GHENT UNIVERSITY

Modelling single-cell dynamics with trajectories and gene regulatory networks

Robrecht Cannoodt

Supervisor:
Prof. Dr. Yvan Saeys
Co-supervisor:
Prof. Dr. Katleen De Preter

December 2019

Thesis submitted in fulfilment of the requirements
for the degree of Doctor in Computer Science

FACULTY OF SCIENCES
VIB-UGENT CENTER FOR INFLAMMATION RESEARCH
DaMBi
CMGG
fwo

GHENT UNIVERSITY

Modelling single-cell dynamics with trajectories and gene regulatory networks

Robrecht Cannoodt

Thesis submitted in fulfilment of the requirements for the degree of
Doctor in Computer Science, 2019

Supervisors:
Prof. Dr. Yvan Saeys
Prof. Dr. Katleen De Preter

FACULTY OF SCIENCES · VIB-UGENT CENTER FOR INFLAMMATION RESEARCH · DaMBi · CMGG · fwo

Vakgroep Toegepaste Wiskunde, Informatica, en Statistiek
Faculteit Wetenschappen, Universiteit Gent
Krijgslaan 281 - S2, 9000 Gent

For Remi,

You never fail to put a smile on my face.

Contents

Nomenclature

kNN	k-nearest-neighbour
CART	Classification And Regression Trees
DE	Differential Expression
DNA	Deoxyribonucleic Acid
DR	Dimensionality Reduction
GRN	Gene Regulatory Network
HCA	Human Cell Atlas
LDMS	Landmark Multi-Dimensional Scaling
MDS	Multi-Dimensional Scaling
mRNA	Messenger RNA
NI	Network Inference
PCA	Principal Component Analysis
RF	Random Forests
RNA	Ribonucleic Acid
t-SNE	t-distributed Stochastic Neighbor Embedding
TCGA	The Cancer Genome Atlas
TDE	Trajectory Differential Expression
TF	Transcription Factor
TI	Trajectory Inference
UMAP	Uniform Manifold Approximation and Projection

Summary

Recent developments in the *single-cell omics* technologies triggered a major milestone in the field of (cell)biology. Current techniques can be used to profile both the genome and the transcriptome of individual cells, leading to important insights into cellular heterogeneity and dynamic cell processes. This evolution has led, among other things, to the Human Cell Atlas initiative, an international collaboration project with the aim of mapping all human cell types in terms of its biomolecular contents.

During this doctoral project I have been able to follow these inspiring advancements first hand, but I also made scientific contributions myself, by extensively evaluating new computational tools and by developing various tools and algorithms.

A common problem of pioneering computational tools is the lack of sufficient available datasets to enable the assessment of performances in a quantitative manner. During this PhD, I developed a flexible and extendible *in silico* simulator for individual cells, with the aim of supporting current and future computational research fields and to help kick-start emerging computational sub-domains (Chapter 2).

I conducted an extensive study of 45 *trajectory inference* methods (Chapter 3). Trajectory inference is a new type of computational analysis with the aim of identifying and studying transitions between main cellular states. While this class of methods is one of the fastest growing subdomains within the *single-cell omics*, a quantitative study of the advantages and disadvantages was hitherto lacking.

In addition, I developed four new software tools. These methods include an extensive toolkit for inferring, visualising and interpreting trajectories (Chapter 5), a robust tool for inferring linear trajectories (Chapter 4), a novel algorithm for studying gene regulation at a single-cell or single-sample level (Chapter 6), and a tool for quantifying the topological changes resulting from minor perturbations in a network (Chapter 7).

Finally, I reflect on current challenges within the field, by providing guidelines on how to perform large-scale benchmarking experiments of computational methods (Chapter 8).

Throughout this project I particularly enjoyed working in an open-source ecosystem, collaborating with scientists around the world to tackle unsolved problems, whilst learning best practices by inspecting the source code of exemplary software projects.

Samenvatting

Recente ontwikkelingen in de *single-cell omics* technologieën vormen een belangrijke mijlpaal in het domein van de (cel)biologie. Met de huidige technieken kunnen zowel het genoom als het transcriptoom van individuele cellen geprofileerd worden wat leidt tot belangrijke inzichten in de cellulaire heterogeniteit en de dynamische celprocessen. Deze evolutie heeft onder meer geleid tot het Human Cell Atlas initiatief, een internationaal samenwerkingsproject met als doel het in kaart brengen van alle humane cell types.

Gedurende dit doctoraatsproject heb ik niet alleen deze inspirerende ontwikkelingen van nabij kunnen volgen, maar evenzeer zelf wetenschappelijke bijdragen kunnen aanleveren door het uitgebreid evalueren van nieuwe computationele tools en door het ontwikkelen van verscheiden tools en algoritmen.

Een veelvoorkomend probleem bij de ontwikkeling van innovatieve computationele algoritmen is het gebrek aan voldoende beschikbare datasets om de performantie op een kwantitatieve wijze te kunnen beoordelen. Tijdens dit doctoraat ontwikkelde ik een flexibele en uitbreidbare *in silico* simulator voor individuele cellen, met als doel huidige en toekomstige computationele onderzoeksvelden te ondersteunen en kick-starter projecten een duwtje in de rug te geven (Chapter 2).

Eveneens voerden we een uitgebreide studie van 45 *trajectvoorspellende* methoden uit (Hoofdstuk 3). Trajectvoorspellende methoden is een nieuwe soort van computationele analyse met als doel transities tussen verscheidene cellulaire toestanden te identificeren en te bestuderen. Terwijl deze klasse van methoden een van de snelst groeiende subdomeinen binnen de *single-cell omics* vormt, was er tot op dat moment nog geen kwantitatieve studie van de voor- en nadelen van dergelijke methoden uitgevoerd.

Ik ontwikkelde ook zelf vier nieuwe softwareprogramma's, waaronder een uitgebreide toolkit om trajecten te voorspellen, te visualiseren en te interpreteren (Hoofdstuk 5), een robuust programma om lineaire trajecten te voorspellen (Hoofdstuk 4), een nieuwe manier om genregulatie op het niveau van individuele cellen of stalen te bestuderen (Hoofdstuk 6), en een programma om topologische veranderingen ten gevolge van kleine veranderingen in een netwerk te kwantificeren (Hoofdstuk 7).

Ik reflecteer op huidige uitdagingen binnen het veld, door richtlijnen aan te bieden aan ontwikkelaars over hoe grootschalige benchmarkingsexperimenten van computationele methoden kunnen worden uitgevoerd (Hoofdstuk 8).

Gedurende dit project heb ik vooral genoten van het werken in een *open-source* omgeving, en het samenwerken met nationale en internationale collega's uit het veld om onopgeloste problemen aan te pakken. Door de broncode van excellente softwareprojecten te bestuderen, leerde ik mezelf de aanbevolen werkwijzen aan.

1 | Introduction

1.1 The cell

1

The cell is the smallest unit of life, of which all known living organisms are composed. Every cell houses a plethora of biomolecular processes that allows it to continuously adapt to changes in its environment. Due to the dynamic nature of these processes, it can be very challenging to comprehend the cellular response to a signal. A reductionist approach to understanding a complex biological system is to study the biochemical components of which it is comprised [1].

Recent advances in experimental technologies are playing a crucial role in reductionist biology, allowing to measure the abundance of thousands of different biochemical molecules in tens of thousands of individual cells. With it comes the challenge of analysing large amounts of data that are not easily interpretable by hand. The sheer volume of the data generated from such highly-integrative and high-throughput experiments are not the only reason why they are so challenging to interpret. For instance, the generated data contains high levels of noise arising from inherent biomolecular stochasticity in the cells and from the experimental profiling techniques used, as well as batch effects arising from differences between donors and labs [2]. Biologists thus turn to computer scientists to develop new tools to tackle these problems and help them to extract meaningful biological insights from the data. In this work, incremental contributions were made to the field in order to be able to address the aforementioned problems in a more comprehensive context.

Observing the biomolecular insides of cells can ultimately provide fundamental understanding into the processes that govern these cells and help uncover novel approaches for disease diagnosis, prognosis, and treatment. For example, the Human Cell Atlas (HCA) consortium [3] has set out to develop a comprehensive reference map of all the different types of cells in the human body. Experts in the field often metaphorically describe the HCA initiative as aiming to develop a 'Google Maps' of the human body. Even in its infancy, the HCA has profiled 3.8 million cells from 248 donors across 42 labs [4], and this number is likely to increase well above one hundred million.

The next part of the chapter highlights several key concepts in both cell biology and computer science, upon which the remainder of this work relies.

1.1.1 The origin of life and the RNA world

The discovery of the double helix shape of deoxyribonucleic acid (DNA) [5] is often considered the pivot point in our understanding of the origin of life and evolution. By now, it is well known that DNA serves as a medium for storing the genetic information

required to reproduce a whole organism. With other words, the DNA of an organism contains the complete set of instructions required to build all of the biomolecular machinery present in its body.

Life (or cells) did not originate from DNA, however. A widely-accepted hypothesis states that life originates from its lesser-known cousin, ribonucleic acid (RNA). According to the RNA world hypothesis [6], the very first primitive cells used RNA both to store genetic information and to perform the chemical reactions required to sustain themselves (Figure 1.1). Only later did cells develop the ability to use the more chemically stable DNA molecules to self-sustain in a process commonly referred to as the central dogma.

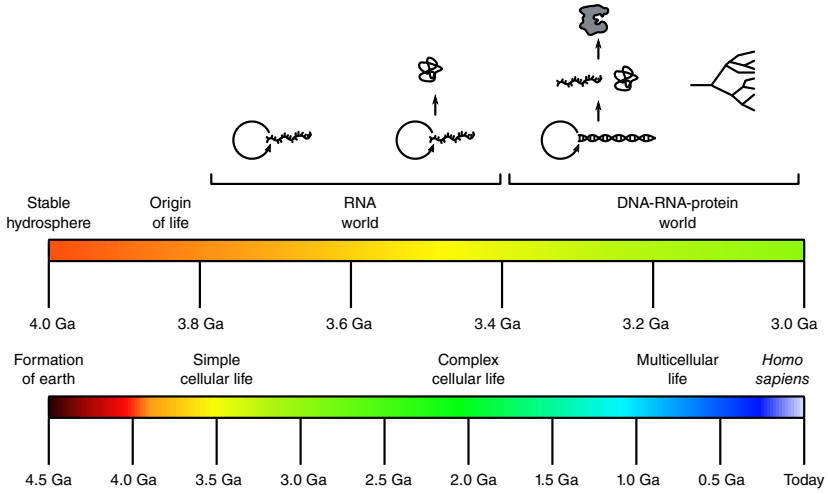


Figure 1.1: RNA world. The postulated rise and fall of the RNA world during the evolution of life, from early self-replicating RNA to complex, RNA-controlled metabolism, to the invention of translation, followed by diversification of all modern branches of life. Adapted from Horning (2011) [7].

1.1.2 Central dogma

The central dogma describes the general flow of genetic information in almost all existing living cells: DNA is decoded to RNA, which in turn encodes proteins [8]. Main processes involved in the central dogma are **transcription**, **splicing**, and **translation** (Figure 1.2).

During the process of **transcription** that takes place in the cell nucleus, a complementary RNA copy is transcribed from the template DNA. The initial RNA transcript is a precursor messenger RNA (pre-mRNA) that needs to undergo series of maturation steps to ultimately form the mature messenger RNA (mRNA). This maturation includes pre-mRNA **splicing** to remove non-protein coding intervening sequences

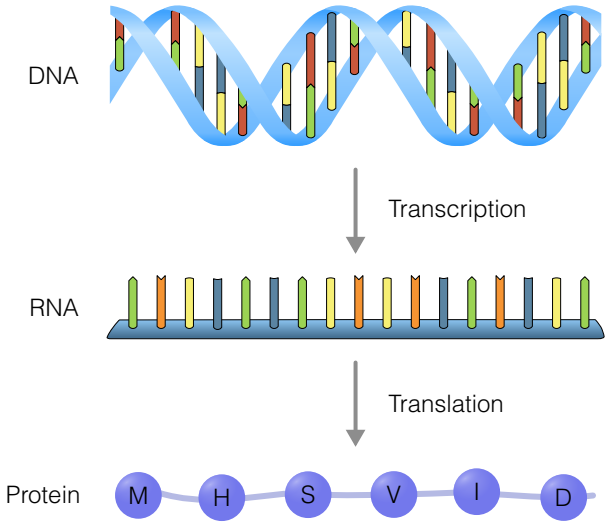


Figure 1.2: The central dogma of molecular biology explains the flow of genetic information, from DNA to RNA, to protein.

(the introns) and to join the neighbouring protein-coding sequences (the exons). A single pre-mRNA can be alternatively spliced to generate multiple forms of mRNAs that will result in the production of multiple protein isoforms. This process of alternative splicing is essential to generate more than 100'000 different proteins starting from just 20'000 genes [9].

The mature mRNA is then transported to the cytoplasm, where it engages with ribosomes to initiate **translation**. During this highly evolutionary conserved process, a chain of amino acids, known as the protein building blocks, is being synthesised. Each amino acid is specified by three nucleotides (a codon) in the mRNA, according to a nearly universal genetic code. After being released by the ribosomes, the translation product undergoes a variety of chemical modifications to form the final folded protein, the structure of which is determined by the sequence of different amino acids in the chain. In addition, polypeptides may be cleaved to yield more than one active polypeptide product. The structure of a protein determines its functionality, which includes catalysing biochemical reactions, providing structure, and transportation of molecules.

1.1.3 Cell types

Ever since Robert Hook first described the different structures of cells in 1665, biologists have been classifying cells into different "cell types", by grouping them according to form and function. The human body is said to contain more than 210 different cell types that are classified into four groups: epithelial, connective, muscle, and nervous.

This however, is a major underestimation of the real number of cell types. Neurons, for instance, that are known to be extremely diverse, are estimated to reach numbers above 10,000 different types [8].

The concept of cell types eases reasoning and our understanding about many aspects of biology (e.g. the process of cell differentiation, cell-cell communication, cellular response to certain stimuli). Some cells are known to be highly specialised toward performing a particular function (e.g. memory B cells accelerate immune response by remembering previously encountered pathogens), or they can maintain a strong ability to differentiate into other cell types.

One common approach for understanding the functionality of a particular cell is to observe which molecules are present in the cell and to associate those set of molecules with functionality. Taking a snapshot of the protein or RNA transcript content in a particular cell, might already provide us with major insights into its functionality. However, in order to fulfil a particular task, the biochemical machinery of the cell gradually changes over time. Therefore it is highly informative to also consider the transition states between cell types and the dynamic processes involved therein.

1.1.4 Cell dynamics and gene regulation

Cells are dynamic entities that can gradually produce the molecules needed to acquire new functionality. The naturally occurring cell-to-cell variability happens at the level of gene expression. Gene expression itself can be controlled at different levels (Figure 1.3), one of which is gene regulation by transcription.

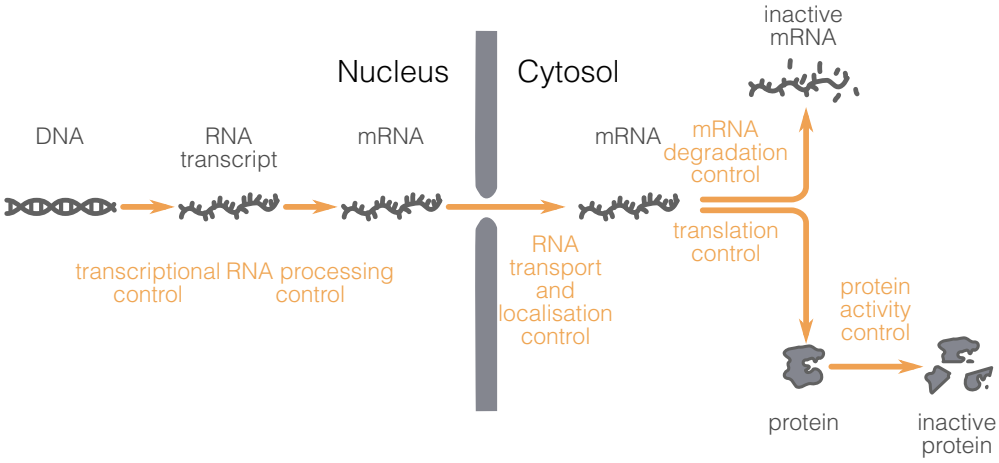


Figure 1.3: Levels of controlling gene expression can happen at the level of transcription, RNA processing (i.e. splicing), RNA transport and localization, mRNA translation, mRNA degradation and protein activity [10].

According to the needs of a cell, different genes are being transcribed. Housekeeping genes are being expressed in essentially every cell, while other genes are cell type or tissue specific or may be expressed in response to developmental and environmental signals [8].

Transcription factors (TFs) modulate the rate of gene transcription by binding and recruiting the transcriptional machinery to *cis*-regulatory regions (enhancers, and silencers) that are typically located in the promotor region of target genes. These bindings may result in increased or decreased gene expression. There are several TF families of which members share structural characteristics (e.g. zinc finger, helix-loop-helix).

Many TFs are commonly present in virtually all cell types (e.g. NF-κB), while others are specific for cells and developmental stages [11]. Typically, the same TF can regulate the rate of transcription of many target genes in different cell types, indicating that these gene regulatory networks (GRNs) are dynamic. Moreover, the production of a specific molecule might require several gene regulatory cascades. Studying the active parts of a cell's GRN can thus reveal which dynamic processes are taking place within a cell.

1.1.5 Profiling single cells

Several technologies are now available to profile (i.e. observe) biomolecular components, allowing us to gain better understanding in the biological processes that take place within a cell. The single-cell "omics" technologies originated from the convergence of two different fields, *"single-cell"* and *"omics"*.

Single-cell

The earliest approach for measuring the abundance of a particular molecule in *single cells* is the microscope. Since its development by Coons et al. (1941), immunohistochemistry (IHC) has been instrumental in visualising proteins [12]. A cell can present a particular type of protein, also called an antigen, on its cell surface. In many multicellular organisms, antigens can stimulate the immune system to produce antibodies. IHC realises the visualisation of proteins by exploiting the principle of antibodies binding to specific antigens.

IHC (and many other biotechnologies) visualises antigen-antibody reactions by attaching particular molecules to the antibody, such as an enzyme that catalyses a colour-producing reaction, or a fluorescent chemical compound that can re-emit

light upon excitation. The use of several colours (wavelengths) allows measuring expression levels of different antibodies simultaneously. Characterising cells in a semi-quantifiable way is labour intensive, however; since it involves acquiring an image of many cells and drawing a contour around each cell (called cell segmentation). Modern implementations of IHC improve the throughput drastically by using robots and computer software to provide semi-automated image acquisition and cell segmentation [13].

Flow cytometry [14] circumvents imaging and segmentation issues by measuring fluorescently labelled proteins as cells pass through a fluidic system. Since cells need to be suspended in a buffer, flow cytometry is particularly useful for analysing non-adherent cells such as the many different immune cells in blood. However, many protocols already exist to extract viable single cells from tissues and tumours [15]. Conventional flow cytometry devices enable to measure protein expression levels of millions of cells using up to eight different antibody fluorochromes simultaneously, while state-of-the-art instrumentation allows detection of up to 27 biomarkers simultaneously [16].

Besides IHC and flow cytometry, many new technologies have been developed which allow quantifying expression levels of molecules in single cells (e.g. mass cytometry, single-cell quantitative polymerase chain reaction, fluorescence *in situ* hybridization). All of these single-cell (non-omics) technologies are limited by the number of different molecules they measure, however. Selecting molecules of interest prior to analysis, makes the experiment biased towards the preconceptions of the experimenter.

Omics

On the other side of the spectrum are the so-called "omics" technologies. "Omics"[1] is a collective term for profiling all molecules of a particular type in a high-throughput manner. There are at least ten types of "omics". In this work, we mostly consider genomics, transcriptomics, proteomics, and regulomics. Genomics studies the complete DNA sequence of an organism's genome, while transcriptomics and proteomics study the RNA transcripts and proteins, respectively. Regulomics studies the regulatory molecules (e.g. genes, RNAs, proteins) which play a role in determining gene regulation.

Specific examples of omics technologies are whole genome sequencing to determine the DNA sequence of an organism, and RNA sequencing to profile the sequence of RNA transcripts, both using next-generation sequencing technologies. A gene expression profile can be obtained by mapping the sequences of RNA transcripts to the

[1] The etymology of "omics" is quite interesting [17].

genome.

1

Several high-throughput technologies have been developed to investigate proteomes in depth. The most commonly applied are mass spectrometry-based and gel-based techniques (e.g. differential in-gel electrophoresis).

Typically for these methods, to capture enough material to generate a profile, numerous cells need to be pooled and lysed together, thereby granting the technology's name "bulk" omics. Bulk omics is a major workhorse in molecular genetics and has applications in cancer research and in diagnostic screening of inheritable disorders.

Increasing evidence shows that cells are biomoleculary heterogeneous, even in very similar cell types [18] (Figure 1.4A). Since a bulk profile is a population average (or rather, a summation), important cell-to-cell variability is not discernible (Figure 1.4B).

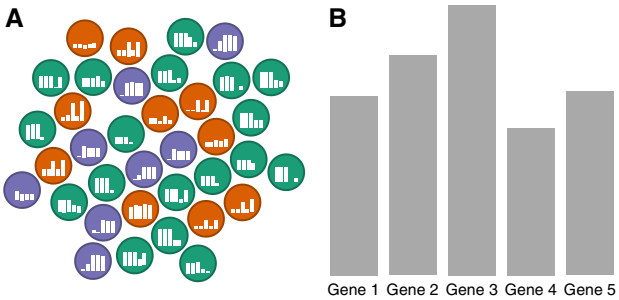


Figure 1.4: The 'masking' effect of bulk omics. A: Cells from several subpopulations are incorrectly assumed to be homogeneous and are profiled with a single bulk omics experiment. **B:** The signals from the different subpopulations are masked. The resulting profile is dissimilar from the majority of cells it is supposed to represent.

Single-cell omics

Comparing single-cell technologies with omics technologies shows that they have both clear advantages but also significant drawbacks (Figure 1.5A). Single-cell biology allows profiling thousands or even millions of cells, but only for a select number of genes. On the other hand, omics biology provides a broader view – since genes do not need to be selected beforehand – but is a profile of ensemble of cells and thus masks important cellular heterogeneity.

Advances in microvolume sequencing allowed profiling the transcriptome at single-cell resolution, thereby bringing single-cell biology and omics together to create single-cell omics. During the decade that followed, the number of single-cell omics technologies has sky-rocketed, allowing to profile >100'000 cells [19] and measuring other levels of information (e.g. protein abundance and spatial location) [20].

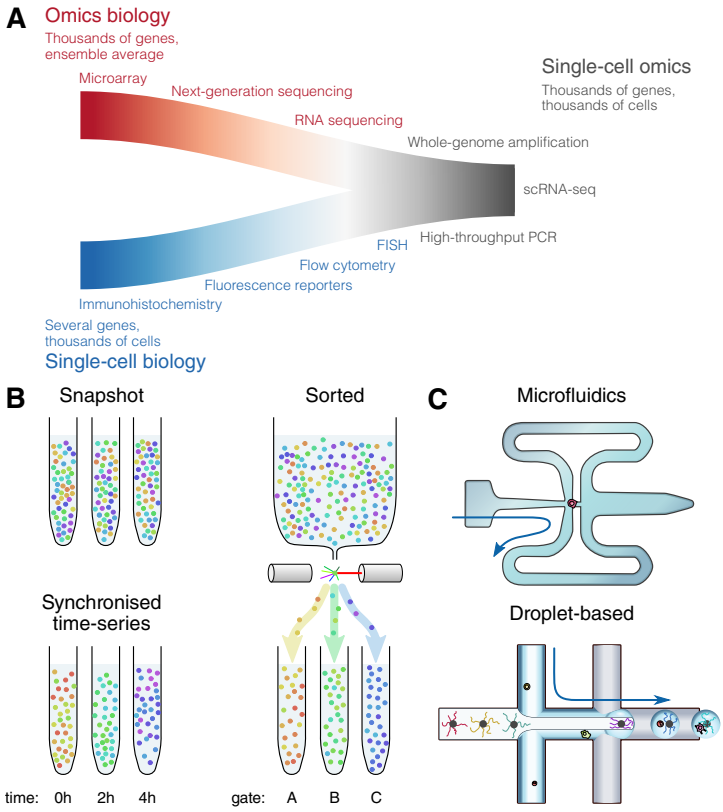


Figure 1.5: A: Technological improvements in microfluidics and low-volume sequencing were instrumental in the convergence of the fields of single-cell biology and omics biology. **B:** Different approaches for sampling cells with decreasing levels of cellular heterogeneity within the different sub-populations: snapshot, time-series, sorted. **C:** Two common single-cell RNA sequencing technologies. Microfluidics systems let cells travel through nanometer scale tubing, capturing individual cells at intersections. Droplet-based systems encapsulate individual cells in droplets.

In this work, unless noted otherwise, we will be working with transcriptomics data resulting from a single-cell RNA-sequencing experiment (scRNA-seq). The workflow of generating scRNA-seq profiles is as follows. Same as other single-cell (non-omics) profiling methods, cells first need to be isolated (Figure 1.5B). Different sampling techniques yield different levels of information about cellular state. By now, many protocols for extracting and tagging RNA from single cells have been developed [19], the most popular of which are based on microfluidics or droplets (Figure 1.5C). By sequencing the transcripts and the attached unique cell identifier tags, each read can be mapped and tallied up. scRNA-seq data can thus be summarised in a matrix, where each column represents a single cell, each row a gene, and each value represents the number of transcripts that were sequenced for that gene and cell.

The rapidly advancing field of single-cell omics harbours exceptional opportunities to discover new aspects of biology and redefine existing knowledge. Some of these

opportunities lie in efforts like the HCA consortium [3]. They have set out to redefine all human cell types in both terms of their gene expression and location, and as well as the developmental trajectories connecting the different cell types. As part of this endeavour, the consortium will likely profile the whole transcriptomes of tens or even hundreds of millions of cells [4].

1.2 Computational tools

Whole-genome profiling at single-cell level allows new types of analyses with which to study cellular heterogeneity at a hitherto unseen throughput. The new types of analyses permitted by single-cell omics present several computational challenges [21, 22, 23]. This necessitates the development of novel computational tools, either because the problem statement of the performed analysis is completely novel, or to adapt existing methodology to new data characteristics – dimensionality and noise.

scRNA-seq data is typically very sparse – while the human genome has more than 20'000 genes, they only contain non-zero values of a few thousand genes (typically less than 4'000). This is partially due to cells being specialised in particular functions and thus they do not need proteins of every time, but also due to RNA transcription occurring in bursts rather than continuously [24]. This contributes to the high levels of noise seen in scRNA-seq data: no two cells have the same set of non-zero genes.

Over the past five years, already 450 new tools have been developed to perform various analyses of single-cell omics data [25]. Most of these tools belong to one (or more) of the five main classes of computational methods used within single-cell omics (Figure 1.6). The main concepts and applications of these types of analyses are discussed in the following sections.

1.2.1 Dimensionality reduction

Single-cell omics datasets are usually one or more high-dimensional matrices, containing between M = 10^3 to 10^5 cells and typically about N = 10^3 to 10^4 genes (Figure 1.7A). The dimensionality of such datasets is typically too high for humans to interpret manually and for most modelling algorithms to tackle directly. Moreover, in reality, the intrinsic dimensionality of biological systems is much lower. For example, a differentiating hematopoietic cell could be described by just three dimensions: the first two dimensions lays out the hematopoietic lineage tree, and a third dimension allows for reprogramming between branches to occur.

Dimensionality reduction (DR) methods transform high-dimensional data into a mean-

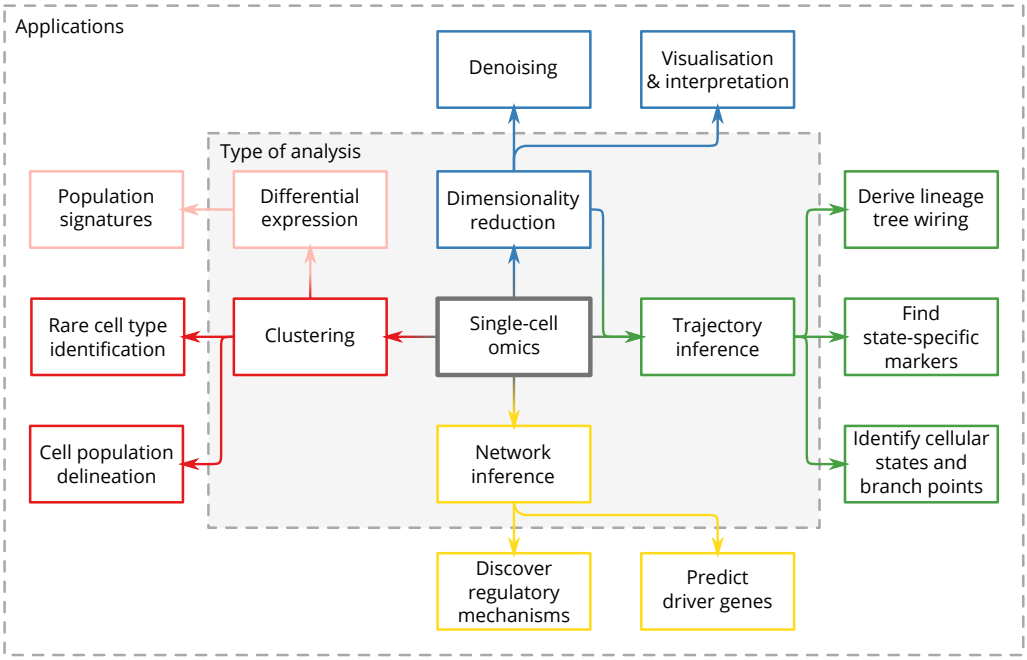


Figure 1.6: Five main types of analyses typically performed with single-cell omics, and the applications thereof.

ingful low-dimensional representation. DR methods are typically one of the first methods to apply on the data as a visual quality control by asserting that cells group together in ways that are expected.

DR methods have two main target audiences; computers – to construct a K-dimensional (with $K \ll N$) representation of the data such that pairwise distances between different samples are retained as well as possible (Figure 1.7B); and humans – to obtain a visual overview of the data (Figure 1.7C). DR methods are also commonly used to denoise the data, which is particularly useful since single-cell omics data is inherently noisy.

Each dimension frequently corresponds to one or more modules of co-expressed genes. The reduced space can be interpreted in way analogous to Waddington's epigenetic landscapes [26, 27, 28], where the landscape dictates the possible states a cell can reside in, and transitions between states correspond to dynamic cellular processes, such as cell differentiation.

DR methods can be classified into two main categories, feature projection-based and manifold learning [30].

Projection-based DR methods aim to perform a transformation of the data while preserving the pairwise distances between samples as much as possible. Examples of

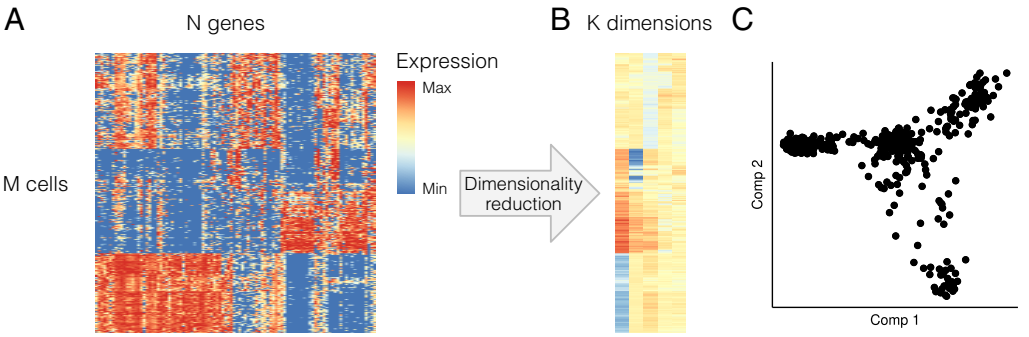


Figure 1.7: Dimensionality reduction for single-cell omics data. A: A heatmap visualisation of an scRNA-seq expression dataset of fibroblasts being reprogrammed to neuron cells [29]. Only the most variable **B:** The reduced space is a M×K-dimensional matrix which attempts to conserve the cellular heterogeneity of the original space as well as possible. **C:** A dot plot of the first two components of the reduced space provides a global overview of the cells in the dataset. Colouring the dots according to prior information (e.g. cell type) or gene expression provides insight into the cellular heterogeneity within the dataset.

commonly used projection-based DR methods in single-cell omics are Principal Component Analysis [31] (PCA) and Multi-Dimensional Scaling [32] (MDS).

Manifold learning DR methods reconstruct a higher-order structure in the original space (e.g. a graph or a grid), visualising the structure in a lower-dimensional space, and mapping the original samples to the lower-dimensional space. Manifold learning can be an iterative optimisation process using a predefined criterion. Examples of manifold learning techniques are t-distributed Stochastic Neighbor Embedding [33] (t-SNE), Diffusion Maps [34, 35] and Uniform Manifold Approximation and Projection [36] (UMAP).

For scalability reasons, this work mostly makes use of Landmark MDS [37, 38] (LMDS) with a Spearman rank correlation distance metric. LMDS is an extension of classical MDS, but rather than calculating a complete distance matrix between all pairs of cells, a set of landmark cells is sampled, only the distances between a set of landmarks and the samples are calculated.

1.2.2 Clustering

Clustering methods divide up the cells into separate groups of highly similar cells in order to delineate cell populations (Figure 1.8A). By visualising gene expression of known genes involved in the cell types of interest, the clusters can be annotated (Figure 1.8B). Particular clustering methods might be developed to better deal with imbalanced data in order to better identify rare cell type populations.

Usually, the number of clusters is determined by the user, either as a direct parameter

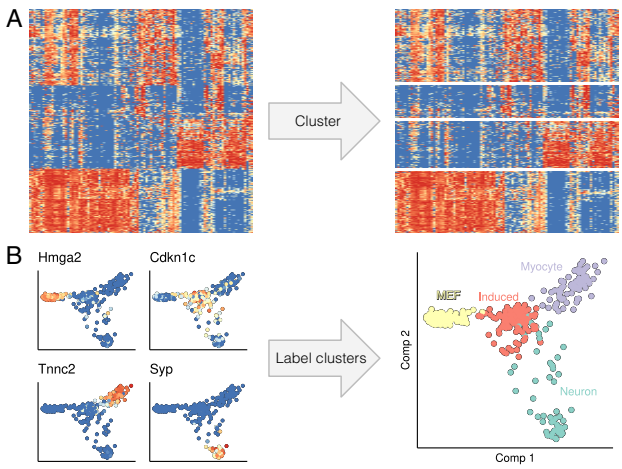


Figure 1.8: Clustering for single-cell omics data. A: Clustering methods group cells with similar omics profiles together. **B:** By overlaying gene expression levels on a dimensionality reduction, the clusters can be annotated to allow better interpretation of the cellular heterogeneity.

(e.g. k-means [39]) or an indirect parameter (e.g. a height cutoff in hierarchical clustering). In some exceptional cases, the number of clusters is strictly determined by the data itself and cannot be altered with a parameter (e.g. Louvain clustering [40]).

Clustering methods used in this work are mostly restricted to k-Means for clustering low-dimensional spaces and Louvain for clustering networks, since both are highly scalable with respect to the number of cells.

1.2.3 Trajectory inference

While clustering methods divide cells into distinct groups, trajectory inference (TI) methods acknowledge that cells are dynamic entities which transition from once cellular state to another via various dynamic processes. Rather than making distinct groups, TI methods attempt to predict how cells develop or undergo dynamic processes. The main applications of trajectory inference are to identify cellular states and branchpoints, reconstruct the topology of dynamic processes, and identify state-specific markers.

TI methods try to reconstruct the topology of a dynamic process as a trajectory and subsequently map the cells onto that trajectory. A trajectory is a graph where the nodes represent noteworthy cellular states, and each cell is predicted to be progressing along transitions between the different states (Figure 1.9A).

A trajectory can be visualised as a graph to better highlight the topology of the trajectory(Figure 1.9A middle), as a heatmap to better depict the changes in gene expression

along the different transitions (Figure 1.9A right), or may be embedded in a dimensionality reduction of the cells to better demonstrate cellular heterogeneity along the trajectory (Figure 1.9B right). Similar to clustering, by colouring the cells according to the expression of genes known to be involved in the dynamic process of interest, the milestones in the trajectory can be annotated (Figure 1.9B).

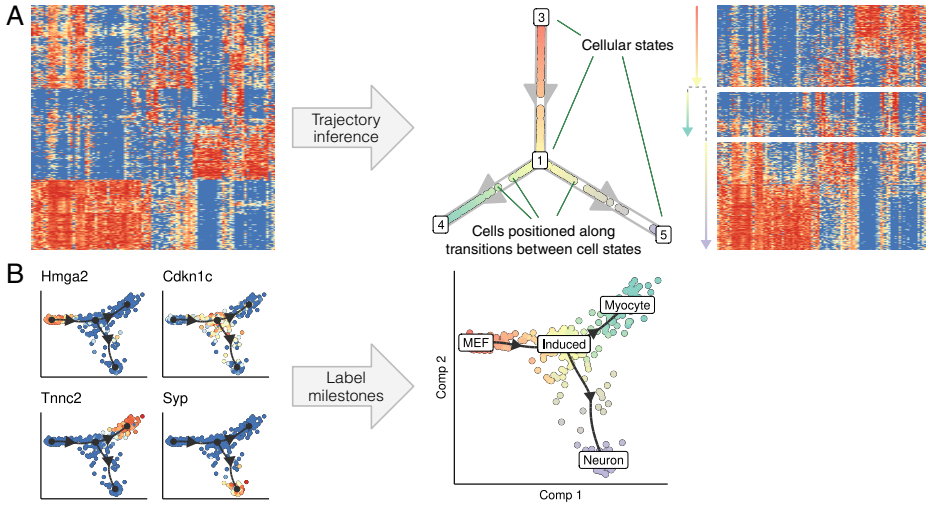


Figure 1.9: Trajectory inference for single-cell omics data. A: During a dynamic process cells pass through several transitional states, characterized by different waves of transcriptional, morphological, epigenomic and/or surface marker changes [41]. TI methods provide an unbiased approach to identifying and correctly ordering different transitional stages. **B:** By overlaying gene expression levels on a dimensionality reduction, the milestones can be annotated to allow better interpretation of the cellular heterogeneity.

Groups of TI methods use similar algorithms to infer a trajectory. By breaking down each method into its set of core algorithms, we can create a map of TI methodology [42] (Figure 1.10A), which is divided into two major classes. In the first step, dimensionality reduction techniques such as manifold learning, clustering, or graph-based methods are used to convert the dataset to a more simplified representation. This representation of the data then allows the trajectory itself to be more easily modelled in a second step. In this second step, the trajectory is modelled within the data using graph-based or curve-based approaches, after which the cells themselves can be ordered using a variety of methods.

A common way to classify TI methods is by the types of trajectories they can infer [43] (Figure 1.10B). About half of TI methods specialise in inferring linear or cyclic trajectories (i.e. they order the cells). Others model the trajectory as a rooted tree, allowing for one or more bifurcations to occur. Only a few methods are able to infer more generalised trajectories containing disconnected subgraphs or cycles.

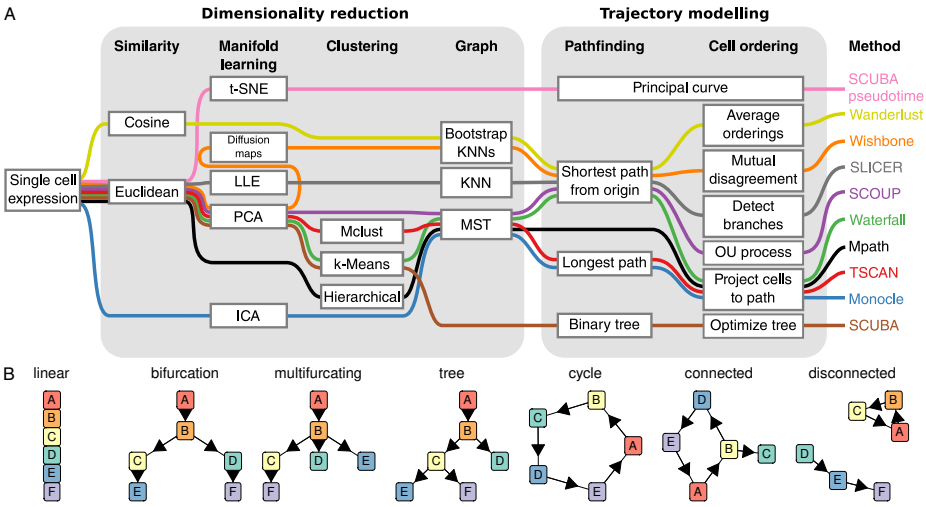


Figure 1.10: TI methods use several common building blocks and can be organized in a unifying modular framework. A: Most TI methods consist of two major steps, dimensionality reduction and trajectory modelling. TI methods require some form of dimensionality reduction in order to summarise cell heterogeneity in a lower dimensional space. Subsequently, a trajectory modelling step then operates in this reduced space, aiming to identify cell states, constructing a trajectory through the different states, and projecting the cells back on to the trajectory. **B:** TI methods can be classified according to the trajectory topologies they can infer.

1.2.4 Differential expression

Given that cells are split up two or more groups – either through clustering or prior knowledge – differential expression (DE) methods attempt to identify a group of genes that are expressed significantly higher or lower in one group in comparison to the others. Main applications of DE methods are the development of population signatures or simply the visualisation of the main difference between several groups in a compact manner (Figure 1.11A).

Trajectory differential expression (TDE) is an extension of DE where instead genes are prioritised according to whether their gene expression changes smoothly but significantly along a parts of a trajectory (Figure 1.11).

1.2.5 Network inference

One of the central cellular processes underlying development is transcriptional regulation. Modelling gene regulation is essential to better understand what drives cells to develop in the way that they do and to identify what goes wrong in the case of disease.

Network inference (NI) methods predict for every gene (or a subset thereof) by which

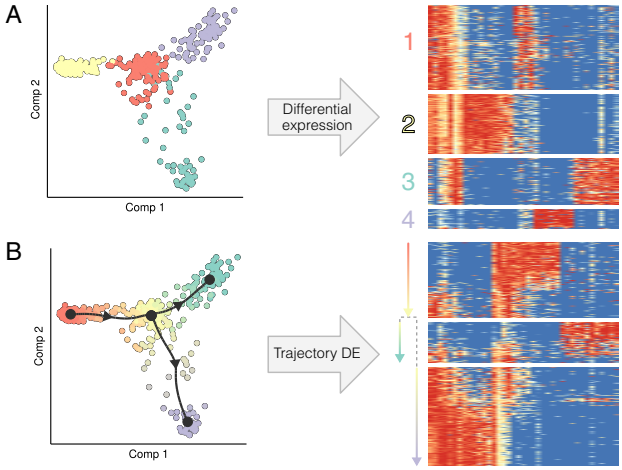


Figure 1.11: Differential expression for single-cell omics data. A: Differential expression methods prioritises genes that are expressed significantly higher or lower in particular given groups. **B:** Trajectory differential expression prioritises genes that change smoothly but significantly along particular transitions in a trajectory.

transcription factors they are regulated (Figure 1.12). The output of a network inference is a graph in which nodes represent genes and edges denote a regulatory interaction between a regulator and a target gene. Interactions can have two properties: its regulatory strength (a positive real value) and its effect (promoting or repressing).

Before single-cell omics, these methods rely on multiple experiments, amongst which perturbation and time-series experiments, to predict the effect each transcription factor has on the up- or downregulation of a gene. One of the main advantages of single-cell omics is the heterogeneity between cells caused by naturally occurring biological randomness [44] can be exploited to infer regulatory interactions between TFs and their target genes at much lower costs.

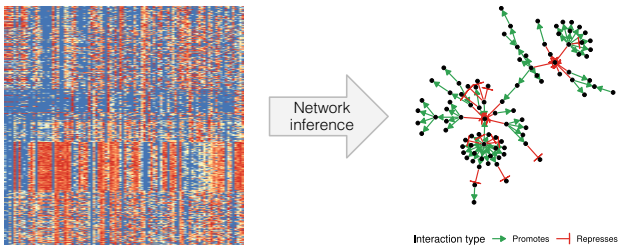


Figure 1.12: Network inference for single-cell omics data.

1.3 Benchmarking computational tools

Recent technological advancements in profiling single cells are having significant repercussions in many fields of biology. Profiling thousands of individual cells in a genome-wide manner provides opportunities to study cell heterogeneity and dynamics, for example inferring mechanisms for cellular development or intercellular communication. Hundreds of new software tools were developed [25] to perform these new types of analyses, or to fit existing analytical tools to deal with new data characteristics (e.g. differential expression, dimensionality reduction, normalisation).

One major shortcoming during the advent of single-cell omics was that the majority of newly developed computational tools were not quantitatively and comparatively evaluated. Rather, they relied on anecdotal evidence to demonstrate its usefulness.

As a demonstration, we perform a brief review of 75 articles introducing new TI methods, but the findings presented are generalisable to single-cell omics in general. Out of 75 articles, only about 37% of articles contain a self-assessment (Figure 1.13). Peer-reviewed articles fared even worse, self-assessing in only 35% of cases, whereas articles first published as a pre-print self-assess in 44% of cases.

While self-assessments are universally biased in favour of the authors [45] (intentionally or not), it is dangerous and unusual to publish a computational tool without quantitatively demonstrating its performance compared to state-of-the-art methods. Uncontrolled development of software tools without comprehensive benchmarking poses serious problems. For one, it slows down scientific progress. Every end-user needs to make a large commitment researching the domain in order to make an informed decision of which tool to use, or risk a higher incidence of false positive discoveries (either way, valuable resources are being wasted). In addition, it also negatively impacts the credibility of the field, thus discouraging potential users or researchers from entering.

This issue is likely not due to the tool developer's malevolence, but instead due to the absence of benchmarking methodology for this application, which can include standardised problem definitions, readily available benchmarking datasets, and suitable metrics. Each of these topics is discussed in more detail below.

1.3.1 Problem definition

One main reason why benchmarking TI methods is difficult is due to there being slight variations of the problem a method is attempting to solve. For example, a TI method might infer linear or cyclic trajectories, or predict the probability of a cell ending up in

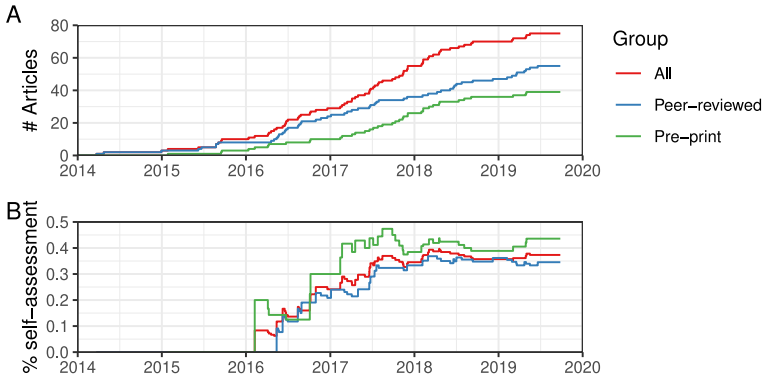


Figure 1.13: Less than half of all TI articles perform quantitative self-assessment. A: Since 2016, the number of TI articles has been increasing rapidly. Note that TI methods with both a pre-print and a peer-reviewed article only count once in the overall tally. **B:** Less than 50% of articles feature a self-assessment. Peer-reviewed articles self-assess only in 34% of cases.

one of several end states.

These differences reflect themselves in the input and output interfaces of each method, making it harder to directly compare methods directly. By constructing a generalised version that encompasses the different sub-problems and wrapping each method accordingly allows to fix the interface to make methods directly comparable.

Additionally, differences in problem definitions might complicate knowledge discovery to find similar methods, as certain articles might only show up with specific search terms. For the discoverability of a new TI method, it is therefore essential to use include generalised terminology, or at least list it as one of the keywords.

1.3.2 Datasets

During the infancy of the relatively domain of single-cell omics, large well-annotated datasets that can be used to benchmark novel computational tools are scarce and expensive to generate. When adequate benchmarking datasets are lowly abundant, synthetic data is often used to evaluate computational methods, either standalone or to complement real data.

Benefits of synthetic data are that they offer more control over the data characteristics and that they can be generated in large quantities. This allows evaluating the performance of a method in function of a changing parameter (e.g. dataset size or noise levels), which provides information on how well the method will work on real datasets.

A common counterargument of synthetic data is that they generate unrealistic datasets and thus provide no additional value in evaluating a method. In contrast, we

argue that a good set of synthetic datasets should allow benchmarkers to verify that a method should *at least* work well on the synthetic datasets, but good performance on synthetic datasets does not guarantee good performance on real datasets.

1.3.3 Metrics

To evaluate a computational tools, a quantitative metric is needed to compare the often complex multi-layered data structures of the ground-truth dataset to predictions thereof, or to compare the data structures of multiple predictions. When adequate metrics are lacking, it is advisable to re-purpose metrics from other domains and adapt them, if necessary.

A second important criterion is to evaluate the robustness of predictions by comparing multiple executions of the same method. Computing the robustness does not replace the necessity of a relevant metric that captures whether a predicted trajectory resembles the ground truth.

1.4 Research context and objectives

Technological breakthroughs in single-cell omics profiling have resulted in a wealth of information on the biomolecular contents of individual cells. In this work, we aim to speed up scientific discovery by developing and benchmarking computational tools for generating high-quality hypotheses in a high-throughput and data-driven manner.

The objectives of this work can be reduced to three main questions. How do cells change over time? What drives cells to change? How good are these predictions?

We tackle these three questions using three respective approaches, namely trajectory inference, network inference, and benchmarking (Figure 1.14). Each of the chapters in this work contributes to answering a part of one or more of these questions.

Trajectory inference We develop a simulator of developing single cells (Chapter 2). We use synthetic data to complement real data in benchmarking TI methods in terms of accuracy, stability, scalability and usability (Chapter 3). Based on these results, we provide guidelines to end-users on how to perform trajectory inference analysis. We introduce a novel TI method specialised in inferring linear trajectories which outperforms state-of-the-art existing methods (Chapter 4). Finally, we provide a full-stack toolkit for performing inferring, visualising and interpreting trajectories from more than 50 different TI methods (Chapter 5).

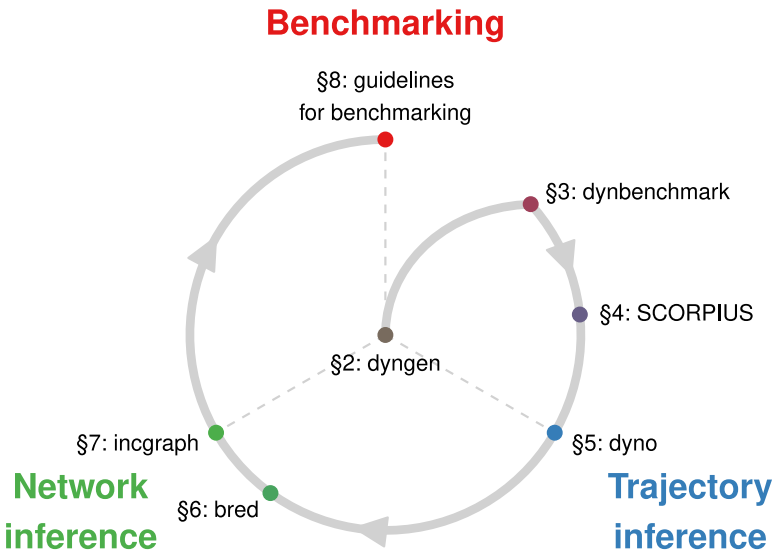


Figure 1.14: Overview

Network inference We explore a novel algorithm for inferring gene regulatory interactions at a single-cell level (Chapter 6). We demonstrate our NI methodology by inferring regulatory interactions for 14'963 profiles of cancer patients. We provide a tool for analysing the topological properties of large, evolving networks and use this to iteratively optimise GRN predictions (Chapter 7).

Benchmarking Using our synthetic single-cell generator, we develop benchmarking strategies for assessing the performance of novel types of single-cell omics tools which are currently in a conceptual or prototypical state (Chapter 2). We summarise our experience in benchmarking computational methods by providing a list of essential guidelines in how to benchmark computational tools (Chapter 8).

1.5 Hippocratic oath for method developers

Before setting out on this perilous journey (i.e. the rest of this dissertation), I wrote a Hippocratic oath which I believe all developers of computational tools should swear by before haphazardly developing and publishing new software tools.

- In developing a new computational tool, I swear to fulfil, to the best of my ability, the following items:
- I will write software that works reliably but fails gracefully when it does not.
- I will assert that my method produces accurate and reproducible results.
- I will validate the performance of my tool with previously unseen data from third-party sources.
- I will use suitable quantitative performance metrics. If possible, I use multiple metrics to avoid overfitting.
- I will compare the performance to relevant methods.
- I will report positive achievements of my method, but also be critical of it, and discuss areas in which it performs sub-optimally
- I will write unit tests to ensure that each component in my tool works as intended. If the tool produces unexpected errors when tested on various edge cases, my tool will be perceived as dysfunctional by others.
- I will follow reproducible research best practices, by making code and data publicly available.

Figure 1.15: A Hippocratic oath for computational method developers. Inspired by Laplante[46] and lists of guidelines for benchmarking[45, 47]

1.6 References

[1] Ingo Brigandt and Alan Love. "Reductionism in Biology". In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N Zalta. Spring 201. Metaphysics Research Lab, Stanford University, 2017.

[2] Chung Chau Hon et al. "The Human Cell Atlas: Technical Approaches and Challenges". In: *Briefings in Functional Genomics* 17.4 (July 2018), pp. 283–294. ISSN: 20412657. DOI: 10.1093/bfgp/elx029.

[3] Aviv Regev et al. "The Human Cell Atlas White Paper". In: (Oct. 2018). URL: http://arxiv.org/abs/1810.05192.

[4] Human Cell Atlas consortium. *Human Cell Atlas Data Portal*. 2018. URL: https://data.humancellatlas.org (visited on 08/11/2019).

[5] James D Watson, Francis HC Crick, et al. "Molecular Structure of Nucleic Acids". In: *Nature* 171.4356 (1953), pp. 737–738.

[6] Bruce Alberts et al. "The RNA World and the Origins of Life". In: *Molecular Biology of the Cell. 4th edition* (2002). URL: https://www.ncbi.nlm.nih.gov/books/NBK26876/ (visited on 08/12/2019).

[7] David P. Horning. "RNA World". In: *Encyclopedia of Astrobiology*. Ed. by Muriel Gargaud et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 1466–1478. ISBN: 978-3-642-11274-4. DOI: 10.1007/978-3-642-11274-4_1740.

[8] T. Strachan, A. Read, and T. Strachan. "Human Molecular Genetics. 4th". In: *New York: Garland Science* (2011).

[9] Timothy W. Nilsen and Brenton R. Graveley. "Expansion of the Eukaryotic Proteome by Alternative Splicing". In: *Nature* 463.7280 (Jan. 1, 2010), pp. 457–463. ISSN: 1476-4687. DOI: 10.1038/nature08909.

[10] Bruce Alberts et al. "An Overview of Gene Control". In: *Molecular Biology of the Cell. 4th Edition*. Garland Science, 2002.

[11] Samuel A. Lambert et al. "The Human Transcription Factors". In: *Cell* 172.4 (Feb. 8, 2018), pp. 650–665. ISSN: 0092-8674. DOI: 10.1016/j.cell.2018.01.029.

[12] Albert H Coons, Hugh J Creech, and R Norman Jones. "Immunological Properties of an Antibody Containing a Fluorescent Group.". In: *Proceedings of the Society for Experimental Biology and Medicine* 47.2 (1941), pp. 200–202.

[13] Zenonas Theodosiou et al. "Automated Analysis of FISH and Immunohistochemistry Images: A Review". In: *Cytometry Part A* 71A.7 (July 1, 2007), pp. 439–450. ISSN: 1552-4922. DOI: 10.1002/cyto.a.20409.

[14] M. J. Fulwyler. "Electronic Separation of Biological Cells by Volume". In: *Science* 150.3698 (1965), pp. 910–911. ISSN: 0036-8075. DOI: 10.1126/science.150.3698.910.

[15] Nalin Leelatian et al. "Preparing Viable Single Cells from Human Tissue and Tumors for Cytomic Analysis". In: *Current Protocols in Molecular Biology* 118.1 (Apr. 1, 2017), pp. 25C.1.1–25C.1.23. ISSN: 1934-3639. DOI: 10.1002/cpmb.37.

[16] Andrea Cossarizza et al. "Guidelines for the Use of Flow Cytometry and Cell Sorting in Immunological Studies". In: *European Journal of Immunology* 47.10 (Oct. 1, 2017), pp. 1584–1797. ISSN: 0014-2980. DOI: 10.1002/eji.201646632.

[17] Satya P. Yadav. "The Wholeness in Suffix -Omics, -Omes, and the Word Om". In: *Journal of Biomolecular Techniques : JBT* 18.5 (Dec. 2007), p. 277. ISSN: 1524-0215. pmid: 18166670. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2392988/ (visited on 08/15/2019).

[18] Byungjin Hwang, Ji Hyun Lee, and Duhee Bang. "Single-Cell RNA Sequencing Technologies and Bioinformatics Pipelines". In: *Experimental & Molecular Medicine* 50.8 (Aug. 7, 2018), p. 96. ISSN: 2092-6413. DOI: 10.1038/s12276-018-0071-8.

[19] Valentine Svensson, Roser Vento-Tormo, and Sarah A Teichmann. "Exponential Scaling of Single-Cell RNA-Seq in the Past Decade". In: *Nature Protocols* 13.4 (Apr. 2018), pp. 599–604. ISSN: 1750-2799. DOI: 10.1038/nprot.2017.149.

[20] Arnav Moudgil. *Multimodal scRNA-Seq*. Feb. 25, 2019. DOI: 10.5281/zenodo.2628012.

[21] Oliver Stegle, Sarah A. Teichmann, and John C. Marioni. "Computational and Analytical Challenges in Single-Cell Transcriptomics". In: *Nature Reviews Genetics* 16.3 (Mar. 2015), pp. 133–145. ISSN: 1471-0064. DOI: 10.1038/nrg3833.

[22] Guo-Cheng Yuan et al. "Challenges and Emerging Directions in Single-Cell Analysis". In: *Genome Biology* 18.1 (May 8, 2017), p. 84. ISSN: 1474-760X. DOI: 10.1186/s13059-017-1218-y.

[23] Geng Chen, Baitang Ning, and Tieliu Shi. "Single-Cell RNA-Seq Technologies and Related Computational Data Analysis". In: *Frontiers in Genetics* 10 (2019). ISSN: 1664-8021. DOI: 10.3389/fgene.2019.00317.

[24] Damien Nicolas, Nick E. Phillips, and Felix Naef. "What Shapes Eukaryotic Transcriptional Bursting?". In: *Molecular BioSystems* 13.7 (2017), pp. 1280–1290. ISSN: 1742-206X. DOI: 10.1039/C7MB00154A.

[25] Luke Zappia, Belinda Phipson, and Alicia Oshlack. "Exploring the Single-Cell RNA-Seq Analysis Landscape with the scRNA-Tools Database". In: *PLOS Computational Biology* 14.6 (June 2018), e1006245. ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1006245.

[26] Conrad Hal Waddington et al. "The Strategy of the Genes. A Discussion of Some Aspects of Theoretical Biology. With an Appendix by H. Kacser". In: *The strategy of the genes. A discussion of some aspects of theoretical biology. With an appendix by H. Kacser.* (1957), pp. ix+–262.

[27] James E Ferrell. "Bistability, Bifurcations, and Waddington's Epigenetic Landscape". In: *Current Biology* 22.11 (June 2012), R458–R466. ISSN: 0960-9822. DOI: 10.1016/j.cub.2012.03.045.

[28] Jonathan A. Rebhahn et al. "An Animated Landscape Representation of CD4+ T-Cell Differentiation, Variability, and Plasticity: Insights into the Behavior of Populations versus Cells". In: *European Journal of Immunology* 44.8 (Aug. 1, 2014), pp. 2216–2229. ISSN: 0014-2980. DOI: 10.1002/eji.201444645.

[29] Barbara Treutlein et al. "Dissecting Direct Reprogramming from Fibroblast to Neuron Using Single-Cell RNA-Seq". In: *Nature* 534.7607 (2016), pp. 391–395.

[30] Daniel Engel, Lars Hüttenberger, and Bernd Hamann. "A Survey of Dimension Reduction Methods for High-Dimensional Data Analysis and Visualization". In: *Visualization of Large and Unstructured Data Sets: Applications in Geospatial Planning, Modeling and Engineering - Proceedings of IRTG 1131 Workshop 2011*. Ed. by Christoph Garth, Ariane Middel, and Hans Hagen. Vol. 27. OpenAccess Series in Informatics (OASIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2012, pp. 135–149. ISBN: 978-3-939897-46-0. DOI: 10.4230/OASIcs.VLUDS.2011.135.

[31] Karl Pearson. "LIII. On Lines and Planes of Closest Fit to Systems of Points in Space". In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2.11 (Nov. 1, 1901), pp. 559–572. ISSN: 1941-5982. DOI: 10.1080/14786440109462720.

[32] J. B. Kruskal. "Multidimensional Scaling by Optimizing Goodness of Fit to a Nonmetric Hypothesis". In: *Psychometrika* 29.1 (Mar. 1964), pp. 1–27. ISSN: 0033-3123, 1860-0980. DOI: 10.1007/BF02289565.

[33] Laurens van der Maaten and Geoffrey Hinton. "Visualizing Data Using T-SNE". In: *The Journal of Machine Learning Research* 9.2579-2605 (2008), p. 85.

[34] Boaz Nadler et al. "Diffusion Maps, Spectral Clustering and Eigenfunctions of Fokker Planck Operations". In: *Proceedings of the 18th International Conference on Information Processing Systems*. NIPS'05. Cambridge, MA, USA: MIT Press, 2005, pp. 955–962.

[35] Ronald R. Coifman and Stéphane Lafon. "Diffusion Maps". In: *Applied and Computational Harmonic Analysis* 21.1 (July 2006), pp. 5–30. ISSN: 10635203. DOI: 10.1016/j.acha.2006.04.006.

[36] Leland McInnes, John Healy, and James Melville. "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction". In: (2018).

[37] Vin D. Silva and Joshua B. Tenenbaum. "Global Versus Local Methods in Nonlinear Dimensionality Reduction". In: *Advances in Neural Information Processing Systems 15*. Ed. by S. Thrun and K. Obermayer. Cambridge, MA: MIT Press, 2002, pp. 705–712.

[38] Seunghak Lee and Seungjin Choi. "Landmark MDS Ensemble". In: *Pattern Recognition* 42.9 (Sept. 2009), pp. 2045–2053. ISSN: 00313203. DOI: 10.1016/j.patcog.2008.11.039.

[39] Stuart Lloyd. "Least Squares Quantization in PCM". In: *IEEE Transactions on Information Theory* 28.2 (Mar. 1982), pp. 129–137. DOI: 10.1109/TIT.1982.1056489.

[40] Vincent D Blondel et al. "Fast Unfolding of Communities in Large Networks". In: *Journal of Statistical Mechanics: Theory and Experiment* 2008.10 (Oct. 9, 2008), P10008. ISSN: 1742-5468. DOI: 10.1088/1742-5468/2008/10/p10008.

[41] Tariq Enver et al. "Stem Cell States, Fates, and the Rules of Attraction". In: *Cell Stem Cell* 4.5 (May 8, 2009), pp. 387–397. ISSN: 1875-9777. DOI: 10.1016/j.stem.2009.04.011. pmid: 19427289.

[42] Robrecht Cannoodt, Wouter Saelens, and Yvan Saeys. "Computational Methods for Trajectory Inference from Single-Cell Transcriptomics". In: *European Journal of Immunology* 46.11 (Nov. 1, 2016), pp. 2496–2506. ISSN: 1521-4141. DOI: 10.1002/eji.201646347.

[43] Wouter Saelens et al. "A Comparison of Single-Cell Trajectory Inference Methods". In: *Nature Biotechnology* 37 (May 2019). ISSN: 15461696. DOI: 10.1038/s41587-019-0071-9.

[44] Olivia Padovan-Merhar and Arjun Raj. "Using Variability in Gene Expression as a Tool for Studying Gene Regulation". In: *Wiley Interdisciplinary Reviews. Systems Biology and Medicine* 5.6 (Nov. 2013), pp. 751–759. ISSN: 1939-005X. DOI: 10.1002/wsbm.1243. pmid: 23996796.

[45] Raquel Norel, John Jeremy Rice, and Gustavo Stolovitzky. "The Self-Assessment Trap: Can We All Be Better than Average?". In: *Molecular systems biology* 7.1 (2011), p. 537. ISSN: 1744-4292. DOI: 10.1038/msb.2011.70. pmid: 21988833.

[46] Phillip A Laplante. "First, Do No Harm: A Hippocratic Oath for Software Developers". In: *Queue* 2.4 (2004), p. 14.

[47] Lukas M. Weber et al. "Essential Guidelines for Computational Method Benchmarking". In: *Genome Biology* 20.1 (June 20, 2019), p. 125. ISSN: 1474-760X. DOI: 10.1186/s13059-019-1738-8.

2 | dyngen: Benchmarking with *in silico* single cells

Abstract

Purpose: A common problem of pioneering computational tools is that during their development, there are rarely sufficient datasets available for adequately quantitatively assessing its performance.
Results: We developed dyngen, a multi-modality simulator of single cells. In dyngen, the biomolecular state of an *in silico* cell changes over time according to a gene regulatory network. The simulator is extendible for adding new modalities or experimental procedures. We demonstrate dyngen's flexibility by applying it to benchmark RNA velocity and casewise network inference methods.
Conclusion: dyngen lays the foundations for benchmarking a wide variety of computational single-cell tools and can be used to help kick-start the development of future types of analyses.

Publication status

Manuscript in preparation.
Cannoodt R*, Saelens W*, Deconinck L, and Saeys Y.
* Equal contribution

Author contributions

- **R.C.** and W.S. designed the study.
- **R.C.**, W.S., and L.D. performed the experiments and analysed the data.
- **R.C.** and W.S. implemented software packages.
- **R.C.** and W.S. wrote the original manuscript.
- **R.C.**, W.S., L.D., and Y.S. reviewed and edited the manuscript.
- Y.S. supervised the project.

2.1 Introduction

Continuous technological advancements to single-cells omics are having profound effects on how researchers can validate biological hypotheses. Early experimental technologies typically only allowed profiling a single modality (e.g. DNA sequence, RNA or protein expression). However, recent developments permit profiling multiple modalities simultaneously, and every modality added allows for new types of analyses that can be performed.

This presents method developers with a problem. The majority of the 250+ peer-reviewed computational tools for analysing single cell omics data were published without a quantitative assessment of the accuracy of the tool. This is partially due to low availability of suitable benchmarking datasets; even if there are sufficient suitable input datasets available, these are often not accompanied by the necessary metadata to serve as ground-truth for a benchmark.

Here, synthetic data plays a crucial role in asserting minimum performance requirements for novel tools in anticipation of adequate real data. Generators of scRNA-seq data (e.g. splatter [1], powsimR [2], PROSSTT [3] and SymSim [4]) have already been widely used to explore the strengths and weaknesses of computational tools, both by method developers [5, 6, 7, 8] and independent benchmarkers [9, 10, 11]. However, a limitation of scRNA-seq profiles generators is that they would require significant methodological alterations to add additional modalities or experimental conditions, thereby limiting the applicability of each generator.

