotype [135, 80]. [62] further proposed to exploit the De Bruijn graph (DBG) to guide the interpretation of selected k-mers. The nodes of the DBG are k-mers, and two nodes are connected if the corresponding k-mers have an overlap of length $k - 1$. The DBG therefore provides a genomic context for every tested k-mers. However, the association is still tested at the k-mer level, and the DBG is only exploited in a postprocessing step. CALDERA makes it possible to directly test the association of a phenotype with a union of profiles of neighboring k-mers, which could typically represent a polymorphic genetic determinant. We consider the $n = 280$ *Pseudomonas Aeruginosa* genomes used in DBGWAS software Jaillard et al. [62], along with their amikacin resistance phenotype. The bacteria are partitioned based on their phylogenetic tree into three distinct

groups. The DBG is constructed using the k-mers with $k = 31$ using DBGWAS, leading to a graph with over 2.3 million nodes and average degree ~ 2.7.

The full exploration of all elements of $\mathcal{C}$ for this graph is not computationally feasible, even for CALDERA. We therefore limited our search to the first 5 stages of the tree constructed on $\mathcal{C}$. Exploring that space took approximately 5 hours to CALDERA, that identified $k_0 = 2.8 \times 10^6$ testable subgraphs for an FWER level $\alpha = 10^{-8}$. For comparison, after running for 24h, COIN+LAMP2 was exploring the tree structure with a value of $k = 10^5$. 35 of the testable subgraphs were actually significantly associated to amikacin resistance at this FWER level. We restricted ourselves to the 17 ones that were not fully included in another significant subgraph, and annotated the corresponding k-mers using blast [2] against both the NCBI database and a resistance database provided with DBGWAS. The two subgraphs with lowest p-values contained the AAC(6') gene and the pHS87b plasmid, which were the only two confirmed resistance determinants identified by DBGWAS (as its first and third hit respectively). Figure 5.3c and d show these subgraphs which are formed by a succession of small bubbles typical of polymorphic regions as described by [62]. DBGWAS identified similar graphs by testing individual k-mers (nodes) and heuristically adding their neighbors. By contrast, CALDERA allows inference on the subgraph itself—corresponding to an entire gene or plasmid–which paves the way for more powerful and principled bacterial GWAS.

5.6 Discussion

This article presented CALDERA, an algorithm to enumerate all significant closed connected subgraphs. CALDERA easily scales to large datasets, relying on an efficient structure on $\mathcal{C}$ and an exploration scheme that leverages the pruning opportunity offered by discrete statistics. Future work will focus on incorporating pre-processing schemes before CALDERA that could compact the graph to both reduce its size and facilitate pruning by increasing the average $|\mathcal{I}(v_j)|$.

5.7 Proofs

Lemma 4: correctness of the closure

Lemma 4 provides that the operator cl is well defined on connected subgraphs.

Lemma 4 *For any connected subgraph $\mathcal{S}$ of $\mathcal{G}$, there exists a unique subgraph $\mathcal{S}' \in \mathcal{C}$ such that $\mathcal{I}(\mathcal{S}) = \mathcal{I}(\mathcal{S}')$ and $\mathcal{S} \subseteq \mathcal{S}'$.*

Proof of Lemma 4 First let's show that there exists $\mathcal{S}' \in \mathcal{C}$ such that $\mathcal{I}(\mathcal{S}) = \mathcal{I}(\mathcal{S}')$ and $\mathcal{S} \subseteq \mathcal{S}'$. Let $\mathcal{S}'$ be a (inclusionwise) maximal connected subgraph containing $\mathcal{S}$ and such that $\mathcal{I}(\mathcal{S}) = \mathcal{I}(\mathcal{S}')$. By maximality of $\mathcal{S}'$, for every edge $(v_1, v_2) \in E$ with $v_1 \in \mathcal{S}'$ and $v_2 \notin \mathcal{S}'$, we have $\mathcal{I}(\mathcal{S}' \cup \{v_2\}) \neq \mathcal{I}(\mathcal{S}) = \mathcal{I}(\mathcal{S}')$, thus $\mathcal{S}' \in \mathcal{C}$.

Now let's show that such a subgraph is unique. Assume that there exits two different subgraphs $\mathcal{S}_1$ and $\mathcal{S}_2$ in $\mathcal{C}$ such that $\mathcal{S} \subseteq \mathcal{S}_1$ and $\mathcal{S} \subseteq S_2$ with $\mathcal{I}(\mathcal{S}) = \mathcal{I}(\mathcal{S}_1) = \mathcal{I}(\mathcal{S}_2)$. Since $\mathcal{S}_1 \neq \mathcal{S}_2$, at least one of the subgraphs $\mathcal{S}_1 \setminus \mathcal{S}_2$ and $\mathcal{S}_2 \setminus \mathcal{S}_1$ is not empty. Assume without loss of generality that $\mathcal{S}_1 \setminus \mathcal{S}_2 \neq \emptyset$. Since $\mathcal{S}_1$ is connected and since $\mathcal{S}_1 \cap \mathcal{S}_2 \supseteq \mathcal{S} \neq \emptyset$, there is at least one edge (u, v) with $u \in \mathcal{S}_1 \cap \mathcal{S}_2$ and $v \in \mathcal{S}_1 \setminus \mathcal{S}_2$. This leads to a contradiction since the edge (u, v) is such that $u \in \mathcal{S}_2$, $v \notin \mathcal{S}_2$ and $\mathcal{I}(\mathcal{S}_2 \cup v) = \mathcal{I}(\mathcal{S}) = \mathcal{I}(\mathcal{S}_2)$, which is in contradiction with $\mathcal{S}_2 \in \mathcal{C}$.

Lemma 1: $\mathcal{P}$ is a valid reduction

Case if $\mathcal{I}(\mathcal{S}) = \mathcal{J}(\mathcal{S})$: Then, either $\mathcal{S} = \emptyset$ which has trivially no parent by this reduction. Or all nodes of $\mathcal{S}$ contain exactly the same pattern. For any $v \in \mathcal{S}$, $\mathcal{S} = cl(v)$. $\mathcal{S}$ is a root of our exploration. Its parent is $\emptyset \subseteq \mathcal{S}$. Note that, to avoid enumerating those roots more than once, we only start from $v_{\max} = \max \mathcal{S}$.

Case if $i_{\mathcal{S}}$ is defined: Then, $i_{\mathcal{S}} \in \mathcal{I}(\mathcal{S})$ so $\mathcal{S} \bigcap \mathbb{V}_{i_{\mathcal{S}}} \neq \emptyset$ and $i_{\mathcal{S}} \notin \mathcal{I}(\mathcal{S})$ so $\mathcal{S} \setminus \mathbb{V}_i \neq \emptyset$. Therefore, there is at least one connected component in $\mathcal{S} \setminus \mathbb{V}_i$. Moreover, any connected component of $\mathcal{S} \setminus \mathbb{V}_i$ is included but not equal to $\mathcal{S}$. From [53], Lemma 1, we know that, if $\mathcal{S} \in \mathcal{C}$, any connected component of $\mathcal{S} \setminus \mathbb{V}_i$ is also in $\mathcal{C}$. So any connected component of $\mathcal{S} \setminus \mathbb{V}_i$ can be defined as a parent of $\mathcal{S}$. To identify a unique parent, we select the one with the highest node number, $\mathcal{S}_p$. This proves that reduction defines a unique parent. It is a strictly smaller subgraph by inclusion. Indeed, note that since $\mathcal{S} \setminus \mathbb{V}_i \neq \emptyset$ and $\mathcal{S}_p \subset \left(\mathcal{S} \bigcap \mathbb{V}_i\right)$, then $\mathcal{S}_p \subseteq \mathcal{S}$.

Lemma 2: conditions (C1-3) are necessary and sufficient for $\mathcal{S} = \mathcal{P}(\mathcal{S}')$

Proof that for any $\mathcal{S}'$, $(\mathcal{S} = \mathcal{P}(\mathcal{S}'), \mathcal{S}')$ verify $(C1-3)$

$\mathcal{S} \subset \mathcal{S}' \setminus \mathbb{V}_{i_{\mathcal{S}'}}$ so $i_{\mathcal{S}'} \notin \mathcal{I}(\mathcal{S})$. This proves (1). $\max\{v' \in \mathcal{S}' \setminus \mathbb{V}_{i_{\mathcal{S}'}}\} \in \mathcal{S}$ by construction of the parent so $\max\{v' \in \mathcal{S}' \setminus \mathbb{V}_{i_{\mathcal{S}'}}\} \leq \max \mathcal{S}$. Moreover, $\mathcal{S} \subset \mathcal{S}' \setminus \mathbb{V}_{i_{\mathcal{S}'}}$ so $\max \mathcal{S} \leq \max\{v' \in \mathcal{S}' \setminus \mathbb{V}_{i_{\mathcal{S}'}}\}$. So $\max\{v' \in \mathcal{S}' \setminus \mathbb{V}_{i_{\mathcal{S}'}}\} = \max \mathcal{S}$, this proves (2).

Suppose (3) is false. Then, we have $v \in Ne(\mathcal{S}) \bigcap (\mathcal{S}' \setminus \mathbb{V}_{i_{\mathcal{S}'}})$. $\mathcal{S}_2 = cl(\mathcal{S} \bigcup \{v\}) \subset \mathcal{S}'$, $\max \mathcal{S}_2 = \max\{v' \in \mathcal{S}' \setminus \mathbb{V}_{i_{\mathcal{S}'}}\}$ since $\mathcal{S} \subset \mathcal{S}_2$ and $\mathcal{S}_2 \bigcap \mathbb{V}_{i_{\mathcal{S}'}} = \emptyset$ so $\mathcal{S}_2 \subset \mathcal{P}(\mathcal{S}')$. But $\mathcal{S}2 \supsetneq \mathcal{S} = \mathcal{P}(\mathcal{S})$. This is not possible. So (3) is true.

This proves the implication in the first sense.

Proof that for any $(\mathcal{S}, \mathcal{S}')$ that verify $(C1-3)$, $\mathcal{S} = \mathcal{P}(\mathcal{S}')$

We consider two closed connected subgraph $\mathcal{S}, \mathcal{S}' \in \mathcal{C}$ that verify (1-3). We want to prove that $\mathcal{P}(\mathcal{S}') = \mathcal{S}$. Point (1) insures that $\mathcal{S} \subseteq (\mathcal{S}' \setminus \mathbb{V}_{i_{\mathcal{S}'}})$. Since $\mathcal{S} \in \mathcal{C}$ and contains the maximal node (from (2)), this ensures that $\mathcal{S} \subseteq \mathcal{P}(\mathcal{S}')$.

Suppose $\mathcal{S} \subsetneq \mathcal{P}(\mathcal{S}')$. Then, $\mathcal{P}(\mathcal{S}') \setminus \mathcal{S} \neq \emptyset$. In particular, since $\mathcal{S}$ and $\mathcal{P}(\mathcal{S}')$ are both connected subgraphs, there exists $v' \in \big(\mathcal{P}(\mathcal{S}') \setminus \mathcal{S}\big) \bigcap Ne(\mathcal{S})$. Since this neighbour is in $\mathcal{P}(\mathcal{S}')$, it is also in $\mathcal{S}' \setminus \mathbb{V}_{i_{\mathcal{S}'}}\}$. That is impossible from (3). So $\mathcal{S} = \mathcal{P}(\mathcal{S}')$. (Note that point (3) includes the fact that $i_{\mathcal{S}'} \in \mathcal{I}(v)$).

This proves the converse implication.

Theorem 1: Algorithm 1 correctly inverts the reduction

We consider a subgraph $\mathcal{S}' \in \mathcal{S}$ and its parent $\mathcal{S} = \mathcal{P}(\mathcal{S}')$.

We first show two lemmas

Lemma 5 *For two subgraphs $\mathcal{S}_1, \mathcal{S}_2 \in \mathcal{C}$, if $\mathcal{S}_1 \subset \mathcal{S}_2$, then $i_{\mathcal{S}_1} \leq i_{\mathcal{S}_2}$.*

Proof:

$$\mathcal{S}_1 \subset \mathcal{S}_2 \implies \mathcal{I}(\mathcal{S}_1) \subset \mathcal{I}(\mathcal{S}_2) \quad \text{and} \tag{5.1}$$

$$\mathcal{S}_1 \subset \mathcal{S}_2 \implies \mathcal{J}(\mathcal{S}_2) \subset \mathcal{J}(\mathcal{S}_1) \tag{5.2}$$

$$(1) \text{ and } (2) \implies (\mathcal{I}(\mathcal{S}_1) \setminus \mathcal{J}(\mathcal{S}_1)) \subset (\mathcal{I}(\mathcal{S}_2) \setminus \mathcal{J}(\mathcal{S}_2)) \tag{5.3}$$

$$\implies i_{\mathcal{S}_1} \leq i_{\mathcal{S}_2} \tag{5.4}$$

Lemma 6 *For a subgraph $\mathcal{S}' \in \mathcal{C}$ such that $\mathcal{S} = \mathcal{P}(\mathcal{S}') \neq \emptyset$, any subgraph $\mathcal{S}_2 \in \mathcal{C}$ that verifies:*

- $\mathcal{S} \subsetneq \mathcal{S}_2$
- $\mathcal{S}_2 \subset \mathcal{S}'$

is a child of $\mathcal{S}$, that is $\mathcal{P}(\mathcal{S}_2) = \mathcal{S}$

Proof: We know that $\mathcal{S} \subsetneq \mathcal{S}_2$ so $Ne(\mathcal{S}) \bigcap \mathcal{S}_2 \neq \emptyset$. Since $\mathcal{S}_2 \subset \mathcal{S}'$, $Ne(\mathcal{S}) \bigcap \mathcal{S}_2 \subset \mathcal{S}'$ so, from (3) for $\mathcal{S}, \mathcal{S}'$, we have $Ne(\mathcal{S}) \bigcap \mathcal{S}_2 \subset \mathbb{V}_{i_{\mathcal{S}'}}$. So $i_{\mathcal{S}'} \in \mathcal{I}(\mathcal{S}_2)$. $i_{\mathcal{S}'} \notin \mathcal{I}(\mathcal{S})$ so $i_{\mathcal{S}'} \notin \mathcal{J}(\mathcal{S}_2)$. Therefore, $i_{\mathcal{S}'} \leq i_{\mathcal{S}_2}$. But since $\mathcal{S}_2 \subset \mathcal{S}'$, $i_{\mathcal{S}'} \geq i_{\mathcal{S}_2}$. So $i_{\mathcal{S}'} = i_{\mathcal{S}_2}$. Then, we know that $\mathcal{S}, \mathcal{S}_2$ verifies (1). Since $\mathcal{S} \subsetneq \mathcal{S}_2$, we also have (2). Finally $\{v' \in \mathcal{S}_2 \setminus \mathbb{V}_{i_{\mathcal{S}_2}} : v' \in Ne(\mathcal{S})\} = \{v' \in \mathcal{S}_2 \setminus \mathbb{V}_{i_{\mathcal{S}'}} : v' \in Ne(\mathcal{S})\} \subset \{v' \in \mathcal{S}' \setminus \mathbb{V}_{i_{\mathcal{S}'}} : v' \in Ne(\mathcal{S})\} = \emptyset$. This proves (3). Since we have (1-3), we know that $\mathcal{P}(\mathcal{S}_2) = \mathcal{S}$.

Main proof: Now let us prove the main result: Assume that we cannot generate $\mathcal{S}'$ with the procedure from algorithm 1. Let's then consider the largest $\mathcal{S}'' \subsetneq \mathcal{S}'$ generated with the algorithm 1, that is the one with the largest number of nodes. Since $\mathcal{S} = \mathcal{P}(\mathcal{S}')$, we at least have $\mathcal{S}_d \subset \mathcal{S}'$ so at minimum we can take $\mathcal{S}'' = \mathcal{S}_d$.

By assumption, $\mathcal{S}'' \subsetneq \mathcal{S}'$. Therefore, there exists a neighbour $v \in Ne(\mathcal{S}'') \bigcap \mathcal{S}'$ since $\mathcal{S}'$ and $\mathcal{S}''$ are connected subgraphs. Note that we know that $i_{\mathcal{S}''} = i_{\mathcal{S}'}$ by construction. Moreover, $(\mathcal{S}, \mathcal{S}_2 = cl(\mathcal{S}'' \bigcup \{v\}))$ verify (1-3) by Lemma 2.

Case 1: $\mathcal{I}(cl(\mathcal{S}'' \bigcup \{v\}))$ **does not include any pattern of** `forbidden` $(\mathcal{S}'')$**:** Since $\mathcal{S}_2 \subset \mathcal{S}'$, $i_{\mathcal{S}_2} \leq i_{\mathcal{S}'}$. However, since $\mathcal{S}'' \subset \mathcal{S}_2$, $i_{\mathcal{S}_2} \geq i_{\mathcal{S}''} = i_{\mathcal{S}'}$. So $i_{\mathcal{S}_2} = i_{\mathcal{S}'}$. Moreover, we already know that $(\mathcal{S}, \mathcal{S}_2)$ verify (1-3). That means we can create $\mathcal{S}_2 \supsetneq \mathcal{S}''$ which contradicts our assumption that $\mathcal{S}''$ is the largest closed subgraph strictly included in $\mathcal{S}'$ that could be generated.

Case 2: $\mathcal{I}(cl(\mathcal{S}'' \bigcup \{v\}))$ **includes one of the pattern of** `forbidden` $(\mathcal{S}'')$**:** We note $v_1, \ldots, v_l$ the sequence that created $\mathcal{S}''$ from $\mathcal{S}_d$. At one point in that process, we added a pattern to `forbidden`$(\mathcal{S}'')$ that is now contained in $\mathcal{I}(cl(\mathcal{S}'' \bigcup \{v\}))$, let's say when adding v_k. This pattern was linked to another equivalence group. If we consider v' a node from that group, we will then construct a subgraph using the sequence $v_1, \ldots, v_{k-1}, v', v_k, \ldots, ...v_l$. Note that since at each new addition, the constructed graph is included in $\mathcal{S}_2$, it's also included in $\mathcal{S}'$. Moreover, each one contains $\mathcal{S}$ and a node from $\mathbb{V}_{i_{\mathcal{S}'}}$ by construction (since it contains $\mathcal{S}_d$). So, using Lemma 2, we know that those additions all respect (1-3), i.e they are valid additions according to our algorithm. This way, we can create a subgraph that contains $\mathcal{S}''$ and v' with our procedure. This contradicts our assumption that $\mathcal{S}''$ is the largest closed subgraph strictly included in $\mathcal{S}'$ that could be generated.

This proves that all $\mathcal{S}'$ will be generated from $\mathcal{S}$ and therefore that we have properly inverted the reduction

Proof of Lemma 3

The next two lemmas are directly taken from [105].

Minimal p-value of the CMH test

Lemma 7 (Papaxanthos et al. [105]) *The minimal p-value of the CMH test can be computed in $O(J)$.*

Proof The p-value associated with the CHM test, conditioning on the margins of all the tables, is:

$$\begin{aligned} p_{CMH}(\mathcal{S}, \mathbf{Y}, \mathbf{C}) =& 1 - \mathbf{F}_{\chi^2_1}\Big(\frac{\big(\sum_{j=1}^{J} a_{\mathcal{S},j} - \frac{x_{\mathcal{S},j} n_{1,j}}{n_j}\big)^2}{\sum_{j=1}^{J} \frac{n_{1,j}}{n_j - 1} \frac{n_{2,j}}{n_j} x_{\mathcal{S},j} (1 - \frac{x_{\mathcal{S},j}}{n_j})}\Big) \\ =& 1 - \mathbf{F}_{\chi^2_1}\Big(\frac{\big(a_{\mathcal{S}} - \sum_{j=1}^{J} \frac{x_{\mathcal{S},j} n_{1,j}}{n_j}\big)^2}{\sum_{j=1}^{J} \frac{n_{1,j}}{n_j - 1} \frac{n_{2,j}}{n_j} x_{\mathcal{S},j} (1 - \frac{x_{\mathcal{S},j}}{n_j})}\Big) \\ =& 1 - \mathbf{F}_{\chi^2_1}\Big(T_{\mathcal{S}}(a_{\mathcal{S}}, x_{\mathcal{S}})\Big) \end{aligned}$$

Since $\mathbf{F}_{\chi^2_1}$ is monotonically increasing, the minimal p-value is obtained for the smallest value $T_{\mathcal{S}}(a_{\mathcal{S}}, x_{\mathcal{S}}$. This is a function of $a_{\mathcal{S}}$ that is quadratic with a positive definite hessian[105] so the function is maximal for $\min a_{\mathcal{S}}$ or $\max a_{\mathcal{S}}$. We have that $a_{\mathcal{S},j,\min} = 0$ if $x_{\mathcal{S},j} \leq n_{2,j}$ and $a_{\mathcal{S},j,\min} = x_{\mathcal{S},j} - n_{2,j}$; and $a_{\mathcal{S},j,\max} = x_{\mathcal{S},j}$ for $x_{\mathcal{S},j} \leq n_{1,j}$ and $a_{\mathcal{S},j,\max} = n_{1,j}$ otherwise. So, $T^{\max}_{\mathcal{S}}(x_{\mathcal{S}}) = \max(T^l_{\mathcal{S}}, T^r_{\mathcal{S}})$ where

$$T^l_{\mathcal{S}} = \frac{\left(\sum_{j=1}^J a_{\mathcal{S},j,\min} - \frac{x_{\mathcal{S},j} n_{1,j}}{n_j}\right)^2}{\sum_{j=1}^J \frac{n_{1,j}}{n_j - 1} \frac{n_{2,j}}{n_j} x_{\mathcal{S},j}(1 - \frac{x_{\mathcal{S},j}}{n_j})}$$
$$T^r_{\mathcal{S}} = \frac{\left(\sum_{j=1}^J a_{\mathcal{S},j,\max} - \frac{x_{\mathcal{S},j} n_{1,j}}{n_j}\right)^2}{\sum_{j=1}^J \frac{n_{1,j}}{n_j - 1} \frac{n_{2,j}}{n_j} x_{\mathcal{S},j}(1 - \frac{x_{\mathcal{S},j}}{n_j})}$$

Computing the envelope

Definition 5 *For each $\mathcal{S} \in \mathcal{C}$, the envelope of $\mathcal{S}$ is defined as $\tilde{p}^\star(\mathcal{S}) \equiv \min_{\mathcal{S}'': x_{\mathcal{S}''} \geq x_{\mathcal{S}}} p^*(\mathcal{S}'')$.*

Lemma 8 (Papaxanthos et al. [105]) *If a subgraph is prunable, i.e $\tilde{p}^\star(\mathcal{S}) > \alpha/k$, then any subgraph $\mathcal{S}' \supset \mathcal{S}$ is also prunable*

Proof :

$$\begin{aligned} \mathcal{S}' \supset \mathcal{S} &\implies \{\mathcal{S}'' : x_{\mathcal{S}''} \geq x_{\mathcal{S}}\} \supset \{\mathcal{S}'' : x_{\mathcal{S}''} \geq x'_{\mathcal{S}}\} \\ &\implies \min_{\mathcal{S}'': x_{\mathcal{S}''} \geq x_{\mathcal{S}}} p^*(\mathcal{S}'') \leq \min_{\mathcal{S}'': x_{\mathcal{S}''} \geq x'_{\mathcal{S}}} p^*(\mathcal{S}'') \\ \tilde{p}^\star(\mathcal{S}) > \alpha/k \text{ and above} &\implies \tilde{p}^\star(\mathcal{S}') > \alpha/k \end{aligned}$$

Proof of Lemma 3 We consider the function

$$T_l(x'_{\mathcal{S}}) = \frac{\left(a_{\mathcal{S}',\min} - \sum_{j=1}^J \frac{x_{\mathcal{S}',j} n_{1,j}}{n_j}\right)\right)^2}{\sum_{j=1}^J \frac{n_{1,j}}{n_j - 1} \frac{n_{2,j}}{n_j} x_{\mathcal{S}',j}(1 - \frac{x_{\mathcal{S}',j}}{n_j})}$$

with $x_{\mathcal{S},j} \leq x_{\mathcal{S}',j} \leq n_j$ and we will look at the partial derivatives. We add a few notations:

$$
\begin{aligned}
\mu =& \frac{n_{1,j}}{n_j - 1}\frac{n_{2,j}}{n_j} \\
K =& \sum_{j=1}^{J} \frac{n_{1,j}}{n_j - 1}\frac{n_{2,j}}{n_j} x_{\mathcal{S}',j}(1 - \frac{x_{\mathcal{S}',j}}{n_j}) \\
K_{-i} =& \sum_{j=1, j\neq i}^{J} \frac{n_{1,j}}{n_j - 1}\frac{n_{2,j}}{n_j} x_{\mathcal{S}',j}(1 - \frac{x_{\mathcal{S}',j}}{n_j}) \\
S =& \sum_{j=1}^{J} \frac{x_{\mathcal{S}',j} n_{1,j}}{n_j} - a_{\mathcal{S},j,\min} \\
S_{-i} =& \sum_{j=1, j\neq i}^{J} \frac{x_{\mathcal{S}',j} n_{1,j}}{n_j} - a_{\mathcal{S},j,\min} \\
a'_{\mathcal{S},\min} =& \frac{\partial a_{\mathcal{S},\min}}{\partial x_{\mathcal{S}',i}}
\end{aligned}
$$

We then have

$$
\begin{aligned}
\frac{\partial T_l(x_{\mathcal{S}'})}{\partial x_{\mathcal{S}',i}} =& N_l(x_{\mathcal{S}'}) \times D_l(x_{\mathcal{S}'}) && \text{where} \\
N_l(x_{\mathcal{S}'}) =& 2(\frac{n_{1,i}}{n_i} - a'_{\mathcal{S},\min})K - (1 - 2\frac{x_{\mathcal{S}',i}}{n_i})\mu S && \text{and} \\
D_l(x_{\mathcal{S}'}) =& \frac{S}{K^2}
\end{aligned}
$$

For all j, $\frac{x_{\mathcal{S}',j} n_{1,j}}{n_j} - a_{\mathcal{S},j,\min} \geq \frac{x_{\mathcal{S}',j} n_{1,j}}{n_j} - x_{\mathcal{S}',j} + n_{2,j} = n_{2,j}(1 - \frac{x_{\mathcal{S}',j}}{n_j}) \geq 0$ so $D_l(x_{\mathcal{S}'}) \geq 0$. We only need look at $N_l(x_{\mathcal{S}'})$ to find maxima. We can also note that consequently, $S \geq 0$ and $S_{-i} \geq 0$.

$$
\begin{aligned}
N_l(x_{\mathcal{S}'}) =& 2(\frac{n_{1,i}}{n_j} - a'_{\mathcal{S},\min})K - (1 - 2\frac{x_{\mathcal{S}',i}}{n_j})\mu S \\
=& 2\frac{n_{1,i}}{n_i}K_{-i} + 2\frac{n_{1,i}}{n_i}\mu x_{\mathcal{S}',i}(1 - \frac{x_{\mathcal{S}',i}}{n_i}) - 2a'_{\mathcal{S},\min}K_{-i} - 2a'_{\mathcal{S},\min}\mu x_{\mathcal{S}',i}(1 - \frac{x_{\mathcal{S}',i}}{n_i}) - \\
& \mu S_{-i} - \mu(\frac{x_{\mathcal{S}',i} n_{1,i}}{n_i} - a_{\mathcal{S},i,\min}) + 2\frac{x_{\mathcal{S}',i}}{n_i}\mu S_{-i} + 2\frac{x_{\mathcal{S}',i}}{n_i}\mu(\frac{x_{\mathcal{S}',i} x n_{1,i}}{n_i} - a_{\mathcal{S},i,\min}) \\
=& \big[2\frac{n_{1,i}}{n_i}K_{-i} - 2a'_{\mathcal{S},\min}K_{-i} - \mu S_{-i}\big] + x_{\mathcal{S}',i}\big[2\mu\frac{n_{1,i}}{n_i} - 2a'_{\mathcal{S},\min}\mu - \mu\frac{n_{1,i}}{n_i} + 2\frac{\mu}{n_i}S_{-i}\big] + \\
& - 2\frac{n_{1,i}}{n_i^2}\mu x_{\mathcal{S}',i}^2 + 2\frac{n_{1,i}}{n_i^2}\mu x_{\mathcal{S}',i}^2 + 2a'_{\mathcal{S},\min}\mu\frac{x_{\mathcal{S}',i}^2}{n_i} - 2a_{\mathcal{S},i,\min}\mu\frac{x_{\mathcal{S}',i}}{n_i} + \mu a_{\mathcal{S},\min}
\end{aligned}
$$

We then have two cases:

$x_{\mathcal{S}',i} \leq n_{2,i}$**:** Then, $a_{\mathcal{S},i,\min} = 0$ and $a'_{\mathcal{S},\min} = 0$. So

$$N_l(x_{\mathcal{S}'}) = \big[2\frac{n_{1,i}}{n_i}K_{-i} - \mu S_{-i}\big] + x_{\mathcal{S}',i}\big[\mu\frac{n_{1,i}}{n_i} + 2\frac{\mu}{n_i}S_{-i}\big]$$

It is an affine function with a positive slope. Moreover, at the boundary at $n_{2,i}$, the function is positive. So the function is maximal at $n_{2,i}$.

$x_{\mathcal{S}',i} \geq n_{2,i}$**:** Then, $a_{\mathcal{S},i,\min} = x_{\mathcal{S}',i} - n_{2,i}$ and $a'_{\mathcal{S},\min} = 1$. So

$$\begin{aligned} N_l(x_{\mathcal{S}'}) =& \big[2\frac{n_{1,i}}{n_i}K_{-i} - 2K_{-i} - \mu S_{-i} - \mu n_{2,i}\big] + x_{\mathcal{S}',i}\big[\mu\frac{n_{1,i}}{n_i} - 2\mu + \mu + 2\mu\frac{n_{1,i}}{n_i} + 2\frac{\mu}{n_i}S_{-i}\big] + \\ & 2\mu\frac{x^2_{\mathcal{S}',i}}{n_i} - 2\mu\frac{x^2_{\mathcal{S}',i}}{n_i} \\ =& \big[2\frac{n_{1,i}}{n_i}K_{-i} - 2K_{-i} - \mu S_{-i}\big] + x_{\mathcal{S}',i}\big[\mu\frac{n_{2,i}}{n_i} + 2\frac{\mu}{n_i}S_{-i}\big] \end{aligned}$$

So $N_l(x_{\mathcal{S}'})$ is an affine by-piece function of $x_{\mathcal{S}',i}$, whose slope $A_l(x_{\mathcal{S}',-i}) \geq 0$. So, the only possible maxima are at the boundary, where $x_{\mathcal{S}',i} = n_{2,i}$ or $x_{\mathcal{S}',i} = n_i$. Since this is true for all values of $x_{\mathcal{S}',-i}$, we know that we can only achieve a maximum for T_l at the boundaries. So the only two possible maxima are $n_{2,i}$ and n_i. Note that, in the case where $x_{\mathcal{S}',i} \geq n_{2,i}$, then the possible maxima becomes $x_{\mathcal{S}',i}$ and n_i so in general, the two possible maxima for $T_l(x'_{\mathcal{S}})$ are $\max\{n_{2,i}, x_{\mathcal{S}',i}\}$ and n_i.

The same proof holds for T_r (given the symmetry of the expressions), where the maxima is in $\{\max(x_{\mathcal{S},i}, n_{1,i}), n_i\}$. This proves the lemma.We have reduced the space of possibilities from $O(m^J)$ to $O(2^J)$ (with m the geometric mean of $x_{\mathcal{S}'}$).

We now need to show how to compute this value in $O(J\log(J))$. For this, we rely on the following theorem.

Lemma 9 ([105]) *Let $\mathcal{S}$ be a potentially testable subgraph and define $\beta^l_{\mathcal{S}',1} = \frac{n_{1,j}x_{\mathcal{S}',j}}{n_j^2}$ and $\beta^l_{\mathcal{S}',1} = \frac{n_{2,j}x_{\mathcal{S}',j}}{n_j^2}$, for $j \in \{1,\dots,J\}$ Let π_l and π_r be permutations of $\{1,\dots,J\}$ such that $\beta^l_{\mathcal{S}',\pi_l(1)} \leq \dots \leq \beta^l_{\mathcal{S}',\pi_l(J)}$ and $\beta^r_{\mathcal{S}',\pi_r(1)} \leq \dots \leq \beta^r_{\mathcal{S}',\pi_r(J)}$, respectively.*
Then, there exist $\kappa \in \{1,\dots,J\}$ such that the optimum $x^*_{\mathcal{S}'}$ satisfies either

- $x^*_{\mathcal{S}',\pi_l(j)} = x_{\mathcal{S},\pi_l(j)}$ *for $j \leq \kappa$ and $x^*_{\mathcal{S}',\pi_l(j)} = n_j$ otherwise*

or

- $x^*_{\mathcal{S}',\pi_r(j)} = x_{\mathcal{S},\pi_r(j)}$ *for $j \leq \kappa$ and $x^*_{\mathcal{S}',\pi_r(j)} = n_j$ otherwise*

Proof of lemma 9 We will do the proof for $T_l(x'_{\mathcal{S}})$. The proof for $T_r(x'_{\mathcal{S}})$ is identical, up to notation.