An ideal experiment would be able to observe all aspects of a cell, including a full history of its molecular states, spatial positions and environmental interactions [12]. While this falls outside the reach of current experimental technologies, generating synthetic data in anticipation of new experimental technologies would allow already developing the next wave of computational tools.

We developed a multi-modality simulator of single cells called dyngen (Figure 2.1A). dyngen uses Gillespie's stochastic simulation algorithm [13] to simulate gene regulation, splicing and translation at a single-molecule level. Its methodology allows tracking of many layers of information throughout the simulation, including the abundance of any molecule in the cell, progression of the cell along a dynamic process, and the activation strength of individual regulatory interactions. dyngen can simulate a large variety of dynamic processes (e.g. cyclic, branching, disconnected) as well as a broad range of experimental conditions (e.g. batch effects and time-series, perturbation and knockdown experiments). The fine-grained controls over simulation parameters allow dyngen to be applicable to a broad range of use-cases. We demonstrate this by performing first quantitative evaluations of two types of novel computational ap-

Hello there! Did you stumble across this phrase by accident,
or are you looking for what I think you're looking for?

proaches, namely RNA velocity and casewise network inference.

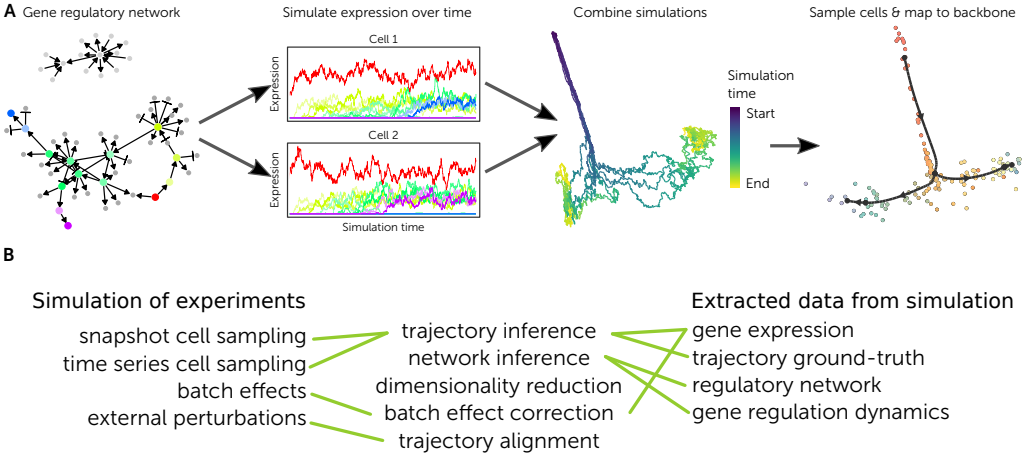


Figure 2.1: Showcase of dyngen functionality. **A:** The typical process of generating a dataset with dyngen. **B:** Evaluating different types of computational tools requires simulating different types of experiments and extracting different layers of information from the simulation.

2.2 Results

A cell consists of a set of molecules, the abundance of which are affected by a set of reactions: transcription, splicing, translation, and degradation (Figure 2.2A). A gene regulatory network (GRN) defines the reactions that are allowed to occur (Figure 2.2B), which is constructed in such a way that cells slowly develop over time (Figure 2.2C,D). With every time step dt in the simulation, the probability of a reaction occurring is computed (not shown). From the probabilities are sampled which reactions occur during this time step dt (Figure 2.2E).

dyngen returns many modalities throughout the whole simulation: molecular abundance, cellular state, number of reaction firings, reaction likelihoods, and regulation activations (Figure 2.2C–F). These modalities can serve both as input data and ground truth for benchmarking many types of computational approaches. For example, a network inference method could use mRNA abundance and cellular states as inputs and its output could be benchmarked against the gold standard GRN.

Depending on how the GRN is designed, different cellular developmental processes can be simulated. dyngen includes generators of GRNs which result in many different developmental topologies (Figure 2.3), including branching, converging, cyclic and even disconnected. Custom-defined GRNs offer more fine-grained control over the simulation.

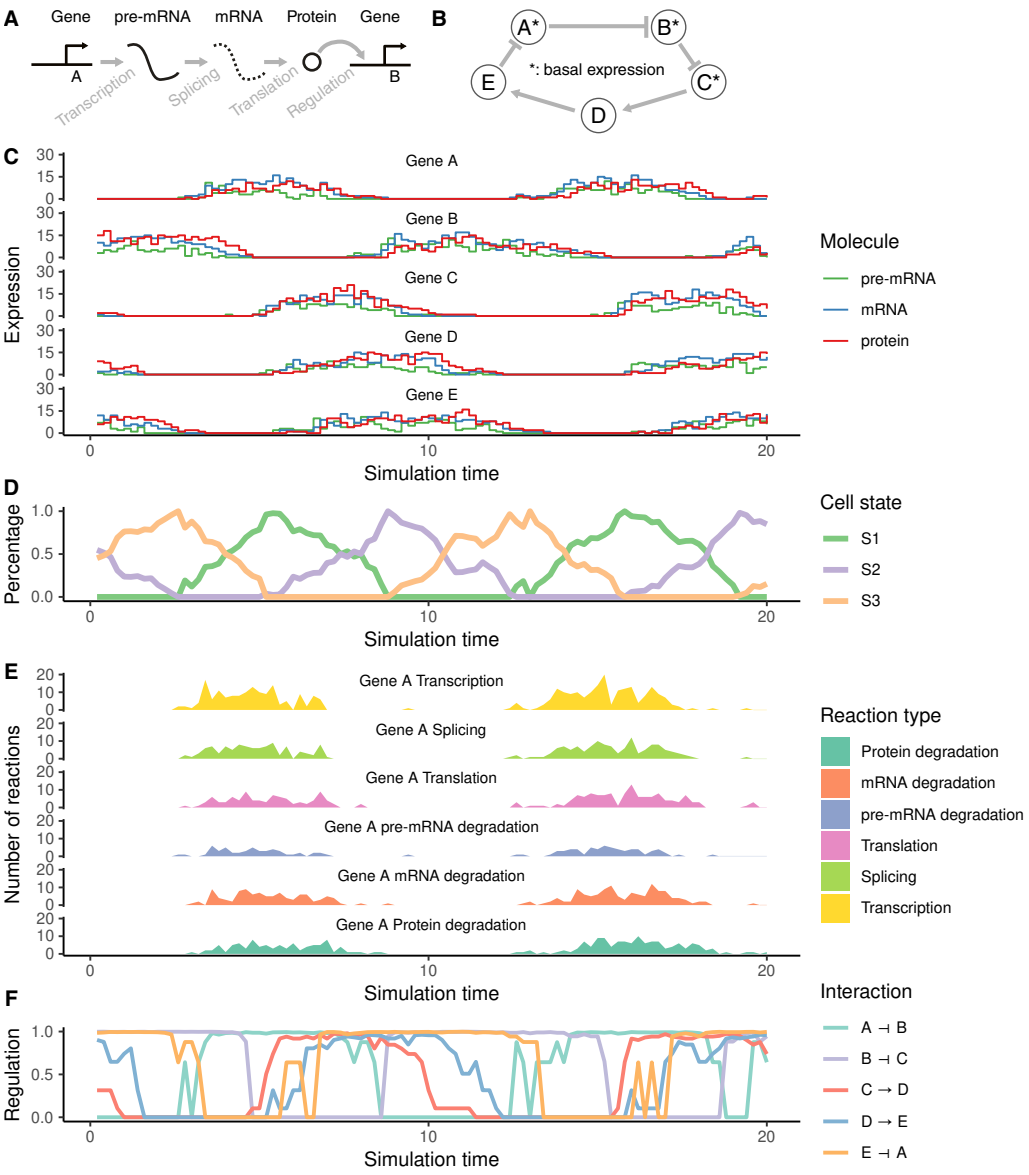


Figure 2.2: dyngen models reactions at a single-molecule level and keeps track of multiple levels of information throughput a simulation. A: Changes in abundance levels are driven strictly these gene regulatory reactions. **B:** The input GRN is defined such that it models a dynamic process of interest. **C:** The reactions define how abundance levels of molecules change at any particular timepoint. **D:** Firing many reactions can significantly alter the cellular state over time. **E:** dyngen keeps track of the reactions that were fired during small intervals of time. **F:** Similarly, dyngen can also keep track of the regulatory activity of every interaction.

Together, these qualities allow it to be applicable in benchmarking a broad range of use-cases. In practice, dyngen has already successfully been used to evaluate trajectory inference [10], trajectory-based differential expression [14], and network in-

a thousand times over for their continued love, sacrifices and support.
So here we go, until the end of the book: Thank you! Thank you!

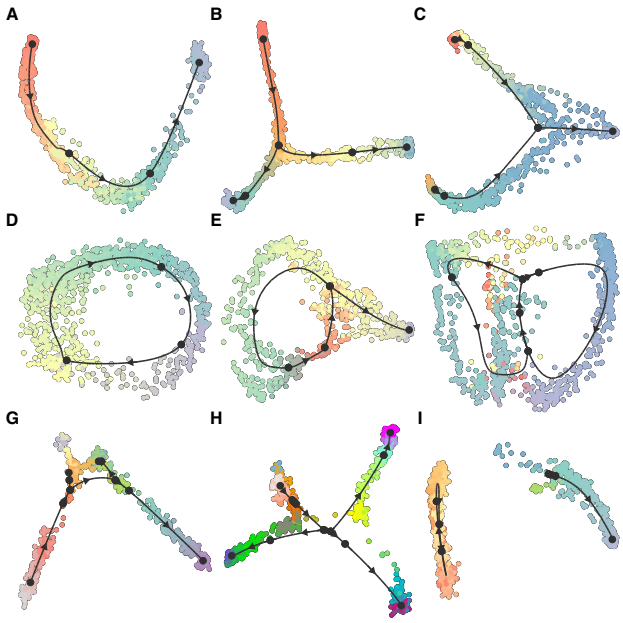


Figure 2.3: Multiple executions of dyngen with different predefined backbones. From each simulation of about 200 genes, 1000 cells were sampled. **A:** Linear. **B:** Bifurcating. **C:** Converging. **D:** Cyclic. **E:** Bifurcating loop. **F:** Bifurcating converging. **G:** Consecutive branching. **H:** Binary tree. **I:** Disconnected.

ference [15] methods. To demonstrate this point even further, we apply dyngen on several promising novel computational approaches which have recently had a major impact on how single-cell analyses are performed but for which quantitative assessment of the performance was hitherto lacking.

2.2.1 RNA velocity

In eukaryotes, a gene is first transcribed to a pre-mRNA, and subsequently spliced into mature mRNA. Because reads coming from both unspliced and spliced transcripts are observed in expression data, the relative ratio between the two can tell us something about which genes are increasing, decreasing or remaining the same [16, 17]. To determine this, some parameters have to be estimated to determine which fraction of unspliced and spliced mRNAs corresponds to an increase or decrease. The estimation of these parameters makes some assumptions, and can be handled in different ways in the two main algorithms that are now available for RNA velocity estimation: velocyto [17] and scvelo [18]. It can be difficult to obtain ground truth data to benchmark these algorithms, given that it would require continuous data of transcriptional dynamics in individual cells. On the other hand, the ground truth velocity is rapidly extracted from the dyngen model, by looking whether each transcript is currently increasing or

decreasing in expression.

We tested scvelo and velocyto on 8 datasets containing linear, bifurcating, disconnected and cyclic trajectories, and varied the main parameter settings in which they estimate the velocity. We found that the original velocyto implementation, which assumes that the velocity remains constant in some cells, performed the best across all datasets. The dynamical estimation of velocyto, as implemented in scvelo, performed the worst of all parameter settings. This was mainly due to scvelo overestimating the dynamics of a gene, especially towards upregulation, while velocyto correctly estimated not only when a gene changes, but also when it remained in a steady state.

2.2.2 Casewise network inference

Casewise network inference (CNI) methods[1] predict not only which transcription factors regulate which target genes (Figure 2.5A, top left), but also how active each interaction is in every case (Figure 2.5A). In CNI, a 'case' might be a cell, but it might also refer to a bulk sample.

While a few pioneering CNI approaches have already been developed [19, 20, 21], a quantitative assessment of the performance is hitherto lacking. This is not surprising, as neither real nor in silico datasets of cell-specific or even cell-type-specific interactions exists that is large enough so that it can be used as a ground-truth for evaluating CNI methods.

Since dyngen computes tracks the probability of transcription, temporarily 'knocking down' the expression of a regulator and observing the change in transcription probability. This is a much more accurate ground-truth for regulatory interaction between a regulator and target in comparison to observing the change in transcript abundance levels when knocking down a regulator, as the regulation of the target could be indirect.

We used this ground-truth to compare the performance of three CNI methods (Figure 2.5B). We calculated the AUROC and AUPR score – which are common metrics for NI benchmarking – for each cell individually. Computing the mean AUROC and AUPR per dataset showed that pySCENIC significantly outperforms LIONESS and SSN.

This comparison could be extended to include analyses on the scalability of execution time w.r.t dataset size, the stability of the results in function of noise, and the usability toward end users.

[1] Other terms are commonly used when dealing with data from a particular source. For example, single-cell NI when applied to single-cell transcriptomics data; sample-specific NI when applied to bulk transcriptomics; patient-derived NI when applied to bulk profiles of patients.

Thank you! Thank you! Thank you! Thank you! Thank you! Thank you!
Thank you! Thank you! Thank you! Thank you! Thank you! Thank you!

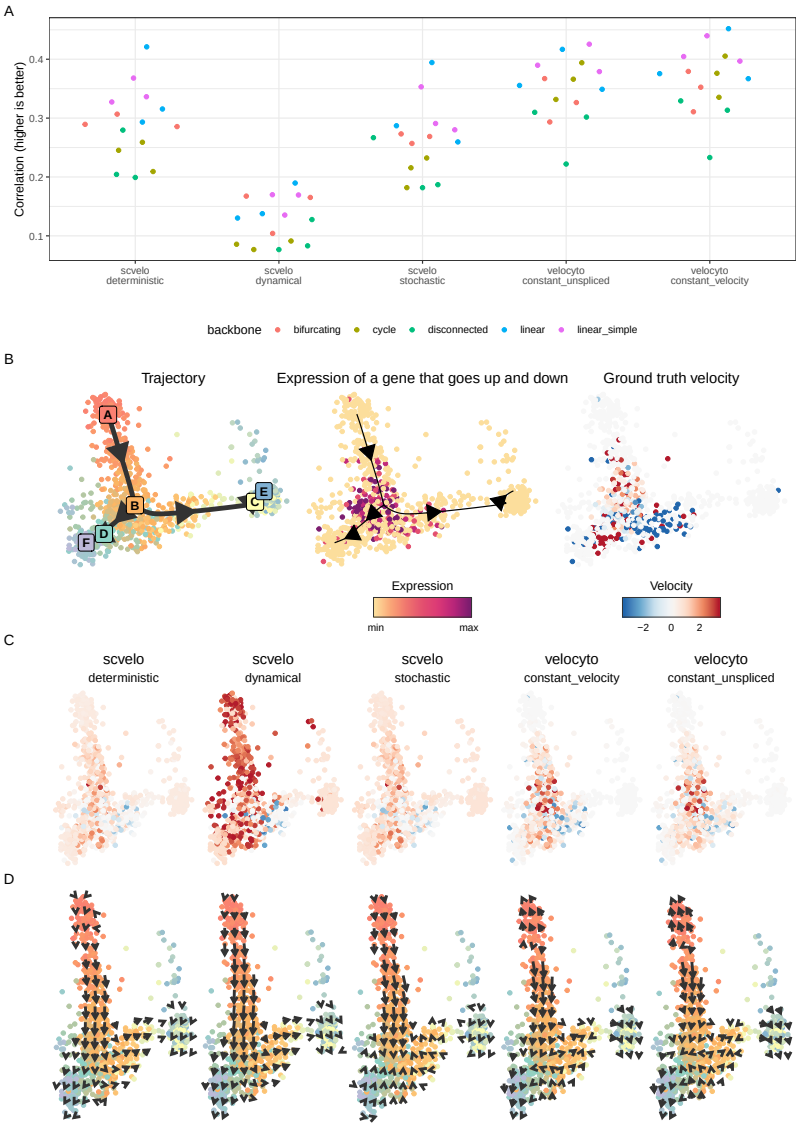


Figure 2.4: dyngen allows benchmarking of RNA velocity methods. A: We tested five different methods and parameter settings for the estimation of RNA velocity on datasets with varying backbones (colours). Overall, the velocyto method with the constant velocity assumption performed the best overall. **B:** An example bifurcating dataset, with as illustration the expression and ground truth velocity of a gene that goes up and down in the trajectory. **C:** The RNA velocity estimates of the different methods. **D:** The embedded RNA velocity of the different methods.

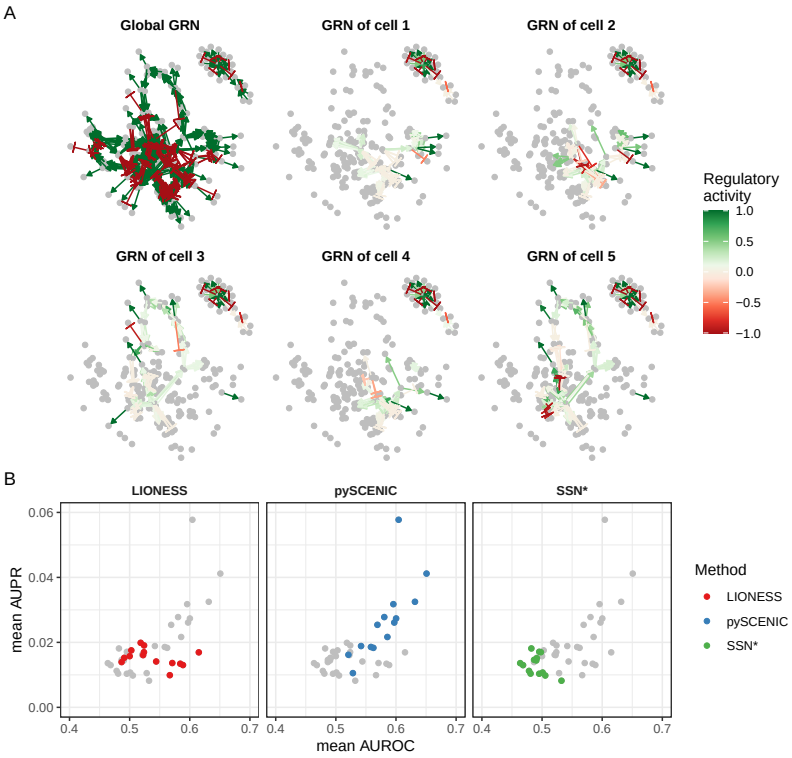


Figure 2.5: dyngen allows benchmarking Casewise Network Inference (CNI) methods. A: A cell is simulated using the global gene regulatory network (GRN, top left). However, at any particular state in the simulation, only a fraction of the gene regulatory interactions are active. **B:** CNI methods were executed to predict the regulatory interactions that are active in each cell specifically. Using the ground-truth casewise GRN, the performance of each method was quantified on 14 dyngen datasets.

2.3 Discussion

As is, dyngen's single cell simulations can be used to evaluate common single-cell omics computational methods such as clustering, batch correction, trajectory inference and network inference. However, the combined effect of these advantages results in a framework that is flexible enough to adapt to a broad range of applications. This may include methods that integrate clustering, network inference and trajectory inference. In this respect, dyngen may promote the development of new tools in the single-cell field similarly as other simulators have done in the past [22, 23].

dyngen ultimately allows anticipating technological developments in single-cell multi-omics. In this way, it is possible to design and evaluate the performance and robustness of new types of computational analyses before experimental data becomes available. In addition, it could also be used to compare which experimental protocol is the

Go to the next chapter for more acknowledgements.

most cost-effective in producing the qualitative and robust results in downstream analysis.

Currently, dyngen focuses on simulating cells as standalone entities that are well mixed. Splitting up the simulation space into separate subvolumes could pave the way to better study key cellular processes such as cell division, intercellular communication and migration [24].

2.4 Methods

The workflow to generate *in silico* single cell data consists of six main steps (Figure 2.6).

2.4.1 Defining the backbone: modules and states

One of the main processes involved in cellular dynamic processes is gene regulation, where regulatory cascades and feedback loops lead to progressive changes in expression and decision making. The exact way a cell chooses a certain path during its differentiation is still an active research field, although certain models have already emerged and been tested *in vivo*. One driver of bifurcation seems to be mutual antagonism, where two genes strongly repress each other [25, 26], forcing one of the two to become inactive [27]. Such mutual antagonism can be modelled and simulated [28, 29]. Although the two-gene model is simple and elegant, the reality is frequently more complex, with multiple genes (grouped into modules) repressing each other [30].

To start a dyngen simulation, the user needs to define a module network and a backbone. The module network defines how sets of co-regulated genes, called modules, regulate each other. The module network is what mainly determines which dynamic processes occur within the simulated cells. The backbone is a separate set of simulations in which the ground-truth topology of the dynamic processes are defined, as it is difficult to determine the topology of the dynamic processes from the module network itself.

A module network consists of modules connected together by regulatory interactions. A module may have basal expression, which means genes in this module will be transcribed without the presence of transcription factor molecules. A module marked as "active during the burn phase" means that this module will be allowed to generate expression of its genes during an initial warm-up phase (See section 2.4.5). At the end of the dyngen process, cells will not be sampled from the burn phase simulations. Inter-

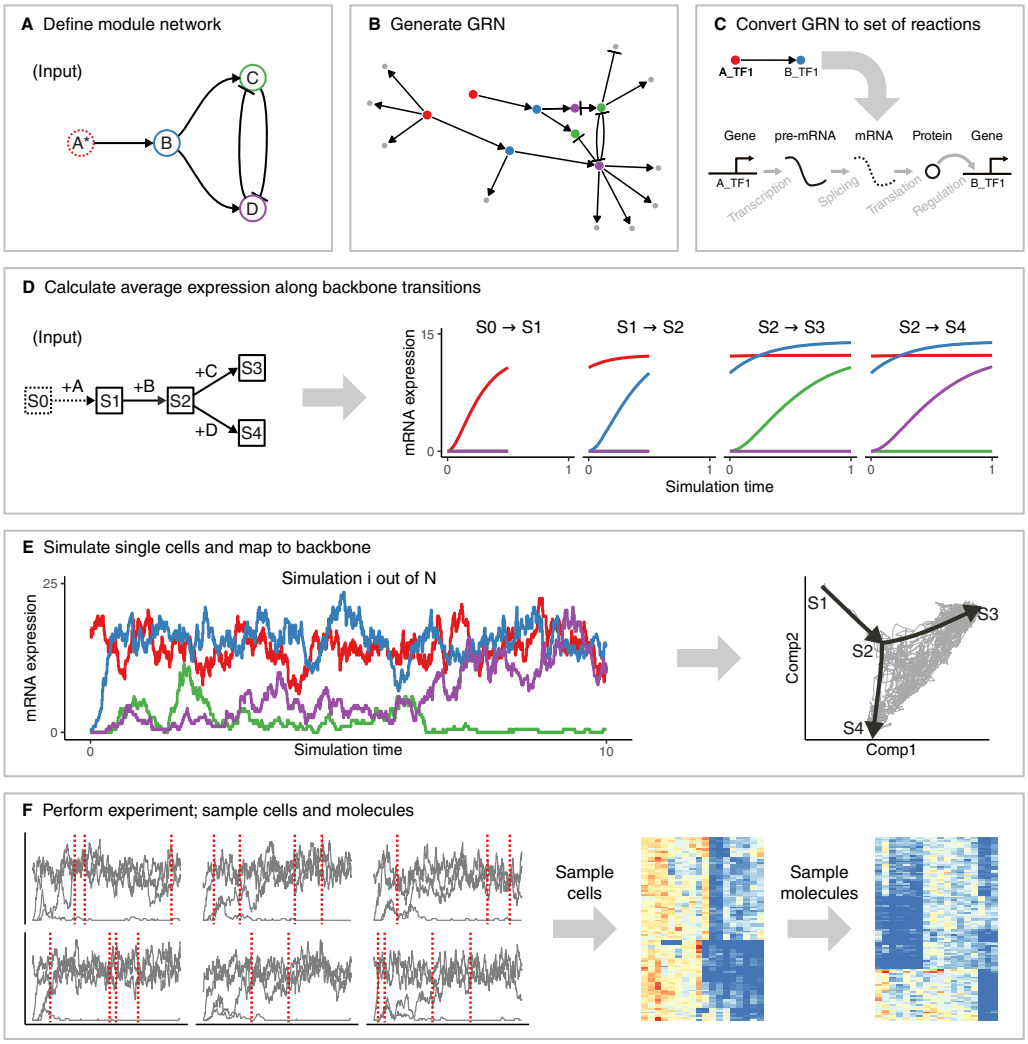


Figure 2.6: The workflow of dyngen is comprised of six main steps. A: The user needs to specify the desired module network or use a predefined module network. **B:** Each gene in a module is is regulated by one or more transcription factors from the upstream module. Additional target genes are generated. **C:** Each gene regulatory interaction in the GRN is converted to a set of biochemical reactions. **D:** Along with the module network, the user also needs to specify the backbone structure of expected cell states. The average expression of each edge in the backbone is simulated by activating a restricted set of genes for each edge. **E:** Multiple Gillespie SSA simulations are run using the reactions defined in step C. The counts of each of the molecules at each time step are extracted. Each time step is mapped to a point in the backbone. **F:** The molecule levels of multiple simulations are shown over time (left). From each simulation, multiple cells are sampled (from left to middle). Technical noise from profiling is simulated by sampling molecules from the set of molecules inside each cell (from middle to right).

actions between modules have a strength (which is a positive integer) and an effect (+1 for upregulating, -1 for downregulating).

Several examples of module networks are given (Figure 2.7). A simple chain of modules (where one module upregulates the next) results in a *linear* process. By having the last module repress the first module, the process becomes *cyclic*. Two modules repressing each other is the basis of a *bifurcating* process, though several chains of modules have to be attached in order to achieve progression before and after the bifurcation process. Finally, a *converging* process has a bifurcation occurring during the burn phase, after which any differences in module regulation is removed.

Note that these examples represent the bare minimum in terms of number of modules used. Using longer chains of modules is typically desired. In addition, the fate decisions made in this example of a bifurcation is reversible, meaning cells can be reprogrammed to go down a different differentiation path. If this effect is undesirable, more safeguards need to be put in place to prevent reprogramming from occurring (Section 2.4.1).

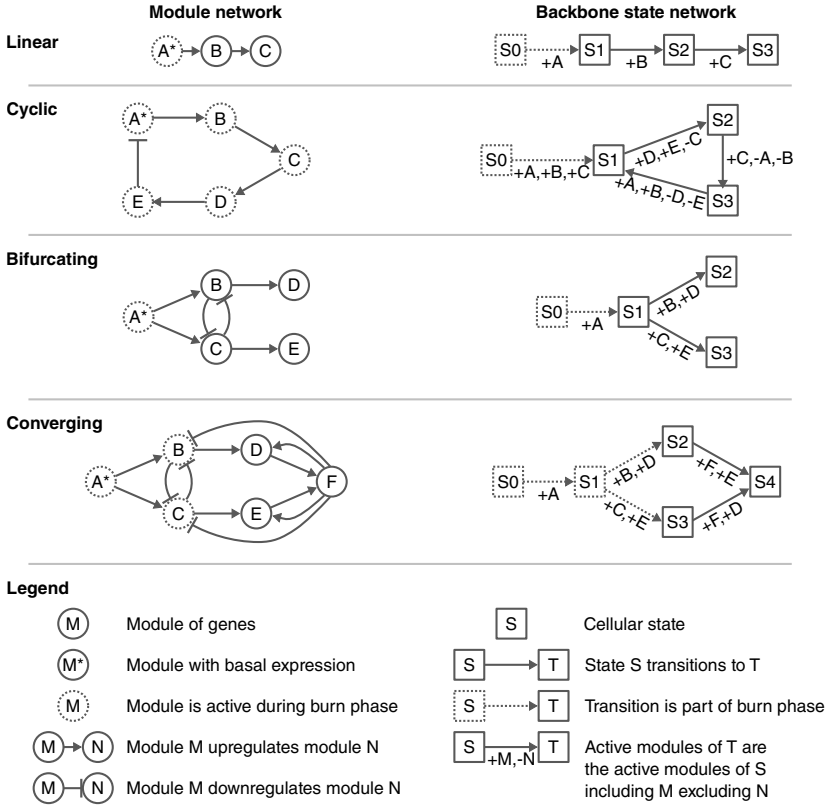


Figure 2.7: Example module networks.

In addition to the module network, the user also needs to define a network of cellular states called the "backbone". Before simulating any cells, each transition in the backbone is simulated separately to obtain the average changes in expression along

that transition (Figure 2.6D). As part of the backbone, the user needs to specify which modules are allowed to alter its expression from one state to another. For example, in order to transition from state S0 to S1 in the cyclic example, gene modules A, B and C are turned on and a simulation is allowed to run. To transition from S1 to S2, gene modules D and E are turned on, and expression of gene module C is kept constant. To transition from S2 to S3, C is turned on again and now A and B are fixed. Finally, to transition from S3 to S1 again, A and B are turned on again and D and E are fixed again. Demonstrations of the backbone will be explained in more detail in section 2.4.4.

Backbone lego

The backbone can make use of one or more "backbone lego" (BBL) pieces (Figure 2.8). A BBL consists of one or more modules which regulate each other such that the output modules present a specific behaviour, depending on the input module (Figure 2.8A). Parameters allow determining the number of modules involved in the process and the number of outputs. Multiple BBLs can be chained together in order to intuitively create module networks and corresponding state networks (Figure 2.8B). Note that not all dynamic processes can be represented by a combination of BBLs, but they can serve as common building blocks to aid the construction of the backbone.

When the input node of a **linear BBL** (Figure 2.8C) is upregulated, the module the BBL is connected to will be upregulated. A *simple chain* is a set of modules where a module upregulates the next. A *chain with double repression* has an uneven number of modules forming a chain where each module downregulates the next but all modules (except the input) have basal expression. A *grid with double repression* is similar; except that modules do not have basal expression but instead get upregulated by an upstream module in the chain. Finally, a *flip flop*} consists of a simple chain where first the modules (except the last) are upregulated. Once the second to last module is upregulated, that module upregulates itself and the first module is strongly repressed, causing all other modules to lose expression and finally the last module to be upregulated. The *flip flop* retains this output state, even when the input changes.

When the input node of a **branching BBL** (Figure 2.8D) is upregulated, a subset of its output modules will eventually be upregulated. A *simple branching* uses reciprocal inhibition to drive the upregulation of one of the output modules. Due to its simplicity, however, multiple output modules might be upregulated simultaneously and over long periods of simulation time it might be possible that the choice of upregulated module changes. A *robust branching* improves upon the simple branching by preventing upregulation of output modules until an internal branching decision has been made, and by repressing the decision mechanism to avoid other output modules

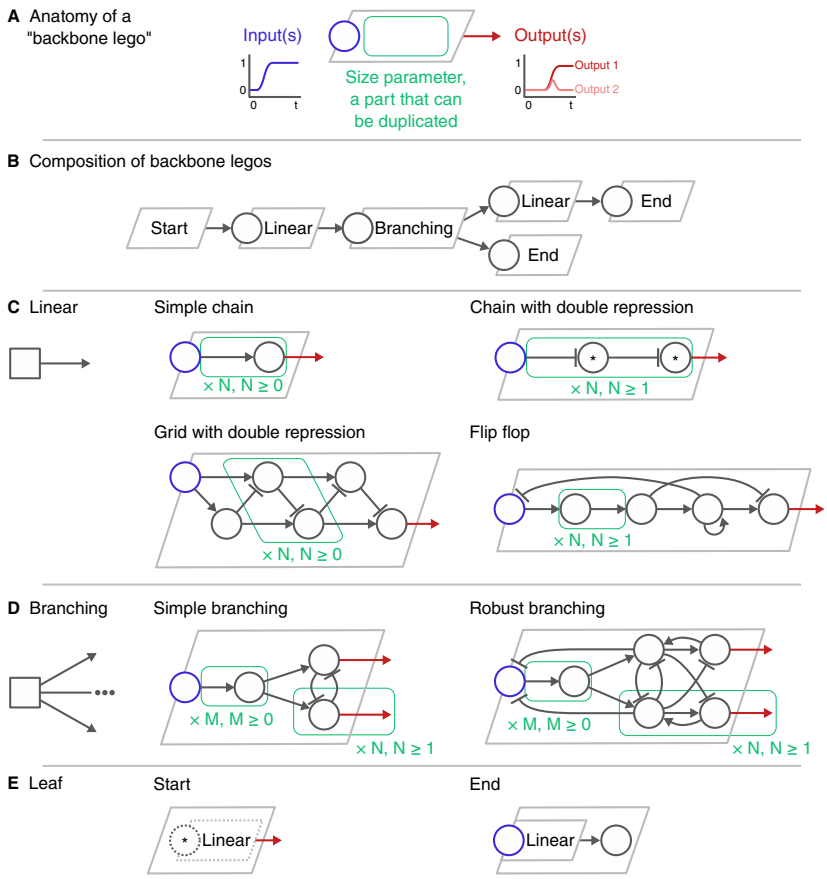


Figure 2.8: Reusable regulatory building blocks that can be used to construct GRNs that result in dynamic processes following an expected pattern. A: Each building block contains genes which serve as inputs that can be regulated by external transcription factors. The connections inside the building block transform the input signal in a particular way and has one or more output interactions of which the targets are not yet determined. **B:** By connecting multiple building blocks together, a backbone of regulatory modules can be formed. **C:** dyngen offers several variations of linear building blocks which allow for differing regulatory mechanisms. **D:** A branching module will upregulate the expression of just one of the output interactions. Two variations are given of which the simple variant is more prone to reprogramming events in comparison to the robust branching module. **E,F:** These components allow to specify the beginning (no inputs) and end (no outputs) of the backbone.

being upregulated other than the one that has been chosen.

A **leaf BBL** (Figure 2.8E) is a linear BBL that has either no inputs or no outputs. A *start* BBL is a linear BBL where the first module has basal expression, and all modules in this module will be active during the burn-in phase of the simulation (Section 2.4.4). An *end* BBL is also a linear BBL with its output regulating one final module.

2.4.2 Generating the gene regulatory network

The GRN is generated based on the given backbone in four main steps (Figure 2.9).

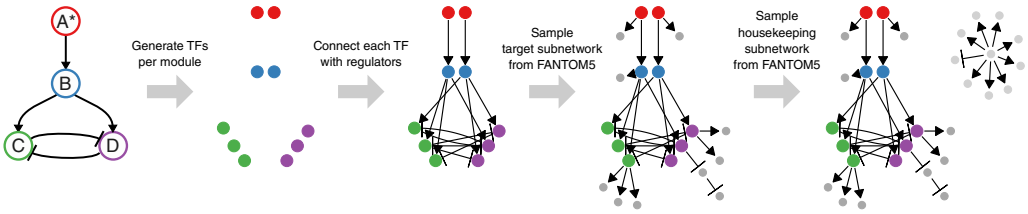


Figure 2.9: Generating the feature network from a backbone consists of four main steps.

Step 1, sampling the transcription factors (TF). The TFs are the main drivers of the molecular changes in the simulation. The user provides a backbone and the number of TFs to generate. Each TF is assigned to a module such that each module has at least x parameters (default $x = 1$). A TF inherits the 'burn' and 'basal expression' from the module it belongs to.

Step 2, generating the TF interactions. Let each TF be regulated according to the interactions in the backbone. These interactions inherit the effect, strength, and cooperativity parameters from the interactions in the backbone. A TF can only be regulated by other TFs or itself.

Step 3, sampling the target subnetwork. A user-defined number of target genes are added to the GRN. Target genes are regulated by a TF or another target gene, but is always downstream of at least one TF. To sample the interactions between target genes, one of the many FANTOM5 [31] GRNs is sampled. The currently existing TFs are mapped to regulators in the FANTOM5 GRN. The targets are drawn from the FANTOM5 GRN, weighted by their page rank value. For each target, at most x regulators are sampled from the induced FANTOM5 GRN (default $x = 5$). The interactions connecting a target gene and its regulators are added the GRN.

Step 4, sampling the housekeeping subnetwork. Housekeeping genes are completely separate from any TFs or target genes. A user-defined set of housekeeping genes are also sampled from the FANTOM5 GRN. The interactions of the FANTOM5 GRN are first subsampled such that the maximum in-degree of each gene is x (default $x = 5$). A random gene is sampled and a breadth-first-search is performed to sample the desired number of housekeeping genes.

2.4.3 Convert gene regulatory network to a set of reactions

Simulating a cell's GRN makes use of a stochastic framework which tracks the abundance levels of molecules over time in a discrete quantity. For every gene G, the abundance levels of three molecules are tracked, namely of corresponding pre-mRNAs, mature mRNAs and proteins, which are represented by the terms w_G, x_G and y_G respectively. The GRN defines how a reaction affects the abundance levels of molecules and how likely it will occur. Gibson and Bruck [32] provide a good introduction to modelling gene regulation with stochastic frameworks, on which many of the concepts below are based.

For every gene in the GRN a set of reactions are defined, namely transcription, splicing, translation, and degradation. Each reaction consists of a propensity function – a formula $f(.)$ to calculate the probability $f(.) \times \mathrm{d}t$ of it occurring during a time interval $\mathrm{d}t$ – and the effect – how it will affect the current state if triggered.

The effects of each reaction mimic the respective biological processes (Table 2.1, middle). Transcription of gene G results in the creation of a single pre-mRNA molecule w_G. Splicing turns one pre-mRNA w_G into a mature mRNA w_G. Translation uses a mature mRNA x_G to produce a protein y_G. Pre-mRNA, mRNA and protein degradation results in the removal of a w_G, x_G, and y_G molecule, respectively.

The propensity of all reactions except transcription are all linear functions (Table 2.1, right) of the abundance level of some molecule multiplied by a constant drawn from a normal distribution (Table 2.2). The propensity of transcription of a gene G depends on the abundance levels of its TFs.

Table 2.1: Reactions affecting the abundance levels of pre-mRNA w_G, mature mRNA x_G and proteins y_G of gene G. Define the set of regulators of G as R_G, the set of upregulating regulators of G as R_G^+, and the set of downregulating regulators of G as R_G^-. Parameters used in the propensity formulae are defined in Table 2.2.

Reaction	Effect	Propensity
Transcription	$\emptyset \rightarrow w_G$	$\mathrm{wpr}_G \times \frac{\mathrm{ba}_G - \mathrm{co}_G^{\lvert R_G^+ \rvert} + \prod_{H \in R_G^+} (\mathrm{co}_G + \chi_{G,H})}{\prod_{H \in R_G} (1 + \chi_{G,H})}$
Pre-mRNA degradation	$w_G \rightarrow \emptyset$	$\mathrm{wdr}_G \times w_G$
Splicing	$w_G \rightarrow x_G$	$\mathrm{wsr}_G \times w_G$
Mature mRNA degradation	$x_G \rightarrow \emptyset$	$\mathrm{xdr}_G \times x_G$
Translation	$x_G \rightarrow x_G + y_G$	$\mathrm{ypr}_G \times x_G$
Protein degradation	$y_G \rightarrow \emptyset$	$\mathrm{ydr}_G \times y_G$

The propensity of the transcription of a gene G is inspired by thermodynamic models of gene regulation [33], in which the promoter of G can be bound or unbound by a

Table 2.2: Default parameters defined for the calculation of reaction propensity functions.

Parameter	Symbol	Definition
Transcription rate	wpr_G	$\in N(50, 10),\ \geq 10$
Splicing rate	wsr_G	$\in N(5, 1),\ \geq 1$
Translation rate	ypr_G	$\in N(5, 1),\ \geq 1$
Pre-mRNA half-life	whl_G	$\in N(0.15, 0.03),\ \geq 0.05$
Mature mRNA half-life	xhl_G	$\in N(0.15, 0.03),\ \geq 0.05$
Protein half-life rate	yhl_G	$\in N(0.25, 0.05),\ \geq 0.1$
Interaction strength	$\text{str}_{G,H}$	$\in 10^{U(0,2)}$ *
Hill coefficient	$\text{hill}_{G,H}$	$\in U(0.5, 2)$ *
Cooperativity factor	co_G	$\in [0, 1]$ *
Pre-mRNA degradation rate	wdr_G	$= \ln(2)\ /\ \text{whl}_G$
Mature mRNA degradation rate	xdr_G	$= \ln(2)\ /\ \text{xhl}_G$
Protein degradation rate	ydr_G	$= \ln(2)\ /\ \text{yhl}_G$
Dissociation constant	dis_H	$= 0.5 \times \frac{\text{wpr}_H \times \text{wsr}_H \times \text{ypr}_H}{(\text{wdr}_H + \text{wsr}_H) \times \text{xdr}_H \times \text{ydr}_H}$
Binding	$\chi_{G,H}$	$= (\text{str}_{G,H} \times \text{y}_H\ /\ \text{dis}_H)^{\text{hill}_{G,H}}$
Basal expression	ba_G	$= \begin{cases} 1 & \text{if } \text{R}_G^+ = \emptyset \\ 0.0001 & \text{if } \text{R}_G^- = \emptyset \text{ and } \text{R}_G^+ \neq \emptyset \text{ *} \\ 0.5 & \text{otherwise} \end{cases}$

*: unless G is a TF, then the value is determined by the backbone.

set of N transcription factors H_i. Let $f(\text{y}_1, \text{y}_2, \ldots, \text{y}_N)$ denote the propensity function of G, in function of the abundance levels of the transcription factors. The following subsections explain and define the propensity function when $N = 1$, $N = 2$, and finally for an arbitrary N.

Propensity of transcription when $N = 1$

In the simplest case when $N = 1$, the promoter can be in one of two states. In state S_0, the promoter is not bound by any transcription factors, and in state S_1 the promoter is bound by H_1. Each state S_j is linked with a relative activation α_j, a number between 0 and 1 representing the activity of the promoter at this particular state. The propensity function is thus equal to the expected value of the activity of the promoter multiplied by the pre-mRNA production rate of G.

$$f(y_1, y_2, \ldots, y_N) = \text{wpr} \cdot \sum_{j=0}^{2^N - 1} \alpha_j \cdot P(S_j) \tag{2.1}$$

(2.2)

For $N = 1$, $P(S_1)$ is equal to the Hill equation, where k_i represents the concentration

of H_i at half-occupation and n_i represents the Hill coefficient. Typically, n_i is between [1,10]

$$P(S_1) = \frac{y_1^{n_1}}{k_1^{n_1} + y_1^{n_1}} \tag{2.3}$$

$$= \frac{(y_1/k_1)^{n_1}}{1 + (y_1/k_1)^{n_1}} \tag{2.4}$$

The Hill equation can be simplified by letting $\nu_i = \left(\frac{y_i}{k_i}\right)^{n_i}$.

$$P(S_1) = \frac{\nu_1}{1 + \nu_1} \tag{2.5}$$

Since $P(S_0) = 1 - P(S_1)$, the activation function is formulated and simplified as follows.

$$f(y_1) = \text{wpr} \cdot (\alpha_0 \cdot P(S_0) + \alpha_1 \cdot P(S_1)) \tag{2.6}$$

$$= \text{wpr} \cdot \left(\alpha_0 \cdot \frac{1}{1 + \nu_1} + \alpha_1 \cdot \frac{\nu_1}{1 + \nu_1}\right) \tag{2.7}$$

$$= \text{wpr} \cdot \frac{\alpha_0 + \alpha_1 \cdot \nu_1}{1 + \nu_1} \tag{2.8}$$

$$\tag{2.9}$$

Propensity of transcription when $N = 2$

When $N = 2$, there are four states S_j. The relative activations α_j can be defined such that H_1 and H_2 are independent (additive) or synergistic (multiplicative). In order to define the propensity of transcription $f(.)$, the Hill equation $P(S_j)$ is extended for two transcription factors.

Let w_j be the numerator of $P(S_j)$, defined as the product of all transcription factors bound in that state:

$$w_0 = 1 \tag{2.10}$$

$$w_1 = \nu_1 \tag{2.11}$$

$$w_2 = \nu_2 \tag{2.12}$$

$$w_3 = \nu_1 \cdot \nu_2 \tag{2.13}$$

The denominator of $P(S_j)$ is then equal to the sum of all w_j. The probability of state S_j is thus defined as:

$$P(S_j) = \frac{w_j}{\sum_{j=0}^{j<2^N} w_j} \tag{2.14}$$

$$= \frac{w_j}{1 + \nu_1 + \nu_2 + \nu_1 \cdot \nu_2} \tag{2.15}$$

$$= \frac{w_j}{\prod_{i=1}^{i \leq N} (\nu_i + 1)} \tag{2.16}$$

Substituting $P(S_j)$ and w_j into $f(.)$ results in the following equation:

$$f(y_1, y_2) = \text{wpr} \cdot \sum_{j=0}^{2^N - 1} \alpha_j \cdot P(S_j) \tag{2.17}$$

$$= \text{wpr} \cdot \frac{\sum_{j=0}^{2^N-1} \alpha_j \cdot w_j}{\prod_{i=1}^{i \leq N} (\nu_i + 1)} \tag{2.18}$$

$$= \text{wpr} \cdot \frac{\alpha_0 + \alpha_1 \cdot \nu_1 + \alpha_2 \cdot \nu_2 + \alpha_3 \cdot \nu_1 \cdot \nu_2}{(\nu_1 + 1) \cdot (\nu_2 + 1)} \tag{2.19}$$

$$\tag{2.20}$$

Propensity of transcription for an arbitrary N

For an arbitrary N, there are 2^N states S_j. The relative activations α_j can be defined such that H_1 and H_2 are independent (additive) or synergistic (multiplicative). In order to define the propensity of transcription $f(.)$, the Hill equation $P(S_j)$ is extended for N transcription factors.

Let w_j be the numerator of $P(S_j)$, defined as the product of all transcription factors bound in that state:

$$w_j = \prod_{i=1}^{i \leq N} (j \bmod i) = 1 \ ? \ \nu_i : 1 \tag{2.21}$$

The denominator of $P(S_j)$ is then equal to the sum of all w_j. The probability of state S_j is thus defined as:

$$P(S_j) = \frac{w_j}{\sum_{j=0}^{j<2^N} w_j} \tag{2.22}$$

$$= \frac{w_j}{\prod_{i=1}^{i\leq N}(\nu_i + 1)} \tag{2.23}$$

Substituting $P(S_j)$ into $f(.)$ yields:

$$f(y_1, y_2, \ldots, y_N) = \text{wpr} \cdot \sum_{j=0}^{2^N-1} \alpha_j \cdot P(S_j) \tag{2.24}$$

$$= \text{wpr} \cdot \frac{\sum_{j=0}^{2^N-1} \alpha_j \cdot w_j}{\prod_{i=1}^{i\leq N}(\nu_i + 1)} \tag{2.25}$$

Propensity of transcription for a large N

For large values of N, computing $f(.)$ is practically infeasible as it requires performing 2^N summations. In order to greatly simplify $f(.)$, α_j could be defined as 0 when one of the regulators inhibits transcription and 1 otherwise.

$$\alpha_j = \begin{cases} 0 & \text{if } \exists i : j \bmod i = 1 \text{ and } H_i \text{ represses } G \\ 1 & \text{otherwise} \end{cases} \tag{2.26}$$

Substituting equation 2.26 into equation 2.25 and defining $R = \{1, 2, \ldots, N\}$ and $R^+ = \{i | H_i \text{ activates } G\}$ yields the simplified propensity function:

$$f(y_1, y_2, \ldots, y_N) = \text{wpr} \cdot \frac{\prod_{i \in R^+}(\nu_i + 1)}{\prod_{i \in R}(\nu_i + 1)} \tag{2.27}$$

Independence, synergism and basal expression

The definition of α_j as in equation 2.26 presents two main limitations. Firstly, since $\alpha_0 = 1$, it is impossible to tweak the propensity of transcription when no transcription factors are bound. Secondly, it is not possible to tweak the independence and synergism of multiple regulators.

Let ba $\in [0, 1]$ denote the basal expression strength G (i.e. how much will G be expressed when no transcription factors are bound), and sy $\in [0, 1]$ denote the synergism

of regulators H_i of G, the transcription propensity becomes:

$$f(y_1, y_2, \ldots, y_N) = \text{wpr} \cdot \frac{\text{ba} - \text{sy}^{|R^+|} + \prod_{i \in R^+}(\nu_i + \text{sy})}{\prod_{i \in R}(\nu_i + 1)} \tag{2.28}$$

2.4.4 Compute average expression along backbone transitions

When simulating the developmental backbone, we go through the edges of the backbone state network defined in an earlier step (Section 2.4.1), starting from the root state. It is assumed the root state has no modules active and has no expression of any molecules. To get to the next state, we follow a transition starting from the root state, activate and deactivate the modules as indicated by the transition, and compute the average molecule abundance along the transition. To compute the average abundance, we perform small time steps $t = 0.001$ and let each reaction (Section 2.4.3) occur t times its propensity.

2.4.5 Simulate single cells

dyngen uses Gillespie's stochastic simulation algorithm (SSA) [13] to simulate dynamic processes. An SSA simulation is an iterative process where at each iteration one reaction is triggered.

Each reaction consists of its propensity – a formula to calculate the probability of the reaction occurring during an infinitesimal time interval – and the effect – how it will affect the current state if triggered. Each time a reaction is triggered, the simulation time is incremented by $\tau = \frac{1}{\sum_j prop_j} \ln\left(\frac{1}{r}\right)$, with $r \in U(0, 1)$ and $prop_j$ the propensity value of the jth reaction for the current state of the simulation.

GillespieSSA2 is an optimised library for performing SSA simulations. The propensity functions are compiled to C++ and SSA approximations can be used which allow to trigger many reactions simultaneously at each iteration. The framework also allows to store the abundance levels of molecules only after a specific interval has passed since the previous census. By setting the census interval to 0, the whole simulation's trajectory is retained but many of these time points will contain very similar information. In addition to the abundance levels, also the propensity values and the number of firings of each of the reactions at each of the time steps can be retained, as well as specific sub-calculations of the propensity values, such as the regulator activity level $reg_{G,H}$.

Map SSA simulations to backbone

We compute the Pearson correlation between the state vectors in the simulation and the average expression levels along a transition in the backbone. Each timepoint in the SSA simulation is mapped to the point in the backbone that has the highest correlation value.

2.4.6 Simulate experiment

From the SSA simulation we obtain the abundance levels of all the molecules at every state. We need to replicate technical effects introduced by experimental protocols in order to obtain data that is similar to real data. For this, the cells are sampled from the simulations and molecules are sampled for each of the cells. Real datasets are used in order to achieve similar data characteristics.

Sample cells

In this step, N cells are sampled the simulations. Two approaches are implemented: sampling from an unsynchronised population of single cells (snapshot) or sampling at multiple time points in a synchronised population (time series).

Snapshot The backbone consists of several states linked together by transition edges with length L_i, to which the different states in the different simulations have been mapped (Figure 2.10A). From each transition, $N_i = N/\frac{L_i}{\sum L_i}$ cells are sampled uniformly, rounded such that $\sum N_i = N$.

Time series Assuming that the final time of the simulations is T, the interval $[0, T]$ is divided into k equal intervals of width w separated by $k - 1$ gaps of width g. $N_i = N/k$ cells are sampled uniformly from each interval (Figure 2.10B), rounded such that $\sum N_i = N$. By default, $k = 8$ and $g = 0.75$. For usual dyngen simulations, $10 \leq T \leq 20$. For larger values of T, k and g should be increased accordingly.

Sample molecules

Molecules are sampled from the simulation to replicate how molecules are experimentally sampled. A real dataset is downloaded from a repository of single-cell RNA-seq datasets [34]. For each *in silico* cell i, draw its library size $ls_i$from the distribution of transcript counts per cell in the real dataset. The capture rate cr_j of each *in silico* molecule type j is drawn from $N(1, 0.05)$. Finally, for each cell i, draw ls_i molecules

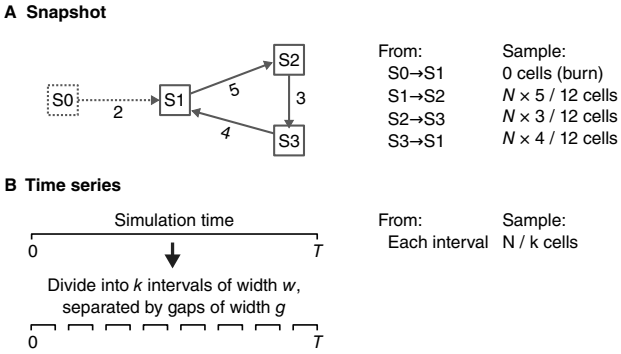


Figure 2.10: Two approaches can be used to sample cells from simulations: snapshot and time-series.

from the multinomial distribution with probabilities $cr_j \times ab_{i,j}$ with $ab_{i,j}$ the molecule abundance level of molecule j in cell i.

2.4.7 Determining the casewise ground-truth regulatory network

Calculating the regulatory effect of a regulator R on a target T (Figure 2.6F) requires determining the contribution of R in the propensity function of the transcription of T (section 2.4.3) with respect to other regulators. This information is useful, amongst others, for benchmarking casewise network inference methods.

The regulatory effect of R on T at a particular state S is defined as the change in the propensity of transcription when R is set to zero, scaled by the inverse of the pre-mRNA production rate of T. More formally:

$$\text{regeffect}_G = \frac{\text{proptrans}_G(S) - \text{proptrans}_G(S[\text{y}_T \leftarrow 0])}{\text{wpr}_G}$$

Determining the regulatory effect for all interactions and cells in the dataset yields the complete casewise ground-truth GRN (Figure 2.11). The regulatory effect lie between $[-1, 1]$, where -1 represents complete inhibition of T by R, 1 represents maximal activation of T by R, and 0 represents inactivity of the regulatory interaction between R and T.

2.4.8 Comparison of casewise network inference methods

Several datasets were generated using the different predefined backbones. For every cell in the dataset, the transcriptomics profile and the corresponding casewise ground-

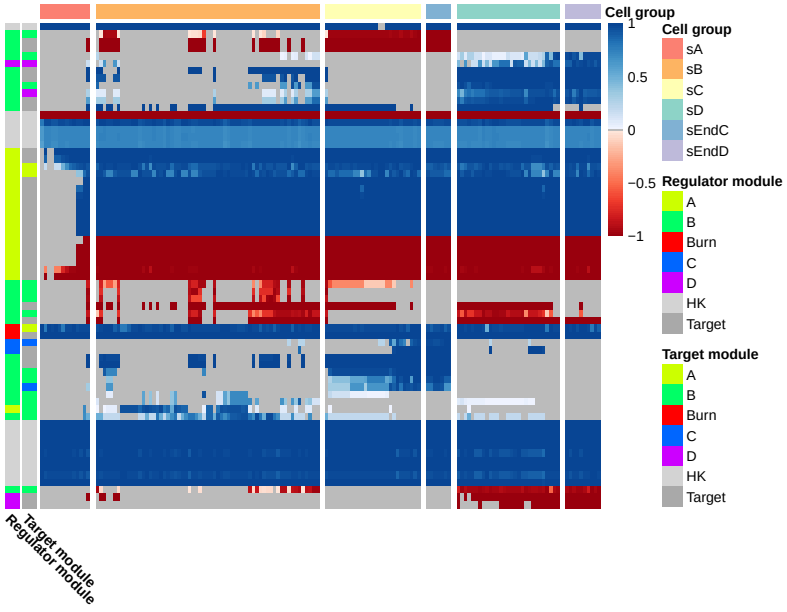


Figure 2.11: The casewise regulatory effects of all interactions, computed on cells part of a bifurcation trajectory. Negative values correspond to inhibitory interactions, positive values to activating interactions, and zero values correspond to inactive interactions.

truth regulatory network was determined (Section 2.4.7).

Several casewise NI methods were considered for comparison: SCENIC [19], LIONESS [35, 20], and SSN [21].

LIONESS [20, 36] uses the Pearson correlation to infer casewise GRNs. To do so, first the Pearson correlation is calculated between regulators and targets for all samples. Next, the Pearson correlation is again calculated for all samples excluding one sample. The difference between the two correlation matrices is considered a casewise GRN for that particular profile. This process is repeated for all profiles, resulting in a casewise GRN.

SSN [21] has, in essence, the exact same methodology as LIONESS. It is worth noting that the LIONESS preprint was released before the publication of SSN. Since no implementation was provided by the authors, we implemented SSN in R using basic R and tidyverse functions [37] and marked results from this implementation as "SSN*".

SCENIC [19] is a pipeline that consists of four main steps. Step 1: classical network inference is performed with arboreto, which is similar to GENIE3 [38]. Step 2: select the top 10 regulators per target. Interactions are grouped together in 'modules'; each module contains one regulator and all of its targets. Step 3: filter the modules using motif analysis. Step 4: for each cell, determine an activity score of each module using AUCell. As a post-processing of this output, all modules and the corresponding

activity scores are combined back into a casewise GRN consisting of (cell, regulator, target, score) pairs. For this analysis, the Python implementation of SCENIC was used, namely pySCENIC. Since dyngen does not generate motif data, step 3 in this analysis is skipped.

The AUROC and AUPR metrics are common metrics for evaluating a predicted GRN with a ground-truth GRN. To compare a predicted casewise GRN with the ground-truth casewise GRN, the top 10'000 interactions per cell were retained. For each cell-specific network, the AUROC and AUPR were calculated. ## Comparison of RNA velocity methods {#sec:dyngen-velcompare}

15 datasets were generated with 5 different backbones: linear, linear simple, bifurcating, cyclic, and disconnected. We extracted a ground truth RNA velocity by subtracting for each mRNA molecule the propensity of its production by the propensity of its degradation. If the expression of an mRNA will increase in the future, this value is positive, while it is negative if it is going to decrease. For each gene, we compared the ground truth velocity with the observed velocity by calculating the Spearman rank correlation.

We compared two RNA velocity methods. The velocyto method [17], as implemented in the velocyto.py package, in which we varied the "assumption" parameter between "constant_unspliced" and "constant_velocity". The scvelo method [18], as implemented in the python scvelo package (http://scvelo.de), in which we varied the "mode" parameter between "deterministic", "stochastic" and "dynamical". For both methods, we used the same normalized data as provided by dyngen, with no extra cell or feature filtering. We also matched the parameters between both methods as best as possible, i.e. the k parameter for smoothing was set to 20 for both methods.

To visualize the velocity on an embedding, we used the "velocity_embedding()" function, implemented in the scvelo Python package.

2.4.9 Code availability

dyngen is available as an open source software package at https://github.com/dynverse/dyngen. All code used in this study is made publicly available at https://github.com/rcannood/dyngen_analysis.

2.5 References

[1] Luke Zappia, Belinda Phipson, and Alicia Oshlack. "Splatter: Simulation of Single-Cell RNA Sequencing Data". In: *Genome Biology* 18 (Sept. 2017), p. 174. ISSN: 1474-760X. DOI: 10.1186/s13059-017-1305-0.

[2] Beate Vieth et al. "powsimR: Power Analysis for Bulk and Single Cell RNA-Seq Experiments". In: *Bioinformatics* 33.21 (Nov. 1, 2017), pp. 3486–3488. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btx435.

[3] Nikolaos Papadopoulos, Rodrigo Gonzalo Parra, and Johannes Soeding. "PROSSTT: Probabilistic Simulation of Single-Cell RNA-Seq Data for Complex Differentiation Processes". In: *bioRxiv* (Jan. 2018), p. 256941. DOI: 10.1101/256941.

[4] Xiuwei Zhang, Chenling Xu, and Nir Yosef. "Simulating Multiple Faceted Variability in Single Cell RNA Sequencing". In: *Nature Communications* 10.1 (June 13, 2019), pp. 1–16. ISSN: 2041-1723. DOI: 10.1038/s41467-019-10500-w.

[5] Kelly Street et al. "Slingshot: Cell Lineage and Pseudotime Inference for Single-Cell Transcriptomics". In: *BMC Genomics* 19.1 (June 2018), p. 477. ISSN: 1471-2164. DOI: 10.1186/s12864-018-4772-0.

[6] R Gonzalo Parra et al. "Reconstructing Complex Lineage Trees from scRNA-Seq Data Using MERLoT". In: *bioRxiv* (Feb. 2018), p. 261768. DOI: 10.1101/261768.

[7] Edroaldo Lummertz da Rocha et al. "Reconstruction of Complex Single-Cell Trajectories Using CellRouter". In: *Nature Communications* 9.1 (Mar. 1, 2018), p. 892. ISSN: 2041-1723. DOI: 10.1038/s41467-018-03214-y.

[8] Yingxin Lin et al. "scClassify: Hierarchical Classification of Cells". In: *bioRxiv* (Jan. 1, 2019), p. 776948. DOI: 10.1101/776948.

[9] 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), p. 1141. ISSN: 2046-1402. DOI: 10.12688/f1000research.15666.2. pmid: 30271584.

[10] Wouter Saelens et al. "A Comparison of Single-Cell Trajectory Inference Methods". In: *Nature Biotechnology* 37 (May 2019). ISSN: 15461696. DOI: 10.1038/s41587-019-0071-9.

[11] Charlotte Soneson and Mark D. Robinson. "Bias, Robustness and Scalability in Single-Cell Differential Expression Analysis". In: *Nature Methods* 15.4 (Apr. 2018), pp. 255–261. ISSN: 1548-7105. DOI: 10.1038/nmeth.4612. pmid: 29481549.

[12] Tim Stuart and Rahul Satija. "Integrative Single-Cell Analysis". In: *Nature Reviews Genetics* 20.5 (May 2019), pp. 257–272. ISSN: 1471-0064. DOI: 10.1038/s41576-019-0093-7.

[13] Daniel T. Gillespie. "Exact Stochastic Simulation of Coupled Chemical Reactions". In: *The Journal of Physical Chemistry* 81.25 (Dec. 1, 1977), pp. 2340–2361. ISSN: 0022-3654. DOI: 10.1021/j100540a008.

[14] Koen Van den Berge et al. "Trajectory-Based Differential Expression Analysis for Single-Cell Sequencing Data". In: *bioRxiv* (Jan. 1, 2019), p. 623397. DOI: 10.1101/623397.

[15] Aditya Pratapa et al. "Benchmarking Algorithms for Gene Regulatory Network Inference from Single-Cell Transcriptomic Data". In: *bioRxiv* (June 4, 2019), p. 642926. DOI: 10.1101/642926.

[16] Amit Zeisel et al. "Coupled Pre-mRNA and mRNA Dynamics Unveil Operational Strategies Underlying Transcriptional Responses to Stimuli". In: *Molecular Systems Biology* 7.1 (Jan. 1, 2011), p. 529. ISSN: 1744-4292. DOI: 10.1038/msb.2011.62.

[17] Gioele La Manno et al. "RNA Velocity of Single Cells". In: *Nature* 560.7719 (Aug. 2018), pp. 494–498. ISSN: 1476-4687. DOI: 10.1038/s41586-018-0414-6.

[18] Volker Bergen et al. "Generalizing RNA Velocity to Transient Cell States through Dynamical Modeling". In: *bioRxiv* (Oct. 29, 2019), p. 820936. DOI: 10.1101/820936.

[19] Sara Aibar et al. "SCENIC: Single-Cell Regulatory Network Inference and Clustering". In: *Nature Methods* (Oct. 2017). ISSN: 1548-7091. DOI: 10.1038/nmeth.4463.

[20] Marieke Lydia Kuijjer et al. "Estimating Sample-Specific Regulatory Networks". In: *iScience* 14 (Mar. 28, 2019), pp. 226–240. ISSN: 2589-0042. DOI: 10.1016/j.isci.2019.03.021. pmid: 30981959.

[21] Xiaoping Liu et al. "Personalized Characterization of Diseases Using Sample-Specific Networks". In: *Nucleic Acids Research* 44.22 (2016), e164–e164. ISSN: 0305-1048. DOI: 10.1093/nar/gkw772. pmid: 27596597.

[22] Thomas Schaffter, Daniel Marbach, and Dario Floreano. "GeneNetWeaver: In Silico Benchmark Generation and Performance Profiling of Network Inference Methods.". In: *Bioinformatics* 27.16 (Aug. 2011), pp. 2263–2270. ISSN: 1367-4811. DOI: 10.1093/bioinformatics/btr373. pmid: 21697125.

[23] Adam D. Ewing et al. "Combining Tumor Genome Simulation with Crowdsourcing to Benchmark Somatic Single-Nucleotide-Variant Detection". In: *Nature Methods* 12.7 (July 2015), pp. 623–630. ISSN: 1548-7105. DOI: 10.1038/nmeth.3407.

[24] Stephen Smith and Ramon Grima. "Spatial Stochastic Intracellular Kinetics: A Review of Modelling Approaches". In: *Bulletin of Mathematical Biology* 81.8

(Aug. 1, 2019), pp. 2960–3009. ISSN: 1522-9602. DOI: 10.1007/s11538-018-0443-1.

[25] N. Rekhtman et al. "Direct Interaction of Hematopoietic Transcription Factors PU.1 and GATA-1: Functional Antagonism in Erythroid Cells". In: *Genes & Development* 13.11 (June 1, 1999), pp. 1398–1411. ISSN: 0890-9369. DOI: 10.1101/gad.13.11.1398. pmid: 10364157.

[26] Heping Xu et al. "Regulation of Bifurcating {B} Cell Trajectories by Mutual Antagonism between Transcription Factors {IRF4} and {IRF8}". In: *Nat. Immunol.* 16.12 (Dec. 2015), pp. 1274–1281.

[27] Thomas Graf and Tariq Enver. "Forcing Cells to Change Lineages". In: *Nature* 462.7273 (Dec. 2009), p. 587. ISSN: 1476-4687. DOI: 10.1038/nature08533.

[28] Jin Wang et al. "Quantifying the Waddington Landscape and Biological Paths for Development and Differentiation". In: *Proceedings of the National Academy of Sciences* 108.20 (May 2011), pp. 8257–8262. ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.1017017108. pmid: 21536909.

[29] James E Ferrell. "Bistability, Bifurcations, and Waddington's Epigenetic Landscape". In: *Current Biology* 22.11 (June 2012), R458–R466. ISSN: 0960-9822. DOI: 10.1016/j.cub.2012.03.045.

[30] Nir Yosef et al. "Dynamic Regulatory Network Controlling {TH17} Cell Differentiation". In: *Nature* 496.7446 (2013), pp. 461–468.

[31] Marina Lizio et al. "Gateways to the FANTOM5 Promoter Level Mammalian Expression Atlas". In: *Genome Biology* 16.1 (Jan. 5, 2015), p. 22. ISSN: 1465-6906. DOI: 10.1186/s13059-014-0560-6.

[32] Michael A. Gibson and Jehoshua Bruck. "A Probabilistic Model of a Prokaryotic Gene and Its Regulation". In: *Computational Methods in Molecular Biology: From Genotype to Phenotype, MIT press, Cambridge* (2000).

[33] Maria J. Schilstra and Chrystopher L. Nehaniv. "Bio-Logic: Gene Expression and the Laws of Combinatorial Logic". In: *Artificial Life* 14.1 (Jan. 1, 2008), pp. 121–133. ISSN: 1064-5462. DOI: 10.1162/artl.2008.14.1.121.

[34] Robrecht Cannoodt et al. "Single-Cell -Omics Datasets Containing a Trajectory". In: *Zenodo* (Oct. 2018). DOI: 10.5281/zenodo.1211532.

[35] Marieke Lydia Kuijjer et al. "Estimating Sample-Specific Regulatory Networks". In: (2015), pp. 1–19. URL: http://arxiv.org/abs/1505.06440.

[36] Marieke L. Kuijjer et al. "lionessR: Single Sample Network Inference in R". In: *BMC Cancer* 19.1 (Oct. 25, 2019), p. 1003. ISSN: 1471-2407. DOI: 10.1186/s12885-019-6235-7.

[37] Hadley Wickham et al. "Welcome to the Tidyverse". In: (Nov. 21, 2019). DOI: 10.21105/joss.01686.

[38] Vân Anh Huynh-Thu et al. “Inferring Regulatory Networks from Expression Data Using Tree-Based Methods”. In: *PLoS ONE* 5.9 (Jan. 2010), e12776. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0012776. pmid: 20927193.

3 | dynbenchmark: A comparison of single-cell trajectory inference methods

Abstract

Purpose: Trajectory inference approaches analyse genome-wide omics data from thousands of single cells and computationally infer the order of these cells along developmental trajectories. Although more than 75 trajectory inference tools have already been developed, it is challenging to compare their performance because the input they require and output models they produce vary substantially.

Results: Here, we benchmark 45 of these methods on 110 real and 229 synthetic datasets for cellular ordering, topology, scalability and usability. Our results highlight the complementarity of existing tools, and that the choice of method should depend mostly on the dataset dimensions and trajectory topology. We develop a set of guidelines to help users select the best method for their dataset.

Conclusion: Our freely available data and evaluation pipeline will aid in the development of improved tools designed to analyse increasingly large and complex single-cell datasets.

Publication status

Published in Nature Biotechnology 37, 5 (2019). doi:10.1038/s41587-019-0071-9.

Saelens W*, **Cannoodt R***, Todorov H, and Saeys Y.

* Equal contribution

Author contributions

- **R.C.**, W.S., H.T. and Y.S. designed the study.
- **R.C.** and W.S. performed the experiments and analysed the data.
- **R.C.**, W.S., and H.T. implemented software packages.
- **R.C.** and W.S. wrote the original manuscript.
- **R.C.**, W.S., H.T., and Y.S. reviewed and edited the manuscript.
- Y.S. supervised the project.

3.1 Introduction

Single-cell omics data, including transcriptomics, proteomics and epigenomics data, provide new opportunities for studying cellular dynamic processes, such as the cell cycle, cell differentiation and cell activation [1, 2]. Such dynamic processes can be modelled computationally using trajectory inference (TI) methods, also called pseudotime analysis, which order cells along a trajectory based on similarities in their expression patterns [3, 4, 5]. The resulting trajectories are most often linear, bifurcating or tree-shaped, but more recent methods also identify more complex trajectory topologies, such as cyclic [6] or disconnected graphs [7]. TI methods offer an unbiased and transcriptome-wide understanding of a dynamic process [1], thereby allowing the objective identification of new (primed) subsets of cells [8], delineation of a differentiation tree [9, 10] and inference of regulatory interactions responsible for one or more bifurcations [11]. Current applications of TI focus on specific subsets of cells, but ongoing efforts to construct transcriptomic catalogues of whole organisms [12, 13, 14] underline the urgency for accurate, scalable [11, 15] and user-friendly TI methods.

A plethora of TI methods has been developed over the past few years and even more are being created every month (Supplementary Table 3.1). Indeed, in several repositories listing single-cell tools, such as omictools.org [16], the 'awesome-single-cell' list [17] and scRNA-tools.org [18], TI methods are one of the largest categories. While each method has its own unique set of characteristics in terms of underlying algorithm, required prior information and produced outputs, two of the most distinctive differences between TI methods are whether they fix the topology of the trajectory and what type(s) of graph topologies they can detect. Early TI methods typically fixed the topology algorithmically (for example, linear [19, 8, 20, 21] or bifurcating trajectories [22, 23]) or through parameters provided by the user [24, 25]. These methods therefore mainly focus on correctly ordering the cells along the fixed topology. More recent methods also infer the topology [26, 27, 7], which increases the difficulty of the problem at hand, but allows the unbiased identification of both the ordering inside a branch and the topology connecting these branches.

Given the diversity in TI methods, it is important to quantitatively assess their performance, scalability, robustness and usability. Many attempts at tackling this issue have already been made [22, 28, 29, 25, 30, 4, 31, 32, 7], but a comprehensive comparison of TI methods across a large number of different datasets is still lacking. This is problematic, as new users to the field are confronted with an overwhelming choice of TI methods, without a clear idea of which would optimally solve their problem. Moreover, the strengths and weaknesses of existing methods need to be assessed, so that new developments in the field can focus on improving the current state-of-the-art.

Yvan, you are an incredible supervisor and group leader.
You always tell the baby PhD students that the first step in a PhD

In this study, we evaluated the accuracy, scalability, stability and usability of 45 TI methods (Figure 3.1a). We found substantial complementarity between current methods, with different sets of methods performing most optimally depending on the characteristics of the data. For method users, we created an interactive set of guidelines (available at http://guidelines.dynverse.org), which gives context-specific recommendations for method usage. Our evaluation also highlights some challenges for current methods, and our evaluation strategy can be useful to spearhead the development of new tools that accurately infer trajectories on ever more complex use cases.

3.2 Results

3.2.1 Trajectory inference methods

To make the outputs from different methods directly comparable to each other, we developed a common probabilistic model for representing trajectories from all possible sources (Figure 3.1b). In this model, the overall topology is represented by a network of 'milestones', and the cells are placed within the space formed by each set of connected milestones. Although almost every method returned a unique set of outputs, we were able to classify these outputs into seven distinct groups (Figure 3.2) and we wrote a common output converter for each of these groups (Figure 3.3a). When strictly required, we also provided prior information to the method. These different priors can range from weak priors that are relatively easy to acquire, such as a start cell, to strong priors, such as a known grouping of cells, that are much harder to know a priori, and which can potentially introduce a large bias into the analysis (Figure 3.3a).

The largest difference between TI methods is whether a method fixes the topology and, if it does not, what kind of topology it can detect. We defined seven possible types of topology, ranging from very basic topologies (linear, cyclical and bifurcating) to the more complex ones (connected and disconnected graphs). Most methods either focus on inferring linear trajectories or limit the search to tree or less complex topologies, with only a selected few attempting to infer cyclic or disconnected topologies (Figure 3.3a).

We evaluated each method on four core aspects: (1) accuracy of a prediction, given a gold or silver standard on 110 real and 229 synthetic datasets; (2) scalability with respect to the number of cells and features (for example, genes); (3) stability of the predictions after subsampling the datasets; and (4) the usability of the tool in terms of software, documentation and the manuscript. Overall, we found a large diversity across the four evaluation criteria, with only a few methods, such as PAGA, Sling-

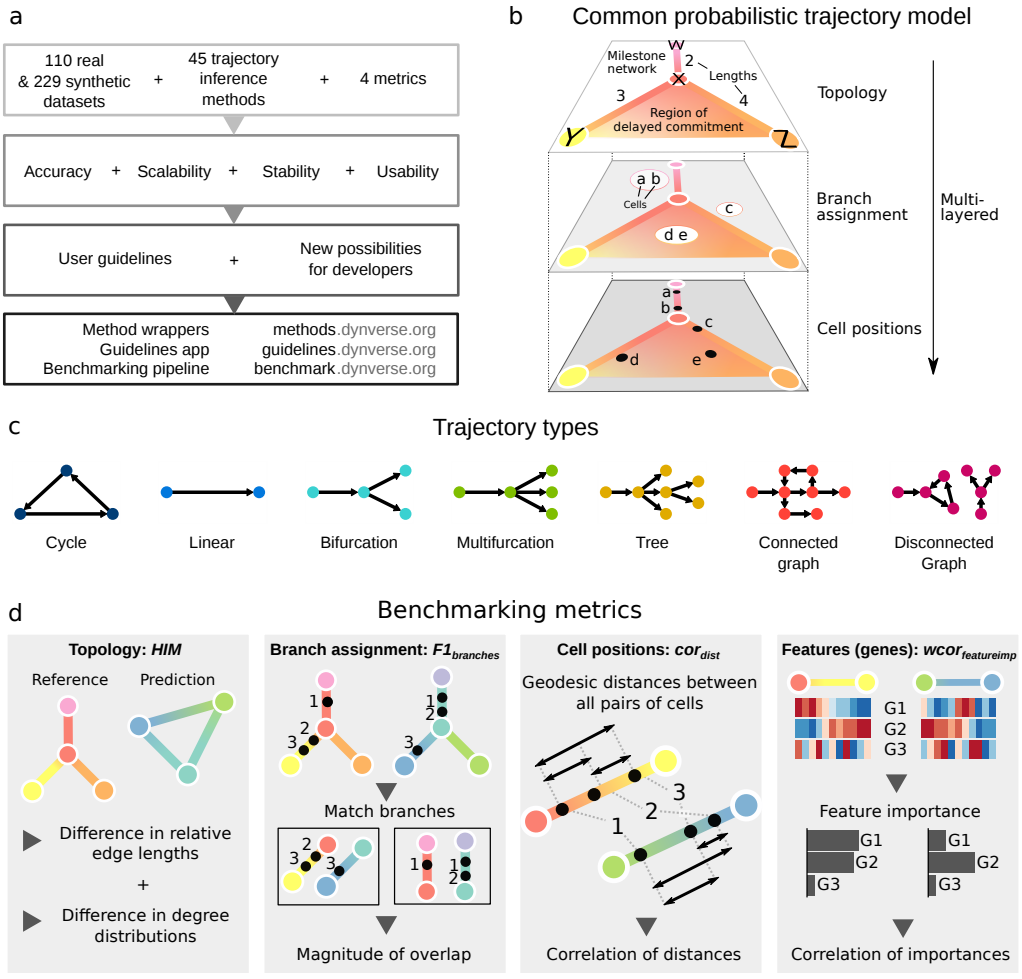


Figure 3.1: Overview of several key aspects of the evaluation. **a**, A schematic overview of our evaluation pipeline. **b**, To make the trajectories comparable to each other, a common trajectory model was used to represent reference trajectories from the real and synthetic datasets, as well as any predictions of TI methods. **c**, Trajectories are automatically classified into one of seven trajectory types, with increasing complexity. **d**, We defined four metrics, each assessing the quality of a different aspect of the trajectory. The HIM score assesses the similarity between the two topologies, taking into account differences in edge lengths and degree distributions. The F1branches assesses the similarity of the assignment of cells onto branches. The cordist quantifies the similarity in cellular positions between two trajectories, by calculating the correlation between pairwise geodesic distances. Finally, wcorfeatures quantifies the agreement between trajectory differentially expressed features from the known trajectory and the predicted trajectory.

Katleen, thank you for your dedication and perseverance
in guiding me and making sure I got off to a good start in my PhD.

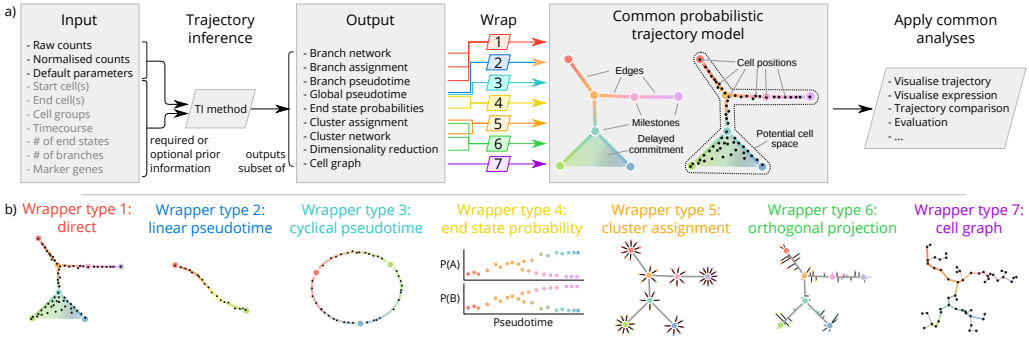


Figure 3.2: A common interface for TI methods. **a** The input and output of each TI method is standardized. As input, each TI method receives either raw or normalized counts, several parameters, and a selection of prior information. After its execution, a method uses one of the seven wrapper functions to transform its output to the common trajectory model. This common model then allows to perform common analysis functions on trajectory models produced by any TI method. **b** Illustrations of the specific transformations performed by each of the wrapper functions.

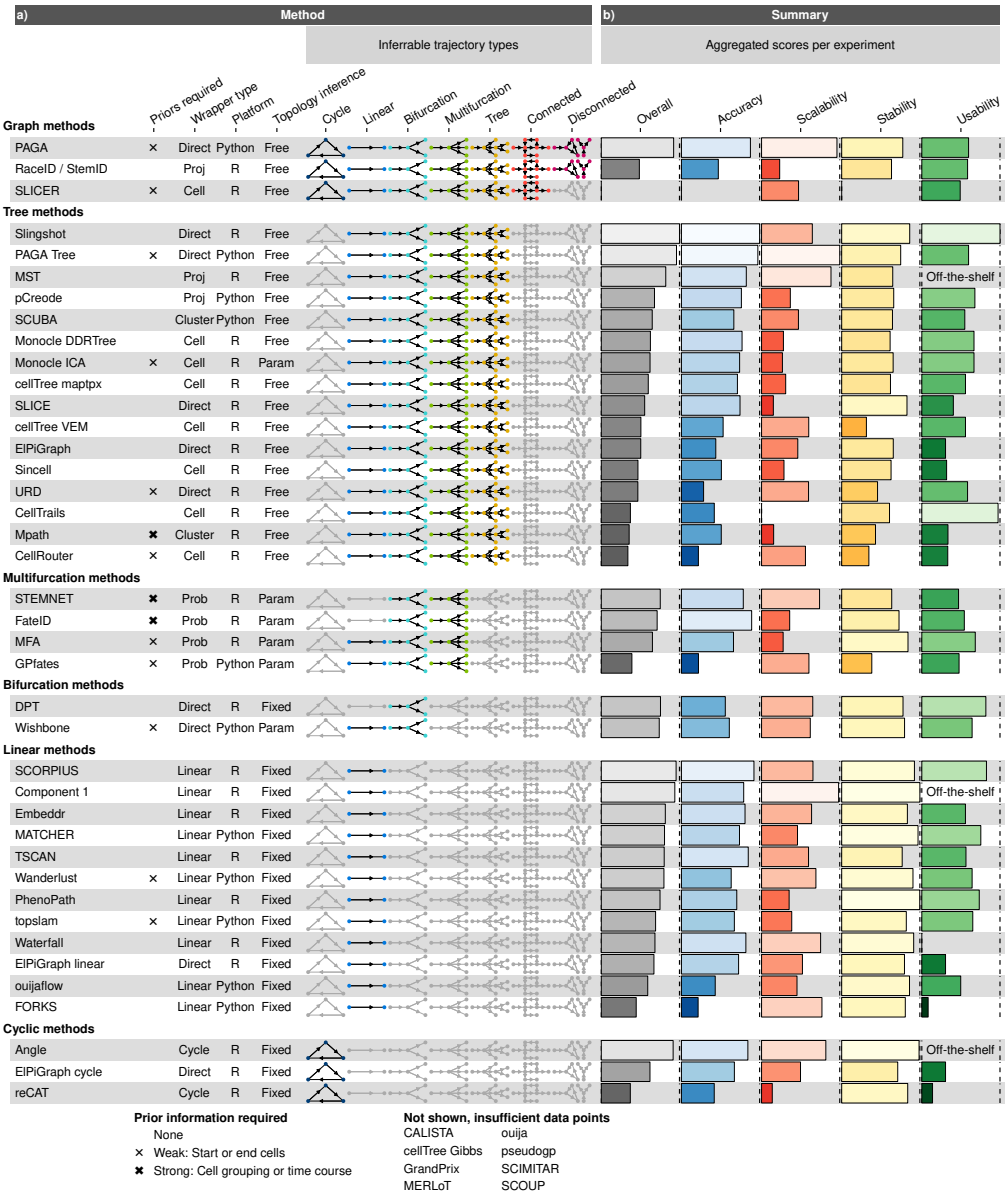


Figure 3.3: A characterization of the 45 methods evaluated in this study and their overall evaluation results. **a**, We characterized the methods according to the wrapper type, their required priors, whether the inferred topology is constrained by the algorithm (fixed) or a parameter (param), and the types of inferable topologies. The methods are grouped vertically based on the most complex trajectory type they can infer. **b**, The overall results of the evaluation on four criteria: accuracy using a reference trajectory on real and synthetic data, scalability with increasing number of cells and features, stability across dataset subsamples and quality of the implementation. Methods that errored on more than 50% of the datasets are not included in this figure and are shown instead in Supplementary Figure 3.1.

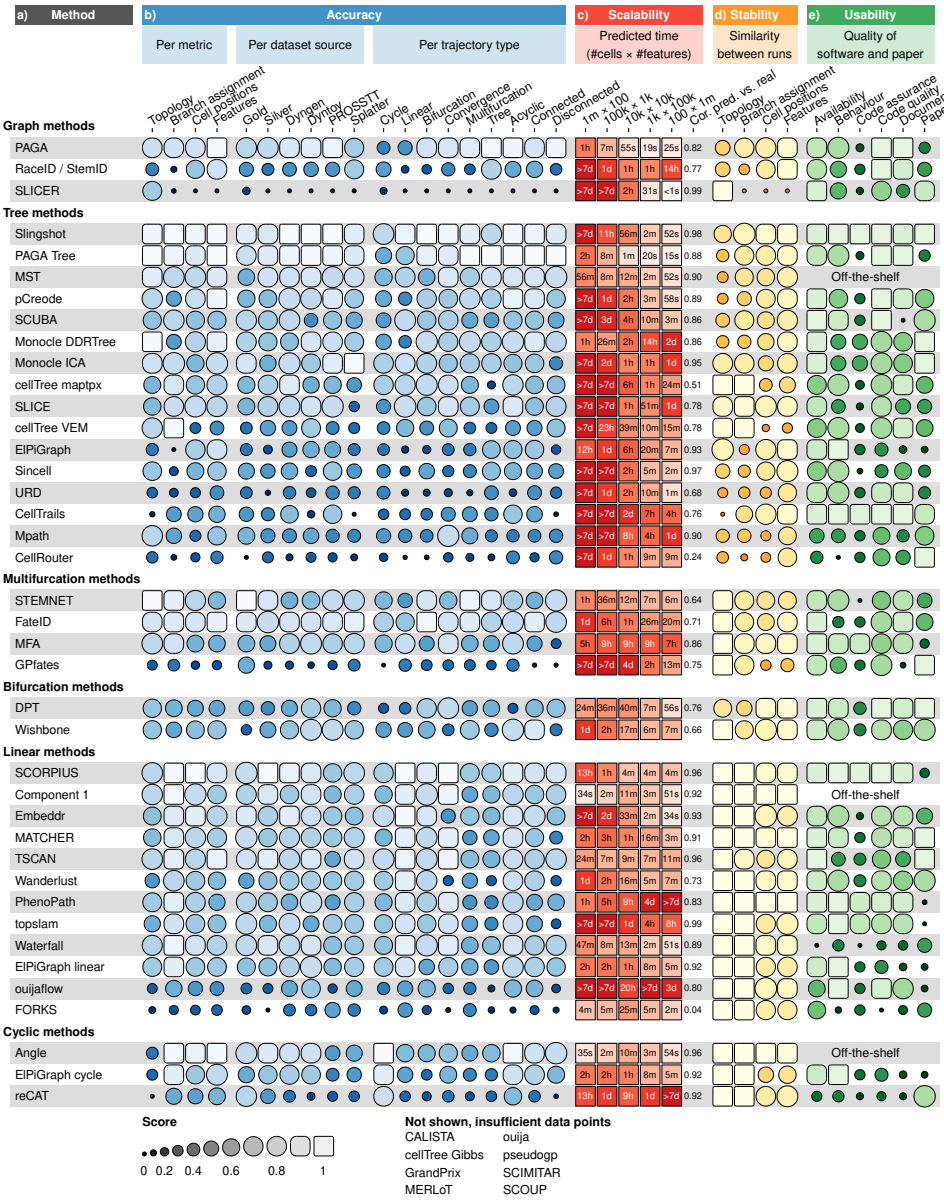


Figure 3.4: Detailed results of the four main evaluation criteria: accuracy, scalability, stability and usability. a, The names of the methods, ordered as in Figure 3.3. **b**, Accuracy of trajectory inference methods across metrics, dataset sources and dataset trajectory types. The performance of a method is generally more stable across dataset sources, but very variable depending on the metric and trajectory type. **c**, Predicted execution times for varying numbers of cells and features (no. of cells × no. of features). Predictions were made by training a regression model after running each method on bootstrapped datasets with varying numbers of cells and features. k, thousands; m, millions; cor, correlation. **d**, Stability results by calculating the average pairwise similarity between models inferred across multiple runs of the same method. **e**, Usability scores of the tool and corresponding manuscript, grouped per category. Off-the-shelf methods were directly implemented in R and thus do not have a usability score.

shot and SCORPIUS, performing well across the board (Figure 3.3b). We will discuss each evaluation criterion in more detail (Figure 3.4 and Supplementary Figure 3.1), after which we conclude with guidelines for method users and future perspectives for method developers.

3.2.2 Accuracy

We defined several metrics to compare a prediction to a reference trajectory (Section 3.6). Based on an analysis of their robustness and conformity to a set of rules (Section 3.6), we chose four metrics each assessing a different aspect of a trajectory (Figure 3.1d): the topology (Hamming–Ipsen–Mikhailov, HIM), the quality of the assignment of cells to branches (F1branches), the cell positions (cordist) and the accuracy of the differentially expressed features along the trajectory (wcorfeatures). The data compendium consisted of both synthetic datasets, which offer the most exact reference trajectory, and real datasets, which provide the highest biological relevance. These real datasets come from a variety of single-cell technologies, organisms and dynamic processes, and contain several types of trajectory topologies (Supplementary Table 3.2). Real datasets were classified as 'gold standard' if the reference trajectory was not extracted from the expression data itself, such as via cellular sorting or cell mixing [33]. All other real datasets were classified as 'silver standard'. For synthetic datasets we used several data simulators, including a simulator of gene regulatory networks using a thermodynamic model of gene regulation [34]. For each simulation, we used a real dataset as a reference, to match its dimensions, number of differentially expressed genes, drop-out rates and other statistical properties [35].

We found that method performance was very variable across datasets, indicating that there is no 'one-size-fits-all' method that works well on every dataset (Figure 3.5a). Even methods that can detect most of the trajectory types, such as PAGA, RaceID/StemID and SLICER were not the best methods across all trajectory types (Figure 3.4b). The overall score between the different dataset sources was moderately to highly correlated (Spearman rank correlation between 0.5–0.9) with the scores on real datasets containing a gold standard (Figure 3.5b), confirming both the accuracy of the gold standard trajectories and the relevance of the synthetic data. On the other hand, the different metrics frequently disagreed with each other, with Monocle and PAGA Tree scoring better on the topology scores, whereas other methods, such as Slingshot, were better at ordering the cells and placing them into the correct branches (Figure 3.4b).

The performance of a method was strongly dependent on the type of trajectory present in the data (Figure 3.4b). Slingshot typically performed better on datasets containing more simple topologies, while PAGA, pCreode and RaceID/StemID had

higher scores on datasets with trees or more complex trajectories (Figure 3.5c). This was reflected in the types of topologies detected by every method, as those predicted by Slingshot tended to contain less branches, whereas those detected by PAGA, pCreode and Monocle DDRTree gravitated towards more complex topologies (Figure 3.5d). This analysis therefore indicates that detecting the right topology is still a difficult task for most of these methods, because methods tend to be either too optimistic or too pessimistic regarding the complexity of the topology in the data.

The high variability between datasets, together with the diversity in detected topologies between methods, could indicate some complementarity between the different methods. To test this, we calculated the likelihood of obtaining a top model when using only a subset of all methods. A top model in this case was defined as a model with an overall score of at least 95% as the best model. On all datasets, using one method resulted in getting a top model about 27% of the time. This increased up to 74% with the addition of six other methods (Figure 3.6a). The result was a relatively diverse set of methods, containing both strictly linear or cyclic methods, and methods with a broad trajectory type range such as PAGA. We found similar indications of complementarity between the top methods on data containing only linear, bifurcation or multifurcating trajectories (Figure 3.6b), although in these cases less methods were necessary to obtain at least one top model for a given dataset. Altogether, this shows that there is considerable complementarity between the different methods and that users should try out a diverse set of methods on their data, especially when the topology is unclear a priori. Moreover, it also opens up the possibilities for new ensemble methods that utilize this complementarity.

3.2.3 Scalability

While early TI methods were developed at a time where profiling more than a thousand cells was exceptional, methods now have to cope with hundreds of thousands of cells, and perhaps soon with more than ten million [36]. Moreover, the recent application of TI methods on multi-omics single-cell data also showcases the increasing demands on the number of features [37]. To assess the scalability, we ran each method on up- and downscaled versions of five distinct real datasets. We modeled the running time and memory usage using a Shape Constrained Additive Model [38] (Figure 3.7a). As a control, we compared the predicted time (and memory) with the actual time (respectively memory) on all benchmarking datasets, and found that these were highly correlated overall (Spearman rank correlation >0.9, Figure 3.8, and moderately to highly correlated (Spearman rank correlation of 0.5–0.9) for almost every method, depending to what extent the execution of a method succeeded during the scalability experiments (Figure 3.4c and Supplementary Figure 3.1a).

We found that the scalability of most methods was overall very poor, with most graph and tree methods not finishing within an hour on a dataset with ten thousand cells and ten thousand features (Figure 3.4c), which is around the size of a typical droplet-based single-cell dataset [36]. Running times increased further with increasing number of cells, with only a handful of graph/tree methods completing within a day on a million cells (PAGA, PAGA Tree, Monocle DDRTree, Stemnet and GrandPrix). Some methods, such as Monocle DDRTree and GrandPrix, also suffered from unsatisfactory running times when given a high number of features.

Methods with a low running time typically had two defining aspects: they had a linear time complexity with respect to the features and/or cells, and adding new cells or features led to a relatively low increase in time (Figure 3.7b). We found that more than half of all methods had a quadratic or superquadratic complexity with respect to the number of cells, which would make it difficult to apply any of these methods in a reasonable time frame on datasets with more than a thousand cells (Figure 3.7b).

We also assessed the memory requirements of each method (Supplementary Figure 3.1c). Most methods had reasonable memory requirements for modern workstations or computer clusters ($\leq$12 GB) with PAGA and STEMNET in particular having a low memory usage with both a high number of cells or a high number of features. Notably, the memory requirements were very high for several methods on datasets with high numbers of cells (RaceID/StemID, pCreode and MATCHER) or features (Monocle DDRTree, SLICE and MFA).

Altogether, the scalability analysis indicated that the dimensions of the data are an important factor in the choice of method, and that method development should pay more attention to maintaining reasonable running times and memory usage.

3.2.4 Stability

It is not only important that a method is able to infer an accurate model in a reasonable time frame, but also that it produces a similar model when given very similar input data. To test the stability of each method, we executed each method on ten different subsamples of the datasets (95% of the cells, 95% of the features), and calculated the average similarity between each pair of models using the same scores used to assess the accuracy of a trajectory (Figure 3.4d).

Given that the trajectories of methods that fix the topology either algorithmically or through a parameter are already very constrained, it is to be expected that such methods tend to generate very stable results. Nonetheless, some fixed topology methods still produced slightly more stable results, such as SCORPIUS and MATCHER for linear methods and MFA for multifurcating methods. Stability was much more diverse

among methods with a free topology. Slingshot produced more stable models than PAGA (Tree), which in turn produced more stable results than pCreode and Monocle DDRTree.

3.2.5 Usability

While not directly related to the accuracy of the inferred trajectory, it is also important to assess the quality of the implementation and how user-friendly it is for a biological user [39]. We scored each method using a transparent checklist of important scientific and software development practices, including software packaging, documentation, automated code testing and publication into a peer-reviewed journal (Table 3.1). It is important to note that there is a selection bias in the tools chosen for this analysis, as we did not include a substantial set of tools due to issues with installation, code availability and executability on a freely available platform (which excludes MATLAB). The reasons for not including certain tools are all discussed on our repository (https://github.com/dynverse/dynmethods/issues?q=label:unwrappable). Installation issues seem to be quite general in bioinformatics [40] and the trajectory inference field is no exception.

We found that most methods fulfilled the basic criteria, such as the availability of a tutorial and elemental code quality criteria (Figure 3.4d and Figure 3.9). While recent methods had a slightly better quality score than older methods, several quality aspects were consistently lacking for the majority of the methods (Figure 3.9 right) and we believe that these should receive extra attention from developers. Although these outstanding issues covered all five categories, code assurance and documentation in particular were problematic areas, notwithstanding several studies pinpointing these as good practices [41, 42]. Only two methods had a nearly perfect usability score (Slingshot and Celltrails), and these could be used as an inspiration for future methods. We observed no clear relation between usability and method accuracy or usability (Figure 3.3b).

3.3 Discussion

In this study, we presented a large-scale evaluation of the performance of 45 TI methods. By using a common trajectory representation and four metrics to compare the methods' outputs, we were able to assess the accuracy of the methods on more than 200 datasets. We also assessed several other important quality measures, such as the quality of the method's implementation, the scalability to hundreds of thousands of cells and the stability of the output on small variations of the datasets.

Table 3.1: Scoring sheet for assessing usability of trajectory inference methods. Each quality aspect was given a weight based on how many times it was mentioned in a set of articles discussing best practices for tool development. Part 1.

Aspect	Items	References
Availability		
Open source	(1) Method's code is freely available (2) The code can be run on a freely available platform	[43, 41, 39, 44, 42, 45, 46]
Version control	The code is available on a public version controlled repository, such as Github	[43, 41, 39, 44, 42, 45]
Packaging	(1) The code is provided as a "package", exposing functionality through functions or shell commands (2) The code can be easily installed through a repository such as CRAN, Bioconductor, PyPI, CPAN, debian packages, ...	[43, 44, 46, 45]
Dependencies	(1) Dependencies are clearly stated in the tutorial or in the code (2) Dependencies are automatically installed	[39, 44, 42, 47]
License	(1) The code is licensed (2) License allows academic use	[43, 39, 44, 42, 45, 46]
Interface	(1) The tool can be run using a graphical user interface, either locally or on a web server (2) The tool can be run through the command line or through a programming language	[45]
Code quality		
Function and object naming	(1) Functions/commands have well chosen names (2) Arguments/parameters have well chosen names	[41, 44]
Code style	(1) Code has a consistent style (2) Code follows (basic) good practices in the programming language of choice, for example PEP8 or the tidyverse style guide	[41, 44, 42]
Code duplication	Duplicated code is minimal	[41, 44]
Self-contained functions	The method is exposed to the user as self-contained functions or commands	[48, 39, 45]
Plotting	Plotting functions are provided for the final and/or intermediate results	
Dummy proofing	Package contains dummy proofing, i.e. testing whether the parameters and data supplied by the user make sense and are useful	[43, 47]
Code assurance		
Unit testing	Method is tested using unit tests	[43, 41, 48, 44, 45]
Unit testing	Tests are run automatically using functionality from the programming language	[43, 41, 48, 44, 45]
Continuous integration	The method uses continuous integration, for example on Travis CI	[49, 44, 42, 45]
Code coverage	(1) The code coverage of the repository is assessed. (2) What is the percentage of code coverage	

Based on the results of our benchmark, we propose a set of practical guidelines for method users (Figure 3.10 and https://guidelines.dynverse.org). We postulate that, as a method's performance is heavily dependent on the trajectory type being studied, the choice of method should currently be primarily driven by the anticipated trajectory topology in the data. For most use cases, the user will know very little about the expected trajectory, except perhaps whether the data is expected to contain multiple disconnected trajectories, cycles or a complex tree structure. In each of these use cases, our evaluation suggests a different set of optimal methods, as shown in Figure 3.10. Several other factors will also impact the choice of methods, such as the dimensions of the dataset and the prior information that is available. These factors and several others can all be dynamically explored in our interactive application

Go to the next chapter for more acknowledgements.

Table 3.2: Scoring sheet for assessing usability of trajectory inference methods. Part 2.

Aspect	Items	References
Documentation		
Support	(1) There is a support ticket system, for example on Github (2) The authors respond to tickets and issues are resolved within a reasonable time frame	[41, 44, 42, 45, 46]
Development model	(1) The repository separates the development code from master code, for example using git master en developer branches (2) The repository has created releases, or several branches corresponding to major releases. (3) The repository has branches for the development of separate features.	[50]
Tutorial	(1) A tutorial or vignette is available (2) The tutorial has example results (3) The tutorial has real example data (4) The tutorial showcases the method on several datasets (1=0, 2=0.5, >2=1)	[44, 45, 46, 47, 51]
Function documentation	(1) The purpose and usage of functions/commands is documented (2) The parameters of functions/commands are documented (3) The output of functions/commands is documented	[41, 39, 44, 45, 47]
Inline documentation	Inline documentation is present in the code	[41, 39, 44, 45, 47]
Parameter transparency	All important parameters are exposed to the user	[39]
Behaviour		
Seed setting	The method does not artificially become deterministic (1), for example by setting some (0.5) or a lot (0) of seeds	[52]
Unexpected output	(1) No unexpected output messages are generated by the method (2) No unexpected files, folders or plots are generated (3) No unexpected warnings during runtime or compilation are generated	[42]
Trajectory format	The postprocessing necessary to extract the relevant output from the method is minimal (1), moderate (0.5) or extensive (0)	
Prior information	Prior information is required (0), optional (1) or not required (1)	
Paper		
Publishing	The method is published	
Peer review	The paper is published in a peer-reviewed journal	[47, 53, 54]
Evaluation on real data	(1) The paper shows the method's usefulness on several (1), one (0.25) or no real datasets. (2) The paper quantifies the accuracy of the method given a gold or silver standard trajectory	[55, 56]
Evaluation of robustness	The paper assessed method robustness (to eg. noise, subsampling, parameter changes, stability) in one (0.5) or several (1) ways	[47, 55, 51, 56]

(https://guidelines.dynverse.org). This application can also be used to query the results of this evaluation, such as filtering the datasets or changing the importance of the evaluation metrics for the final ranking.

When inferring a trajectory on a dataset of interest, it is important to take two further points into account. First, it is critical that a trajectory, and the downstream results and/or hypotheses originating from it, are confirmed by multiple TI methods. This is to make sure that the prediction is not biased due to the given parameter setting or the particular algorithm underlying a TI method. The value of using different methods is further supported by our analysis indicating substantial complementarity between the different methods. Second, even if the expected topology is known, it can be beneficial to also try out methods that make less assumptions about the trajectory topology. When the expected topology is confirmed using such a method, it provides additional evidence to the user. When a more complex topology is produced, this could indicate that the underlying biology is much more complex than anticipated by

the user.

Critical to the broad applicability of TI methods is the standardization of the input and output interfaces of TI methods, so that users can effortlessly execute TI methods on their dataset of interest, compare different predicted trajectories and apply downstream analyses, such as finding genes important for the trajectory, network inference [11] or finding modules of genes [57]. Our framework is an initial attempt at tackling this problem, and we illustrate its usefulness here by comparing the predicted trajectories of several top-performing methods on datasets containing a linear, tree, cyclic and disconnected graph topology (Figure 3.11). Using our framework, this figure can be recreated using only a couple of lines of R code (https://methods.dynverse.org). In the future, this framework could be extended to allow additional input data, such as spatial and RNA velocity information [58], and easier downstream analyses. In addition, further discussion within the field is required to arrive at a consensus concerning a common interface for trajectory models, which can include additional features such as uncertainty and gene importance.

Our study indicates that the field of trajectory inference is maturing, primarily for linear and bifurcating trajectories (Figure 3.11a,b). However, we also highlight several ongoing challenges, which should be addressed before TI can be a reliable tool for analyzing single-cell omics datasets with complex trajectories. Foremost, new methods should focus on improving the unbiased inference of tree, cyclic graph and disconnected topologies, as we found that methods repeatedly overestimate or underestimate the complexity of the underlying topology, even if the trajectory could easily be identified using a dimensionality reduction method (Figure 3.11c,d). Furthermore, higher standards for code assurance and documentation could help in adopting these tools across the single-cell omics field. Finally, new tools should be designed to scale well with the increasing number of cells and features. We found that only a handful of current methods can handle datasets with more than 10,000 cells within a reasonable time frame. To support the development of these new tools, we provide a series of vignettes on how to wrap and evaluate a method on the different measures proposed in this study at https://benchmark.dynverse.org.

We found that the performance of a method can be very variable between datasets, and therefore included a large set of both real and synthetic data within our evaluation, leading to a robust overall ranking of the different methods. However, 'good-yet-not-the-best' methods [59] can still provide a very valuable contribution to the field, especially if they make use of novel algorithms, return a more scalable solution or provide a unique insight in specific use cases. This is also supported by our analysis of method complementarity. Some examples for the latter include PhenoPath, which can include additional covariates in its model, ouija, which returns a measure of un-

certainty of each cell's position within the trajectory, and StemID, which can infer the directionality of edges within the trajectory.

3.4 Methods

3.4.1 Trajectory inference methods

We gathered a list of 71 trajectory inference tools (Supplementary Table 3.1) by searching the literature for 'trajectory inference' and 'pseudotemporal ordering', and based on two existing lists found online: awesome-single-cell [17] and single-cell-pseudotime [60]. We welcome any contributions by creating an issue at https://methods.dynverse.org.

Methods were excluded from the evaluation based on several criteria: (1) not freely available; (2) no code available; (3) superseded by another method; (4) requires data types other than expression; (5) no programming interface; (6) unresolved errors during wrapping; (7) too slow (requires more than 1 h on a 100×100 dataset); (8) does not return an ordering; and (9) requires additional user input during the algorithm (other than prior information). The discussions on why these methods were excluded can be found at https://github.com/dynverse/dynmethods/issues?q=label:unwrappable. In the end, we included 45 methods in the evaluation.

3.4.2 Method wrappers

To make it easy to run each method in a reproducible manner, each method was wrapped within Docker and Singularity containers (available at https://methods.dynverse.org). These containers are automatically built and tested using Travis continuous integration (https://travis-ci.org/dynverse) and can be ran using both Docker and Singularity. For each method, we wrote a wrapper script based on example scripts or tutorials provided by the authors (as mentioned in the respective wrapper scripts). This script reads in the input data, runs the method and outputs the files required to construct a trajectory. We also created a script to generate an example dataset, which is used for automated testing.

We used the GitHub issues system to contact the authors of each method, and asked for feedback on the wrappers, the metadata and the usability scores. About one-third of the authors responded and we improved the wrappers based on their feedback. These discussions can be viewed on GitHub: https://github.com/dynverse/dynmethods/issues?q=label:method_discussion

Method input

As input, we provided each method with either the raw count data (after cell and gene filtering) or normalized expression values, based on the description in the method documentation or from the study describing the method. A large portion of the methods requires some form of prior information (for example, a start cell) to be executable. Other methods optionally allow the exploitation of certain prior information. Prior information can be supplied as a starting cell from which the trajectory will originate, a set of important marker genes or even a grouping of cells into cell states. Providing prior information to a TI method can be both a blessing and a curse. In one way, prior information can help the method to find the correct trajectory among many, equally likely, alternatives. On the other hand, incorrect or noisy prior information can bias the trajectory towards current knowledge. Moreover, prior information is not always easily available, and its subjectivity can therefore lead to multiple equally plausible solutions, restricting the applicability of such TI methods to well-studied systems.

The prior information was extracted from the reference trajectory as follows:

- **Start cells**: the identity of one or more start cells. For both real and synthetic data, a cell was chosen that was the closest (in geodesic distance) to each milestone with only outgoing edges. For ties, one random cell was chosen. For cyclic datasets, a random cell was chosen.
- **End cells**: the identity of one or more end cells. This is similar to the start cells, but now for every state with only incoming edges.
- **No. of end states**: number of terminal states, i.e., the number of milestones with only incoming edges.
- **Grouping**: for each cell a label showing which state/cluster/branch it belongs to. For real data, the states were from the gold/silver standard. For synthetic data, each milestone was seen as one group and cells were assigned to their closest milestone.
- **No. of branches**: number of branches/intermediate states. For real data, this was the number of states in the gold/silver standard. For synthetic data, this was the number of milestones.
- **Discrete time course**: for each cell a time point from which it was sampled. If available, this was directly extracted from the reference trajectory; otherwise the geodesic distance from the root milestone was used. For synthetic data, the simulation time was uniformly discretised into four timepoints.
- **Continuous time course**: for each cell a time point from which it was sampled. For real data, this was equal to the discrete time course. For synthetic data, we

used the internal simulation time of each simulator.

Common trajectory model

Due to the absence of a common format for trajectory models, most methods return a unique set of output formats with few overlaps. We therefore post-processed the output of each method into a common probabilistic trajectory model (Figure 3.2a). This model consisted of three parts. (1) The milestone network represents the overall network topology, and contains edges between different milestones and the length of the edge between them. (2) The milestone percentages contain, for each cell, its position between milestones and sums for each cell to one. (3) The regions of delayed commitment define connections between three or more milestones. These must be explicitly defined in the trajectory model and per region one milestone must be directly connected to all other milestones of the region.

Depending on the output of a method, we used different strategies to convert the output to our model (Figure 3.2b). Special conversions are denoted by an asterisk (*) and will be explained in more detail in the second list below.

- **Type 1, direct**: CALISTA*, DPT*, ElPiGraph, ElPiGraph cycle, ElPiGraph linear, MERLoT, PAGA, SLICE*, Slingshot, URD* and Wishbone. The wrapped method directly returned a network of milestones, the regions of delayed commitment and for each cell it is given to what extent it belongs to a milestone. In some cases, this indicates that additional transformations were required for the method, not covered by any of the following output formats. Some methods returned a branch network instead of a milestone network and this network was converted by calculating the line graph of the branch network.
- **Type 2, linear pseudotime**: Component 1, Embeddr, FORKS, MATCHER, ouija, ouijaflow, PhenoPath, pseudogp, SCIMITAR, SCORPIUS, topslam, TSCAN, Wanderlust and Waterfall. The method returned a pseudotime, which is translated into a linear trajectory where the milestone network contains two milestones and cells are positioned between these two milestones.
- **Type 3, cyclical pseudotime**: Angle and reCAT. The method returned a pseudotime, which is translated into a cyclical trajectory where the milestone network contains three milestones and cells are positioned between these three milestones. These milestones were positioned at pseudotime 0, 1/3 and 2/3.
- **Type 4, end state probability**: FateID, GPfates, GrandPrix, MFA*, SCOUP and STEMNET. The method returned a pseudotime and for each cell and end state a probability (Pr) for how likely a cell will end up in a certain end state. This was translated into a star-shaped milestone network, with one starting milestone

(M0) and several outer milestones (Mi), with regions of delayed commitment between all milestones. The milestone percentage of a cell to one of the outer milestones was equal to pseudotime$\times$PrMi. The milestone percentage to the starting milestone was equal to 1 $-$ pseudotime.

- **Type 5, cluster assignment**: Mpath and SCUBA. The method returned a milestone network and an assignment of each cell to a specific milestone. Cells were positioned onto the milestones they are assigned to, with milestone percentage equal to 1.
- **Type 6, orthogonal projection**: MST, pCreode and RaceID/StemID. The method returned a milestone network, and a dimensionality reduction of the cells and milestones. The cells were projected to the closest nearest segment, thus determining the cells' position along the milestone network. If a method also returned a cluster assignment (type 5), we limited the projection of each cell to the closest edge connecting to the milestone of a cell. For these methods, we usually wrote two wrappers, one which included the projection and one without.
- **Type 7, cell graph**: CellRouter, CellTrails, cellTree Gibbs, cellTree maptpx, cellTree VEM, Monocle DDRTree, Monocle ICA, Sincell* and SLICER. The method returned a network of cells and which cell–cell transitions were part of the 'backbone' structure. Backbone cells with degree $\neq 2$ were regarded as milestones and all other cells were placed on transitions between the milestones. If a method did not return a distance between pairs of cells, the cells were uniformly positioned between the two milestones. Otherwise, we first calculated the distance between two milestones as the sum of the distances between the cells and then divided the distance of each pair of cells with the total distance to get the milestone percentages.

Special conversions were necessary for certain methods:

- **CALISTA**: We assigned the cells to the branch at which the sum of the cluster probabilities of two connected milestones was the highest. The cluster probabilities of the two selected milestones were then used as milestone percentages. This was then processed as a type 1, direct, method.
- **DPT**: We projected the cells onto the cluster network, consisting of a central milestone (this cluster contained the cells that were assigned to the 'unknown' branch) and three terminal milestones, each corresponding to a tip point. This was then processed as a type 1, direct, method.
- **Sincell**: To constrain the number of milestones this method creates, we merged two cell clusters iteratively until the percentage of leaf nodes was below a certain

cutoff, with the default cutoff set to 25%. This was then processed as a type 7, cell graph, method.

- **SLICE**: As discussed in the vignette of SLICE (https://research.cchmc.org/pbge/slice.html), we ran principal curves one by one for every edge detected by SLICE. This was then processed as a type 1, direct, method.
- **MFA**: We used the branch assignment as state probabilities, which together with the global pseudotime were processed as a type 4, end state probabilities, method.
- **URD**: We extracted the pseudotime of a cell within each branch using the y positions in the tree layout. This was then further processed as a type 1, direct, method.

More information on how each method was wrapped can be found within the comments of each wrapper script, listed at https://methods.dynverse.org.

Off-the-shelf methods

For baseline performance, we added several 'off-the-shelf' TI methods that can be run using a few lines of code in R.

- **Component 1**: This method returns the first component of a principal component analysis (PCA) dimensionality reduction as a linear trajectory. This method is especially relevant as it has been used in a few studies already [61, 62].
- **Angle**: Similar to the previous method, this method computes the angle with respect to the origin in a two-dimensional PCA and uses this angle as a pseudotime for generating a cyclical trajectory.
- **MST**: This method performs PCA dimensionality reduction, followed by clustering using the R mclust package, after which the clusters are connected using a minimum spanning tree. The trees are orthogonally projected to the nearest segment of the tree. This baseline is highly relevant as many methods follow the same methodology: dimensionality reduction, clustering, topology inference and project cells to topology.

3.4.3 Trajectory types

We classified all possible trajectory topologies into distinct trajectory types, based on topological criteria (Figure 3.1c). These trajectory types start from the most general trajectory type, a disconnected graph, and move down (within a directed acyclic graph

structure), progressively becoming more simple until the two basic types are reached: linear and cyclical. A disconnected graph is a graph in which only one edge can exist between two nodes. A (connected) graph is a disconnected graph in which all nodes are connected. An acyclic graph is a graph containing no cycles. A tree is an acyclic graph containing no convergences (no nodes with in-degree higher than 1). A convergence is an acyclic graph in which only one node has a degree larger than 1 and this same node has an in-degree of 1. A multifurcation is a tree in which only one node has a degree larger than 1. A bifurcation is a multifurcation in which only one node has a degree equal to 3. A linear topology is a graph in which no node has a degree larger than 3. Finally, a cycle is a connected graph in which every node has a degree equal to 2. In most cases, a method that was able to detect a complex trajectory type was also able to detect less complex trajectory types, with some exceptions shown in Figure 3.3a.

For simplicity, we merged the bifurcation and convergence trajectory type, and the acyclic graph and connected graph trajectory type in the main figures of the paper.

3.4.4 Real datasets

We gathered real datasets by searching for 'single-cell' at the Gene Expression Omnibus and selecting those datasets in which the cells are sampled from different stages in a dynamic process (Supplementary Table 3.2). The scripts to download and process these datasets are available on our repository (https://benchmark.dynverse.org/tree/master/scripts/01-datasets). Whenever possible, we preferred to start from the raw counts data. These raw counts were all normalized and filtered using a common pipeline, as discussed later. Some original datasets contained more than one trajectory, in which case we split the dataset into its separate connected trajectory, but also generated several combinations of connected trajectories to include some datasets with disconnected trajectories in the evaluation. In the end, we included 110 datasets for this evaluation.

For each dataset, we extracted a reference trajectory, consisting of two parts: the cellular grouping (milestones) and the connections between these groups (milestone network). The cellular grouping was provided by the authors of the original study, and we classified it as a gold standard when it was created independently from the expression matrix (such as from cell sorting, the origin of the sample, the time it was sampled or cellular mixing) or as a silver standard otherwise (usually by clustering the expression values). To connect these cell groups, we used the original study to determine the network that the authors validated or otherwise found to be the most likely. In the end, each group of cells was placed on a milestone, having a percentage of 1 for that particular milestone. The known connections between these groups were

used to construct the milestone network. If there was biological or experimental time data available, we used this as the length of the edge; otherwise we set all the lengths equal to one.

3.4.5 Synthetic datasets

To generate synthetic datasets, we used four different synthetic data simulators:

- **dyngen**: simulations of gene regulatory networks, available at https://github.com/dynverse/dyngen
- **dyntoy**: random gradients of expression in the reduced space, available at https://github.com/dynverse/dyntoy
- **PROSSTT**: expression is sampled from a linear model that depends on pseudotime [63]
- **Splatter**: simulations of non-linear paths between different expression states [35]

For every simulator, we took great care to make the datasets as realistic as possible. To do this, we extracted several parameters from all real datasets. We calculated the number of differentially expressed features within a trajectory using a two-way Mann–Whitney U test between every pair of cell groups. These values were corrected for multiple testing using the Benjamini-Hochberg procedure (FDR < 0.05) and we required that a gene was expressed in at least 5% of cells, and had at least a fold-change of 2. We also calculated several other parameters, such as drop-out rates and library sizes using the Splatter package [35]. These parameters were then given to the simulators when applicable, as described for each simulator below. Not every real dataset was selected to serve as a reference for a synthetic dataset. Instead, we chose a set of ten distinct reference real datasets by clustering all the parameters of each real dataset, and used the reference real datasets at the cluster centers from a pam clustering (with $k = 10$, implemented in the R cluster package) to generate synthetic data.

dyngen

The dyngen (Chapter 2) workflow to generate synthetic data is based on the well established workflow used in the evaluation of network inference methods [34, 64] and consists of four main steps: network generation, simulation, gold standard extraction and simulation of the scRNA-seq experiment. At every step, we tried to mirror real regulatory networks, while keeping the model simple and easily extendible We simulated a total of 110 datasets, with 11 different topologies.

dyntoy

For more simplistic data generation ("toy" datasets), we created the dyntoy workflow (https://github.com/dynverse/dyntoy). We created 12 topology generators (described below), and with 10 datasets per generator, this lead to a total of 120 datasets.

We created a set of topology generators, were $B(n, p)$ denotes a binomial distribution, and $U(a, b)$ denotes a uniform distribution:

- Linear and cyclic, with number of milestones $\sim B(10, 0.25)$
- Bifurcating and converging, with four milestones
- Binary tree, with number of branching points $\sim U(3, 6)$
- Tree, with number of branching points $\sim U(3, 6)$ and maximal degree $\sim U(3, 6)$

For more complex topologies we first calculated a random number of "modifications" $\sim U(3, 6)$ and a $deg_{max} \sim B(10, 0.25) + 1$. For each type of topology, we defined what kind of modifications are possible: divergences, loops, convergences and divergence-convergence. We then iteratively constructed the topology by uniformly sampling from the set of possible modifications, and adding this modification to the existing topology. For a divergence, we connected an existing milestone to a number of a new milestones. Conversely, for a convergence we connected a number of new nodes to an existing node. For a loop, we connected two existing milestones with a number of milestones in between. Finally for a divergence-convergence we connected an existing milestone to several new milestones which again converged on a new milestone. The number of nodes was sampled from $\sim B(deg_{max} - 3, 0.25) + 2$

- Looping, allowed loop modifications
- Diverging-converging, allowed divergence and converging modifications
- Diverging with loops, allowed divergence and loop modifications
- Multiple looping, allowed looping modifications
- Connected, allowed looping, divergence and convergence modifications
- Disconnected, number of components sampled from $\sim B(5, 0.25) + 2$, for each component we randomly chose a topology from the ones listed above

After generating the topology, we sampled the length of each edge $\sim U(0.5, 1)$. We added regions of delayed commitment to a divergence in a random half of the cases. We then placed the number of cells (same number as from the reference real dataset), on this topology uniformly, based on the length of the edges in the milestone network.

For each gene (same number as from the reference real dataset), we calculated the Kamada-Kawai layout in 2 dimensions, with edge weight equal to the length of the edge. For this gene, we then extracted for each cell a density value using a bivariate normal distribution with $\mu \sim U(x_{min}, x_{min})$ and $\sigma \sim U(x_{min}/10, x_{min}/8)$. We used this density as input for a zero-inflated negative binomial distribution with $\mu\, U(100, 1000) \times$ *density*, $k\ U(\mu/10, \mu/4)$ and pi from the parameters of the reference real dataset, to get the final count values.

This count matrix was then filtered and normalised using the pipeline described below.

PROSSTT

PROSSTT is a recent data simulator [63], which simulates expression using linear mixtures of expression programs and random walks through the trajectory. We used 5 topology generators from dyntoy (linear, bifurcating, multifurcating, binary tree and tree), and simulated for each topology generator 10 datasets using different reference real datasets. However, due to frequent crashes of the tool, only 19 datasets created output and were thus used in the evaluation.

Using the simulate_lineage function, we simulated the lineage expression, with parameters $a \sim U(0.01, 0.1)$, *branch-tol*$_{intra} \sim U(0, 0.9)$ and *branch-tol*$_{inter} \sim U(0, 0.9)$. These parameter distributions were chosen very broad so as to make sure both easy and difficult datasets are simulated. After simulating base gene expression with simulate_base_gene_exp, we used the sample_density function to finally simulate expression values of a number of cells (the same as from the reference real dataset), with $\alpha \sim$ *Lognormal* ($\mu = 0.3$ and $\sigma = 1.5$) and $\beta \sim$ *Lognormal* ($\mu = 2$ and $\sigma = 1.5$). Each of these parameters were centred around the default values of PROSSTT, but with enough variability to ensure a varied set of datasets.

This count matrix was then filtered and normalised using the pipeline described below.

Splatter

Splatter [35] simulates expression values by constructing non-linear paths between different states, each having a distinct expression profile. We used 5 topology generators from dyntoy (linear, bifurcating, multifurcating, binary tree and tree), and simulated for each topology generator 10 datasets using different reference real datasets, leading to a total of 50 datasets.

We used the splatSimulatePaths function from Splatter to simulate datasets, with number of cells and genes equal to those in the reference real dataset, and with parameters *nonlinearProb*, *sigmaFac* and *skew* all sampled from $U(0, 1)$.

3.4.6 Dataset filtering and normalization

We used a standard single-cell RNA-seq preprocessing pipeline that applies parts of the scran and scater Bioconductor packages [65]. The advantages of this pipeline are that it works both with and without spike-ins, and it includes a harsh cell filtering that looks at abnormalities in library sizes, mitochondrial gene expression and the number of genes expressed using median absolute deviations (which we set to 3). We required that a gene was expressed in at least 5% of the cells and that it should have an average expression higher than 0.02. Furthermore, we used the pipeline to select the most highly variable genes, using a false discovery rate of 5% and a biological component higher than 0.5. As a final filter, we removed both all-zero genes and cells until convergence.

3.4.7 Benchmark metrics

The importance of using multiple metrics to compare complex models has been stated repeatedly [59]. Furthermore, a trajectory is a model with multiple layers of complexity, which calls for several metrics each assessing a different layer. We therefore defined several possible metrics for comparing trajectories, each investigating different layers. These are all discussed in Section 3.6 along with examples and robustness analyses when appropriate.

Next, we created a set of rules to which we think a good trajectory metric should conform, and tested this empirically for each metric by comparing scores before and after perturbing a dataset (Section 3.6). Based on this analysis, we chose four metrics for the evaluation, each assessing a different aspect of the trajectory: (1) the HIM measures the topological similarity; (2) the F1branches compares the branch assignment; (3) the cordist assesses the similarity in pairwise cell–cell distances and thus the cellular positions; and (4) the wcorfeatures looks at whether similar important features (genes) are found in both the reference dataset and the prediction.

The Hamming–Ipsen–Mikhailov metric

The HIM metric [66] uses the two weighted adjacency matrices of the milestone networks as input (weighted by edge length). It is a linear combination of the normalized Hamming distance, which gives an indication of the differences in edge lengths, and the normalized Ipsen–Mikhailov distance, which assesses the similarity in degree distributions. The latter has a parameter γ, which was fixed at 0.1 to make the scores comparable between datasets. We illustrate the metric and discuss alternatives in Section 3.6.

The F1 between branch assignments

To compare branch assignment, we used an F1 score, also used used for comparing biclustering methods [57]. To calculate this metric, we first calculated the similarity of all pairs of branches between the two trajectories using the Jaccard similarity. Next, we defined the 'Recovery' (respectively 'Relevance') as the average maximal similarity of all branches in the reference dataset (respectively prediction). The F1branches was then defined as the harmonic mean between Recovery and Relevance. We illustrate this metric further in Section 3.6.

Correlation between geodesic distances

When the position of a cell is the same in both the reference and the prediction, its relative distances to all other cells in the trajectory should also be the same. This observation is the basis for the cordist metric. To calculate the cordist, we first sampled 100 waypoint cells in both the prediction and the reference dataset, using stratified sampling between the different milestones, edges and regions of delayed commitment, weighted by the number of cells in each collection. We then calculated the geodesic distances between the union of waypoint cells from both datasets and all other cells. The calculation of the geodesic distance depended on the location of the two cells within the trajectory, further discussed in Section 3.6, and was weighted by the length of the edge in the milestone network. Finally, the cordist was defined as the Spearman rank correlation between the distances of both datasets. We illustrate the metric and assess the effect of the number of waypoint cells in Section 3.6.

The correlation between important features

The wcorfeatures assesses whether the same differentially expressed features are found using the predicted trajectory as in the known trajectory. To calculate this metric, we used Random Forest regression (implemented in the R ranger package [67]), to predict expression values of each gene, based on the geodesic distances of a cell to each milestone. We then extracted feature importance values for each feature and calculated the similarity of the feature importances using a weighted Pearson correlation, weighted by the feature importance in the reference dataset to give more weight to large differences. As hyperparameters we set the number of trees to 10,000 and the number of features on which to split to 1% of all available features. We illustrate this metric and assess the effect of its hyperparameters in Section 3.6.

Score aggregation

To rank methods, we needed to aggregate the different scores on two levels: across datasets and across different metrics. This aggregation strategy is explained in more detail in Section 3.6.

To ensure that easy and difficult datasets have equal influence on the final score, we first normalized the scores on each dataset across the different methods. We shifted and scaled the scores to $\sigma = 1$ and $\mu = 0$, and then applied the unit probability density function of a normal distribution on these values to get the scores back into the [0,1] range.

Since there is a bias in dataset source and trajectory type (for example, there are many more linear datasets), we aggregated the scores per method and dataset in multiple steps. We first aggregated the datasets with the same dataset source and trajectory type using an arithmetic mean of their scores. Next, the scores were averaged over different dataset sources, using an arithmetic mean that was weighted based on how much the synthetic and silver scores correlated with the real gold scores. Finally, the scores were aggregated over the different trajectory types again using an arithmetic mean.

Finally, to get an overall benchmarking score, we aggregated the different metrics using a geometric mean.

3.4.8 Method execution

Each execution of a method on a dataset was performed in a separate task as part of a gridengine job. Each task was allocated one CPU core of an Intel(R) Xeon(R) CPU E5-2665 at 2.40 GHz, and one R session was started for each task. During the execution of a method on a dataset, if the time limit (>1 h) or memory limit (16 GB) was exceeded, or an error was produced, a zero score was returned for that execution.

3.4.9 Complementarity

To assess the complementarity between different methods, we first calculated for every method and dataset whether the overall score was equal to or higher than 95% of the best overall score for that particular dataset. We then calculated for every method the weighted percentage of datasets that fulfilled this rule, weighted similarly as in the benchmark aggregation, and chose the best method. We iteratively added new methods until all methods were selected. For this analysis, we did not include any methods

that require any strong prior information and only included methods that could detect the trajectory types present in at least one of the datasets.

3.4.10 Scalability

To assess the scalability of each method, we started from five real datasets, selected using the centers from a k-medoids as discussed before. We up- and downscaled these datasets between 10 and 100,000 cells and 10 and 100,000 features, while never going higher than 1,000,000 values in total. To generate new cells or features, we first generated a 10-nearest-neighbor graph of both the cells and features from the expression space. For every new cell or feature, we used a linear combination of one to three existing cells or features, where each cell or feature was given a weight sampled from a uniform distribution between 0 and 1.

We ran each method on each dataset for maximally 1 h and gave each process 10 GB of memory. To determine the running time of each method, we started the timer right after data loading and the loading of any packages, and stopped the clock before postprocessing and saving of the output. Pre- and postprocessing steps specific to a method, such as dimensionality reduction and gene filtering, were included in the time. To estimate the maximal memory usage, we used the max_vmem value from the qacct command provided by a gridengine cluster. We acknowledge, however, that these memory estimates are very noisy and the averages provided in this study are therefore only rough estimates.

The relationship between the dimensions of a dataset and the running time or maximal memory usage was modeled using shape constrained additive models [38], with $log_{10}|\textit{cells}|$ and $log_{10}|\textit{features}|$ as predictor variables, and fitted this model using the scam function as implemented in the R scam package, with $log_{10}\textit{time}$ (or $log_{10}\textit{memory}$) as outcome.

To classify the time complexity of each method with respect to the number of cells, we predicted the running time at 10,000 features with increasing number of cells from 100 to 100,000, with steps of 100. We trained a generalized linear model with the following function: $y \approxeq \log x + \sqrt{x} + x + x^2 + x^3$ with y as running time and x as the number of cells or features. The time complexity of a method was then classified using the weights w from this model:

$$\begin{cases} \text{superquadratic} & \text{if } w_{x^3} > 0.25, \\ \text{quadratic} & \text{if } w_{x^2} > 0.25, \\ \text{linear} & \text{if } w_x > 0.25, \\ \text{sublinear} & \text{if } w_{\log(x)} > 0.25 \text{ or } w_{\text{sqrt}(x)} > 0.25, \\ \text{case with highest weight} & \text{else.} \end{cases}$$

This process was repeated for classifying the time complexity with respect to the number of features, and the memory complexity both with respect to the number of cells and features.

3.4.11 Stability

In the ideal case, a method should produce a similar trajectory, even when the input data is slightly different. However, running the method multiple times on the same input data would not be the ideal approach to assess its stability, given that a lot of tools are artificially deterministic by internally resetting the pseudorandom number generator (for example, using the 'set.seed' function in R or the 'random.seed' function in numpy). To assess the stability of each method, we therefore selected a number of datasets, which consisted of 25% of the datasets accounting for 15% of the total runtime, chosen such that after aggregation the overall scores still has > 0.99 correlation with the original overall ranking. We subsampled each dataset 10 times with 95% of the original cells and 95% of the original features. We ran every method on each of the bootstraps, and assessed the stability by calculating the benchmarking scores between each pair of subsequent models (run i is compared to run $i + 1$). For the cordist and F1branches, we only used the intersection between the cells of two datasets, while the intersection of the features was used for the wcorfeatures.

3.4.12 Usability

We created a transparent scoring scheme to quantify the usability of each method based on several existing tool quality and programming guidelines in the literature and online (Table 3.1). The main goal of this quality control is to stimulate the improvement of current methods, and the development of user- and developer-friendly new methods. The quality control assessed six categories, each looking at several aspects, which are further divided into individual items. The availability category checks whether the method is easily available, whether the code and dependencies can be easily installed, and how the method can be used. The code quality assesses the quality of the code both from a user perspective (function naming, dummy proofing

and availability of plotting functions) and a developer perspective (consistent style and code duplication). The code assurance category is frequently overlooked, and checks for code testing, continuous integration [49] and an active support system. The documentation category checks the quality of the documentation, both externally (tutorials and function documentation) and internally (inline documentation). The behavior category assesses the ease by which the method can be run, by looking for unexpected output files and messages, prior information and how easy the trajectory model can be extracted from the output. Finally, we also assessed certain aspects of the study in which the method was proposed, such as publication in a peer-reviewed journal, the number of datasets in which the usefulness of the method was shown and the scope of method evaluation in the paper.

Each quality aspect received a weight depending on how frequently it was found in several papers and online sources that discuss tool quality (Table 3.1). This was to make sure that more important aspects, such as the open source availability of the method, outweighed other less important aspects, such as the availability of a graphical user interface. For each aspect, we also assigned a weight to the individual questions being investigated (Table 3.1). For calculating the final score, we weighed each of the six categories equally.

3.4.13 Guidelines

For each set of outcomes in the guidelines figure, we selected one to four methods, by first filtering the methods on those that can detect all required trajectory types, and ordering the methods according to their average accuracy score on datasets containing these trajectory types (aggregated according to the scheme presented in the section Accuracy).

We used the same approach for selecting the best set of methods in the guidelines app (http://guidelines.dynverse.org), developed using the R shiny package. This app will also filter the methods, among other things, depending on the predicted running time and memory requirements, the prior information available and the preferred execution environment (using the dynmethods package or standalone).

3.4.14 Reporting Summary

Further information on research design is available in the Nature Research Reporting Summary, available at https://www.nature.com/articles/s41587-019-0071-9#MOESM2.

3.5 Supplementary Figures and Tables

Supplementary Table 3.1: Overview of available trajectory inference tools, and whether they were included in this study. Download link: https://static-content.springer.com/esm/art%3A10.1038%2Fs41587-019-0071-9/MediaObjects/41587_2019_71_MOESM3_ESM.xlsx.

Supplementary Table 3.2: Overview of the real datasets used in this study. Download link: https://static-content.springer.com/esm/art%3A10.1038%2Fs41587-019-0071-9/MediaObjects/41587_2019_71_MOESM4_ESM.xlsx.

Supplementary Figure 3.1: Results from the evaluation, for all methods and across all evaluation criteria. (a) We characterized the methods according to the wrapper type, their required priors, whether the inferred topology is constrained by the algorithm (fixed) or a parameter (param), and the types of inferable topologies. The methods are grouped vertically based on the most complex trajectory type the y can infer. (b) The overall results of the evaluation on four criteria: benchmarking using a reference trajectory on real and synthetic data, scalability with increasing number of cells and features, stability across dataset subsamples, and quality of the implementation. (c) Accuracy of trajectory inference methods across metrics, dataset sources and dataset trajectory types. The performance of a method is generally more stable across dataset sources, but very variable depending on the metric and trajectory type. (d) Predicted execution times and memory usage for varying numbers of cells and features (# cells $\times$ # features). Predictions were made by training a regression model after running each method on bootstrapped datasets with varying numbers of cells and features. (e) Stability results by calculating the average pairwise similarity between models inferred across multiple runs of the same method. (f) Usability scores of the tool and corresponding manuscript, grouped per category. Download link: https://github.com/dynverse/dynbenchmark_results/raw/master/08-summary/results_suppfig.pdf.

3.6 Supplementary Note 1: Metrics to compare two trajectories

A trajectory, as defined in our evaluation, is a model with multiple abstractions. The top abstraction is the topology which contains information about the paths each cell can take from their starting point. Deeper abstractions involve the mapping of each

cell to a particular branch within this network, and the position (or ordering) of each cells within these branches. Internally, the topology is represented by the milestone network and regions of delayed commitment, the branch assignment and cellular positions are represented by the milestone percentages (Figure 3.12).