We can note that since the optimal is at least equal to $n_{2,j}$, the value of $a_{\mathcal{S},j,\min}$ is $x_{\mathcal{S}',j} - n_{2,j}$. We note $\beta_j = \frac{n_{1,j}x^*_{\mathcal{S}',j}}{n_j^2}$. We also note $l(\beta_j) = n_{1,j}n_j(1 - \frac{n_j\beta_j}{n_{1,j}}) = n_{2,j}(1 - \frac{x^*_{\mathcal{S}',j}}{n_j})$. With those notations, we can write

$$T_l(x'_{\mathcal{S}}) = \frac{\sum_{j=1}^{J} l(\beta_j)}{\sum_{j=1}^{J} \beta_j l(\beta_j)}$$

It is straightforward to see that, if $x^*_{\mathcal{S}',j} = n_j$, then $l(\beta_j) = 0$. We are then exactly in the setting of Papaxanthos et al. [105] and we refer the reader to the proof in the supplementary, p.3-6.

This shows that we can compute the envelope in $J \log(J)$

Bibliography

[1] Ayelet Alpert et al. "Alignment of single-cell trajectories to compare cellular expression dynamics". In: *Nature Methods* 15.4 (Mar. 2018), pp. 267–270. ISSN: 1548-7091. DOI: 10.1038/nmeth.4628. URL: http://www.ncbi.nlm.nih.gov/pubmed/29529018%20http://www.nature.com/doifinder/10.1038/nmeth.4628.

[2] S. Altschul et al. "Basic local alignment search tool". In: *Journal of Molecular Biology* 215 (1990), pp. 403–410.

[3] David Avis and Komei Fukuda. "Reverse Search for Enumeration". In: *Discrete Applied Mathematics* 65 (1993), pp. 21–46.

[4] Chloé-Agathe Azencott et al. "Efficient network-guided multi-locus association mapping with graph cuts". In: *Bioinformatics* 29.13 (June 2013), pp. i171–i179. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btt238. eprint: https://academic.oup.com/bioinformatics/article-pdf/29/13/i171/18536066/btt238.pdf. URL: https://doi.org/10.1093/bioinformatics/btt238.

[5] Rhonda Bacher et al. "Trendy: segmented regression analysis of expression dynamics in high-throughput ordered profiling experiments". In: *BMC Bioinformatics* 19.1 (Dec. 2018), p. 380. ISSN: 1471-2105. DOI: 10.1186/s12859-018-2405-x. URL: https://bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-018-2405-x.

[6] Ali Bagherinia et al. "Elite fuzzy clustering ensemble based on clustering diversity and quality measures". In: *Applied Intelligence* 49.5 (May 2019), pp. 1724–1747. ISSN: 15737497. DOI: 10.1007/s10489-018-1332-x. URL: https://doi.org/10.1007/s10489-018-1332-x.

[7] Armando Sosthene Kali Balogoun, Guy Martia Nkiet, and Carlos Ogouyandjou. "k-Sample problem based on generalized maximum mean discrepancy". In: *arxiv* (Nov. 2018). arXiv: 1811.09103. URL: http://arxiv.org/abs/1811.09103.

[8] J. Baran-Gale et al. "Ageing compromises mouse thymus function and remodels epithelial cell differentiation". In: *eLife* 9 (Aug. 2020), pp. 1–71. ISSN: 2050084X. DOI: 10.7554/ELIFE.56221.

[9] W. Brad Barbazuk et al. "SNP discovery via 454 transcriptome sequencing". In: *The Plant Journal* 51.5 (June 2007), pp. 910–918. ISSN: 09607412. DOI: 10.1111/j.1365-313X.2007.03193.x. URL: http://doi.wiley.com/10.1111/j.1365-313X.2007.03193.x.

[10] Maayan Baron et al. "A Single-Cell Transcriptomic Map of the Human and Mouse Pancreas Reveals Inter- and Intra-cell Population Structure". In: *Cell Systems* 3.4 (Oct. 2016), 346–360.e4. ISSN: 24054720. DOI: 10.1016/j.cels.2016.08.011. URL: https://www.sciencedirect.com/science/article/pii/S2405471216302666?via%7B%5C%%7D3Dihub.

[11] Etienne Becht et al. "Dimensionality reduction for visualizing single-cell data using UMAP". In: *Nature Biotechnology* 37.1 (Jan. 2019), pp. 38–44. ISSN: 1087-0156. DOI: 10.1038/nbt.4314. URL: http://www.nature.com/articles/nbt.4314.

[12] E. T. Bell. "The Iterated Exponential Integers". In: *The Annals of Mathematics* 39.3 (July 1938), p. 539. ISSN: 0003486X. DOI: 10.2307/1968633.

[13] Yosef Benjamini, Yoav ; Hochberg. "Controlling the False Discovery Rate - a Practical and Powerful Approach to Multiple Testing. Journal of the Royal Statistical Society Series B-Methodological 1995.pdf". In: *Journal of the Royal Statistical Society Series B (Methological)* 57.1 (1995), pp. 289–300. DOI: 10.2307/2346101. arXiv: 95/57289 [0035-9246]. URL: https://www.jstor.org/stable/2346101%20http://www.jstor.org/stable/2346101.

[14] Jessica M.A. Blair et al. *Molecular mechanisms of antibiotic resistance*. Jan. 2015. DOI: 10.1038/nrmicro3380. URL: www.nature.com/reviews/micro.

[15] Vincent D Blondel et al. "Fast unfolding of communities in large networks". In: *Journal of Statistical Mechanics: Theory and Experiment* 2008.10 (Oct. 2008), P10008. ISSN: 17425468. DOI: 10.1088/1742-5468/2008/10/P10008. URL: http://stacks.iop.org/1742-5468/2008/i=10/a=P10008?key=crossref.46968f6ec61eb8f907a760be1c5ace52

[16] B M Bolstad et al. "A comparison of normalization methods for high density oligonucleotide array data based on variance and bias." In: *Bioinformatics (Oxford, England)* 19.2 (Jan. 2003), pp. 185–93. ISSN: 1367-4803. URL: http://www.ncbi.nlm.nih.gov/pubmed/12538238.

[17] CE Bonferroni. "Teoria Statistica Delle Classi e Calcolo Delle Probabilità". In: *Pubblicazioni del R Istituto Superiore di Scienze Economiche e Commerciali di Firenze* 8 (1936), pp. 3–62. DOI: 10.4135/9781412961288.n455.

[18] Alexis Boukouvalas, James Hensman, and Magnus Rattray. "BGP: identifying gene-specific branching dynamics from single-cell data with a branching Gaussian process". In: *Genome Biology* 19.1 (Dec. 2018), p. 65. ISSN: 1474-760X. DOI: 10.1186/s13059-018-1440-2. URL: https://genomebiology.biomedcentral.com/articles/10.1186/s13059-018-1440-2.

[19] Nicolas L. Bray et al. "Near-optimal probabilistic RNA-seq quantification". In: *Nature Biotechnology* 34.5 (May 2016), pp. 525–527. ISSN: 15461696. DOI: 10.1038/nbt.3519.

[20] Leo Breiman. "Random forests". In: *Machine Learning* 45.1 (Oct. 2001), pp. 5–32. ISSN: 08856125. DOI: 10.1023/A:1010933404324. URL: https://link.springer.com/article/10.1023/A:1010933404324.

[21] James H Bullard et al. "Evaluation of statistical methods for normalization and differential expression in {mRNA-Seq} experiments". In: *BMC Bioinformatics* 11 (Feb. 2010), p. 94.

[22] Daniel B Burkhardt et al. *Quantifying the effect of experimental perturbations in single-cell RNA-sequencing data using graph signal processing*. 2019. DOI: 10.1101/532846. URL: http://dx.doi.org/10.1101/532846.

[23] Lauren E. Byrnes et al. "Lineage dynamics of murine pancreatic development at single-cell resolution". In: *Nature Communications* 9.1 (Dec. 2018), p. 3922. ISSN: 2041-1723. DOI: 10.1038/s41467-018-06176-3. URL: http://www.nature.com/articles/s41467-018-06176-3.

[24] Robrecht Cannoodt, Wouter Saelens, and Yvan Saeys. "Computational methods for trajectory inference from single-cell transcriptomics". In: *European Journal of Immunology* 46.11 (Nov. 2016), pp. 2496–2506. ISSN: 00142980. DOI: 10.1002/eji.201646347. URL: http://doi.wiley.com/10.1002/eji.201646347.

[25] Robrecht Cannoodt et al. "Dyngen: A multi-modal simulator for spearheading new single-cell omics analyses". In: *bioRxiv* (Feb. 2020), p. 2020.02.06.936971. DOI: 10.1101/2020.02.06.936971. URL: https://doi.org/10.1101/2020.02.06.936971.

[26] Junyue Cao et al. "A human cell atlas of fetal gene expression". In: *Science* 370.6518 (Nov. 2020). ISSN: 10959203. DOI: 10.1126/science.aba7721. URL: https://science.sciencemag.org/content/370/6518/eaba7721.abstract.

[27] Junyue Cao et al. "The single-cell transcriptional landscape of mammalian organogenesis". In: *Nature* 566.7745 (Feb. 2019), pp. 496–502. ISSN: 0028-0836. DOI: 10.1038/s41586-019-0969-x. URL: http://www.nature.com/articles/s41586-019-0969-x.

[28] Michelle Chang, Lin He, and Lei Cai. "An overview of genome-wide association studies". In: *Methods in Molecular Biology*. Vol. 1754. Humana Press Inc., 2018, pp. 97–108. DOI: 10.1007/978-1-4939-7717-8_6. URL: https://doi.org/10.1007/978-1-4939-7717-8%7B%5C_%7D6,.

[29] Konika Chawla et al. "TFcheckpoint: a curated compendium of specific DNA-binding RNA polymerase II transcription factors". In: *Bioinformatics* 29.19 (Oct. 2013), pp. 2519–2520. ISSN: 1460-2059. DOI: 10.1093/bioinformatics/btt432. URL: https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btt432.

[30] Huidong Chen et al. "Single-cell trajectories reconstruction, exploration and mapping of omics data with STREAM". In: *Nature Communications* 10.1 (Dec. 2019), p. 1903. ISSN: 2041-1723. DOI: 10.1038/s41467-019-09670-4. URL: http://www.nature.com/articles/s41467-019-09670-4.

[31] Peter E. Chen and B. Jesse Shapiro. *The advent of genome-wide association studies for bacteria*. June 2015. DOI: 10.1016/j.mib.2015.03.002.

[32] P. Cingolani et al. "A program for annotating and predicting the effects of single nucleotide polymorphisms, SnpEff: SNPs in the genome of Drosophila melanogaster strain w1118; iso-2; iso-3". In: *Fly* 6.2 (2012), pp. 80–92.

[33] Megan Crow et al. "Characterizing the replicability of cell types defined by single cell RNA-sequencing data using MetaNeighbor". In: *Nature Communications* 9.1 (Dec. 2018), p. 884. ISSN: 2041-1723. DOI: 10.1038/s41467-018-03282-0. URL: http://www.nature.com/articles/s41467-018-03282-0.

[34] Emma Dann et al. "Milo: differential abundance testing on single-cell data using k-NN graphs 2". In: *bioRxiv* (Nov. 2020), p. 2020.11.23.393769. DOI: 10.1101/2020.11.23.393769. URL: https://doi.org/10.1101/2020.11.23.393769.

[35] Nicolaas Govert De Bruijn. "A combinatorial problem". In: *Proc. Koninklijke Nederlandse Academie van Wetenschappen*. Vol. 49. 1946, pp. 758–764.

[36] Qiaolin Deng et al. "Single-cell RNA-seq reveals dynamic, random monoallelic gene expression in mammalian cells". In: *Science* (2014). DOI: 10.1126/science.1245316.

[37] Roland L Dobrushin. "Prescribing a system of random variables by conditional distributions". In: *Theory of Probability & Its Applications* 15.3 (1970), pp. 458–486.

[38] Sergii Domanskyi et al. "Polled Digital Cell Sorter (p-DCS): Automatic identification of hematological cell types from single cell RNA-sequencing clusters". In: *BMC Bioinformatics* 20.1 (Dec. 2019), p. 369. ISSN: 14712105. DOI: 10.1186/s12859-019-2951-x. URL: https://bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-019-2951-x.

[39] Angelo Duò, Mark D. Robinson, and Charlotte Soneson. "A systematic performance evaluation of clustering methods for single-cell RNA-seq data". In: *F1000Research* 7 (2018), pp. 377–382. ISSN: 1759796X. DOI: 10.5256/f1000research.17093.r36544. URL: https://f1000researchdata.s3.amazonaws.com/manuscripts/17687/49bacf17-03b0-4f80-bde4-e892c8c3e22f%7B%5C_%7D15666%7B%5C_%7D-%7B%5C_%7Dcharlotte%7B%5C_%7Dsoneson%7B%5C_%7Dv2.pdf?doi=10.12688/f1000research.15666.2%7B%5C&%7DnumberOfBrowsableCollections=17%7B%5C&%7DnumberOfBrowsableInstitutionalCollections=4%7B%5C&%7DnumberOfBrows.

[40] Sarah G. Earle et al. "Identifying lineage effects when controlling for population structure improves power in bacterial association studies". In: *Nature Microbiology* 1.5 (2016), pp. 1–8. ISSN: 20585276. DOI: 10.1038/nmicrobiol.2016.41. arXiv: 1510.06863. URL: http://dx.doi.org/10.1038/nmicrobiol.2016.41.

[41] David S Fischer, Fabian J Theis, and Nir Yosef. "Impulse model-based differential expression analysis of time course sequencing data". In: *Nucleic Acids Research* 46.20 (Aug. 2018), e119–e119. ISSN: 0305-1048. DOI: 10.1093/nar/gky675. URL: https://academic.oup.com/nar/advance-article/doi/10.1093/nar/gky675/5068248.

[42] RA Fisher. *Statistical methods for research workers*. Edinburgh: Oliver & Boyd, 1925. URL: https://scholar.google.com/scholar_lookup?title=Statistical%20methods%20for%20research%20workers&author=RA.%20Fisher&publication_year=1932.

[43] Ronald A Fisher. "On the Interpretation of χ 2 from Contingency Tables, and the Calculation of P". In: *Journal of the Royal Statistical Society* 85.1 (1922), pp. 87–94. ISSN: 09528385. DOI: 10.2307/2340521.

[44] Russell B. Fletcher et al. "Deconstructing Olfactory Stem Cell Trajectories at Single-Cell Resolution". In: *Cell Stem Cell* 20.6 (June 2017), pp. 817–830. ISSN: 19345909. DOI: 10.1016/j.stem.2017.04.003. URL: http://www.ncbi.nlm.nih.gov/pubmed/28506465%20http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC5484588%20https://linkinghub.elsevier.com/retrieve/pii/S1934590917301273.

[45] Saskia Freytag et al. "Comparison of clustering tools in R for medium-sized 10x genomics single-cell RNA-sequencing data". In: *F1000Research* 7 (2018). ISSN: 1759796X. DOI: 10.12688/f1000research.15809.1.

[46] Levi Gadye et al. "Injury Activates Transient Olfactory Stem Cell States with Diverse Lineage Capacities". In: *Cell Stem Cell* 21.6 (Dec. 2017), 775–790.e9. ISSN: 18759777. DOI: 10.1016/j.stem.2017.10.014.

[47] Carolyn A. de Graaf et al. "Haemopedia: An Expression Atlas of Murine Hematopoietic Cells". In: *Stem Cell Reports* 7.3 (Sept. 2016), pp. 571–582. ISSN: 22136711. DOI: 10.1016/j.stemcr.2016.07.007. URL: http://www.ncbi.nlm.nih.gov/pubmed/27499199%20http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC5031953%20http://linkinghub.elsevier.com/retrieve/pii/S221367111630131X.

[48] A. Gretton et al. "A Kernel Two-Sample Test". In: *undefined* (2012).

[49] Dominik G. Grimm et al. "easyGWAS: A Cloud-Based Platform for Comparing the Results of Genome-Wide Association Studies". In: *The Plant Cell* 29.1 (2017), pp. 5–19. ISSN: 1040-4651. DOI: 10.1105/tpc.16.00551. eprint: http://www.plantcell.org/content/29/1/5.full.pdf. URL: http://www.plantcell.org/content/29/1/5.

[50] Christoph Hafemeister and Rahul Satija. "Normalization and variance stabilization of single-cell RNA-seq data using regularized negative binomial regression". In: *Genome Biology* 20.1 (Dec. 2019), p. 296. ISSN: 1474760X. DOI: 10.1186/s13059-019-1874-1. URL: https://genomebiology.biomedcentral.com/articles/10.1186/s13059-019-1874-1.

[51] Daniel H. Haft et al. "RefSeq: An update on prokaryotic genome annotation and curation". In: *Nucleic Acids Research* 46.D1 (Jan. 2018), pp. D851–D860. ISSN: 13624962. DOI: 10.1093/nar/gkx1068. URL: https://www.ncbi.nlm.nih.gov/refseq/..

[52] Keren Bahar Halpern et al. "Single-cell spatial reconstruction reveals global division of labour in the mammalian liver". In: *Nature* 542.7641 (Feb. 2017), pp. 1–5. ISSN: 14764687. DOI: 10.1038/nature21065. URL: https://www.nature.com/articles/nature21065.

[53] Kazuya Haraguchi et al. "COOMA: A components overlaid mining algorithm for enumerating connected subgraphs with common itemsets". In: *Journal of Graph Algorithms and Applications* 23.2 (2019), pp. 434–458. ISSN: 15261719. DOI: 10.7155/jgaa.00497. URL: http://jgaa.info/vol.

[54] Colleen C Hegg et al. "Microvillous cells expressing IP3 receptor type 3 in the olfactory epithelium of mice." In: *The European journal of neuroscience* 32.10 (Nov. 2010), pp. 1632–45. ISSN: 1460-9568. DOI: 10.1111/j.1460-9568.2010.07449.x. URL: http://www.ncbi.nlm.nih.gov/pubmed/20958798%20http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC4331646.

[55] Ruth Heller et al. "A flexible two-stage procedure for identifying gene sets that are differentially expressed." In: *Bioinformatics (Oxford, England)* 25.8 (Apr. 2009), pp. 1019–25. ISSN: 1367-4811. DOI: 10.1093/bioinformatics/btp076. URL: http://www.ncbi.nlm.nih.gov/pubmed/19213738.

[56] Charles A Herring et al. "Unsupervised Trajectory Analysis of Single-Cell RNA-Seq and Imaging Data Reveals Alternative Tuft Cell Origins in the Gut." In: *Cell systems* 6.1 (Jan. 2018), pp. 37–51. ISSN: 2405-4712. DOI: 10.1016/j.cels.2017.10.012. URL: http://www.ncbi.nlm.nih.gov/pubmed/29153838%20http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC5799016.

[57] Holger Hesse and Rainer Hoefgen. "Molecular aspects of methionine biosynthesis". In: *Trends in Plant Science* 8.6 (June 2003), pp. 259–262. ISSN: 13601385. DOI: 10.1016/S1360-1385(03)00107-9.

[58] Lawrence Hubert and Phipps Arabie. "Comparing partitions". In: *Journal of Classification* 2.1 (Dec. 1985), pp. 193–218. ISSN: 1432-1343. DOI: 10.1007/BF01908075. URL: https://doi.org/10.1007/BF01908075.

[59] Paul Jaccard. "Distribution de la Flore Alpine dans le Bassin des Dranses et dans quelques régions voisines." In: *Bulletin de la Societe Vaudoise des Sciences Naturelles* 37 (Jan. 1901), pp. 241–72. DOI: 10.5169/seals-266440.

[60] Laurent Jacob, Pierre Neuvial, and Sandrine Dudoit. "More power via graph-structured tests for differential expression of gene networks". In: *Ann. Appl. Stat.* 6.2 (June 2012), pp. 561–600. DOI: 10.1214/11-AOAS528. URL: https://doi.org/10.1214/11-AOAS528.

[61] Laurent Jacob, Guillaume Obozinski, and Jean-Philippe Vert. "Group Lasso with overlaps and graph Lasso". In: *In Proceedings of the 26th International Conference on Machine learning*. 2009.

[62] Magali Jaillard et al. "A fast and agnostic method for bacterial genome-wide association studies: Bridging the gap between k-mers and genetic events." In: *PLoS genetics* 14.11 (2018), e1007758. ISSN: 1553-7404. DOI: 10.1371/journal.pgen.1007758. URL: http://www.ncbi.nlm.nih.gov/pubmed/30419019.

[63] Magali Jaillard et al. "Correlation between phenotypic antibiotic susceptibility and the resistome in Pseudomonas aeruginosa". In: *International Journal of Antimicrobial Agents* 50.2 (Aug. 2017), pp. 210–218. ISSN: 18727913. DOI: 10.1016/j.ijantimicag.2017.02.026.

[64] Zhicheng Ji and Hongkai Ji. "TSCAN: Pseudo-time reconstruction and evaluation in single-cell RNA-seq analysis". In: *Nucleic Acids Research* 44.13 (July 2016), e117–e117. ISSN: 0305-1048. DOI: 10.1093/nar/gkw430. URL: https://academic.oup.com/nar/article-lookup/doi/10.1093/nar/gkw430.

[65] Zhicheng Ji and Hongkai Ji. "TSCAN: Pseudo-time reconstruction and evaluation in single-cell RNA-seq analysis". In: *Nucleic Acids Research* 44.13 (July 2016), e117–e117. ISSN: 0305-1048. DOI: 10.1093/nar/gkw430. URL: https://academic.oup.com/nar/article-lookup/doi/10.1093/nar/gkw430.

[66] Olga B Jonas et al. *Drug-resistant infections : a threat to our economic future (Vol. 2) : final report (English)*. Tech. rep. 3. HNP/Agriculture Global Antimicrobial Resistance Initiative Washington, D.C. : World Bank Group., 2017, p. 141. arXiv: License:CreativeCommonsAttributionCCBY3.0IGO. URL: http://documents.worldbank.org/curated/en/323311493396993758/final-report.

[67] M Kanehisa and S Goto. "KEGG: kyoto encyclopedia of genes and genomes". In: *Nucleic Acids Res* 28.1 (Jan. 2000), pp. 27–30. URL: http://www.ncbi.nlm.nih.gov/pubmed/10592173.

[68] J. Kiefer. "K-Sample Analogues of the Kolmogorov-Smirnov and Cramer-V. Mises Tests". In: *The Annals of Mathematical Statistics* 30.2 (June 1959), pp. 420–447. ISSN: 0003-4851. DOI: 10.1214/aoms/1177706261. URL: https://projecteuclid.org/euclid.aoms/1177706261.

[69] Taiyun Kim et al. "Impact of similarity metrics on single-cell RNA-seq data clustering". In: *Briefings in Bioinformatics* 20.6 (Nov. 2019), pp. 2316–2326. ISSN: 1467-5463. DOI: 10.1093/bib/bby076. URL: https://academic.oup.com/bib/article/20/6/2316/5077112.

[70] Vladimir Yu Kiselev, Tallulah S. Andrews, and Martin Hemberg. *Challenges in unsupervised clustering of single-cell RNA-seq data*. May 2019. DOI: 10.1038/s41576-018-0088-9. URL: http://www.nature.com/articles/s41576-018-0088-9.

[71] Vladimir Yu Kiselev et al. "SC3: consensus clustering of single-cell RNA-seq data". In: *Nature Methods* 14.5 (May 2017), pp. 483–486. ISSN: 1548-7091. DOI: 10.1038/nmeth.4236. URL: http://www.nature.com/articles/nmeth.4236.

[72] Vladimir Yu. Kiselev et al. "Perturbations of PIP3 signalling trigger a global remodelling of mRNA landscape and reveal a transcriptional feedback loop". In: *Nucleic Acids Research* 43.20 (Oct. 2015), gkv1015. ISSN: 0305-1048. DOI: 10.1093/nar/gkv1015. URL: https://academic.oup.com/nar/article-lookup/doi/10.1093/nar/gkv1015.

[73] Aleksandra A. Kolodziejczyk et al. *The Technology and Biology of Single-Cell RNA Sequencing*. May 2015. DOI: 10.1016/j.molcel.2015.04.005.

[74] Gennady Korotkevich, Vladimir Sukhov, and Alexey Sergushichev. "Fast gene set enrichment analysis". In: *bioRxiv* (Feb. 2016), p. 060012. DOI: 10.1101/060012. URL: https://doi.org/10.1101/060012.

[75] Jesse H. Krijthe. *Rtsne: T-Distributed Stochastic Neighbor Embedding using Barnes-Hut Implementation*. R package version 0.15. 2015. URL: https://github.com/jkrijthe/Rtsne.

[76] Daisuke Kurotaki et al. "IRF8 inhibits C/EBPα activity to restrain mononuclear phagocyte progenitors from differentiating into neutrophils". In: *Nature Communications* 5.1 (Dec. 2014), p. 4978. ISSN: 2041-1723. DOI: 10.1038/ncomms5978. URL: http://www.nature.com/articles/ncomms5978.

[77] Mark van der Laan and Katherine Pollard. "Hybrid Clustering of Gene Expression Data with Visualization and the Bootstrap". In: *Mark J. van der Laan* 117 (Jan. 2001).

[78] Neil D. Lawrence. "Gaussian process latent variable models for visualisation of high dimensional data". In: *NIPS'03 Proceedings of the 16th International Conference on Neural Information Processing Systems* (2003).

[79] Miguel Lázaro-Gredilla, Steven Van Vaerenbergh, and Neil D. Lawrence. "Overlapping Mixtures of Gaussian Processes for the data association problem". In: *Pattern Recognition* 45.4 (Apr. 2012), pp. 1386–1395. ISSN: 0031-3203. DOI: 10.1016/J.PATCOG.2011.10.004. URL: https://www.sciencedirect.com/science/article/pii/S0031320311004109.

[80] John A. Lees et al. "Sequence element enrichment analysis to determine the genetic basis of bacterial phenotypes". In: *Nature Communications* 7.1 (Sept. 2016), p. 12797. ISSN: 2041-1723. DOI: 10.1038/ncomms12797. URL: https://doi.org/10.1038/ncomms12797.

[81] Yingxin Lin et al. "scClassify: hierarchical classification of cells". In: *bioRxiv* (2019), p. 776948. DOI: 10.1101/776948.

[82] Felipe Llinares-López et al. "Genome-wide detection of intervals of genetic heterogeneity associated with complex traits". In: *Bioinformatics* 31.12 (2015), pp. i240–i249. ISSN: 14602059. DOI: 10.1093/bioinformatics/btv263.

[83] Felipe Llinares-López et al. "Genome-wide genetic heterogeneity discovery with categorical covariates". In: *Bioinformatics* 33.12 (2017), pp. 1820–1828. ISSN: 14602059. DOI: 10.1093/bioinformatics/btx071.

[84] Tapio Lönnberg et al. "Single-cell RNA-seq and computational analysis using temporal mixture modeling resolves TH1/TFH fate bifurcation in malaria". In: *Science Immunology* 2.9 (2017). URL: http://immunology.sciencemag.org/content/2/9/eaal2192.full.

[85] David Lopez-Paz and Maxime Oquab. "Revisiting Classifier Two-Sample Tests". In: *arxiv* (Oct. 2016), pp. 1–15. arXiv: 1610.06545. URL: http://arxiv.org/abs/1610.06545.

[86] Michael I Love, Wolfgang Huber, and Simon Anders. "Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2". In: *Genome Biology* 15.12 (Dec. 2014), p. 550. ISSN: 1465-6906. DOI: 10.1186/s13059-014-0550-8. URL: http://genomebiology.com/2014/15/12/550.

[87] Rohan Lowe et al. "Transcriptomics technologies". In: *PLoS Computational Biology* 13.5 (May 2017), e1005457. ISSN: 15537358. DOI: 10.1371/journal.pcbi.1005457. URL: https://doi.org/10.1371/journal.pcbi.1005457.t001.

[88] Aaron T.L. Lun, Arianne C. Richard, and John C. Marioni. "Testing for differential abundance in mass cytometry data". In: *Nature Methods* 14.7 (June 2017), pp. 707–709. ISSN: 15487105. DOI: 10.1038/nmeth.4295. URL: http://bioconductor.org/packages/.

[89] Julien Mairal and Bin Yu. "Supervised Feature Selection in Graphs with Path Coding Penalties and Network Flows". In: *Journal of Machine Learning Research* 14.39 (2013), pp. 2449–2485. URL: http://jmlr.org/papers/v14/mairal13a.html.

[90] Jean-François Marquis et al. "Interferon Regulatory Factor 8 Regulates Pathways for Antigen Presentation in Myeloid Cells and during Tuberculosis". In: *PLoS Genetics* 7.6 (June 2011). Ed. by Derry C. Roopenian, e1002097. ISSN: 1553-7404. DOI: 10.1371/journal.pgen.1002097. URL: http://dx.plos.org/10.1371/journal.pgen.1002097.

[91] Davis J McCarthy, Yunshun Chen, and Gordon K Smyth. "Differential expression analysis of multifactor RNA-Seq experiments with respect to biological variation." In: *Nucleic acids research* 40.10 (May 2012), pp. 4288–97. ISSN: 1362-4962. DOI: 10.1093/nar/gks042. URL: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3378882&tool=pmcentrez&rendertype=abstract.

[92] José L. McFaline-Figueroa et al. "A pooled single-cell genetic screen identifies regulatory checkpoints in the continuum of the epithelial-to-mesenchymal transition". In: *Nature Genetics* 51.9 (Sept. 2019), pp. 1389–1398. ISSN: 15461718. DOI: 10.1038/s41588-019-0489-5. URL: https://doi.org/10.1038/s41588-019-0489-5.

[93] Leland McInnes, John Healy, and James Melville. "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction". In: *arxiv* (Feb. 2018). arXiv: 1802.03426. URL: http://arxiv.org/abs/1802.03426.

[94] Paul J. McMurdie and Susan Holmes. "phyloseq: An R Package for Reproducible Interactive Analysis and Graphics of Microbiome Census Data". In: *PLoS ONE* 8.4 (Apr. 2013). Ed. by Michael Watson, e61217. DOI: 10.1371/journal.pone.0061217. URL: http://dx.plos.org/10.1371/journal.pone.0061217.

[95] David Merrick et al. "Identification of a mesenchymal progenitor cell hierarchy in adipose tissue." In: *Science (New York, N.Y.)* 364.6438 (Apr. 2019). ISSN: 1095-9203. DOI: 10.1126/science.aav2501. URL: http://www.ncbi.nlm.nih.gov/pubmed/31023895%20http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC6816238.

[96] Shin Ichi Minato et al. "A fast method of statistical assessment for combinatorial hypotheses based on frequent itemset enumeration". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 8725 LNAI. Springer Verlag, 2014, pp. 422–436. ISBN: 9783662448502. DOI: 10.1007/978-3-662-44851-9_27.

[97] John F. Monahan. *Numerical methods of statistics, second edition*. Cambridge University Press, Jan. 2011, pp. 1–447. ISBN: 9780511977176. DOI: 10.1017/CBO9780511977176.

[98] Andrew J. Murphy et al. "ApoE regulates hematopoietic stem cell proliferation, monocytosis, and monocyte accumulation in atherosclerotic lesions in mice". In: *Journal of Clinical Investigation* 121.10 (Oct. 2011), pp. 4138–4149. ISSN: 0021-9738. DOI: 10.1172/JCI57559. URL: http://www.ncbi.nlm.nih.gov/pubmed/21968112%20http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3195472%20http://www.jci.org/articles/view/57559.

[99] Ugrappa Nagalakshmi et al. "The transcriptional landscape of the yeast genome defined by RNA sequencing". In: *Science* 320.5881 (June 2008), pp. 1344–1349. ISSN: 00368075. DOI: 10.1126/science.1158441. URL: https://science.sciencemag.org/content/320/5881/1344%20https://science.sciencemag.org/content/320/5881/1344.abstract.

[100] Rance Nault et al. "Single-Nuclei RNA Sequencing Assessment of the Hepatic Effects of 2,3,7,8-Tetrachlorodibenzo-p-dioxin". In: *CMGH* 11.1 (Jan. 2021), pp. 147–159. ISSN: 2352345X. DOI: 10.1016/j.jcmgh.2020.07.012. URL: https://doi.org/10.1016/j.jcmgh.2020.07.012.

[101] N. J. Nelson. *Microarrays have arrived: Gene expression tool matures*. Apr. 2001. DOI: 10.1093/jnci/93.7.492. URL: https://academic.oup.com/jnci/article-lookup/doi/10.1093/jnci/93.7.492.

[102] M. J. Nueda, S. Tarazona, and A. Conesa. "Next maSigPro: updating maSigPro bioconductor package for RNA-seq time series". In: *Bioinformatics* 30.18 (Sept. 2014), pp. 2598–2602. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btu333. URL: https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btu333.

[103] Shingo Okuno et al. "Parallelization of extracting connected subgraphs with common itemsets in distributed memory environments". In: *Journal of Information Processing* 25.3 (2017), pp. 256–267. ISSN: 18826652. DOI: 10.2197/ipsjjip.25.256.