Given the multilayered complexity of a trajectory model, it is not trivial to compare the similarity of two trajectory models using only one metric. We therefore sought to use different comparison metrics, each serving a different purpose:

- **Specific metrics** investigate one particular aspect of the trajectory. Such metrics make it possible to find particular weak points for methods, e.g. that a method is very good at ordering but does not frequently find the correct topology. Moreover, having multiple individual metrics allow personalised rankings of methods, for example for users which are primarily interested in using the method correct topology.
- **Application metrics** focus on the quality of a downstream analysis using the trajectory. For example, it measures whether the trajectory can be used to find accurate differentially expressed genes.
- **Overall metrics** should capture all the different abstractions, in other words such metrics measure whether the resulting trajectory has a good topology, that the cells belong to similar branches *and* that they are ordered correctly.

Here, we first describe and illustrate several possible specific, application and overall metrics. Next, we test these metrics on several test cases, to make sure they robustly identify "wrong" trajectory predictions.

All metrics described here were implemented within the dyneval R package (https://github.com/dynverse/dyneval).

3.6.1 Metric characterisation and testing

isomorphic, *edgeflip* and *HIM*: Edit distance between two trajectory topologies

We used three different scores to assess the similarity in the topology between two trajectories, irregardless of where the cells were positioned.

For all three scores, we first simplified the topology of the trajectory to make both graph structures comparable:

- As we are only interested in the main structure of the topology without start or end, the graph was made undirected.

- All milestones with degree 2 were removed. For example in the topology A ⇒ B ⇒ C ⇒ D, C ⇒ D, the B milestone was removed
- A linear topology was converted to A ⇒ B ⇒ C
- A cyclical topology such as A ⇒ B ⇒ C ⇒ D or A ⇒ B ⇒ A were all simplified to A ⇒ B ⇒ C ⇒ A
- Duplicated edges such as A ⇒ B, A ⇒ B were decoupled to A ⇒ B, A ⇒ C ⇒ B

The *isomorphic* score returns 1 if two graphs are isomorphic, and 0 if they were not. For this, we used the used the BLISS algorithm [68], as implemented in the R igraph package.

The *edgeflip* score was defined as the minimal number of edges which should be added or removed to convert one network into the other, divided by the total number of edges in both networks. This problem is equivalent to the maximum common edge subgraph problem, a known NP-hard problem without a scalable solution [69]. We implemented a branch and bound approach for this problem, using several heuristics to speed up the search:

- First check all possible edge additions and removals corresponding to the number of different edges between the two graphs.
- For each possible solution, first check whether:
  1. The maximal degree is the same
  2. The minimal degree is the same
  3. All degrees are the same after sorting
- Only then check if the two graphs are isomorphic as described earlier.
- If no solution is found, check all possible solutions with two extra edge additions/removals.

The *HIM* metric (Hamming-Ipsen-Mikhailov distance) [66] which was adopted from the R nettools package (https://github.com/filosi/nettoolshttps://github.com/filosi/nettools). It uses an adjacency matrix which was weighted according to the lengths of each edges within the milestone network. Conceptually, *HIM* is a linear combination of:

- The normalised Hamming distance [70], which calculates the distance between two graphs by matching individual edges in the adjacency matrix, but disregards overall structural similarity.
- The normalised Ipsen-Mikhailov distance [71], which calculates the overall distance of two graphs based on matches between its degree and adjacency matrix, while disregarding local structural similarities. It requires a γ parameter, which

is usually estimated based on the number of nodes in the graph, but which we fixed at 0.1 so as to make the score comparable across different graph sizes.

We compared the three scores on several common topologies (Figure 3.13a). While conceptually very different, the *edgeflip* and *HIM* still produce similar scores (Figure 3.13b). The *HIM* tends to punish the detection of cycles, while the *edgeflip* is more harsh for differences in the number of bifurcations (Figure 3.13b). The main difference however is that the *HIM* takes into account edge lengths when comparing two trajectories, as illustrated in (Figure 3.13c). Short "extra" edges in the topology are less punished by the *HIM* than by the *edgeflip*.

To summarise, the different topology based scores are useful for different scenarios:

- If the two trajectories should only be compared when the topology is exactly the same, the *isomorphic* should be used.
- If it is important that the topologies are similar, but not necessarily isomorphic, the *edgeflip* is most appropriate.
- If the topologies should be similar, but shorter edges should not be punished as hard as longer edges, the *HIM* is most appropriate.

F1$_{branches}$ and *F1*$_{milestones}$: Comparing how well the cells are clustered in the trajectory

Perhaps one of the simplest ways to calculate the similarity between the cellular positions of two topologies is by mapping each cell to its closest milestone *or* branch 3.14. These clusters of cells can then be compared using one of the many external cluster evaluation measures [57]. When selecting a cluster evaluation metric, we had two main conditions:

- Because we allow methods to filter cells in the trajectory, the metric should be able to handle "non-exhaustive assignment", where some cells are not assigned to any cluster. - The metric should give each cluster equal weight, so that rare cell stages are equally important as large stages.

The *F1* score between the *Recovery* and *Relevance* is a metric which conforms to both these conditions. This metric will map two clustersets by using their shared members based on the *Jaccard* similarity. It then calculates the *Recovery* as the average maximal *Jaccard* for every cluster in the first set of clusters (in our case the reference trajectory). Conversely, the *Relevance* is calculated based on the average maximal similarity in the second set of clusters (in our case the prediction). Both the *Recovery* and *Relevance* are then given equal weight in a harmonic mean (*F1*). Formally, if C and C' are two cell clusters:

$$\textit{Jaccard}(c, c') = \frac{|c \cap c'|}{|c \cup c'|}$$

$$\textit{Recovery} = \frac{1}{|C|} \sum_{c \in C} \max_{c' \in C'} \textit{Jaccard(c, c')}$$

$$\textit{Relevance} = \frac{1}{|C'|} \sum_{c' \in C'} \max_{c \in C} \textit{Jaccard(c, c')}$$

$$\textit{F1} = \frac{2}{\frac{1}{\textit{Recovery}} + \frac{1}{\textit{Relevance}}}$$

cor_{dist}: Correlation between geodesic distances

When the position of a cell is the same in both the reference and the prediction, its *relative* distances to all other cells in the trajectory should also be the same. This observation is the basis for the cor_{dist} metric.

The geodesic distance is the distance a cell has to go through the trajectory space to get from one position to another. The way this distance is calculated depends on how two cells are positioned, showcased by an example in Figure 3.15:

- **Both cells are on the same edge in the milestone network.** In this case, the geodesic distance is defined as the product of the difference in milestone percentages and the length of their shared edge. For cells a and b in the example, $d(a, b)$ is equal to $1 \times (0.9 - 0.2) = 0.7$.
- **Cells reside on different edges in the milestone network.** First, the distance of the cell to all its nearby milestones is calculated, based on its percentage within the edge and the length of the edge. These distances in combination with the milestone network are used to calculate the shortest path distance between the two cells. For cells a and c in the example, $d(a, X) = 1 \times 0.9$ and $d(c, X) = 3 \times 0.2$, and therefore $d(a, c) = 1 \times 0.9 + 3 \times 0.2$.

The geodesic distance can be easily extended towards cells within regions of delayed commitment. When both cells are part of the same region of delayed commitment, the geodesic distance was defined as the manhattan distances between the milestone percentages weighted by the lengths from the milestone network. For cells d and e in the example, $d(d, e)$ is equal to $0 \times (0.3 - 0.2) + 2 \times (0.7 - 0.2) + 3 \times (0.4 - 0.1) = 1.9$. The distance between two cells where only one is part of a region of delayed commitment is calculated similarly to the previous paragraph, by first calculating the distance between the cells and their neighbouring milestones first, then calculating the shortest path distances between the two.

Calculating the pairwise distances between cells scales quadratically with the number of cells, and would therefore not be scaleable for large datasets. For this reason, a set of waypoint cells are defined *a priori*, and only the distances between the waypoint cells and all other cells is calculated, in order to calculate the correlation of geodesic distances of two trajectories (Figure 3.16a). These cell waypoints are determined by viewing each milestone, edge and region of delayed commitment as a collection of cells. We do stratified sampling from each collection of cells by weighing them by the total number of cells within that collection. For calculating the cor_{dist} between two trajectories, the distances between all cells and the union of both waypoint sets is computed.

To select the number of cell waypoints, we need to find a trade-off between the accuracy versus the time to calculate cor_{dist}. To select an optimal number of cell waypoints, we used the synthetic dataset with the most complex topology, and determined the cor_{dist} at different levels of both cell shuffling and number of cell waypoints (Figure 3.16a). We found that using cell waypoints does not induce a systematic bias in the cor_{dist}, and that its variability was relatively minimal when compared to the variability between different levels of cell shuffling when using 100 or more cell waypoints.

Although the cor_{dist}'s main characteristic is that it looks at the positions of the cells, other features of the trajectory are also (partly) captured. To illustrate this, we used the geodesic distances themselves as input for dimensionality reduction (Figure 3.17) with varying topologies. This reduced space captures the original trajectory structure quite well, including the overall topology and branch lengths.

$NMSE_{rf}$ and $NMSE_{lm}$: Using the positions of the cells within one trajectory to predict the cellular positions in the other trajectory

An alternative approach to detect whether the positions of cells are similar between two trajectories, is to use the positions of one trajectory to predict the positions within the other trajectory. If the cells are at similar positions in the trajectory (relative to its nearby cells), the prediction error should be low.

Specifically, we implemented two metrics which predict the milestone percentages from the reference by using the predicted milestone percentages as features (Figure 3.18). We did this with two regression methods, linear regression (*lm*, using the R lm function) and Random Forest (*rf*, implemented in the *ranger* package [67]). In both cases, the accuracy of the prediction was measured using the Mean Squared error (MSE), in the case of Random forest we used the out-of-bag mean-squared error. Next, we calculated MSE_{worst} equal to the MSE when predicting all milestone percentages as the average. We used this to calculate the normalised mean squared

error as $NMSE = 1 - \frac{MSE}{MSE_{worst}}$. We created a regression model for every milestone in the gold standard, and averaged the $NMSE$ values to finally obtain the *NMSE*$_{rf}$ and *NMSE*$_{lm}$ scores.

cor$_{features}$ and _wcor_$_{features}$: The accuracy of dynamical differentially expressed features/genes.

Although most metrics described above already assess some aspects directly relevant to the user, such as whether the method is good at finding the right topology, these metrics do not assess the quality of downstream analyses and hypotheses which can be generated from these models.

Perhaps the main advantage of studying cellular dynamic processes using single-cell -omics data is that the dynamics of gene expression can be studied for the whole transcriptome. This can be used to construct other models such as dynamic regulatory networks and gene expression modules. Such analyses rely on a "good-enough" cellular ordering, so that it can be used to identify dynamical differentially expressed genes.

To calculate the *cor*$_{features}$ we used Random forest regression to rank all the features according to their importance in predicting the positions of cells in the trajectory. More specifically, we first calculated the geodesic distances for each cell to all milestones in the trajectory. Next, we trained a Random Forest regression model (implemented in the R *ranger* package [67], https://github.com/imbs-hl/ranger) to predict these distances for each milestone, based on the expression of genes within each cell. We then extracted feature importances using the Mean Decrease in Impurity (importance = 'impurity' parameter of the ranger function), as illustrated in Figure 3.19. The overall importance of a feature (gene) was then equal to the mean importance over all milestones. Finally, we compared the two rankings by calculating the Pearson correlation, with values between -1 and 0 clipped to 0.

Random forest regression has two main hyperparameters. The number of trees to be fitted (num_tree parameter) was fixed to 10000 to provide accurate and stable estimates of the feature importance (Figure 3.20. The number of features on which can be split (mtry parameter) was set to 1% of all available features (instead of the default square-root of the number of features), as to make sure that predictive but highly correlated features, omnipresent in transcriptomics data, are not suppressed in the ranking.

For most datasets, only a limited number of features will be differentially expressed in the trajectory. For example, in the dataset used in Figure 3.20 only the top 10%-20% show a clear pattern of differential expression. The correlation will weight each of

these features equally, and will therefore give more weight to the bottom, irrelevant features. To prioritise the top differentially expressed features, we also implemented the $wcor_{features}$, which will weight the correlation using the feature importance scores in the reference so that the top features have relatively more impact on the score (Figure 3.21).

3.6.2 Metric conformity

Although most metrics described in the previous section make sense intuitively, this does not necessarily mean that these metrics are robust and will generate reasonable results when used for benchmarking. This is because different methods and datasets will all lead to a varied set of trajectory models:

- Real datasets have all cells grouped onto milestones
- Some methods place all cells in a region of delayed commitment, others never generate a region of delayed commitment
- Some methods always return a linear trajectory, even if a bifurcation is present in the data
- Some methods filter cells

A good metric, especially a good overall metric, should work in all these circumstances. To test this, we designed a set of rules to which a good metric should conform, and assessed empirically whether a metric conforms to these rules.

We generated a panel of toy datasets (using our *dyntoy* package, https://github.com/dynverse/dyntoy) with all possible combinations of:

- # cells: 10, 20, 50, 100, 200, 500
- # features: 200
- topologies: linear, bifurcation, multifurcating, tree, cycle, connected graph and disconnected graph
- Whether cells are placed on the milestones (as in real data) or on the edges/regions of delayed commitment between the milestones (as in synthetic data)

We then perturbed the trajectories in these datasets in certain ways, and tested whether the scores follow an expected pattern. An overview of the conformity of every metric is first given in Table 3.3. The individual rules and metric behaviour are discussed in the Supplementary Material that can be found at https://www.nature.com/articles/s41587-019-0071-9#Sec34.

Table 3.3: Overview of whether a particular metric conforms to a particular rule

name	cor_{dist}	$NMSE_{rf}$	$NMSE_{lm}$	*edgeflip*	*HIM*	isomorphic	$cor_{features}$	$wcor_{features}$	$F1_{branches}$	$F1_{milestones}$	$mean_{geometric}$
Same score on identity	✓	✗	✓	✓	✓	✓	✓	✗	✓	✓	✓
Local cell shuffling	✓	✓	✓	✗	✗	✗	✓	✓	✗	✓	✓
Edge shuffling	✓	✓	✓	✗	✗	✗	✓	✓	✓	✓	✓
Local and global cell shuffling	✓	✓	✓	✗	✗	✗	✓	✓	✓	✓	✓
Changing positions locally and/or globally	✓	✓	✓	✗	✗	✗	✓	✓	✗	✗	✓
Cell filtering	✓	✓	✓	✗	✗	✗	✓	✓	✓	✓	✓
Removing divergence regions	✓	✓	✓	✗	✗	✗	✓	✓	✗	✓	✓
Move cells to start milestone	✓	✓	✓	✗	✗	✗	✓	✓	✗	✓	✓
Move cells to closest milestone	✓	✓	✓	✗	✗	✗	✓	✓	✗	✓	✓
Length shuffling	✓	✗	✓	✗	✓	✗	✗	✗	✗	✓	✓
Cells into small subedges	✗	✓	✗	✓	✓	✓	✓	✗	✓	✓	✓
New leaf edges	✓	✓	✗	✓	✓	✓	✗	✗	✓	✓	✓
New connecting edges	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓
Changing topology and cell position	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓
Bifurcation merging	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓
Bifurcation merging and changing cell positions	✓	✓	✓	✗	✗	✗	✓	✓	✓	✓	✓
Bifurcation concatentation	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓
Cycle breaking	✓	✗	✓	✓	✓	✓	✓	✓	✗	✓	✓
Linear joining	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓
Linear splitting	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Change of topology	✓	✗	✗	✓	✓	✓	✓	✓	✗	✓	✓
Cells on milestones vs edges	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

3.6.3 Score aggregation

To rank the methods, we need to aggregate on two levels: across **datasets** and across specific/application metrics to calculate an **overall metric**.

Aggregating over datasets

When combining different datasets, it is important that the biases in the datasets does not influence the overall score. In our study, we define three such biases, although there are potentially many more:

- **Difficulty of the datasets** Some datasets are more difficult than others. This can have various reasons, such as the complexity of the topology, the amount of biological and technical noise, or the dimensions of the data. It is important that a small increase in performance on a more difficult dataset has an equal impact on the final score as a large increase in performance on easier datasets.
- **Dataset sources** It is much easier to generate synthetic datasets than real datasets, and this bias is reflected in our set of datasets. However, given their higher biological relevance, real datasets should be given at least equal importance than synthetic datasets.
- **Trajectory types** There are many more linear and disconnected real datasets, and only a limited number of tree or graph datasets. This imbalance is there because historically most datasets have been linear datasets, and because it is easy to create disconnected datasets by combining different datasets. However, this imbalance in trajectory types does not necessarily reflect the general importance of that trajectory type.

We designed an aggregation scheme which tries to prevent these biases from influencing the ranking of the methods.

The difficulty of a dataset can easily have an impact on how much weight the dataset gets in an overall ranking. We illustrate this with a simple example in Figure 3.22. One method consistently performs well on both the easy and the difficult datasets. But because the differences are small in the difficult datasets, the mean would not give this method a high score. Meanwhile, a variable method which does not perform well on the difficult dataset gets the highest score, because it scored so high on the easier dataset.

To avoid this bias, we normalise the scores of each dataset by first scaling and centering to $\mu = 0$ and $\sigma = 1$, and then moving the score values back to $[0, 1]$ by applying the

unit normal density distribution function. This results in scores which are comparable across different datasets (Figure 3.22). In contrast to other possible normalisation techniques, this will still retain some information on the relative difference between the scores, which would have been lost when using the ranks for normalisation. An example of this normalisation, which will also be used in the subsequent aggregation steps, can be seen in Figure 3.23.

After normalisation, we aggregate step by step the scores from different datasets. We first aggregate the datasets with the same dataset source and trajectory type using an arithmetic mean of their scores (Figure 3.24a). Next, the scores are averaged over different dataset sources, using a arithmetic mean which was weighted based on how much the synthetic and silver scores correlated with the real gold scores (Figure 3.24b). Finally, the scores are aggregated over the different trajectory types again using a arithmetic mean (Figure 3.24c).

Overall metrics

Undoubtedly, a single optimal overall metric does not exist for trajectories, as different users may have different priorities:

- A user may be primarily interested in defining the correct topology, and only use the cellular ordering when the topology is correct
- A user may be less interested in how the cells are ordered within a branch, but primarily in which cells are in which branches
- A user may already know the topology, and may be primarily interested in finding good features related to a particular branching point

Each of these scenarios would require a combinations of *specific* and *application* metrics with different weights. To provide an "overall" ranking of the metrics, which is impartial for the scenarios described above, we therefore chose a metric which weighs every aspect of the trajectory equally:

- Its **ordering**, using the cor_{dist}
- Its **branch assignment**, using the $F1_{branches}$
- Its **topology**, using the *HIM*
- The accuracy of **differentially expressed features**, using the $wcor_{features}$

Next, we considered three different ways of averaging different scores: the arithmetic mean, geometric mean and harmonic mean. Each of these types of mean have different use cases. The harmonic mean is most appropriate when the scores would

all have a common denominator (as is the case for the *Recovery* and *Relevance* described earlier). The arithmetic mean would be most appropriate when all the metrics have the same range. For our use case, the geometric mean is the most appropriate, because it is low if one of the values is low. For example, this means that if a method is not good at inferring the correct topology, it will get a low overall score, even if it performs better at all other scores. This ensures that a high score will only be reached if a prediction has a good ordering, branch assignment, topology, and set of differentially expressed features.

The final overall score (Figure 3.25) for a method was thus defined as:

$$Overall = mean_{geometric} = \sqrt[4]{cor_{dist} \times F1_{branches} \times HIM \times wcor_{features}}$$

We do however want to stress that different use cases will require a different overall score to order the methods. Such a context-dependent ranking of all methods is provided through the dynguidelines app (http://guidelines.dynverse.org).

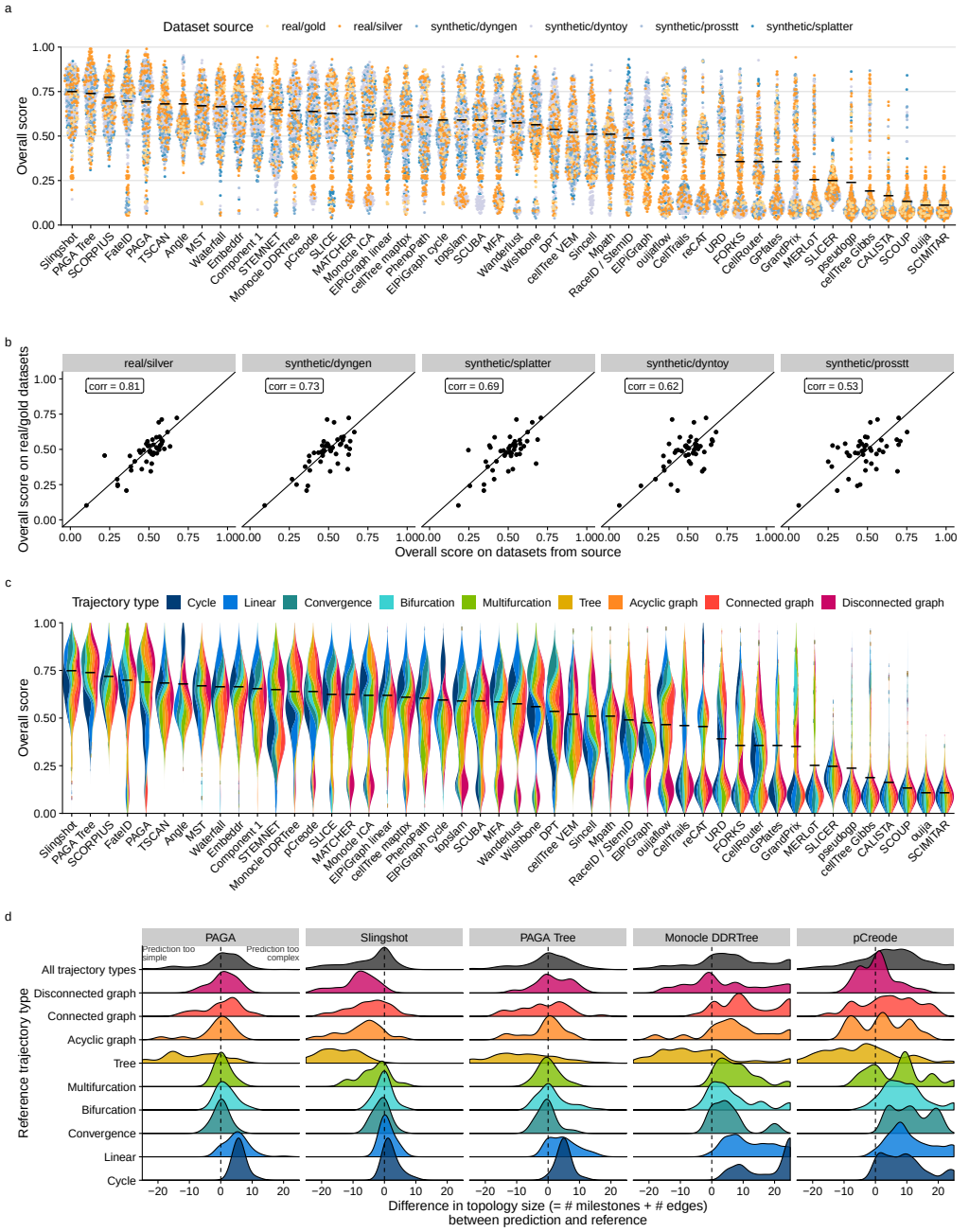


Figure 3.5: Accuracy of trajectory inference methods. **a** Overall score for all methods across 339 datasets, colored by the source of the datasets. Black line indicates the mean. **b** Similarity between the overall scores of all dataset sources, compared to real datasets with a gold standard, across all methods (n = 46, after filtering out methods that errored too frequently). Shown in the top left is the Pearson correlation. **c** Bias in the overall score towards trajectory types for all methods across 339 datasets. Black line indicates the mean. **d** Distributions of the difference in size between predicted and reference topologies. A positive difference means that the topology predicted by the method is more complex than the one in the reference.

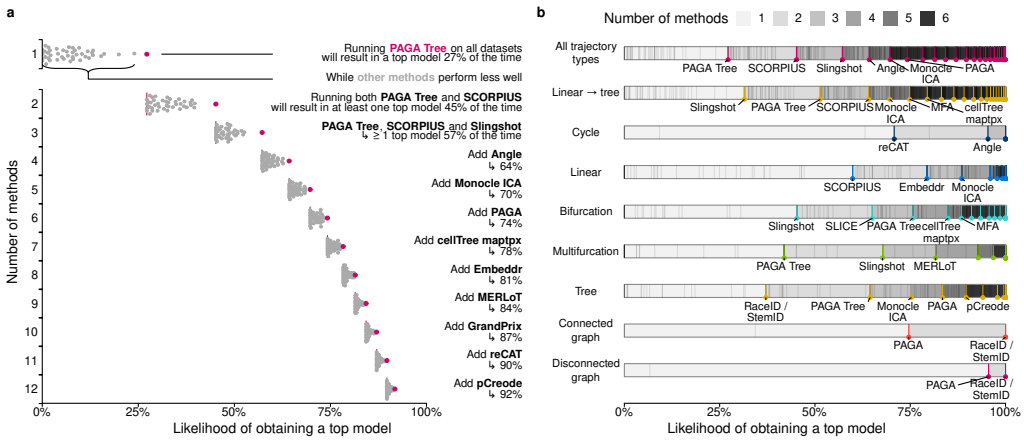


Figure 3.6: Complementarity between different trajectory inference methods. **a**, We assessed the likelihood for different combinations of methods to lead to a 'top model' (defined as a model with an overall score of at least 95% of the best model) when applied to all datasets. **b**, The likelihood for different combinations of methods to lead to a 'top model' was assessed separately on different trajectory types. For this figure, we did not include any methods requiring a cell grouping or a time course as prior information.

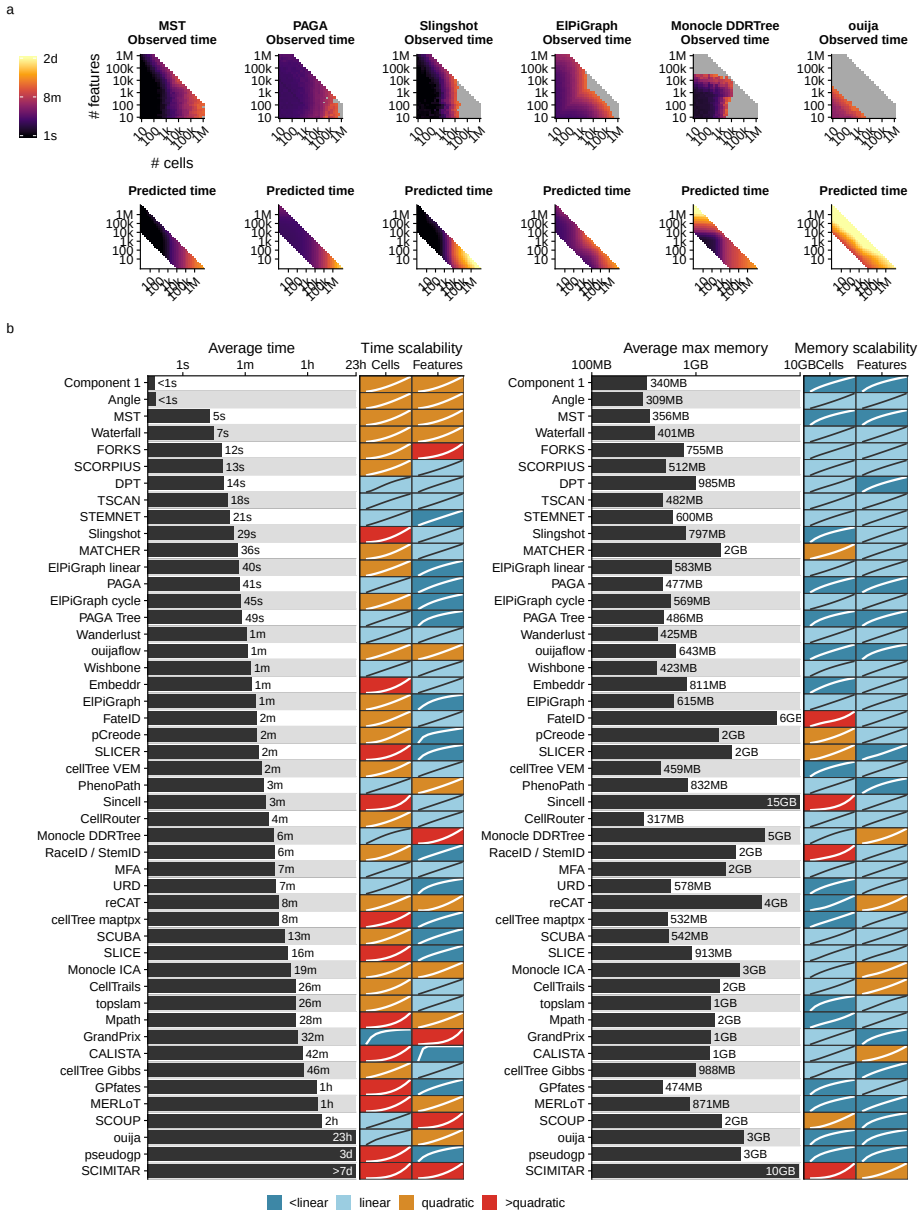


Figure 3.7: Scalability of trajectory inference methods. **a** Three examples of average observed running times across five datasets (left) and the predicted running time (right). **b** Overview of the scalability results of all methods, ordered by their average predicted running time from (a). We predicted execution times and memory usage for each method with increasing number of features or cells, and used these values to classify each method into sublinear, linear, quadratic and superquadratic based on the shape of the curve.

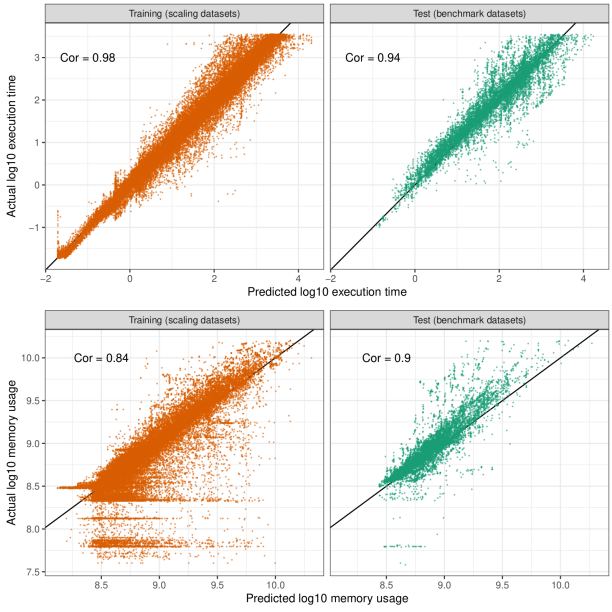


Figure 3.8: Agreement between actual values and predictions for execution times and memory usage. We created a predictive model of the running time and memory usage based on a set of scaling datasets (left), and validated this model based on the similarity of the predictions and actual values on all benchmark datasets (right). Shown are the values for each method and dataset (n = 65618 for training, n = 11939 for test). Top left indicates the Pearson correlation coefficient.

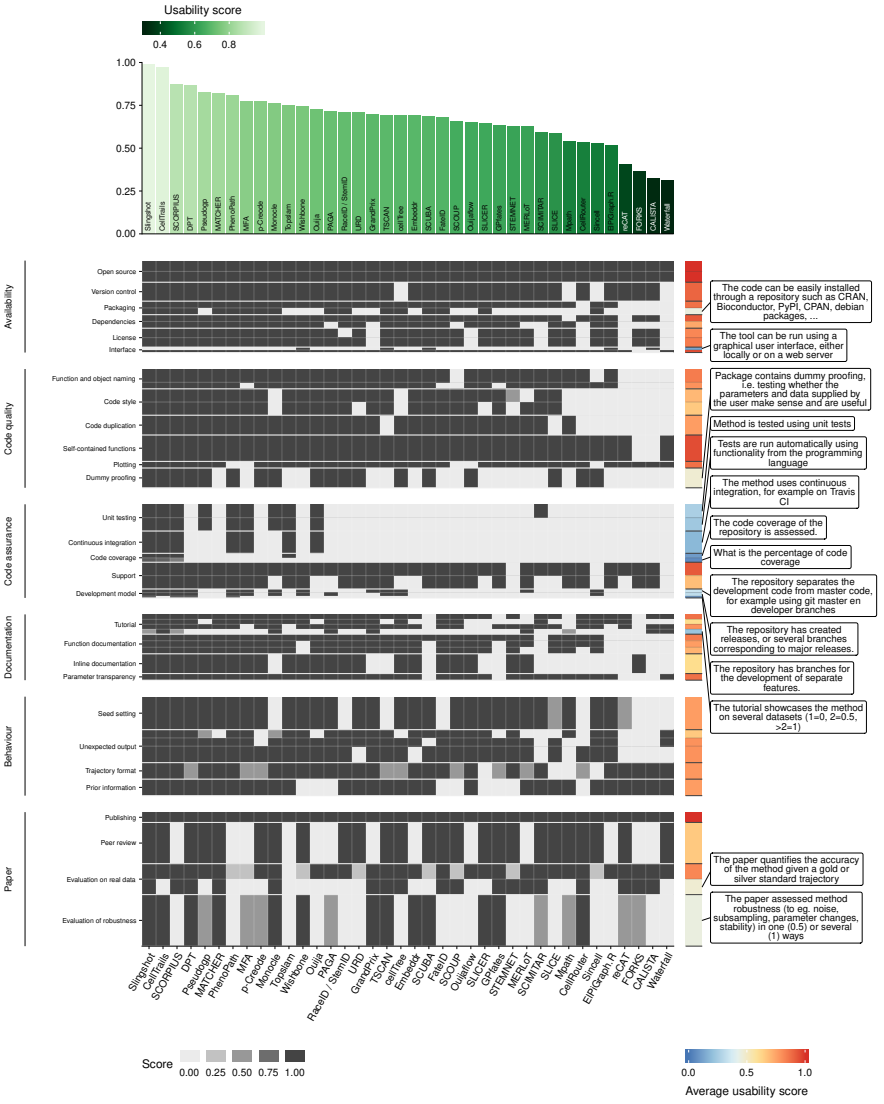


Figure 3.9: Usability of trajectory inference methods. Shown is the score given for each method on every item from the usability score sheet(Table 3.1). Each aspect of the quality control was part of a category, and each category was weighted so that it contributed equally to the final quality score. Within each category, each aspect also received a weight depending on how often it was mentioned in a set of papers discussing good practices in tool development and evaluation. This is represented in the plot as the height on the y-axis. Top: Average usability score for each method. Right: The average score of each quality control item. Shown into more detail are those items which had an average score lower than 0.5.

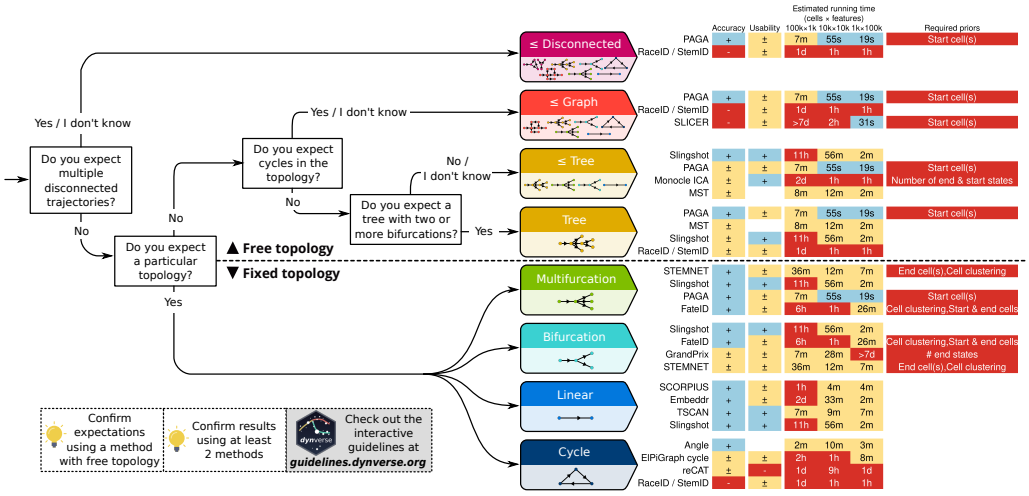


Figure 3.10: Practical guidelines for method users. As the performance of a method mostly depends on the topology of the trajectory, the choice of TI method will be primarily influenced by the user's existing knowledge about the expected topology in the data. We therefore devised a set of practical guidelines, which combines the method's performance, user friendliness and the number of assumptions a user is willing to make about the topology of the trajectory. Methods to the right are ranked according to their performance on a particular (set of) trajectory type. Further to the right are shown the accuracy (+: scaled performance $\geq$ 0.9, $\pm$: >0.6), usability scores (+:$\geq$0.9, $\pm$ $\geq$0.6), estimated running times and required prior information. k, thousands; m, millions.

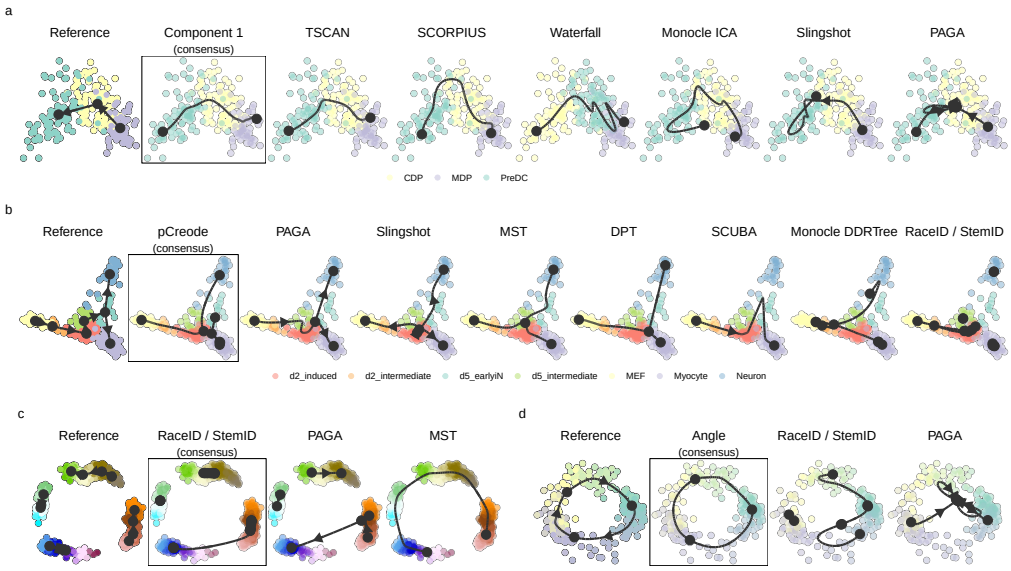
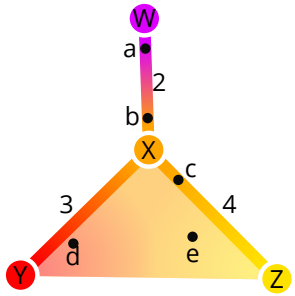


Figure 3.11: Demonstration of how a common framework for TI methods facilitates broad applicability using some example datasets. Trajectories inferred by each method were projected to a common dimensionality reduction using multidimensional scaling. For each dataset, we also calculated a 'consensus' prediction, by calculating the cordist between each pair of models and picking the model with the highest score on average. **a**, The top methods applied on a dataset containing a linear trajectory of differentiation dendritic cells, going from MDP, CDP to PreDC. **b**, The top methods applied on a dataset containing a bifurcating trajectory of reprogrammed fibroblasts. **c**, A synthetic dataset generated by dyntoy, containing four disconnected trajectories. **d**, A synthetic dataset generated by dyngen, containing a cyclic trajectory.



Milestone network

from	to	length
W	X	2
X	Y	3
X	Z	4

Regions of delayed commitment

region	to	is_begin
XYZ	X	TRUE
XYZ	Y	FALSE
XYZ	Z	FALSE

Branch assignment

Represented by the milestone percentages

cell	milestone	percentage
a	W	0.9
a	X	0.1
b	W	0.2
b	X	0.8
c	X	0.8
c	Z	0.2
d	X	0.2
d	Y	0.7
d	Z	0.1
e	X	0.3
e	Y	0.2
e	Z	0.5

Cell positions

Figure 3.12: An example trajectory that will be used throughout this section. It contains contains four milestones (W to Z) and five cells (a to e).

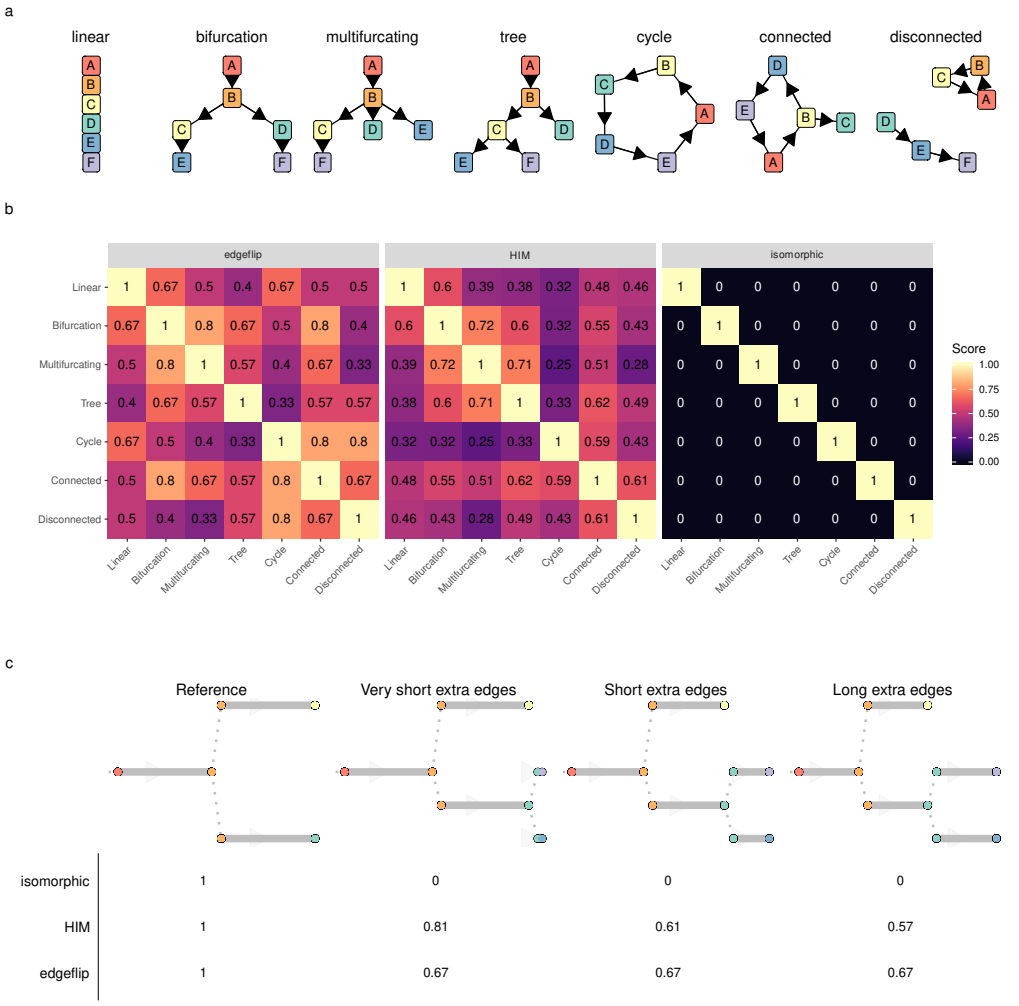


Figure 3.13: Showcase of three metrics to evaluate topologies: *isomorphic*, *edgeflip* and *HIM* **(a)** The used topologies. **(b)** The scores when comparing each pair of trajectory types. (c) Four datasets in which aan extra edge is added and made progressively longer. This shows how the HIM can take into account edge lengths.

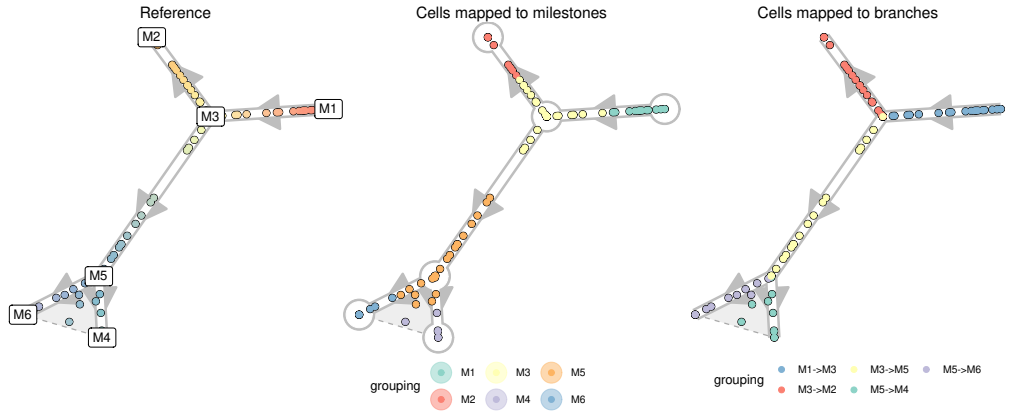
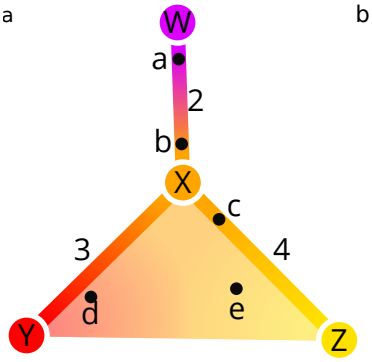


Figure 3.14: Mapping cells to their closest milestone or branch for the calculation of the *F1*$_{milestones}$ and *F1*$_{branches}$. To calculate the *F1*$_{milestones}$, cells are mapped towards the nearest milestone, i.e. the milestone with the highest milestone percentage. For the *F1*$_{branches}$, the cells are mapped to the closest edge.

a



b

Milestone network

from	to	length
W	X	2
X	Y	3
X	Z	4

Region of delayed commitment

region	to	is_begin
XYZ	X	TRUE
XYZ	Y	FALSE
XYZ	Z	FALSE

Milestone percentages

cell	milestone	percentage
a	W	0.9
a	X	0.1
b	W	0.2
b	X	0.8
c	X	0.8
c	Z	0.2
d	X	0.2
d	Y	0.7
d	Z	0.1
e	X	0.3
e	Y	0.2
e	Z	0.5

c

$d(a, b) = 2 \times (0.9 - 0.2) = 1.4$
$d(a, c) = 2 \times 0.9 + 4 \times 0.2 = 2.6$
$d(b, c) = 2 \times 0.2 + 4 \times 0.2 = 1.2$
$d(a, d) = 2 \times 0.9 + 3 \times 0.7 + 4 \times 0.1 = 4.3$
$d(b, d) = 2 \times 0.2 + 3 \times 0.7 + 4 \times 0.1 = 2.9$
$d(c, d) = 3 \times (0.7 - 0) + 4 \times (0.2 - 0.1) = 2.5$
$d(a, e) = 2 \times 0.9 + 3 \times 0.2 + 4 \times 0.5 = 4.4$
$d(b, e) = 2 \times 0.2 + 3 \times 0.2 + 4 \times 0.5 = 3.0$
$d(c, e) = 3 \times (0.2 - 0) + 4 \times (0.5 - 0.2) = 1.8$
$d(d, e) = 3 \times (0.7 - 0.2) + 4 \times (0.5 - 0.1) = 3.1$

d

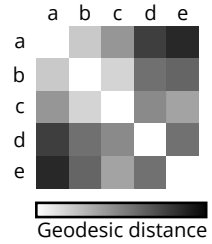


Figure 3.15: The calculation of geodesic distances on a small example trajectory. a) A toy example containing four milestones (W to Z) and five cells (a to e). **b)** The corresponding milestone network, milestone percentages and regions of delayed commitment, when the toy trajectory is converted to the common trajectory model. **c)** The calculations made for calculating the pairwise geodesic distances. **d)** A heatmap representation of the pairwise geodesic distances.

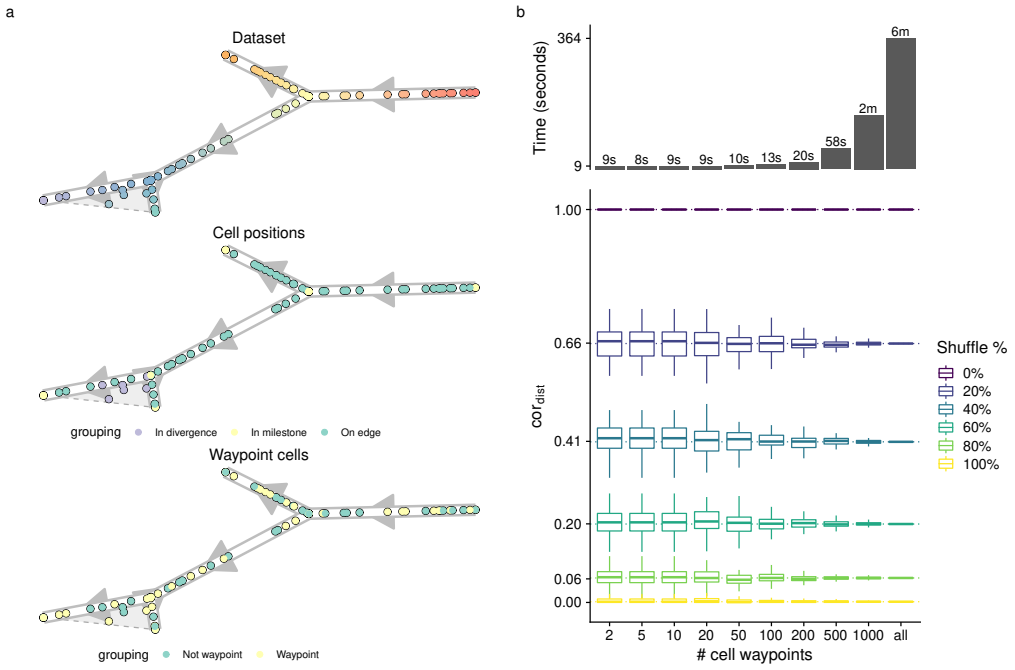


Figure 3.16: Determination of cell waypoints a) Illustration of the stratified cell sampling using an example dataset (top). Each milestone, edge between two milestones and region of delayed commitment is seen as a collection of cells (middle), and the number of waypoints (100 in this case) are divided over each of these collection of cells (bottom). **b)** Accuracy versus time to calculate cor_{dist}. Shown are distributions over 100 random waypoint samples. The upper whisker of the boxplot extends from the hinge (75% percentile) to the largest value, no further than 1.5× the IQR of the hinge. The lower whisker extends from the hinge (25% percentile) to the smallest value, at most 1.5× the IQR of the hinge.

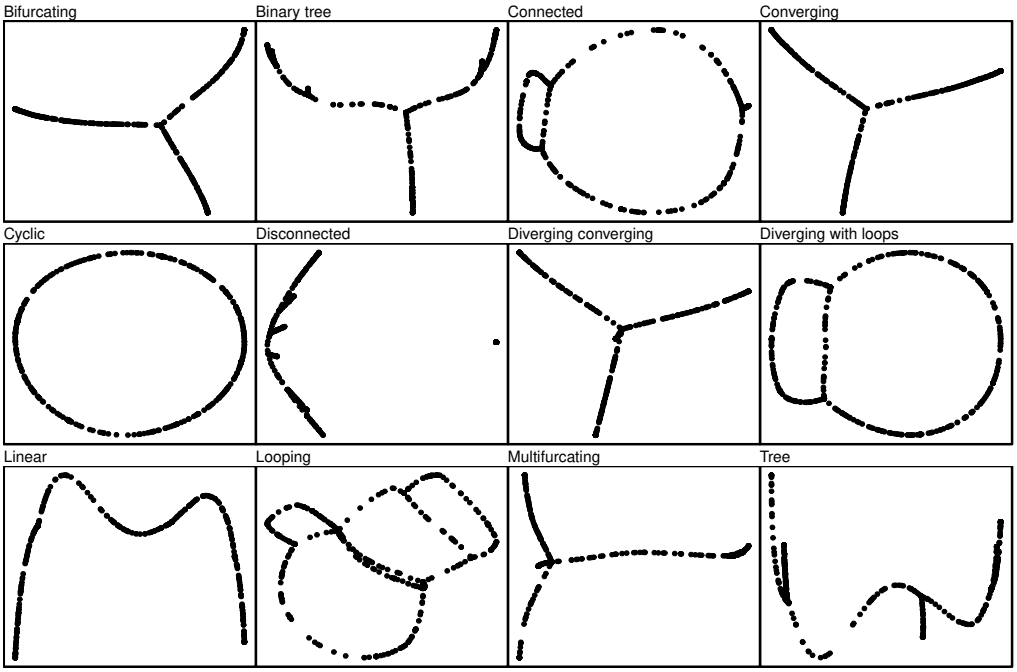


Figure 3.17: Determination of cell waypoints. We generated different toy trajectory datasets with varying topologies and calculated the geodesic distances between all cells within the trajectory. We then used these distances as input for classical multidimensional scaling. This shows that the geodesic distances do not only contain information regarding the cell's positions, but also information on the lengths and wiring of the topology.

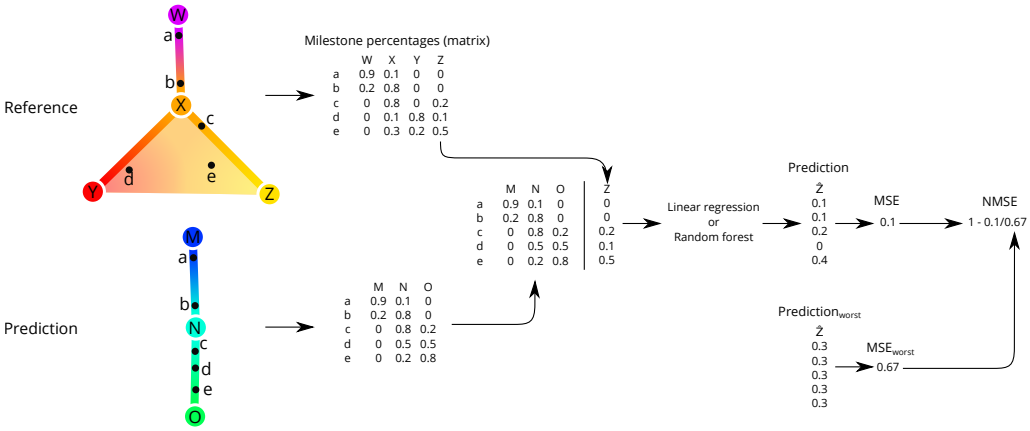


Figure 3.18: The calculation of $NMSE_{lm}$ distances on a small example trajectory. The milestone percentages of the reference are predicted based on the milestone percentages of the prediction, using regression models such as linear regression or random forests. The predicted trajectory is then scored by comparing the mean-squared error (MSE) of this regression model with the baseline MSE where the prediction is the average milestone percentage.

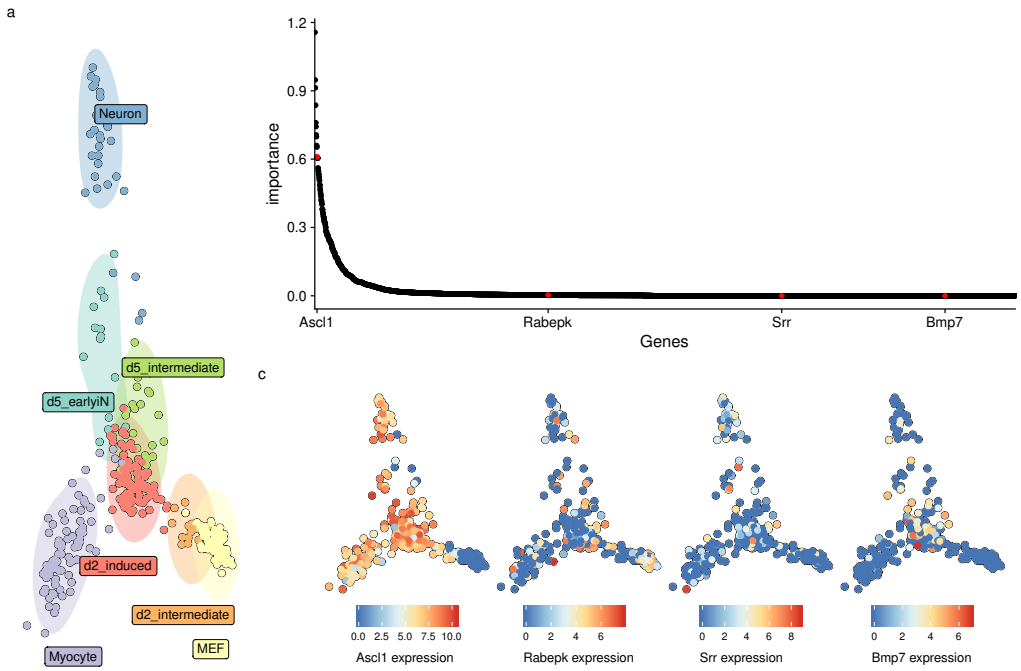


Figure 3.19: An illustration of ranking features based on their importance in a trajectory. (a) A MDS dimensionality reduction of a real dataset in which mouse embryonic fibroblasts (MEF) differentiate into Neurons and Myocytes. (b) The ranking of feature importances from high to low. The majority of features have a very low importance. (c) Some examples, which were also highlighted in b. Higher features in the ranking are clearly specific to certain parts of the trajectory, while features lower on the ranking have a more dispersed expression pattern.

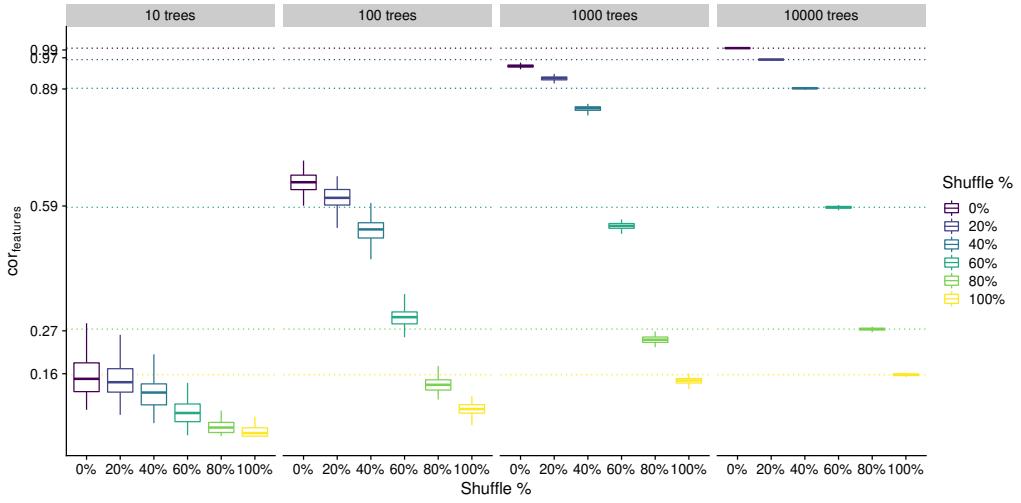


Figure 3.20: Effect of the number of trees parameter on the accuracy and variability of the $cor_{features}$. We used the dataset from Figure 3.19 and calculated the $cor_{features}$ after shuffling a percentage of cells.

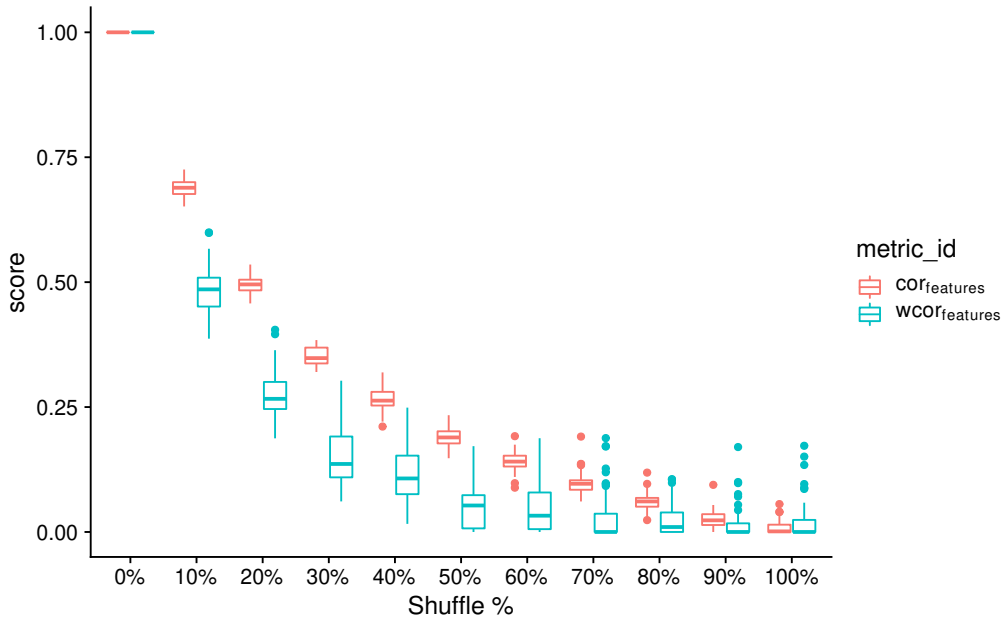


Figure 3.21: Effect of weighting the features based on their feature importance in the reference. We used the same dataset as in Figure 3.19, and calculated the $cor_{features}$ after shuffling a percentage of cells.

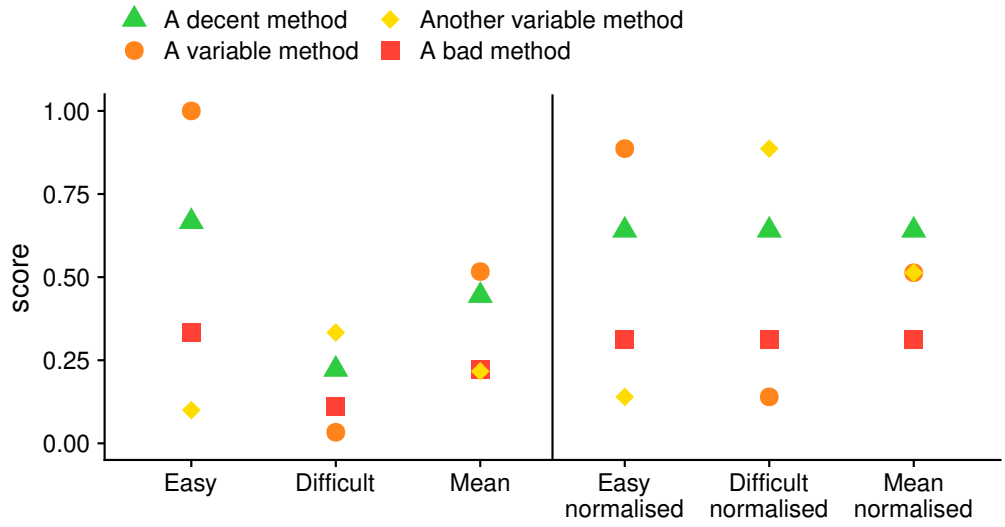


Figure 3.22: An illustration of how the difficulty of a dataset can influence the overall ranking. A decent method, which consistently ranks high on an easy and difficult dataset, does not get a high score when averaging. On the other hand, a method which ranks high on the easy dataset, but very low on the difficult dataset does get a high score on average. After normalising the scores (right), this problem disappears.

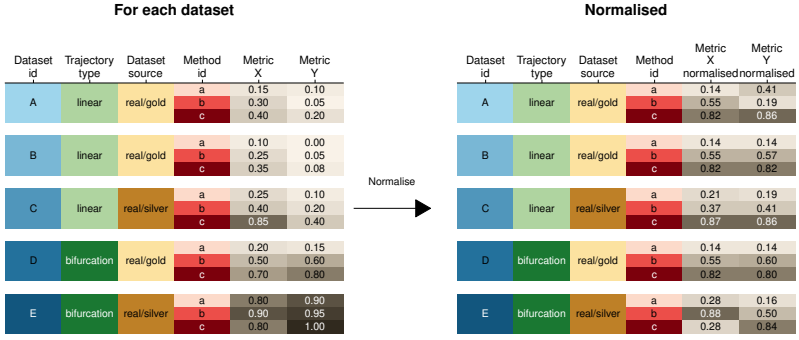


Figure 3.23: An example of the normalisation procedure. Shown are some results of a benchmarking procedure, where every row contains the scores of a particular method (red shading) on a particular dataset (blue shading), with a trajectory type (green shading) and dataset source (orange shading).

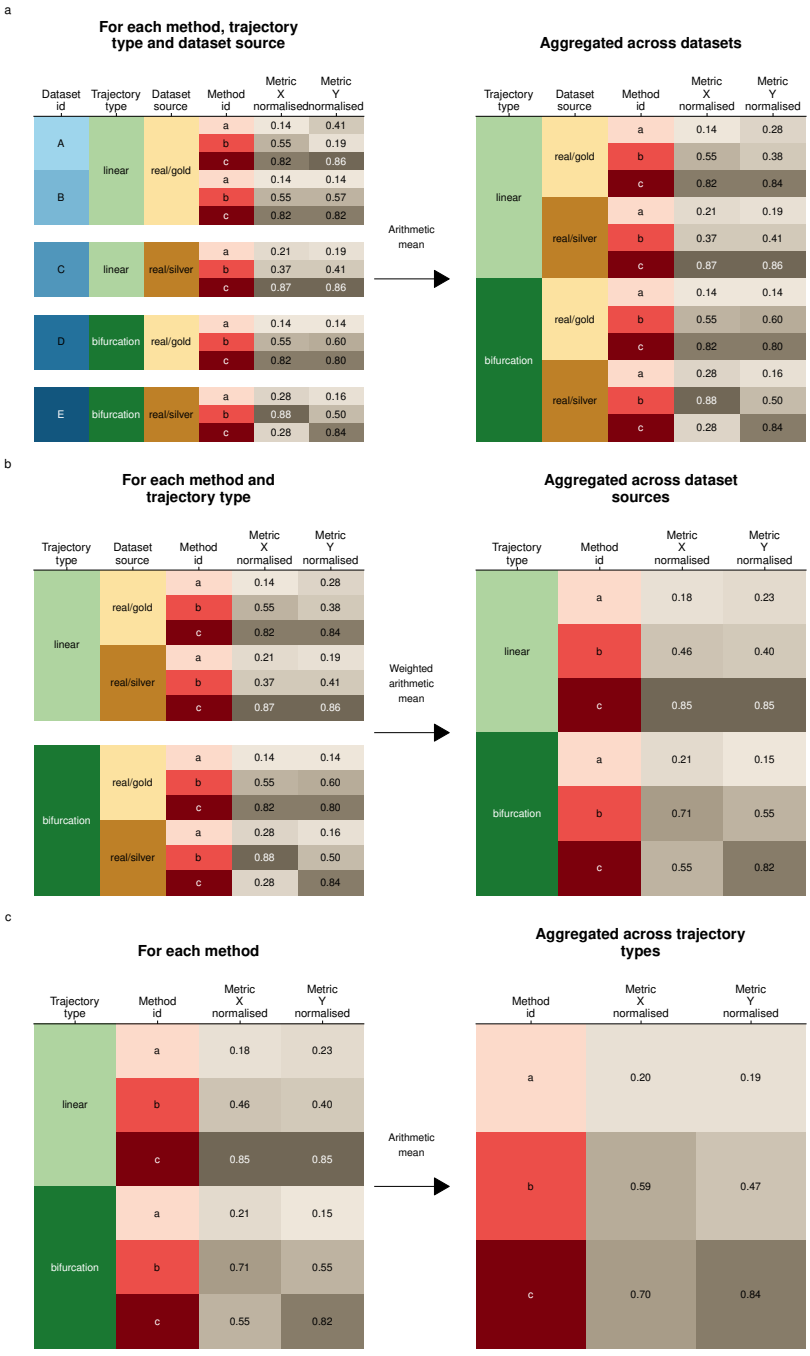


Figure 3.24: An example of the aggregation procedure. In consecutive steps we aggregated across **(a)** different datasets with the same source and trajectory type, **(b)** different dataset sources with the same trajectory type (weighted for the correlation of the dataset source with the real gold dataset source) and **(c)** all trajectory types.

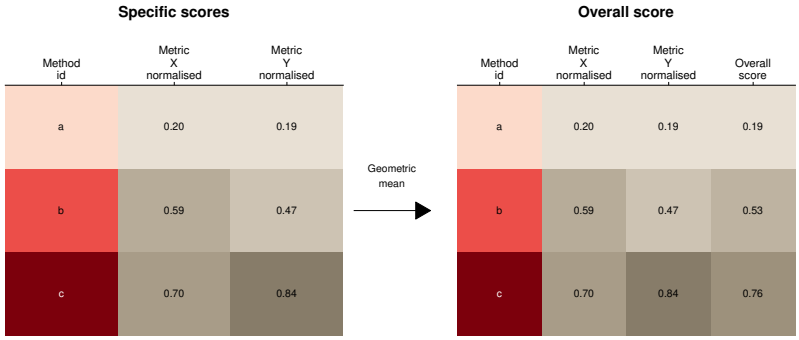


Figure 3.25: An example of the averaging procedure. For each method, we calculated the geometric mean between its normalised and aggregated scores

3.7 References

[1] Amos Tanay and Aviv Regev. "Scaling Single-Cell Genomics from Phenomenology to Mechanism". In: *Nature* 541.7637 (Jan. 2017), nature21350. ISSN: 1476-4687. DOI: 10.1038/nature21350.

[2] Martin Etzrodt, Max Endele, and Timm Schroeder. "Quantitative Single-Cell Approaches to Stem Cell Research". In: *Cell Stem Cell* 15.5 (2014), pp. 546–558.

[3] Cole Trapnell. "Defining Cell Types and States with Single-Cell Genomics". In: *Genome Research* 25.10 (2015), pp. 1491–1498. ISSN: 15495469. DOI: 10.1101/gr.190595.115. pmid: 26430159.

[4] Robrecht Cannoodt, Wouter Saelens, and Yvan Saeys. "Computational Methods for Trajectory Inference from Single-Cell Transcriptomics". In: *European Journal of Immunology* 46.11 (Nov. 1, 2016), pp. 2496–2506. ISSN: 1521-4141. DOI: 10.1002/eji.201646347.

[5] Kevin R Moon et al. "Manifold Learning-Based Methods for Analyzing Single-Cell RNA-Sequencing Data". In: *Current Opinion in Systems Biology*. $\backslash$textbullet{} Future of Systems Biology$\backslash$textbullet{} Genomics and Epigenomics 7 (Feb. 2018), pp. 36–46. ISSN: 2452-3100. DOI: 10.1016/j.coisb.2017.12.008.

[6] Zehua Liu et al. "Reconstructing Cell Cycle Pseudo Time-Series via Single-Cell Transcriptome Data". In: *Nature Communications* 8.1 (June 2017), p. 22. ISSN: 2041-1723. DOI: 10.1038/s41467-017-00039-z.

[7] F Alexander Wolf et al. "Graph Abstraction Reconciles Clustering with Trajectory Inference through a Topology Preserving Map of Single Cells". In: *bioRxiv* (Oct. 2017), p. 208819. DOI: 10.1101/208819.

[8] Andreas Schlitzer et al. "Identification of cDC1- and cDC2-Committed DC Progenitors Reveals Early Lineage Priming at the Common DC Progenitor Stage in the Bone Marrow". In: *Nature Immunology* 16.7 (July 2015), pp. 718–728. ISSN: 1529-2916. DOI: 10.1038/ni.3200.

[9] Lars Velten et al. "Human Haematopoietic Stem Cell Lineage Commitment Is a Continuous Process". In: *Nature Cell Biology* 19.4 (Apr. 2017), pp. 271–281. ISSN: 1476-4679. DOI: 10.1038/ncb3493.

[10] Peter See et al. "Mapping the Human DC Lineage through the Integration of High-Dimensional Techniques". In: *Science* 356.6342 (June 2017), eaag3009. ISSN: 0036-8075, 1095-9203. DOI: 10.1126/science.aag3009. pmid: 28473638.

[11] Sara Aibar et al. "SCENIC: Single-Cell Regulatory Network Inference and Clustering". In: *Nature Methods* (Oct. 2017). ISSN: 1548-7091. DOI: 10.1038/nmeth.4463.

[12] Aviv Regev et al. "The Human Cell Atlas". In: *eLife* 6 (Dec. 2017). ISSN: 2050084X. DOI: 10.7554/eLife.27041.

[13] Xiaoping Han et al. "Mapping the Mouse Cell Atlas by Microwell-Seq". In: *Cell* 172.5 (Feb. 2018), 1091–1107.e17. ISSN: 1097-4172. DOI: 10.1016/j.cell.2018.02.001. pmid: 29474909.

[14] Nicholas Schaum et al. "Single-Cell Transcriptomics of 20 Mouse Organs Creates a Tabula Muris". In: *Nature* 562.7727 (Oct. 2018), pp. 367–372. ISSN: 1476-4687. DOI: 10.1038/s41586-018-0590-4.

[15] Philipp Angerer et al. "Single Cells Make Big Data: New Challenges and Opportunities in Transcriptomics". In: *Current Opinion in Systems Biology*. Big Data Acquisition and Analysis $\backslash$textbullet{} Pharmacology and Drug Discovery 4 (Aug. 2017), pp. 85–91. ISSN: 2452-3100. DOI: 10.1016/j.coisb.2017.07.004.

[16] Vincent J Henry et al. "OMICtools: An Informative Directory for Multi-Omic Data Analysis". In: *Database: The Journal of Biological Databases and Curation* 2014 (July 2014). ISSN: 1758-0463. DOI: 10.1093/database/bau069. pmid: 25024350.

[17] Sean Davis et al. *Awesome Single Cell*. June 2018.

[18] Luke Zappia, Belinda Phipson, and Alicia Oshlack. "Exploring the Single-Cell RNA-Seq Analysis Landscape with the scRNA-Tools Database". In: *bioRxiv* (Oct. 2017), p. 206573. DOI: 10.1101/206573.

[19] Sean C. Bendall et al. "Single-Cell Trajectory Detection Uncovers Progression and Regulatory Coordination in Human B Cell Development". In: *Cell* 157.3 (2014), pp. 714–725. ISSN: 00928674. DOI: 10.1016/j.cell.2014.04.005.

[20] Jaehoon Shin et al. "Single-Cell RNA-Seq with Waterfall Reveals Molecular Cascades Underlying Adult Neurogenesis". In: *Cell Stem Cell* 17.3 (Sept. 3, 2015), pp. 360–372. ISSN: 1875-9777. DOI: 10.1016/j.stem.2015.07.013. pmid: 26299571.

[21] Kieran Campbell and Christopher Yau. "Bayesian Gaussian Process Latent Variable Models for Pseudotime Inference in Single-Cell RNA-Seq Data". In: *bioRxiv* (Sept. 2015), p. 26872. DOI: 10.1101/026872.

[22] Laleh Haghverdi et al. "Diffusion Pseudotime Robustly Reconstructs Lineage Branching". In: *Nature Methods* 13.10 (Oct. 2016), pp. 845–848. ISSN: 1548-7105. DOI: 10.1038/nmeth.3971.

[23] Manu Setty et al. "Wishbone Identifies Bifurcating Developmental Trajectories from Single-Cell Data". In: *Nat. Biotechnol.* 34 (April June 2016), pp. 1–14. ISSN: 1087-0156. DOI: 10.1038/nbt.3569. pmid: 27136076.

[24] 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 (Mar. 2014), pp. 381–386. ISSN: 1546-1696. DOI: 10.1038/nbt.2859. pmid: 24658644.

[25] Hirotaka Matsumoto, Matsumoto Hirotaka, and Kiryu Hisanori. "SCOUP: A Probabilistic Model Based on the Ornstein-Uhlenbeck Process to Analyze Single-Cell Expression Data during Differentiation". In: *BMC Bioinformatics* 17.1 (2016).

[26] Xiaojie Qiu et al. "Reversed Graph Embedding Resolves Complex Single-Cell Trajectories". In: *Nature Methods* 14.10 (Oct. 2017), pp. 979–982. ISSN: 1548-7105. DOI: 10.1038/nmeth.4402.

[27] Kelly Street et al. "Slingshot: Cell Lineage and Pseudotime Inference for Single-Cell Transcriptomics". In: *BMC Genomics* 19.1 (June 2018), p. 477. ISSN: 1471-2164. DOI: 10.1186/s12864-018-4772-0.

[28] Zhicheng Ji and Hongkai Ji. "{TSCAN}: Pseudo-Time Reconstruction and Evaluation in Single-Cell {RNA-Seq} Analysis". In: *Nucleic Acids Res.* (2016).

[29] Joshua D. Welch, Alexander J. Hartemink, and Jan F. Prins. "SLICER: Inferring Branched, Nonlinear Cellular Trajectories from Single Cell RNA-Seq Data". In: *Genome Biology* 17 (2016), p. 106. ISSN: 1474-760X. DOI: 10.1186/s13059-016-0975-3.

[30] David A DuVerle et al. "CellTree: An R/Bioconductor Package to Infer the Hierarchical Structure of Cell Populations from Single-Cell RNA-Seq Data". In: *BMC Bioinformatics* 17 (Sept. 2016), p. 363. ISSN: 1471-2105. DOI: 10.1186/s12859-016-1175-6.

[31] 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 (Mar. 2017), eaal2192. ISSN: 2470-9468. DOI: 10.1126/sciimmunol.aal2192. pmid: 28345074.

[32] Kieran R Campbell and Christopher Yau. "Probabilistic Modeling of Bifurcations in Single-Cell Gene Expression Data Using a Bayesian Mixture of Factor Analyzers". In: *Wellcome Open Research* 2 (Mar. 2017), p. 19. ISSN: 2398-502X. DOI: 10.12688/wellcomeopenres.11087.1.

[33] Luyi Tian et al. "scRNA-Seq Mixology: Towards Better Benchmarking of Single Cell RNA-Seq Protocols and Analysis Methods". In: *bioRxiv* (Oct. 2018), p. 433102. DOI: 10.1101/433102.

[34] Thomas Schaffter, Daniel Marbach, and Dario Floreano. "GeneNetWeaver: In Silico Benchmark Generation and Performance Profiling of Network Inference Methods.". In: *Bioinformatics* 27.16 (Aug. 2011), pp. 2263–2270. ISSN: 1367-4811. DOI: 10.1093/bioinformatics/btr373. pmid: 21697125.

[35] Luke Zappia, Belinda Phipson, and Alicia Oshlack. "Splatter: Simulation of Single-Cell RNA Sequencing Data". In: *Genome Biology* 18 (Sept. 2017), p. 174. ISSN: 1474-760X. DOI: 10.1186/s13059-017-1305-0.

[36] Valentine Svensson, Roser Vento-Tormo, and Sarah A Teichmann. "Exponential Scaling of Single-Cell RNA-Seq in the Past Decade". In: *Nature Protocols* 13.4 (Apr. 2018), pp. 599–604. ISSN: 1750-2799. DOI: 10.1038/nprot.2017.149.

[37] Junyue Cao et al. "Joint Profiling of Chromatin Accessibility and Gene Expression in Thousands of Single Cells". In: *Science* (Aug. 2018), eaau0730. ISSN: 0036-8075, 1095-9203. DOI: 10.1126/science.aau0730. pmid: 30166440.

[38] Natalya Pya and Simon N Wood. "Shape Constrained Additive Models". In: *Statistics and Computing* 25.3 (May 2015), pp. 543–559. ISSN: 1573-1375. DOI: 10.1007/s11222-013-9448-7.

[39] Morgan Taschuk and Greg Wilson. "Ten Simple Rules for Making Research Software More Robust". In: *PLOS Computational Biology* 13.4 (Apr. 2017), e1005412. ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1005412.

[40] Serghei Mangul et al. "A Comprehensive Analysis of the Usability and Archival Stability of Omics Computational Tools and Resources". In: *bioRxiv* (Oct. 2018), p. 452532. DOI: 10.1101/452532.

[41] Greg Wilson et al. "Best Practices for Scientific Computing". In: *PLOS Biology* 12.1 (Jan. 2014), e1001745. ISSN: 1545-7885. DOI: 10.1371/journal.pbio.1001745.

[42] Haydee Artaza et al. "Top 10 Metrics for Life Science Software Good Practices". In: *F1000Research* 5 (Aug. 2016), p. 2000. ISSN: 2046-1402. DOI: 10.12688/f1000research.9206.1.

[43] Jeff Lee. *Rpackages: R Package Development - the Leek Group Way!* Dec. 2017. URL: https://github.com/jtleek/rpackages.

[44] Hadley Wickham. *R Packages: Organize, Test, Document, and Share Your Code*. O'Reilly Media, Inc., Mar. 26, 2015. 201 pp. ISBN: 978-1-4919-1056-6. Google Books: DqSxBwAAQBAJ.

[45] Luis Bastiao Silva et al. "General Guidelines for Biomedical Software Development". In: *F1000Research* 6 (July 2017). ISSN: 2046-1402. DOI: 10.12688/f1000research.10750.2. pmid: 28443186.

[46] Rafael C Jiménez et al. "Four Simple Recommendations to Encourage Best Practices in Research Software". In: *F1000Research* 6 (June 2017). ISSN: 2046-1402. DOI: 10.12688/f1000research.11407.1. pmid: 28751965.

[47] Mehran Karimzadeh and Michael M. Hoffman. "Top Considerations for Creating Bioinformatics Software Documentation". In: *Briefings in Bioinformatics* 19.4 (July 20, 2018), pp. 693–699. ISSN: 1467-5463. DOI: 10.1093/bib/bbw134.

[48] Alex Anderson. *Writing Great Scientific Code*. Oct. 2016. URL: http://alexanderganderson.github.io/code/2016/10/12/coding_tips.html.

[49] Brett K Beaulieu-Jones and Casey S Greene. "Reproducibility of Computational Workflows Is Automated Using Continuous Analysis". In: *Nature Biotechnology* 35.4 (Mar. 2017), nbt.3780. ISSN: 1546-1696. DOI: 10.1038/nbt.3780.

[50] Vincent Driessen. *A Successful Git Branching Model*. Jan. 2010. URL: http://nvie.com/posts/a-successful-git-branching-model/.

[51] Anne-Laure Boulesteix. "Ten Simple Rules for Reducing Overoptimistic Reporting in Methodological Computational Research". In: *PLOS Computational Biology* 11.4 (Apr. 2015), e1004191. ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1004191.

[52] Jean Francois Puget. *Green Dice Are Loaded (Welcome to p-Hacking)*. Mar. 2016. URL: https://www.ibm.com/developerworks/community/blogs/jfp/entry/Green_dice_are_loaded_welcome_to_p_hacking.

[53] Frank Gannon. "The Essential Role of Peer Review". In: *EMBO Reports* 2.9 (Sept. 2001), p. 743. ISSN: 1469-221X. DOI: 10.1093/embo-reports/kve188. pmid: 11559578.

[54] Melinda Baldwin. "In Referees We Trust?". In: *Physics Today* 70.2 (Feb. 2017), pp. 44–49. ISSN: 0031-9228. DOI: 10.1063/PT.3.3463.

[55] Mohamed Radhouene Aniba, Olivier Poch, and Julie D Thompson. "Issues in Bioinformatics Benchmarking: The Case Study of Multiple Sequence Alignment". In: *Nucleic Acids Research* 38.21 (Nov. 2010), pp. 7353–7363. ISSN: 0305-1048. DOI: 10.1093/nar/gkq625. pmid: 20639539.

[56] Monika Jelizarow et al. "Over-Optimism in Bioinformatics: An Illustration". In: *Bioinformatics* 26.16 (Aug. 2010), pp. 1990–1998. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btq323.

[57] Wouter Saelens, Robrecht Cannoodt, and Yvan Saeys. "A Comprehensive Evaluation of Module Detection Methods for Gene Expression Data". In: *Nature Communications* 9.1 (Mar. 2018), p. 1090. ISSN: 2041-1723. DOI: 10.1038/s41467-018-03424-4.

[58] Gioele La Manno et al. "RNA Velocity of Single Cells". In: *Nature* 560.7719 (Aug. 2018), pp. 494–498. ISSN: 1476-4687. DOI: 10.1038/s41586-018-0414-6.

[59] Raquel Norel, John Jeremy Rice, and Gustavo Stolovitzky. "The Self-Assessment Trap: Can We All Be Better than Average?". In: *Molecular systems biology* 7.1 (2011), p. 537. ISSN: 1744-4292. DOI: 10.1038/msb.2011.70. pmid: 21988833.

[60] Anthony Gitter. *Single-Cell RNA-Seq Pseudotime Estimation Algorithm*. June 2018.

[61] Tsukasa Kouno et al. "Temporal Dynamics and Transcriptional Control Using Single-Cell Gene Expression Analysis". In: *Genome Biol.* 14.10 (2013), R118.

[62] Chun Zeng et al. "Pseudotemporal Ordering of Single Cells Reveals Metabolic Control of Postnatal β Cell Proliferation". In: *Cell Metabolism* 25.5 (May 2017), 1160–1175.e11. ISSN: 15504131. DOI: 10.1016/j.cmet.2017.04.014.

[63] Nikolaos Papadopoulos, Rodrigo Gonzalo Parra, and Johannes Soeding. "PROSSTT: Probabilistic Simulation of Single-Cell RNA-Seq Data for Complex Differentiation Processes". In: *bioRxiv* (Jan. 2018), p. 256941. DOI: 10.1101/256941.

[64] Daniel Marbach et al. "Wisdom of Crowds for Robust Gene Network Inference". In: *Nature methods* 9.8 (July 2012), pp. 796–804. ISSN: 1548-7091. DOI: 10.1038/nmeth.2016. pmid: 22796662.

[65] Aaron T L Lun, Davis J McCarthy, and John C Marioni. "A Step-by-Step Workflow for Low-Level Analysis of Single-Cell RNA-Seq Data with Bioconductor". In: *F1000Research* 5 (Oct. 2016), p. 2122. ISSN: 2046-1402. DOI: 10.12688/f1000research.9501.2.

[66] G Jurman et al. "The HIM Glocal Metric and Kernel for Network Comparison and Classification". In: *2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*. Oct. 2015, pp. 1–10. DOI: 10.1109/DSAA.2015.7344816.

[67] Marvin N Wright and Andreas Ziegler. "Ranger: A Fast Implementation of Random Forests for High Dimensional Data in C++ and R". In: *Journal of Statistical Software* 77.1 (Mar. 2017). DOI: 10.18637/jss.v077.i01.

[68] T Junttila and P Kaski. "Engineering an Efficient Canonical Labeling Tool for Large and Sparse Graphs". In: *2007 Proceedings of the Ninth Workshop on Algorithm Engineering and Experiments (ALENEX)*. Proceedings. Society for Industrial and Applied Mathematics, Jan. 2007, pp. 135–149. DOI: 10.1137/1.9781611972870.13.

[69] Laura Bahiense et al. "The Maximum Common Edge Subgraph Problem: A Polyhedral Investigation". In: *Discrete Applied Mathematics*. V Latin American Algorithms, Graphs, and Optimization Symposium $\backslash$textemdash{} Gramado, Brazil, 2009 160.18 (Dec. 2012), pp. 2523–2541. ISSN: 0166-218X. DOI: 10.1016/j.dam.2012.01.026.

[70] Edward R Dougherty. "Validation of Gene Regulatory Networks: Scientific and Inferential". In: *Briefings in Bioinformatics* 12.3 (May 2011), pp. 245–252. ISSN: 1477-4054. DOI: 10.1093/bib/bbq078. pmid: 21183477.

[71] Mads Ipsen and Alexander S Mikhailov. "Evolutionary Reconstruction of Networks". In: *Physical Review. E, Statistical, Nonlinear, and Soft Matter Physics* 66 (4 Pt 2 Oct. 2002), p. 46109. ISSN: 1539-3755. DOI: 10.1103/PhysRevE.66.046109. pmid: 12443261.

4 | SCORPIUS: Fast, accurate, and robust single-cell pseudotime

Abstract

Purpose: Recent advances in single-cell RNA-sequencing allow modelling progression of single cells along a dynamic process of interest as a trajectory. While already more than 75 trajectory inference methods have been developed, comprehensive benchmarking demonstrated that pseudotemporal ordering cells robustly is still a challenging task.
Results: SCORPIUS is a trajectory inference method specialised in pseudotemporally ordering single-cell profiles. Applying SCORPIUS to a large collection of datasets shows that it consistently produces more accurate and robust trajectories in comparison to state-of-the-art trajectory inference methods. We used SCORPIUS to generate novel hypotheses regarding dendritic cell development, which were subsequently validated *in vivo*.
Conclusion: By providing a robust and accurate toolkit for inferring linear trajectories, we aim to make pseudotime analysis as easy-to-use and prevalent as other types of analysis such as differential expression and gene set enrichment.

Publication status

Published in bioRxiv 079509. doi:10.1101/079509.
Revised manuscript in preparation.
Cannoodt R, Saelens W, Sichien D, Tavernier S, Janssens S, Guilliams M, Lambrecht B, De Preter K, and Saeys Y.

Author contributions

- **R.C.** and Y.S. designed the study.
- **R.C.** and W.S. benchmarks and analysed the data.
- S.T. and D.S. generated experimental data.
- **R.C.** implemented the SCORPIUS software package.
- **R.C.** wrote the original manuscript.
- **R.C.**, S.J., M.G., B.L., K.D.P., and Y.S. reviewed and edited the manuscript.
- K.D.P. and Y.S. supervised the project.

4.1 Introduction

Technological advancements in single-cell omics allow studying a dynamic process in a high-throughput manner. This raises concerns regarding biological fundamentals, such as how to define cell types or transitions between them [1, 2]. Trajectory inference (TI) methods aim to give insight into a dynamic process by inferring a trajectory from omics profiles of cells in which the dynamic process takes place [3].

In linear TI, also sometimes called pseudotemporal ordering, the user assumes that the dynamic process of interest is linear and is interested in how gene expression changes along the dynamic process. Linear TI is a special case of generalised TI which should be easier to tackle since topology is fixed. However, a recent benchmarking study showed that even linear TI is a non-trivial task [4], with most TI methods not capable of producing accurate models for many linear datasets.

In this work, we explain the workings of SCORPIUS, a toolbox specialised in inferring and interpreting linear trajectories. We show that SCORPIUS obtains higher accuracy scores on linear datasets in comparison to state-of-the-art TI methods. Finally, we demonstrate its usage by extracting novel findings from an existing single-cell omics dataset containing developing dendritic cells [5].

4.2 Results

In essence, SCORPIUS reduces the dimensionality of the dataset using Multi-Dimensional Scaling (MDS) [6], and derives a smooth curve that goes through the middle of the dataset using principal curves [7] (Figure 4.1A). However, both MDS and principal curves scale poorly with respect to the number of cells in the dataset, so these were adapted to scale linearly instead (See Methods). In addition, SCORPIUS produces a heatmap of the genes which are strongly up- or downregulated in function of the pseudotemporal ordering (Figure 4.1B). The genes are prioritised using the Random Forest feature importance score [8]. By clustering the genes into sets of coexpressed genes, the user can more easily reason about the functional aspect of the different gene modules.

Examples of other (linear and non-linear) TI methods illustrate common sources of low-accuracy predictions in linear TI (Figure 4.1C), namely the inference of false positive branches or incorrect pseudotemporal orderings.

Wouter, I cannot imagine a better partner to do a PhD with. If we were genes, we'd be reciprocal activators, constantly stimulating each other to do just a little bit more!

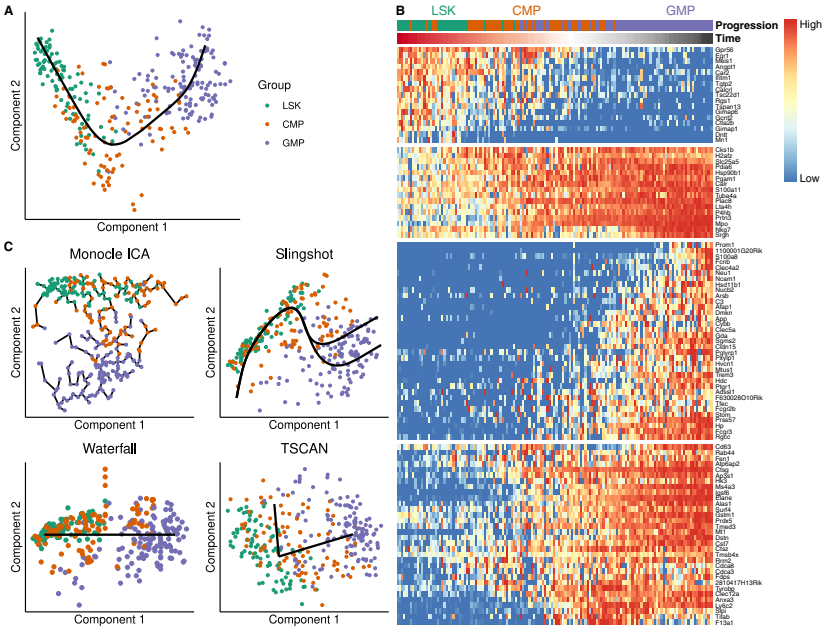


Figure 4.1: **A:** SCORPIUS derives a smooth curve that passes through the middle of the dataset. **B:** Prioritising genes in function of the pseudotemporal ordering allows easier interpretation of the dynamic process at hand. **C:** Low accuracy predictions are a result of false positive branches or incorrect pseudotemporally orderings.

4.2.1 SCORPIUS outperforms existing TI tools in inferring linear trajectories

In the TI method benchmark, SCORPIUS outperforms all other TI methods in inferring accurate models for datasets containing a linear trajectory [4]. Out of 45 TI methods – of which 14 were linear TI methods – SCORPIUS was the only method capable of producing top-scoring predictions on more than 50% of datasets containing linear trajectories (Figure 4.2A). Overall, SCORPIUS obtained the highest mean accuracy score on linear datasets, and was also one of the top ranked methods in terms of scalability, stability, and usability (Figure 4.2B).

We evaluated the gain in execution time due to optimisations made in the dimensionality reduction and the smoothing of the principal curve. In classical MDS, a square distance matrix between all cells is calculated. In Landmark MDS (LMDS), only the distances between a randomly selected set of landmarks and all other cells needs to be computed, reducing the execution time of the dimensionality reduction significantly (Figure 4.3A). In the standard principal curves algorithm, a curve consisting of $n - 1$ segments is iteratively smoothed with respect to the positions of n cells. By approximating the principal curve between iterations using a fixed number of seg-

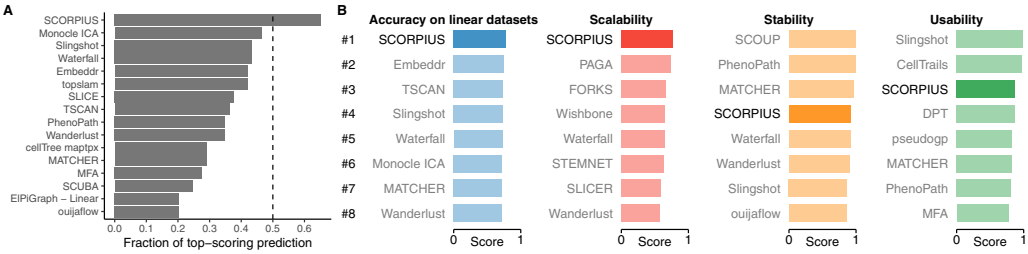


Figure 4.2: SCORPIUS outperforms 44 TI methods in inferring linear trajectories. A: It is the only method to produce top-scoring predictions on more than 50% of linear datasets. A predicted model is considered top-scoring if its accuracy is larger than 95% of the maximum accuracy obtained by any method on the same dataset. **B:** SCORPIUS ranks highly in all other categories: scalability, stability and usability. The scalability experiments were performed by upsampling a toy dataset and measuring the execution time and memory usage of each method. Stability experiments were performed by running each method multiple times on subsampled datasets and calculating the similarities between results. The usability of each method was determined by defining a list of good scientific and programming practices and determining to what extent each of these methods adhered to each aspect.

ments (e.g. 100), again the execution time of the principal curve algorithm is reduced significantly (Figure 4.3B).

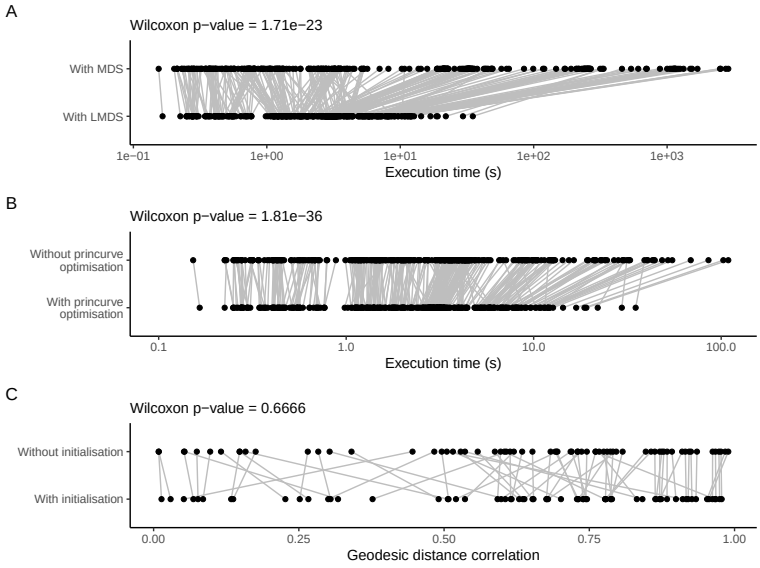


Figure 4.3: Comparison of the effect of different optimisations made in SCORPIUS. The optimisations in the dimensionality reduction (**A**) and principal curves (**B**) steps significantly reduce the execution times of SCORPIUS. Initialising the principal curve does not yield significant improvements on the accuracy of the predicted trajectory (**C**).

4.2.2 Functional modules in dendritic cell development

Applying SCORPIUS to a dataset of dendritic cell (DC) progenitors [5] reveals several sets of functional modules which are up- and down-regulated during development. DC progenitors are derived from hematopoietic stem cells in the bone marrow, and transition through multiple cellular states before becoming fully developed DCs [9]. The dataset contains 57 Monocyte and Dendritic cell Progenitors (MDPs), 95 Common Dendritic cell Progenitors (CDPs) and 96 Pre-Dendritic Cells (PreDCs). SCORPIUS correctly orders the cells with regard to their differentiation status, as indicated by comparing the inferred trajectory with the known transition states (Figure 4.4A).

In order to predict which genes are involved in DC development, SCORPIUS computes the importance value of the genes with respect to the pseudotemporal ordering and selects the top genes for further visualisation. Clustering the genes into gene modules allows to discover similar gene regulation patterns and gene functionality along the pseudotime (Figure 4.4C). In this dataset, modules 1 to 3 are downregulated during the development, while modules 4 to 6 are upregulated, indicating that as part of development the cells lose some functionality but gain others.

The gene expression changes shown in modules 1 to 3 are very gradual and mainly contain genes involved in early hematopoiesis or parallel hematopoietic lineage branches (module 1 and 2), and protein synthesis (module 3). These expression patterns of modules 1 and 2 are expected; as a DC progenitor develops into a DC, it will lose expression of genes associated with pluripotency. In addition, the protein synthesis rate has been shown to gradually decrease during granulocyte and B-cell development [11]. Module 3 suggests that an analogous process exists during DC development. We quantified the protein synthesis rate of murine bone marrow cells *in vivo* by intraperitoneally injecting O-propargyl-puromycin (OP-Puro). While the OP-Puro fluorescence intensities varied across the five individual mice, the relative fluorescence levels are very similar across replicates (Figure 4.4C) and show that indeed protein synthesis rates initially increase during early hematopoiesis but subsequently decrease during DC development.

While module 4 contains mostly genes that are already known to be involved in dendritic cell development, it nicely demonstrates the added benefit of pseudotemporal ordering as it is possible to distinguish which genes are upregulated first. Module 5 and 6 capture essential functionality of DCs: actin polymerisation plays a crucial role in determining a DC's morphology, migratory behaviour, and antigen internalisation (module 5, [12, 13]), and presenting antigens is one of the core responsibilities of a DC (module 6, [14]).

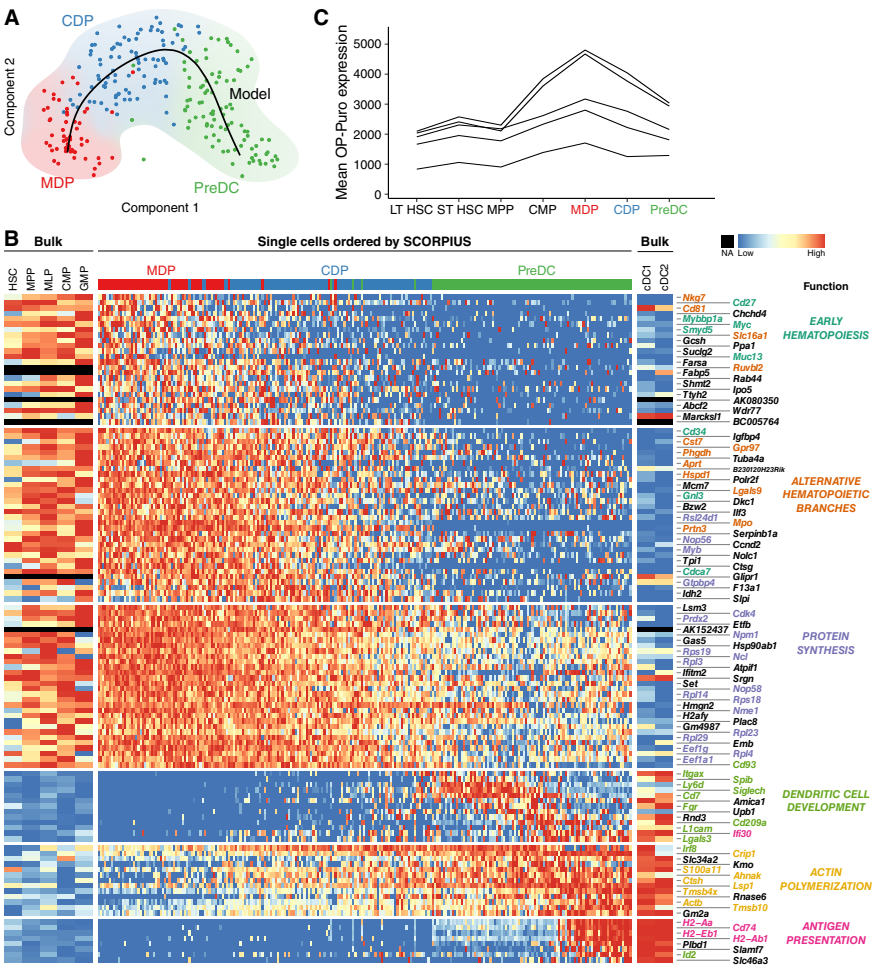


Figure 4.4: SCORPIUS sheds new, data-driven light on dendritic cell development. A: SCORPIUS creates an accurate model for DC development from scRNA-seq data. **B:** These genes are clustered into six gene modules. Each module is responsible for different aspects of DC development. Bulk microarray expression for up- and downstream stages of dendritic cell development [10] shows that the gene expression patterns uncovered by SCORPIUS are replicable in other datasets. **C:** In line with the decreasing transcript expression levels of protein translation genes, decreasing OP-Puro fluorescence levels indicates that the protein synthesis rate of preDCs progressively decreases during DC development.

4.3 Discussion

SCORPIUS is a significant milestone in accurately modelling a multi-stage progression of a dynamic process using single-cell omics datasets. It provides a complete pipeline for inferring, visualising and interpreting linear trajectories. While linear TI is a simpler case of generalised TI, we showed that it is still a challenging task, as most TI methods generally are not capable of deriving accurate pseudotemporal orderings on linear

Helena, thank you for being such a lovely colleague and an even better friend.
I enjoy our scientific talks and our non-scientific walks.

datasets. SCORPIUS outperforms the 44 other TI methods included in the benchmark in terms of accuracy and stability, and is amongst the top performing methods in terms of stability and usability.

4.4 Methods

SCORPIUS consists of three main steps: dimensionality reduction, trajectory modelling, and feature importance (Figure 4.5). The respective main algorithms for these steps are Multi-Dimensional Scaling (MDS) [6], Principal Curves [7], and Random Forests [8]. However, scRNA-seq datasets can have very high dimensionality (e.g. 100'000 cells and 10'000 features) but are typically very sparse (only 10% of values are non-zero). Each of these steps require modifications in order to be scaleable to large datasets (Sections 4.4.1-4.4.4).

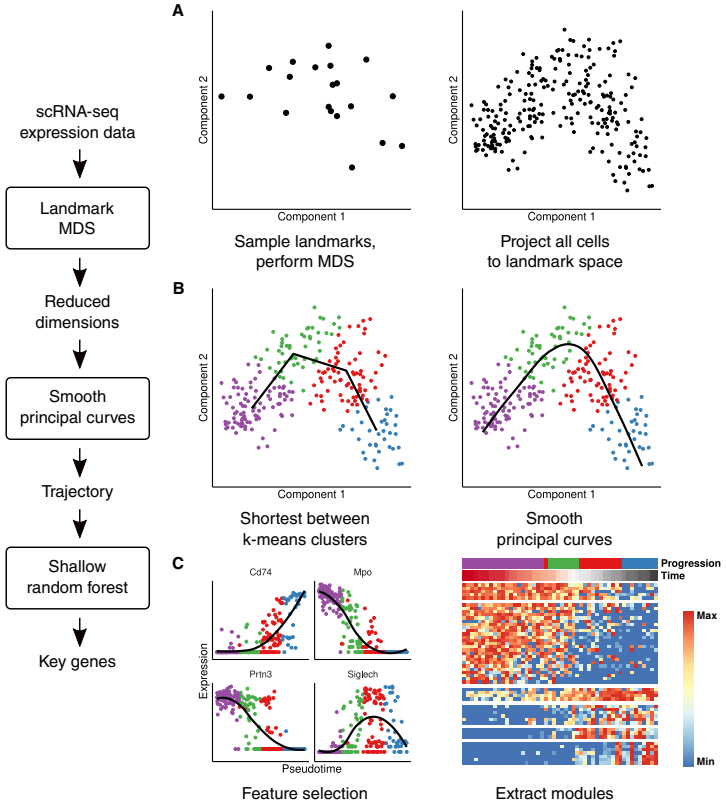


Figure 4.5: SCORPIUS consists of three main steps. A: Landmark MDS reduces the dimensionality of a small set of randomly sampled cells called landmarks. Afterwards, all other cells are projected to the landmark space. **B:** Smooth principal curves are used to pseudotemporally order the cells. The principal curve is initialised by connecting k-means clusters to improve robustness. **C:** Shallow random forests prioritise which genes best explain the pseudotemporal ordering.

4.4.1 Sparse Spearman Rank Correlation

Before dimensionality reduction, the distance between two cells x and y is calculated as the Spearman distance for tied ranks (Equation 4.1).

$$\begin{aligned} \text{dist}(x, y) &= \frac{1}{2} - \frac{\text{cov}(\overrightarrow{r_x}, \overrightarrow{r_y})}{2 \times \text{sd}(\overrightarrow{r_x}) \times \text{sd}(\overrightarrow{r_y})} \\ \text{with } \overrightarrow{r_x} &= \text{The rank of the expression values of } x, \\ \text{cov}(x, y) &= \text{The covariance between } x \text{ and } y, \\ \text{sd}(x) &= \text{The standard deviation of } x. \end{aligned} \tag{4.1}$$

The expression matrix is sparse, however, and computing the rank $\overrightarrow{r_x}$ of a gene X would result in a non-sparse vector. Instead, a transformed rank s_X is computed (Equation 4.2) such that $\text{cov}(\overrightarrow{r_x}, \overrightarrow{r_y}) = \text{cov}(\overrightarrow{s_x}, \overrightarrow{s_y})$ and $\text{sd}(\overrightarrow{r_x}) = \text{sd}(\overrightarrow{s_x})$. In practice, the expression values are strictly non-negative, but this solution generalises to matrices where negative values are allowed. Calculating the covariance and standard deviation of sparse vectors is relatively trivial.

$$\begin{aligned} s_{x,g} &= \begin{cases} 0 & \text{if } E_{x,g} = 0 \\ r_{x,g} - N_x + (T_{x,g} + Z_x)/2 & \text{if } E_{x,g} > 0 \\ r_{x,g} - N_x + (T_{x,g} - Z_x)/2 & \text{if } E_{x,g} < 0 \end{cases} \\ \text{with } E_{x,g} &= \text{the expression value of gene } g \text{ in cell } x \\ r_{x,g} &= \text{the rank of } E_{x,g} \text{ in } \overrightarrow{E_x} \\ N_x &= \text{the number of negative values in } \overrightarrow{E_x}, \\ Z_x &= \text{the number of zero values in } \overrightarrow{E_x}, \\ T_{x,g} &= \text{the number of values equal to } E_{x,g} \text{ in } \overrightarrow{E_x}, \end{aligned} \tag{4.2}$$

The distance can be computed using the `calculate_distance()` function from the CRAN package `dynutils`. This function is a wrapper for calculating the transformed rank on a sparse matrix, and calculating the Pearson correlation using `proxyC`.

4.4.2 Landmark Multi-Dimensional Scaling

Landmark MDS [15] is an extension of classical Torgerson MDS [6]. Classical MDS requires the computation of the distance matrix between all pairs of cells, which does

Paulina, like a soft summer wind you passed through our lives.
Thanks for putting so many healthy "vege-tables" and strange Polish delicacies in our fridge,

not scale well to large datasets. Instead, Landmark MDS only computes the pairwise distances between a small set of landmark cells (which should be representative of the whole population), and project all other cells to the landmark space. Landmark MDS is computed using the `lmds()` function from the CRAN package `lmds`.

4.4.3 Approximated Principal Curves

A principal curve is a smooth one-dimensional curve that passes through the middle of the dimensionality reduction [7]. To best fit the data, the curve is first initialised by k-means clustering the data into k clusters and calculating the shortest path going through each of the cluster centres. The curve is then iteratively refined by smoothing the coordinates curve in function of the distance from the start, and then orthogonally projecting all cells to the curve. The next iteration uses the curve defined by the segments spanned between the projections of the cells. The distance of a cell from the start of the curve is called its pseudotime.

In the original implementation of the principal curves algorithm, the curve would consist of $N - 1$ segments for a dataset containing N cells; thus the algorithm would not scale well to large datasets. After the smoothing step, a simplification of the curve has been added such that the curve is approximated by a fixed number of segments. The principal curve algorithm is implemented in the `principal_curve()` function from the CRAN package `princurve`.

4.4.4 Gene Importances

Gene importances are calculated by training a Random Forest [16] to predict the pseudotime values from the expression values in the dataset. The algorithm intrinsically computes a feature importance score which ranks the genes in terms of how well a feature is able to predict the pseudotime values. For computing the gene importance values, the `ranger()` function from the CRAN package `ranger` is used [17].

4.4.5 Datasets and benchmark results

The results of the benchmark analysis were obtained directly from the benchmark of 45 TI methods [4], available at github.com/dynverse/dynbenchmark_results. The accuracy, scalability, stability and usability metrics are described by Saelens et al. [4]. All datasets were obtained from a repository of single-cell omics datasets containing a trajectory hosted on Zenodo record number 1443566 [18].

4.4.6 Measurement of protein synthesis

O-Propargyl Puromycin (Jena Bioscience - NU-931-5) was dissolved in DMSO, further diluted in PBS ($10\,\mathrm{mg\,mL^{-1}}$) and injected intraperitoneally (50mg per kg mouse weight). 1 hour after injection mice were euthanized by cervical dislocation and hind bones were collected. Bone marrow cells were obtained by crushing of bones with pestle and mortar and subsequent lysis of red blood cells. The remaining cells were filtered through a $70\,\mathrm{\mu m}$ mesh and resuspended in a Ca^{2+} and Mg^{2+} free phosphate buffered solution (PBS; Gibco). Viable cell numbers were assessed with a FACS Verse (BD Biosciences).

7×10^6 cells were stained with mixtures of antibodies directed against cell surface markers. Each staining lasted approximately $30\,\mathrm{min}$ and was performed on ice protected from direct light. Monoclonal antibodies labeled with fluorochromes or biotin recognizing following surface markers were used: CD3 (145-2C11; Tonbo), TCRb (H57-597; BD Pharmingen), CD4 (RM4-5; eBioscience), CD8a (53-6.7; BD Pharmingen), CD19 (1D3; Tonbo), CD45R (RA3-6B2; BD-Pharmingen), TER119 (TER119; eBioscience), Ly-6G (1A8; BD-Pharmingen), NK1.1 (PK136; eBioscience), F4/80 (BM8; eBioscience), CD11c (N418; eBioscience), MHCII (M5/114.15.2; eBioscience), CD135 (A2F10; eBioscience), CD172a (P84; eBioscience), CD45 (30-F11; eBioscience), SiglecH (eBio440c; eBioscience), Ly-6C (HK1.4; eBioscience), CD115 (AFS98; eBioscience), CD117 (2B8; eBioscience), CD127 (SB/199; BD-Pharmingen), Ly-6A/E (D7; eBioscience), CD34(RAM34; eBioscience), CD11b (M1/70; BD Pharmingen). Viable cells were discriminated by the use of the fixable viability dye eFluor506 or eFluor786 (eBioscience).

Next, cells were fixed and permeabilized using the FoxP3 Fixation/Permeabilization kit (eBioscience, 00-5521-00). For OP-Puro labeling, Azide-AF647 is chemically linked to OP-Puro is through a copper-catalyzed azide–alkyne cyloaddition. In short, $2.5\,\mathrm{\mu M}$ azide-AF647 (Invitrogen, A10277) is dissolved in the Click-iT Cell Reaction Buffer (Invitrogen, C10269) containing $400\,\mu\mathrm{M}$ CuSO_4. Immediately after preparation, cells are incubated with this mixture at room temperature. After $10\,\mathrm{min}$ incubation, the reaction is quenched by addition of PBS supplemented with 5% heat-inactivated fetal calf serum (FCS; Sigma) and $5\,\mathrm{mM}$ EDTA (Lonza; 51234). Cells are washed twice to remove unbound azide-AF647. A Fortessa X20 (BD Biosciences) was used for data acquisition and data was analyzed using FlowJo 10 (LLC).

4.4.7 Code availability

SCORPIUS is available as an open source software package on CRAN. All code used in this study is made publicly available at github.com/rcannood/scorpius_analysis.

A big thank you to all the Dambini's for taking care of me while I was hard at work:
An,ArS,ArG,Art,C,D,H,I,Ja,Jon,Jor,Ka,Ke,Li,Lo,M,N,Pa,Pi,Q,RVd,RB,Ru,Sa,Sl,So,St,W,Y.

4.5 References

[1] Martin Etzrodt, Max Endele, and Timm Schroeder. "Quantitative Single-Cell Approaches to Stem Cell Research". In: *Cell Stem Cell* 15.5 (2014), pp. 546–558.

[2] Amos Tanay and Aviv Regev. "Scaling Single-Cell Genomics from Phenomenology to Mechanism". In: *Nature* 541.7637 (Jan. 2017), nature21350. ISSN: 1476-4687. DOI: 10.1038/nature21350.

[3] Robrecht Cannoodt, Wouter Saelens, and Yvan Saeys. "Computational Methods for Trajectory Inference from Single-Cell Transcriptomics". In: *European Journal of Immunology* 46.11 (Nov. 1, 2016), pp. 2496–2506. ISSN: 1521-4141. DOI: 10.1002/eji.201646347.

[4] Wouter Saelens et al. "A Comparison of Single-Cell Trajectory Inference Methods". In: *Nature Biotechnology* 37 (May 2019). ISSN: 15461696. DOI: 10.1038/s41587-019-0071-9.

[5] Andreas Schlitzer et al. "Identification of cDC1- and cDC2-Committed DC Progenitors Reveals Early Lineage Priming at the Common DC Progenitor Stage in the Bone Marrow". In: *Nature Immunology* 16.7 (July 2015), pp. 718–728. ISSN: 1529-2916. DOI: 10.1038/ni.3200.

[6] Warren S Torgerson. *Theory and Methods of Scaling*. John Wiley & Sons, 1958.

[7] Trevor Hastie and Werner Stuetzle. "Principal Curves". In: *Journal of the American Statistical Association* 84.406 (1989), pp. 502–516. ISSN: 01621459. DOI: 10.2307/2289936.

[8] Leo Breiman. "Random Forests". In: *Machine Learning* 45 (2001), pp. 5–32.

[9] Miriam Merad et al. "The Dendritic Cell Lineage: Ontogeny and Function of Dendritic Cells and Their Subsets in the Steady State and the Inflamed Setting.". In: *Annual review of immunology* 31 (2013), pp. 563–604. ISSN: 1545-3278. DOI: 10.1146/annurev-immunol-020711-074950. pmid: 23516985.

[10] Jennifer C Miller et al. "Deciphering the Transcriptional Network of the Dendritic Cell Lineage". In: *Nature Immunology* 13.9 (2012), pp. 888–899. ISSN: 1529-2908. DOI: 10.1038/ni.2370. pmid: 22797772.

[11] Robert A J Signer et al. "Haematopoietic Stem Cells Require a Highly Regulated Protein Synthesis Rate.". In: *Nature* 509.7498 (2014), pp. 49–54. ISSN: 1476-4687. DOI: 10.1038/nature13035. pmid: 24670665.

[12] Pablo Vargas et al. "Innate Control of Actin Nucleation Determines Two Distinct Migration Behaviours in Dendritic Cells.". In: *Nature cell biology* 18.1 (2016), pp. 43–53. ISSN: 1476-4679. DOI: 10.1038/ncb3284. pmid: 26641718.

[13] Zhenzhen Liu and Paul A. Roche. "Macropinocytosis in Phagocytes: Regulation of MHC Class-II-Restricted Antigen Presentation in Dendritic Cells". In:

Frontiers in Physiology 6 (JAN 2015), pp. 1–6. ISSN: 1664042X. DOI: 10.3389/fphys.2015.00001. pmid: 25688210.

[14] Ralph M Steinman. "The Dendritic Cell System". In: (1991), pp. 203–208.

[15] Vin de Silva and Joshua B Tenenbaum. "Sparse Multidimensional Scaling Using Landmark Points". In: *Technical report, Stanford University* (2004), p. 41.

[16] L Breiman et al. *Classification and Regression Trees*. Wadsworth Publishing Company, 1984.

[17] Marvin N Wright and Andreas Ziegler. "Ranger: A Fast Implementation of Random Forests for High Dimensional Data in C++ and R". In: *Journal of Statistical Software* 77.1 (Mar. 2017). DOI: 10.18637/jss.v077.i01.

[18] Robrecht Cannoodt et al. "Single-Cell -Omics Datasets Containing a Trajectory". In: *Zenodo* (Oct. 2018). DOI: 10.5281/zenodo.1211532.

Go to the next chapter for more acknowledgements.

5 | dyno: A toolkit for inferring and interpreting trajectories

Abstract

Purpose: Recent technological advances allow studying dynamic processes by computationally ordering single-cell omics profiles along a trajectory. Since 2014, at least 75 tools for trajectory inference have been developed. For end-users, however, these methods are difficult to execute and compare, mainly due to high variability in input/output data structures, software requirements, and programming interfaces.

Results: We developed dyno, a toolkit for inferring, visualising and interpreting single-cell trajectories, giving access to more than 50 TI methods. dyno provides downstream analyses such as visualising the trajectory with dot-plots or heatmaps, automated labelling using gene sets, automated inference of the directionality, and detecting genes that are differentially expressed along the trajectory.

Conclusion: By providing a uniform interface for executing and visualising trajectories of 50 methods, dyno allows inferring and comparing trajectories for a much broader audience than previously possible.

Publication status

Manuscript in preparation.

Cannoodt R*, Saelens W*, and Saeys Y.

* Equal contribution

Author contributions

- **R.C.** and W.S. designed the project.
- **R.C.** and W.S. implemented software packages.
- **R.C.** and W.S. wrote the original manuscript.
- Y.S. supervised the project.

5.1 Introduction

Recent technological advances allow unbiased investigation of cellular dynamic processes in a high-throughput manner [1, 2]. Trajectory inference (TI) methods aim to give insight into a dynamic process by inferring a trajectory from omics profiles of cells in which the dynamic process takes place [3]. In a recent study, we benchmarked 45 TI methods in terms of their accuracy, scalability, stability, and robustness [4]. We construct a set of guidelines to help end-users select an appropriate TI method for their dataset of interest. However, executing and comparing multiple methods remains challenging, due to high variability in input/output data structures, software requirements, and programming interfaces.

We developed dyno, a toolkit to easily infer, visualise and interpret single-cell trajectories. The user can select the most optimal set of TI methods based on characteristics of the dataset and user preferences. More than 50 TI methods can easily be run within a common interface, and the outputs thereof are converted into a common format. dyno provides downstream analysis such as: visualising a trajectory in a low-dimensional space or a heatmap, detecting genes differentially expressed at different stages of the tragectory, comparing multiple trajectories in a common dimensionality reduction, and manipulating the trajectory such as adding directionality or labelling different cell stages.

5.2 Results

This section will present the various components in the dyno workflow (Figure 5.1).

5.2.1 Preparing the dataset

In order to use your dataset of interest, you will first need to wrap it in a dynwrap object. We will use a commonly used dataset of 392 mouse embryonic fibroblasts (MEF) undergoing direct reprogramming to induced neuronal cells [5]. By wrapping the counts information and possible prior information (e.g. known cell types, a start cell, time course information) into a dynwrap object, the dataset can be used as input for the various dyno components.

Because relaxing is as important as work,
a special thanks to a selection of the friends and family.

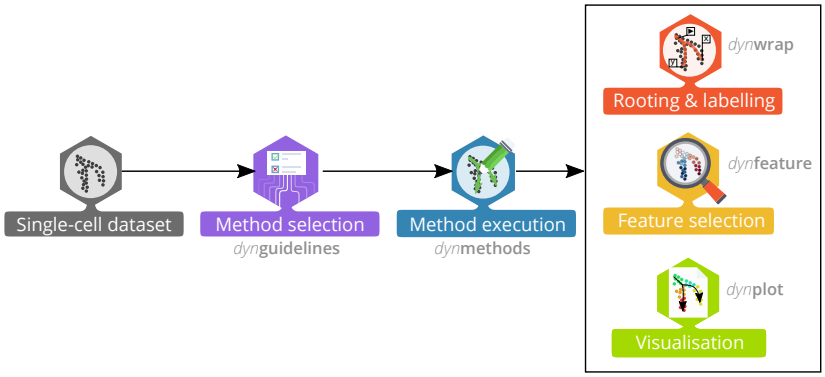


Figure 5.1: The dyno workflow for inferring, visualising and interpreting single-cell trajectories. dynguidelines allows you to choose a suitable TI method for your dataset of interest. dynmethods contains wrappers for 'r dynmethods::methods dynwrap provides main functionality for (pre/post-)processing trajectory data structures. dynfeature calculates gene importance scores based on a trajectory. dynplot provides visualisation tools for trajectories, including scatterplots and heatmaps.

5.2.2 Selecting the best methods for a dataset

We performed a comparative study of 45 trajectory inference methods [4]. We evaluated each method in terms of four main aspects:

- **Accuracy**: How similar is the inferred trajectory to the "true" (or "expected") trajectory in the data. We used several metrics in order to assess similarity in pairwise cellular ordering and also the similarity in topology of the trajectories. We used both real datasets – which has the highest biological relevance – and synthetic datasets – which allow to push methods to their limits more easily.
- **Scalability**: How long the method takes to run and how much memory it consumes. This mainly depends on the dimensions of the input data, i.e. the number of cells and features.
- **Stability**: How stable the results are when rerunning the method with slightly different input data.
- **Usability**: The quality of the corresponding documentation, software package, and manuscript. We assessed how easy it is to run the method, whether the software adheres to good software development practices, and whether the manuscript follows good scientific practices. We reviewed 'good practices' literature and created a 'consensus' score-sheet of good practices, and filled in to what extent each method adhered to each of the good practices.

We found a high diversity in method performance, and that not many methods perform well across the board. The performance of a method depended on many factors, mainly the dimensions of the data and the kind of trajectory present in the data. Based

on this, we developed an interactive Shiny [6] app which can be used to explore the results and select an optimal set of methods for a particular analysis (Figure 5.2).

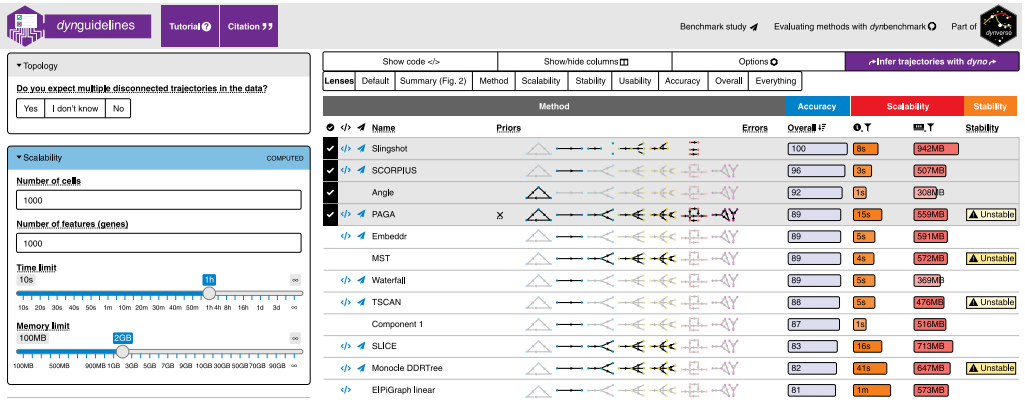


Figure 5.2: A static screenshot of the dynguidelines Shiny interface.

5.2.3 Inferring trajectories

There are over 50 TI methods available as part of dynmethods, and executing a TI method on a dataset of interest is as simple as running just one command. Behind the screens, dyno will save the dataset, prior information and parameters as an H5 file and execute a container containing the desired TI method. This ensures that the TI method will not have any issues due to software compatibilities on the host environment. The first time a TI method is run, it will automatically download the corresponding container from Docker Hub.

We inferred a trajectory on the dataset of mouse embryonic fibroblasts using the Slingshot [7] approach. The outputted model contains the main information on the trajectory, namely the milestone network and cell progressions (Table 5.1). These data are hard to interpret manually. Luckily, dyno provides many different visualisation functions for interpreting trajectories.

5.2.4 Execution details

Prior information: The method will error if it requires some prior information that is not already provided in the dataset. Take note that the stronger the assumed priors for a given method, the more biased the output will be (Table 5.2).

Reproducibility: When using this framework for analysing real data, remember to set the seed to ensure reproducibility.

Table 5.1: The minimum amount of information to define a trajectory.

Milestone network:

from	to	length	directed
5	1	13.828399	TRUE
1	2	4.853443	TRUE
2	3	11.325468	TRUE
2	4	14.022826	TRUE

Cell progressions:

cell_id	from	to	percentage
1_iN1_C04	2	3	0.3508990
1_iN1_C05	2	3	0.5287099
1_iN1_C07	1	2	0.1953806
1_iN1_C08	2	3	0.8589863
1_iN1_C09	1	2	0.9076217
1_iN1_C10	1	2	0.4170644
1_iN1_C11	2	3	0.7316277
1_iN1_C12	1	2	0.7108862
1_iN1_C13	1	2	0.0644059
1_iN1_C14	1	2	0.7178277
	...		

Table 5.2: Possible prior information accepted or required by TI methods.

Name	Description	Type
Start cell(s)	One or more start cell identifiers	soft
End cell(s)	One or more end cell identifiers	soft
# end states	The number of end states	soft
# start states	The number of start states	soft
# leaves	The number of leaves	soft
Cell clustering	Named character vector linking the cell identifiers to different states/branches	hard
# states	Number of states/branches, including start, end and intermediary states	soft
State network	Dataframe containing the known network between states/branches. Contains a from and to column	hard
Time course (continuous)	Named numeric vector linking the cell ids to time points	hard
Time course (discrete)	Named numeric vector linking the cell ids to time course points	hard
Marker genes	Genes/features known to be important in the dynamic process	soft

Multiple executions: Often it is useful to run multiple methods and/or use multiple datasets. While you can easily parallelise this yourself, we provide a helper function which integrates well with our visualisation functions.

Errors: Some methods can generate errors which are beyond our control. To know what and where a method generated an error, remember to turn on the verbosity.

Command-line: Each method is also executable from command-line using the Docker container. Check out the help documentation by running a TI method container without extra arguments.

5.2.5 Visualising trajectories

The most common way to visualise a trajectory is to plot it on a dimensionality reduction of the cells (Figure 5.3A). Often (but not always), a TI method will already output a dimensionality reduction itself, which was used to construct the trajectory. dynplot will use this dimensionality reduction if available, otherwise it will calculate a dimensionality reduction under the hood. You can also supply it with your own dimensionality reduction. In this case, the trajectory will be *projected* onto this dimensionality reduction (Figure 5.3B). On this plot, you can colour the cells using various layers of information (Figure 5.4): cell ordering, cell grouping, feature expression, and pseudotime.

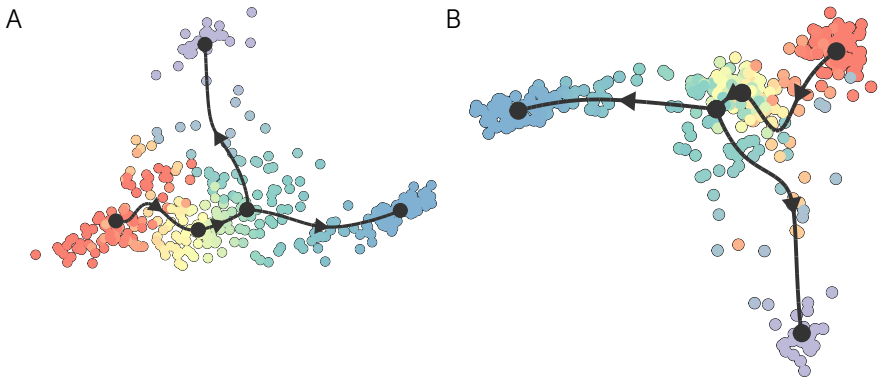


Figure 5.3: A: Often, the TI method provides its own dimensionality reduction for the trajectory, which will be plotted by default. This is a visualisation of Slingshot [7] executed on MEF cells from the previous example. **B:** Alternatively, you can provide your own dimensionality reduction, and the trajectory will be projected to it. In this example the Slingshot trajectory is projected to a UMAP dimensionality reduction [8].

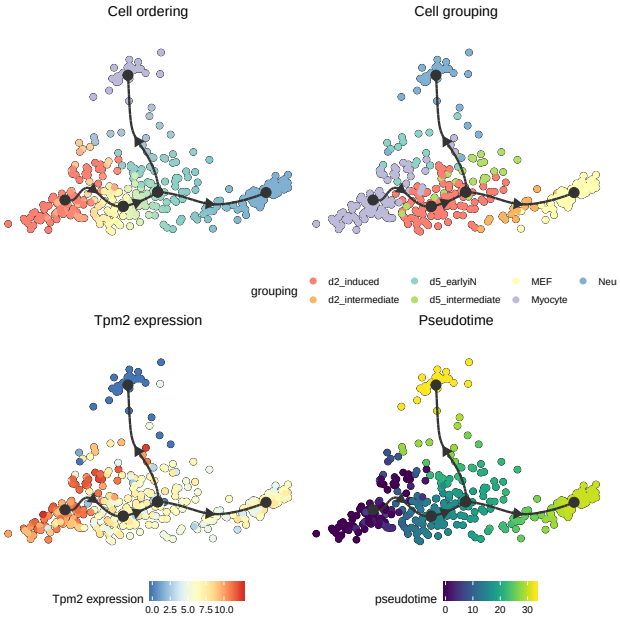


Figure 5.4: Cell ordering: Cells are coloured by their proximity to the different milestones. **Cell grouping:** A given character vector mapping a cell to a group. **Feature expression:** Colour by the expression levels of a particular gene. **Pseudotime:** The distance to a particular root milestone.

Various layout and colouring options: To visualise a trajectory, it is good practice to take into account what the purpose and intended message of the visualisation is (Figure 5.5). The cells and trajectory can be positioned to place more emphasis on the topology of the inferred trajectory, or on the cellular heterogeneity between the cells, that the trajectory might or might not have been able to detect. Cells and trajectory can be coloured according to the topology of the trajectory, according to gene expression, or a custom input vector of values.

Visualising many genes along a trajectory: A one-dimensional visualisation is especially useful if you combine it with a heatmap (Figure 5.6).

Comparing multiple trajectories: Visualising each method with their own dimensionality reduction can make it hard to interpret to what extend the methods agree/disagree with each-other (Figure 5.7A). Different trajectories become noticeably more comparable when projected to a common dimensionality reduction (Figure 5.7B).

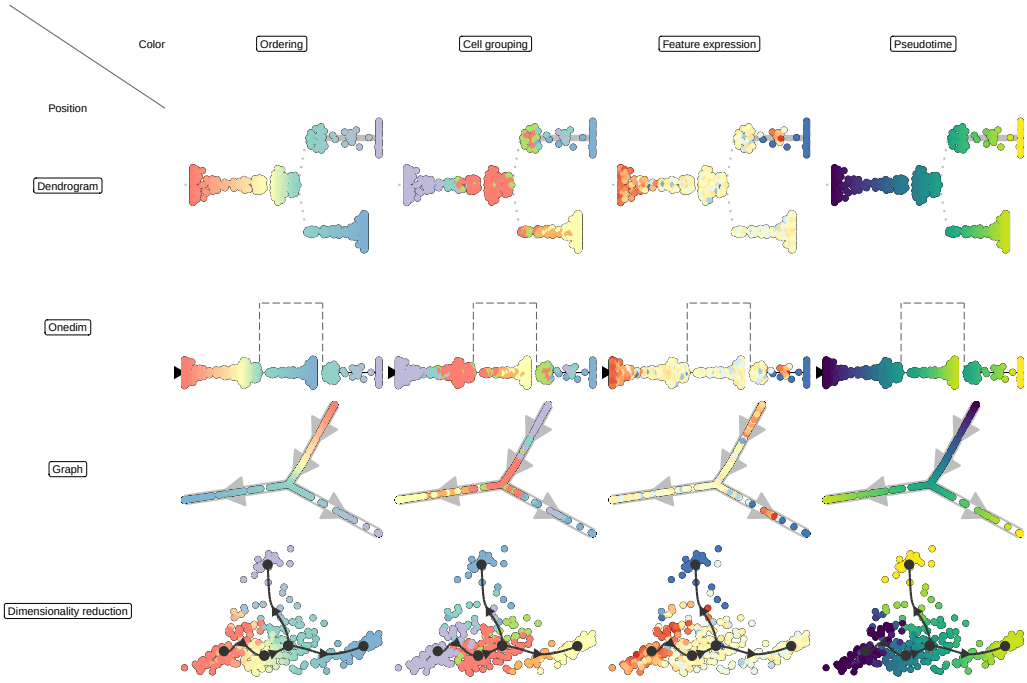


Figure 5.5: Overview of the different combinations of positioning and colouring options, demonstrated on the output of Slingshot [7].