[104] Andrew J. Page et al. "Roary: rapid large-scale prokaryote pan genome analysis". In: *Bioinformatics* 31.22 (Nov. 2015), pp. 3691–3693. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btv421. URL: https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btv421.

[105] Laetitia Papaxanthos et al. "Finding significant combinations of features in the presence of categorical covariates". In: *NEURIPS* (2016), pp. 2271–2279. ISSN: 10495258. DOI: 10.1007/s00464-011-2087-1.

[106] Alison D. Parisian et al. "SMARCB1 loss interacts with neuronal differentiation state to block maturation and impact cell stability". In: *Genes and Development* 34.19-20 (Oct. 2020), pp. 1316–1329. ISSN: 15495477. DOI: 10.1101/gad.339978.120. URL: http://www.genesdev.org/cgi/doi/10.1101/gad.339978.120..

[107] Rob Patro et al. "Salmon provides fast and bias-aware quantification of transcript expression". In: *Nature Methods* 14.4 (Apr. 2017), pp. 417–419. ISSN: 1548-7091. DOI: 10.1038/nmeth.4197. URL: http://www.nature.com/articles/nmeth.4197.

[108] Franziska Paul et al. "Transcriptional Heterogeneity and Lineage Commitment in Myeloid Progenitors". In: *Cell* 163.7 (Dec. 2015), pp. 1663–1677. ISSN: 0092-8674. DOI: 10.1016/J.CELL.2015.11.013. URL: https://www.sciencedirect.com/science/article/pii/S0092867415014932?via%3Dihub#app3.

[109] Yao Fei Pei et al. "Silencing of LAMC2 Reverses Epithelial-Mesenchymal Transition and Inhibits Angiogenesis in Cholangiocarcinoma via Inactivation of the Epidermal Growth Factor Receptor Signaling Pathway". In: *American Journal of Pathology* 189.8 (Aug. 2019), pp. 1637–1653. ISSN: 15252191. DOI: 10.1016/j.ajpath.2019.03.012. URL: https://pubmed.ncbi.nlm.nih.gov/31345467/.

[110] Pavel A. Pevzner, Haixu Tang, and Michael S. Waterman. "An Eulerian path approach to DNA fragment assembly". In: *Proceedings of the National Academy of Sciences of the United States of America* 98.17 (Aug. 2001), pp. 9748–9753. ISSN: 00278424. DOI: 10.1073/pnas.171285098. URL: www.pnas.orgcgidoi10.1073pnas.171285098.

[111] Christina Pfaff et al. "ALY RNA-Binding Proteins Are Required for Nucleocytosolic mRNA Transport and Modulate Plant Growth and Development". In: *Plant physiology* 177.1 (May 2018), pp. 226–240. ISSN: 15322548. DOI: 10.1104/pp.18.00173.

[112] Hannah A. Pliner, Jay Shendure, and Cole Trapnell. "Supervised classification enables rapid annotation of cell atlases". In: *Nature Methods* 16.10 (Oct. 2019), pp. 983–986. ISSN: 15487105. DOI: 10.1038/s41592-019-0535-3.

[113] Robert A. Power, Julian Parkhill, and Tulio De Oliveira. *Microbial genome-wide association studies: lessons from human GWAS*. Dec. 2016. DOI: 10.1038/nrg.2016.132. URL: www.nature.com/nrg.

[114] Alex E. Pozhitkov, Diethard Tautz, and Peter A. Noble. "Oligonucleotide microarrays: Widely applied - Poorly understood". In: *Briefings in Functional Genomics and Proteomics* 6.2 (Sept. 2007), pp. 141–148. ISSN: 14739550. DOI: 10.1093/bfgp/elm014. URL: https://academic.oup.com/bfg/article-lookup/doi/10.1093/bfgp/elm014.

[115] Xiaojie Qiu et al. "Reversed graph embedding resolves complex single-cell trajectories". In: *Nature Methods* (Aug. 2017). DOI: 10.1038/nmeth.4402. URL: https://www.nature.com/nmeth/journal/vaop/ncurrent/full/nmeth.4402.html.

[116] Atif Rahman et al. "Association mapping from sequencing reads using k-mers". In: *eLife* 7 (June 2018). ISSN: 2050084X. DOI: 10.7554/eLife.32920.

[117] P. Ramachandran et al. "Resolving the fibrotic niche of human liver cirrhosis at single-cell level". In: *Nature* 575.7783 (Nov. 2019), pp. 512–518. ISSN: 14764687. DOI: 10.1038/s41586-019-1631-3. URL: https://doi.org/10.1038/s41586-019-1631-3.

[118] William M. Rand. "Objective Criteria for the Evaluation of Clustering Methods". In: *Journal of the American Statistical Association* 66.336 (1971), pp. 846–850. DOI: 10.1080/01621459.1971.10482356. eprint: https://www.tandfonline.com/doi/pdf/10.1080/01621459.1971.10482356. URL: https://www.tandfonline.com/doi/abs/10.1080/01621459.1971.10482356.

[119] Franck Rapaport et al. "Classification of microarray data using gene networks". In: *BMC Bioinformatics* 8.1 (Feb. 2007), p. 35. ISSN: 1471-2105. DOI: 10.1186/1471-2105-8-35. URL: https://doi.org/10.1186/1471-2105-8-35.

[120] Xianwen Ren et al. "COVID-19 immune features revealed by a large-scale single cell transcriptome atlas". In: *Cell* 0.0 (Feb. 2021). ISSN: 00928674. DOI: 10.1016/j.cell.2021.01.053. URL: https://doi.org/10.1016/j.cell.2021.01.053.

[121] Davide Risso et al. "clusterExperiment and RSEC: A Bioconductor package and framework for clustering of single-cell and other large gene expression datasets". In: *PLOS Computational Biology* 14.9 (Sept. 2018). Ed. by Aaron E. Darling, e1006378. ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1006378. URL: http://dx.plos.org/10.1371/journal.pcbi.1006378.

[122] M. E. Ritchie et al. "limma powers differential expression analyses for RNA-sequencing and microarray studies". In: *Nucleic Acids Research* 43.7 (Jan. 2015), e47. ISSN: 0305-1048. DOI: 10.1093/nar/gkv007. URL: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=4402510&tool=pmcentrez&rendertype=abstract.

[123] Hector Roux De Bézieux et al. "Trajectory inference across multiple conditions with condiments: differential topology, progression, differentiation, and expression". In: *bioRxiv* (Mar. 2021), p. 2021.03.09.433671. DOI: 10.1101/2021.03.09.433671. URL: https://doi.org/10.1101/2021.03.09.433671.

[124] Hector Roux de Bézieux et al. "Finding All Significant Closed Connected Subgraphs". In: *Machine Learning in Computational Biology (MLCB)* (2020).

[125] Hector Roux de Bézieux et al. "Improving replicability in single-cell RNA-Seq cell type discovery with Dune". In: *bioRxiv* (Mar. 2020), p. 2020.03.03.974220. DOI: 10.1101/2020.03.03.974220.

[126] Wouter Saelens et al. "A comparison of single-cell trajectory inference methods". In: *Nature Biotechnology* (Apr. 2019), p. 1. ISSN: 1087-0156. DOI: 10.1038/s41587-019-0071-9. URL: http://www.nature.com/articles/s41587-019-0071-9.

[127] Tiziana Sanavia, Francesca Finotello, and Barbara Di Camillo. "FunPat: function-based pattern analysis on RNA-seq time series data". In: *BMC Genomics* 16.Suppl 6 (2015), S2. ISSN: 1471-2164. DOI: 10.1186/1471-2164-16-S6-S2. URL: http://www.ncbi.nlm.nih.gov/pubmed/26046293%20http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC4460925%20http://bmcgenomics.biomedcentral.com/articles/10.1186/1471-2164-16-S6-S2.

[128] F. Sanger et al. "Cloning in single-stranded bacteriophage as an aid to rapid DNA sequencing". In: *Journal of Molecular Biology* 143.2 (Oct. 1980), pp. 161–178. ISSN: 00222836. DOI: 10.1016/0022-2836(80)90196-5.

[129] Mark Schena et al. "Quantitative monitoring of gene expression patterns with a complementary DNA microarray". In: *Science* 270.5235 (Oct. 1995), pp. 467–470. ISSN: 00368075. DOI: 10.1126/science.270.5235.467. URL: https://science.sciencemag.org/content/270/5235/467%20https://science.sciencemag.org/content/270/5235/467.abstract.

[130] Åsa Segerstolpe et al. "Single-Cell Transcriptome Profiling of Human Pancreatic Islets in Health and Type 2 Diabetes". In: *Cell Metabolism* 24.4 (Oct. 2016), pp. 593–607. ISSN: 19327420. DOI: 10.1016/j.cmet.2016.08.020. URL: https://www.sciencedirect.com/science/article/pii/S1550413116304363?via%7B%5C%%7D3Dihub.

[131] Mio Seki and Jun Sese. "Identification of active biological networks and common expression conditions". In: *8th IEEE International Conference on BioInformatics and BioEngineering, BIBE 2008*. 2008. ISBN: 9781424428458. DOI: 10.1109/BIBE.2008.4696746.

[132] Alexey Sergushichev. "An algorithm for fast preranked gene set enrichment analysis using cumulative statistic calculation". In: *bioRxiv* (June 2016), p. 060012. DOI: 10.1101/060012. URL: https://www.biorxiv.org/content/early/2016/06/20/060012.

[133] Jun Sese, Mio Seki, and Mutsumi Fukuzaki. "Mining networks with shared items". In: *International Conference on Information and Knowledge Management, Proceedings*. New York, New York, USA: ACM Press, 2010, pp. 1681–1684. ISBN: 9781450300995. DOI: 10.1145/1871437.1871703. URL: http://portal.acm.org/citation.cfm?doid=1871437.1871703.

[134] Jun Sese et al. "Statistically significant subgraphs for genome-wide association study". In: *SDM* 47 (2014), pp. 1–7.

[135] Samuel K. Sheppard et al. "Genome-wide association study identifies vitamin B5 biosynthesis as a host specificity factor in Campylobacter". In: *Proceedings of the National Academy of Sciences of the United States of America* 110.29 (July 2013), pp. 11923–11927. ISSN: 00278424. DOI: 10.1073/pnas.1305559110. URL: http://pubmlst.org/campylobacter/..

[136] Dong-Mi Shin, Chang-Hoon Lee, and Herbert C. Morse. "IRF8 Governs Expression of Genes Involved in Innate and Adaptive Immunity in Human and Mouse Germinal Center B Cells". In: *PLoS ONE* 6.11 (Nov. 2011). Ed. by Michael B. Fessler, e27384. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0027384. URL: http://dx.plos.org/10.1371/journal.pone.0027384.

[137] Nikolai V Smirnov. "On the estimation of the discrepancy between empirical curves of distribution for two independent samples". In: *Bull. Math. Univ. Moscou* 2.2 (1939), pp. 3–14.

[138] GK Smyth. "Linear models and empirical bayes methods for assessing differential expression in microarray experiments". In: *Statistical applications in genetics and molecular biology* 3.1 (2004), pp. 1–26. URL: http://www.degruyter.com/view/j/sagmb.2004.3.1/sagmb.2004.3.1.1027/sagmb.2004.3.1.1027.xml.

[139] Charlotte Soneson and Mark D Robinson. "iCOBRA: open, reproducible, standardized and live method benchmarking". In: *Nature Methods* 13.4 (Mar. 2016), pp. 283–283. ISSN: 1548-7091. DOI: 10.1038/nmeth.3805. URL: http://www.nature.com/doifinder/10.1038/nmeth.3805.

[140] Samuel Andrew Stouffer. *Adjustment during army life*. Princeton University Press, 1949.

[141] Kelly Street et al. "Slingshot: cell lineage and pseudotime inference for single-cell transcriptomics". In: *BMC Genomics* 19.1 (Dec. 2018), p. 477. ISSN: 1471-2164. DOI: 10.1186/s12864-018-4772-0. URL: https://bmcgenomics.biomedcentral.com/articles/10.1186/s12864-018-4772-0.

[142] Tim Stuart et al. "Comprehensive Integration of Single-Cell Data". In: *Cell* 177.7 (June 2019), 1888–1902.e21. ISSN: 10974172. DOI: 10.1016/j.cell.2019.05.031. URL: http://www.ncbi.nlm.nih.gov/pubmed/31178118%20http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC6687398.

[143] Mahito Sugiyama and Karsten M. Borgwardt. "Finding Significant Combinations of Continuous Features". In: (2017), pp. 1–11. arXiv: 1702.08694. URL: http://arxiv.org/abs/1702.08694.

[144] Mahito Sugiyama et al. "Significant subgraph mining with multiple testing correction". In: *SIAM International Conference on Data Mining 2015, SDM 2015*. Society for Industrial and Applied Mathematics Publications, 2015, pp. 37–45. ISBN: 9781510811522. DOI: 10.1137/1.9781611974010.5. arXiv: 1407.0316.

[145] Valentine Svensson, Eduardo da Veiga Beltrame, and Lior Pachter. "A curated database reveals trends in single-cell transcriptomics". In: *Database : the journal of biological databases and curation* 2020 (Nov. 2020). ISSN: 17580463. DOI: 10.1093/database/baaa073. URL: https://academic.oup.com/database/article/doi/10.1093/database/baaa073/6008692.

[146] Fuchou Tang et al. "mRNA-Seq whole-transcriptome analysis of a single cell". In: *Nature Methods* 6.5 (Apr. 2009), pp. 377–382. ISSN: 15487091. DOI: 10.1038/nmeth.1315. URL: https://www.nature.com/articles/nmeth.1315.

[147] Fuchou Tang et al. "Tracing the derivation of embryonic stem cells from the inner cell mass by single-cell RNA-seq analysis". In: *Cell Stem Cell* 6.5 (May 2010), pp. 468–478. ISSN: 19345909. DOI: 10.1016/j.stem.2010.03.015.

[148] R. E. Tarone. "A Modified Bonferroni Method for Discrete Data". In: *Biometrics* 46.2 (June 1990), p. 515. ISSN: 0006341X. DOI: 10.2307/2531456.

[149] Bosiljka Tasic et al. "Shared and distinct transcriptomic cell types across neocortical areas". In: *Nature* 563.7729 (Nov. 2018), pp. 72–78. ISSN: 14764687. DOI: 10.1038/s41586-018-0654-5. URL: http://www.nature.com/articles/s41586-018-0654-5.

[150] D Tenenbaum. *KEGGREST: Client-side REST access to KEGG*. 2020.

[151] Aika Terada et al. "Statistical significance of combinatorial regulations". In: *Proceedings of the National Academy of Sciences of the United States of America* 110.32 (Aug. 2013), pp. 12996–13001. DOI: 10.1073/pnas.90.1.203.

[152] J. Towns et al. "XSEDE: Accelerating Scientific Discovery". In: *Computing in Science & Engineering* 16.5 (Sept. 2014), pp. 62–74. ISSN: 1521-9615. DOI: 10.1109/MCSE.2014.80. URL: doi.ieeecomputersociety.org/10.1109/MCSE.2014.80.

[153] V. A. Traag, L. Waltman, and N. J. van Eck. "From Louvain to Leiden: guaranteeing well-connected communities". In: *Scientific Reports* 9.1 (Dec. 2019), p. 5233. ISSN: 2045-2322. DOI: 10.1038/s41598-019-41695-z. URL: http://www.nature.com/articles/s41598-019-41695-z.

[154] Cole Trapnell et al. "The dynamics and regulators of cell fate decisions are revealed by pseudotemporal ordering of single cells". In: *Nature Biotechnology* 32.4 (Apr. 2014), pp. 381–386. ISSN: 1087-0156. DOI: 10.1038/nbt.2859. URL: http://www.ncbi.nlm.nih.gov/pubmed/24658644%20http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC4122333%20http://www.nature.com/articles/nbt.2859.

[155] T Uno, M Kiyomi, and H Arimura. "Efficient mining algorithms for frequent/closed/maximal itemsets". In: *IEEE ICDM Workshop on Frequent Itemset Mining Implementations*. 2004.

[156] Koen Van den Berge et al. "Observation weights unlock bulk RNA-seq tools for zero inflation and single-cell applications". In: *Genome Biology* 19.1 (Dec. 2018), p. 24. ISSN: 1474-760X. DOI: 10.1186/s13059-018-1406-4. URL: https://genomebiology.biomedcentral.com/articles/10.1186/s13059-018-1406-4.

[157] Koen Van den Berge et al. "stageR: a general stage-wise method for controlling the gene-level false discovery rate in differential expression and differential transcript usage". In: *Genome Biology* 18.1 (2017), p. 151. ISSN: 1474-760X. DOI: 10.1186/s13059-017-1277-0. URL: http://www.ncbi.nlm.nih.gov/pubmed/28784146%20http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC5547545%20http://genomebiology.biomedcentral.com/articles/10.1186/s13059-017-1277-0.

[158] Koen Van den Berge et al. "Trajectory-based differential expression analysis for single-cell sequencing data". In: *Nature Communications* 11.1 (Dec. 2020), p. 1201. ISSN: 2041-1723. DOI: 10.1038/s41467-020-14766-3. URL: http://www.nature.com/articles/s41467-020-14766-3.

[159] L.J.P. van der Maaten. "Accelerating t-SNE using Tree-Based Algorithms". In: *Journal of Machine Learning Research* 15 (2014), pp. 3221–3245.

[160] L.J.P. van der Maaten and G.E. Hinton. "Visualizing High-Dimensional Data Using t-SNE". In: *Journal of Machine Learning Research* 9 (2008), pp. 2579–2605.

[161] Fabio Vandin, Eli Upfal, and Benjamin J. Raphael. "Algorithms for Detecting Significantly Mutated Pathways in Cancer". In: *Proceedings of the 14th Annual International Conference on Research in Computational Molecular Biology*. RECOMB'10. Lisbon, Portugal: Springer-Verlag, 2010, pp. 506–521. ISBN: 3642126820. DOI: 10.1007/978-3-642-12683-3_33. URL: https://doi.org/10.1007/978-3-642-12683-3_33.

[162] Charles J. Vaske et al. "Inference of patient-specific pathway activities from multi-dimensional cancer genomics data using PARADIGM". In: *Bioinformatics* 26.12 (June 2010), pp. i237–i245. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btq182. eprint: https://academic.oup.com/bioinformatics/article-pdf/26/12/i237/16893608/btq182.pdf. URL: https://doi.org/10.1093/bioinformatics/btq182.

[163] Peter M. Visscher et al. *10 Years of GWAS Discovery: Biology, Function, and Translation*. June 2017. DOI: 10.1016/j.ajhg.2017.06.005. URL: http://dx.doi.org/10.1016/j.ajhg.2017.06.005..

[164] Florian Wagner and Itai Yanai. "Moana: A robust and scalable cell type classification framework for single-cell RNA-Seq data". In: *bioRxiv* (2018), p. 456129. DOI: 10.1101/456129. URL: https://www.biorxiv.org/content/10.1101/456129v1.

[165] Zhong Wang, Mark Gerstein, and Michael Snyder. *RNA-Seq: A revolutionary tool for transcriptomics*. Jan. 2009. DOI: 10.1038/nrg2484. URL: www.nature.com/reviews/genetics.

[166] Leonid N Wasserstein. "Markov processes over denumerable products of spaces describing large systems of automata". In: *Problems of Information Transmission* 5.3 (1969), pp. 47–52.

[167] Kristen L. Wells et al. *Combined transient ablation and single cell RNA sequencing reveals the development of medullary thymic epithelial cells*. June 2020. DOI: 10.1101/2020.06.19.160424. URL: https://doi.org/10.1101/2020.06.19.160424.

[168] Jeffery Westbrook and Robert E. Tarjan. "Maintaining bridge-connected and biconnected components on-line". In: *Algorithmica* 7.1-6 (Dec. 1992), pp. 433–464. ISSN: 01784617. DOI: 10.1007/BF01758773.

[169] KA Wetterstrand. *DNA Sequencing Costs: Data from the NHGRI Genome Sequencing Program (GSP)*. (Visited on 03/13/2021).

[170] WHO. *ANTIBACTERIAL AGENTS IN CLINICAL DEVELOPMENT*. Tech. rep. WHO/EMP/IAU/2017.11. World Health Organization, 2017. URL: http://www.who.int/medicines/areas/rational%7B%5C_%7Duse/antibacterial%7B%5C_%7Dagents%7B%5C_%7Dclinical%7B%5C_%7Ddevelopment/en/.

[171] Brian T. Wilhelm et al. "Dynamic repertoire of a eukaryotic transcriptome surveyed at single-nucleotide resolution". In: *Nature* 453.7199 (June 2008), pp. 1239–1243. ISSN: 14764687. DOI: 10.1038/nature07002. URL: https://www.nature.com/articles/nature07002.

[172] Simon N. Wood. *Generalized additive models : an introduction with R*. Ed. by Joseph K Blitzstein et al. Second. Boca Ration, FL: Chapman and Hall/CRC, 2017, p. 476. ISBN: 9781498728331.

[173] Simon N. Wood. *Mixed GAM Computation Vehicle with Automatic Smoothness Estimation*. CRC press, 2019, pp. 1–476. ISBN: 9781498728348. DOI: 10.1201/9781315370279.

[174] Simon N. Wood, Natalya Pya, and Benjamin Säfken. "Smoothing Parameter and Model Selection for General Smooth Models". In: *Journal of the American Statistical Association* 111.516 (Oct. 2016), pp. 1548–1563. ISSN: 0162-1459. DOI: 10.1080/01621459.2016.1180986. URL: https://www.tandfonline.com/doi/full/10.1080/01621459.2016.1180986.

[175] Jenny Y. Xue et al. "Rapid non-uniform adaptation to conformation-specific KRAS(G12C) inhibition". In: *Nature* 577.7790 (Jan. 2020), pp. 421–425. ISSN: 14764687. DOI: 10.1038/s41586-019-1884-x. URL: https://doi.org/10.1038/s41586-019-1884-x.

[176] Zizhen Yao et al. "An integrated transcriptomic and epigenomic atlas of mouse primary motor cortex cell types". In: *bioRxiv* (Mar. 2020), p. 2020.02.29.970558. DOI: 10.1101/2020.02.29.970558.

[177] Luke Zappia and Alicia Oshlack. "Clustering trees: a visualization for evaluating clusterings at multiple resolutions". In: *GigaScience* 7.7 (2018), pp. 1–9. ISSN: 2047217X. DOI: 10.1093/gigascience/giy083. URL: http://orcid.org/0000-0001-9788-5690Address:.

[178] Luke Zappia, Belinda Phipson, and Alicia Oshlack. "Splatter: Simulation of single-cell RNA sequencing data". In: *Genome Biology* 18.1 (Sept. 2017), p. 174. ISSN: 1474760X. DOI: 10.1186/s13059-017-1305-0. URL: http://genomebiology.biomedcentral.com/articles/10.1186/s13059-017-1305-0.

[179] Allen W. Zhang et al. "Probabilistic cell-type assignment of single-cell RNA-seq for tumor microenvironment profiling". In: *Nature Methods* 16.10 (Oct. 2019), pp. 1007–1015. ISSN: 15487105. DOI: 10.1038/s41592-019-0529-1.

[180] Wenyu Zhang et al. "A Practical Comparison of De Novo Genome Assembly Software Tools for Next-Generation Sequencing Technologies". In: *PLoS ONE* 6.3 (Mar. 2011). Ed. by I. King Jordan, e17915. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0017915. URL: https://dx.plos.org/10.1371/journal.pone.0017915.

[181] Ze Zhang et al. "Scina: Semi-supervised analysis of single cells in silico". In: *Genes* 10.7 (July 2019), p. 531. ISSN: 20734425. DOI: 10.3390/genes10070531. URL: https://www.mdpi.com/2073-4425/10/7/531.

[182] Jun Zhao et al. "Detecting regions of differential abundance between scRNA-Seq datasets". In: *bioRxiv* (Oct. 2020), p. 711929. DOI: 10.1101/711929. URL: https://doi.org/10.1101/711929.

[183] Grace X.Y. Zheng et al. "Massively parallel digital transcriptional profiling of single cells". In: *Nature Communications* 8.1 (Jan. 2017), pp. 1–12. ISSN: 20411723. DOI: 10.1038/ncomms14049. URL: www.nature.com/naturecommunications.

Appendix A

Supplementary Figures for chapter 2

Seurat's tuning parameters

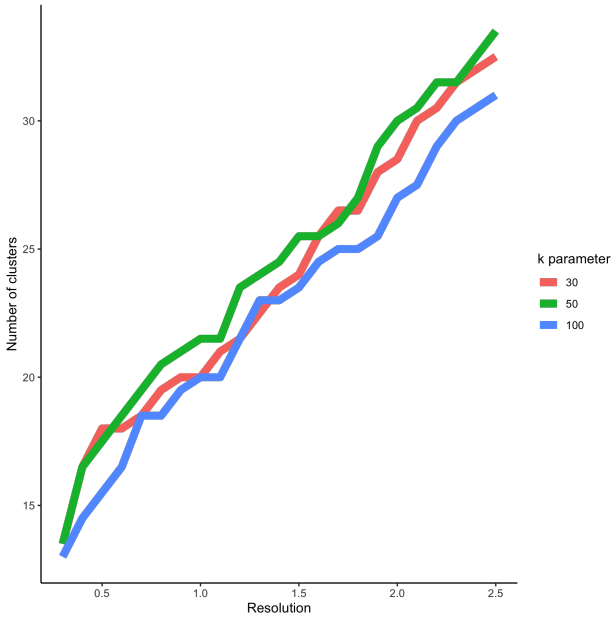
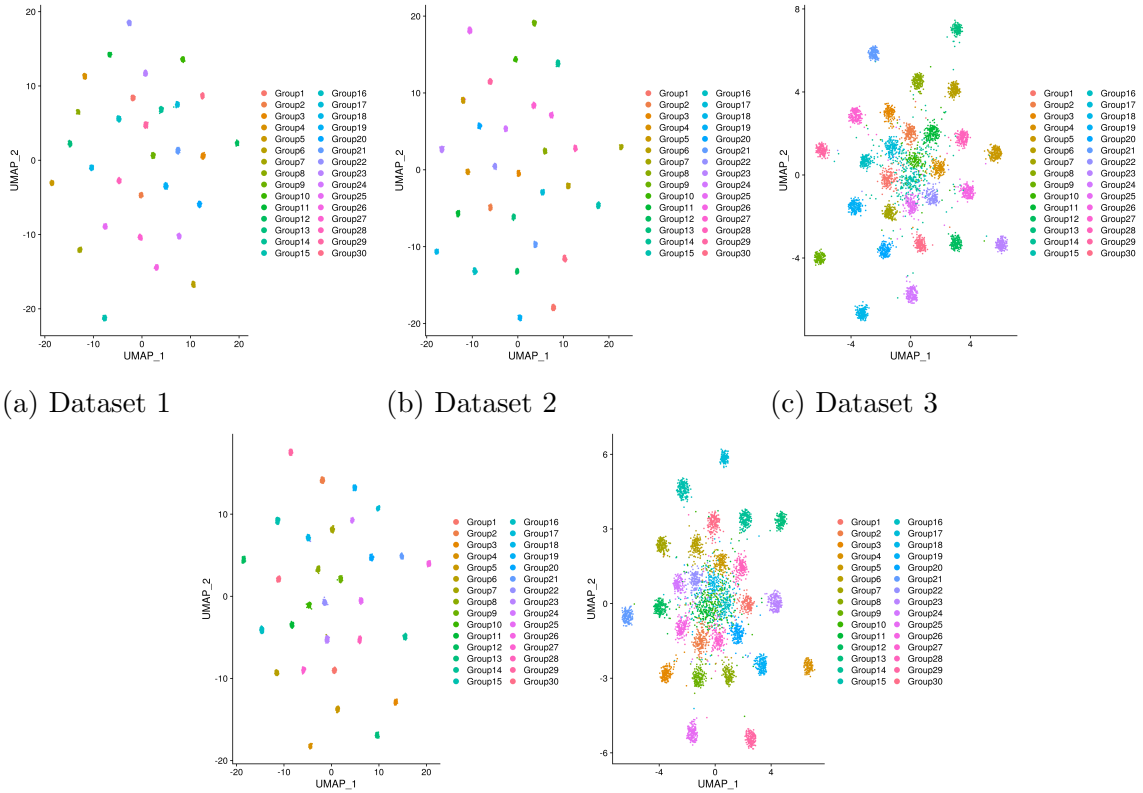


Figure A.1: *Impact of Seurat's two main tuning parameters on the number of clusters.* The Seurat algorithm is run on the two AIBS snRNA-Smart datasets, for a grid of tuning parameter values. For increasing values of `resolution` and fixed values of k, the number of clusters is always increasing. For increasing values of k and fixed values of `resolution`, the number of clusters can either increase or decrease.

Simulated datasets



(a) Dataset 1 (b) Dataset 2 (c) Dataset 3

(d) Dataset 4 (e) Dataset 5

Figure A.2: *Simulated datasets.* Two-dimensional representations of the simulated datasets using UMAP; plotting symbols for the cells are colored by the ground-truth cluster labels.

Resolution-replicability trade-off

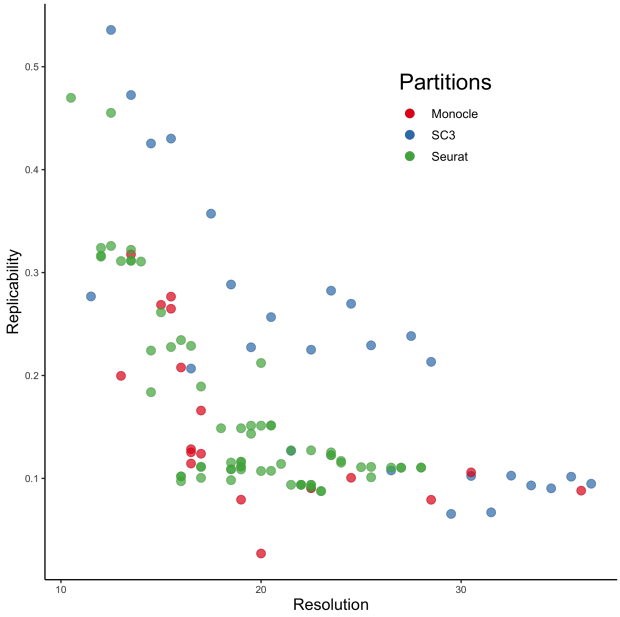


Figure A.3: *Resolution-replicability trade-off for the Pancreas datasets.* Seurat, SC3, and Monocle are run on the two Pancreas datasets, as described in "Methods", for a wide range of tuning parameter values. Then, the MetaNeighbor method is used to compute replicability scores for the resulting clusters between these two datasets. A trade-off between replicability and resolution is visible.

Results for **Dune** with ARI merging

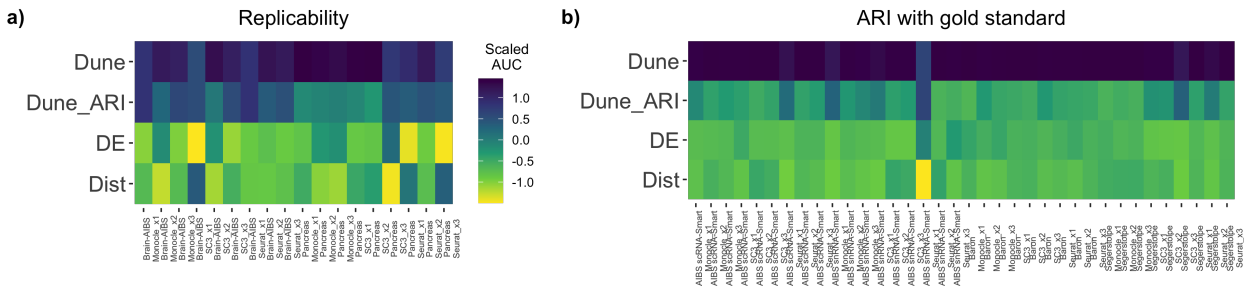


Figure A.4: *Comparison of methods.* SC3, Seurat, and Monocle were run on all datasets, for a wide range of tuning parameter values. Then, merging by Dune with ARI, Dune with NMI, and the two hierarchical procedures is evaluated using either replicability, measured via the MetaNeighbor method, or ARI with gold-standard labels. This yields 18 comparisons of AUC for replicabilty (**a.**) and 36 comparison of AUC for ARI with gold standard (**b.**). AUC values are displayed in the pseudocolor image, after being scaled to have a column mean of zero and column variance of 1. This was done to make AUC values comparable across datasets, clustering methods, and parameter values, since the AUC can have different scales across scenarios.

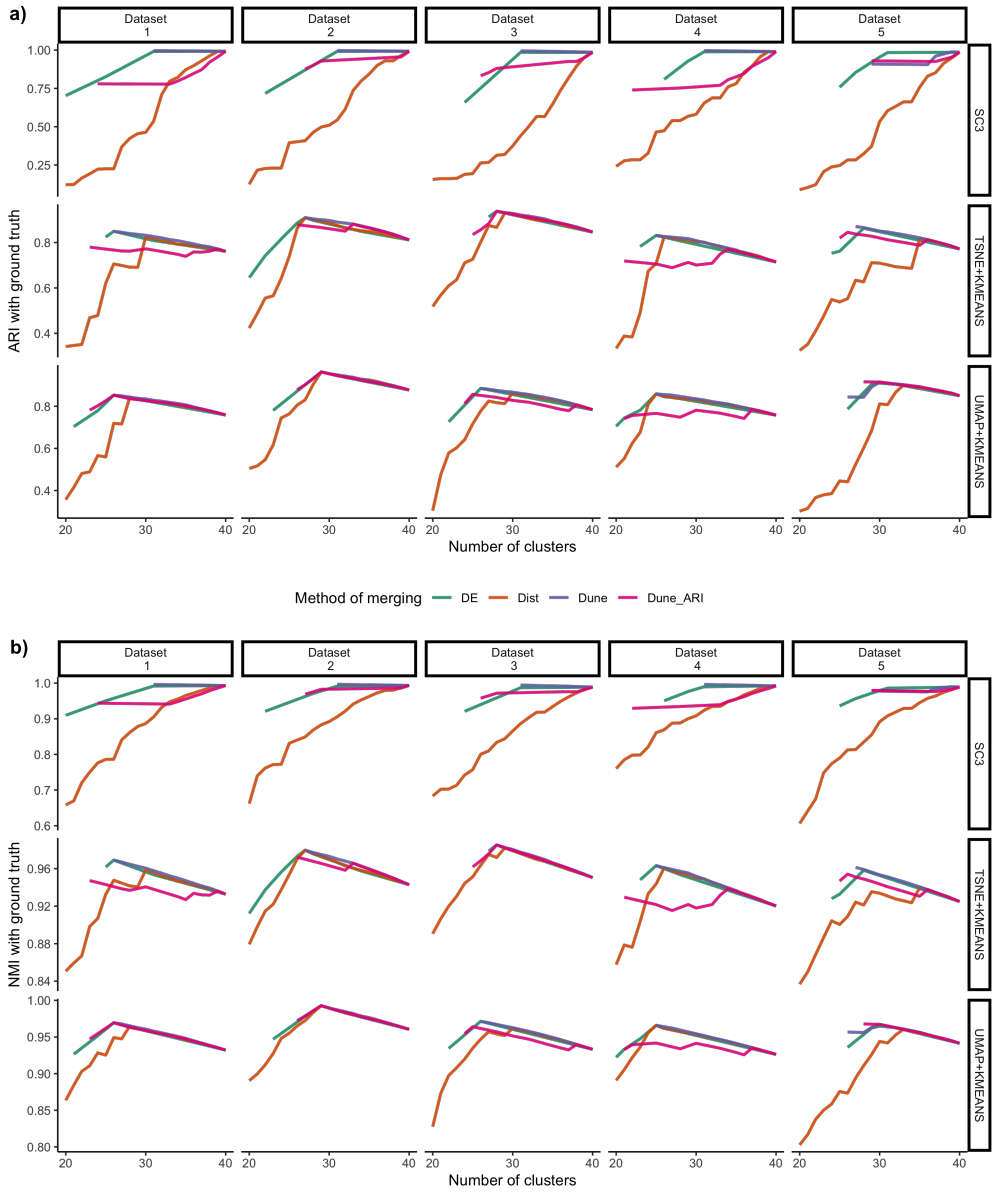


Figure A.5: *Simulation results including the ARI-based version of* **Dune**. For each of the four merging methods, as merging occurs on a simulated dataset, the ARI (**a.**) or NMI (**b.**) with the ground truth is tracked as the number of clusters decreases. **Dune** with ARI performs either on-par or sub-par with the NMI-based version.

Appendix B

Dune workflow on the baron dataset

Dune workflow on the baron dataset

Hector Roux de Bézieux

30 April , 2021

Contents

1 Load data

2 Pre-processing

3 Creating inputs to Dune

3.1 **SC3** . . . . .

3.2 **Seurat** . . . . .

3.3 **Seurat** with **scvi** . . . . .

4 Dune

4.1 Running **Dune** . . . . .

4.2 Vizualing the merging . . . . .

5 Picking the final clustering result to use

5.1 Manual selection . . . . .

5.2 Selection based on sihouette . . . . .

5.3 Comparing with the original labels . . . . .

6 Runtimes

References

In this workflow, we will demonstrate a full full scRNA-Seq workflow using **Dune** on an example dataset. We rely on the the data from (Baron et al. 2016), a human pancreas dataset of 8569 samples. We will demonstrate how to generate various input clustering results, how to merge clusters using **Dune** and how to select the best output for use in downstream analysis. We will also monitor run times to show the impact of running **Dune** versus a workflow without it.

1 Load data

We rely on a pre-processed dataset where the count matrix has already been computed, using the **scRNAseq** R package. The dataset also contains the id of the human donor for each cell, which are used as batch labels. It also contains the cell labels assignements from the original publication. Note that, in that publication, cells were clustered using hierarchical clustering with a final manual merging step.

```
set.seed(19)
suppressPackageStartupMessages({
  library(SingleCellExperiment)
  library(stringr)
  library(scRNAseq)
})
# Load pre-processed dataset
sce <- BaronPancreasData()
# Filter very lowly expressed genes for computational practices.
filt <- rowSums(counts(sce) >= 2) >= 10
sce <- sce[filt, ]
print(sce)
```

```
## class: SingleCellExperiment
## dim: 12336 8569
## metadata(0):
## assays(1): counts
## rownames(12336): A1CF A2M ... ZZZ3 pk
## rowData names(0):
## colnames(8569): human1_lib1.final_cell_0001 human1_lib1.final_cell_0002
##   ... human4_lib3.final_cell_0700 human4_lib3.final_cell_0701
## colData names(2): donor label
## reducedDimNames(0):
## altExpNames(0):
```

2 Pre-processing

Before running clustering algorithms, we will rely on two normalization pipelines.

- The default pipeline of **Seurat** (Stuart et al. 2019).

```
suppressPackageStartupMessages({
  library(Seurat)
})
pre_process_time <- system.time({
  se <- CreateSeuratObject(counts = counts(sce),
                           min.cells = 0,
                           min.features = 0,
                           project = "de")
  se <- AddMetaData(se, as.data.frame(colData(sce)))
  se <- NormalizeData(se, verbose = FALSE)
  se <- FindVariableFeatures(se, selection.method = 'vst', nfeatures = 4000,
                             verbose = FALSE)
  se <- se[VariableFeatures(se), ]
  se <- ScaleData(object = se, vars.to.regress = c("nCount_RNA", "donor"))
  sce <- as.SingleCellExperiment(se)
})
```

```
## Regressing out nCount_RNA, donor

## Centering and scaling data matrix
```

- The **scvi** method (Lopez et al. 2018).

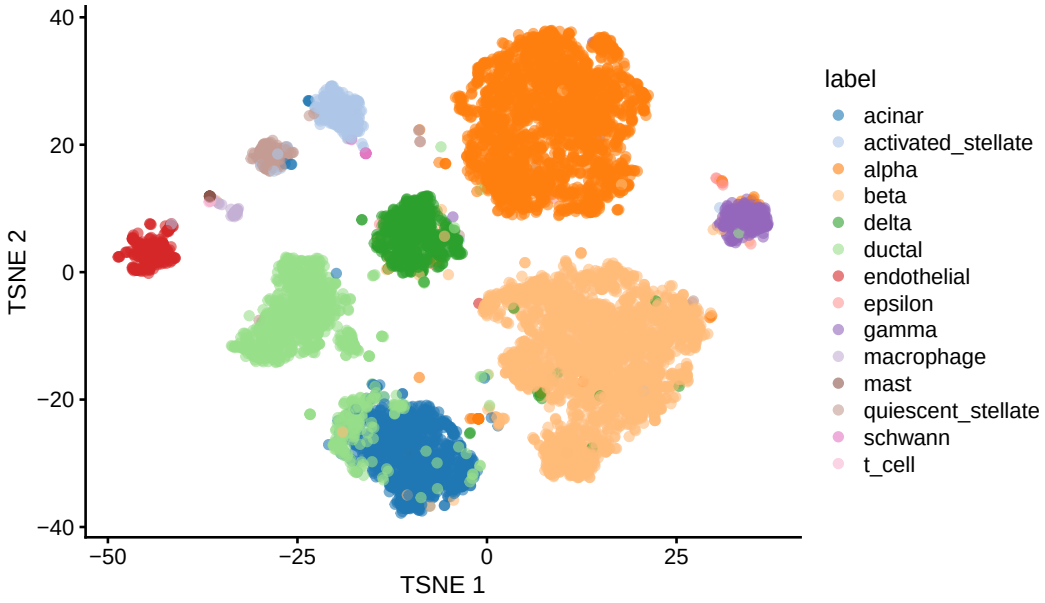
```
suppressPackageStartupMessages(library(reticulate))
scvi <- import('scvi', convert = FALSE)
anndata <- import("anndata")
np <- import("numpy")
sc <- import("scanpy")
scvi_time <- system.time({
  scvi$settings$seed = 0L
  adata <- anndata$AnnData(X = as.sparse(t(counts(sce))),
                           obs = data.frame(cells = colnames(sce),
                                            batch = sce$donor))
  scvi$data$setup_anndata(adata, batch_key = "batch")
  model <- scvi$model$SCVI(adata)
  model$train(n_epochs = 100L, n_epochs_kl_warmup = 25L)
})
```

We can visualize the latent space produced by **scvi** using the labels from the original publication, and reducing the 10 dimensions of the latent space to 2 using t-SNE (van der Maaten and Hinton 2008, @tsne2, @tsne3).

```
suppressPackageStartupMessages(library(scater))
```

```
## Warning: package 'scater' was built under R version 4.0.4
```

```
reducedDim(sce, "scvi") <- py_to_r(model$get_latent_representation())
denoised <- t(model$get_normalized_expression(adata, library_size = 10e4) %>%
                py_to_r())
dimnames(denoised) <- dimnames(counts(sce))
assay(sce, "denoised") <- log1p(denoised)
sce <- runTSNE(sce, dimred = "scvi")
plotTSNE(sce, colour_by = "label")
```



As we can see, **scvi** mostly produces a latent space that is consistent with the original labels. Note however that this is information that would not available while analyzing a new dataset. One would instead need to rely on known-marker genes.

3 Creating inputs to Dune

Dune takes as input a set of clustering results. We will generate a set of such results using a combination of clustering methods and normalization techniques:

- **SC3** (Kiselev et al. 2017) using as input the denoised count matrix from **scvi**.
- **Seurat** using as input the latent space from **scvi**.
- **Seurat** using as input the top pcs from the count matrix normalized using the **Seurat** pre-processing pipeline.

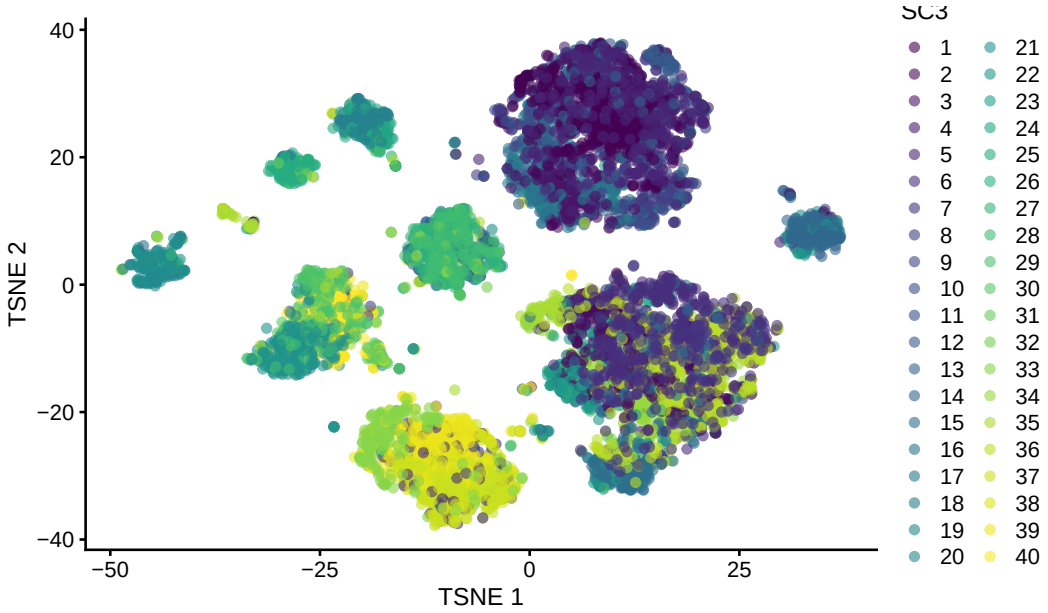
3.1 SC3

SC3 is a consensus method that takes as input a normalized count matrix and outputs a set of cluster labels. The **SC3** package provides a function to estimate the value of K, the exact number of clusters, which we will use.

Since the dataset has more than 5000 cells, **SC3** is automatically run in hybrid mode to lower runtime. However, the process can still be quite slow. The code below is run in the default mode. If you want it to run, we recommand seting `default=FALSE`.

```
suppressPackageStartupMessages(library(SC3))
default <- FALSE
sc3_time <- system.time({
  sce_sc3 <- sce
  logcounts(sce_sc3) <- assay(sce, "denoised")
  rowData(sce_sc3)$feature_symbol <- rownames(sce_sc3)
  counts(sce_sc3) <- as.matrix(counts(sce_sc3))
  logcounts(sce_sc3) <- as.matrix(logcounts(sce_sc3))
  sce_sc3 <- sc3_estimate_k(sce_sc3)
  K <- metadata(sce_sc3)$sc3$k_estimation
  # Note: with R >= 4.0, RStudio and Mac OS, this can fails.
  # A workaround is running
  # parallel:::setDefaultClusterOptions(setup_strategy = "sequential")
  if (default) {
    sce_sc3 <- sc3(sce_sc3, ks = K, n_cores = NCORES, rand_seed = 786907)
  } else {
    sce_sc3 <- sc3(sce_sc3, ks = K, n_cores = NCORES, rand_seed = 786907,
                   svm_num_cells = round(.1 * ncol(sce)))
  }
  sce_sc3 <- sc3_run_svm(sce_sc3, ks = K)
  sce$SC3 <- colData(sce_sc3)[, paste0("sc3_", K, "_clusters")] %>% as.factor()
})
```

```
plotTSNE(sce, colour_by = "SC3")
```



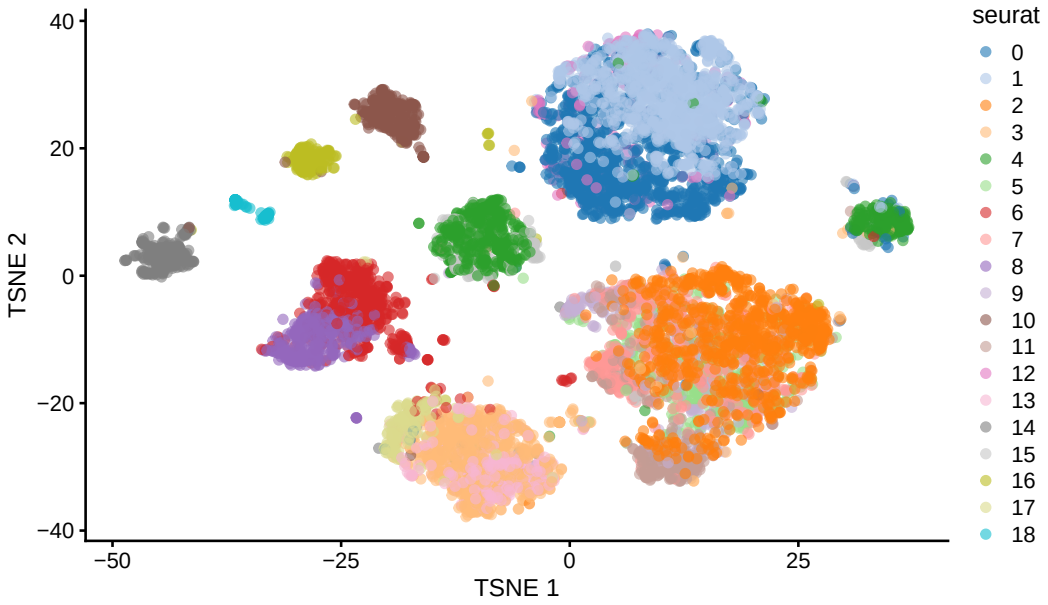
As we can see, **SC3** seems to overcluster the data, when compared either to the labels from the original publication, or to the reduced dimension representation. However, this is not a problem since **Dune** will work better on overclustered results.

3.2 Seurat

The second method we use is the clustering algorithm from the **Seurat** R package, which first construct a Shared Nearest Neighbor (SNN) Graph and then runs the Louvain algorithm on the graph to identify clusters. The SNN graph is build using a reduced dimension representation of the dataset. We first use the default, which is to use the top pcs from the normalized count matrix.

```
seurat_time <- system.time({
  se <- RunPCA(se, verbose = FALSE)
  se <- FindNeighbors(se, verbose = FALSE)
  se <- FindClusters(object = se, verbose = FALSE)
  sce$seurat <- Idents(se)
})
```

```
plotTSNE(sce, colour_by = "seurat")
```



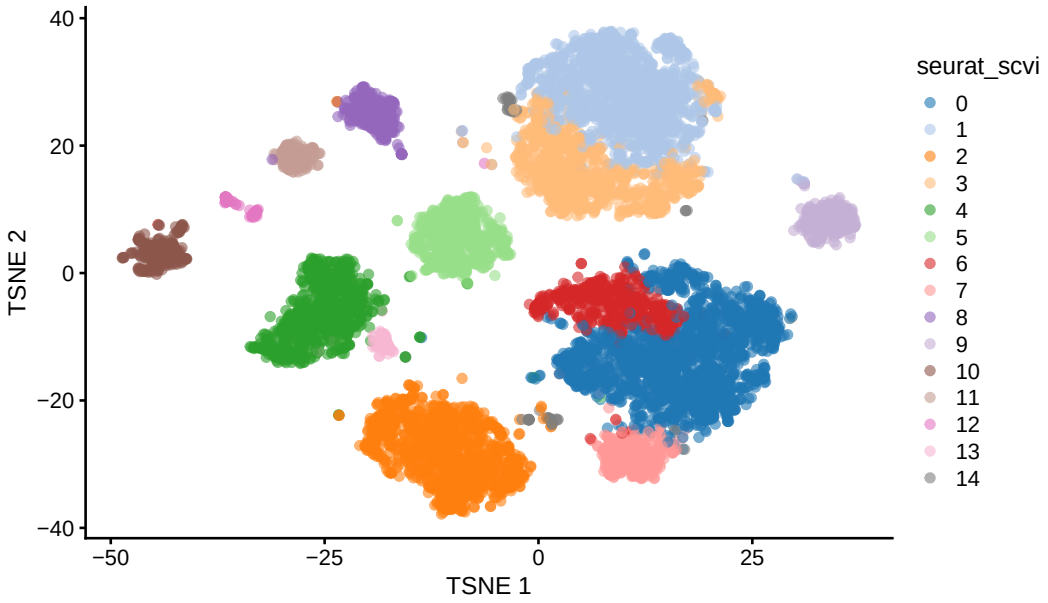
Seurat seems to perform better than **SC3** here but still seems to overpartition the data when run with the default parameters. Once again, it will not be a problem if used as input to **Dune**.

3.3 Seurat with scvi

Finally, we run the **Seurat** clustering workflow, but instead of building the SNN using the top 10 pcs, we build it using the latent space from **scvi**.

```
seurat_scvi_time <- system.time({
  seu <- as.Seurat(x = sce, counts = "counts", data = "counts")
  seu <- FindNeighbors(seu, reduction = "scvi", verbose = FALSE)
  seu <- FindClusters(object = seu, verbose = FALSE)
  sce$seurat_scvi <- Idents(seu)
})
```

```
plotTSNE(sce, colour_by = "seurat_scvi")
```



This seems to produce the best result, at least on the latent space of **scvi**, which is not suprising. It also better matches the labels from the original publication but still results in possible over-partition.

4 Dune

4.1 Running Dune

We can now run **Dune**, using the three clustering results as input. Since all clusterings seem to reflect over-partitioning of the data, **Dune** will identify the common underlying structure and polish all inputs, using the Normalized Mutual Information (NMI) as a merging criterion.

```
library(Dune)
```

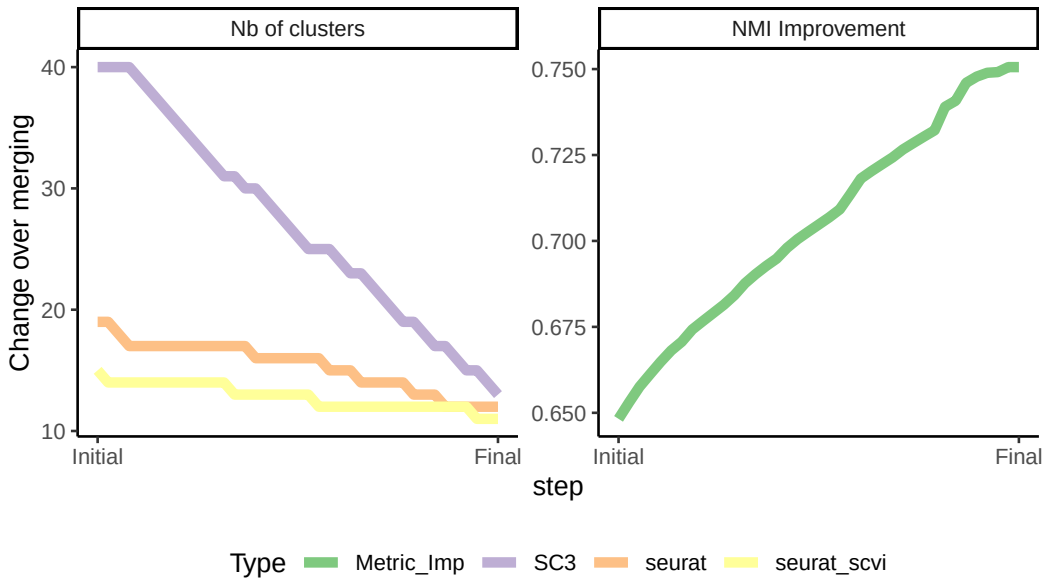
```
## Dune now uses the Normalized Mutual Informationinstead of the adjusted Rand Index. You can res
```

```
df <- colData(sce)[, c("SC3", "seurat", "seurat_scvi")] %>% as.matrix()
dune_time <- system.time(merger <- Dune(clusMat = df, metric = "NMI"))
colData(sce)[, c("SC3_final", "seurat_final", "seurat_scvi_final")] <-
  lapply(merger$currentMat, as.factor) %>% as.data.frame()
```

4.2 Vizualing the merging

We can first see how the number of clusters in each clustering set decreased as merging occurred, and how the mean NMI increased when merging.

```
NMItrend(merger) + theme(legend.position = "bottom")
```



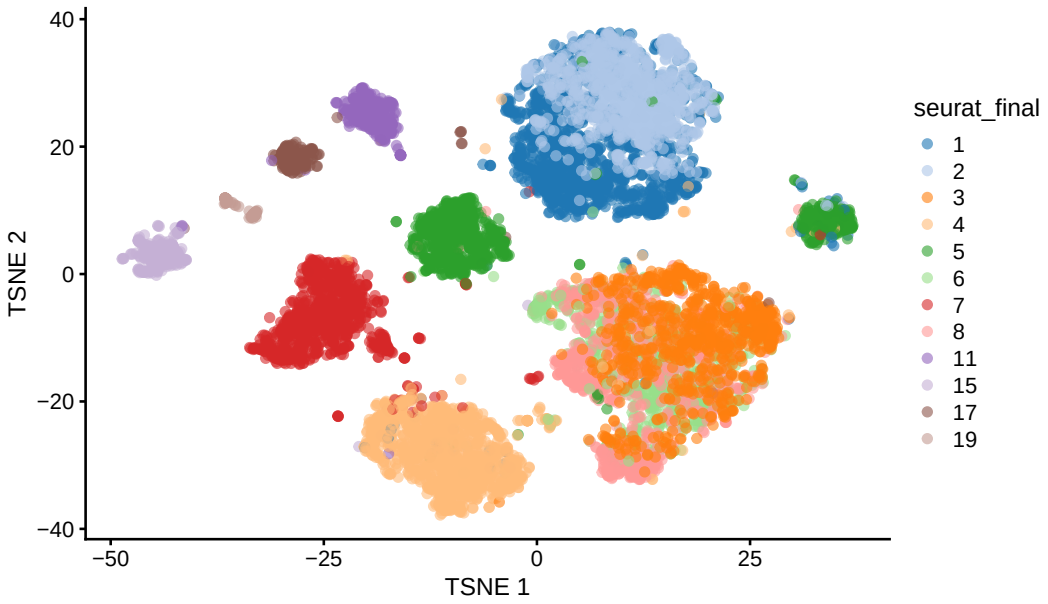
5 Picking the final clustering result to use

While **Dune** increases the concordance between the three sets of clusters, it does not pick one at the end. That choice remains up to the user. **Dune** does not seek to replace biological knowledge or other metrics used to rank clustering methods. Instead, it aims to improve all its inputs, and to lessen the impact of the selection of one set of clusters.

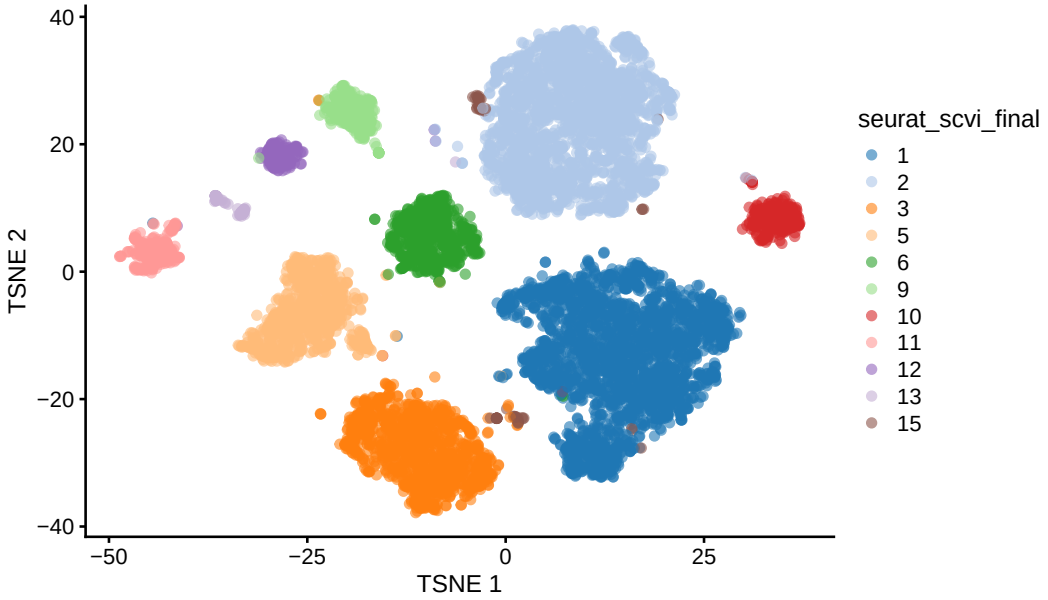
5.1 Manual selection

One common way to pick clustering results is still manual, using visualization.

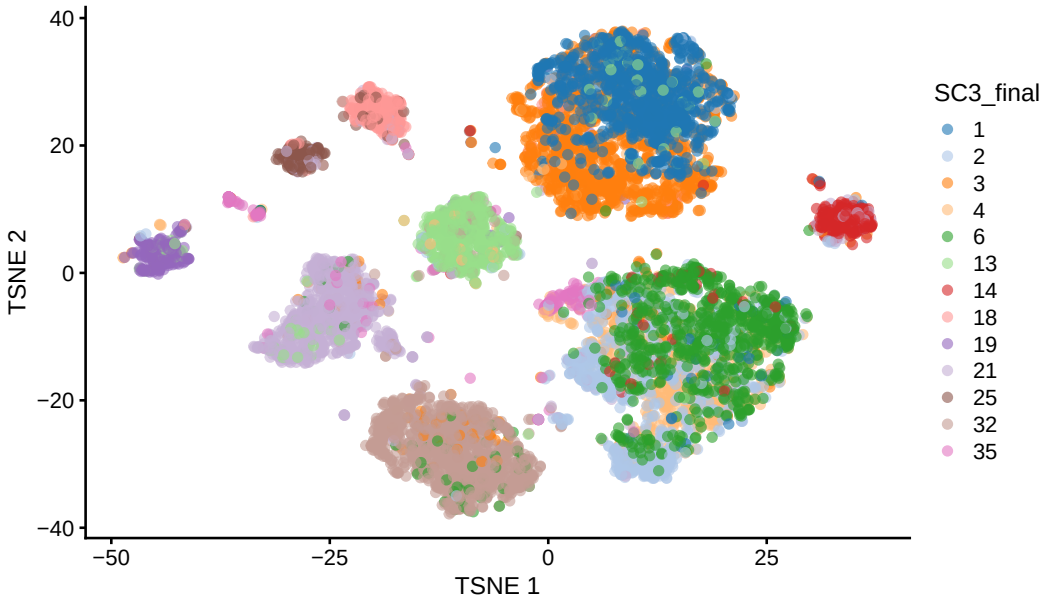
```
plotTSNE(sce, colour_by = "seurat_final")
```



```
plotTSNE(sce, colour_by = "seurat_scvi_final")
```




```
plotTSNE(sce, colour_by = "SC3_final")
```



Here, we can see that all clustering results look more consistent with the low dimensionality representation. Moreover, it clearly looks like **Seurat** using the latent space from **scvi** produces better results.

5.2 Selection based on sihouette

To provide a more quantitative selection criterion, we can rely on the average silhouette width. This is a number between -1 and 1 that quantify the quality of clustering using the distance matrix between all cells. We compute the distance on the **scvi** latent space.

```
library(cluster)
dist_mat <- dist(as.matrix(reducedDim(sce, "scvi")))
sils_init <- lapply(merger$initialMat %>% as.data.frame, function(label){
  silhouette(label, dist = dist_mat)[,3] %>%  mean()
}) %>% unlist()
sils_init
```

```
##         SC3      seurat seurat_scvi
## -0.02385097  0.05998455  0.21037407
```

This confirm the visual impression: the cluster labels from **Seurat_scvi** are clearly better on this dataset than the others before merging with **Dune**.

```
sils_final <- lapply(merger$currentMat %>% as.data.frame, function(label){
  silhouette(label, dist = dist_mat)[,3] %>%  mean()
}) %>% unlist()
sils_final
```

```
##         SC3      seurat seurat_scvi
##  0.09578247  0.15157494  0.25300568
```

For all methods, the average silhouette information increased after merging with **Dune**. Even the best method, **Seurat_scvi**, is improved by the merging. However, the ranking of methods is unchanged: consistent with the visual representation, **Seurat** using the latent space from **scvi** clearly outperforms the other two. That is the one that should be used for downstream analysis such as trajectory inference, differential expression or cell type annotation.