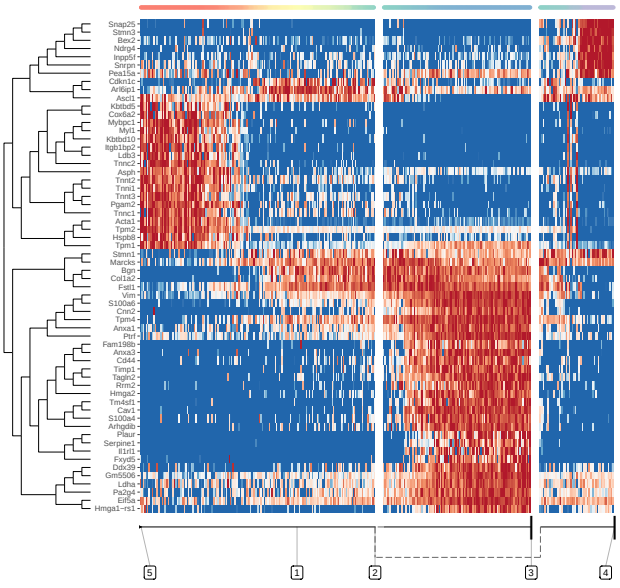


Figure 5.6: Selecting relevant features for this heatmap is discussed in a later section, By default, features will be selected that best explain the main differences over the whole trajectory.

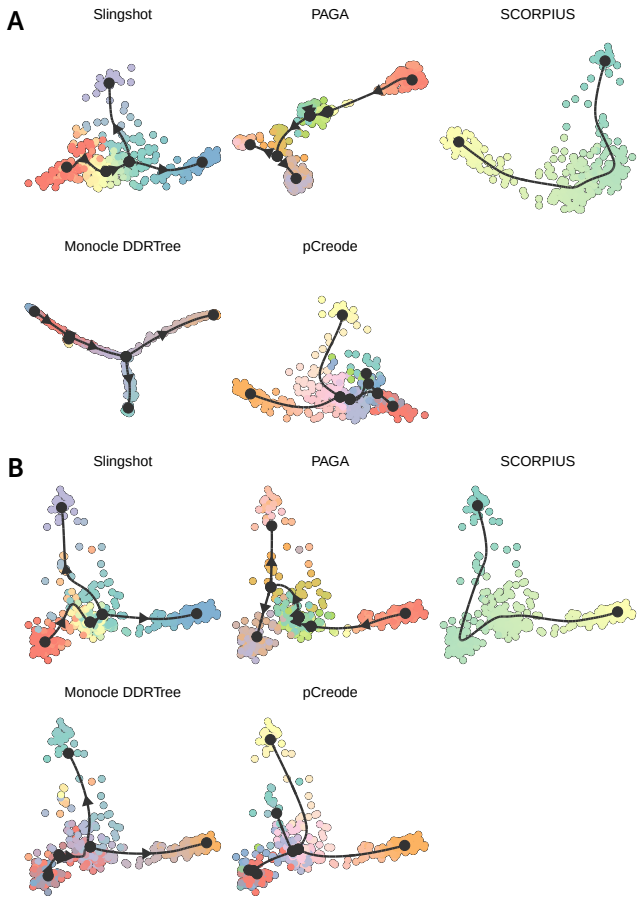


Figure 5.7: A: Comparing outputs of multiple TI methods is tedious when they are each visualised using their own visualisation method of interest. **B:** In contrast, by projecting each output to the same dimensionality reduction, the methods become immediately visually comparable.

5.2.6 Manipulating the trajectory

dyno allows manipulating trajectories by simplifying, rooting and annotating them.

Simplifying linear segments: Intermediate milestones can be removed by simplifying the trajectory (Figure 5.8A).

Rooting: Most TI methods return undirected trajectories. We provide two ways of rooting a trajectory, namely by manually specifying the root milestone, or by specifying highly expressed marker genes. After rooting, all other edges will point away from the root (Figure 5.8B).

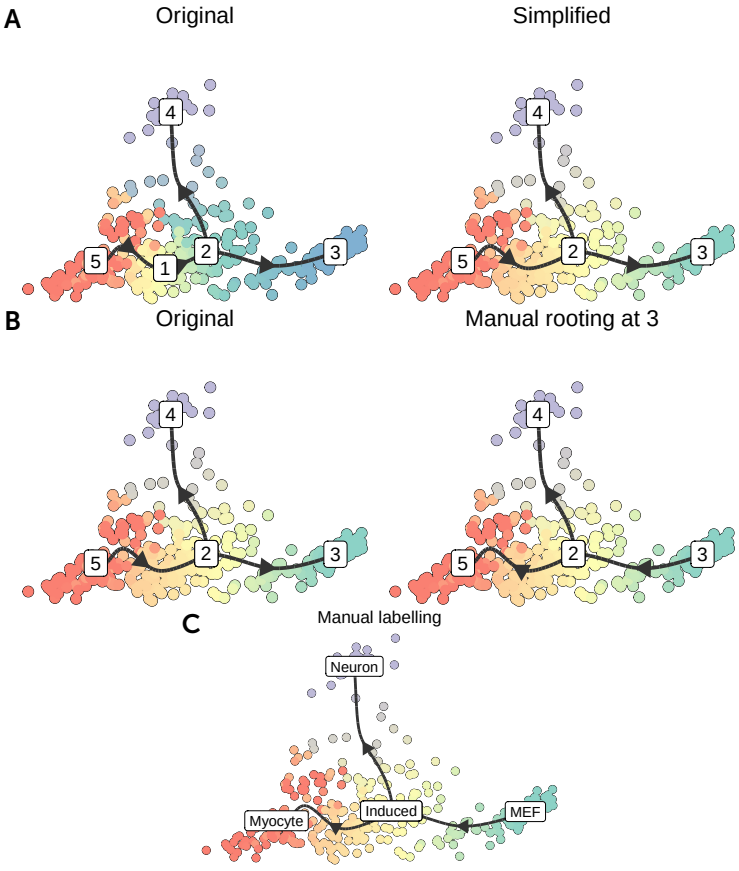


Figure 5.8: **A:** Simplification of the trajectory. **B:** Rooting the trajectory. **C:** Labelling the trajectory.

Annotating the trajectory: Similar as with rooting, annotating the trajectory by renaming the milestones can be done either manually, or with given highly expressed gene sets (Figure 5.8C).

5.2.7 Differentially expressed genes along the trajectory

Compared to differential expression between clusters of cells, defining differential expression on trajectories is not so straightforward. What constitutes a trajectory differentially expressed gene?

- A gene that is uniquely expressed in a particular branch?
- A gene that changes at a branching point?
- A gene that changes along pseudotime?

dynfeature allows you to find these different kinds of differential expression in a trajectory. It first defines a particular variable that needs to be predicted (for example,

feature_id	from	to	importance
Cav1	3	2	3.289198
S100a6	3	2	2.942703
1810020D17Rik	2	5	2.822247
Tagln2	3	2	2.763266
Bex2	2	4	2.672312
Anxa3	3	2	2.630013
Tm4sf1	3	2	2.547472
Vim	3	2	2.431169
Hmga2	3	2	2.376114
Rrm2	3	2	2.346754

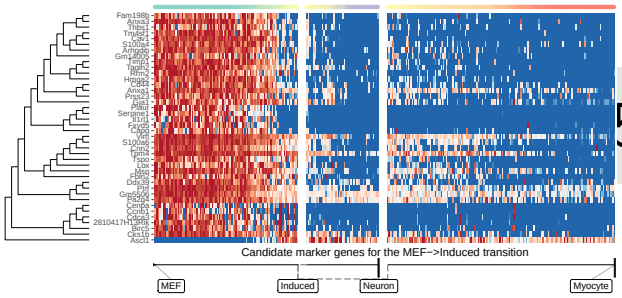


Figure 5.9: Candidate markers genes for the MEF → transition.

milestone_id	feature_id	importance
2	S100a6	2.654416
2	Cav1	2.567051
2	Tagln2	2.501074
2	Fam198b	2.334142
2	Timp1	2.056474
2	Anxa3	2.009431
2	Tm4sf1	1.982274
2	Vim	1.833045
2	Bex2	1.738471
2	Rrm2	1.678349

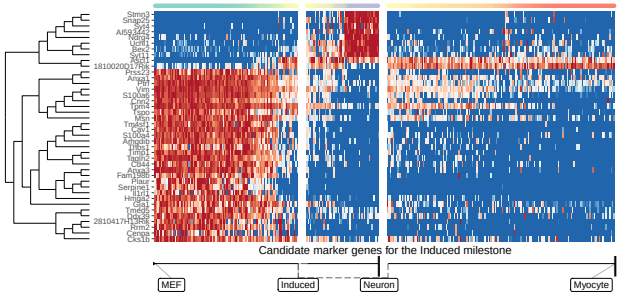


Figure 5.10: Candidate markers genes for the 'induced' milestone.

whether a cell is present in a branch or not), and ranks each gene based on their predictive capability with respect to that variable. This section reviews the types of feature selection supported by dynfeature.

Lineage / transition marker genes: We can identify genes that are specifically upregulated or downregulated at a specific branch (Figure 5.9).

Milestone marker genes: We can identify genes that are specifically upregulated or downregulated at a particular milestone (Figure 5.10).

Marker genes for the trajectory as a whole: We can identify genes that change in function of the ordering of a part of the trajectory (Figure 5.11).

5.3 Discussion

With dyno, we provide a feature-complete toolbox for inferring, visualising and annotating trajectory data. In this work, we demonstrated its usefulness by applying all of

feature_id	importance
Tpm2	0.4539798
Tagln2	0.3364078
Vim	0.3319566
Plaur	0.3252199
Cav1	0.3224918
Hmga2	0.3189412
Ptrf	0.3180152
Tnnc2	0.3092800
Acta1	0.3087145
Myl1	0.3014400

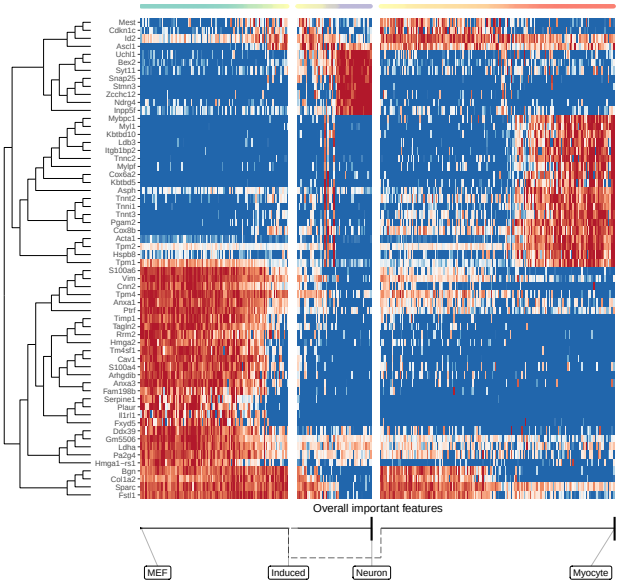


Figure 5.11: Candidate markers genes for the overall trajectory.

its visualisation and manipulation functionality on a particular dataset.

Alternative modalities such as ATAC-Seq or cytometry data are not yet supported, although it is possible to simply include this data as expression and counts. RNA velocity [9] would be particularly useful, as it would allow rooting the directory without any further input from the user.

We are also planning to implement additional tools for interpreting the trajectories, such as alignment and differential expression. While the feature importance tools are incredibly useful towards interpreting trajectories, they do not yet provide a statistical ground to find features which are significantly differentially expressed. In addition, depending on the size of the dataset and of the predicted trajectory, it might take a long time to compute.

5.4 References

[1] Amos Tanay and Aviv Regev. “Scaling Single-Cell Genomics from Phenomenology to Mechanism”. In: *Nature* 541.7637 (Jan. 2017), nature21350. ISSN: 1476-4687. DOI: 10.1038/nature21350.

[2] Martin Etzrodt, Max Endele, and Timm Schroeder. “Quantitative Single-Cell Approaches to Stem Cell Research”. In: *Cell Stem Cell* 15.5 (2014), pp. 546–558.

[3] Robrecht Cannoodt, Wouter Saelens, and Yvan Saeys. “Computational Methods for Trajectory Inference from Single-Cell Transcriptomics”. In: *European Journal of Immunology* 46.11 (Nov. 1, 2016), pp. 2496–2506. ISSN: 1521-4141. DOI: 10.1002/eji.201646347.

[4] Wouter Saelens et al. “A Comparison of Single-Cell Trajectory Inference Methods”. In: *Nature Biotechnology* 37 (May 2019). ISSN: 15461696. DOI: 10.1038/s41587-019-0071-9.

[5] Barbara Treutlein et al. “Dissecting Direct Reprogramming from Fibroblast to Neuron Using Single-Cell RNA-Seq”. In: *Nature* 534.7607 (2016), pp. 391–395.

[6] Rstudio. “A Web Application Framework for R”. In: (2016).

[7] Kelly Street et al. “Slingshot: Cell Lineage and Pseudotime Inference for Single-Cell Transcriptomics”. In: *BMC Genomics* 19.1 (June 2018), p. 477. ISSN: 1471-2164. DOI: 10.1186/s12864-018-4772-0.

[8] Leland McInnes, John Healy, and James Melville. “UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction”. In: (2018).

[9] Gioele La Manno et al. “RNA Velocity of Single Cells”. In: *Nature* 560.7719 (Aug. 2018), pp. 494–498. ISSN: 1476-4687. DOI: 10.1038/s41586-018-0414-6.

6 | bred: Inferring single cell regulatory networks

Abstract

Purpose: Network inference methods are computational tools which use large omics datasets to predict which genes are regulated by which transcription factors. Since regulatory interactions are context-dependent, attempting to model regulatory dynamics in the form of a single regulatory network may have little relevance.

Results: In this work, we introduce bred, the first algorithm for inferring case-wise regulatory networks. Analysing case-wise regulatory networks of 14'963 profiles from The Cancer Genome Atlas project demonstrated that regulatory interactions are often specific to particular cancer types, but can also be shared between several cancer types.

Conclusion: Ultimately, this analysis has resulted in a list of candidate oncogenic genes and interactions per cancer type, though further analysis and validation is required to confirm their oncogenicity.

Publication status

Manuscript in preparation.

Cannoodt R, Saeys Y*, and De Preter K*.

* Equal contribution

Author contributions

- **R.C.**, K.D.P., and Y.S. designed the study.
- **R.C.** performed the experiments and analysed the data.
- **R.C.** implemented the bred software package.
- **R.C.** wrote the original manuscript.
- **R.C.**, K.D.P., and Y.S. reviewed and edited the manuscript.
- K.D.P. and Y.S. supervised the project.

6.1 Introduction

6

One of the central cellular processes underlying development is transcriptional regulation. Modelling the dynamics of gene regulation is therefore essential to better understand why a cellular dynamic processes progresses through several steps, and what goes wrong in the case of disease. Regulatory dynamics is classically studied using time series data [1]. When dynamic processes progress asynchronously, such as in hematopoiesis, time series data are usually obtained by sorting different transition states and assessing bulk gene expression and transcription factor binding within the population [2, 3, 4, 5]. Alternatively, time series data can also be generated by synchronising the dynamic process between cells. However, issues with time-resolution, heterogeneity and good *in vivo* synchronisation models can often limit the predictive power of the dynamic models of gene regulation which can be constructed [1].

Network inference (NI) methods are computational tools which use large omics datasets to predict which genes are regulated by which transcription factors. While accuracy of the network of predicted regulatory interactions is lower in comparison to experimental validation techniques, NI methods offer an unbiased and high-throughput insight into the regulatory dynamics of a biological system. The output of a network inference is thus a graph, where nodes represent genes and edges denote a regulatory interaction between a regulator and a target gene. Interactions have two properties: its regulatory strength (a positive real value) and its effect (promoting or repressing).

Several studies have highlighted how some regulatory interactions can be very dynamic while others show evidence of being static during consecutive developmental stages [6, 7]. Since regulatory interactions are context-dependent [8], attempting to create an accurate model of those processes by inferring a static regulatory network may have limited relevance. Case-wise NI methods[1] avoid predicting a static GRN and instead infer one GRN per cell (or per sample, for bulk omics data).

In order to compute a case-wise GRN for a single sample, Kuijjer et al. [9] and Liu et al. [10] employ similar strategies, namely by computing the difference of computing a static GRN for all the cases, and computing a static GRN for all the cases minus one. Since this procedure needs to be repeated for every case in the dataset, and because NI methods are already amongst the most computationally intensive analyses to perform on omics data, this methodology is not applicable for large omics datasets. Another case-wise NI method, SCENIC [11] infers case-wise GRNs by first inferring a static GRN using GENIE3 [12]. GENIE3 is a static NI method which uses Random Forests (RFs) [13] variable importance scores to prioritise candidate regulators for

[1] Case-wise NI is sometimes also called sample-specific NI or case-specific NI.

Big hugs for my family who supported me (Vanessa, Marilou, Franco, Alain, Luc, Malika, Sien, Silke, Liam, Anne, Patrick, Evelyne, Stefaan, Arne, Laure,

a particular target gene. SCENIC then post-processes the static GRN to determine whether an interaction is enriched for particular cases, resulting in a case-wise GRN.

Thus, while several case-wise NI methods have already been proposed, their implementation consisted of post-processing a static GRN to arrive at a case-wise GRN. As such, these methods will most likely recover the interactions that are prevalent in the whole population, and will miss interactions that are specific to only a sub-population.

In this work, we introduce `bred`, the first 'true' case-wise NI method. It uses a modified version of the RF variable importance scores used by GENIE3 and SCENIC to compute importance values for each profile and each interaction separately, as well as predict the effect of each interaction (activating or repressing). We generate case-wise GRNs – or case-wise regulomes – for 14'963 profiles from The Cancer Genome Atlas (TCGA) [14], resulting in 6'464'915 predicted case-wise interactions. We analyse these case-wise regulomes by clustering them and detecting highly activated interactions.

6.2 Results

At the time of writing, the TCGA database contains 14'963 profiles from 44 different cancer entities. We used the `bred` algorithm to infer case-wise regulomes for each of these profiles. In total, we detected 73'140 unique interactions and, on average, 7'231 active interactions per profile. Dimensionality reduction and clustering of the case-wise regulome data provides a visual overview of the different subpopulations (Figure 6.1).

For each cluster, we computed the average importance of each interaction. Retaining the 100 strongest interactions (Figure 6.2) shows that different cancer entities have vastly different regulomes, yet often have shared regulators and interactions (Table 6.1).

Performing a more in-depth analysis of particular clusters can reveal distinct subpopulations within. For example, when comparing samples of normal and healthy tissue within breast carcinoma (Figure 6.3), a significant number if interactions are predicted to have been turned off (top), while others have been turned on (middle, bottom). Similar observations can be made for other clusters, such as for melanocytic neoplasm (Figure 6.13) and kidney carcinoma (Figure 6.14).

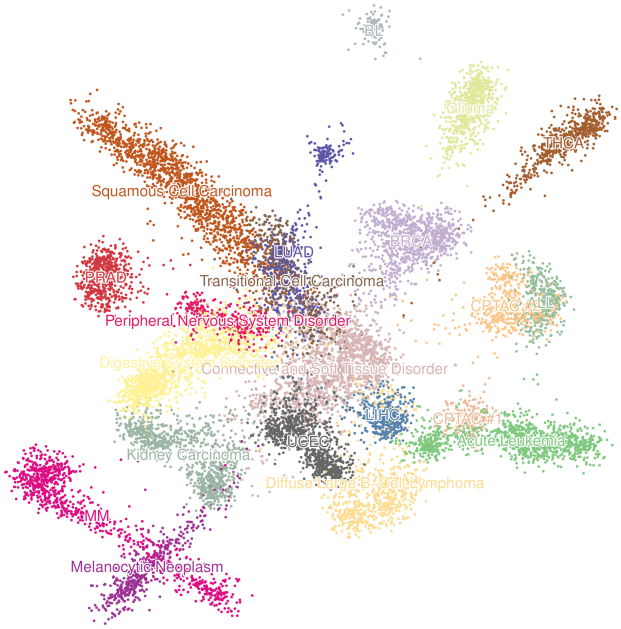


Figure 6.1: Visualisation of 14'963 case-wise regulomes of cancer profiles from The Cancer Genome Atlas project. Dimensionality reduction of the case-wise regulomes was performed by applying Fruchterman-Reingold [15] on the k-nearest-neighbour graph ($k = 100$) of highly similar regulomes (Spearman correlation). The samples were clustered with Louvain clustering [16] and each cluster was assigned an ontology term from the NCI thesaurus ontology [17] which best fits the sample in the cluster. Clusters for which the term has a positive predictive value (PPV) lower than 0.5 are marked with an asterisk (*). More detailed information on the enrichment of particular terms or metadata for each of the clusters can be found in Figures 6.6–6.12.

6.3 Discussion

The `bred` algorithm is a novel approach for directly computing case-wise regulomes for single-cell omics and bulk omics profiles alike. We used `bred` to infer case-wise regulomes for 14'963 cancer profiles from The Cancer Genome Atlas project. Analysing the results using common omics algorithms (dimensionality reduction, clustering, enrichment) has shown that regulatory interactions are often specific to particular cancer types, but can also be shared between several cancer types. Ultimately, this analysis has resulted in a list of candidate oncogenic genes and interactions per cancer type, though further analysis and validation is required to confirm their oncogenicity.

Future perspectives include applying it on single-cell omics data (e.g. from Tabula Muris [18]) and benchmarking the algorithm against *in silico* datasets (e.g. produced by `dyngen` [19]).

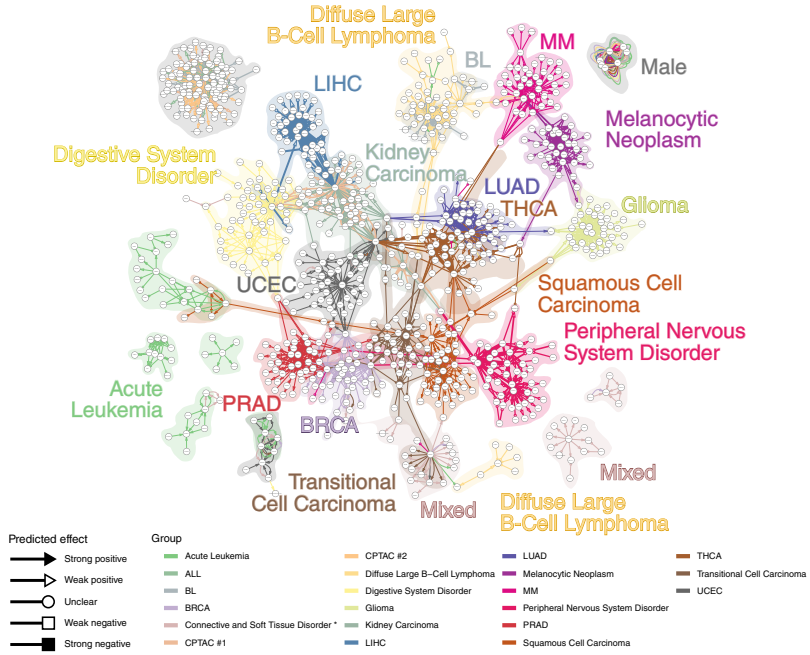


Figure 6.2: Visualisation of the strongest interactions per cluster shows both pathways distinct to particular cancer entities as well as pathways common to multiple cancer entities.

6.4 Methods

6.4.1 Inferring case-wise regulomes

The task of inferring a static GRN (Figure 6.4A) can be reduced to a simpler problem, namely: for every target T, predict which of the potential regulators R_i regulate T (Figure 6.4B). This simplification allowed GENIE3 [12] to use Random Forest's [13] feature importance scores for inferring GRNs. Namely, a Random Forest is trained to predict the expression of a target gene of interest from the expression of potential regulators. The resulting Random Forest inherently allows to extract a feature importance score by observing the effect of each regulator in making a good prediction for the target expression. As in GENIE3, the target expression is first scaled to normalise feature importance scores across different targets.

We make the same simplification in order to build case-wise GRNs, also using Random Forests to compute the feature importance scores. A Random Forest consists of K trees, each of which produces feature importance scores, and the feature importance scores of a forest is simply the mean feature importance scores of each of the trees.

Computing the case-wise feature importances of a tree consists of the following 8 steps (Figure 6.5). The 'randomness' of a Random Forest is in part due to using a subset

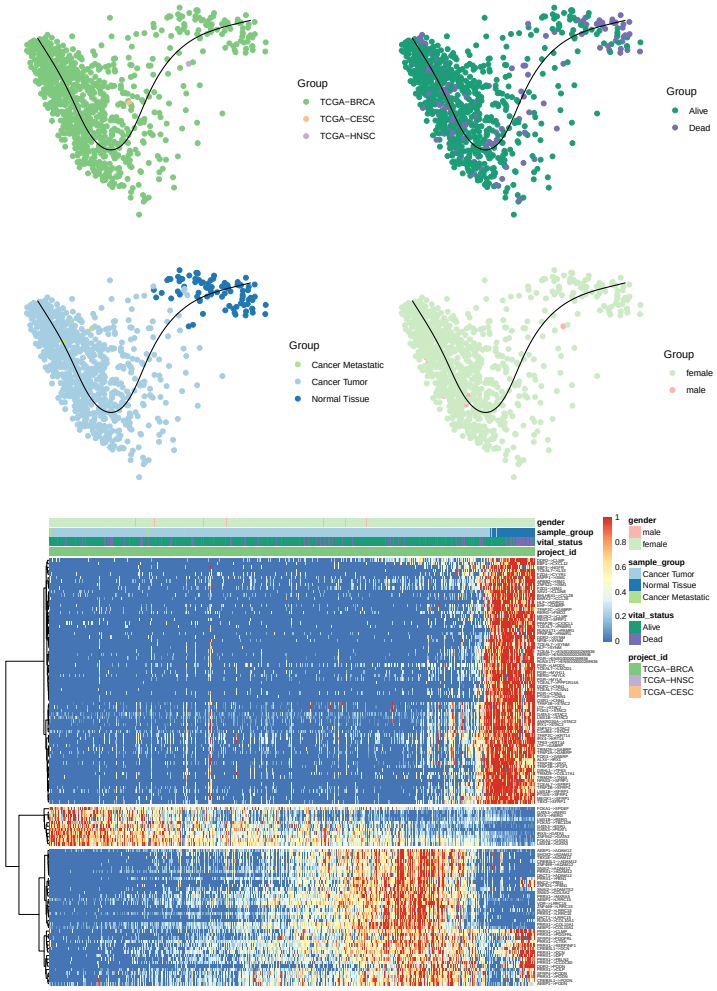


Figure 6.3: In-depth view of cluster 17, breast carcinoma. Top: A dimensionality reduction of the samples, coloured according to multiple sources of meta-data. **Bottom:** The samples were ordered linearly with SCORPIUS (see trajectory in **Top**) in order to visualise regulome activity in the form of a heatmap.

of the samples in the dataset in order to build a single decision tree. The samples are split into two groups, the 'in-bag' data and the 'out-of-bag' data (Figure 6.5A). A decision tree [20] is trained on the in-bag expression of the potential regulators in trying to predict the in-bag target gene expression (Figure 6.5B). The target expression of the out-of-bag samples is predicted using the decision tree (Figure 6.5C), and the squared error between the real and target expression is computed (Figure 6.5D). For each sample in the out-of-bag set, this vector represents how well the decision tree was able to predict the expression of the target gene.

The next few steps are repeated for every potential regulator R_i. Within the out-of-

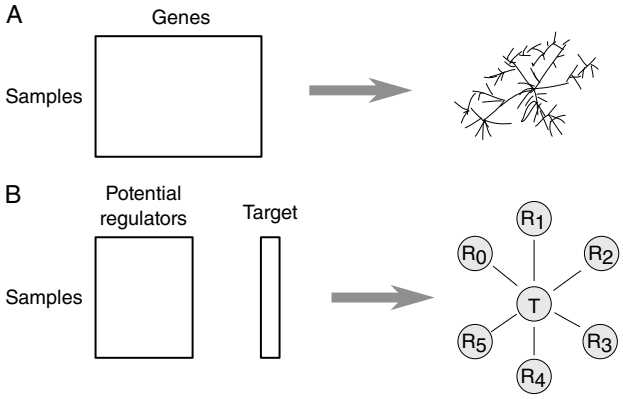


Figure 6.4: A: Inferring a gene regulatory network from an omics dataset can be reduced to a simpler problem. **B:** Given the expression of a target of interest and a set of potential regulators, predict which regulators regulate the target gene.

bag samples, the expression of R_i is randomly shuffled. The target expression of the out-of-bag samples is again calculated (Figure 6.5F), as well as the squared error between the real target expression and the predicted expression is calculated (Figure 6.5G). The importance of regulator R_i for an out-of-bag sample S_j is defined as the increase in squared error between the predicted target expression and the real target expression, after perturbing the expression of R_i (Figure 6.5H).

Steps F-G are repeated for every potential regulator R_i. By aggregating all of the feature importance scores over all the samples, regulators and targets, we obtain an M-by-N-by-P tensor[2].

A moderately-sized dataset could contain $M = 10'000$ samples, $N = 2'000$ regulators, and $P = 10'000$ target genes. Due to memory constraints, only interactions with an average importance value (across all samples) higher than a minimum threshold are retained.

To compute the case-wise GRNs, we implemented the abovementioned methodology in C++ in a modified version of the `ranger` R/C++ package [21].

6.4.2 Predicting the effect of an interaction

To predict the effect of a potential regulator R_i on a target gene T for a given tree, the Pearson correlation is calculated between the difference in regulator expression (before and after shuffling the values), and the difference in target expression prediction.

[2]This is the origin of the name of the method, "`bred`".

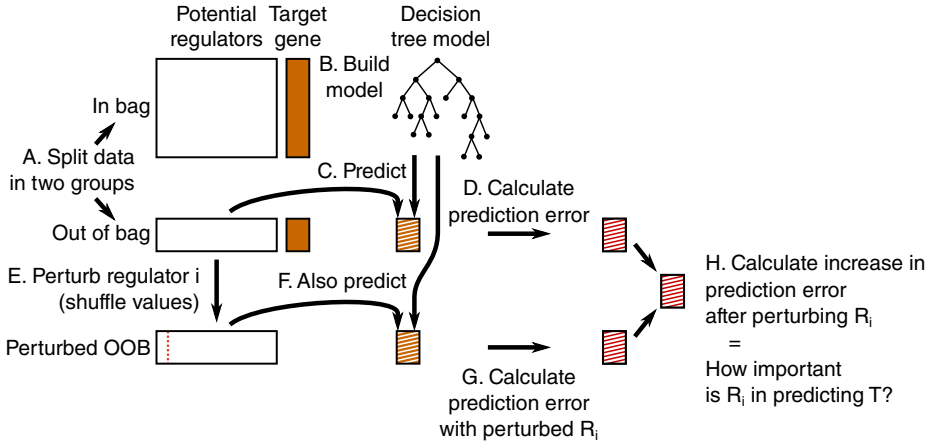


Figure 6.5: Calculating the feature importance score for one decision tree and one target consists of 8 distinct steps. A: Randomly split the data into two groups, the in-bag data and the out-of-bag data. **B:** The in-bag data is used to train a decision tree to try to predict the expression of the target gene from the expression values of the regulators. **C:** The decision tree is used to predict the gene expression of the target gene of the out-of-bag samples. **D:** Sample-specific squared error values are computed. **E:** Repeat steps E-H for every regulator R_i. Perturb the expression of regulator R_i in the out-of-bag samples. **F:** Again predict the gene expression of the target gene with the perturbed expression values. **G:** Again compute the sample-specific squared error values. **H:** The difference between the prediction error on the perturbed dataset versus the prediction error on the unperturbed is the importance in R_i in predicting T

$$\begin{aligned}\text{effect}(R_i \to T) &= \text{cor}(x, y),\\ \text{with } x &= \text{expr\_shuffled}[:, R_i] - \text{expr}[:, R_i],\\ \text{and } y &= \text{predict}(\text{tree}, \text{expr\_shuffled}) - \text{predict}(\text{tree}, \text{expr}).\end{aligned}$$

The Pearson correlation between two variables x and y is usually defined as shown in Equation 6.1. Computing r_{xy} for each (regulator, target) pairs, across all trees, would require storing large amounts of data.

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x}) \times (y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2} \times \sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{6.1}$$

However, by rearranging the formula, it can be defined as Equation 6.2.

$$r_{xy} = \frac{\sum(x_i \times y_i) - \sum x \times \sum y/n}{\sqrt{(\sum x_i^2 - (\sum x)^2/n)} \times \sqrt{(\sum y_i^2 - (\sum y)^2/n)}} \tag{6.2}$$

For every regulator R_i during a perturbation in a given tree, only 6 values need to be stored, namely $A = \sum x_i$, $B = \sum y_i$, $C = n$, $D = \sum x_i \times y_i$, $E = \sum x_i \times x_i$, and

$F = \sum y_i \times y_i$.

For every (regulator, target) pair, these values are summed, and the r_{xy} is calculated as shown in Equation 6.3.

$$r_{xy} = \frac{D - A \times B/C}{\sqrt{(E - A^2/C)} \times \sqrt{(F - B^2/C)}} \tag{6.3}$$

The following cutoffs were used to determine the effect.

- Strong negative: $r_{xy} < -0.4$
- Weak negative: $-0.4 \leq r_{xy} < -0.2$
- Unclear: $-0.2 \leq r_{xy} \leq 0.2$
- Weak positive: $0.2 < r_{xy} \leq 0.4$
- Strong positive: $0.4 < r_{xy}$

6.4.3 Clustering of case-wise GRNs

To perform downstream analysis on the cases, first a k-nearest neighbour (kNN) graph of the cases is computed. In order for the kNN graph to better emphasise similarities in GRNs rather than absolute euclidean distances, we first reduce the dimensionality of the case-by-interaction matrix to case-by-20 matrix using Landmark Multi-Dimensional Scaling [22] with a Spearman rank distance metric.

Next, KD-trees are used to calculate the kNN graph efficiently. The cases in the dataset are visualised and clustered using the Fruchterman-Reingold [15] and Louvain clustering [16], respectively.

The following R packages provided implementations for each of these algorithms: lmds, RANN, igraph [23].

6.4.4 Visualising clustered GRNs

After Louvain clustering, the interactions of the 50 interactions with highest mean importance per cluster are retained. These interactions are visualised in Cytoscape [24], in which nodes depict genes, edges depict predicted regulatory interactions, coloured according to which cluster they are predicted for. The shape of the arrow denotes the predicted effect of the regulatory interaction.

6.5 Supplementary information

Table 6.1: Prioritisation of genes, ranked according to page-rank value. The majority of these genes are already known to be involved in oncogenesis (data not shown).

Cluster name	Top 4 genes, page rank
Acute Leukemia	HSPA1A, HSPA1B, HOXA10, HOXA9
ALL	DDX3Y, XIST, RN7SL128P, TXLNGY
BL	KIAA0226L, XIST, TCL1A, NUGGC
BRCA	LINC00993, IRX5, ATP8A2P1, ENSG00000234918
Connective and Soft Tissue Disorder *	HOXA11, HOXC9, HOXA10, HOXC10
CPTAC #1	SLC22A2, CUBN, XIST, UMOD
CPTAC #2	RN7SL752P, SCARNA13, XIST, RNY1
Diffuse Large B-Cell Lymphoma	PLA2G2D, ENSG00000224137, CCL25, MS4A1
Digestive System Disorder	MEP1A, LGALS4, MUC13, CDX1
Glioma	NCAN, GPM6A, MBP, OLIG2
Kidney Carcinoma	ACSM2A, SLC22A2, NAT8, UMOD
LIHC	AGXT, ITIH3, ITIH1, HRG
LUAD	SFTPA2, SFTPD, NAPSA, SFTA2
Melanocytic Neoplasm	MLANA, RPS4Y1, PMEL, TYR
MM	CD96, GPRC5D, FGFR3, IGLV1-41
Peripheral Nervous System Disorder	TH, DBH, PENK, HAND2
PRAD	TRGC1, NKX3-1, ACPP, RPL7P16
Squamous Cell Carcinoma	TP63, SPRR2F, RPS4Y1, CDKN2B
THCA	ENSG00000240237, TG, TPO, RPS4Y1
Transitional Cell Carcinoma	DHRS2, UPK2, PADI3, VGLL1
UCEC	DLX5, MSX1, HOXB5, HOXB8

Hendrik, as the one who I always looked up to, I blame you for getting me interested in computers. With you taking computers apart on the table, how could I resist.

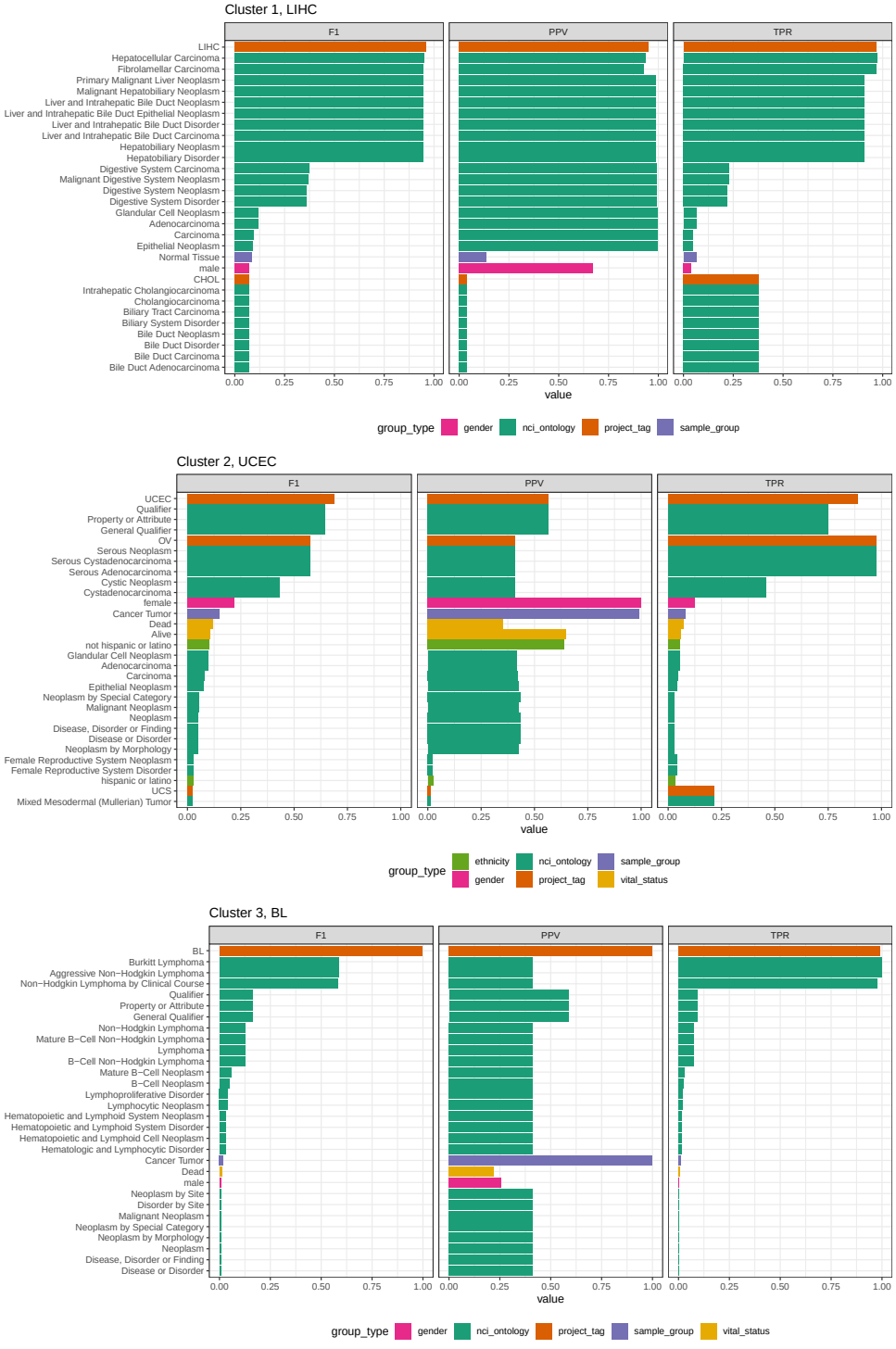


Figure 6.6: Prioritisation of NCI terms and other forms of meta-data used to annotate clusters 1–3. Cluster 2 is mainly made up of disorders in female reproductive organs, and should have been labelled as such.

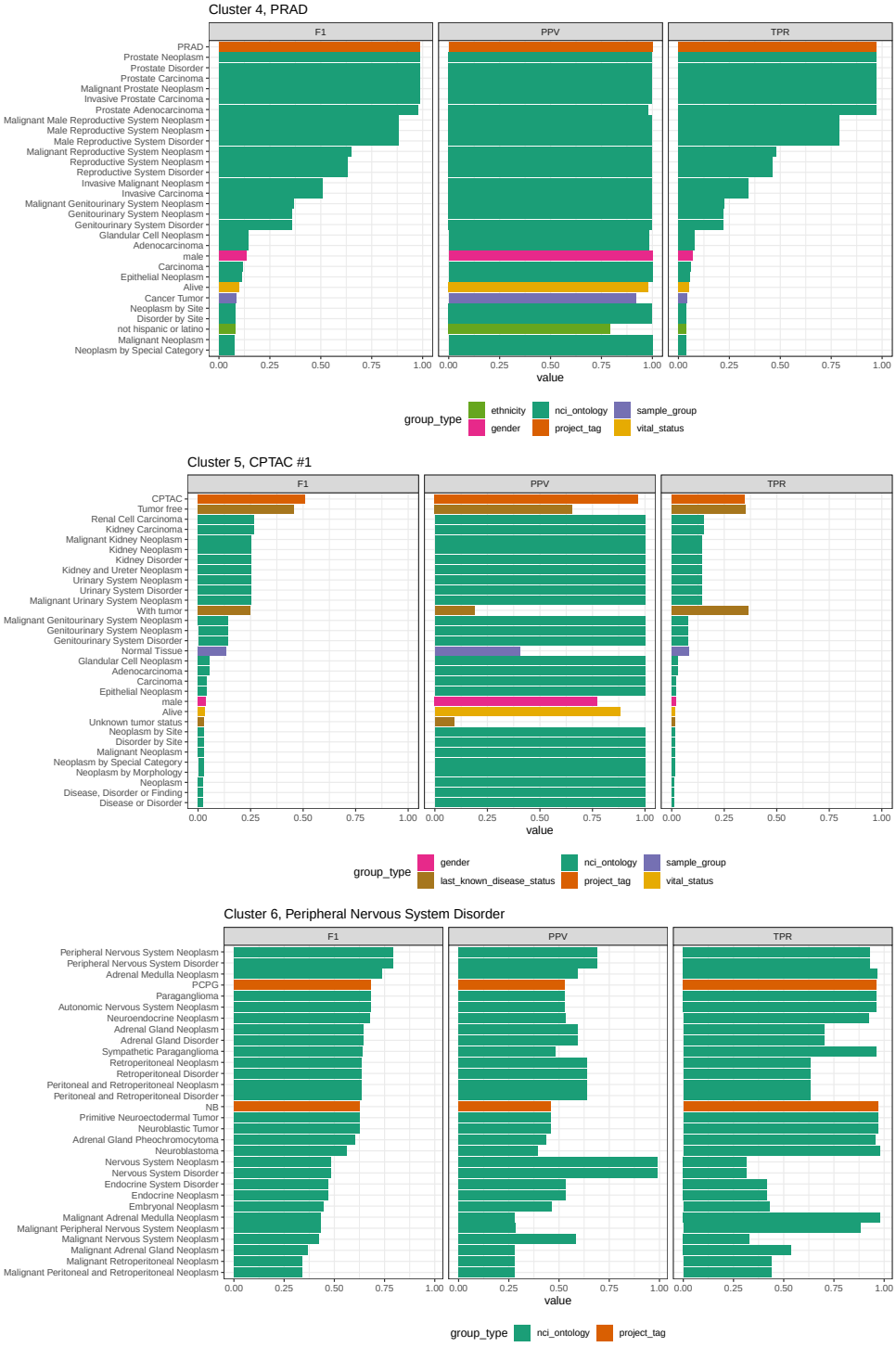


Figure 6.7: Prioritisation of NCI terms and other forms of meta-data used to annotate clusters 4–6. Cluster 5 is mainly made up of renal cell carcinoma, and should have been labelled as such. Cluster 6 is aptly named, but mainly consists of neuroblastoma (NB), pheochromocytoma (PC), and paraganglioma (PG) disorders.

Figure 6.8: Prioritisation of NCI terms and other forms of meta-data used to annotate clusters 7–9.

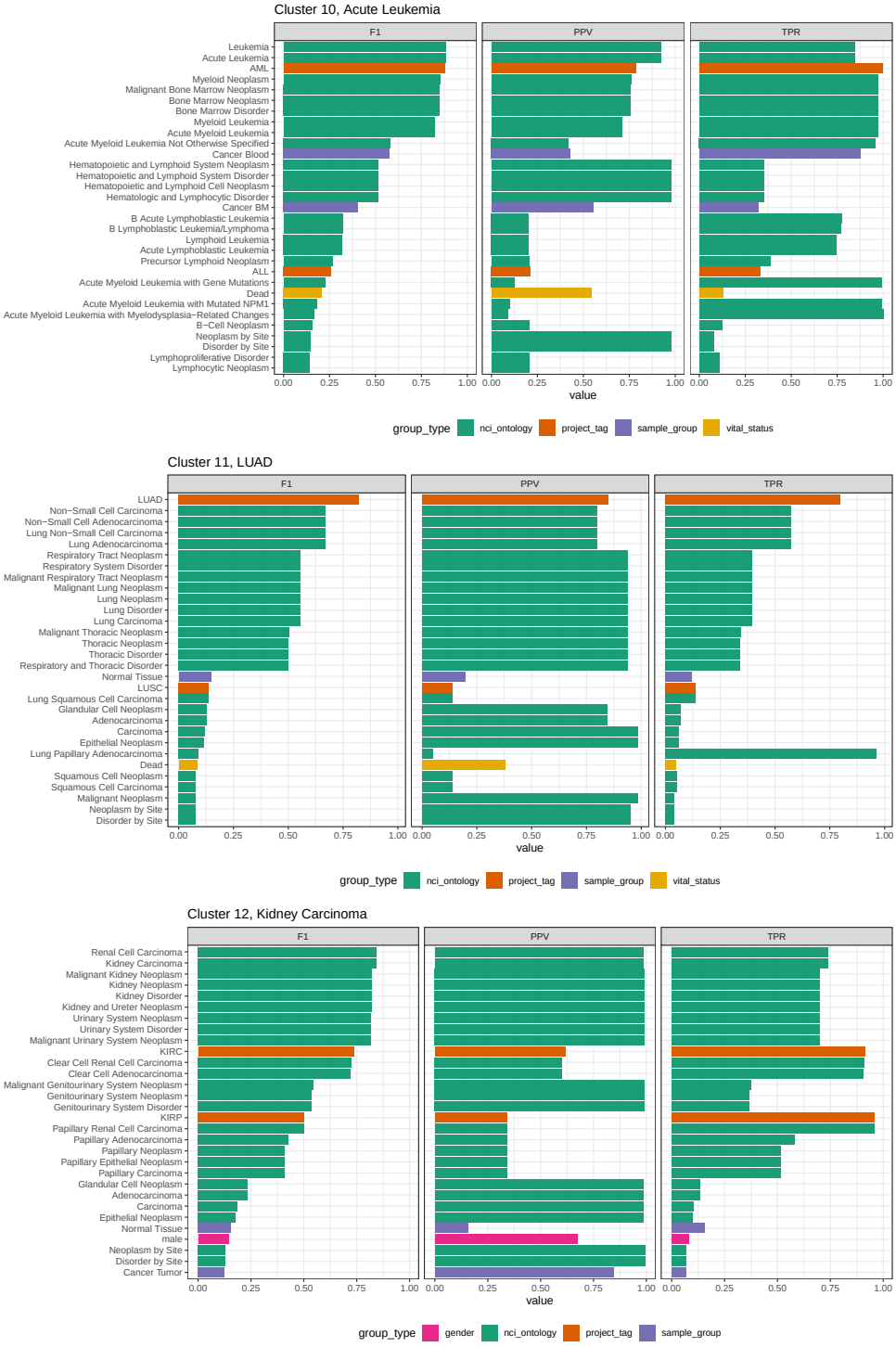


Figure 6.9: Prioritisation of NCI terms and other forms of meta-data used to annotate clusters 10–12. Cluster 11 contains mostly Lung Adenocarcinoma (LUAD), but also Lung Squamous Cell Carcinoma (LUSC), and thus should have been labelled Lung Carcinoma instead.

Go to the next chapter for more acknowledgements.

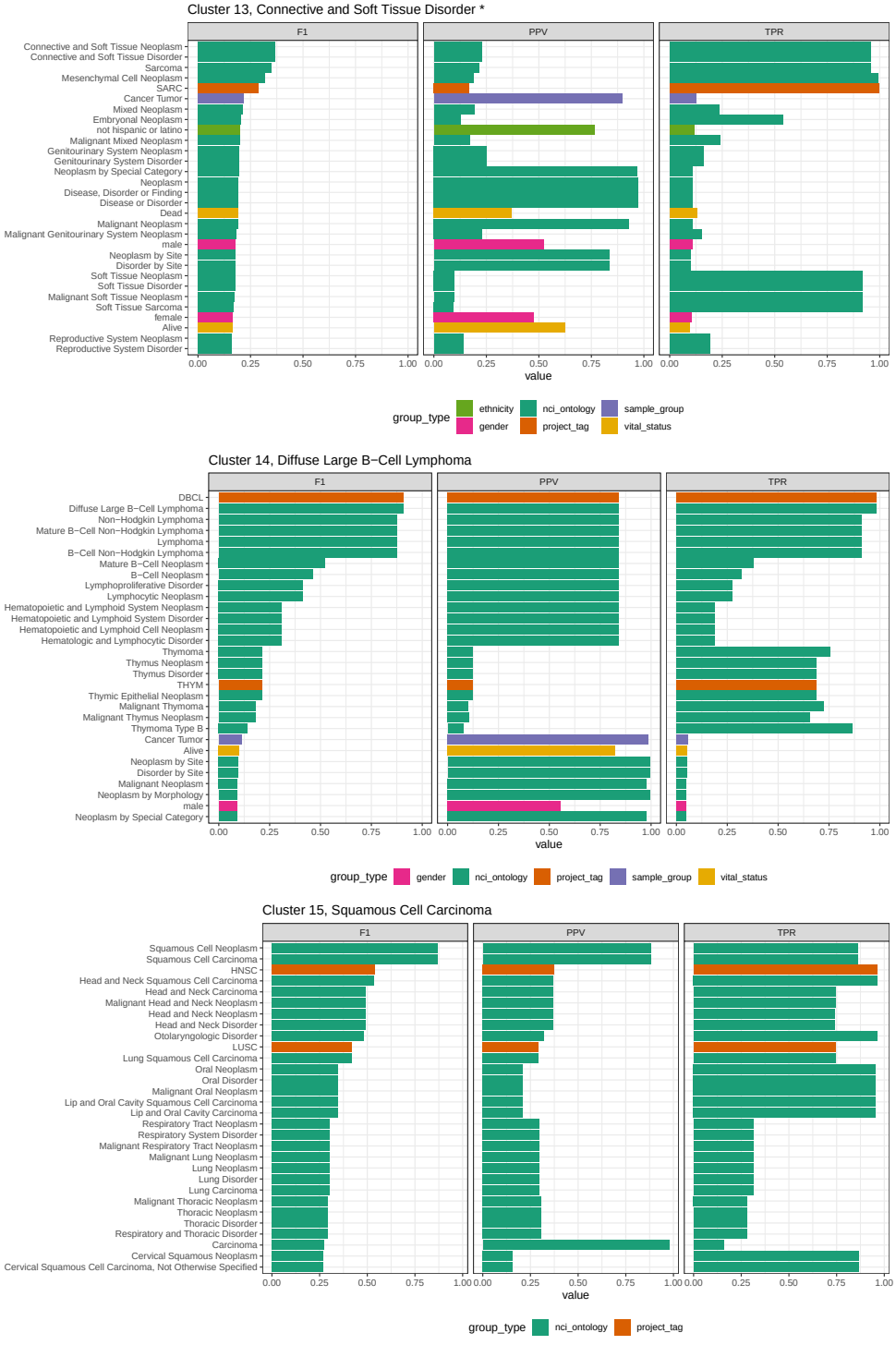


Figure 6.10: Prioritisation of NCI terms and other forms of meta-data used to annotate clusters 13–15. Cluster 13 contains a mixture of cancer types, and should have been labelled 'Mixed'.

Figure 6.11: Prioritisation of NCI terms and other forms of meta-data used to annotate clusters 16–18.

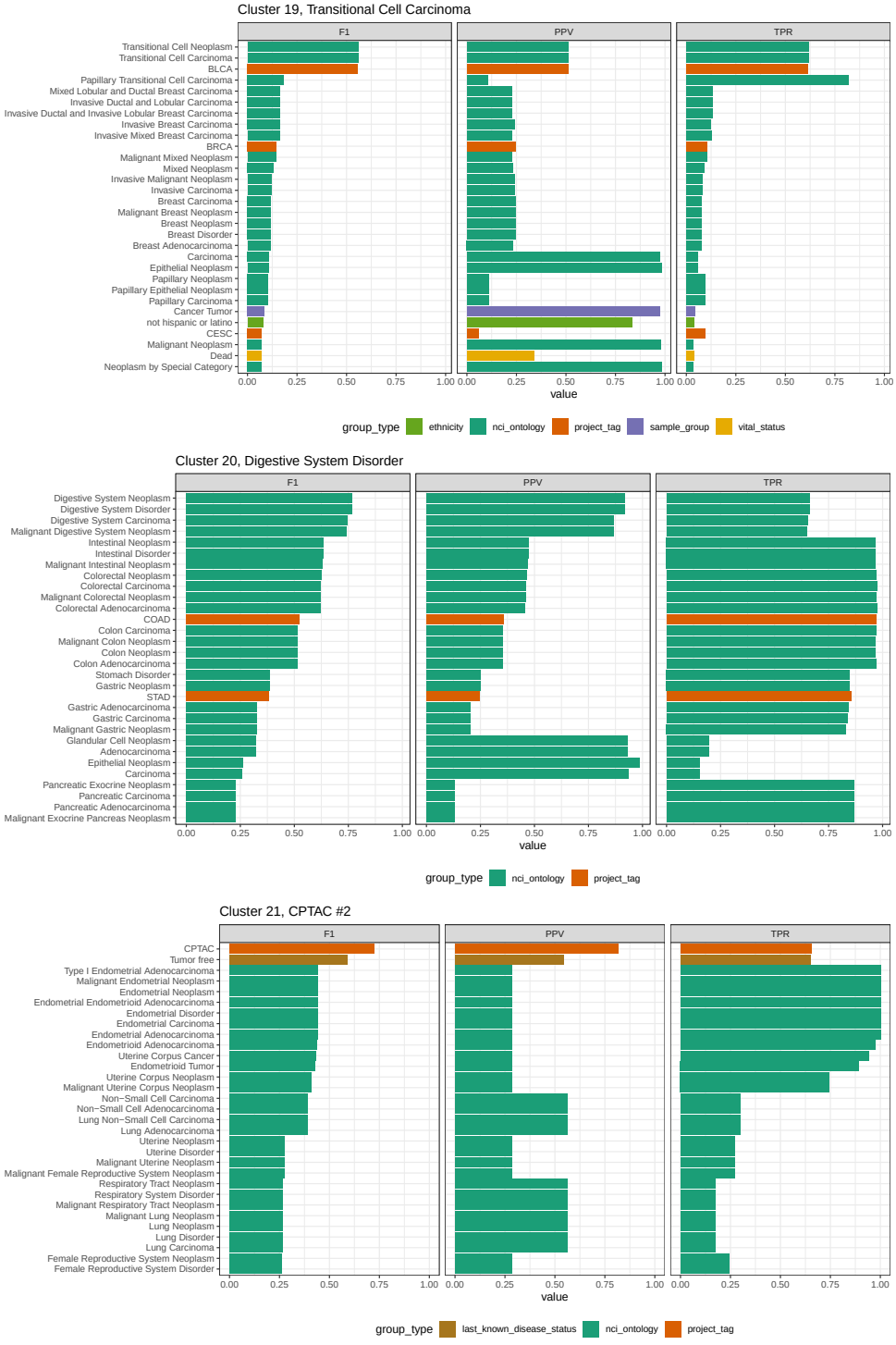
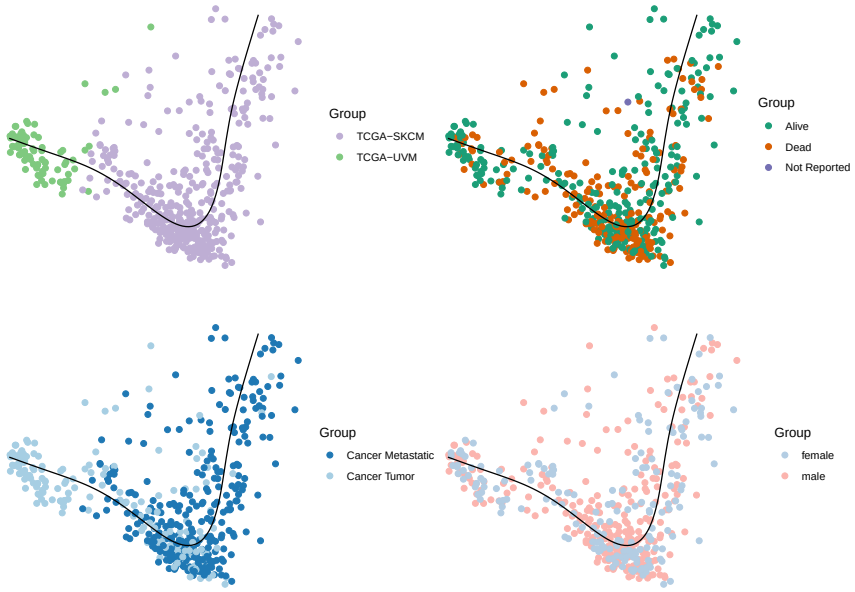


Figure 6.12: Prioritisation of NCI terms and other forms of meta-data used to annotate clusters 19–21. Cluster 21 contains a mixture of cancer types, mostly lung carcinoma and endometrial carcinoma.

6.5.1 Melanocytic neoplasm

A



B

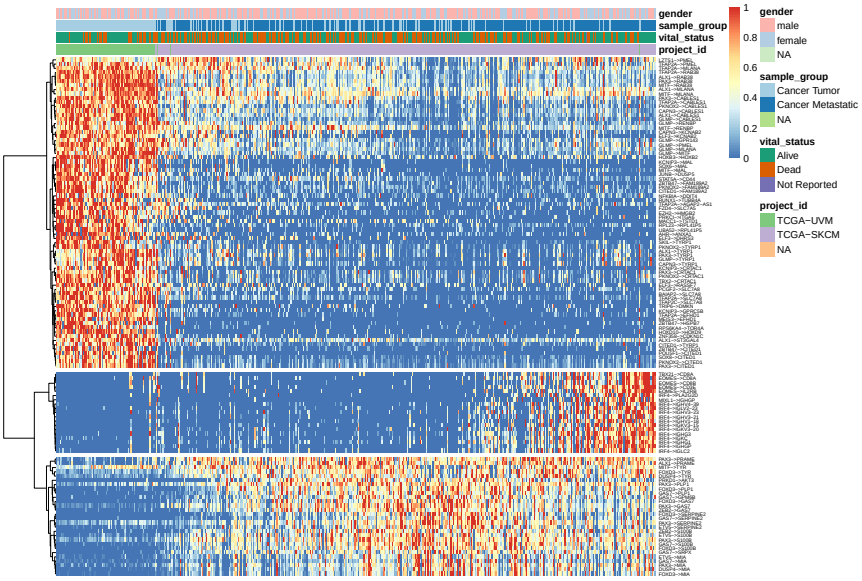


Figure 6.13: In-depth view of cluster 9, melanocytic neoplasm. A: A dimensionality reduction of the samples, coloured according to multiple sources of meta-data. **B:** The samples were ordered linearly with SCORPIUS (see trajectory in **A**) in order to visualise regulome activity in the form of a heatmap.

6.5.2 Kidney carcinoma

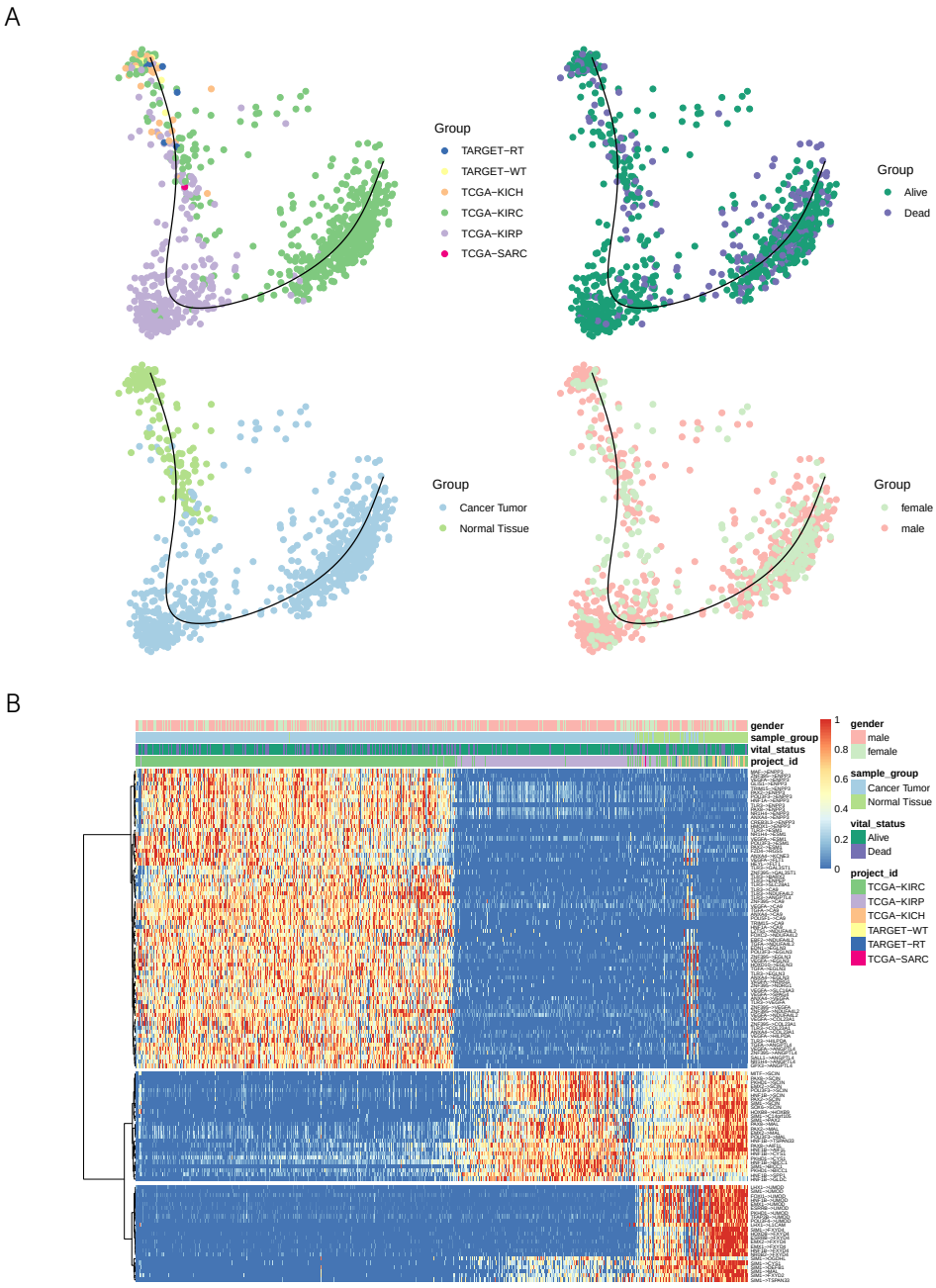


Figure 6.14: In-depth view of cluster 12, kidney carcinoma. **A:** A dimensionality reduction of the samples, coloured according to multiple sources of meta-data. **B:** The samples were ordered linearly with SCORPIUS (see trajectory in **A**) in order to visualise regulome activity in the form of a heatmap.

6.6 References

[1] Ziv Bar-Joseph, Anthony Gitter, and Itamar Simon. "Studying and Modelling Dynamic Biological Processes Using Time-Series Gene Expression Data". In: *Nat. Rev. Genet.* 13.8 (Aug. 2012), pp. 552–564.

[2] Noa Novershtern et al. "Densely Interconnected Transcriptional Circuits Control Cell States in Human Hematopoiesis". In: *Cell* 144.2 (2011), pp. 296–309.

[3] Gillian May et al. "Dynamic Analysis of Gene Expression and Genome-Wide Transcription Factor Binding during Lineage Specification of Multipotent Progenitors". In: *Cell Stem Cell* 13.6 (2013), pp. 754–768.

[4] Vladimir Jojic et al. "Identification of Transcriptional Regulators in the Mouse Immune System". In: *Nat. Immunol.* 14.6 (2013), pp. 633–643. DOI: 10.1038/ni.2587.Identification.

[5] Debbie K Goode et al. "Dynamic Gene Regulatory Networks Drive Hematopoietic Specification and Differentiation". In: *Dev. Cell* 36.5 (2016), pp. 572–587.

[6] Victoria Moignard et al. "Characterization of Transcriptional Networks in Blood Stem and Progenitor Cells Using High-Throughput Single-Cell Gene Expression Analysis". In: *Nat. Cell Biol.* 15.4 (Apr. 2013), pp. 363–372.

[7] Cristina Pina et al. "Single-Cell Network Analysis Identifies DDIT3 as a Nodal Lineage Regulator in Hematopoiesis.". In: *Cell reports* 11.10 (2015), pp. 1503–1510. ISSN: 2211-1247. DOI: 10.1016/j.celrep.2015.05.016. pmid: 26051941.

[8] Balázs Papp and Stephen Oliver. "Genome-Wide Analysis of the Context-Dependence of Regulatory Networks". In: *Genome Biology* 6.2 (Jan. 27, 2005), p. 206. ISSN: 1474-760X. DOI: 10.1186/gb-2005-6-2-206.