5.3 Comparing with the original labels

This last step is not possible on a normal analysis of a new dataset. However, here, we can see how, running all methods using default, we recover cluster labels that match closely clusters from the original publication that had require manual merging using outside biological knowledge.

```
suppressPackageStartupMessages({
  library(aricode)
  library(mclust)
})
NMI(sce$label, sce$seurat_scvi_final) %>% round(2)
```

```
## [1] 0.91
```

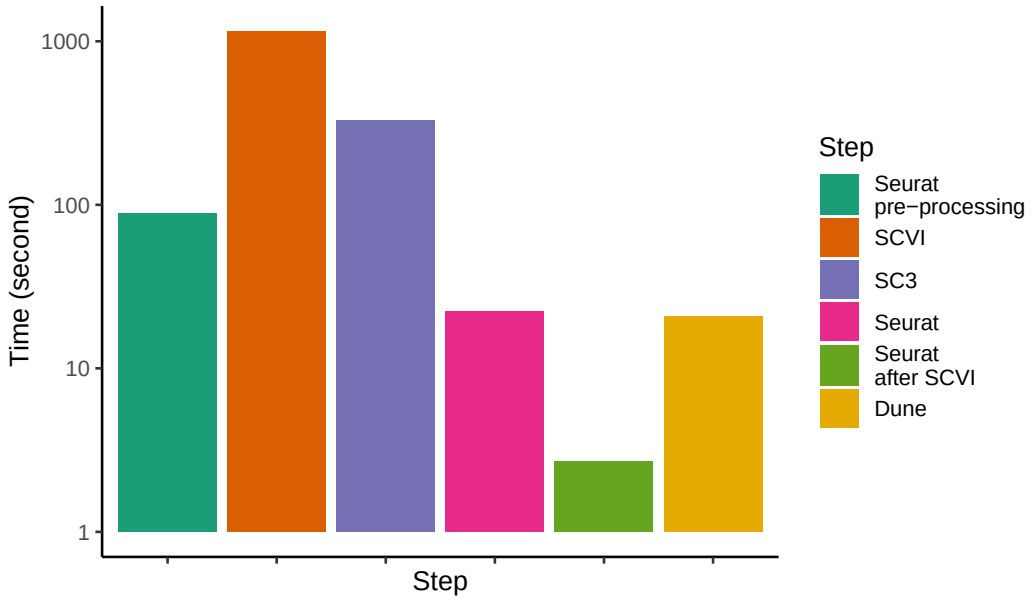
```
adjustedRandIndex(sce$label, sce$seurat_scvi_final) %>% round(2)
```

```
## [1] 0.93
```

6 Runtimes

We can also compare the runtimes of all parts of the workflow. Running **SC3** in default mode is quite slow, followed by **scvi**. Running **Dune** itself is quite quick compared to other steps. Using **Dune** in a workflow increased total runtime but not by orders of magnitudes.

```
times <- c(pre_process_time[1],
           scvi_time[1],
           sc3_time[1],
           seurat_time[1],
           seurat_scvi_time[1],
           dune_time[1])
names(times) <- c("Seurat\npre-processing",
                  "SCVI",
                  "SC3",
                  "Seurat",
                  "Seurat\nafter SCVI",
                  "Dune")
df <- data.frame(times = times,
                 Name = factor(names(times), levels = names(times)))
ggplot(df, aes(x = Name, y = times, fill = Name)) +
  geom_col() +
  theme_classic() +
  labs(x = "Step", y = "Time (second)", fill = "Step") +
  scale_fill_brewer(palette = "Dark2") +
  theme(axis.text.x = element_blank()) +
  scale_y_log10() +
  guides(col = FALSE)
```




```
sessionInfo()
```

```
## R version 4.0.3 (2020-10-10)
## Platform: x86_64-apple-darwin17.0 (64-bit)
## Running under: macOS Big Sur 10.16
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.0/Resources/lib/libRblas.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.0/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats4    parallel  stats     graphics  grDevices utils     datasets
## [8] methods   base
##
## other attached packages:
##  [1] mclust_5.4.7                aricode_1.0.0
##  [3] cluster_2.1.1               Dune_1.3.01
##  [5] SC3_1.18.0                  scater_1.18.6
##  [7] ggplot2_3.3.3               reticulate_1.18
##  [9] SeuratObject_4.0.0          Seurat_4.0.1
## [11] scRNAseq_2.4.0              stringr_1.4.0
## [13] SingleCellExperiment_1.12.0 SummarizedExperiment_1.20.0
## [15] Biobase_2.50.0              GenomicRanges_1.42.0
## [17] GenomeInfoDb_1.26.5         IRanges_2.24.1
```

```
## [19] S4Vectors_0.28.1            BiocGenerics_0.36.0
## [21] MatrixGenerics_1.2.1        matrixStats_0.58.0
## [23] knitr_1.31
##
## loaded via a namespace (and not attached):
##   [1] utf8_1.2.1                    tidyselect_1.1.0
##   [3] RSQLite_2.2.5                 AnnotationDbi_1.52.0
##   [5] htmlwidgets_1.5.3             grid_4.0.3
##   [7] BiocParallel_1.24.1           Rtsne_0.15
##   [9] munsell_0.5.0                 codetools_0.2-18
##  [11] ica_1.0-2                     future_1.21.0
##  [13] miniUI_0.1.1.1                withr_2.4.1
##  [15] colorspace_2.0-0              ROCR_1.0-11
##  [17] robustbase_0.93-7             tensor_1.5
##  [19] listenv_0.8.0                 labeling_0.4.2
##  [21] GenomeInfoDbData_1.2.4        polyclip_1.10-0
##  [23] farver_2.1.0                  pheatmap_1.0.12
##  [25] bit64_4.0.5                   parallelly_1.24.0
##  [27] vctrs_0.3.7                   generics_0.1.0
##  [29] xfun_0.22                     BiocFileCache_1.14.0
##  [31] doParallel_1.0.16             R6_2.5.0
##  [33] ggbeeswarm_0.6.0              rsvd_1.0.3
##  [35] AnnotationFilter_1.14.0       bitops_1.0-6
##  [37] spatstat.utils_2.1-0          cachem_1.0.4
##  [39] DelayedArray_0.16.3           assertthat_0.2.1
##  [41] promises_1.2.0.1              scales_1.1.1
##  [43] beeswarm_0.3.1                gtable_0.3.0
##  [45] beachmat_2.6.4                globals_0.14.0
##  [47] goftest_1.2-2                 ensembldb_2.14.0
##  [49] rlang_0.4.10                  splines_4.0.3
##  [51] rtracklayer_1.50.0            lazyeval_0.2.2
##  [53] spatstat.geom_2.0-1           BiocManager_1.30.12
##  [55] yaml_2.2.1                    reshape2_1.4.4
##  [57] abind_1.4-5                   GenomicFeatures_1.42.3
##  [59] httpuv_1.5.5                  tools_4.0.3
##  [61] ellipsis_0.3.1                spatstat.core_2.0-0
##  [63] RColorBrewer_1.1-2            proxy_0.4-25
##  [65] ggridges_0.5.3                Rcpp_1.0.6
##  [67] plyr_1.8.6                    sparseMatrixStats_1.2.1
##  [69] progress_1.2.2                zlibbioc_1.36.0
##  [71] purrr_0.3.4                   RCurl_1.98-1.3
##  [73] prettyunits_1.1.1             rpart_4.1-15
##  [75] openssl_1.4.3                 deldir_0.2-10
##  [77] viridis_0.5.1                 pbapply_1.4-3
##  [79] cowplot_1.1.1                 zoo_1.8-9
##  [81] ggrepel_0.9.1                 magrittr_2.0.1
##  [83] magick_2.7.1                  data.table_1.14.0
##  [85] scattermore_0.7               lmtest_0.9-38
```

```
##  [87] RANN_2.6.1                    mvtnorm_1.1-1
##  [89] ProtGenerics_1.22.0           fitdistrplus_1.1-3
##  [91] hms_1.0.0                     patchwork_1.1.1
##  [93] mime_0.10                     evaluate_0.14
##  [95] xtable_1.8-4                  XML_3.99-0.6
##  [97] gridExtra_2.3                 compiler_4.0.3
##  [99] biomaRt_2.46.3                tibble_3.1.0
## [101] KernSmooth_2.23-18            crayon_1.4.1
## [103] htmltools_0.5.1.1             pcaPP_1.9-73
## [105] mgcv_1.8-34                   later_1.1.0.1
## [107] rrcov_1.5-5                   tidyr_1.1.3
## [109] DBI_1.1.1                     tweenr_1.0.2
## [111] ExperimentHub_1.16.0          WriteXLS_6.3.0
## [113] dbplyr_2.1.1                  MASS_7.3-53.1
## [115] rappdirs_0.3.3                Matrix_1.3-2
## [117] igraph_1.2.6                  pkgconfig_2.0.3
## [119] GenomicAlignments_1.26.0      plotly_4.9.3
## [121] scuttle_1.0.4                 spatstat.sparse_2.0-0
## [123] foreach_1.5.1                 xml2_1.3.2
## [125] vipor_0.4.5                   rngtools_1.5
## [127] XVector_0.30.0                doRNG_1.8.2
## [129] digest_0.6.27                 sctransform_0.3.2
## [131] RcppAnnoy_0.0.18              spatstat.data_2.1-0
## [133] Biostrings_2.58.0             rmarkdown_2.7
## [135] leiden_0.3.7                  uwot_0.1.10
## [137] DelayedMatrixStats_1.12.3     curl_4.3
## [139] shiny_1.6.0                   Rsamtools_2.6.0
## [141] lifecycle_1.0.0               nlme_3.1-152
## [143] jsonlite_1.7.2                BiocNeighbors_1.8.2
## [145] viridisLite_0.3.0             askpass_1.1
## [147] fansi_0.4.2                   pillar_1.5.1
## [149] lattice_0.20-41               DEoptimR_1.0-8
## [151] fastmap_1.1.0                 httr_1.4.2
## [153] survival_3.2-10               gganimate_1.0.7
## [155] interactiveDisplayBase_1.28.0 glue_1.4.2
## [157] iterators_1.0.13              png_0.1-7
## [159] BiocVersion_3.12.0            bit_4.0.4
## [161] class_7.3-18                  stringi_1.5.3
## [163] blob_1.2.1                    BiocSingular_1.6.0
## [165] AnnotationHub_2.22.0          memoise_2.0.0
## [167] dplyr_1.0.5                   e1071_1.7-6
## [169] irlba_2.3.3                   future.apply_1.7.0
```

References

Baron, Maayan, Adrian Veres, Samuel L. Wolock, Aubrey L. Faust, Renaud Gaujoux, Amedeo Vetere, Jennifer Hyoje Ryu, et al. 2016. “A Single-Cell Transcriptomic Map of

the Human and Mouse Pancreas Reveals Inter- and Intra-cell Population Structure." *Cell Systems* 3 (4): 346–360.e4. https://doi.org/10.1016/j.cels.2016.08.011.

Kiselev, Vladimir Yu, Kristina Kirschner, Michael T Schaub, Tallulah Andrews, Andrew Yiu, Tamir Chandra, Kedar N Natarajan, et al. 2017. "SC3: Consensus clustering of single-cell RNA-seq data." *Nature Methods* 14 (5): 483–86. https://doi.org/10.1038/nmeth.4236.

Krijthe, Jesse H. 2015. *Rtsne: T-Distributed Stochastic Neighbor Embedding Using Barnes-Hut Implementation*. https://github.com/jkrijthe/Rtsne.

Lopez, Romain, Jeffrey Regier, Michael B. Cole, Michael I. Jordan, and Nir Yosef. 2018. "Deep generative modeling for single-cell transcriptomics." *Nature Methods* 15 (12): 1053–8. https://doi.org/10.1038/s41592-018-0229-2.

Stuart, Tim, Andrew Butler, Paul Hoffman, Christoph Hafemeister, Efthymia Papalexi, William M Mauck, Yuhan Hao, Marlon Stoeckius, Peter Smibert, and Rahul Satija. 2019. "Comprehensive Integration of Single-Cell Data." *Cell* 177 (7): 1888–1902.e21. https://doi.org/10.1016/j.cell.2019.05.031.

van der Maaten, L. J. P. 2014. "Accelerating T-Sne Using Tree-Based Algorithms." *Journal of Machine Learning Research* 15: 3221–45.

van der Maaten, L. J. P., and G. E. Hinton. 2008. "Visualizing High-Dimensional Data Using T-Sne." *Journal of Machine Learning Research* 9: 2579–2605.

Appendix C

Supplementary Figures and Tables for chapter 3

Table C.1: *Mouse olfactory epithelium dataset.* The top 20 significant GO sets for the top 250 genes when assessing global differential expression between the progenitor and differentiated cell populations using the tradeSeq `startVsEndTest` procedure. The "Overlap" column records the number of genes, out of the 250 top genes, that are included in a particular gene set. The significance of a gene set is measured by a q-value, obtained by assessing the significance of the overlap of the DE genes with the gene set, as obtained from the Molecular Signatures Database v6.2 (http://software.broadinstitute.org/gsea/msigdb).

Gene set	**Overlap**	**Size**	**q-value**
1. neurogenesis	46	1402	1.31E-21
2. response to external stimulus	51	1821	1.41E-21
3. tissue development	47	1518	1.41E-21
4. cellular response to organic substance	49	1848	7.41E-20
5. regulation of multicellular organismal development	46	1672	3.5E-19
6. neuron differentiation	35	874	3.69E-19
7. response to endogenous stimulus	43	1450	4.07E-19
8. regulation of cell differentiation	43	1492	1.05E-18
9. regulation of cellular component movement	32	771	5.84E-18
10. cell development	41	1426	8.94E-18
11. cellular response to endogenous stimulus	35	1008	1.91E-17
12. regulation of intracellular signal transduction	43	1656	3.5E-17
13. organ morphogenesis	31	841	4.86E-16
14. negative regulation of response to stimulus	38	1360	4.86E-16
15. positive regulation of cell communication	40	1532	5.3E-16
16. regulation of phosphorus metabolic process	40	1618	3.24E-15
17. response to oxygen containing compound	37	1381	4.61E-15
18. circulatory system development	29	788	5.19E-15
19. epithelium development	31	945	8.78E-15
20. locomotion	33	1114	1.44E-14

Table C.2: *Mouse olfactory epithelium dataset.* The top 20 significant GO sets for the 827 genes that were found to be significant in all pairwise comparisons between the three trajectories using the tradeSeq `patternTest` procedure. The "Overlap" column records the number of genes, out of the 827 top genes, that are included in a particular gene set. The significance of a gene set is measured by a q-value, obtained by assessing the significance of the overlap of the DE genes with the gene set, as obtained from the Molecular Signatures Database v6.2 (http://software.broadinstitute.org/gsea/msigdb).

Gene set	**Overlap**	**Size**	**q-value**
1. cell cycle	121	1316	3.97E-57
2. cell cycle process	108	1081	2.66E-54
3. mitotic cell cycle	86	766	1.16E-46
4. establishment of localization in cell	108	1676	2.37E-36
5. cell division	59	460	1.33E-34
6. chromosome organization	79	1009	1.3E-31
7. cellular response to stress	97	1565	3.59E-31
8. organonitrogen compound metabolic process	104	1796	3.59E-31
9. regulation of cell cycle	75	949	2.53E-30
10. organelle fission	54	496	3.91E-28
11. cytoskeleton organization	68	838	4.64E-28
12. microtubule based process	55	522	4.9E-28
13. cellular catabolic process	84	1322	1.31E-27
14. regulation of cell differentiation	89	1492	2.03E-27
15. neurogenesis	86	1402	2.68E-27
16. catabolic process	97	1773	3.09E-27
17. mitotic nuclear division	46	361	6.11E-27
18. positive regulation of molecular function	96	1791	2.65E-26
19. protein complex subunit organization	88	1527	3.81E-26
20. positive regulation of gene expression	94	1733	3.94E-26

Table C.3: *Mouse olfactory epithelium dataset.* The top 20 significant GO sets based on the unique 1,994 genes that were only discovered with the ZINB-tradeSeq analysis, and not the ZINB-edgeR analysis, when comparing mean expression between the endpoints of the lineages using the tradeSeq `diffEndTest` procedure. The "Overlap" column records the number of genes, out of the 1,994 top genes, that are included in a particular gene set. The significance of a gene set is measured by a q-value, obtained by assessing the significance of the overlap of the DE genes with the gene set, as obtained from the Molecular Signatures Database v6.2 (http://software.broadinstitute.org/gsea/msigdb).

Gene set	**Overlap**	**Size**	**q-value**
1. phosphate containing compound metabolic process	225	1977	3.42E-53
2. protein localization	211	1805	8.95E-52
3. regulation of anatomical structure morphogenesis	154	1021	5.52E-51
4. positive regulation of molecular function	207	1791	3.41E-50
5. positive regulation of catalytic activity	186	1518	1.59E-48
6. regulation of multicellular organismal development	193	1672	1.34E-46
7. single organism biosynthetic process	169	1340	1.29E-45
8. lipid metabolic process	153	1158	1.35E-43
9. small molecule metabolic process	193	1767	3.5E-43
10. positive regulation of biosynthetic process	194	1805	1.95E-42
11. positive regulation of gene expression	189	1733	3.31E-42
12. regulation of protein modification process	187	1710	6.25E-42
13. positive regulation of response to stimulus	199	1929	4.54E-41
14. cellular macromolecule localization	154	1234	5.03E-41
15. regulation of phosphorus metabolic process	178	1618	3.29E-40
16. catabolic process	187	1773	7.94E-40
17. intracellular signal transduction	174	1572	1.2E-39
18. regulation of response to stress	167	1468	1.59E-39
19. regulation of transcription from rna polymerase ii promoter	187	1784	1.59E-39
20. tissue development	170	1518	2.08E-39

Table C.4: *Mouse olfactory epithelium dataset.* The top 20 significant GO sets based on the 84 transcription factors that were discovered with `earlyDETest` around the branching of the OE trajectory. The "Overlap" column records the number of genes, out of the 1,994 top genes, that are included in a particular gene set. The significance of a gene set is measured by a q-value, obtained by assessing the significance of the overlap of the DE genes with the gene set, as obtained from the Molecular Signatures Database v6.2 (http://software.broadinstitute.org/gsea/msigdb).

Gene set	**Overlap**	**Size**	**q-value**
1. regulation of transcription from RNA Pol. II promoter	68	1784	1.01E-77
2 . positive regulation of gene expression	57	1733	6.09E-58
3. positive regulation of biosynthetic process	57	1805	4.12E-57
4. positive regulation of transcription from RNA Pol. II promoter	48	1004	1.06E-54
5. transcription from RNA Pol. II promoter	40	724	1.01E-46
6. negative regulation of transcription from RNA Pol. II promoter	34	740	3.94E-36
7. negative regulation of gene expression	40	1493	2.33E-34
8. negative regulation of nitrogen compound metabolic process	40	1517	3.8E-34
9. tissue development	33	1518	1.42E-24
10. epithelium development	25	945	7.63E-20
11. muscle structure development	19	432	1.24E-18
12. epithelial cell differentiation	19	495	1.46E-17
13. regulation of cell differentiation	27	1492	1.55E-17
14. cell fate commitment	15	227	4.56E-17
15. embryo development	22	894	1.18E-16
16. cell development	25	1426	8.58E-16
17. neuron differentiation	20	874	2.14E-14
18. neurogenesis	23	1402	9.12E-14
19. positive regulation of cell differentiation	19	823	1.07E-13
20. positive regulation of developmental process	21	1142	2.11E-13

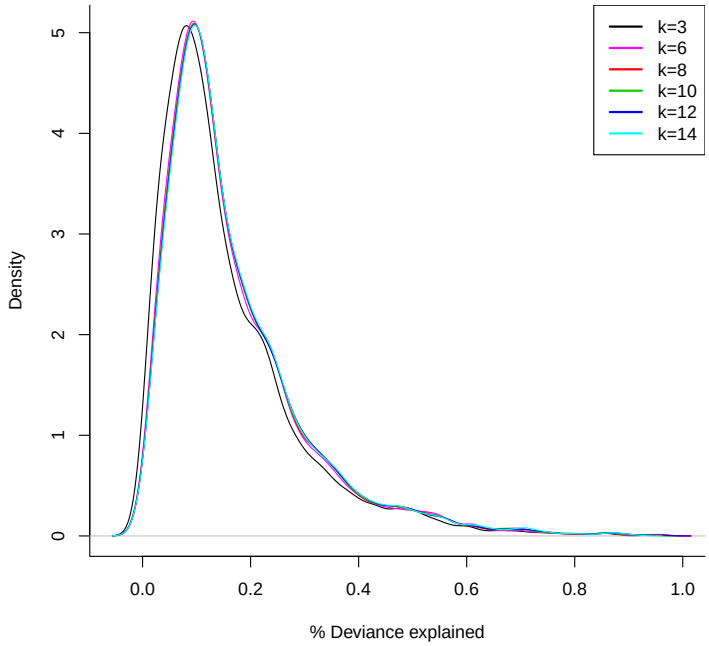


Figure C.1: *Mouse bone marrow dataset: The NB-GAM is robust to the number of knots k.* Gaussian kernel density plot of the percentage of deviance explained by the NB-GAM applied to each of the genes in the dataset from Paul et al. [108], with number of knots k ranging from 3 to 14. The distributions are nearly identical for the different numbers of knots, except for 3 knots, suggesting we might want to select more than 3 knots for this dataset.

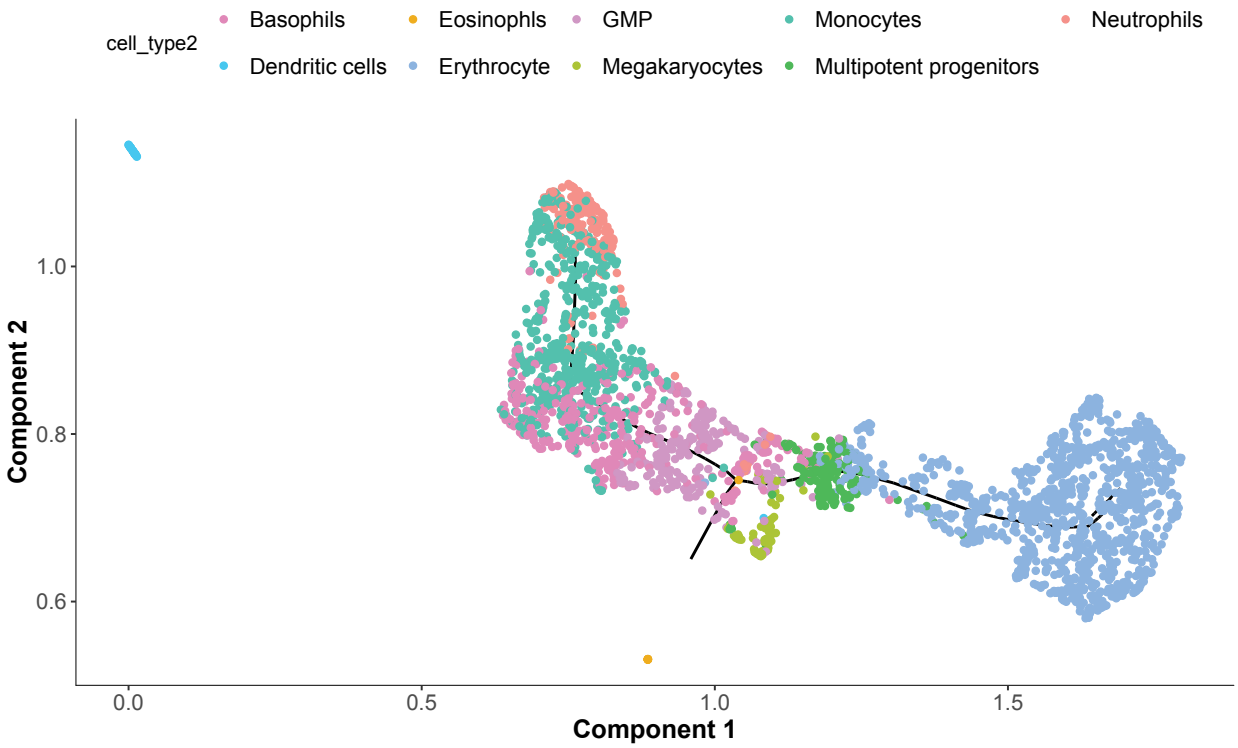


Figure C.2: *Mouse bone marrow dataset: Outlying dendritic cells and eosinophils in UMAP space for TI with* Monocle 3.

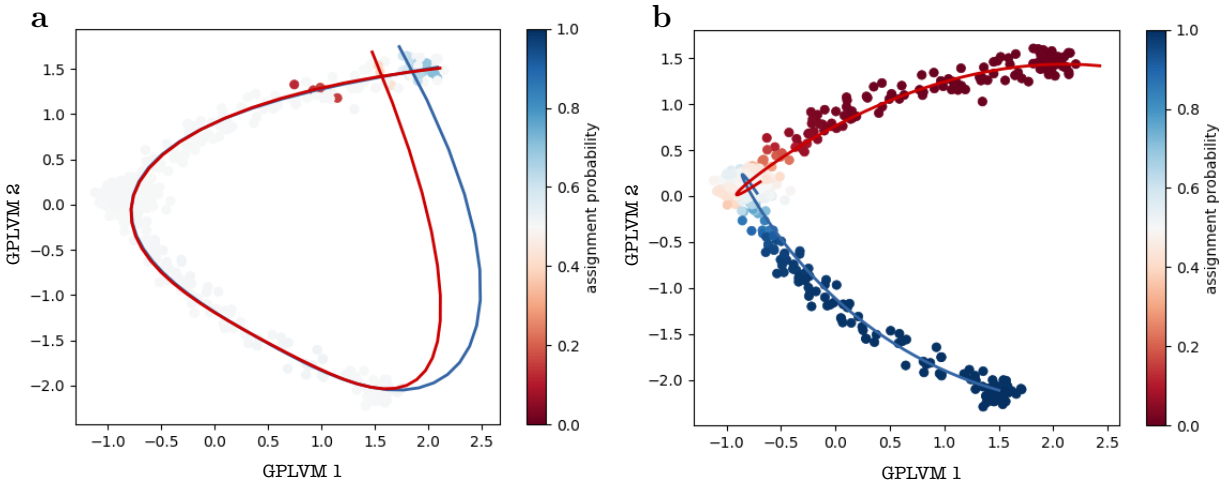


Figure C.3: *Bifurcating simulation scenario: GPfates only recovers meaningful trajectories if the true pseudotime is provided as input.* Example of a bifurcating dataset from the dyn-verse framework. The dataset is represented in low-dimensional space using Gaussian latent variable models as implemented in GPfates. Cells are colored according to their assignment probability to the blue lineage. Trajectories inferred by GPfates are shown when **(a)** pseudotime is estimated by GPfates and **(b)** true pseudotime is provided as input to GPfates.

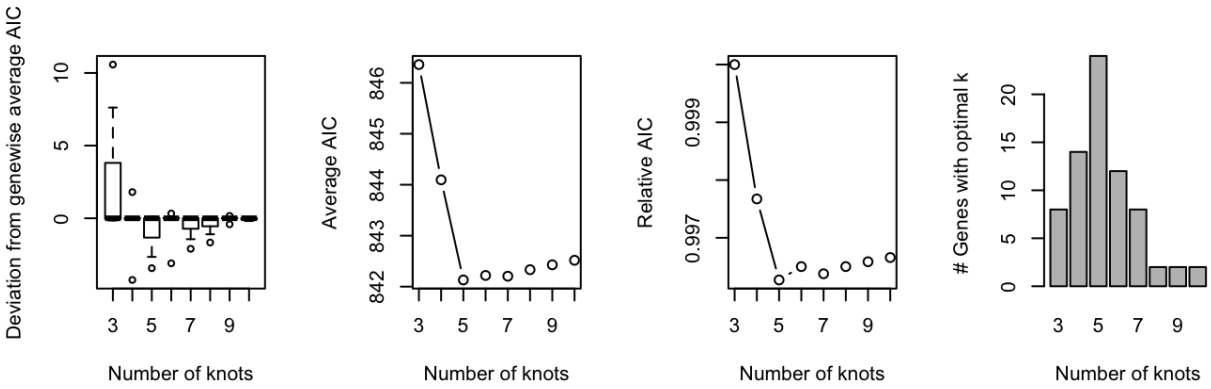


Figure C.4: *Cyclic simulation scenario: Selecting the optimal number of knots k using the AIC.* Selecting the optimal number of knots, $k \in \{3, \ldots, 10\}$, using the Akaike information criterion (AIC) for a random subset of 250 genes, as implemented in the `evaluateK` function in tradeSeq. The left panel shows boxplots (center line, median; box limits, upper and lower quartiles; whiskers, 1.5× interquartile range) of the differences in AIC value with respect to the gene-wise average AIC for the range of k. The middle panels show the evolution of the average AIC (second panel) and relative AIC (third panel) across k. The relative AIC is defined as the relative change with respect to the average AIC at $k = 3$. The barplot in the right panel shows the number of genes which achieve their lowest AIC value for a given k. Here, only genes for which the AIC value varied substantially enough across k (i.e., range in AIC greater than 2) are considered.

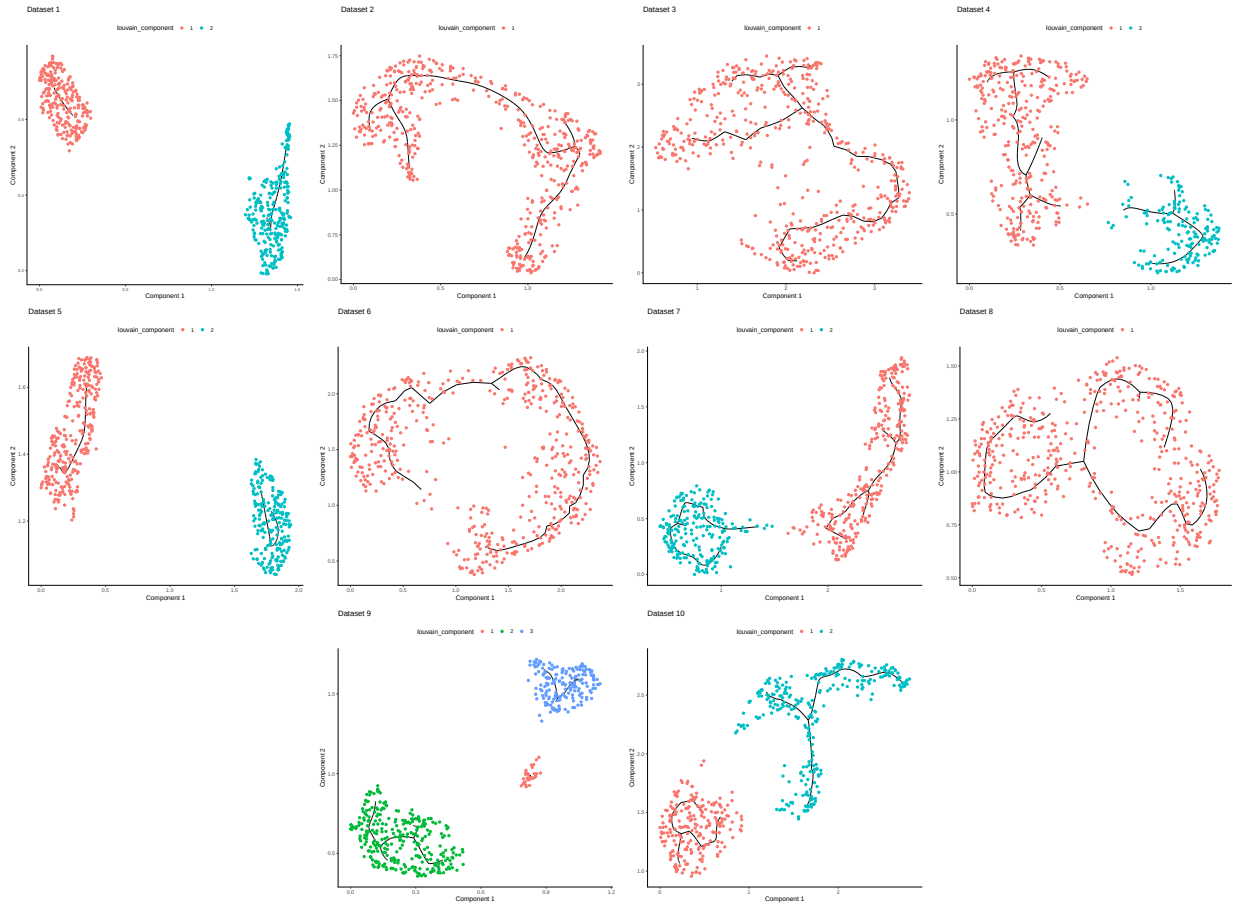


Figure C.5: *Cyclic simulation scenario: Monocle 3 inferred trajectories for each of the* 10 *simulated datasets.* The first two components from UMAP dimensionality reduction, as implemented in Monocle 3, are plotted along with the Monocle 3 inferred trajectories. Cells are colored according to a Louvain clustering implemented in Monocle 3. Monocle 3 often fails to recover the cyclic pattern.

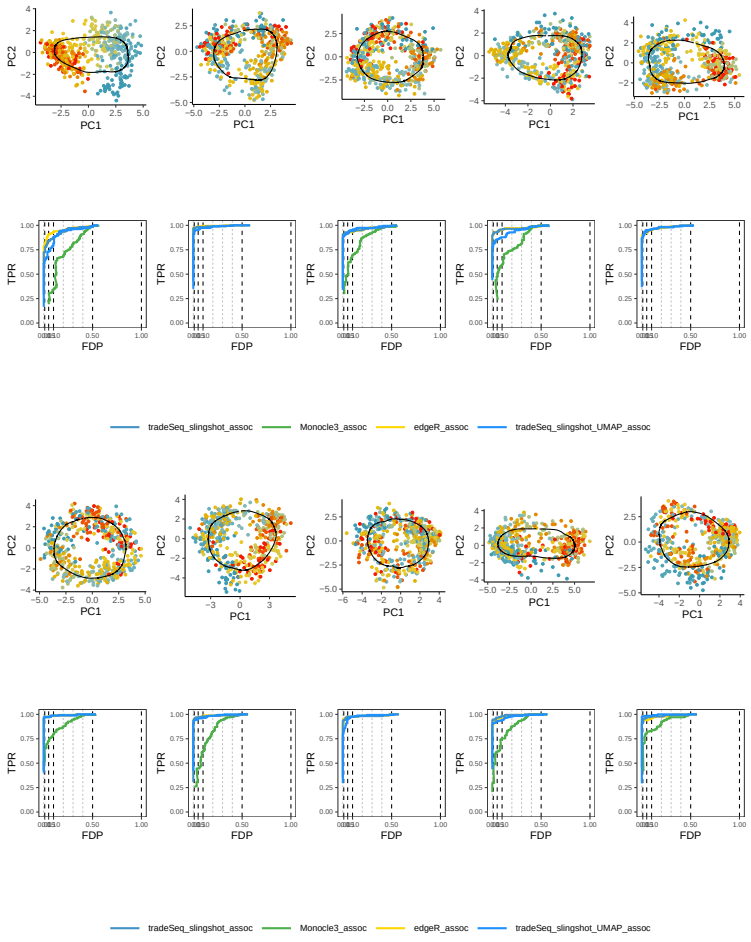


Figure C.6: *Cyclic simulation scenario: PCA plots with* slingshot *inferred trajectory and FDP-TPR performance curves for trajectory-based differential expression analysis for each of the* 10 *simulated datasets.* Monocle 3 *errored on three datasets.* edgeR_assoc *is the* edgeR*-based version of the* `associationTest`*.*

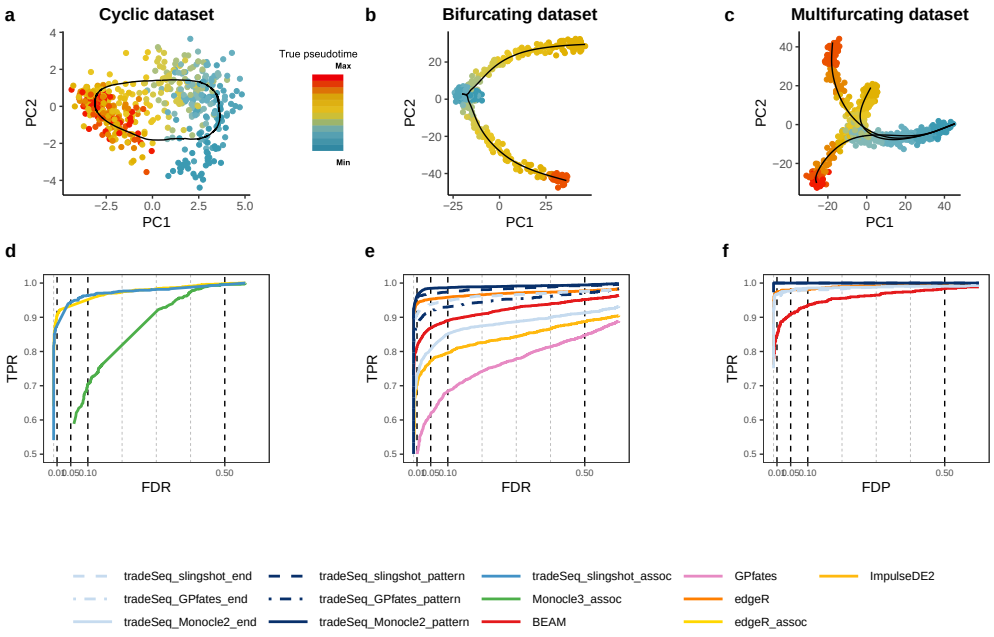


Figure C.7: *Simulation study results, including edgeR-based* `associationTest` *for the cyclic scenario.* PCA plots for the **(a)** cyclic, **(b)** bifurcating, and **(c)** multifurcating simulated trajectories. The plotting symbol for each cell is colored according to its true pseudotime; trajectories (in black) were inferred by princurve in (a) and slingshot in (b) and (c). **(d-f)** Scatterplot of the true positive rate (TPR) vs. the false discovery rate (FDR) or false discovery proportion (FDP) for various DE methods applied to the simulated datasets. Panel (d) displays the average performance curves of DE methods across seven out of 10 cyclic datasets for which all DE methods worked (Monocle 3 errored on three datasets). The `associationTest` from tradeSeq has superior performance for discovering genes whose expression is associated with pseudotime, as compared to Monocle 3. When investigating differential expression between lineages of a trajectory, the `patternTest` of tradeSeq consistently outperforms the `diffEndTest` across all three TI methods, since it is capable of comparing expression across entire lineages. Panel (e) displays the average performance curves across the three bifurcating datasets where all TI methods recovered the correct topology. Here, all tradeSeq `patternTest` workflows, tradeSeq_slingshot_end, and edgeR have similar performance and all are superior to BEAM, ImpulseDE2, and GPfates. Note that the performance of tradeSeq_Monocle2_end deteriorates as compared to tradeSeq_slingshot_end; the curve for tradeSeq_GPfates_end is not visible in this panel due to its low performance. For the multifurcating dataset of panel (f), tradeSeq_slingshot has the highest performance, closely followed by tradeSeq_Monocle2 and edgeR. Source data are provided as a Source Data file.

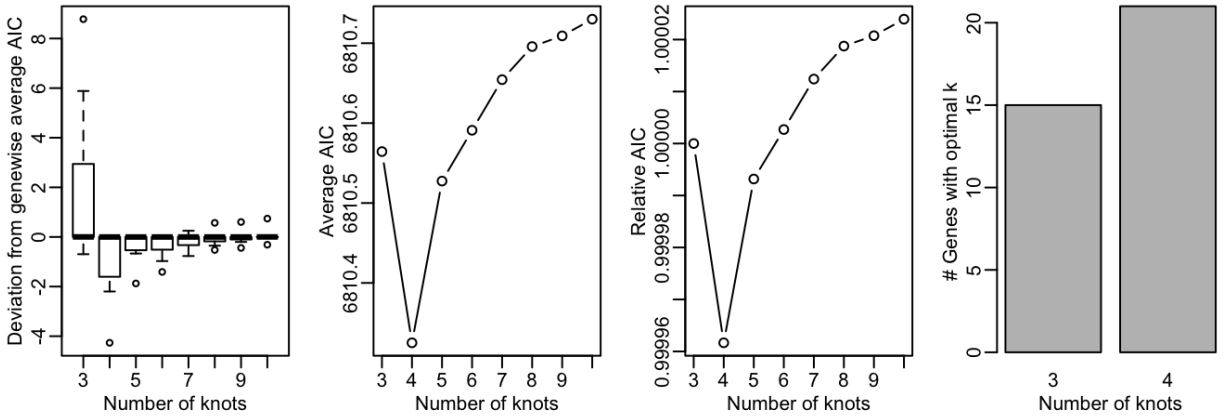


Figure C.8: *Bifurcating simulation scenario: Selecting the optimal number of knots k using the AIC.* Selecting the optimal number of knots, $k \in \{3, \ldots, 10\}$, using the AIC for a random subset of 250 genes, as implemented in the `evaluateK` function in tradeSeq. The left panel shows boxplots (center line, median; box limits, upper and lower quartiles; whiskers, 1.5× interquartile range) of the differences in AIC value with respect to the gene-wise average AIC for the range of k. The middle panels show the evolution of the average AIC (second panel) and relative AIC (third panel) across k. The relative AIC is defined as the relative change with respect to the average AIC at $k = 3$. The barplot in the right panel shows the number of genes which achieve their lowest AIC value for a given k. Here, only genes for which the AIC value varied substantially enough across k (i.e., range in AIC greater than 2) are considered.

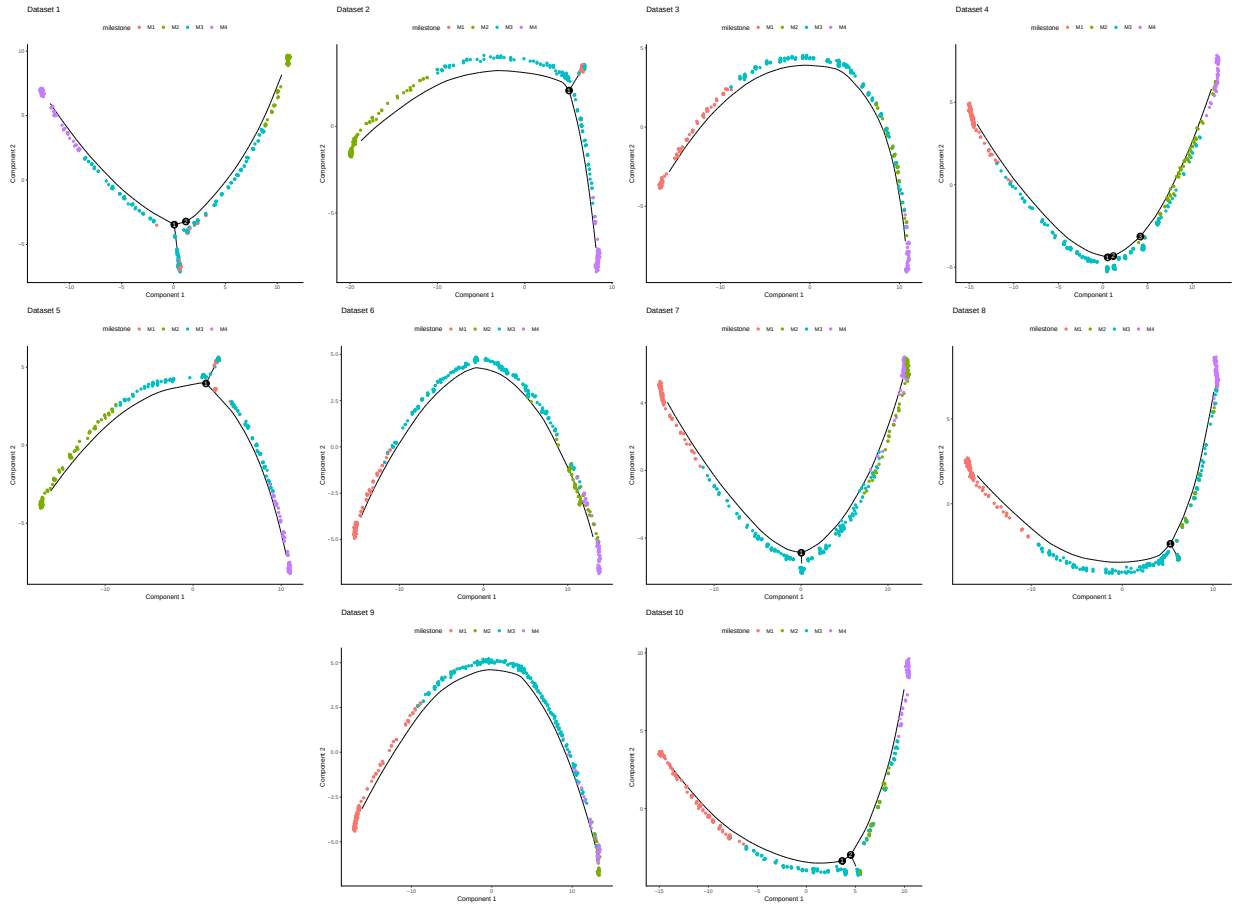


Figure C.9: *Bifurcating simulation scenario: Monocle 2 inferred trajectories for each of the 10 the simulated datasets.* Cells are plotted in two-dimensional space using DDRTree dimensionality reduction [115]. The simulated trajectory starts at milestone 1 and then continues into milestone 3, generating the two lineages that consist of milestone 2 and milestone 4. The trajectory is correctly recovered in, for example, Dataset 1 (top left panel). Dataset 4, on the other hand, wrongly assigns milestone 2 and milestone 4 to the same lineage, hence failing to recover the true bifurcation point.

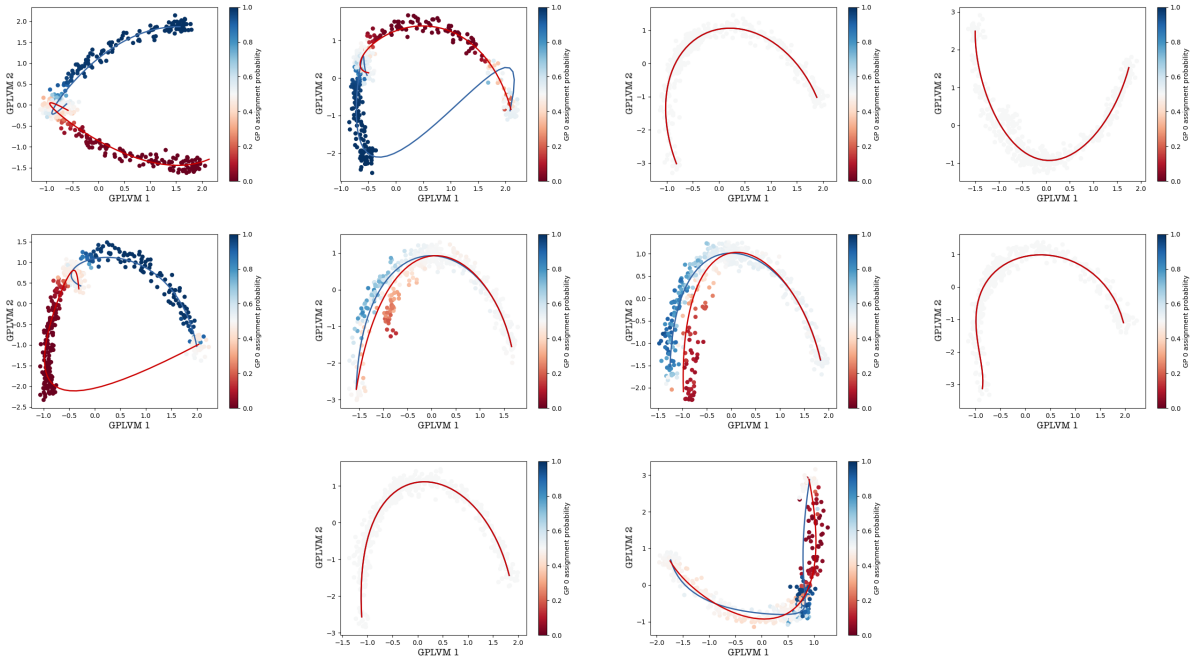


Figure C.10: *Bifurcating simulation scenario: GPfates inferred trajectories for each of the 10 simulated datasets.* Two-dimensional representation of the datasets for the bifurcating simulation scenario (dynverse toolbox) using Gaussian latent variable models, as implemented in GPfates. Cells are colored according to their assignment probability to the blue lineage.

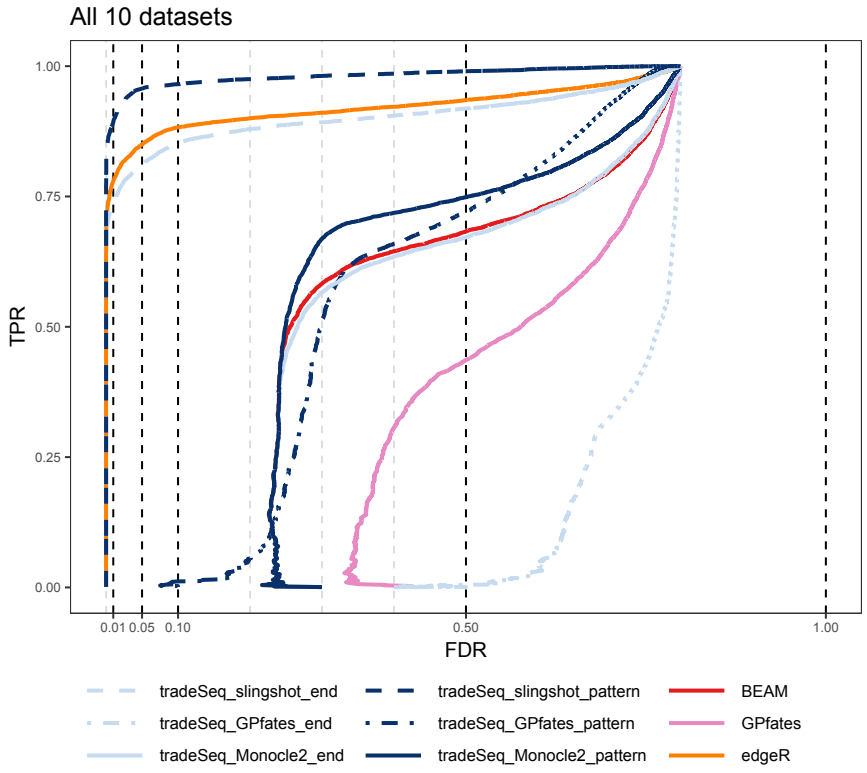


Figure C.11: *Bifurcating simulation scenario: Mean FDR-TPR performance curves for trajectory-based differential expression analysis across all* 10 *simulated datasets.* ImpulseDE2 is not plotted since we were unable to run it on several datasets.

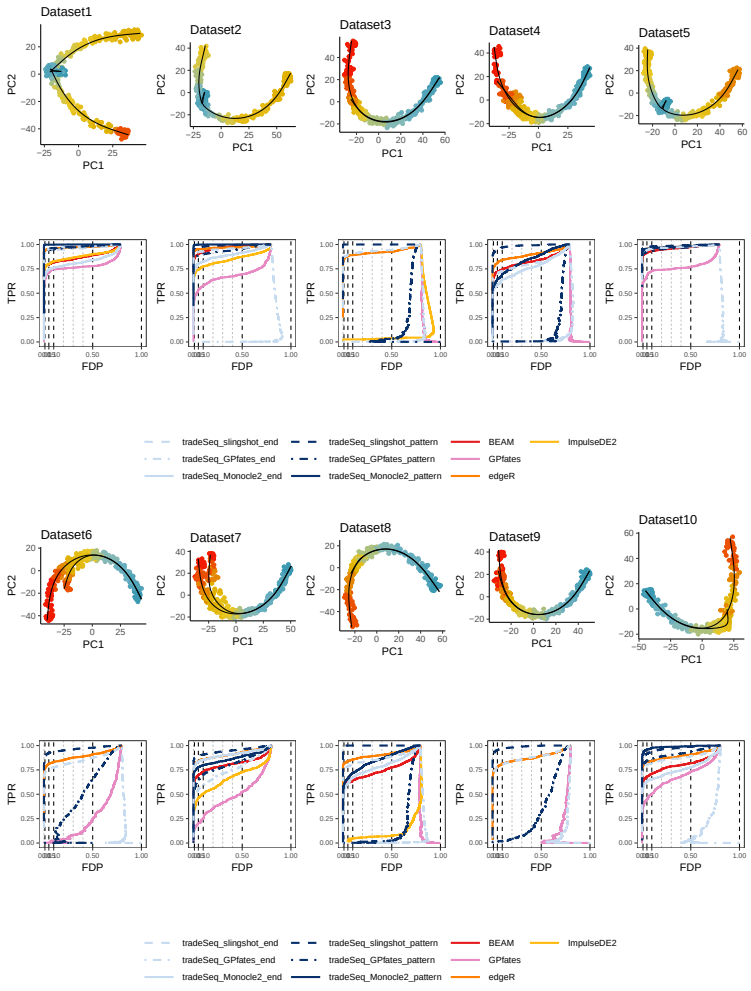


Figure C.12: *Bifurcating simulation scenario: FDP-TPR performance curves for trajectory-based differential expression analysis for each of the* 10 *simulated datasets.* Note that the BEAM and tradeSeq_Monocle2 methods are not plotted for Datasets 3, 6, and 9, since Monocle2 failed to discover a branching trajectory for those datasets. We were unable to run ImpulseDE2 on datasets $4, 5, 6, 9$, and 10 due to errors.

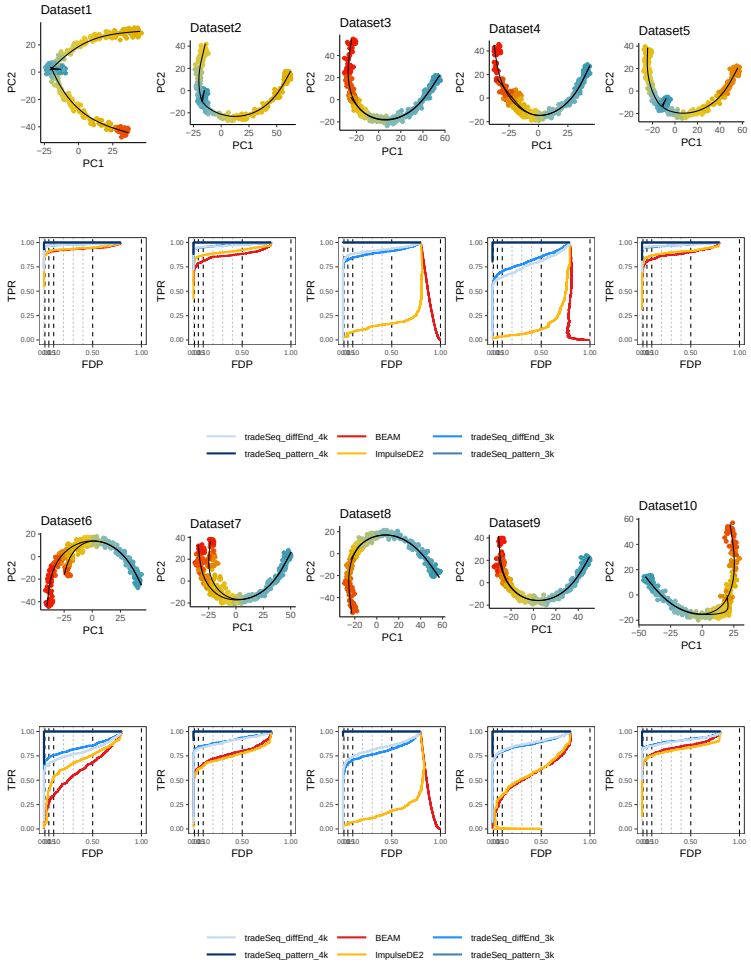


Figure C.13: *Bifurcating simulation scenario: FDP-TPR performance curves for trajectory-based differential expression analysis based on the simulation ground truth.* To allow a comparison with the BEAM approach, which fits smoothers using 3 knots, we fitted the tradeSeq NB-GAM once with 3 knots and once with 4 knots. We found the latter to provide an optimal fit in terms of AIC. The tradeSeq `patternTest` is unaffected by the number of knots, hence the performance curves overlap. tradeSeq consistently outperforms both BEAM and ImpulseDE2 in all datasets. Note that we did not include the GPfates method in this evaluation, since we were unable to provide the simulation ground truth as input to the method.

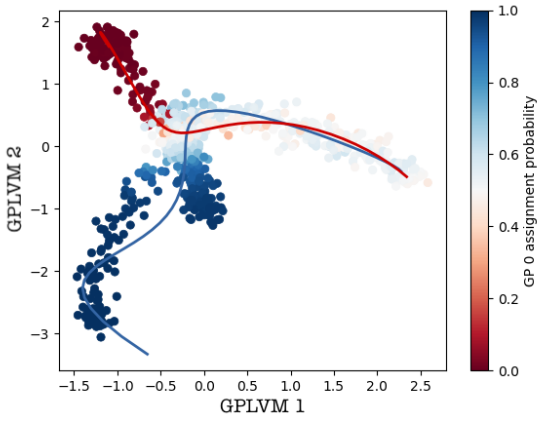


Figure C.14: *Multifurcating simulation scenario: GPfates inferred trajectory on one simulated dataset.* Two-dimensional representation of the dataset for the multifurcating simulation scenario using Gaussian latent variable models, as implemented in GPfates. Cells are colored according to their assignment probability to the blue lineage.

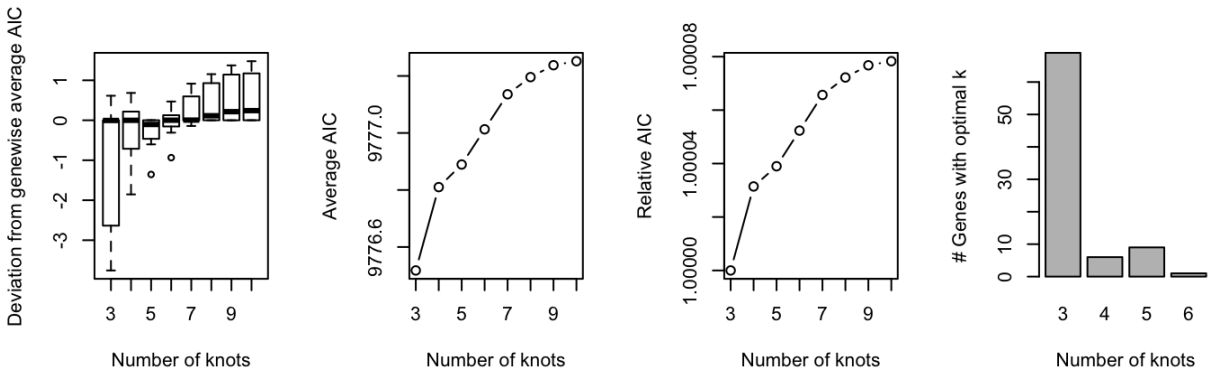


Figure C.15: *Multifurcating simulation scenario: Selecting the optimal number of knots k using the AIC.* Selecting the optimal number of knots, $k \in \{3, \ldots, 10\}$, using the AIC for a random subset of 500 genes, as implemented in the `evaluateK` function in tradeSeq. The left panel shows boxplots (center line, median; box limits, upper and lower quartiles; whiskers, 1.5× interquartile range) of the differences in AIC value with respect to the gene-wise average AIC for the range of k. The middle panels show the evolution of the average AIC (second panel) and relative AIC (third panel) across k. The relative AIC is defined as the relative change with respect to the average AIC at $k = 3$. The barplot in the right panel shows the number of genes which achieve their lowest AIC value for a given k. Here, only genes for which the AIC value varied substantially enough across k (i.e., range in AIC greater than 2) are considered.

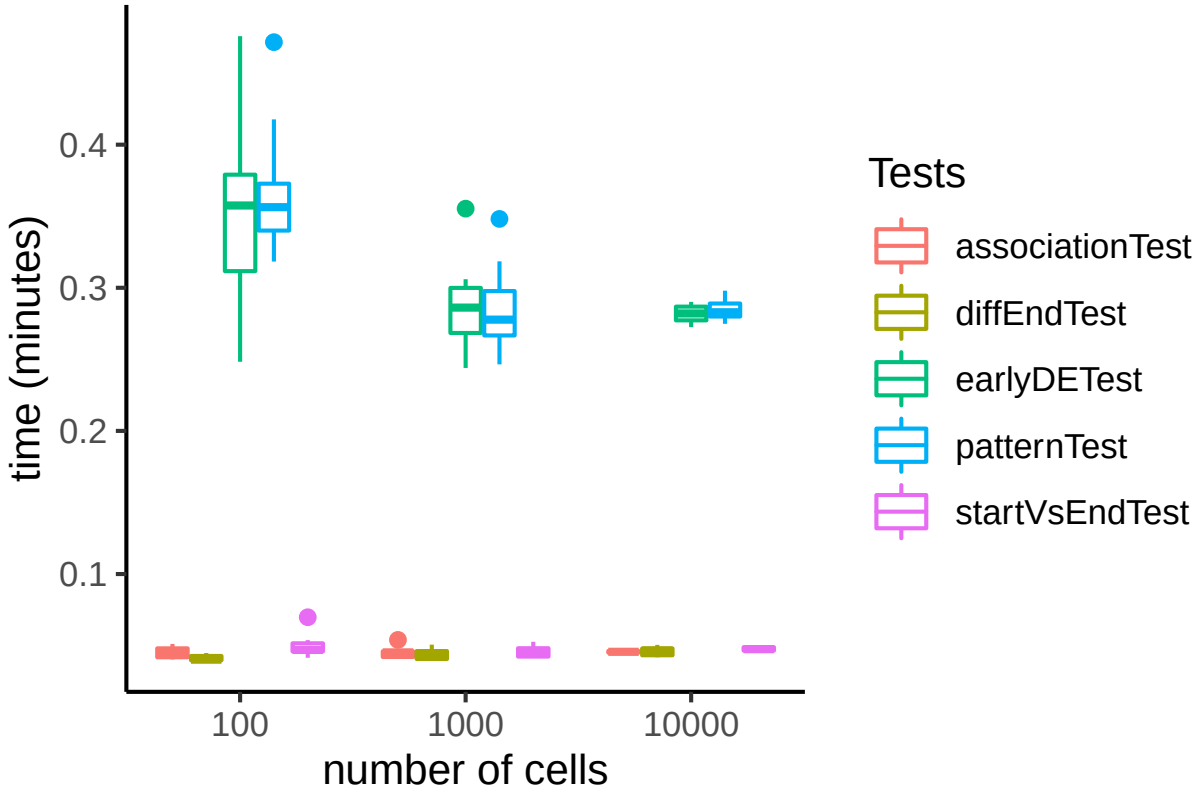


Figure C.16: *Computational time benchmark for the various tests implemented in* tradeSeq. Datasets of increasing size (in terms of number of cells) were simulated, each consisting of 5,000 genes. The `fitGAM` function of tradeSeq was ran with 4 knots. The computational time required to run the tests for all genes is benchmarked using the microbenchmark package, with 10 iterations each. `patternTest` and `earlyDETest` are slower than `associationTest`, `diffEndTest`, and `startVsEndTest`, but all take under 30 seconds to run. The time requirement is constant with respect to the number of cells.

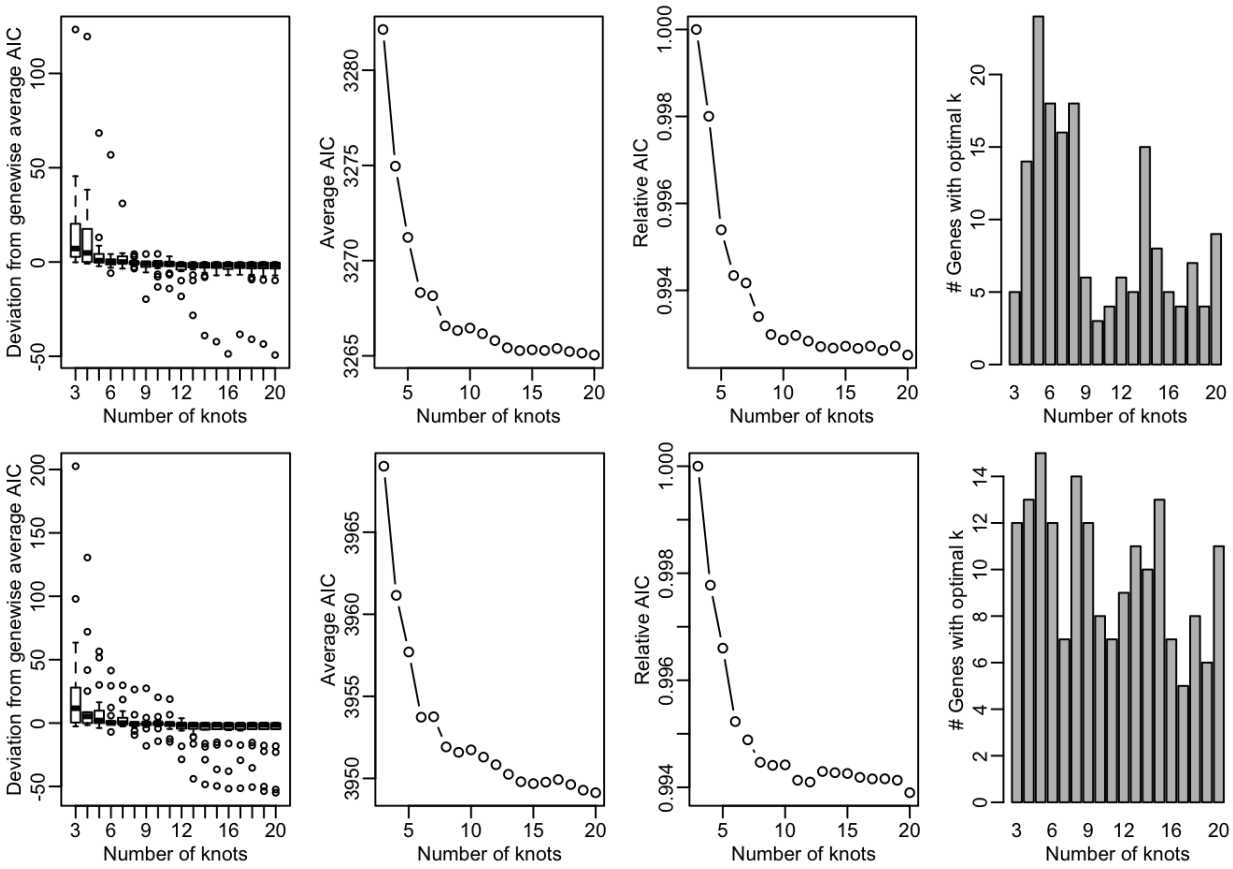


Figure C.17: *Mouse bone marrow dataset: Selecting the optimal number of knots k using the AIC.* Selecting the optimal number of knots, $k \in \{3, \ldots, 20\}$, using the AIC for two random subsets (top and bottom rows represent one subset each) of 250 genes, as implemented in the `evaluateK` function in tradeSeq. The left panel shows boxplots (center line, median; box limits, upper and lower quartiles; whiskers, 1.5× interquartile range) of the differences in AIC value with respect to the gene-wise average AIC for the range of k. The middle panels show the evolution of the average AIC (second panel) and relative AIC (third panel) across k. The relative AIC is defined as the relative change with respect to the average AIC at $k = 3$. The barplot in the right panel shows the number of genes which achieve their lowest AIC value for a given k. Here, only genes for which the AIC value varied substantially enough across k (i.e., range in AIC greater than 2) are considered.