[9] Marieke Lydia Kuijjer et al. "Estimating Sample-Specific Regulatory Networks". In: *iScience* 14 (Mar. 28, 2019), pp. 226–240. ISSN: 2589-0042. DOI: 10.1016/j.isci.2019.03.021. pmid: 30981959.

[10] Xiaoping Liu et al. "Personalized Characterization of Diseases Using Sample-Specific Networks". In: *Nucleic Acids Research* 44.22 (2016), e164–e164. ISSN: 0305-1048. DOI: 10.1093/nar/gkw772. pmid: 27596597.

[11] Sara Aibar et al. "SCENIC: Single-Cell Regulatory Network Inference and Clustering". In: *Nature Methods* (Oct. 2017). ISSN: 1548-7091. DOI: 10.1038/nmeth.4463.

[12] Vân Anh Huynh-Thu et al. "Inferring Regulatory Networks from Expression Data Using Tree-Based Methods". In: *PLoS ONE* 5.9 (Jan. 2010), e12776. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0012776. pmid: 20927193.

[13] Leo Breiman. "Random Forests". In: *Machine Learning* 45 (2001), pp. 5–32.

[14] John N Weinstein et al. "The Cancer Genome Atlas Pan-Cancer Analysis Project.". In: *Nature genetics* 45.10 (Oct. 2013), pp. 1113–20. ISSN: 1546-1718. DOI: 10.1038/ng.2764. pmid: 24071849.

[15] Thomas M. J. Fruchterman and Edward M. Reingold. "Graph Drawing by Force-Directed Placement". In: *Software: Practice and Experience* 21.11 (1991), pp. 1129–1164. ISSN: 1097-024X. DOI: 10.1002/spe.4380211102.

[16] Vincent D Blondel et al. "Fast Unfolding of Communities in Large Networks". In: *Journal of Statistical Mechanics: Theory and Experiment* 2008.10 (Oct. 9, 2008), P10008. ISSN: 1742-5468. DOI: 10.1088/1742-5468/2008/10/p10008.

[17] Nicholas Sioutos et al. "NCI Thesaurus: A Semantic Model Integrating Cancer-Related Clinical and Molecular Information". In: *Journal of Biomedical Informatics*. Bio*Medical Informatics 40.1 (Feb. 1, 2007), pp. 30–43. ISSN: 1532-0464. DOI: 10.1016/j.jbi.2006.02.013.

[18] Nicholas Schaum et al. "Single-Cell Transcriptomics of 20 Mouse Organs Creates a Tabula Muris". In: *Nature* 562.7727 (Oct. 2018), pp. 367–372. ISSN: 1476-4687. DOI: 10.1038/s41586-018-0590-4.

[19] Robrecht Cannoodt, Wouter Saelens, and Yvan Saeys. "Dyngen: Benchmarking with *in Silico* Single Cells". In: *In preparation* (Sept. 2019).

[20] L Breiman et al. *Classification and Regression Trees*. Wadsworth Publishing Company, 1984.

[21] Marvin N Wright and Andreas Ziegler. "Ranger: A Fast Implementation of Random Forests for High Dimensional Data in C++ and R". In: *Journal of Statistical Software* 77.1 (Mar. 2017). DOI: 10.18637/jss.v077.i01.

[22] Seunghak Lee and Seungjin Choi. "Landmark MDS Ensemble". In: *Pattern Recognition* 42.9 (Sept. 2009), pp. 2045–2053. ISSN: 00313203. DOI: 10.1016/j.patcog.2008.11.039.

[23] Gabor Csardi and Tamas Nepusz. "The Igraph Software Package for Complex Network Research". In: *InterJournal, Complex Systems* 1695.5 (2006), pp. 1–9.

[24] Paul Shannon et al. "Cytoscape: A Software Environment for Integrated Models of Biomolecular Interaction Networks". In: *Genome Research* 13.11 (Nov. 1, 2003), pp. 2498–2504. DOI: 10.1101/gr.1239303.

7 | incgraph: Optimising regulatory networks

Abstract

Purpose: Graphlets are small network patterns that can be counted in order to characterise the topology of a network. Counting graphlet can used to iteratively optimise a network in order to achieve certain topological properties. Hitherto graphlets were not suited as a metric for performing topology optimisation due to the computational complexity of recalculating all the graphlet counts for each of the edge modifications.

Results: IncGraph calculates the differences in graphlet counts with respect to the network in its previous state, thereby reducing execution time by several orders of magnitude when compared to classic approaches. We demonstrate the usefulness of this approach optimising gene regulatory networks using a graphlet-based fitness score.

Conclusion: IncGraph is able to quickly quantify the topological impact of small changes to a network, which opens novel research opportunities to study changes in topologies in evolving networks, or develop graphlet-based criteria for topology optimisation.

Publication status

Published in PLOS ONE 13, 4 (2018). doi:10.1371/journal.pone.0195997.
Cannoodt R, Ruyssinck J, Ramon J, De Preter K, and Saeys Y.

Author contributions

- **R.C.**, Jo.R., Ja.R., and Y.S. designed the study.
- **R.C.**, Jo.R., and Ja.R. performed the experiments and analysed the data.
- **R.C.** implemented the incgraph software package.
- **R.C.** wrote the original manuscript.
- **R.C.**, Jo.R., Ja.R., K.D.P., and Y.S. reviewed and edited the manuscript.
- Jo.R., K.D.P. and Y.S. supervised the project.

7.1 Introduction

Even the simplest of living organisms already consist of complex biochemical networks which must be able to respond to a variety of stressful conditions in order to survive. An organism can be characterised using numerous interaction networks, including gene regulation, metabolic, signalling, and protein-protein interaction. The advent of high-throughput profiling methods (e.g. microarrays and RNA sequencing) have allowed to observe the molecular contents of a cell, which in turn have enabled computational network inference methods to reverse engineer the biochemical interaction networks [1]. Improving the accuracy of inferred networks has been a long-standing challenge, but the development of ever more sophisticated algorithms and community-wide benchmarking studies have resulted in significant process [2, 3, 4, 5].

Several recent developments involve incorporating topological priors, either to guide the inference process [6] or post-process the network [7]. A common prior is that biological networks are highly modular [8], as they consist of multiple collections of functionally or physically linked molecules [9, 10]. At the lowest level, each module is made up out of biochemical interactions arranged in small topological patterns, which act as fundamental building blocks [11].

Graphlets [12] are one of the tools which could be used to add a topological prior to a biological network. Graphlets are small connected subnetworks which can be counted to identify which low-level topological patterns are present in a network. By comparing how topologically similar a predicted network is to what would be expected of a true biological network, a predicted network can be optimised in order to obtain a better topology.

By counting the number of occurrences of each of the different graphlets (Fig 7.1A) touching a specific node, one can characterise the topology surrounding it. The graphlet counts of a network can be represented as a matrix with one row for each of the nodes and one column for each of the graphets (Fig 7.1B). An orbit represents a class of isomorphic (i.e. resulting in the same structure) positions of nodes within a graphlet (Fig 7.1A, coloured in red). Both graphlets and orbits have been used extensively for predicting the properties of nodes such as protein functionality [13, 14, 15] and gene oncogenicity [16], by performing network alignment [17, 18] or using them as a similarity measure in machine learning tasks [19, 20].

In this work, we focus on optimising gene regulatory networks by incorporating a topological prior as part of a topology optimisation process. We seek to optimise a predicted network by iteratively modifying the network and accepting modifications that lead to topological properties that resemble better those of true biological net-

 To Amor, a big thank you for contributing countless cuddles and endless drool.

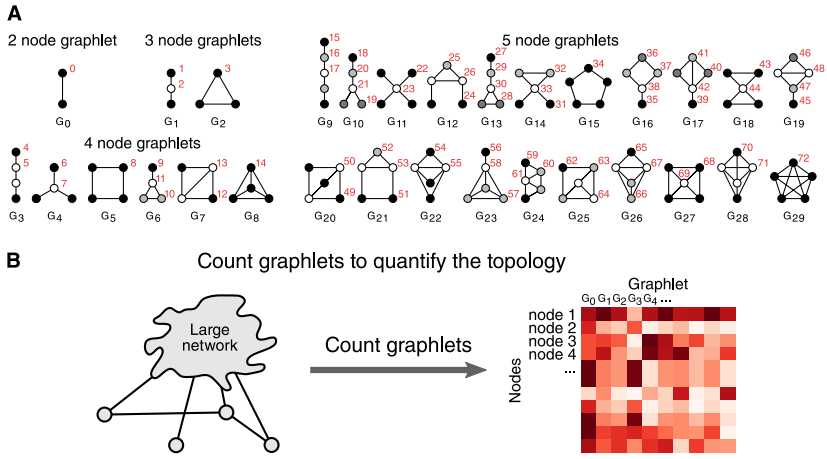


Figure 7.1: Graphlet counting in a network characterises its local topologies. (A) In total, there are 30 different graphlets containing 2 to 5 nodes, ranging from G_0 to G_{29}. Orbits are an extension of graphlets which also take into account the position of a node within a graphlet. The 73 different orbits are coloured in red. (B) By counting the occurrences of these graphlets in the network, the local topology surrounding a node can be quantified.

works.

However, using graphlets to perform topology optimisation was hitherto not possible. Calculating the graphlet counts using the most state-of-the-art graphlet counting of a moderately sized gene regulatory network already has an execution time of about five seconds (*E. coli*, ~ 3000 genes, ~ 10000 interactions, up to graphlets up to 5 nodes). While this computational time poses no issue for regular static networks, recalculating all graphlet counts for every network modification made as part of a topology optimisation is computationally intractable. For example, performing 100'000 iterations of topology optimisation on a similarly sized network and calculating the topological impact of 10 possible edge modification at each iteration, already results in a computational time of about 12 days. Graphlet-based topology optimisation thus quickly becomes infeasible for larger networks.

When adding or removing an edge to a large network, the number of altered graphlets is much smaller than the total number of graphlets in the network. It could therefore be much more efficient to iterate over and count all the graphlets that have been added or removed as a result of the edge modification, than it is to recalculate the graphlet counts from scratch.

Eppstein et al. introduced data structures and algorithms for updating the counts of size-three [21] and size-four [22] subgraphs in a dynamic setting. The data structures were determined such that the amortised time is $O(h)$ and $O(h^2)$, respectively, where h is the h-index of the network [23].

We developed IncGraph, an alternative algorithm and implementation for performing incremental counting of graphlets up to size five. We show empirically that IncGraph is several orders of magnitude faster at calculating the differences in graphlet counts in comparison to non-incremental counting approaches. In addition, we demonstrate the practical applicability by developing a graphlet-based optimisation criterion for biological networks.

7.2 Materials and methods

Assume a network G of which the graphlet counts C_G are known. C_G is an n-by-m matrix, with n the number of vertices in the network, $m = 73$ is the number of different orbits, and where $C_G[i, j]$ is the number of times node i is part of a graphlet at orbit O_j. Further assume that one edge has either been added or removed from G, resulting in G', at which point the counts $C_{G'}$ need to be observed. If multiple edges have been modified, the method described below can be repeated for each edge individually.

7.2.1 Incremental graphlet counting

As stated earlier, recalculating the graphlet counts for each modification made to the network quickly becomes computationally intractable for larger network sizes. However, as the differences in topology between G and G' are small, it is instead possible to calculate the differences in graphlet counts $\Delta_{G,G'}$ instead. This is much more efficient to calculate, as only the neighbourhood of the modified edges needs to be explored. $C_{G'}$ can thus be calculated as $C_{G'} = C_G + \Delta_{G,G'}$.

The differences in graphlet counts $\Delta_{G,G'}$ are calculated by iterating recursively over the neighbours surrounding each of the modified edges (Fig 7.8). Several strategies are used in order to calculate $\Delta_{G,G'}$ as efficiently as possible (Fig 7.2). (A) The delta matrix is calculated separately for each modified edge. Since the edge already contains two out of five nodes and any modified graphlet is a connected subgraph, the neighbourhood of this edge only needs to be explored up to depth 3 in order to iterate over all modified graphlets. (B) We propose a lookup table to look up the graphlet index of each node of a given subgraph. By weighting each possible edge in a 5-node graphlet, the sum of the edges of a subgraph can be used to easily look up the corresponding graphlet index. (C) During the recursive iteration of the neighbourhood, the same combination of nodes is never visited twice. (D) As the network can be relatively large, the adjacency matrix is binary compressed in order to save memory. One integer requires 4 bytes and contains the adjacency boolean values of 32 edges, whereas otherwise 32 booleans would require 32 bytes. For example, 1GB of memory is large

Hello Petit Remi! You're only seven months old,
but it feels like I've known you for much longer already.

enough to store a compressed adjacency matrix of 92681 nodes. If the network contains too many nodes to fit a compressed adjacency matrix into the memory, a list of sets containing each node's neighbours is used instead.

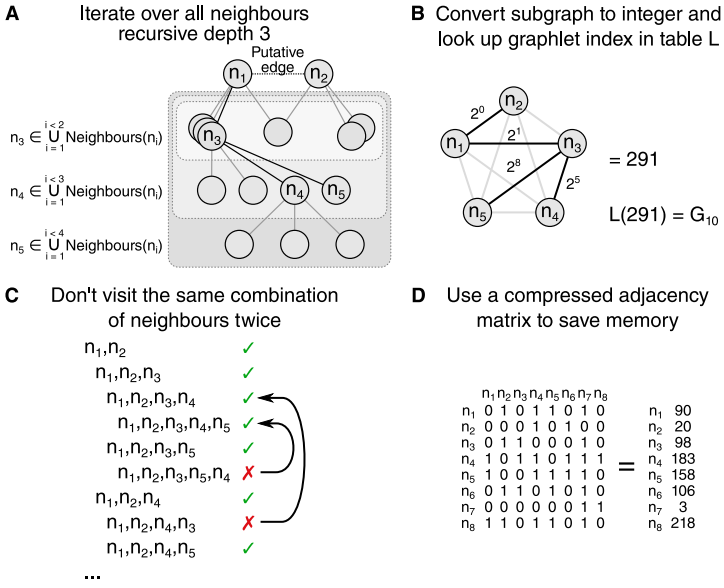


Figure 7.2: Several strategies are employed in order to reduce time and memory consumption. (A) Only the depth 3 neighbourhood of each modified edge needs to be explored in order to have visited all modified five-node graphlets. (B) A lookup table can be used to easily look up the graphlet index of a subgraph, by weighing each edge in a 5-node subgraph by a power of 2. (C) The same combination of five nodes is never repeated, as to avoid counting the same graphlet multiple times. (D) The adjacency matrix of the network is compressed in order to reduce memory usage.

IncGraph supports counting graphlets and orbits of subgraphs up to five nodes in undirected networks. By modifying the lookup table, the method can be easily extended to directed graphlets or larger-node graphlets, or limited to only a selection of graphlets. This allows for variations of the typical graphlets to be studied in an incremental setting.

7.2.2 Timing experiments

We compared the execution time needed to calculate the graphlet counts in iteratively modified networks between our method and a state-of-the-art non-incremental algorithm, Orca [24]. Orca is a heavily optimised algorithm for counting 5-node graphlets in static networks, and outperforms all other static graphlet counting algorithms by an order of magnitude [24].

The timing experiments were performed by generating networks from 3 different net-

work models, making modifications to those networks while still adhering to the network model, and measuring the execution times taken for both approaches to calculate the new graphlet counts (Fig 7.3).

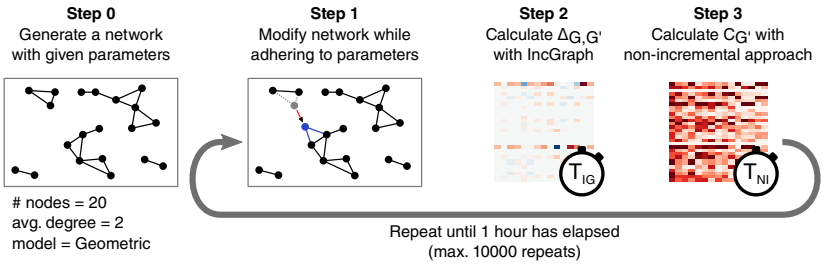


Figure 7.3: Static network model generators were modified to generate dynamic networks. Three network models were used: Barabási–Albert, Erdős–Rényi, and Geometric. Step 0: a network is generated according to the network model and the given parameters. Step 1: the network is modified such that the result is as likely to have been generated by the network model. Step 2: The time for IncGraph to calculate the differences in graphlet counts is measured (T_{IG}). Step 3: The time for the non-incremental approach to calculate the graphlet counts of the new network is measured (T_{NI}). Steps 1 to 3 are repeated until all modifications generated at step 0 are processed, or until an execution time threshold has been reached.

The network models were based on three static network models: Barabási-Albert [25], Erdős–Rényi [26], and Geometric [27]. These models were adapted to generate evolving networks instead (Figs 7.9, 7.10, and 7.11). Each model generates an initial network according to the static network model it is based on, and a list of network modifications (removing an edge from or adding an edge to the network). These network modifications are made such that at any given time point in the evolving network, it is likely that the network at its current state could have been generated by the original static network model.

Networks were generated with varying network models, between 1000 and 16000 nodes, node degrees between 2 and 64, and 10000 time points. We measured the time needed to calculate the delta matrix at random time points until 1 hour has passed. All timings experiments were carried out on Intel(R) Xeon(R) CPU E5-2665 @ 2.40GHz processors, with one thread per processor. The generation of networks with higher node counts or degrees was constrained by the execution time of the network generators, not by IncGraph. All data and scripts are made available at github.com/rcannood/incgraph-scripts.

7.2.3 Gene regulatory network optimisation experiments

We demonstrate the usefulness of IncGraph by using a simple graphlet-based metric to optimise gene regulatory networks. One of the striking differences between real and predicted gene regulatory networks is that the predicted networks contain highly

connected subnetworks, which contain high amounts of false positives. We determine a penalty score such that predicted networks containing graphlets with many redundant edges will be penalised in comparison to very sparse networks.

The *redundancy penalty* (Fig 7.4A) of a network is defined as the sum of occurrences of each graphlet multiplied by the redundancy associated with each individual graphlet. The redundancy of a graphlet is the number of edges that can be removed without disconnecting the nodes from one another. By using the redundancy penalty score, we aim to improve the gene regulatory network (Fig 7.4B).

The topology optimisation procedure uses an empty network as initialisation and grows the network by selecting interactions iteratively. Each iteration, the top $k = 100$ highest ranked interactions that are not currently part of the network are evaluated, and the highest ranked interaction passing the redundancy criterion is selected (Fig 7.4C). This procedure is repeated until a predefined amount of time has passed. As the aim of this experiment is not to obtain the highest performing topology optimisation method, parameter optimisation of k has not been performed and is considered to be outside the scope of this work.

We optimised gene regulatory networks of *E. coli* and *S. cerevisiae*. The predicted networks were generated using the network inference method GENIE3 [28] with default parameters. Gene expression data was obtained from COLOMBOS [29] and GEO [30], respectively. The predicted networks and the optimised versions thereof were compared against respective lists of known gene regulatory interactions [31, 32].

7.3 Results and discussion

The contributions of this work are twofold. Firstly, we propose a new method for incrementally calculating the differences in graphlet counts in changing graphs, and show that it is orders of magnitude faster than non-incremental approaches. Secondly, we demonstrate its applicability by optimising a predicted gene regulatory network in order to reduce the false positive rate therein.

7.3.1 Execution time is reduced by orders of magnitude

Timing experiments show that IncGraph is significantly faster in calculating the delta matrix in comparison to calculating the graphlet counts from scratch at each iteration (Fig 7.5). The observed speedup ratios between IncGraph and the non-incremental approach Orca ranges from about 50× to 10000×. The speedup ratio increases as the network size increases. For larger networks, IncGraph can thus calculate the delta ma-

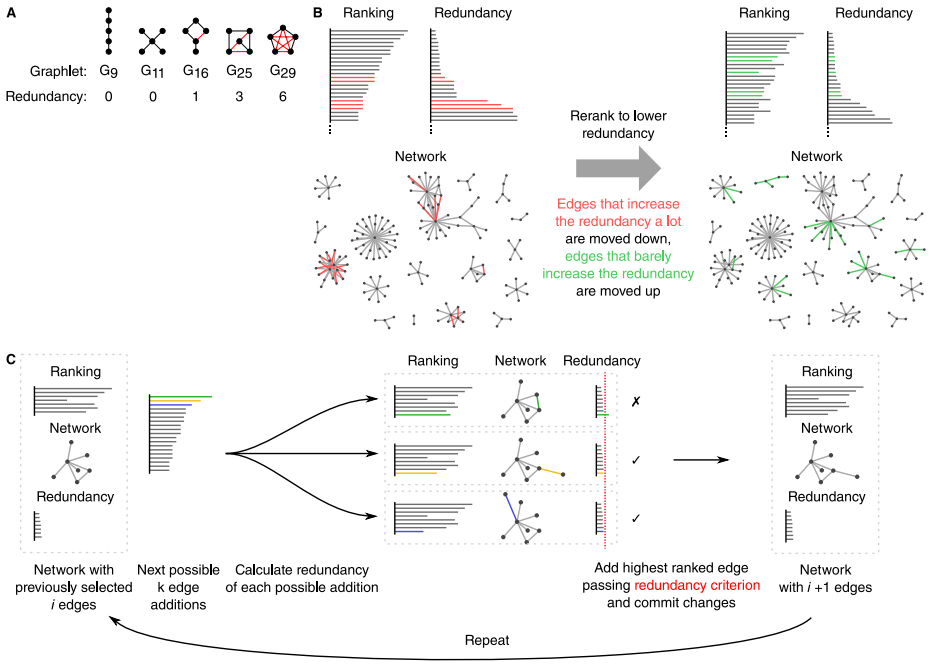


Figure 7.4: Predicted gene regulatory networks of model organisms are optimised to reduce the false positive rate. A) The number of redundant edges in each graphlet are counted. B) The network is optimised in order to obtain a lower redundancy over time. Two networks are shown, one before and one after the optimisation procedure. Edges coloured in red have been removed from the network after optimisation, green edges have been added. C) Starting from an empty network, the interactions are modified by iteratively evaluating the increase in redundancy of the next k interactions, and adding the first edge for which its redundancy is less than the 90^{th} percentile redundancy.

trices of 10000 edge modifications while the non-incremental approach calculates one graphlet count matrix.

Surprisingly, IncGraph obtains higher execution times for networks with 5657 nodes than for networks with 8000 nodes. One possible explanation is that the size of the data structures containing those networks are particularly favourable in avoiding cache misses. Confirmation of this explanation, however, would require further investigation.

Comparing the execution time of IncGraph to the h-index of the networks indicates that the amortised time of IncGraph could be $O(h^3)$ (Fig 7.7). This is in line with the amortised times $O(h)$ and $O(h^2)$ of the algorithm described by Eppstein et al. [22] for counting three-size and four-size subgraphs respectively.

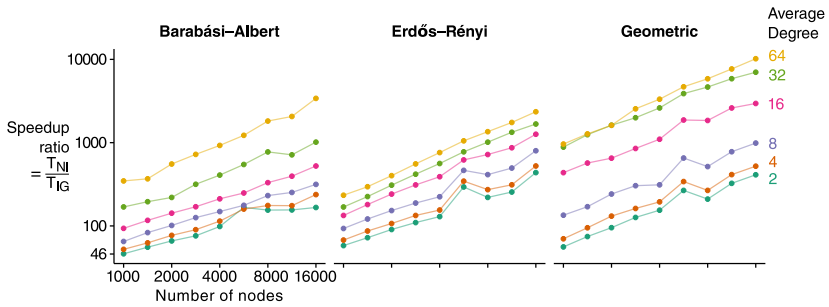


Figure 7.5: IncGraph is significantly faster than non-incremental approaches. For small networks, the execution time of IncGraph T_{IG} is already 50 times less than that of non-incremental approaches T_{NI}. This ratio increases even further for networks with higher numbers of nodes or higher average degrees.

7.3.2 IncGraph allows for better regulatory network optimisation

We implemented a graphlet-based optimisation algorithm for improving the false positive rate of the predicted gene regulatory networks of *E. coli* and *S. cerevisiae*. After reranking the regulatory interactions, the F1 score of the first 1000 interactions had increased by 7.6% and 2.2% respectively (Fig 7.6A). The obtained speedup of about 15-30$\times$ (Fig 7.6B) is in line with the experiments on *in silico* networks. Namely, for the *E. coli* network at 9618 interactions and 889 nodes (average degree = 10.8), a speedup of about 30$\times$ was obtained. Similarly, for the *S. cerevisiae* network at 8013 interactions and 1254 nodes (average degree = 6.4), a speedup of about 15$\times$ was obtained. These speedups are in the same order of magnitude of similarly sized networks (1000 nodes and 8000 interactions) generated with a Barabási-Albert model, with a speedup of 65$\times$. This is to be expected, as such networks share the same scale-free property that gene regulatory networks have.

7.4 Conclusion

Many improvements over the past few years have resulted in efficient graphlet counting algorithms, even for large static networks. However, needing to perform the simplest of tasks tens of thousands of times quickly becomes computationally intractable. As such, recalculating the graphlet counts of a network after each of the many network modification is infeasible.

This study introduces a method for calculating the differences in graphlet (and orbit) counts, which we call incremental graphlet counting or IncGraph for short. We show that IncGraph is at least 10-100 times faster than non-incremental methods for networks of moderate size, and that the speedup ratio increases even further for larger

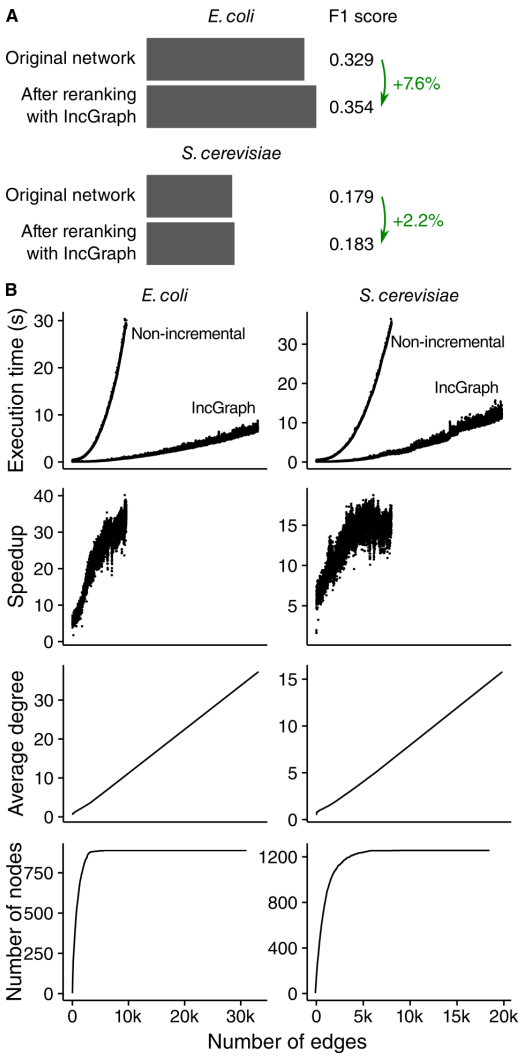


Figure 7.6: A simple graphlet-based scoring method improves predicted regulatory networks. (A) The F1 score was calculated by calculating the harmonic mean of the AUROC and AUPR scores of the first 1000 interactions. (B) IncGraph is significantly faster than the non-incremental approach. Note that for each interaction added to the network, the graphlet counts of 100 putative interactions were evaluated.

networks. To demonstrate the applicability of IncGraph, we optimised a predicted gene regulatory network by enumerating over the ranked edges and observing the graphlet counts of several candidate edges before deciding which edge to add to the network.

IncGraph enables graphlet-based metrics to characterize online networks, e.g. in order to track certain network patterns over time, as a similarity measure in a machine learning task, or as a criterion in a topology optimisation.

7.5 Supplemental information

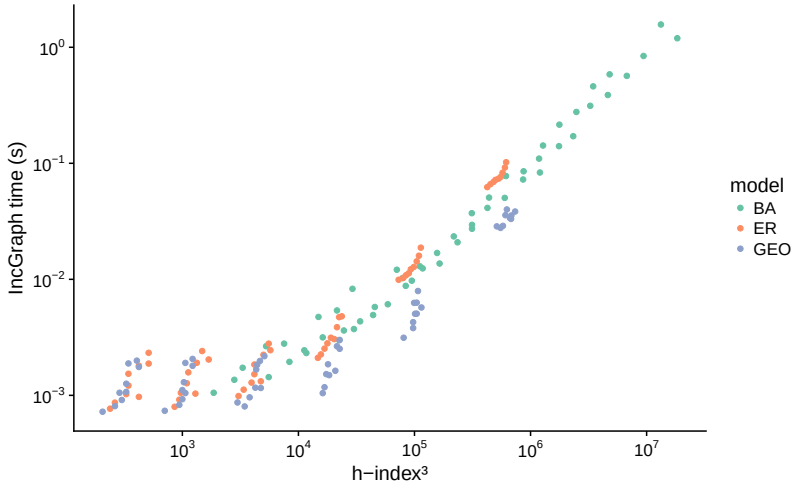


Figure 7.7: Empirical measurements show a strong relation between the execution time of IncGraph and the h-index cubed of the network it was applied to. This is in line with the findings by Eppstein et al., where counting 3-size subgraphs has an amortised time of $O(h)$ and counting 4-size subgraphs has an amortised time of $O(h^2)$.

```
function CalculateDelta(Network G', Node n_0, Node n_1)
                ▷ Assuming an edge (n_0, n_1) has just been added or removed from G'
    Δ⁻ ← Matrix(NumNodes(G), 73) ▷ For storing the orbit counts of the removed graphlets
    Δ⁺ ← Matrix(NumNodes(G), 73)  ▷ For storing the orbit counts of the added graphlets
    B_0 = {n_0, n_1}                          ▷ A blacklist of nodes not to visit anymore
    e = (n_0, n_1)                            ▷ Different name for the edge
    if e ∈ N' then
        (m⁻, m⁺) = (0, 1)          ▷ In other words, m⁻ = 1 iff e ∈ N and m⁺ = 1 iff e ∈ N'
    else
        (m⁻, m⁺) = (1, 0)
    end if
    x_0 = 0
    (Δ⁺, Δ⁻) = CountOrbits(Δ⁺, Δ⁻, (n_0, n_1), x_0, m⁻, m⁺)      ▷ Update delta matrices for
current nodes
    for n_2 ∈ ⋃_{i∈{0,1}} Neighbours(n_i) do
        if n_2 ∉ B_0 then
            B_0 = {n_2} ⋃ B_0                              ▷ Add n_2 to blacklist B_0
            x_1 = x_0 + W(G', (n_0, n_2), 1) + W(G', (n_1, n_2), 2)      ▷ Calculate edge weights for
current nodes
            (Δ⁺, Δ⁻) = CountOrbits(Δ⁺, Δ⁻, (n_0, n_1, n_2), x_1, m⁻, m⁺)
            B_1 = B_0                          ▷ Make a copy for the next iteration depth
            for n_3 ∈ ⋃_{i∈{0,1,2}} Neighbours(n_i) do
                if n_3 ∉ B_1 then
                    B_1 = {n_3} ⋃ B_1
                    x_2 = x_1 + W(G', (n_0, n_3), 3) + W(G', (n_1, n_3), 4) + W(G', (n_2, n_3), 5)
                    (Δ⁺, Δ⁻) = CountOrbits(Δ⁺, Δ⁻, (n_0, n_1, n_2, n_3), x_2, m⁻, m⁺)
                    B_2 = B_1
                    for n_4 ∈ ⋃_{i∈{0,1,2,3}} Neighbours(n_i) do
                        if n_4 ∉ B_2 then
                            B_2 = {n_4} ⋃ B_2
                            x_3 = x_2 + Σ_{i∈{0,1,2,3}} W(G', (n_i, n_4), i + 6)
                            (Δ⁺, Δ⁻) = CountOrbits(Δ⁺, Δ⁻, (n_0, n_1, n_2, n_3, n_4), x_3, m⁻, m⁺)
                        end if
                    end for
                end if
            end for
        end if
    end for
end function
function CountOrbits(Δ⁺, Δ⁻, Nodes S, Edgeweights x, Modifier m⁻, Modifier m⁺)
    L⁻ = L[x + m⁻]                ▷ Look up the orbits of the subgraph induced by S in N
    Δ⁻[S, L⁻] += 1                ▷ Increment orbit counts of nodes S at positions L⁻ in Δ⁻
    L⁺ = L[x + m⁺]                ▷ Look up the orbits of the subgraph induced by S in N'
    Δ⁺[S, L⁺] += 1                ▷ Increment orbit counts of nodes S at positions L⁺ in Δ⁺
    return (Δ⁻, Δ⁺)
end function
function W(Network G, Edge e, Exponent i)
    return e ∈ Edges(G) ? 2^i : 0                          ▷ Return 2^i if G contains edge e
end function
```

Figure 7.8: Pseudocode of IncGraph. IncGraph calculates $\Delta_{G,G'}$ using a strict branch-and-bound strategy.

```
function BarabasiAlbert(number of nodes n, degree d,
                        number of operations o, offset exponent x = 1)
    D ← Rep(0, n)                                  ▷ All nodes start with 0 degree
    T ← Rep({}, n)                                 ▷ List of targets of each node
    S ← Rep({}, n)                                 ▷ List of sources of each node
    O ← ( )                                        ▷ Variable to store the generated operations in
    procedure AddEdge(node i, node j, add as operation b)
        D[i] ← D[i] + 1; D[j] ← D[j] + 1            ▷ Update degrees
        T[i] ← T[i] ∪ {j}; S[j] ← S[j] ∪ {i}        ▷ Update targets and sources
        if b then
            O ← O + (ADD, (i, j))
        end if
    end procedure
    procedure RemoveEdge(node i, node j, add as operation b)
        D[i] ← D[i] − 1; D[j] ← D[j] − 1            ▷ Update degrees
        T[i] ← T[i] \ {j}; S[j] ← S[j] \ {i}        ▷ Update targets and sources
        if b then
            O ← O + (REM, (i, j))
        end if
    end procedure
    procedure AddNode(node i, add as operations b)
        C ← {j | 0 ≤ j < i ∧ j ∉ S[i]}              ▷ Select candidate neighbours
        W ← (D[C] / ΣD[C])^x                       ▷ Calculate weights for preferred attachment
        X ← sample d neighbours from C with weights W
        for j ∈ X do
            AddEdge(i, j, b)
        end for
    end procedure
    procedure RemoveNode(node i, add as operations b)
        while |T[i]| > 0 do
            j ← Head(T[i], 1)
            RemoveEdge(i, j, b)
        end while
    end procedure
    for i ∈ {1 .. m} do
        for j ∈ {0 .. i − 1} do
            AddEdge(i, j, false)                   ▷ Start with m + 1 complete graph
        end for
    end for
    for i ∈ {m + 1 .. n − 1} do
        AddNode(i, false)                          ▷ Add the rest of the nodes
    end for
    G_0 ← {(i, j) | i ∈ {0 .. n − 1} ∧ j ∈ N[i]}   ▷ Initial network
    while |O| < o do                               ▷ Modify network until O is sufficiently large
        i ← Sample 1 index from {0 .. n − 1}
        RemoveNode(i, true)
        AddNode(i, true)
    end while
    O ← Head(O, o)
    return (N_0, O)                                ▷ Return the initial network and o operations
end function
```

Figure 7.9: Pseudo code for generating an evolving Barabási-Albert (BA) network. It first generates a BA network, and then generates o operations such that at any time point, the network is or very closely resembles a BA network.

function ErdosRenyi(number of nodes n, number of edges e, number of operations o)
  $P \leftarrow \{(i, j) \mid i \in \{1 \mathinner{..} n-1\} \wedge j \in \{0 \mathinner{..} i-1\}\}$ ▷ All possible interactions
  $N_0 \leftarrow$ Sample e edges from P
  $N_c \leftarrow N_0$
  while $|O| < o$ **do**
    $e_a \leftarrow$ Sample 1 edge from $P \setminus N_c$
    $e_r \leftarrow$ Sample 1 edge from N_c
    $N_c \leftarrow (N_c \setminus e_r) \cup \{e_a\}$
    $O \leftarrow O + ((\text{ADD}, e_a), (\text{REM}, e_r))$
  end while
  $O \leftarrow \text{Head}(O, o)$
  return (N_0, O) ▷ Return the initial geometric network and o operations
end function

Figure 7.10: Pseudo code for generating an evolving Erdős–Rényi (ER) network. It first generates an ER network, and then generates o operations such that at any time point, the network is or very closely resembles an ER network.

function Geometric(number of nodes n, number of edges e,
    number of operations o, number of dimensions $d = 3$)
  $P \leftarrow$ Sample n points from a multivariate continuous uniform distribution $U_d((0, 1)^d)$
  $D \leftarrow$ Calculate distance matrix from P
  $N_0 \leftarrow \text{Head}(\text{ArgSort}(\text{LowerTriangle}(D)), e)$ ▷ Initial network
  $N_p \leftarrow N_0$
  $O \leftarrow ()$ ▷ Variable to store the generated operations in
  while $|O| < o$ **do** ▷ Modify network until O is sufficiently large
    $i \leftarrow$ Sample 1 index from $\{0 \mathinner{..} n-1\}$
    $P[i] \leftarrow$ Sample 1 point from a $U_d((0, 1)^d)$ ▷ Give node i a new location
    $D[i,] \leftarrow D[, i] \leftarrow$ Calculate new distances between node i and all other nodes
    $N_c \leftarrow \text{Head}(\text{ArgSort}(\text{LowerTriangle}(D)), e)$ ▷ New network
    $O_a \leftarrow \{(\text{ADD}, e) \mid e \notin E(N_p) \wedge e \in E(N_c)\}$ ▷ Gather added edges
    $O_r \leftarrow \{(\text{REM}, e) \mid e \in E(N_p) \wedge e \notin E(N_c)\}$ ▷ Gather removed edges
    $O \leftarrow O + \text{Shuffle}(O_a + O_r)$ ▷ Append new operations to O
    $N_p \leftarrow N_c$
  end while
  $O \leftarrow \text{Head}(O, o)$
  return (N_0, O) ▷ Return the initial geometric network and o operations
end function

Figure 7.11: Pseudo code for generating an evolving geometric network. It first generates a geometric network, and then generates o operations such that at any time point, the network is or very closely resembles a geometric network.

7.6 References

[1] R. Albert. "Network Inference, Analysis, and Modeling in Systems Biology". In: *the Plant Cell Online* 19.11 (2007), pp. 3327–3338. issn: 1040-4651. doi: 10.1105/tpc.107.054700. pmid: 18055607.

[2] Daniel Marbach et al. "Revealing Strengths and Weaknesses of Methods for Gene Network Inference". In: *Proceedings of the National Academy of Sciences* 107.14 (Apr. 2010), pp. 6286–6291. issn: 1091-6490. doi: 10.1073/pnas.0913357107. pmid: 20308593.

[3] Varun Narendra et al. "A Comprehensive Assessment of Methods for De-Novo Reverse-Engineering of Genome-Scale Regulatory Networks". In: *Genomics* 97.1 (2011), pp. 7–18. issn: 08887543. doi: 10.1016/j.ygeno.2010.10.003. pmid: 20951196.

[4] Daniel Marbach et al. "Wisdom of Crowds for Robust Gene Network Inference". In: *Nature methods* 9.8 (July 2012), pp. 796–804. issn: 1548-7091. doi: 10.1038/nmeth.2016. pmid: 22796662.

[5] Tarmo Äijö and Richard Bonneau. "Biophysically Motivated Regulatory Network Inference: Progress and Prospects". In: *Human Heredity* 81.2 (2017), pp. 62–77. issn: 14230062. doi: 10.1159/000446614. pmid: 28076866.

[6] Fabrício M. Lopes et al. "A Feature Selection Technique for Inference of Graphs from Their Known Topological Properties: Revealing Scale-Free Gene Regulatory Networks". In: *Information Sciences* 272 (2014), pp. 1–15. issn: 00200255. doi: 10.1016/j.ins.2014.02.096.

[7] Joeri Ruyssinck et al. "Netter: Re-Ranking Gene Network Inference Predictions Using Structural Network Properties.". In: *BMC Bioinformatics* 17.1 (2016), p. 76. issn: 1471-2105. doi: 10.1186/s12859-016-0913-0. pmid: 26862054.

[8] Alexander W Rives and Timothy Galitski. "Modular Organization of Cellular Networks.". In: *Proceedings of the National Academy of Sciences of the United States of America* 100.3 (2003), pp. 1128–33. issn: 0027-8424. doi: 10.1073/pnas.0237338100. pmid: 12538875.

[9] L H Hartwell et al. "From Molecular to Modular Cell Biology.". In: *Nature* 402 (6761 Suppl 1999), pp. C47–C52. issn: 0028-0836. doi: 10.1038/35011540. pmid: 10591225.

[10] a Barabasi et al. "Network Biology: Understanding the Cell's Functional Organization.". In: *Nature reviews. Genetics* 5.2 (Feb. 2004), pp. 101–13. issn: 1471-0056. doi: 10.1038/nrg1272. pmid: 14735121.

[11] R Milo et al. "Network Motifs: Simple Building Blocks of Complex Networks.". In: *Science (New York, N.Y.)* 298.2002 (2002), pp. 824–827. issn: 00368075. doi: 10.1126/science.298.5594.824. pmid: 12399590.

[12] N Przulj et al. “Modeling Interactome: Scale-Free or Geometric?”. In: *Bioinformatics (Oxford, England)* 20.18 (Dec. 2004), pp. 3508–15. issn: 1367-4803. doi: 10.1093/bioinformatics/bth436. pmid: 15284103.

[13] Tijana Milenković and Nataša Pržulj. “Uncovering Biological Network Function via Graphlet Degree Signatures”. In: *Cancer Informatics* 6 (Jan. 1, 2008), CIN.S680. issn: 1176-9351. doi: 10.4137/CIN.S680.

[14] Cortnie Guerrero et al. “Characterization of the Proteasome Interaction Network Using a QTAX-Based Tag-Team Strategy and Protein Interaction Network Analysis.”. In: *Proceedings of the National Academy of Sciences of the United States of America* 105.36 (2008), pp. 13333–13338. issn: 0027-8424. doi: 10.1073/pnas.0801870105. pmid: 18757749.

[15] Omkar Singh, Kunal Sawariya, and Polamarasetty Aparoy. “Graphlet Signature-Based Scoring Method to Estimate Protein-Ligand Binding Affinity.”. In: *Royal Society open science* 1.4 (2014), p. 140306. issn: 2054-5703. doi: 10.1098/rsos.140306. pmid: 26064572.

[16] Tijana Milenković et al. “Optimal Network Alignment with Graphlet Degree Vectors”. In: *Cancer Informatics* 9 (Jan. 1, 2010), CIN.S4744. issn: 1176-9351. doi: 10.4137/CIN.S4744.

[17] Oleksii Kuchaiev et al. “Topological Network Alignment Uncovers Biological Function and Phylogeny.”. In: *Journal of the Royal Society, Interface / the Royal Society* 7.50 (2010), pp. 1341–1354. issn: 1742-5662. doi: 10.1098/rsif.2010.0063. pmid: 20236959.

[18] Tijana Milenković, Han Zhao, and Fazle E. Faisal. “Global Network Alignment in the Context of Aging”. In: *Proceedings of the International Conference on Bioinformatics, Computational Biology and Biomedical Informatics*. BCB’13. New York, NY, USA: ACM, 2013, 23:23–23:32. isbn: 978-1-4503-2434-2. doi: 10.1145/2506583.2508968.

[19] Nino Shervashidze et al. “Efficient Graphlet Kernels for Large Graph Comparison”. In: *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*. Ed. by David van Dyk and Max Welling. Vol. 5. Proceedings of Machine Learning Research. Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA: PMLR, Apr. 16–18, 2009, pp. 488–495. url: http://proceedings.mlr.press/v5/shervashidze09a.html.

[20] Vladimir Vacic et al. “Graphlet Kernels for Prediction of Functional Residues in Protein Structures.”. In: *Journal of computational biology : a journal of computational molecular cell biology* 17.1 (2010), pp. 55–72. issn: 1557-8666. doi: 10.1089/cmb.2009.0029. pmid: 20078397.

[21] David Eppstein and Emma S. Spiro. “The H-Index of a Graph and Its Application to Dynamic Subgraph Statistics”. In: *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes*

in Bioinformatics). Vol. 5664 LNCS. Springer-Verlag, 2009, pp. 278–289. isbn: 3-642-03366-0. doi: 10.1007/978-3-642-03367-4_25.

[22] David Eppstein et al. "Extended Dynamic Subgraph Statistics Using H-Index Parameterized Data Structures". In: *Theoretical Computer Science* 447 (Aug. 2012), pp. 44–52. issn: 03043975. doi: 10.1016/j.tcs.2011.11.034.

[23] J E Hirsch. "An Index to Quantify an Individual's Scientific Research Output". In: *Proceedings of the National Academy of Sciences of the United States of America* 102.46 (Nov. 2005), pp. 16569–72. issn: 0027-8424. doi: 10.1073/pnas.0507655102. pmid: 16275915.

[24] Tomaž Hočevar and Janez Demšar. "A Combinatorial Approach to Graphlet Counting.". In: *Bioinformatics (Oxford, England)* 30.4 (Feb. 2014), pp. 559–65. issn: 1367-4811. doi: 10.1093/bioinformatics/btt717. pmid: 24336411.

[25] Réka Albert and Albert Laszlo Barabasi. "Statistical Mechanics of Complex Networks". In: *Reviews of Modern Physics* 74 (January 2002), pp. 47–97. issn: 1478-3967. doi: 10.1088/1478-3967/1/3/006. pmid: 16204838.

[26] P. Erdős and A Rényi. "On Random Graphs". In: *Publicationes Mathematicae* 6 (1959), pp. 290–297. issn: 00029947. doi: 10.2307/1999405. pmid: 1205592.

[27] M J B Appel and R P Russo. "The Minimum Vertex Degree of a Graph on Uniform Points in [0,1]d". In: *Adv. in Appl. Probab.* 29.3 (1997), pp. 582–594. issn: 00018678.

[28] Vân Anh Huynh-Thu et al. "Inferring Regulatory Networks from Expression Data Using Tree-Based Methods". In: *PLoS ONE* 5.9 (Jan. 2010), e12776. issn: 1932-6203. doi: 10.1371/journal.pone.0012776. pmid: 20927193.

[29] Marco Moretto et al. "COLOMBOS v3.0: Leveraging Gene Expression Compendia for Cross-Species Analyses". In: *Nucleic Acids Research* 44.D1 (2016), pp. D620–D623. issn: 13624962. doi: 10.1093/nar/gkv1251. pmid: 26586805.

[30] R. Edgar. "Gene Expression Omnibus: NCBI Gene Expression and Hybridization Array Data Repository". In: *Nucleic Acids Research* 30.1 (2002), pp. 207–210. issn: 13624962. doi: 10.1093/nar/30.1.207. pmid: 11752295.

[31] Socorro Gama-Castro et al. "RegulonDB Version 9.0: High-Level Integration of Gene Regulation, Coexpression, Motif Clustering and Beyond". In: *Nucleic Acids Research* 44.D1 (2016), pp. D133–D143. issn: 13624962. doi: 10.1093/nar/gkv1156. pmid: 26527724.

[32] Sisi Ma et al. "De-Novo Learning of Genome-Scale Regulatory Networks in S. Cerevisiae". In: *PLoS ONE* 9.9 (2014). issn: 19326203. doi: 10.1371/journal.pone.0106479. pmid: 25215507.

8 Essential guidelines for computational method benchmarking

Abstract

Purpose: In computational biology and other sciences, researchers are frequently faced with a choice between several computational methods for performing data analyses. Benchmarking studies aim to rigorously compare the performance of different methods using well-characterized datasets, to determine the strengths of each method or to provide recommendations regarding suitable choices of methods for an analysis. However, benchmarking studies must be carefully designed and implemented to provide accurate, unbiased, and informative results.

Results: Here, we summarise key guidelines and recommendations for performing high-quality benchmarking analyses. Our review spans the full pipeline of benchmarking, from defining the scope to best practices for reproducibility. This includes crucial questions regarding design and evaluation principles.

Conclusion: With this review, we aim to guide computational researchers in avoiding common pitfalls in designing, performing, and interpreting benchmarks.

Publication status

Published in Genome Biology 20 (2019). doi:10.1186/s13059-019-1738-8.

Weber LM, Saelens W, **Cannoodt R**, Soneson C, Hapfelmeier A, Gardner PP, Boulesteix AL, Saeys Y, and Robinson MD

Author contributions

- L.M.W. proposed the project and drafted the manuscript.
- W.S., **R.C.**, C.S., A.H., P.P.G., A.L.B., Y.S., and M.D.R. contributed ideas and references and contributed to drafting of the manuscript.
- Y.S. and M.D.R. supervised the project.

8.1 Introduction

Many fields of computational research are characterized by a growing number of available methods for data analysis. For example, at the time of writing, almost 400 methods are available for analysing data from single-cell RNA-sequencing experiments [1]. For experimental researchers and method users, this represents both an opportunity and a challenge, since method choice can significantly affect conclusions.

Benchmarking studies are carried out by computational researchers to compare the performance of different methods, using reference datasets and a range of evaluation criteria. Benchmarks may be performed by authors of new methods to demonstrate performance improvements or other advantages; by independent groups interested in systematically comparing existing methods; or organized as community challenges. Neutral benchmarking studies, i.e., those performed independently of new method development by authors without any perceived bias, and with a focus on the comparison itself, are especially valuable for the research community [2, 3].

From our experience conducting benchmarking studies in computational biology, we have learned several key lessons that we aim to synthesize in this review. A number of previous reviews have addressed this topic from a range of perspectives, including: overall commentaries and recommendations on benchmarking design [2, 4, 5, 6, 7, 8, 9]; surveys of design practices followed by existing benchmarks [7]; the importance of neutral benchmarking studies [3]; principles for the design of real-data benchmarking studies [10, 11] and simulation studies [12]; the incorporation of meta-analysis techniques into benchmarking [13, 14, 15, 16]; the organization and role of community challenges [17, 18]; and discussions on benchmarking design for specific types of methods [19, 20]. More generally, benchmarking may be viewed as a form of meta-research [21].

Our aim is to complement previous reviews by providing a summary of essential guidelines for designing, performing, and interpreting benchmarks. While all guidelines are essential for a truly excellent benchmark, some are more fundamental than others. Our target audience consists of computational researchers who are interested in performing a benchmarking study, or who have already begun one. Our review spans the full pipeline of benchmarking, from defining the scope to best practices for reproducibility. This includes crucial questions regarding design and evaluation principles: for example, using rankings according to evaluation metrics to identify a set of high-performing methods, and then highlighting different strengths and trade-offs among these.

8.2 Ten essential guidelines

The review is structured as a series of guidelines (Figure 8.1), each explained in detail in the following sections. We use examples from computational biology; however, we expect that most arguments apply equally to other fields. We hope that these guidelines will continue the discussion on benchmarking design, as well as assisting computational researchers to design and implement rigorous, informative, and unbiased benchmarking analyses.

1. Define the purpose and scope of the benchmark.
2. Include all relevant methods.
3. Select (or design) representative dataset.
4. Choose appropriate parameter values and software versions.
5. Evaluate methods according to key quantitative performance metrics.
6. Evaluate secondary measures including computational requirements, user-friendliness, installation procedures, and documentation quality.
7. Interpret results and provide recommendations from both user and method developer perspectives.
8. Publish results in an accessible format
9. Design the benchmark to enable future extensions.
10. Follow reproducible research best practices, by making code and data publicly available.

Figure 8.1: Summary of the guidelines as a set of recommendations. Each recommendation is discussed in more detail in the corresponding section in the text.

8.2.1 Defining the purpose and scope

The purpose and scope of a benchmark should be clearly defined at the beginning of the study, and will fundamentally guide the design and implementation. In general, we can define three broad types of benchmarking studies: (i) those by method developers, to demonstrate the merits of their approach (e.g. [22, 23, 24, 25, 26]); (ii) neutral studies performed to systematically compare methods for a certain analysis, either conducted directly by an independent group (e.g. [27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38]) or in collaboration with method authors (e.g. [39]); or (iii) those organized in the form of a community challenge, such as those from the DREAM [40, 41, 42, 43, 44], FlowCAP [45, 46], CASP [47, 48], CAMI [49], Assemblathon [50, 51], MAQC/SEQC [52, 53, 54], and GA4GH [55] consortia.

A neutral benchmark or community challenge should be as comprehensive as possible, although for any benchmark there will be trade-offs in terms of available resources. To minimize perceived bias, a research group conducting a neutral bench-

mark should be approximately equally familiar with all included methods, reflecting typical usage of the methods by independent researchers [3]. Alternatively, the group could include the original method authors, so that each method is evaluated under optimal conditions; methods whose authors decline to take part should be reported. In either case, bias due to focusing attention on particular methods should be avoided – for example, when tuning parameters or fixing bugs. Strategies to avoid these types of biases, such as the use of blinding, have been previously proposed [10].

By contrast, when introducing a new method, the focus of the benchmark will be on evaluating the relative merits of the new method. This may be sufficiently achieved with a less extensive benchmark, for example, by comparing against a smaller set of state-of-the-art and baseline methods. However, the benchmark must still be carefully designed to avoid disadvantaging any methods; for example, extensively tuning parameters for the new method while using default parameters for competing methods would result in a biased representation. Some advantages of a new method may fall outside the scope of a benchmark; for example, a new method may enable more flexible analyses than previous methods (e.g. beyond two-group comparisons in differential analyses [22]).

Finally, results should be summarized in the context of the original purpose of the benchmark. A neutral benchmark or community challenge should provide clear guidelines for method users, and highlight weaknesses in current methods so that these can be addressed by method developers. On the other hand, benchmarks performed to introduce a new method should discuss what the new method offers compared with the current state-of-the-art, such as discoveries that would otherwise not be possible.

8.2.2 Selection of methods

The selection of methods to include in the benchmark will be guided by the purpose and scope of the study. A neutral benchmark should include all available methods for a certain type of analysis. In this case, the publication describing the benchmark will also function as a review of the literature; a summary table describing the methods is a key output (e.g. Figure 2 in [27] or Table 1 in [31]). Alternatively, it may make sense to include only a subset of methods, by defining inclusion criteria: for example, all methods that (i) provide freely available software implementations, (ii) are available for commonly used operating systems, and (iii) can successfully be installed without errors following a reasonable amount of trouble-shooting. Such criteria should be chosen without favouring any methods, and exclusion of any widely used methods should be justified. A useful strategy can be to involve method authors within the process, since they may provide additional details on optimal usage. In addition, community

involvement can lead to new collaborations and inspire future method development. However, the overall neutrality and balance of the resulting research team should be maintained. Finally, if the benchmark is organized as a community challenge, the selection of methods will be determined by the participants. In this case, it is important to communicate the initiative widely – for example, through an established network such as DREAM challenges. However, some authors may choose not to participate; a summary table documenting non-included methods should be provided in this case.

When developing a new method, it is generally sufficient to select a representative subset of existing methods to compare against. For example, this could consist of the current best-performing methods (if known), a simple baseline method, and any methods that are widely used. The selection of competing methods should ensure an accurate and unbiased assessment of the relative merits of the new approach, compared with the current state-of-the-art. In fast-moving fields, for a truly excellent benchmark, method developers should be prepared to update their benchmarks or design them to easily allow extensions as new methods emerge.

8.2.3 Selection (or design) of datasets

The selection of reference datasets is a critical design choice. If suitable publicly accessible datasets cannot be found, they will need to be generated or constructed, either experimentally or by simulation. Including a variety of datasets ensures that methods can be evaluated under a wide range of conditions. In general, reference datasets can be grouped into two main categories: simulated (or synthetic) and real (or experimental).

Simulated data have the advantage that a known true signal (or ground truth) can easily be introduced; for example, whether a gene is differentially expressed. Quantitative performance metrics measuring the ability to recover the known truth can then be calculated. However, it is important to demonstrate that simulations accurately reflect relevant properties of real data, by inspecting empirical summaries of both simulated and real datasets (e.g. using automated tools [56]). The set of empirical summaries to use is context-specific; for example, for single-cell RNA-sequencing, drop-out profiles and dispersion-mean relationships should be compared [29]; for DNA methylation, correlation patterns among neighbouring CpG sites should be investigated [57]; for comparing mapping algorithms, error profiles of the sequencing platforms should be considered [58]. Simplified simulations can also be useful, to evaluate a new method under a basic scenario, or to systematically test aspects such as scalability and stability. However, overly simplistic simulations should be avoided, since these will not provide useful information on performance. A further advantage

of simulated data is that it is possible to generate as much data as required; for example, to study variability and draw statistically valid conclusions.

8

Experimental data often do not contain a ground truth, making it difficult to calculate performance metrics. Instead, methods may be evaluated by comparing them against each other (e.g. overlap between sets of detected differential features [23]), or against a current widely accepted method or gold standard (e.g. manual gating to define cell populations in high-dimensional cytometry [31, 45], or fluorescence *in situ* hybridization to validate absolute copy number predictions [6]). In the context of supervised learning, the response variable to be predicted is known in the manually labelled training and test data. However, individual datasets should not be overused, and using the same dataset for both method development and evaluation should be avoided, due to the risk of overfitting and overly optimistic results [59, 60]. In some cases, it is also possible to design experimental datasets containing a ground truth. Examples include: (i) spiking in synthetic RNA molecules at known relative concentrations [61] in RNA-sequencing experiments (e.g. [54, 62]), (ii) large-scale validation of gene expression measurements by quantitative polymerase chain reaction (e.g. [54]), (iii) using genes located on sex chromosomes as a proxy for silencing of DNA methylation status (e.g. [26, 63]), (iv) using fluorescence-activated cell sorting to sort cells into known sub-populations prior to single-cell RNA-sequencing (e.g. [29, 64, 65]), or (v) mixing different cell lines to create pseudo-cells [66]. However, it may be difficult to ensure that the ground truth represents an appropriate level of variability – for example, the variability of spiked-in material, or whether method performance on cell line data is relevant to out-bred populations. Alternatively, experimental datasets may be evaluated qualitatively, for example, by judging whether each method can recover previous discoveries, although this strategy relies on the validity of previous results.

A further technique is to design semi-simulated datasets that combine real experimental data with an *in silico* (i.e., computational) spike-in signal; for example, by combining cells or genes from null (e.g. healthy) samples with a subset of cells or genes from samples expected to contain a true differential signal (examples include [22, 67, 68]). This strategy can create datasets with more realistic levels of variability and correlation, together with a ground truth.

Overall, there is no perfect reference dataset, and the selection of appropriate datasets will involve trade-offs, for example, regarding the level of complexity. Both simulated and experimental data should not be too simple (e.g. two of the datasets in the FlowCAP-II challenge [45] gave perfect performance for several algorithms) or too difficult (e.g. for the third dataset in FlowCAP-II, no algorithms performed well); in these situations, it can be impossible to distinguish performance. In some cases, individual datasets have also been found to be unrepresentative, leading to over-optimistic or

otherwise biased assessment of methods (e.g. [69]). Overall, the key to truly excellent benchmarking is diversity of evaluations, i.e., using a range of metrics and datasets that span the range of those that might be encountered in practice, so that performance estimates can be credibly extrapolated.

8.2.4 Parameters and software versions

Parameter settings can have a crucial impact on performance. Some methods have a large number of parameters, and tuning parameters to optimal values can require significant effort and expertise. For a neutral benchmark, a range of parameter values should ideally be considered for each method, although trade-offs need to be considered regarding available time and computational resources. Importantly, the selection of parameter values should comply with the neutrality principle, i.e., certain methods should not be favoured over others through more extensive parameter tuning.

There are three major strategies for choosing parameters. The first (and simplest) is to use default values for all parameters. Default parameters may be adequate for many methods, although this is difficult to judge in advance. While this strategy may be viewed as too simplistic for some neutral benchmarks, it reflects typical usage. We used default parameters in several neutral benchmarks where we were interested in performance for untrained users [27, 70, 71]. In addition, for [27], due to the large number of methods and datasets, total runtime was already around a week using 192 processor cores, necessitating judgement in the scope of parameter tuning. The second strategy is to choose parameters based on previous experience or published values. This relies on familiarity with the methods and the literature, reflecting usage by expert users. The third strategy is to use a systematic or automated parameter tuning procedure – for example, a grid search across ranges of values for multiple parameters or techniques such as cross-validation (e.g. [30]). The strategies may also be combined, for example, by setting non-critical parameters to default values and performing a grid search for key parameters. Regardless, neutrality should be maintained: comparing methods with the same strategy makes sense, while comparing one method with default parameters against another with extensive tuning makes for an unfair comparison.

For benchmarks performed to introduce a new method, comparing against a single set of optimal parameter values for competing methods is often sufficient; these values may be selected during initial exploratory work or by consulting documentation. However, as outlined above, bias may be introduced by tuning the parameters of the new method more extensively. The parameter selection strategy should be transparently discussed during the interpretation of the results, to avoid the risk of over-optimistic reporting due to expending more researcher degrees of freedom on the

new method [5, 72].

Software versions can also influence results, especially if updates include major changes to methodology (e.g. [73]). Final results should generally be based on the latest available versions, which may require re-running some methods if updates become available during the course of a benchmark.

8.2.5 Evaluation criteria: key quantitative performance metrics

Evaluation of methods will rely on one or more quantitative performance metrics (Figure 8.2A). The choice of metric depends on the type of method and data. For example, for classification tasks with a ground truth, metrics include the true positive rate (TPR; sensitivity or recall), false positive rate (FPR; 1 - specificity), and false discovery rate (FDR). For clustering tasks, common metrics include the F1 score, adjusted Rand index, normalized mutual information, precision, and recall; some of these can be calculated at the cluster level as well as averaged (and optionally weighted) across clusters (e.g. these metrics were used to evaluate clustering methods in our own work [28, 31] and by others [33, 45, 74]). Several of these metrics can also be compared visually to capture the trade-off between sensitivity and specificity, for example, using receiver operating characteristic (ROC) curves (TPR versus FPR), TPR versus FDR curves, or precision-recall (PR) curves (Figure 8.2B). For imbalanced datasets, PR curves have been shown to be more informative than ROC curves [75, 76]. These visual metrics can also be summarized as a single number, such as area under the ROC or PR curve; examples from our work include [22, 29]. In addition to the trade-off between sensitivity and specificity, a methods operating point' is important; in particular, whether the threshold used (e.g. 5% FDR) is calibrated to achieve the specified error rate. We often overlay this onto TPR-FDR curves by filled or open circles (e.g. Figure 8.2B, generated using the iCOBRA package [77]); examples from our work include [22, 23, 25, 78].

For methods with continuous-valued output (e.g. effect sizes or abundance estimates), metrics include the root mean square error, distance measures, Pearson correlation, sum of absolute log-ratios, log-modulus, and cross-entropy. As above, the choice of metric depends on the type of method and data (e.g. [41, 79] used correlation, while [48] used root mean square deviation). Further classes of methods include those generating graphs, phylogenetic trees, overlapping clusters, or distributions; these require more complex metrics. In some cases, custom metrics may need to be developed (e.g. we defined new metrics for topologies of developmental trajectories in [27]). When designing custom metrics, it is important to assess their reliability across a range of prediction values (e.g. [80, 81]). For some metrics, it may also be useful to assess uncertainty, for example, via confidence intervals. In the context of supervised

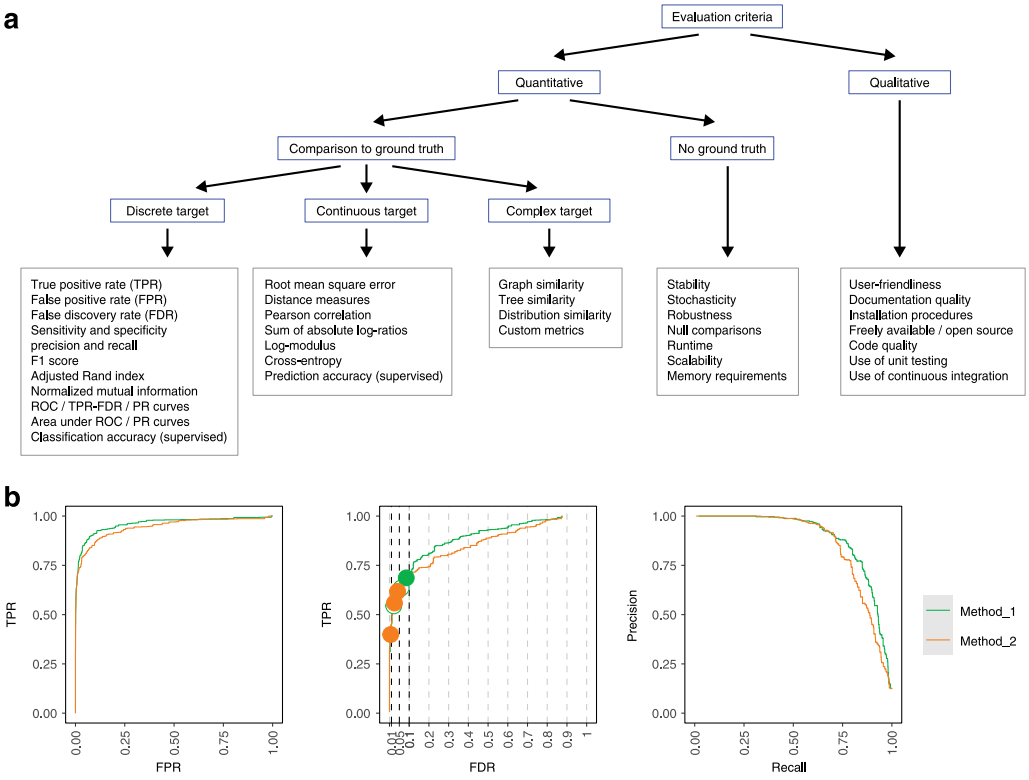


Figure 8.2: Summary and examples of performance metrics. A: Schematic overview of classes of frequently used performance metrics, including examples (*boxes outlined in gray*). **B:** Examples of popular visualizations of quantitative performance metrics for classification methods, using reference datasets with a ground truth. ROC curves (*left*). TPR versus FDR curves (*centre*); circles represent observed TPR and FDR at typical FDR thresholds of 1, 5, and 10%, with filled circles indicating observed FDR lower than or equal to the imposed threshold. PR curves (*right*). Visualizations were generated using iCOBRA R/Bioconductor package [77]. *FDR* false discovery rate, *FPR* false positive rate, *PR* precision-recall, *ROC* receiver operating characteristic, *TPR* true positive rate

learning, classification or prediction accuracy can be evaluated by cross-validation, bootstrapping, or on a separate test dataset (e.g. [13, 46]). In this case, procedures to split data into training and test sets should be appropriate for the data structure and the prediction task at hand (e.g. leaving out whole samples or chromosomes [82]).

Additional metrics that do not rely on a ground truth include measures of stability, stochasticity, and robustness. These measures may be quantified by running methods multiple times using different inputs or sub-sampled data (e.g. we observed substantial variability in performance for some methods in [29, 31]). Missing values may occur if a method does not return any values for a certain metric, for example, due to a failure to converge or other computational issues such as excessive runtime or memory requirements (e.g. [27, 29, 31]). Fall-back solutions such as imputation may

be considered in this case [83], although these should be transparently reported. For non-deterministic methods (e.g. with random starts or stochastic optimization), variability in performance when using different random seeds or sub-sampled data should be characterized. Null comparisons can be constructed by randomizing group labels such that datasets do not contain any true signal, which can provide information on error rates (e.g. [22, 25, 26]). However, these must be designed carefully to avoid confounding by batch or population structure, and to avoid strong within-group batch effects that are not accounted for.