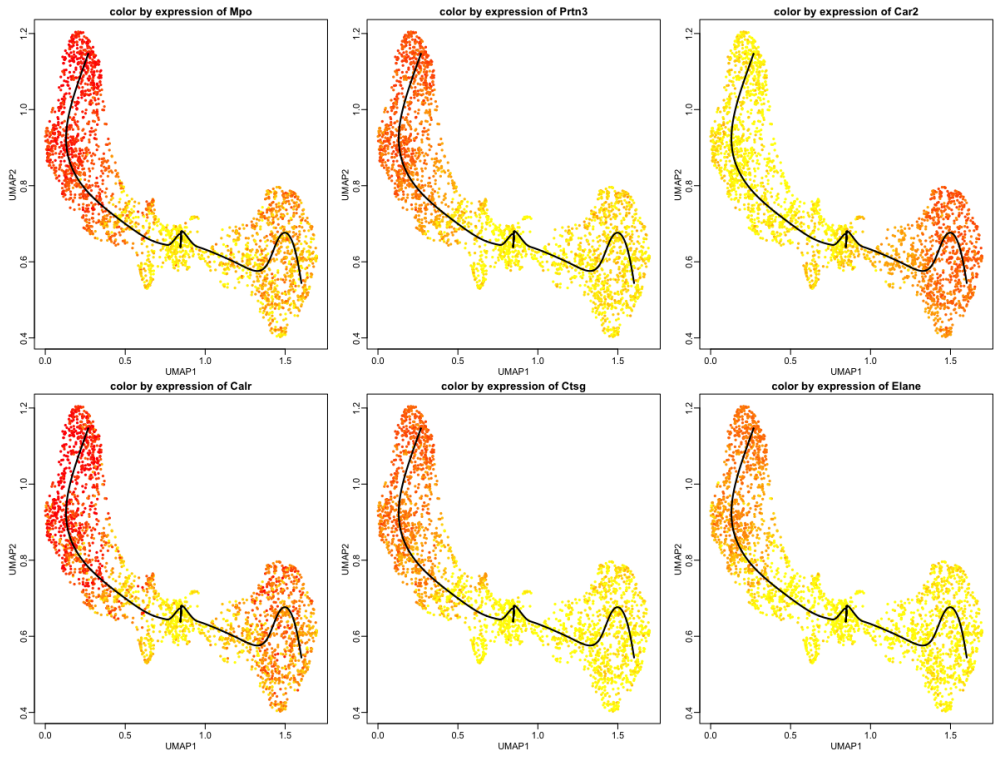


Figure C.18: *Mouse bone marrow dataset:* tradeSeq *recovers markers for the progenitor cell population.* This figure shows the six most significant genes when testing for differential expression between the progenitor cell type (i.e., starting point of the smoother) and differentiated cell types (i.e., endpoint of the smoother) for the data from Paul et al. [108] using tradeSeq. Yellow denotes low expression, while red denotes high expression.

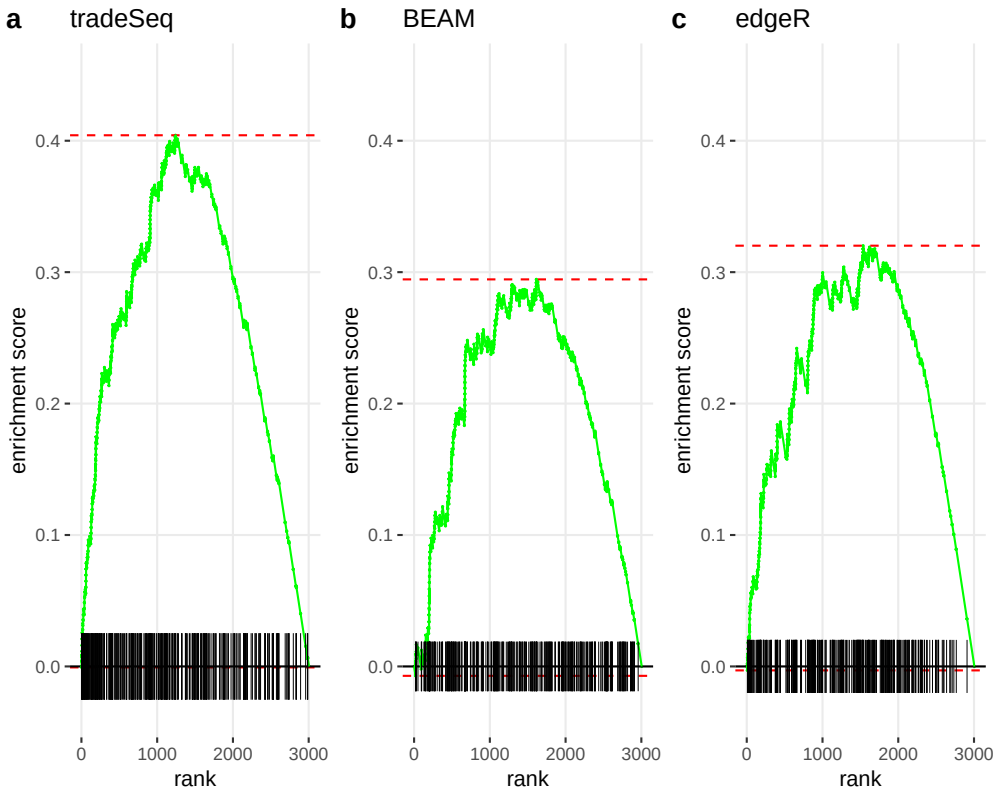


Figure C.19: *Mouse bone marrow dataset: Gene set enrichment plots for the erythrocyte gene set from Graaf et al. [47], for three differential expression methods:* tradeSeq, BEAM, *and* edgeR. Enrichment is determined for each method based on its respective ranking of the genes according to evidence for differential expression. Genes that are contained in the erythrocyte gene set are denoted with vertical lines at the bottom of each figure, and the green curve represents the gene enrichment score along the gene rankings. tradeSeq has the highest enrichment score, as determined by the dashed red line, since genes that are related to erythrocytes predominantly have high rankings for differential expression, while the distribution of erythrocyte genes seems more uniform with, for example, the BEAM approach.

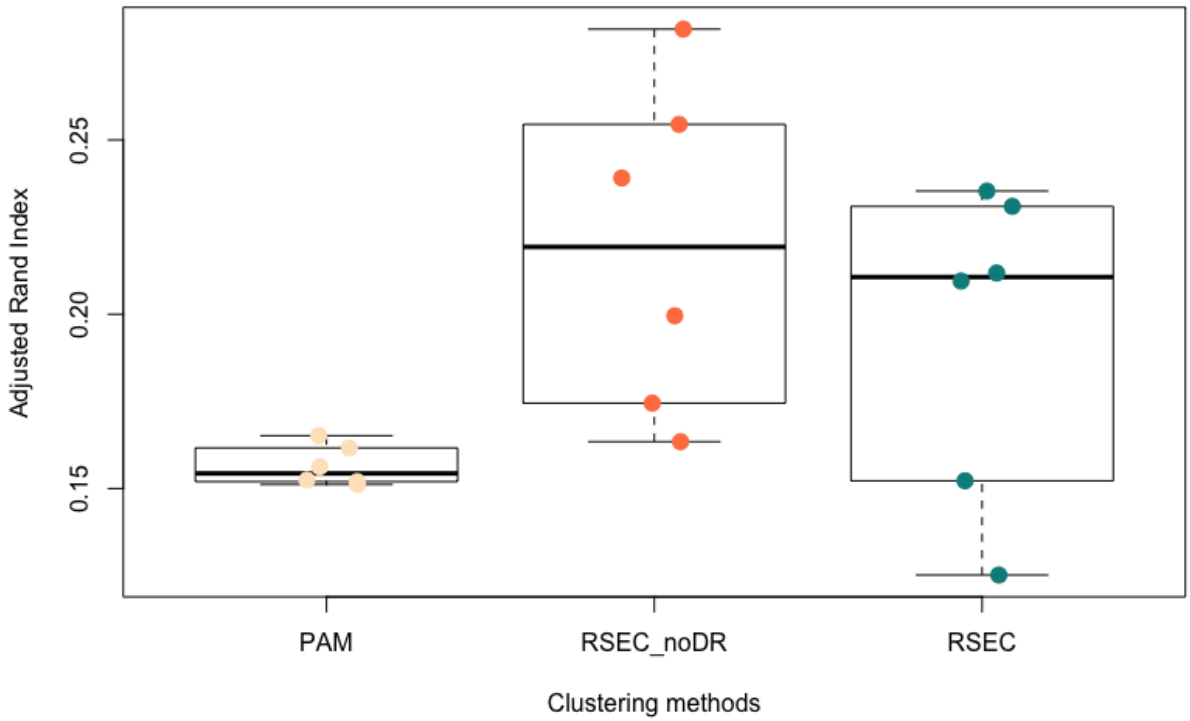


Figure C.20: *Mouse bone marrow dataset: Stability of gene clustering methods across bootstrap samples.* Boxplots (center line, median; box limits, upper and lower quartiles; whiskers, 1.5× interquartile range) of the clustering stability are shown. The stability of gene clustering methods is evaluated using non-parametric bootstrapping of the cells (for computational reasons, we restricted this evaluation to six bootstrap samples). We consider all genes found to be significant at a 5% FDR level by `patternTest` for the dataset of Paul et al. [108]. For each bootstrap sample, cells are sampled at random with replacement, the NB-GAM is refit using tradeSeq, and genes are clustered based on the tradeSeq fitted values, using both partitioning around medoids (PAM) and RSEC. We compare RSEC against PAM since the latter is also used in Monocle for gene clustering. For RSEC, we evaluate both clustering on the fitted values directly (method 'RSEC_noDR' in the figure) as well as clustering after dimensionality reduction with principal component analysis (the default for RSEC, as implemented in clusterExperiment, with automatic determination of the number of principal components; method 'RSEC' in the figure;). The stability of the clustering is evaluated by comparing the bootstrapped clusterings with the original clustering based on the full dataset using the adjusted Rand index (ARI) [58].

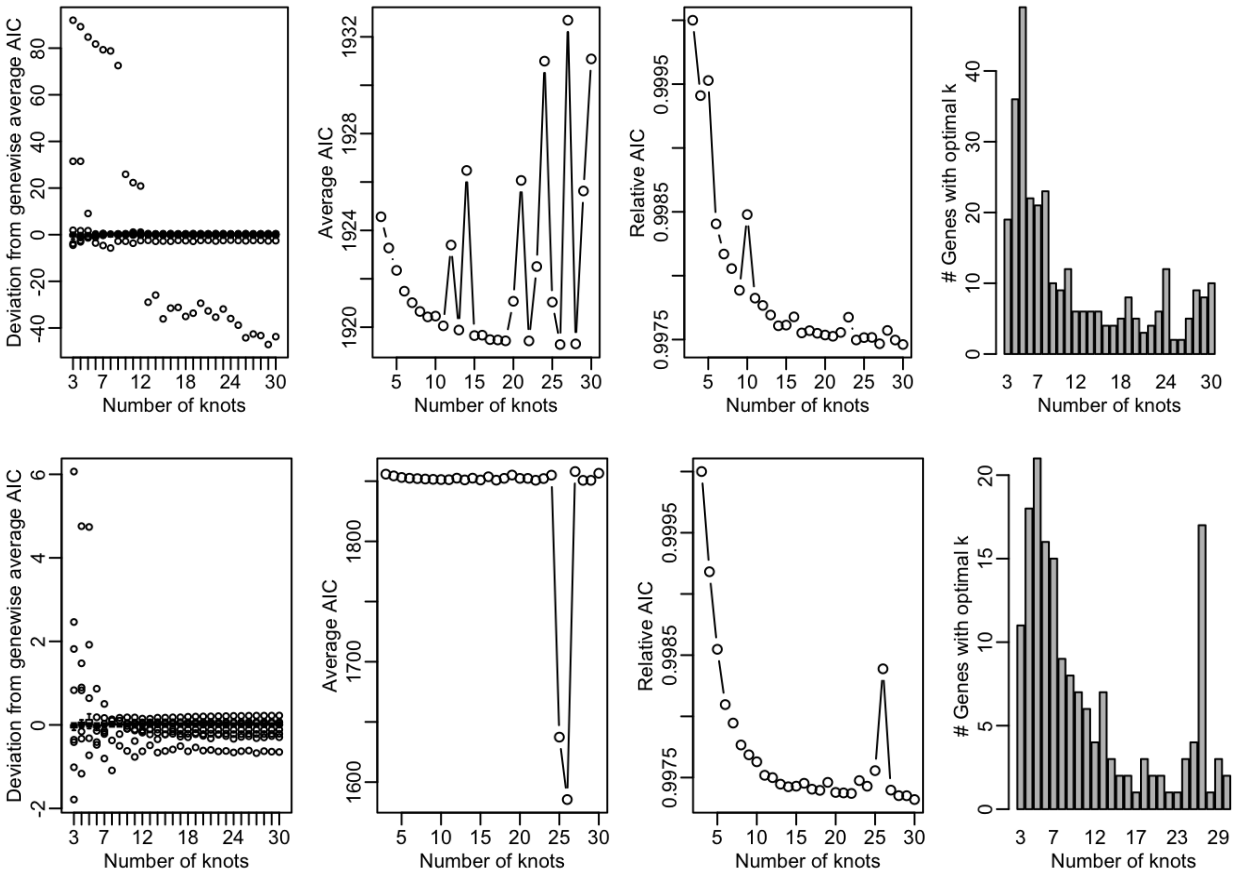


Figure C.21: *Olfactory epithelium dataset: Selecting the optimal number of knots k using the AIC.* Selecting the optimal number of knots, $k \in \{3, \ldots, 30\}$, using the AIC for two random subsets (top and bottom rows represent one subset each) of 1000 genes, as implemented in the `evaluateK` function in `tradeSeq`. The left panel shows boxplots (center line, median; box limits, upper and lower quartiles; whiskers, 1.5× interquartile range) of the differences in AIC value with respect to the gene-wise average AIC for the range of k. The middle panels show the evolution of the average AIC (second panel) and relative AIC (third panel) across k. The relative AIC is defined as the relative change with respect to the average AIC at $k = 3$. The barplot in the right panel shows the number of genes which achieve their lowest AIC value for a given k. Here, only genes for which the AIC value varied substantially enough across k (i.e., range in AIC greater than 2) are considered.

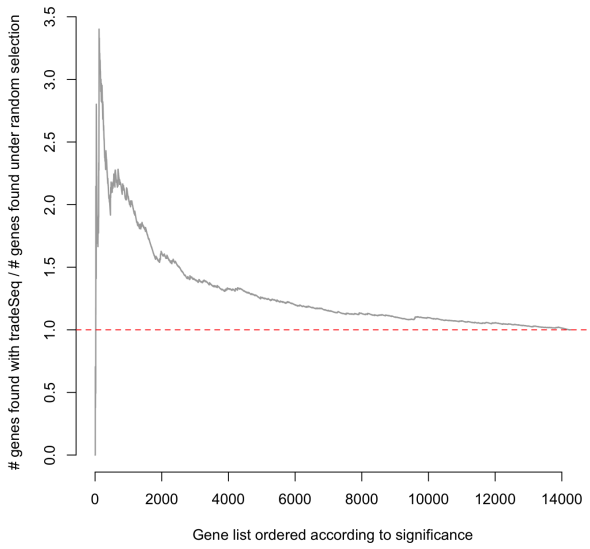


Figure C.22: *Mouse olfactory epithelium dataset: Cell cycle genes in the neuronal lineage.* The figure illustrates the enrichment of cell cyle genes in lists of genes whose expression was found to be most significantly associated with the neuronal lineage according to the `associationTest` procedure in tradeSeq. The list of cell cycle related genes was obtained from the Mouse Genome Informatics (MGI) website at http://www.informatics.jax.org/go/term/GO:0007049. On the x-axis, genes are ordered according to their significance based on `associationTest`. The y-axis shows the ratio of the number of cell cycle genes among a set of top significant genes relative to the number of cell cycle genes one would expect by chance (i.e., if cell cycle genes were randomly found DE/sampled). If we let C denote the proportion of genes associated with the cell cycle according to the MGI database, then, under the hypothesis that cell cycle genes are randomly discovered as DE, the expected number of cell cycle genes in the list of top N genes is NC. The relative number that is plotted on the y-axis is then the ratio between the number of cell cycle genes discovered by tradeSeq for a given top list of size N and NC.

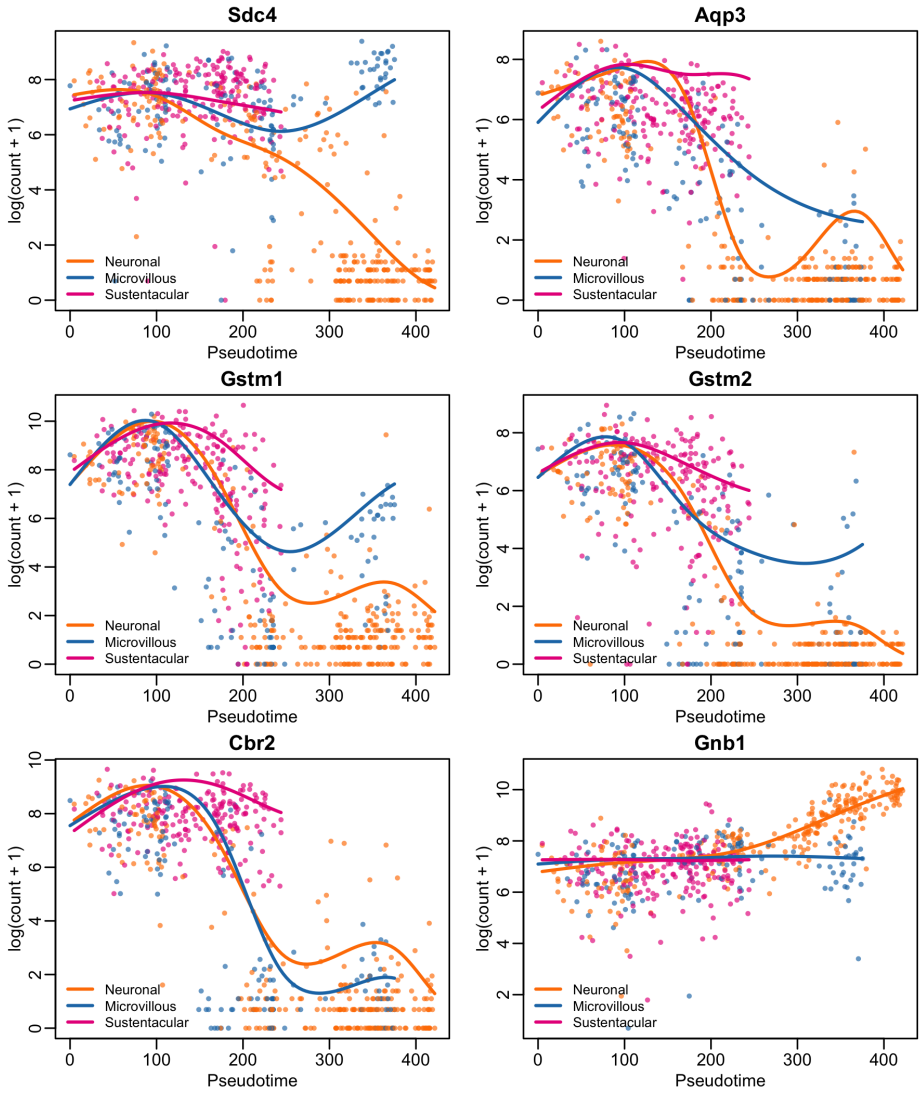


Figure C.23: *Mouse olfactory epithelium dataset: Top six differentially expressed genes as identified by a ZINB analysis with* tradeSeq `patternTest`. Every lineage is represented by a smooth function of gene expression along pseudotime. The cells assigned to a particular lineage based on the slingshot weights are represented with the same color as the lineage.

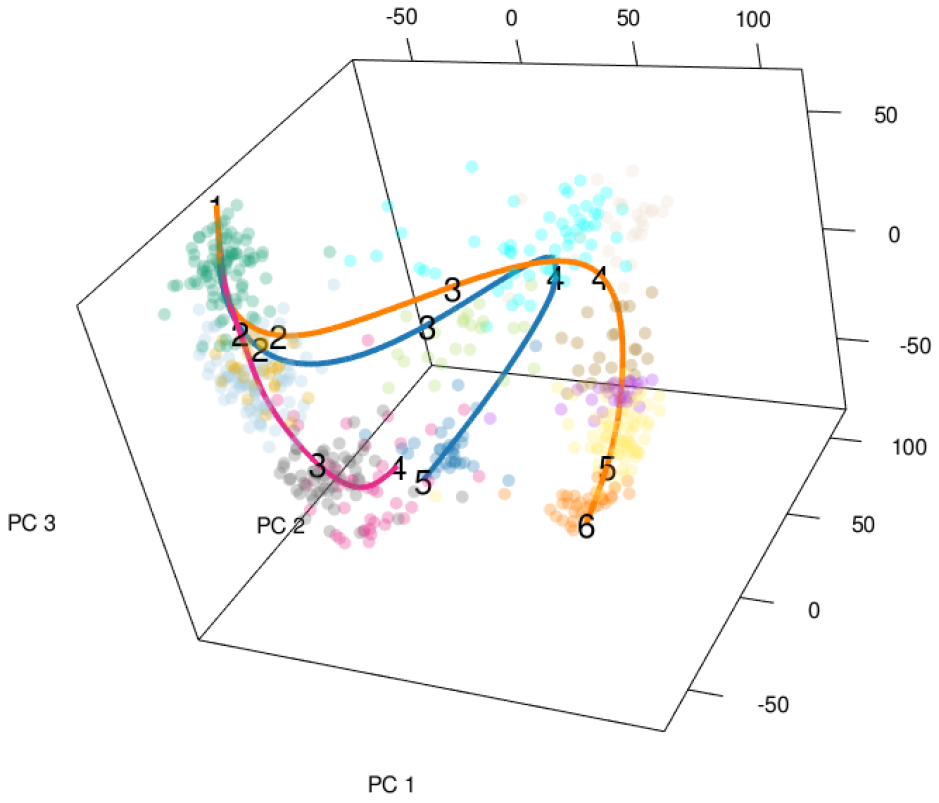


Figure C.24: *Mouse olfactory epithelium dataset: Trajectory with knots.* Three-dimensional PCA plot of the scRNA-seq data from Fletcher et al. [44], where cells are colored according to their cluster membership as defined in the original paper (see Methods). The simultaneous principal curves for the lineages inferred by slingshot are displayed. The numbers on each lineage specify the knot points used to fit the ZINB-GAM in tradeSeq. The first branching event occurs between knots 1 and 3.

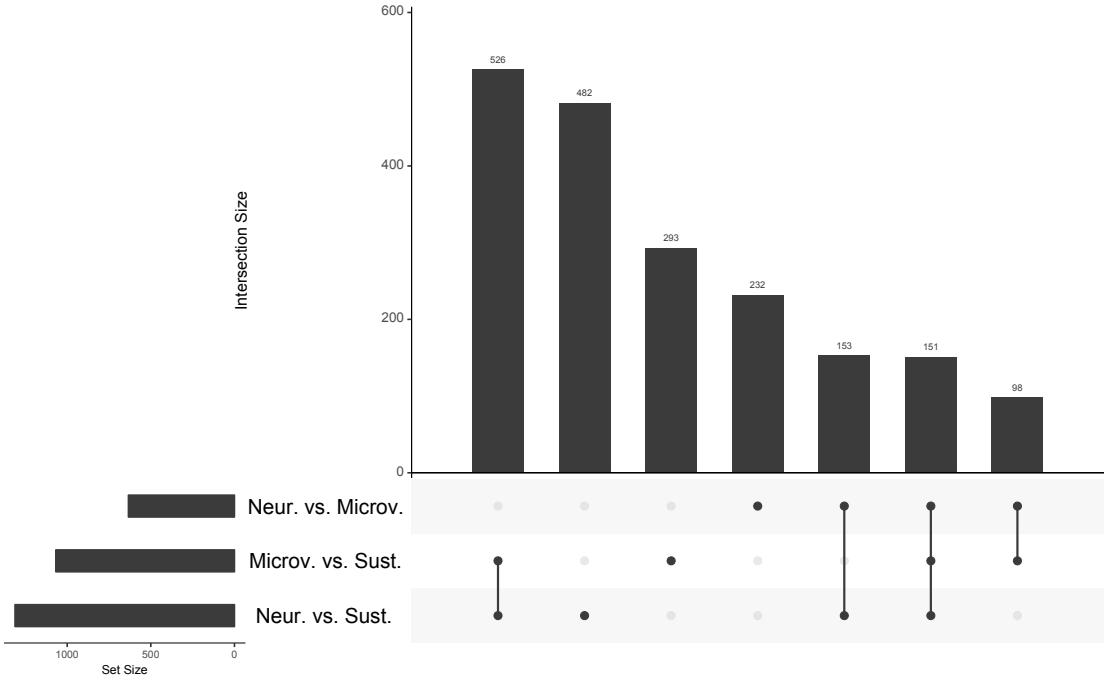


Figure C.25: *Mouse olfactory epithelium dataset: UpSet plot for* `earlyDETest` *applied around the first branching point to each pair among the three lineages.* Left panel: Barplot of the number of DE genes for each contrast/pair (5% nominal FDR level). Top panel: Barplot of the number of DE genes for one, two of the three, or all three contrasts. Center panel: Contrasts being considered for the top panel. For example, the first column shows that 526 genes are DE for both the Microv vs. Sust contrast and the Neur. vs. Sust contrast.

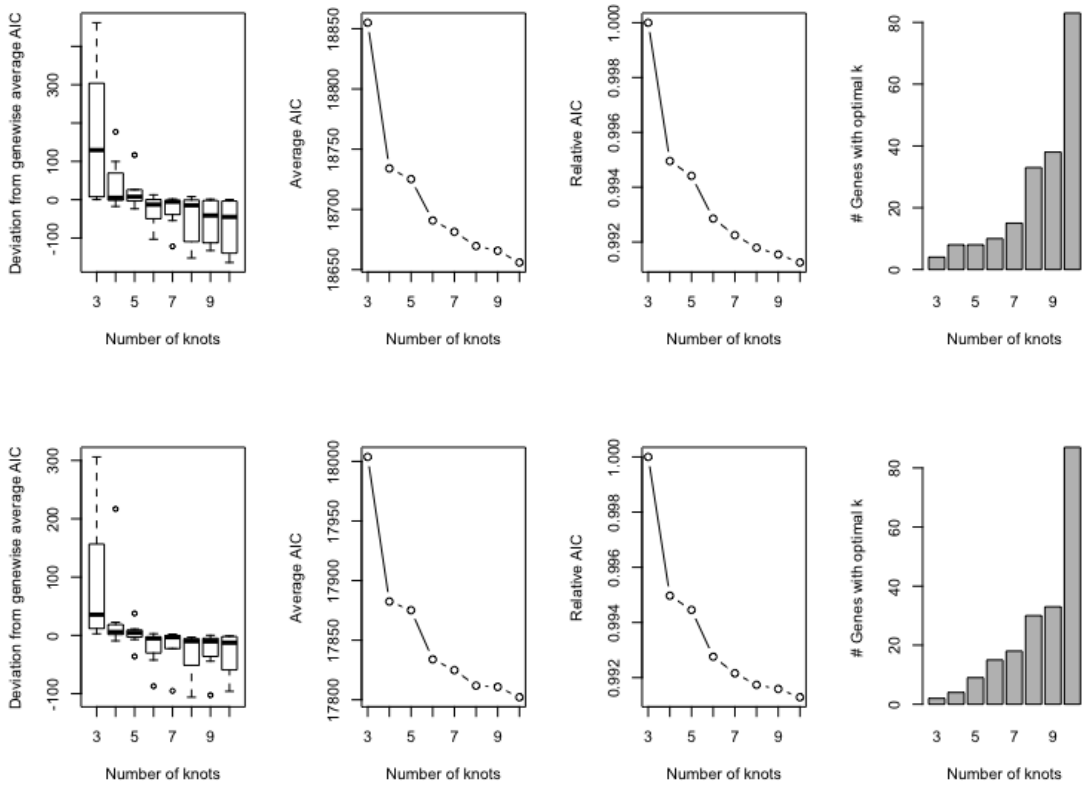


Figure C.26: *Adipocyte differentiation dataset: Selecting the optimal number of knots k using the AIC.* Selecting the optimal number of knots, $k \in \{3, \ldots, 10\}$, using the AIC for a random subset of 200 genes, as implemented in the `evaluateK` function in tradeSeq. The left panel shows boxplots (center line, median; box limits, upper and lower quartiles; whiskers, 1.5× interquartile range) of the differences in AIC value with respect to the gene-wise average AIC for the range of k. The middle panels show the evolution of the average AIC (second panel) and relative AIC (third panel) across k. The relative AIC is defined as the relative change with respect to the average AIC at $k = 3$. The barplot in the right panel shows the number of genes which achieve their lowest AIC value for a given k. Here, only genes for which the AIC value varied substantially enough across k (i.e., range in AIC greater than 2) are considered.

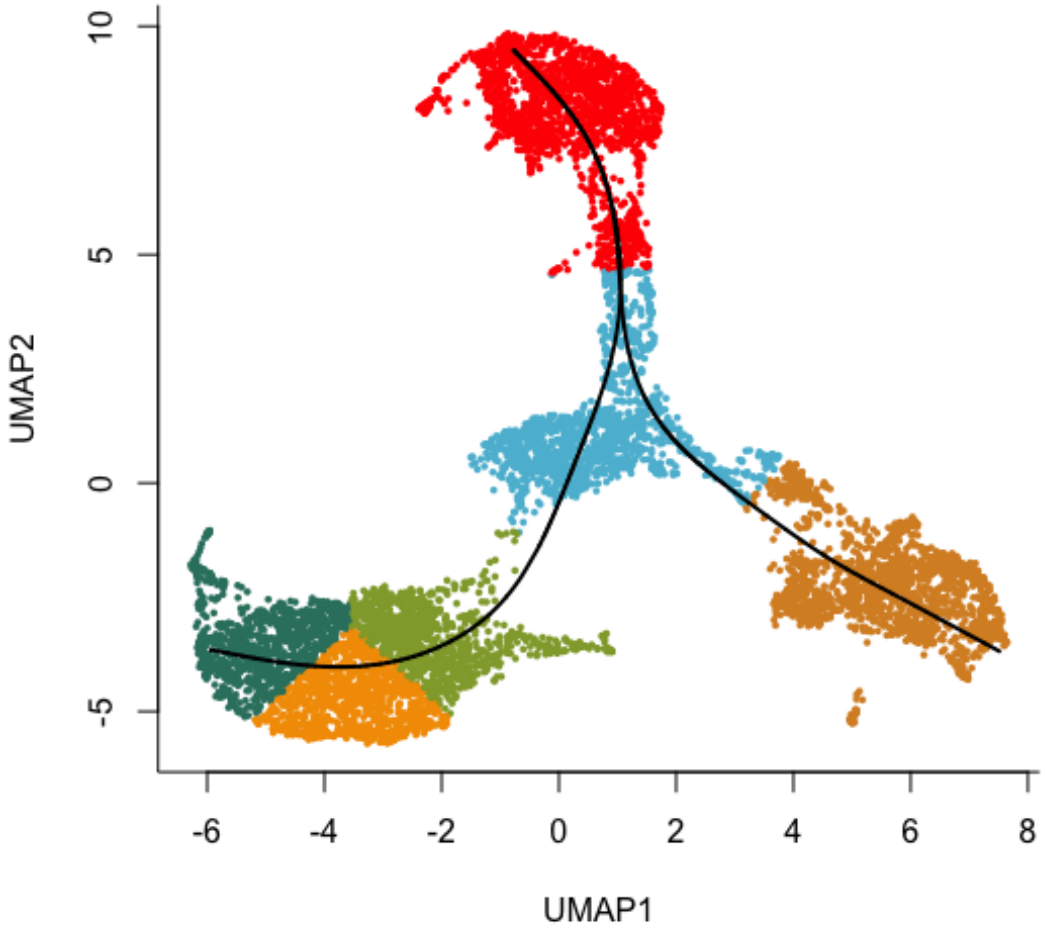


Figure C.27: *Adipoyte differentiation dataset: Inferred trajectory.* The scRNA-seq data are plotted in 2D UMAP space, and each cell is colored according to its cluster membership as derived by k-means clustering with $k = 6$ clusters. The black solid line represents the trajectory as estimated by slingshot.

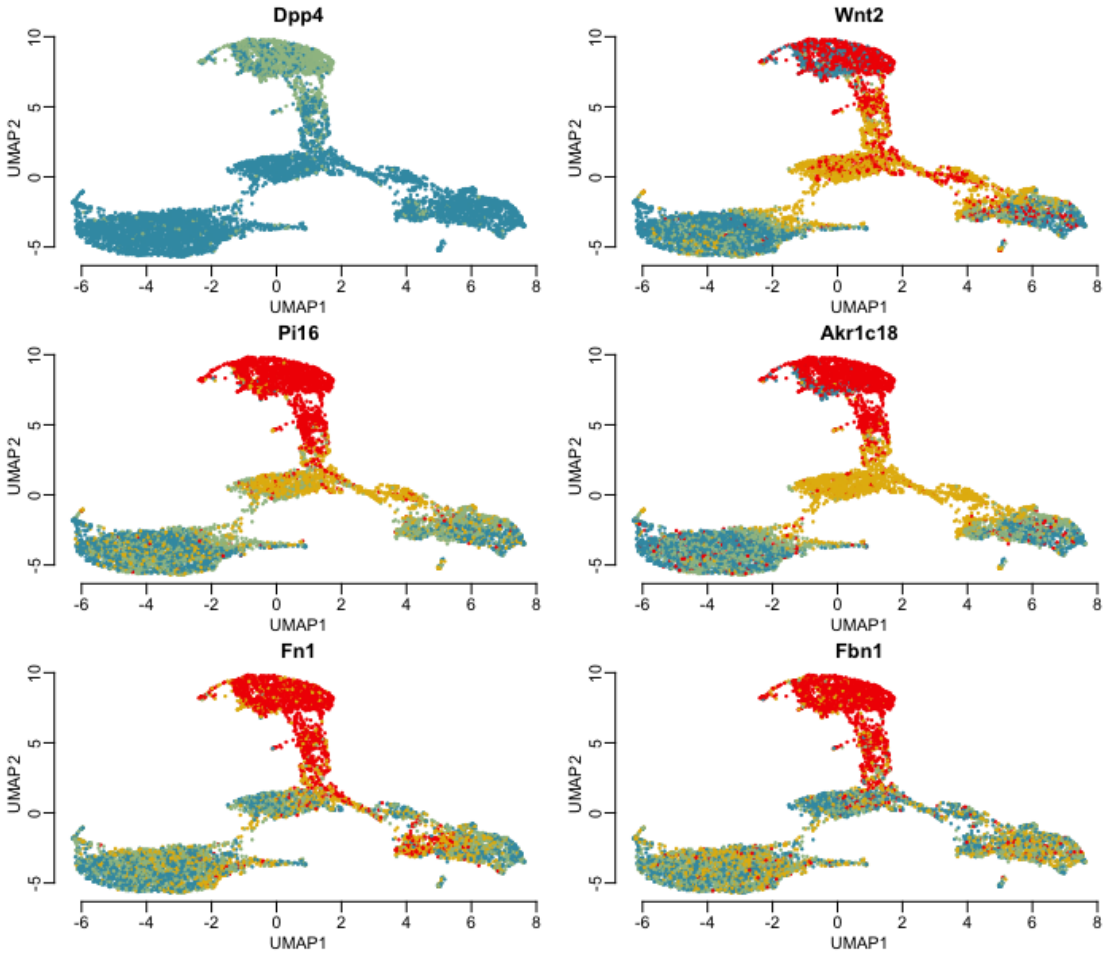


Figure C.28: *Adipocyte differentiation dataset: Top markers for progenitor cell population.* The scRNA-seq data are plotted in 2D UMAP space, and each cell is colored according to the expression of one of six genes (the expression range is divided into 4 bins, where blue corresponds to low expression and red corresponds to high expression). The top row corresponds to two marker genes, *Dpp4+* and *Wnt2*, from the original manuscript [95]. Other plots are top genes identified with the `startVsEndTest` procedure from tradeSeq.

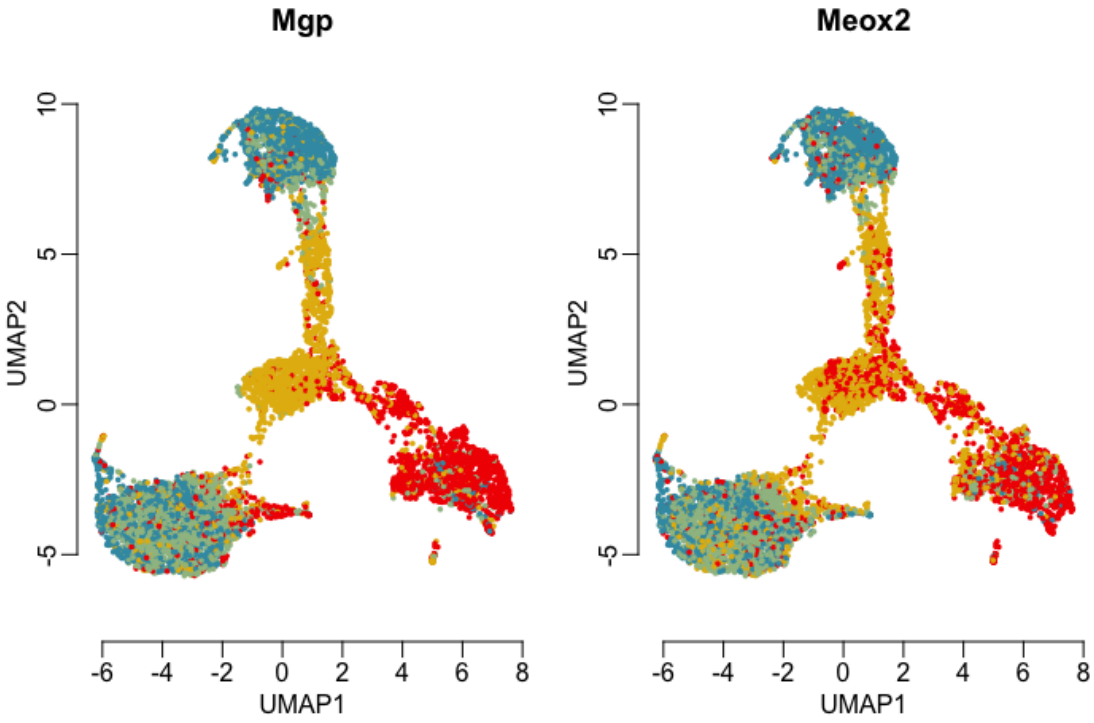


Figure C.29: *Adipocyte differentiation dataset: Genes upregulated in the adipocyte precursor stage and a single differentiated cell type.* The scRNA-seq data are plotted in 2D UMAP space, and each cell is colored according to the expression of one of two genes (the expression range is divided into 4 bins, where blue corresponds to low expression and red corresponds to high expression).

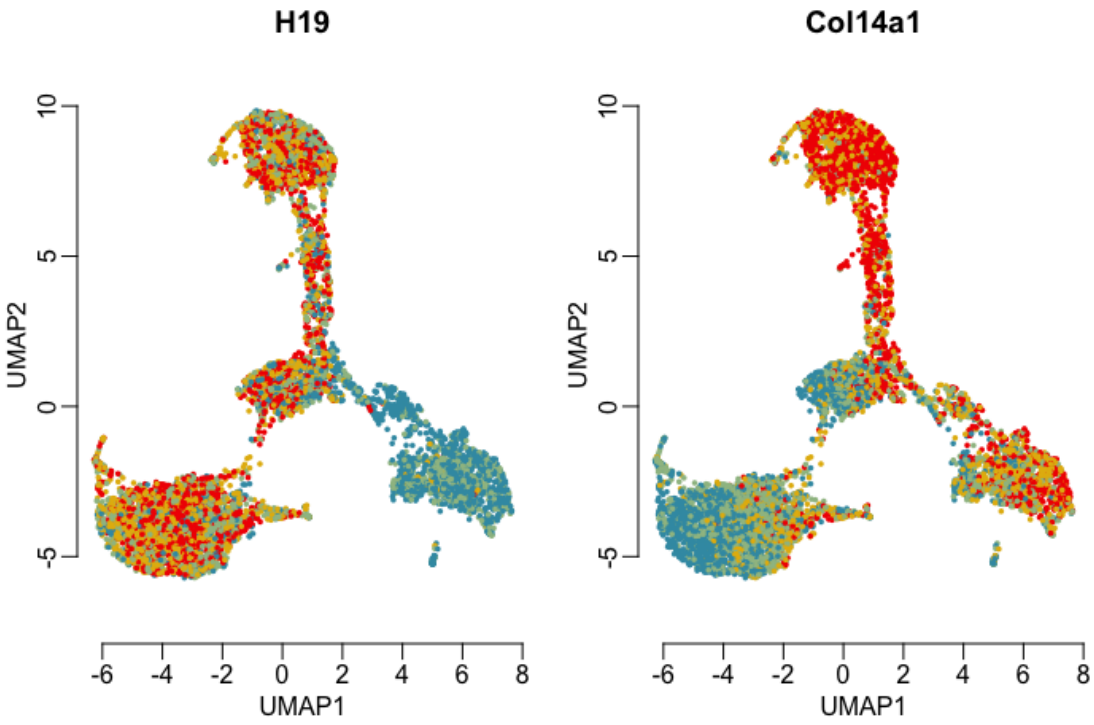


Figure C.30: *Adipocyte differentiation dataset: Genes sporadically upregulated across the entire lineage for a single differentiated cell type.* The scRNA-seq data are plotted in 2D UMAP space, and each cell is colored according to the expression of one of two genes (the expression range is divided into 4 bins, where blue corresponds to low expression and red corresponds to high expression).

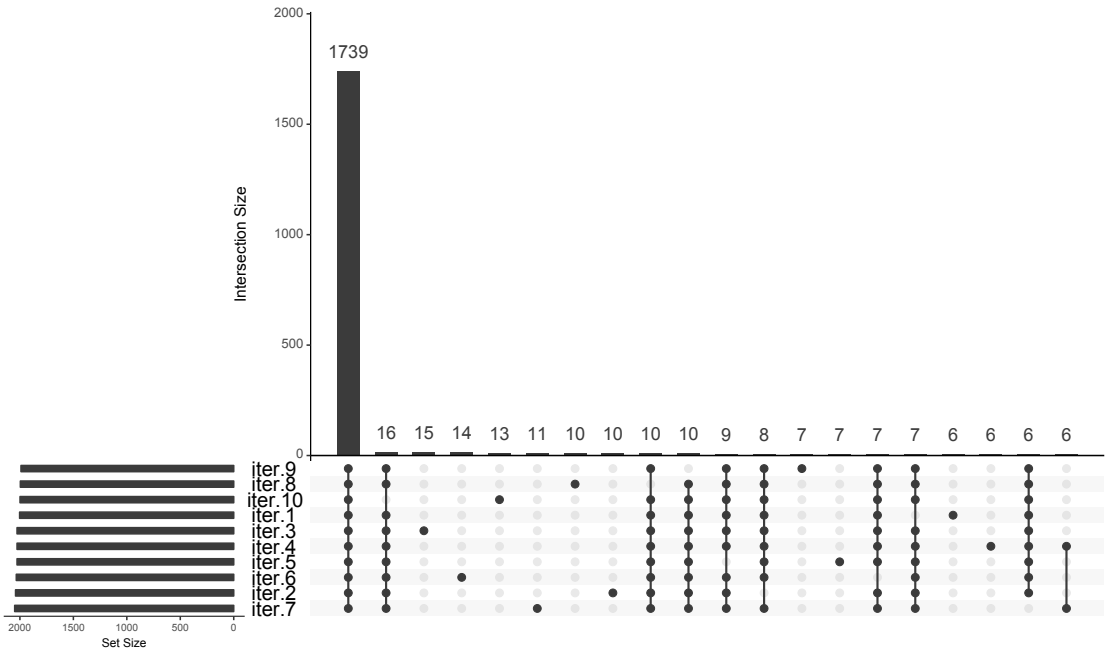


Figure C.31: *Mouse bone marrow dataset: UpSet plot for* `startVsEndTest`*, for multiple assignments of cells to lineages.* Ten random multinomial assignments of cells to lineages are performed and denoted with 'iter.1' to 'iter.10'. Left panel: Barplot of the number of DE genes for each assignment (5% nominal FDR level). Top panel: Barplot of the number of DE genes in common for various combinations of the 10 assignments. Center panel: Assignments being considered for the top panel. Only the 20 combinations with the largest number of common genes are shown in the figure.

Appendix D

Supplementary Figures for chapter 4

Simulations

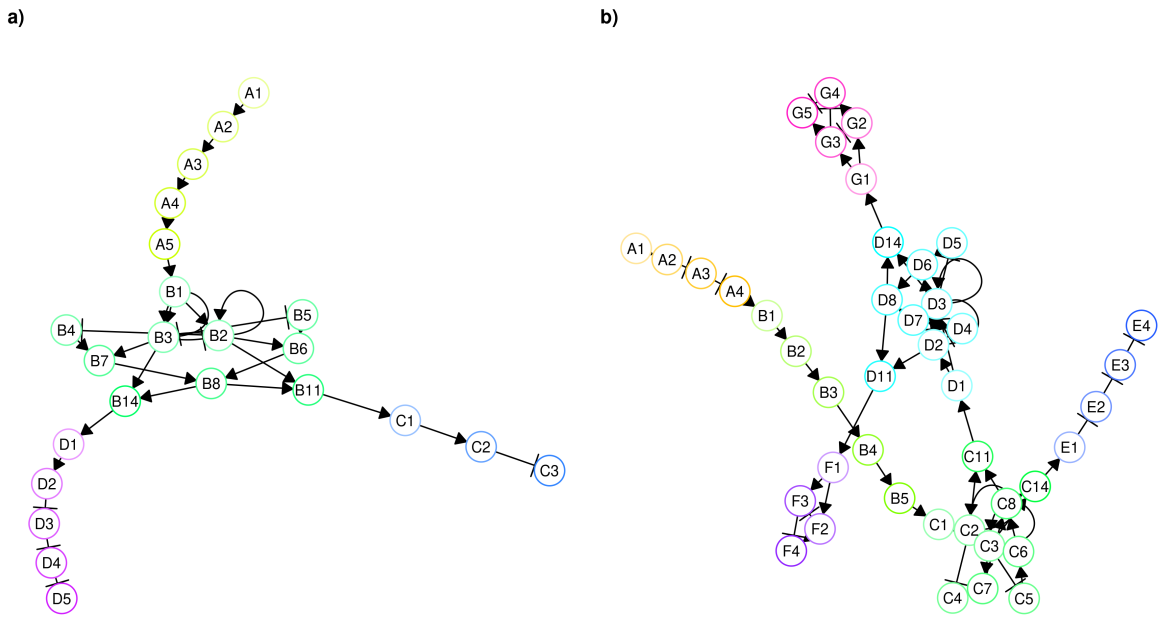


Figure D.1: *Simulation example.* Regulator networks for the **(a.)** two-lineage and **(b.)** three-lineage trajectories.

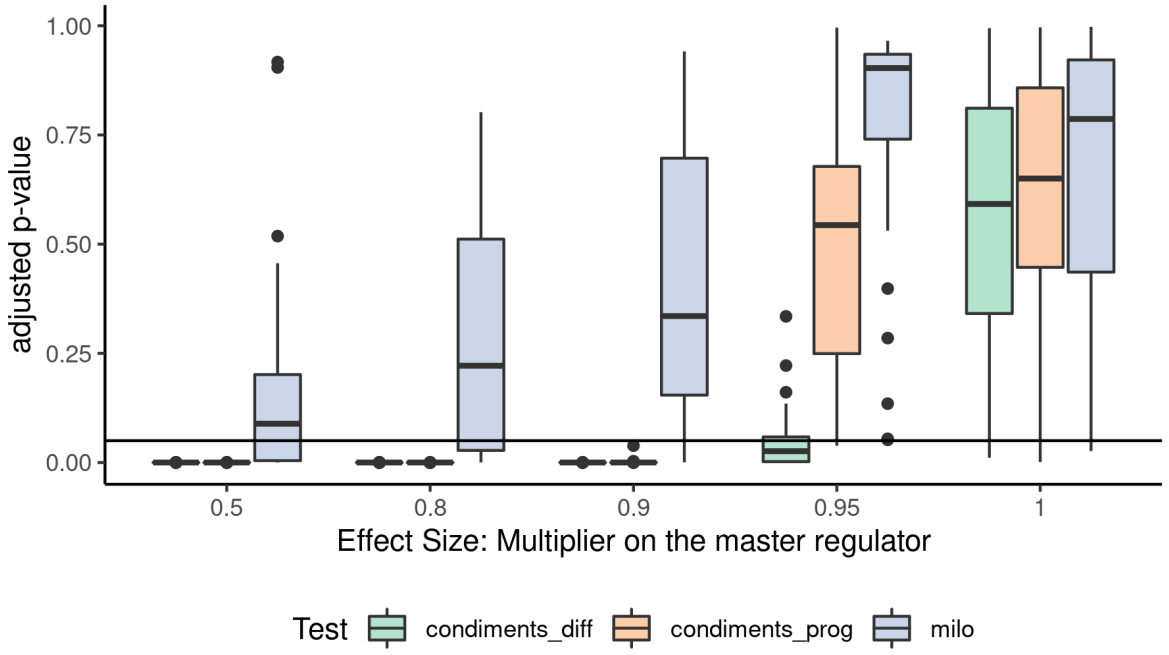


Figure D.2: *Results on the third type of dataset.* For all values of $m \in \{.5, .8, .9, .95, 1\}$, we generate null datasets with two lineages and three conditions and we compute the adjusted p-values of all tests that can handle 3 conditions. The distributions of p-values are then displayed. $m = 1$ is negative (no effect), while $m < 1$ is positive (some effect) with smaller values (toward the left) representing stronger effect.

TCDD

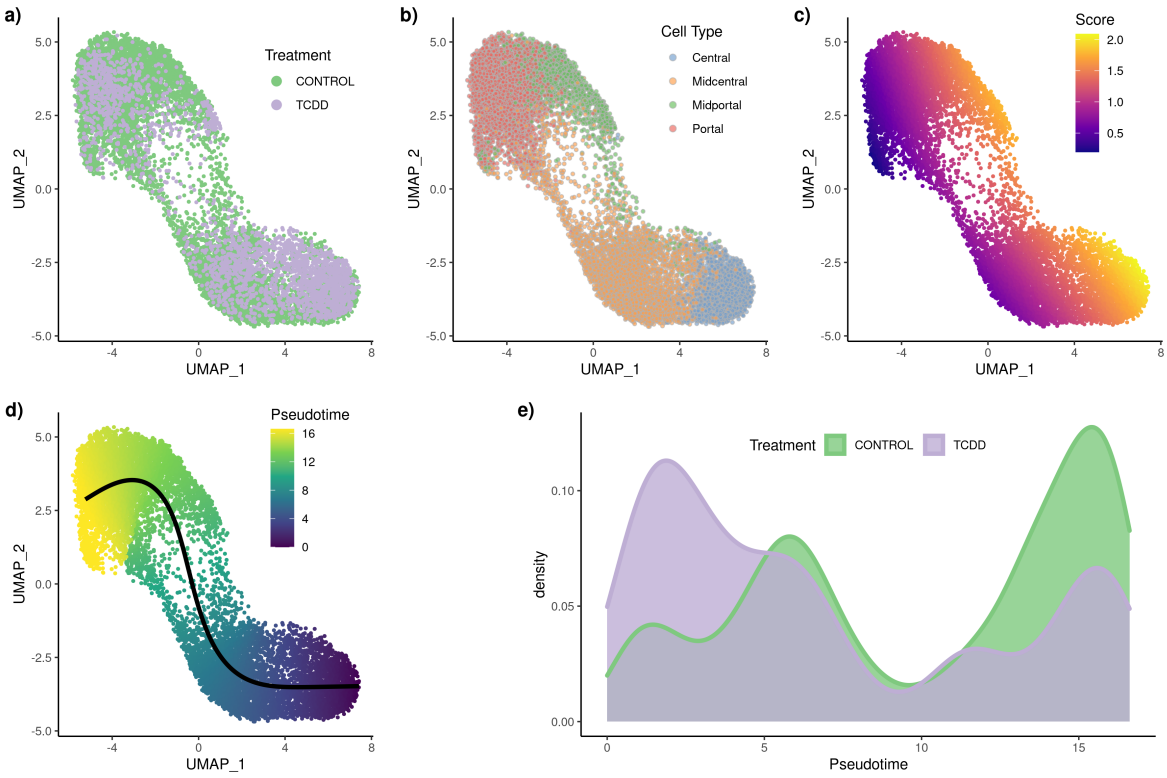


Figure D.3: ***TCDD*** *dataset [100]: Differential topology and differential progression.* After normalization and projection on a reduced-dimensional space, the cells can be represented, colored either by treatment label **(a.)**, cell type **(b.)**, or batch **(c.)**. Using the treatment label and the reduced-dimensional coordinates, an imbalance score is computed and displayed **(d.)**. The `diffTopoTest` rejects the null and separate trajectories are fitted for each condition **(e.)**. After mapping the lineages, there is also differential progression: the pseudotime distribution along the trajectory are not identical **(f.)** and we indeed reject the null using the `diffProgressionTest`.

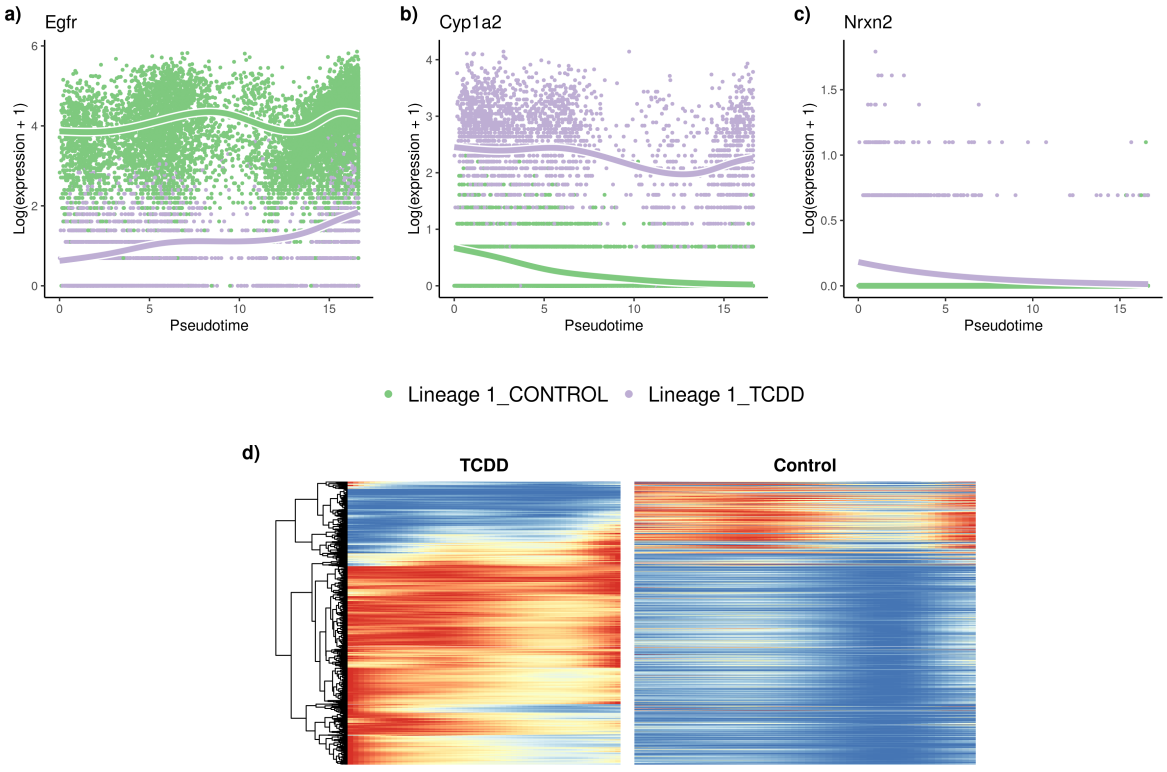


Figure D.4: ***TCDD*** *dataset [100]: Differential expression.* The tradeSeq gene expression model is fitted using the trajectory computed with slingshot. Differential expression between conditions is assessed using the `conditionTest` and genes are ranked according to the test statistics. The genes with the highest **(a.)**, second highest **(b.)**, and smallest **(c.)** test statistics are displayed. After adjusting the p-values to control the FDR at a nominal level of 5%, we display genes in both conditions using a pseudocolor image **(d.)**.

KRAS

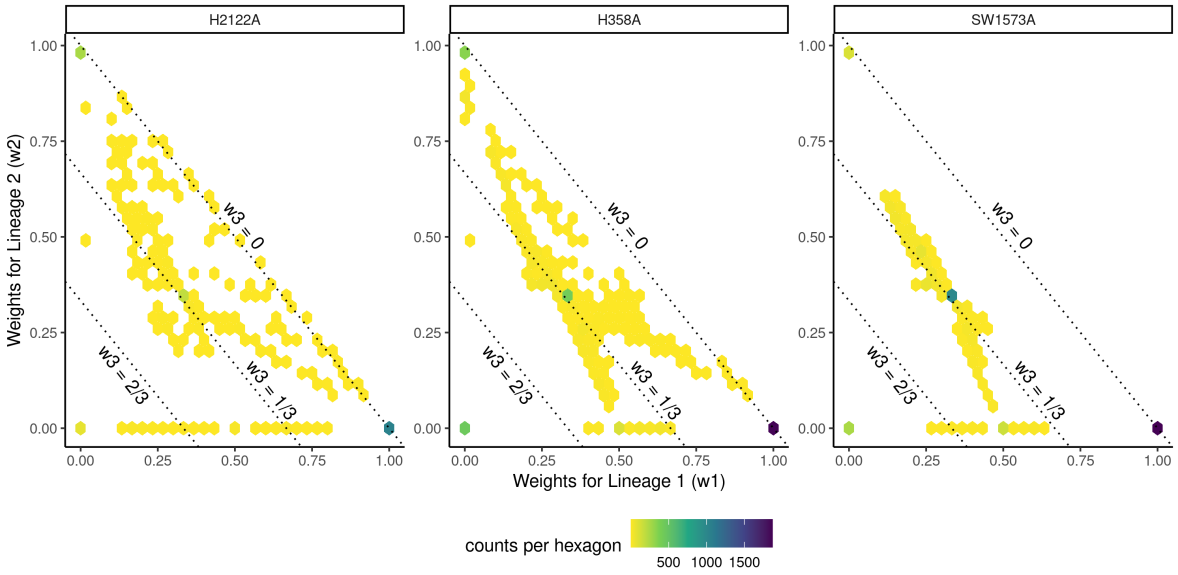


Figure D.5: ***KRAS*** *dataset [175]: Differential differentiation.*

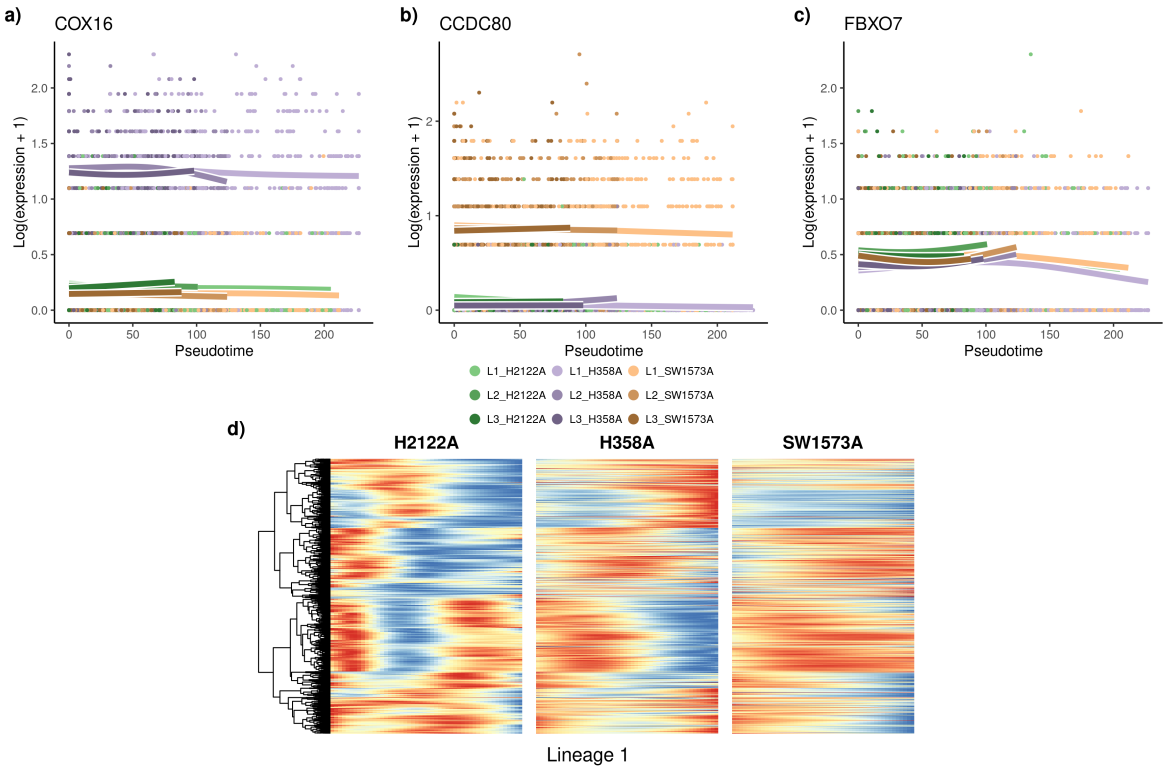


Figure D.6: ***KRAS*** *dataset [175]: Differential expression.* The tradeSeq gene expression model is fitted using the trajectory computed with slingshot. Differential expression between conditions is assessed using the `conditionTest` and genes are ranked according to the test statistics. The genes with the highest **(a.)**, second highest **(b.)**, and smallest **(c.)** test statistics are displayed. Focusing on the first lineage, we select all differentially expressed genes in that lineage after adjusting the p-values to control the FDR at a nominal level of 5%. We display the genes for all three conditions using a pseudocolor image **(d.)** along this first lineage.

Appendix E

Supplementary Figures for chapter 5

Minimal p-value

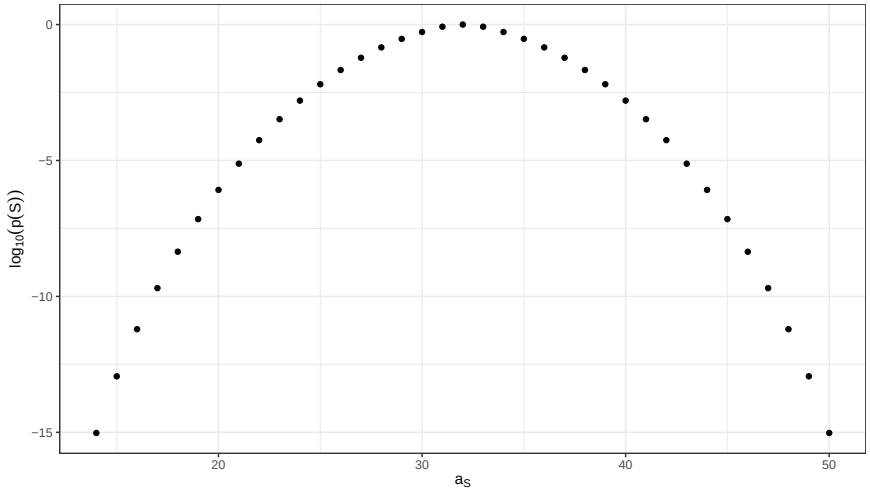


Figure E.1: *Finite numbers of possible p-values (log scale) for a fixed value of* $n1 = 50$ *and* $x_{\mathcal{S}} = 64$. *Using the notation from table 5.1, with* $J = 1$, $n_1 = 50$, $n = 100$ *and* $x_{\mathcal{S}} = 64$, *the p-value of the* χ^2 *test is computed for all possible values of* $a_{\mathcal{S}}$. *Since there are only a finite number of possible* $a_{\mathcal{S}}$ *values, there are a finite number of possible p-values, and therefore a smallest one. This minimal p-value can be computed from* $x_{\mathcal{S}}$, $n1$ *and* n *alone and is* $\sim 10^{-15}$

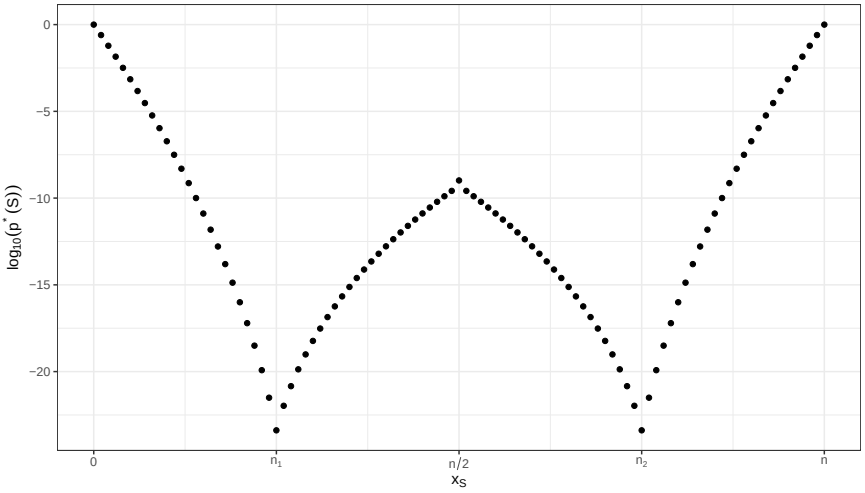


Figure E.2: *Minimum p-value as a function of $x_{\mathcal{S}}$ for fixed values of $n_1 = 25$ and $n = 100$. Using the notation from table 5.1, with $J = 1$, $n_1 = 25$, $n = 100$, the minimal p-value $p^*(\mathcal{S})$ of the χ^2 test is computed for all possible values of $x_{\mathcal{S}}$. For $x_{\mathcal{S}} \geq \max(n_1, n_2)$, the minimal p-value is strictly increasing. If we reach that stage, we can prune the graph and stop the exploration in that direction. Indeed, if $\mathcal{S}' \supseteq \mathcal{S}$ then $x_{\mathcal{S}'} \geq x_{\mathcal{S}}$. So if $p^*(\mathcal{S}) > \frac{\alpha}{k}$, we know that $p^*(\mathcal{S}') > \frac{\alpha}{k}$ without computing it.*