For most benchmarks, multiple metrics will be relevant. Focusing on a single metric can give an incomplete view: methods may not be directly comparable if they are designed for different tasks, and different users may be interested in different aspects of performance. Therefore, a crucial design decision is whether to focus on an overall ranking, for example, by combining or weighting multiple metrics. In general, it is unlikely that a single method will perform best across all metrics, and performance differences between top-ranked methods for individual metrics can be small. Therefore, a good strategy is to use rankings from multiple metrics to identify a set of consistently high-performing methods, and then highlight the different strengths of these methods. For example, in [31], we identified methods that gave good clustering performance, and then highlighted differences in run-times among these. In several studies, we have presented results in the form of a graphical summary of performance according to multiple criteria (examples include Figure 3 in [27] and Figure 5 in [29] from our work; and Figure 2 in [39] and Figure 6 in [32] from other authors). Identifying methods that consistently under-perform can also be useful, to allow readers to avoid these.

8.2.6 Evaluation criteria: secondary measures

In addition to the key quantitative performance metrics, methods should also be evaluated according to secondary measures, including runtime, scalability, and other computational requirements, as well as qualitative aspects such as user-friendliness, installation procedures, code quality, and documentation quality (Figure 8.2A). From the user perspective, the final choice of method may involve trade-offs according to these measures: an adequately performing method may be preferable to a top-performing method that is especially difficult to use.

In our experience, run-times and scalability can vary enormously between methods (e.g. in our work, run-times for cytometry clustering algorithms [31] and metagenome analysis tools [79] ranged across multiple orders of magnitude for the same datasets). Similarly, memory and other computational requirements can vary widely. Run-times and scalability may be investigated systematically, for example, by

varying the number of cells or genes in a single-cell RNA-sequencing dataset [28, 29]. In many cases, there is a trade-off between performance and computational requirements. In practice, if computational requirements for a top-performing method are prohibitive, then a different method may be preferred by some users.

User-friendliness, installation procedures, and documentation quality can also be highly variable [84, 85]. Streamlined installation procedures can be ensured by distributing the method via standard package repositories, such as CRAN and Bioconductor for R, or PyPI for Python. Alternative options include GitHub and other code repositories or institutional websites; however, these options do not provide users with the same guarantees regarding reliability and documentation quality. Availability across multiple operating systems and within popular programming languages for data analysis is also important. Availability of graphical user interfaces can further extend accessibility, although graphical-only methods hinder reproducibility and are thus difficult to include in a systematic benchmark.

For many users, freely available and open source software will be preferred, since it is more broadly accessible and can be adapted by experienced users. From the developer perspective, code quality and use of software development best practices, such as unit testing and continuous integration, are also important. Similarly, adherence to commonly used data formats (e.g. GFF/GTF files for genomic features, BAM/SAM files for sequence alignment data, or FCS files for flow or mass cytometry data) greatly improves accessibility and extensibility.

High-quality documentation is critical, including help pages and tutorials. Ideally, all code examples in the documentation should be continually tested, for example, as Bioconductor does, or through continuous integration.

8.2.7 Interpretation, guidelines, and recommendations

For a truly excellent benchmark, results must be clearly interpreted from the perspective of the intended audience. For method users, results should be summarized in the form of recommendations. An overall ranking of methods (or separate rankings for multiple evaluation criteria) can provide a useful overview. However, as mentioned above, some methods may not be directly comparable (e.g. since they are designed for different tasks), and different users may be interested in different aspects of performance. In addition, it is unlikely that there will be a clear winner across all criteria, and performance differences between top-ranked methods can be small. Therefore, an informative strategy is to use the rankings to identify a set of high-performing methods, and to highlight the different strengths and trade-offs among these methods. The interpretation may also involve biological or other domain knowledge to establish

the scientific relevance of differences in performance. Importantly, neutrality principles should be preserved during the interpretation.

For method developers, the conclusions may include guidelines for possible future development of methods. By assisting method developers to focus their research efforts, high-quality benchmarks can have significant impact on the progress of methodological research.

Limitations of the benchmark should be transparently discussed. For example, in [27] we used default parameters for all methods, while in [31] our datasets relied on manually gated reference cell populations as the ground truth. Without a thorough discussion of limitations, a benchmark runs the risk of misleading readers; in extreme cases, this may even harm the broader research field by guiding research efforts in the wrong directions.

8.2.8 Publication and reporting of results

The publication and reporting strategy should emphasize clarity and accessibility. Visualizations summarizing multiple performance metrics can be highly informative for method users (examples include Figure 3 in [27] and Figure 5 in [29] from our own work; as well as Figure 6 in [32]). Summary tables are also useful as a reference (e.g. [31, 45]). Additional visualizations, such as flow charts to guide the choice of method for different analyses, are a helpful way to engage the reader (e.g. Figure 5 in [27]).

For extensive benchmarks, online resources enable readers to interactively explore the results (examples from our work include [27, 31], which allow users to filter metrics and datasets). Figure 3 displays an example of an interactive website from one of our benchmarks [27], which facilitates exploration of results and assists users with choosing a suitable method. While trade-offs should be considered in terms of the amount of work required, these efforts are likely to have significant benefit for the community.

In most cases, results will be published in a peer-reviewed article. For a neutral benchmark, the benchmark will be the main focus of the paper. For a benchmark to introduce a new method, the results will form one part of the exposition. We highly recommend publishing a preprint prior to peer review (e.g. on bioRxiv or arXiv) to speed up distribution of results, broaden accessibility, and solicit additional feedback. In particular, direct consultation with method authors can generate highly useful feedback (examples from our work are described in the acknowledgements in [79, 86]). Finally, at publication time, considering open access options will further broaden accessibility.

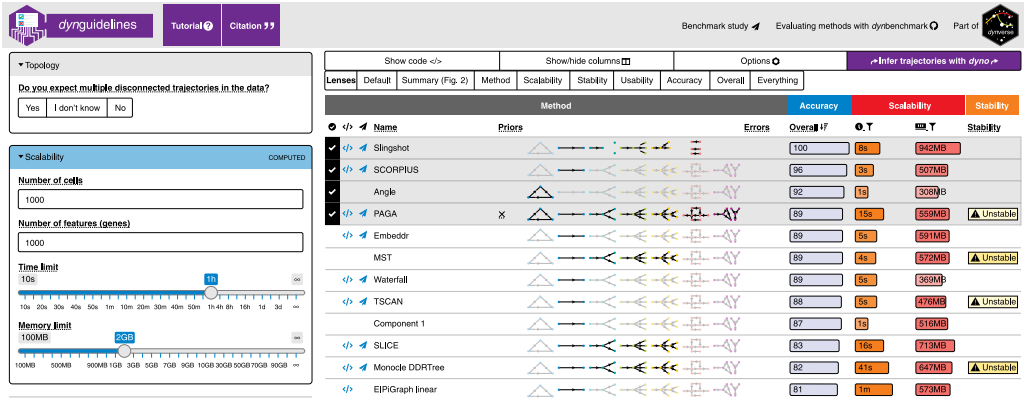


Figure 8.3: Example of an interactive website allowing users to explore the results of one of our benchmarking studies [27]. This website was created using the Shiny framework in R.

8.2.9 Enabling future extensions

Since new methods are continually emerging [1], benchmarks can quickly become out of date. To avoid this, a truly excellent benchmark should be extensible. For example, creating public repositories containing code and data allows other researchers to build on the results to include new methods or datasets, or to try different parameter settings or pre-processing procedures (examples from our work include [27, 28, 29, 30, 31]). In addition to raw data and code, it is useful to distribute pre-processed and/or results data (examples include [28, 29, 77] from our work and [74, 87, 88] from others), especially for computationally intensive benchmarks. This may be combined with an interactive website, where users can upload results from a new method, to be included in an updated comparison either automatically or by the original authors (e.g. [35, 89, 90]). Continuous benchmarks, which are continually updated, are especially convenient (e.g. [91]), but may require significant additional effort.

8.2.10 Reproducible research best practices

Reproducibility of research findings has become an increasing concern in numerous areas of study [92]. In computational sciences, reproducibility of code and data analyses has been recognized as a useful minimum standard that enables other researchers to verify analyses [93]. Access to code and data has previously enabled method developers to uncover potential errors in published benchmarks due to suboptimal usage of methods [73, 94, 95]. Journal publication policies can play a crucial role in encouraging authors to follow these practices [96]; experience shows that statements that code and data are available on request are often insufficient [97]. In the context of benchmarking, code and data availability also provides further benefits: for method

users, code repositories serve as a source of annotated code to run methods and build analysis pipelines, while for developers, code repositories can act as a prototype for future method development work.

Parameter values (including random seeds) and software versions should be clearly reported to ensure complete reproducibility. For methods that are run using scripts, these will be recorded within the scripts. In R, the command `sessionInfo()` gives a complete summary of package versions, the version of R, and the operating system. For methods only available via graphical interfaces, parameters and versions must be recorded manually. Reproducible workflow frameworks, such as the Galaxy platform [98], can also be helpful. A summary table or spreadsheet of parameter values and software versions can be published as supplementary information along with the publication describing the benchmark (e.g. Supporting Information Table S1 in our study [31]).

Automated workflow management tools and specialized tools for organizing benchmarks provide sophisticated options for setting up benchmarks and creating a reproducible record, including software environments, package versions, and parameter values. Examples include SummarizedBenchmark [99], DataPackageR [100], workflowr [101], and Dynamic Statistical Comparisons [102]. Some tools (e.g. workflowr) also provide streamlined options for publishing results online. In machine learning, OpenML provides a platform to organize and share benchmarks [103]. More general tools for managing computational workflows, including Snakemake [104], Make, Bioconda [105], and conda, can be customized to capture setup information. Containerization tools such as Docker and Singularity may be used to encapsulate a software environment for each method, preserving the package version as well as dependency packages and the operating system, and facilitating distribution of methods to end users (e.g. in our study [27]). Best practices from software development are also useful, including unit testing and continuous integration.

Many free online resources are available for sharing code and data, including GitHub and Bitbucket, repositories for specific data types (e.g. ArrayExpress [106], the Gene Expression Omnibus [107], and FlowRepository [108]), and more general data repositories (e.g. figshare, Dryad, Zenodo, Bioconductor ExperimentHub, and Mendeley Data). Customized resources (examples from our work include [29, 77]) can be designed when additional flexibility is needed. Several repositories allow the creation of digital object identifiers (DOIs) for code or data objects. In general, preference should be given to publicly funded repositories, which provide greater guarantees for long-term archival stability [84, 85].

An extensive literature exists on best practices for reproducible computational research (e.g. [109]). Some practices (e.g. containerization) may involve significant

additional work; however, in our experience, almost all efforts in this area prove useful, especially by facilitating later extensions by ourselves or other researchers.

8.3 Discussion

In this review, we have described a set of key principles for designing a high-quality computational benchmark. In our view, elements of all of these principles are essential. However, we have also emphasized that any benchmark will involve trade-offs, due to limited expertise and resources, and that some principles are less central to the evaluation. Table 8.1 provides a summary of examples of key trade-offs and pitfalls related to benchmarking, along with our judgement of how truly essential each principle is.

A number of potential pitfalls may arise from benchmarking studies (Table 8.1). For example, subjectivity in the choice of datasets or evaluation metrics could bias the results. In particular, a benchmark that relies on unrepresentative data or metrics that do not translate to real-world scenarios may be misleading by showing poor performance for methods that otherwise perform well. This could harm method users, who may select an inappropriate method for their analyses, as well as method developers, who may be discouraged from pursuing promising methodological approaches. In extreme cases, this could negatively affect the research field by influencing the direction of research efforts. A thorough discussion of the limitations of a benchmark can help avoid these issues. Over the longer term, critical evaluations of published benchmarks, so-called meta-benchmarks, will also be informative [10, 13, 14].

Well-designed benchmarking studies provide highly valuable information for users and developers of computational methods, but require careful consideration of a number of important design principles. In this review, we have discussed a series of guidelines for rigorous benchmarking design and implementation, based on our experiences in computational biology. We hope these guidelines will assist computational researchers to design high-quality, informative benchmarks, which will contribute to scientific advances through informed selection of methods by users and targeting of research efforts by developers.

Table 8.1: Summary of our views regarding how essential each principle is for a truly excellent benchmark, along with examples of key trade-offs and potential pitfalls relating to each principle. The higher the number of plus signs, the more central the principle is to the evaluation.

Principle	How essential	Trade-offs	Potential pitfalls
1. Defining the purpose and score	+++	How comprehensive the benchmark should be	Scope too broad: too much work given available resources Scope too narrow: unrepresentative and possibly misleading results
2. Selection of methods	+++	Number of methods to include	Excluding key methods
3. Selection (or design) of datasets	+++	Number and types of datasets to include	Subjectivity in the choice of datasets: e.g. selecting datasets that are unrepresentative of real-world applications Too few datasets or simulation scenarios Overly simplistic simulations
4. Parameter and software versions	++	Amount of parameter tuning	Extensive parameter tuning for some methods while using default parameters for others (e.g. competing methods)
5. Evaluation criteria: key quantitative performance metrics	+++	Number and types of performance metrics	Subjectivity in the choice of metrics: e.g. selecting metrics that do not translate to real-world performance Metrics that give over-optimistic estimates of performance Methods may not be directly comparable according to individual metrics (e.g. if methods are designed for different tasks)
6. Evaluation criteria: secondary measures	++	Number and types of performance metrics	Subjectivity of qualitative measures such as user-friendliness, installation procedures, and documentation quality Subjectivity in relative weighting between multiple metrics Measures such as runtime and scalability depend on processor speed and memory
7. Interpretation, guidelines, and recommendations	++	Generality versus specificity of recommendations	Performance differences between top-ranked methods may be minor Different readers may be interested in different aspects of performance
8. Publication and reporting of results	+	Amount of resources to dedicate to building online resources	Online resources may not be accessible (or may no longer run) several years later
9. Enabling future extensions	++	Amount of resources to dedicate to ensuring extensibility	Selection of methods or datasets for future extensions may be unrepresentative (e.g. due to requests from method authors)
10. Reproducible research best practices	++	Amount of resources to dedicate to reproducibility	Some tools may not be compatible or accessible several years later

8.4 References

[1] Luke Zappia, Belinda Phipson, and Alicia Oshlack. "Exploring the Single-Cell RNA-Seq Analysis Landscape with the scRNA-Tools Database". In: *PLOS Computational Biology* 14.6 (June 2018), e1006245. issn: 1553-7358. doi: 10.1371/journal.pcbi.1006245.

[2] Anne-Laure Boulesteix et al. "On the Necessity and Design of Studies Comparing Statistical Methods". In: *Biometrical Journal* 60.1 (2018), pp. 216–218. issn: 1521-4036. doi: 10.1002/bimj.201700129.

[3] Anne-Laure Boulesteix, Sabine Lauer, and Manuel J. A. Eugster. "A Plea for Neutral Comparison Studies in Computational Sciences". In: *PloS One* 8.4 (2013), e61562. issn: 1932-6203. doi: 10.1371/journal.pone.0061562. pmid: 23637855.

[4] Bjoern Peters et al. "Putting Benchmarks in Their Rightful Place: The Heart of Computational Biology". In: *PLoS computational biology* 14.11 (Nov. 2018), e1006494. issn: 1553-7358. doi: 10.1371/journal.pcbi.1006494. pmid: 30408027.

[5] Anne-Laure Boulesteix. "Ten Simple Rules for Reducing Overoptimistic Reporting in Methodological Computational Research". In: *PLOS Computational Biology* 11.4 (Apr. 2015), e1004191. issn: 1553-7358. doi: 10.1371/journal.pcbi.1004191.

[6] Siyuan Zheng. "Benchmarking: Contexts and Details Matter". In: *Genome Biology* 18.1 (May 7, 2017), p. 129. issn: 1474-760X. doi: 10.1186/s13059-017-1258-3. pmid: 28679434.

[7] Serghei Mangul et al. "Systematic Benchmarking of Omics Computational Tools". In: *Nature Communications* 10.1 (Mar. 27, 2019), p. 1393. issn: 2041-1723. doi: 10.1038/s41467-019-09406-4. pmid: 30918265.

[8] Raquel Norel, John Jeremy Rice, and Gustavo Stolovitzky. "The Self-Assessment Trap: Can We All Be Better than Average?". In: *Molecular systems biology* 7.1 (2011), p. 537. issn: 1744-4292. doi: 10.1038/msb.2011.70. pmid: 21988833.

[9] Mohamed Radhouene Aniba, Olivier Poch, and Julie D Thompson. "Issues in Bioinformatics Benchmarking: The Case Study of Multiple Sequence Alignment". In: *Nucleic Acids Research* 38.21 (Nov. 2010), pp. 7353–7363. issn: 0305-1048. doi: 10.1093/nar/gkq625. pmid: 20639539.

[10] Anne-Laure Boulesteix, Rory Wilson, and Alexander Hapfelmeier. "Towards Evidence-Based Computational Statistics: Lessons from Clinical Research on the Role and Design of Real-Data Benchmark Studies". In: *BMC medical research methodology* 17.1 (Sept. 9, 2017), p. 138. issn: 1471-2288. doi: 10.1186/s12874-017-0417-2. pmid: 28888225.

[11] Anne-Laure Boulesteix et al. "A Statistical Framework for Hypothesis Testing in Real Data Comparison Studies". In: *The American Statistician* 69.3 (July 3, 2015), pp. 201–212. issn: 0003-1305. doi: 10.1080/00031305.2015.1005128.

[12] Tim P. Morris, Ian R. White, and Michael J. Crowther. "Using Simulation Studies to Evaluate Statistical Methods". In: *Statistics in Medicine* 38.11 (May 20, 2019), pp. 2074–2102. issn: 1097-0258. doi: 10.1002/sim.8086. pmid: 30652356.

[13] Paul P. Gardner et al. "Identifying Accurate Metagenome and Amplicon Software via a Meta-Analysis of Sequence to Taxonomy Benchmarking Studies". In: *PeerJ* 7 (2019), e6160. issn: 2167-8359. doi: 10.7717/peerj.6160. pmid: 30631651.

[14] Paul P. Gardner et al. "A Meta-Analysis of Bioinformatics Software Benchmarks Reveals That Publication-Bias Unduly Influences Software Accuracy". In: *bioRxiv* (Jan. 2, 2017), p. 092205. doi: 10.1101/092205.

[15] Evangelos Evangelou and John P. A. Ioannidis. "Meta-Analysis Methods for Genome-Wide Association Studies and Beyond". In: *Nature Reviews. Genetics* 14.6 (June 2013), pp. 379–389. issn: 1471-0064. doi: 10.1038/nrg3472. pmid: 23657481.

[16] Fangxin Hong and Rainer Breitling. "A Comparison of Meta-Analysis Methods for Detecting Differentially Expressed Genes in Microarray Experiments". In: *Bioinformatics (Oxford, England)* 24.3 (Feb. 1, 2008), pp. 374–382. issn: 1367-4811. doi: 10.1093/bioinformatics/btm620. pmid: 18204063.

[17] Paul C. Boutros et al. "Toward Better Benchmarking: Challenge-Based Methods Assessment in Cancer Genomics". In: *Genome Biology* 15.9 (Sept. 17, 2014), p. 462. issn: 1474-760X. doi: 10.1186/s13059-014-0462-7. pmid: 25314947.

[18] Iddo Friedberg et al. "Ten Simple Rules for a Community Computational Challenge". In: *PLoS computational biology* 11.4 (Apr. 2015), e1004150. issn: 1553-7358. doi: 10.1371/journal.pcbi.1004150. pmid: 25906249.

[19] Iven Van Mechelen et al. "Benchmarking in Cluster Analysis: A White Paper". In: (Sept. 27, 2018). arXiv: 1809.10496 [stat]. url: http://arxiv.org/abs/1809.10496 (visited on 09/19/2019).

[20] Alexandre Angers-Loustau et al. "The Challenges of Designing a Benchmark Strategy for Bioinformatics Pipelines in the Identification of Antimicrobial Resistance Determinants Using next Generation Sequencing Technologies". In: *F1000Research* 7 (Dec. 7, 2018), p. 459. issn: 2046-1402. doi: 10.12688/f1000research.14509.2.

[21] John P. A. Ioannidis. "Meta-Research: Why Research on Research Matters". In: *PLoS biology* 16.3 (Mar. 2018), e2005468. issn: 1545-7885. doi: 10.1371/journal.pbio.2005468. pmid: 29534060.

[22] Lukas M. Weber et al. "Diffcyt: Differential Discovery in High-Dimensional Cytometry via High-Resolution Clustering". In: *Communications Biology* 2 (2019), p. 183. issn: 2399-3642. doi: 10.1038/s42003-019-0415-5. pmid: 31098416.

[23] Malgorzata Nowicka and Mark D. Robinson. "DRIMSeq: A Dirichlet-Multinomial Framework for Multivariate Count Outcomes in Genomics". In: *F1000Research* 5 (2016), p. 1356. issn: 2046-1402. doi: 10.12688/f1000research.8900.2. pmid: 28105305.

[24] Jacob H. Levine et al. "Data-Driven Phenotypic Dissection of AML Reveals Progenitor-like Cells That Correlate with Prognosis". In: *Cell* 162.1 (July 2, 2015), pp. 184–197. issn: 1097-4172. doi: 10.1016/j.cell.2015.05.047. pmid: 26095251.

[25] Xiaobei Zhou, Helen Lindsay, and Mark D. Robinson. "Robustly Detecting Differential Expression in RNA Sequencing Data Using Observation Weights". In: *Nucleic Acids Research* 42.11 (June 2014), e91. issn: 1362-4962. doi: 10.1093/nar/gku310. pmid: 24753412.

[26] Charity W. Law et al. "Voom: Precision Weights Unlock Linear Model Analysis Tools for RNA-Seq Read Counts". In: *Genome Biology* 15.2 (Feb. 3, 2014), R29. issn: 1474-760X. doi: 10.1186/gb-2014-15-2-r29. pmid: 24485249.

[27] Wouter Saelens et al. "A Comparison of Single-Cell Trajectory Inference Methods". In: *Nature Biotechnology* 37 (May 2019). issn: 15461696. doi: 10.1038/s41587-019-0071-9.

[28] 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), p. 1141. issn: 2046-1402. doi: 10.12688/f1000research.15666.2. pmid: 30271584.

[29] Charlotte Soneson and Mark D. Robinson. "Bias, Robustness and Scalability in Single-Cell Differential Expression Analysis". In: *Nature Methods* 15.4 (Apr. 2018), pp. 255–261. issn: 1548-7105. doi: 10.1038/nmeth.4612. pmid: 29481549.

[30] Wouter Saelens, Robrecht Cannoodt, and Yvan Saeys. "A Comprehensive Evaluation of Module Detection Methods for Gene Expression Data". In: *Nature Communications* 9.1 (Mar. 2018), p. 1090. issn: 2041-1723. doi: 10.1038/s41467-018-03424-4.

[31] Lukas M. Weber and Mark D. Robinson. "Comparison of Clustering Methods for High-Dimensional Single-Cell Flow and Mass Cytometry Data". In: *Cytometry Part A* 89.12 (2016), pp. 1084–1096. issn: 1552-4930. doi: 10.1002/cyto.a.23030.

[32] Keegan Korthauer et al. "A Practical Guide to Methods Controlling False Discoveries in Computational Biology". In: *Genome Biology* 20.1 (Apr. 6, 2019), p. 118. issn: 1474-760X. doi: 10.1186/s13059-019-1716-1. pmid: 31164141.

[33] Saskia Freytag et al. "Comparison of Clustering Tools in R for Medium-Sized 10x Genomics Single-Cell RNA-Sequencing Data". In: *F1000Research* 7 (2018), p. 1297. issn: 2046-1402. doi: 10.12688/f1000research.15809.2. pmid: 30228881.

[34] Giacomo Baruzzo et al. "Simulation-Based Comprehensive Benchmarking of RNA-Seq Aligners". In: *Nature Methods* 14.2 (Feb. 2017), pp. 135–139. issn: 1548-7105. doi: 10.1038/nmeth.4106. pmid: 27941783.

[35] Alexander Kanitz et al. "Comparative Assessment of Methods for the Computational Inference of Transcript Isoform Abundance from RNA-Seq Data". In: *Genome Biology* 16 (July 23, 2015), p. 150. issn: 1474-760X. doi: 10.1186/s13059-015-0702-5. pmid: 26201343.

[36] Charlotte Soneson and Mauro Delorenzi. "A Comparison of Methods for Differential Expression Analysis of RNA-Seq Data". In: *BMC bioinformatics* 14 (Mar. 9, 2013), p. 91. issn: 1471-2105. doi: 10.1186/1471-2105-14-91. pmid: 23497356.

[37] Franck Rapaport et al. "Comprehensive Evaluation of Differential Gene Expression Analysis Methods for RNA-Seq Data". In: *Genome Biology* 14.9 (2013), R95. issn: 1474-760X. doi: 10.1186/gb-2013-14-9-r95. pmid: 24020486.

[38] Marie-Agnès Dillies et al. "A Comprehensive Evaluation of Normalization Methods for Illumina High-Throughput RNA Sequencing Data Analysis". In: *Briefings in Bioinformatics* 14.6 (Nov. 2013), pp. 671–683. issn: 1477-4054. doi: 10.1093/bib/bbs046. pmid: 22988256.

[39] Daniel Sage et al. "Quantitative Evaluation of Software Packages for Single-Molecule Localization Microscopy". In: *Nature Methods* 12.8 (Aug. 2015), pp. 717–724. issn: 1548-7105. doi: 10.1038/nmeth.3442. pmid: 26076424.

[40] Matthew T. Weirauch et al. "Evaluation of Methods for Modeling Transcription Factor Sequence Specificity". In: *Nature Biotechnology* 31.2 (Feb. 2013), pp. 126–134. issn: 1546-1696. doi: 10.1038/nbt.2486. pmid: 23354101.

[41] James C. Costello et al. "A Community Effort to Assess and Improve Drug Sensitivity Prediction Algorithms". In: *Nature Biotechnology* 32.12 (Dec. 2014), pp. 1202–1212. issn: 1546-1696. doi: 10.1038/nbt.2877. pmid: 24880487.

[42] Robert Küffner et al. "Crowdsourced Analysis of Clinical Trial Data to Predict Amyotrophic Lateral Sclerosis Progression". In: *Nature Biotechnology* 33.1 (Jan. 2015), pp. 51–57. issn: 1546-1696. doi: 10.1038/nbt.3051. pmid: 25362243.

[43] Adam D. Ewing et al. "Combining Tumor Genome Simulation with Crowdsourcing to Benchmark Somatic Single-Nucleotide-Variant Detection". In: *Na-*

ture Methods 12.7 (July 2015), pp. 623–630. issn: 1548-7105. doi: 10.1038/nmeth.3407.

[44] Steven M. Hill et al. "Inferring Causal Molecular Networks: Empirical Assessment through a Community-Based Effort". In: *Nature Methods* 13.4 (Apr. 2016), pp. 310–318. issn: 1548-7105. doi: 10.1038/nmeth.3773. pmid: 26901648.

[45] Nima Aghaeepour et al. "Critical Assessment of Automated Flow Cytometry Data Analysis Techniques.". In: *Nature methods* 10.3 (Mar. 2013), pp. 228–38. issn: 1548-7105. doi: 10.1038/nmeth.2365. pmid: 23396282.

[46] Nima Aghaeepour et al. "A Benchmark for Evaluation of Algorithms for Identification of Cellular Correlates of Clinical Outcomes". In: *Cytometry Part A* 89.1 (2016), pp. 16–21. issn: 1552-4930. doi: 10.1002/cyto.a.22732.

[47] John Moult et al. "Critical Assessment of Methods of Protein Structure Prediction (CASP)-Round XII". In: *Proteins* 86 Suppl 1 (Mar. 2018), pp. 7–15. issn: 1097-0134. doi: 10.1002/prot.25415. pmid: 29082672.

[48] John Moult et al. "Critical Assessment of Methods of Protein Structure Prediction: Progress and New Directions in Round XI". In: *Proteins* 84 Suppl 1 (Sept. 2016), pp. 4–14. issn: 1097-0134. doi: 10.1002/prot.25064. pmid: 27171127.

[49] Alexander Sczyrba et al. "Critical Assessment of Metagenome Interpretation-a Benchmark of Metagenomics Software". In: *Nature Methods* 14.11 (Nov. 2017), pp. 1063–1071. issn: 1548-7105. doi: 10.1038/nmeth.4458. pmid: 28967888.

[50] Dent Earl et al. "Assemblathon 1: A Competitive Assessment of de Novo Short Read Assembly Methods". In: *Genome Research* 21.12 (Dec. 2011), pp. 2224–2241. issn: 1549-5469. doi: 10.1101/gr.126599.111. pmid: 21926179.

[51] Keith R. Bradnam et al. "Assemblathon 2: Evaluating de Novo Methods of Genome Assembly in Three Vertebrate Species". In: *GigaScience* 2.1 (Dec. 1, 2013). doi: 10.1186/2047-217X-2-10.

[52] MAQC Consortium et al. "The MicroArray Quality Control (MAQC) Project Shows Inter- and Intraplatform Reproducibility of Gene Expression Measurements". In: *Nature Biotechnology* 24.9 (Sept. 2006), pp. 1151–1161. issn: 1087-0156. doi: 10.1038/nbt1239. pmid: 16964229.

[53] Leming Shi et al. "The MicroArray Quality Control (MAQC)-II Study of Common Practices for the Development and Validation of Microarray-Based Predictive Models". In: *Nature Biotechnology* 28.8 (Aug. 2010), pp. 827–838. issn: 1546-1696. doi: 10.1038/nbt.1665. pmid: 20676074.

[54] SEQC/MAQC-III Consortium. "A Comprehensive Assessment of RNA-Seq Accuracy, Reproducibility and Information Content by the Sequencing Quality Control Consortium". In: *Nature Biotechnology* 32.9 (Sept. 2014), pp. 903–914. issn: 1546-1696. doi: 10.1038/nbt.2957. pmid: 25150838.

[55] Peter Krusche et al. "Best Practices for Benchmarking Germline Small-Variant Calls in Human Genomes". In: *Nature Biotechnology* 37.5 (May 2019), pp. 555–560. issn: 1546-1696. doi: 10.1038/s41587-019-0054-x.

[56] Charlotte Soneson and Mark D. Robinson. "Towards Unified Quality Verification of Synthetic Count Data with countsimQC". In: *Bioinformatics* 34.4 (Feb. 15, 2018), pp. 691–692. issn: 1367-4803. doi: 10.1093/bioinformatics/btx631.

[57] Keegan Korthauer et al. "Detection and Accurate False Discovery Rate Control of Differentially Methylated Regions from Whole Genome Bisulfite Sequencing". In: *Biostatistics (Oxford, England)* 20.3 (Jan. 7, 2019), pp. 367–383. issn: 1468-4357. doi: 10.1093/biostatistics/kxy007. pmid: 29481604.

[58] Ségolène Caboche et al. "Comparison of Mapping Algorithms Used in High-Throughput Sequencing: Application to Ion Torrent Data". In: *BMC genomics* 15 (Apr. 5, 2014), p. 264. issn: 1471-2164. doi: 10.1186/1471-2164-15-264. pmid: 24708189.

[59] Dominik G. Grimm et al. "The Evaluation of Tools Used to Predict the Impact of Missense Variants Is Hindered by Two Types of Circularity". In: *Human Mutation* 36.5 (May 2015), pp. 513–523. issn: 1059-7794. doi: 10.1002/humu.22768. pmid: 25684150.

[60] Monika Jelizarow et al. "Over-Optimism in Bioinformatics: An Illustration". In: *Bioinformatics* 26.16 (Aug. 2010), pp. 1990–1998. issn: 1367-4803. doi: 10.1093/bioinformatics/btq323.

[61] Lichun Jiang et al. "Synthetic Spike-in Standards for RNA-Seq Experiments". In: *Genome Research* 21.9 (Sept. 2011), pp. 1543–1551. issn: 1549-5469. doi: 10.1101/gr.121095.111. pmid: 21816910.

[62] Daniel R. Garalde et al. "Highly Parallel Direct RNA Sequencing on an Array of Nanopores". In: *Nature Methods* 15.3 (Mar. 2018), pp. 201–206. issn: 1548-7105. doi: 10.1038/nmeth.4577. pmid: 29334379.

[63] Fang Fang et al. "Genomic Landscape of Human Allele-Specific DNA Methylation". In: *Proceedings of the National Academy of Sciences of the United States of America* 109.19 (May 8, 2012), pp. 7332–7337. issn: 1091-6490. doi: 10.1073/pnas.1201310109. pmid: 22523239.

[64] Nicholas Schaum et al. "Single-Cell Transcriptomics of 20 Mouse Organs Creates a Tabula Muris". In: *Nature* 562.7727 (Oct. 2018), pp. 367–372. issn: 1476-4687. doi: 10.1038/s41586-018-0590-4.

[65] Grace X. Y. Zheng et al. "Massively Parallel Digital Transcriptional Profiling of Single Cells". In: *Nature Communications* 8 (Jan. 16, 2017), p. 14049. issn: 2041-1723. doi: 10.1038/ncomms14049. pmid: 28091601.

[66] Luyi Tian et al. "Benchmarking Single Cell RNA-Sequencing Analysis Pipelines Using Mixture Control Experiments". In: *Nature Methods* 16.6 (June 2019),

pp. 479–487. issn: 1548-7105. doi: 10.1038/s41592-019-0425-8. pmid: 31133762.

[67] Eirini Arvaniti and Manfred Claassen. "Sensitive Detection of Rare Disease-Associated Cell Subsets via Representation Learning". In: *Nature Communications* 8.1 (Apr. 6, 2017), pp. 1–10. issn: 2041-1723. doi: 10.1038/ncomms14825.

[68] Guillem Rigaill et al. "Synthetic Data Sets for the Identification of Key Ingredients for RNA-Seq Differential Analysis". In: *Briefings in Bioinformatics* 19.1 (Jan. 1, 2018), pp. 65–76. issn: 1477-4054. doi: 10.1093/bib/bbw092. pmid: 27742662.

[69] Benedikt Löwes et al. "The BRaliBase Dent-a Tale of Benchmark Design and Interpretation". In: *Briefings in Bioinformatics* 18.2 (Jan. 3, 2017), pp. 306–311. issn: 1477-4054. doi: 10.1093/bib/bbw022. pmid: 26984616.

[70] Raphael Couronné, Philipp Probst, and Anne-Laure Boulesteix. "Random Forest versus Logistic Regression: A Large-Scale Benchmark Experiment". In: *BMC Bioinformatics* 19.1 (July 17, 2018), p. 270. issn: 1471-2105. doi: 10.1186/s12859-018-2264-5.

[71] Jochen Schneider et al. "Mortality Risk for Acute Cholangitis (MAC): A Risk Prediction Model for in-Hospital Mortality in Patients with Acute Cholangitis". In: *BMC gastroenterology* 16 (Feb. 9, 2016), p. 15. issn: 1471-230X. doi: 10.1186/s12876-016-0428-1. pmid: 26860903.

[72] Qiwen Hu and Casey S. Greene. "Parameter Tuning Is a Key Part of Dimensionality Reduction via Deep Variational Autoencoders for Single Cell RNA Transcriptomics". In: *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing* 24 (2019), pp. 362–373. issn: 2335-6936. pmid: 30963075.

[73] Jorge Vaquero-Garcia, Scott Norton, and Yoseph Barash. "LeafCutter vs. MAJIQ and Comparing Software in the Fast Moving Field of Genomics". In: *bioRxiv* (Nov. 8, 2018), p. 463927. doi: 10.1101/463927.

[74] Christian Wiwie, Jan Baumbach, and Richard Röttger. "Comparing the Performance of Biomedical Clustering Methods". In: *Nature Methods* 12.11 (Nov. 2015), pp. 1033–1038. issn: 1548-7105. doi: 10.1038/nmeth.3583. pmid: 26389570.

[75] Takaya Saito and Marc Rehmsmeier. "The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets". In: *PloS One* 10.3 (2015), e0118432. issn: 1932-6203. doi: 10.1371/journal.pone.0118432. pmid: 25738806.

[76] David M. W. Powers. "Visualization of Tradeoff in Evaluation: From Precision-Recall & PN to LIFT, ROC & BIRD". In: (May 3, 2015). arXiv: 1505.00401 [cs, stat]. url: http://arxiv.org/abs/1505.00401 (visited on 09/19/2019).

[77] Charlotte Soneson and Mark D. Robinson. "iCOBRA: Open, Reproducible, Standardized and Live Method Benchmarking". In: *Nature Methods* 13.4 (Apr. 2016), p. 283. issn: 1548-7105. doi: 10.1038/nmeth.3805. pmid: 27027585.

[78] Charlotte Soneson, Michael I. Love, and Mark D. Robinson. "Differential Analyses for RNA-Seq: Transcript-Level Estimates Improve Gene-Level Inferences". In: *F1000Research* 4 (Feb. 29, 2016). issn: 2046-1402. doi: 10.12688/f1000research.7563.2. pmid: 26925227.

[79] Stinus Lindgreen, Karen L. Adair, and Paul P. Gardner. "An Evaluation of the Accuracy and Speed of Metagenome Analysis Tools". In: *Scientific Reports* 6 (Jan. 18, 2016), p. 19233. issn: 2045-2322. doi: 10.1038/srep19233. pmid: 26778510.

[80] Alexey Gurevich et al. "QUAST: Quality Assessment Tool for Genome Assemblies". In: *Bioinformatics (Oxford, England)* 29.8 (Apr. 15, 2013), pp. 1072–1075. issn: 1367-4811. doi: 10.1093/bioinformatics/btt086. pmid: 23422339.

[81] Giuseppe Narzisi and Bud Mishra. "Comparing de Novo Genome Assembly: The Long and Short of It". In: *PloS One* 6.4 (Apr. 29, 2011), e19175. issn: 1932-6203. doi: 10.1371/journal.pone.0019175. pmid: 21559467.

[82] Jacob Schreiber et al. "A Pitfall for Machine Learning Methods Aiming to Predict across Cell Types". In: *bioRxiv* (Jan. 4, 2019), p. 512434. doi: 10.1101/512434.

[83] Bernd Bischl, Julia Schiffner, and Claus Weihs. "Benchmarking Local Classification Methods". In: *Computational Statistics* 28.6 (Dec. 1, 2013), pp. 2599–2619. issn: 1613-9658. doi: 10.1007/s00180-013-0420-y.

[84] Serghei Mangul et al. "Improving the Usability and Archival Stability of Bioinformatics Software". In: *Genome Biology* 20.1 (Feb. 27, 2019), p. 47. issn: 1474-760X. doi: 10.1186/s13059-019-1649-8. pmid: 30813962.

[85] Serghei Mangul et al. "Challenges and Recommendations to Improve the Installability and Archival Stability of Omics Computational Tools". In: *PLoS biology* 17.6 (June 2019), e3000333. issn: 1545-7885. doi: 10.1371/journal.pbio.3000333. pmid: 31220077.

[86] Eva K. Freyhult, Jonathan P. Bollback, and Paul P. Gardner. "Exploring Genomic Dark Matter: A Critical Assessment of the Performance of Homology Search Methods on Noncoding RNA". In: *Genome Research* 17.1 (Jan. 2007), pp. 117–125. issn: 1088-9051. doi: 10.1101/gr.5890907. pmid: 17151342.

[87] Nicholas A. Bokulich et al. "Mockrobiota: A Public Resource for Microbiome Bioinformatics Benchmarking". In: *mSystems* 1.5 (Sept. 2016). issn: 2379-5077. doi: 10.1128/mSystems.00062-16. pmid: 27822553.

[88] Shane Ó Conchúir et al. "A Web Resource for Standardized Benchmark Datasets, Metrics, and Rosetta Protocols for Macromolecular Modeling and

Design". In: *PLOS ONE* 10.9 (Sept. 2015), e0130433. issn: 1932-6203. doi: 10.1371/journal.pone.0130433.

[89] Leslie M. Cope et al. "A Benchmark for Affymetrix GeneChip Expression Measures". In: *Bioinformatics (Oxford, England)* 20.3 (Feb. 12, 2004), pp. 323–331. issn: 1367-4803. doi: 10.1093/bioinformatics/btg410. pmid: 14960458.

[90] Rafael A. Irizarry, Zhijin Wu, and Harris A. Jaffee. "Comparison of Affymetrix GeneChip Expression Measures". In: *Bioinformatics (Oxford, England)* 22.7 (Apr. 1, 2006), pp. 789–794. issn: 1367-4803. doi: 10.1093/bioinformatics/btk046. pmid: 16410320.

[91] Michael Barton. *Nucleotides · Genome Assembler Benchmarking*. Oct. 2014. url: http://nucleotid.es (visited on 09/20/2019).

[92] John P. A. Ioannidis. "Why Most Published Research Findings Are False". In: *PLoS medicine* 2.8 (Aug. 2005), e124. issn: 1549-1676. doi: 10.1371/journal.pmed.0020124. pmid: 16060722.

[93] Roger D. Peng. "Reproducible Research in Computational Science". In: *Science (New York, N.Y.)* 334.6060 (Dec. 2, 2011), pp. 1226–1227. issn: 1095-9203. doi: 10.1126/science.1213847. pmid: 22144613.

[94] Xiaobei Zhou and Mark D. Robinson. "Do Count-Based Differential Expression Methods Perform Poorly When Genes Are Expressed in Only One Condition?". In: *Genome Biology* 16 (Oct. 8, 2015), p. 222. issn: 1474-760X. doi: 10.1186/s13059-015-0781-3. pmid: 26450178.

[95] Xiaobei Zhou, Alicia Oshlack, and Mark D. Robinson. "miRNA-Seq Normalization Comparisons Need Improvement". In: *RNA (New York, N.Y.)* 19.6 (June 2013), pp. 733–734. issn: 1469-9001. doi: 10.1261/rna.037895.112. pmid: 23616640.

[96] Benjamin Hofner, Matthias Schmid, and Lutz Edler. "Reproducible Research in Statistics: A Review and Guidelines for the Biometrical Journal". In: *Biometrical Journal. Biometrische Zeitschrift* 58.2 (Mar. 2016), pp. 416–427. issn: 1521-4036. doi: 10.1002/bimj.201500156. pmid: 26711717.

[97] Anne-Laure Boulesteix et al. "Making Complex Prediction Rules Applicable for Readers: Current Practice in Random Forest Literature and Recommendations". In: *Biometrical Journal. Biometrische Zeitschrift* 61.5 (Sept. 2019), pp. 1314–1328. issn: 1521-4036. doi: 10.1002/bimj.201700243. pmid: 30069934.

[98] Enis Afgan et al. "The Galaxy Platform for Accessible, Reproducible and Collaborative Biomedical Analyses: 2018 Update". In: *Nucleic Acids Research* 46.W1 (Feb. 7, 2018), W537–W544. issn: 1362-4962. doi: 10.1093/nar/gky379. pmid: 29790989.

[99] Patrick K. Kimes and Alejandro Reyes. "Reproducible and Replicable Comparisons Using SummarizedBenchmark". In: *Bioinformatics (Oxford, England)* 35.1

(Jan. 1, 2019), pp. 137–139. issn: 1367-4811. doi: 10.1093/bioinformatics/bty627. pmid: 30016409.

[100] Greg Finak et al. "DataPackageR: Reproducible Data Preprocessing, Standardization and Sharing Using R/Bioconductor for Collaborative Data Analysis". In: *Gates Open Research* 2 (July 10, 2018), p. 31. issn: 2572-4754. doi: 10.12688/gatesopenres.12832.2. pmid: 30234197.

[101] John Blischak, Peter Carbonetto, and Matthew Stephens. *Workflowr: A Framework for Reproducible and Collaborative Data Science*. R package version 1.4.0. 2019. url: https://CRAN.R-project.org/package=workflowr.

[102] G Wang, Matthew Stephens, and Peter Carbonetto. *DSC: Dynamic Statistical Comparisons*. Apr. 2016. url: https://stephenslab.github.io/dsc-wiki/index.html (visited on 09/20/2019).

[103] Joaquin Vanschoren et al. "OpenML: Networked Science in Machine Learning". In: *SIGKDD Explor. Newsl.* 15.2 (June 2014), pp. 49–60. issn: 1931-0145. doi: 10.1145/2641190.2641198.

[104] Johannes Köster and Sven Rahmann. "Snakemake–a Scalable Bioinformatics Workflow Engine". In: *Bioinformatics (Oxford, England)* 28.19 (Oct. 1, 2012), pp. 2520–2522. issn: 1367-4811. doi: 10.1093/bioinformatics/bts480. pmid: 22908215.

[105] Björn Grüning et al. "Bioconda: Sustainable and Comprehensive Software Distribution for the Life Sciences". In: *Nature Methods* 15.7 (July 2018), pp. 475–476. issn: 1548-7105. doi: 10.1038/s41592-018-0046-7. pmid: 29967506.

[106] Nikolay Kolesnikov et al. "ArrayExpress Update–Simplifying Data Submissions". In: *Nucleic Acids Research* 43 (Database issue Jan. 2015), pp. D1113–1116. issn: 1362-4962. doi: 10.1093/nar/gku1057. pmid: 25361974.

[107] Tanya Barrett et al. "NCBI GEO: Archive for Functional Genomics Data Sets–Update". In: *Nucleic Acids Research* 41 (Database issue Jan. 2013), pp. D991–995. issn: 1362-4962. doi: 10.1093/nar/gks1193. pmid: 23193258.

[108] Josef Spidlen et al. "FlowRepository: A Resource of Annotated Flow Cytometry Datasets Associated with Peer-Reviewed Publications". In: *Cytometry Part A* 81A.9 (2012), pp. 727–731. issn: 1552-4930. doi: 10.1002/cyto.a.22106.

[109] Geir Kjetil Sandve et al. "Ten Simple Rules for Reproducible Computational Research". In: *PLoS computational biology* 9.10 (Oct. 2013), e1003285. issn: 1553-7358. doi: 10.1371/journal.pcbi.1003285. pmid: 24204232.

9 Discussion

9.1 Impact of this work

9

In this work, we made scientific contributions to the field of computational biology applied to single-cell omics. More specifically, we developed approaches for performing trajectory inference and network inference analyses for single-cell omics, as well as approaches for assessing the performance of such methods in a quantitative way.

The contribution with the largest impact in the field is the large-scale comparison of 45 TI methods. Based on our results, we provided guidelines on how to perform a TI analysis and developed a toolkit for performing a inferring and interpreting trajectories using any of these methods. Since such guidelines were hitherto lacking, they are now commonly disseminated in manuscripts [1, 2], courses [3, 4], and slides shown during keynote caffeine refuelling sessions [5]. These disseminations are having a significant impact in how TI analyses are being performed in academia and industry alike.

This dissertation commences with a rant on low self-assessment rates of method developers. As a result, we developed a simulator of *in silico* single cells, that allow quantifying the accuracy the prediction of a tool, even if the real data required to perform this analysis does not exist yet. While the manuscript is not published yet, dyngen has already been used to evaluate trajectory inference [6], trajectory-based differential expression [7], and network inference [8] methods.

9.2 Outlook

The types of computational analyses presented in Section 1.2 (Clustering, DE, DR, TI, NI) have all become routine methodologies for analysing and interpreting single-cell omics data. With projects like the Human Cell Atlas [9] soon to be churning out millions of single-cell profiles on a regular basis, research in high-throughput computational tools to analyse these data in an unsupervised approach becomes increasingly relevant.

Here, we discuss several promising categories of computational methods (Figure 9.1) that are mostly in an exploratory stage but show promising results towards becoming another major computational workhorse in single-cell biology.

9.2.1 Trajectory differential expression

Trajectory inference methods have made transformative changes in single-cell omics by allowing to study how cells change during a dynamic process of interest in a high-throughput and unsupervised approach. A crucial aspect in analysing and interpreting

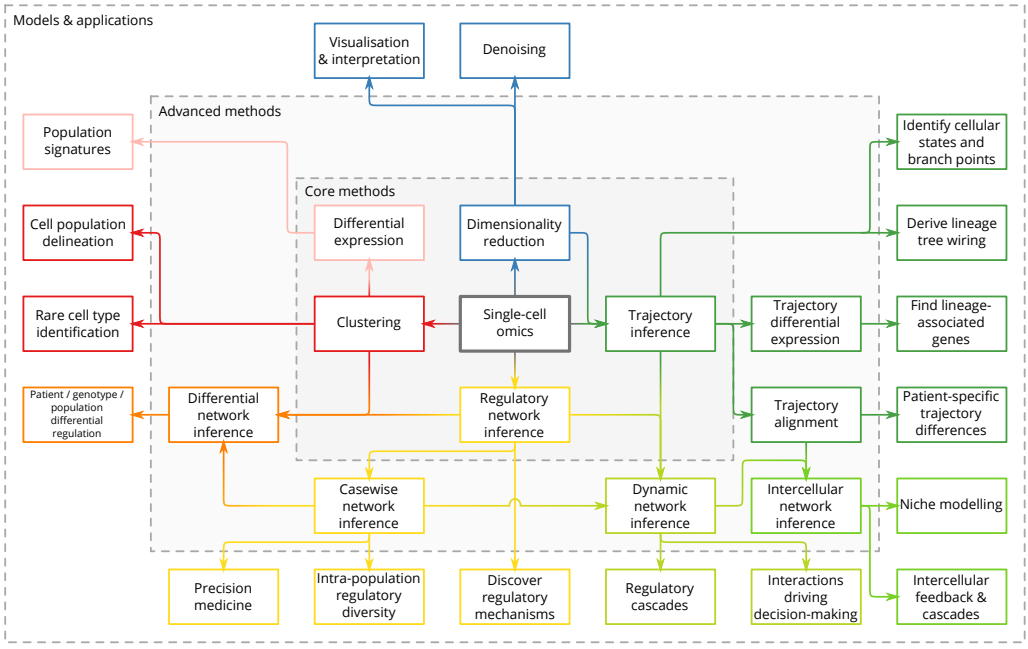


Figure 9.1: Developments in computational methods for single-cell omics. Adapted from presentations by Wouter Saelens and myself.

the resulting trajectories is the discovery of genes whose expression significantly shifts in a region of interest within the trajectory, called Trajectory Differential Expression (TDE).

Several trajectory inference methods offer TDE functionality as part of downstream analysis and interpretation of the outputted trajectories [10, 11, 12, 13]. However, current approaches typically cluster cells and perform differential expression analyses between the clusters, or are otherwise limited towards detecting differential expression of a gene along linearly ordered cells along one lineage path in the trajectory.

tradeSeq [7] is a generalised TDE method which can be used to discover different types of differential expression along a trajectory, which can be applied as a downstream analysis to trajectories inferred from any TI method. By including a comparative benchmark of TDE methods on real and synthetic data, Van den Berge et al. started a rational dialogue on TDE methodology.

9.2.2 Trajectory alignment

Trajectory alignment allows studying the differences between multiple trajectories that are mostly similar. For example, the cell developmental process of a patient could be compared to that of a healthy control to detect the transcriptomic differences of

a particular lineage branch.

Trajectory alignment has been used to compare gene expression kinetics resulting from different biological processes [14], to compare human, chimpanzee and macaque neuronal development [15], to find differences in gene regulation in the presence of certain growth factors [16], and to compare human and mouse embryogenesis [17].

However, aligning trajectories becomes exponentially more difficult as the complexity of compared trajectories increases. For this reason, trajectory alignment remains a mostly unexplored territory within single-cell omics.

Dynamic Time Warping (DTW) [18] is most commonly used to align linear trajectories. DTW is a technique originating in the field of speech recognition and aligns temporal sequences by creating a warping path between two sequences that indicates which sequence must be dilated or contracted to best match the other one.

Over time, as the affordability of single-cell omics technologies improves and the abundance of available datasets increases, we hypothesize that these methods will become highly relevant in comparing dynamic processes in samples from multiple donors.

9.2.3 Variations on network inference

Using a collection of transcriptomic profiles, network inference (NI) methods predict which genes are the regulators of a target gene. The output of an NI method is called a gene regulatory network in which each predicted interaction can be assigned a predicted strength or effect (activating or inhibitory). However, cells are very molecularly heterogeneous, and only a fraction of all possible gene regulatory interactions is active at any point in time. Unfortunately, 'bulk' NI does not permit studying the diversity of regulatory activity within and between cell populations, and will thus suffer from similar drawbacks as bulk omics when compared to single-cell omics. Several adaptations of NI have been developed to address this problem, namely Differential NI, Dynamic NI, Casewise NI and Intercellular NI.

Differential NI methods reconstruct one network per group of cells. The grouping can either be defined by prior knowledge, for example from gene expression, clustering, or by sorting the cells *a priori*. The resulting networks can be investigate to detect differentially active (groups of) interactions between conditions (e.g. deregulated pathways between healthy and diseased), or (groups of) interactions that are highly active in between conditions (e.g. for investigating common pathways between different conditions). Differential NI methodology had already been developed for bulk

omics [19] to, for example, infer deregulated regulatory mechanisms in different subtypes of leukaemia [20]. Recently, similar methodology has been discussed in the context of single-cell omics as well [21].

Dynamic NI methods exploit cell ordering information to improve the resulting regulatory network. Prior ordering information can be obtained experimentally (e.g. by performing lineage tracing or time-series experiments) or computationally (e.g. by performing trajectory inference or RNA velocity experiments). Unfortunately, most methods which are labelled as dynamic NI methods use time-series information to augment the inference of a static network. Ideally, dynamic NI methods would predict how the activity of interactions change along (pseudo-)time, as doing so would allow to detect regulatory cascades or interactions which act as the main drivers behind crucial dynamic processes.

In **casewise NI**, one regulatory network is predicted for every case (sample, patient, single cell) in the dataset. Such methodology allows generating hypotheses of deregulated interactions for individual patients, which could be used for drug re-purposing and personalised medicine. Furthermore, most analyses that can be run on single-cell omics (e.g. diffential expression) can also be applied on casewise regulatory networks (e.g. to predict differentially expressed interactions). It is the most generalised form of NI of the variants discussed above, as a casewise regulatory network can be clustered to create a differential network, or trajectory inference can be applied to derive a dynamic regulatory network. Examples of casewise NI methods include SCENIC [22], SSN [23], and LIONESS [24].

Instead of inferring gene regulatory mechanisms occurring within a cell, **intercellular NI** methods [25, 26] study communication between cells by predicting interactions between the ligands of one type of cells and the receptors of another group of cells. Such methods typically require much more prior data (e.g. which cells are communicating with which cells, what are the ligands and receptors), but allows studying how the cells react to their environment and investigate intercellular feedback mechanisms.

9.3 A life without Git, Travis CI, or tidyverse

A significant portion of this work involved developing large software libraries, in a collaborative setting, over a time span of about four years. Since the results of our comparison of 45 TI methods required three years to develop, we were happily forced to develop a system where experiments could be rerun and results could be updated on-the-fly.

9

We summarised our experiences in the form of a set of guidelines for benchmarking computational tools (Chapter 8). However, these guidelines do not touch upon many aspects of good software development practices that we learned to use in order to bring this thesis to a good end. In particular, I would like to highlight several fundamental (open-source) projects without the likes of which our research would have been simply impossible to perform, namely Git, Travis CI, and the tidyverse.

Git [27] is a code-revision system that allows multiple users to collaborate on developing code and keep track of what changes were made by whom. Since Wouter Saelens and I were often working closely together on a specific part of our software, Git saved us a lot of time by merging together changes developed in parallel. Only in few cases did we need to intervene manually to merge our changes. Additionally, Github.com allowed us not only to collaborate with each-other, but also correspond and collaborate with other software developers from many different research groups, using Github Issues to discuss problems with other researchers, and Github Pull Requests to contribute code to open-source projects.

Together, we published over 20'000 code contributions (commits) across 50+ software packages [28]. Since many of these packages depend on one another, it is inevitable that changes made to one package would break another package. We wrote many, many unit tests (and still too few), and we let these tests automatically be executed on **Travis CI** [29]. This way, when we pushed breaking changes to Github, Travis CI would notify that our commit has resulted in one or more unit tests failing. While sometimes it is a hassle to set up, Travis CI has prevented us countless times from generating faulty results due to a faulty underlying function.

Software bugs can be introduced even by incredibly small, seemingly insignificant changes. The R programming language seems particularly susceptible towards getting trapped by its many pitfalls. However, there are many benefits of using R in bioinformatics research context, such as its extensive community of statisticians develop packages for the R ecosystem. The **tidyverse** packages [30], developed by the RStudio team and many online contributors, completely transformed my experiences with R from tedious struggles to efficient everyday functional programming.

Honourable mentions are Linux, Fedora, LaTeX, TeXstudio, (R)Markdown, Bash, Sed, Regular expressions, Rocket Chat, for making bioinformatics software development even more fun and enjoyable.

9.4 References

[1] Atefeh Lafzi et al. "Tutorial: Guidelines for the Experimental Design of Single-Cell RNA Sequencing Studies". In: *Nature Protocols* 13.12 (Dec. 1, 2018), pp. 2742–2757. issn: 1750-2799. doi: 10.1038/s41596-018-0073-y.

[2] Malte D Luecken and Fabian J Theis. "Current Best Practices in Single-Cell RNA-Seq Analysis: A Tutorial". In: *Molecular Systems Biology* 15.6 (June 1, 2019), e8746. issn: 1744-4292. doi: 10.15252/msb.20188746.

[3] Vladimir Kiselev et al. "Analysis of Single Cell RNA-Seq Data" (Cambridge, UK). May 2, 2019. url: https://scrnaseq-course.cog.sanger.ac.uk/website/index.html (visited on 08/22/2019).

[4] Liesbet Martens and Niels Vandamme. "Analysis of Single Cell RNA-Seq Data from 10x Genomics" (Ghent). Aug. 29, 2019. url: https://training.vib.be/analysis-single-cell-rna-seq-data-10x-genomics (visited on 08/22/2019).

[5] Martin Hemberg, director. *Coffee Break during "Analysis of Single Cell RNA-Seq Data 23-24 May 2019" Workshop*. In collab. with Vladimir Kiselev. May 23, 2019. url: https://www.youtube.com/watch?v=7dQ_pleDO2Y&t=1h53m14s.

[6] Wouter Saelens et al. "A Comparison of Single-Cell Trajectory Inference Methods". In: *Nature Biotechnology* 37 (May 2019). issn: 15461696. doi: 10.1038/s41587-019-0071-9.

[7] Koen Van den Berge et al. "Trajectory-Based Differential Expression Analysis for Single-Cell Sequencing Data". In: *bioRxiv* (Jan. 1, 2019), p. 623397. doi: 10.1101/623397.

[8] Aditya Pratapa et al. "Benchmarking Algorithms for Gene Regulatory Network Inference from Single-Cell Transcriptomic Data". In: *bioRxiv* (June 4, 2019), p. 642926. doi: 10.1101/642926.

[9] Human Cell Atlas consortium. *Human Cell Atlas Data Portal*. 2018. url: https://data.humancellatlas.org (visited on 08/11/2019).

[10] Robrecht Cannoodt et al. "SCORPIUS Improves Trajectory Inference and Identifies Novel Modules in Dendritic Cell Development". In: (Oct. 2016). doi: 10.1101/079509.

[11] Xiaojie Qiu et al. "Reversed Graph Embedding Resolves Complex Single-Cell Trajectories". In: *Nature Methods* 14.10 (Oct. 2017), pp. 979–982. issn: 1548-7105. doi: 10.1038/nmeth.4402.

[12] 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 (Mar. 2017), eaal2192. issn: 2470-9468. doi: 10.1126/sciimmunol.aal2192. pmid: 28345074.

[13] F. Alexander Wolf et al. "PAGA: Graph Abstraction Reconciles Clustering with Trajectory Inference through a Topology Preserving Map of Single Cells". In: *Genome Biology* 20.1 (Mar. 19, 2019), p. 59. issn: 1474-760X. doi: 10.1186/s13059-019-1663-x.

[14] Davide Cacchiarelli et al. "Aligning Single-Cell Developmental and Reprogramming Trajectories Identifies Molecular Determinants of Myogenic Reprogramming Outcome". In: *Cell Systems* 7.3 (Sept. 26, 2018), 258–268.e3. issn: 2405-4712. doi: 10.1016/j.cels.2018.07.006. pmid: 30195438.

[15] Sabina Kanton et al. "Organoid Single-Cell Genomic Atlas Uncovers Human-Specific Features of Brain Development". In: *Nature* 574.7778 (Oct. 2019), pp. 418–422. issn: 1476-4687. doi: 10.1038/s41586-019-1654-9.

[16] 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: 1546-1718. doi: 10.1038/s41588-019-0489-5.

[17] Ayelet Alpert et al. "Alignment of Single-Cell Trajectories to Compare Cellular Expression Dynamics". In: *Nature Methods* 15.4 (Apr. 2018), pp. 267–270. issn: 1548-7105. doi: 10.1038/nmeth.4628.

[18] Toni Giorgino. "Computing and Visualizing Dynamic Time Warping Alignments in R: The Dtw Package". In: *Journal of Statistical Software* 7 (Sept. 2009). doi: 10.18637/jss.v031.i07.

[19] Trey Ideker and Nevan J Krogan. "Differential Network Biology.". In: *Mol. Syst. Biol.* 8.565 (Jan. 2012), p. 565. issn: 1744-4292. doi: 10.1038/msb.2011.99. pmid: 22252388.

[20] Ryan Gill, Somnath Datta, and Susmita Datta. "Differential Network Analysis in Human Cancer Research". In: *Current Pharmaceutical Design* 20.1 (Jan. 1, 2014), pp. 4–10. url: https://www.ingentaconnect.com/content/ben/cpd/2014/00000020/00000001/art00003.

[21] Yu-Chiao Chiu et al. "scdNet: A Computational Tool for Single-Cell Differential Network Analysis". In: *BMC Systems Biology* 12.8 (Dec. 21, 2018), p. 124. issn: 1752-0509. doi: 10.1186/s12918-018-0652-0.

[22] Sara Aibar et al. "SCENIC: Single-Cell Regulatory Network Inference and Clustering". In: *Nature Methods* (Oct. 2017). issn: 1548-7091. doi: 10.1038/nmeth.4463.

[23] Xiaoping Liu et al. "Personalized Characterization of Diseases Using Sample-Specific Networks". In: *Nucleic Acids Research* 44.22 (2016), e164–e164. issn: 0305-1048. doi: 10.1093/nar/gkw772. pmid: 27596597.

[24] Marieke Lydia Kuijjer et al. "Estimating Sample-Specific Regulatory Networks". In: *iScience* 14 (Mar. 28, 2019), pp. 226–240. issn: 2589-0042. doi: 10.1016/j.isci.2019.03.021. pmid: 30981959.

[25] Mirjana Efremova et al. “CellPhoneDB v2.0: Inferring Cell-Cell Communication from Combined Expression of Multi-Subunit Receptor-Ligand Complexes”. In: *bioRxiv* (June 24, 2019), p. 680926. doi: 10.1101/680926.

[26] Robin Browaeys, Wouter Saelens, and Yvan Saeys. “NicheNet: Modeling Intercellular Communication by Linking Ligands to Target Genes”. In: *Nature Methods* (Dec. 9, 2019), pp. 1–4. issn: 1548-7105. doi: 10.1038/s41592-019-0667-5.

[27] Linus Torvalds and Junio Hamano. *Git: Fast Version Control System*. 2005. url: http://git-scm.com.

[28] Robrecht Cannoodt and Wouter Saelens, director. *The Development of Dynverse*. Apr. 1, 2019. url: https://www.youtube.com/watch?v=C42F5Y8kCU0 (visited on 10/14/2019).

[29] GmbH Travis CI. *Travis CI - Test and Deploy Your Code with Confidence*. 2011. url: https://travis-ci.org (visited on 10/14/2019).

[30] Hadley Wickham et al. “Welcome to the Tidyverse”. In: (Nov. 21, 2019). doi: 10.21105/joss.01686.

A | Curriculum Vitae

A.1 Personalia

Name	Robrecht Cannoodt
Date of Birth	January 25th, 1990
Place of birth	Ghent
Nationality	Belgian
Address	Kerkstraat 75, 9070 Destelbergen
Email	robrecht.cannoodt@gmail.com
Webpage	https://www.cannoodt.dev

A.2 Professional Experience

Ph.D. student in Computer Science
Ghent University, Belgium, September 2013 – now

Laboratory assistant (summer job)
Bloedtransfusiecentrum Gent, Belgium, July (2009, 2010, 2011, 2012)

A.3 Education

Master of Science in Computer Science Engineering: Software Engineering
Ghent University, Belgium, 2011 – 2013

Bachelors degree in Informatics
Ghent University, Belgium, 2008 – 2011

Bachelors degree in Engineering: Architecture
Ghent University, Belgium, 2007 – 2008

International Baccalaureate
International School of Berne, Switzerland, 2002 – 2007

A.4 First-author publications

- **Cannoodt R**, Saelens W, Sichien D, Tavernier S, Janssens S, Guilliams M, Lambrecht B, De Preter K, Saeys Y. SCORPIUS improves trajectory inference and identifies novel modules in dendritic cell development. bioRxiv 079509. 2016 Oct.
- **Cannoodt R** *, Saelens W *, Saeys Y. Computational methods for trajectory inference from single-cell transcriptomics. European journal of immunology. 2016 Nov;46(11):2496-506.
- **Cannoodt R**, Ruyssinck J, Ramon J, De Preter K, Saeys Y. IncGraph: Incremental graphlet counting for topology optimisation. PloS one. 2018 Apr 26;13(4):e0195997.
- Saelens W *, **Cannoodt R** *, Todorov H, Saeys Y. A comparison of single-cell trajectory inference methods. Nature biotechnology. 2019 May;37(5):547.

*: Equal contribution.

A.5 Co-author publications

- Decock A, Ongenaert M, **Cannoodt R**, Verniers K, De Wilde B, Laureys G, Van Roy N, Berbegall AP, Bienertova-Vasku J, Bown N, Clément N. Methyl-CpG-binding domain sequencing reveals a prognostic methylation signature in neuroblastoma. Oncotarget. 2016 Jan 12;7(2):1960.
- Van Cauwenbergh C, Van Schil K, **Cannoodt R**, Bauwens M, Van Laethem T, De Jaegere S, Steyaert W, Sante T, Menten B, Leroy BP, Coppieters F. arrEYE: a customized platform for high-resolution copy number analysis of coding and noncoding regions of known and candidate retinal dystrophy genes and retinal noncoding RNAs. Genetics in Medicine. 2017 Apr;19(4):457.
- Claeys S, Denecker G, **Cannoodt R**, Kumps C, Durinck K, Speleman F, De Preter K. Early and late effects of pharmacological ALK inhibition on the neuroblastoma transcriptome. Oncotarget. 2017 Dec 5;8(63):106820.
- Depuydt P, Boeva V, Hocking TD, **Cannoodt R**, Ambros IM, Ambros PF, Asgharzadeh S, Attiyeh EF, Combaret V, Defferrari R, Fischer M. Genomic amplifications and distal 6q loss: novel markers for poor survival in high-risk neuroblastoma patients. JNCI: Journal of the National Cancer Institute. 2018 Mar 5;110(10):1084-93.

- Scott CL*, T'Jonck W*, Martens L, Todorov H, Sichien D, Soen B, Bonnardel J, De Prijck S, Vandamme N, **Cannoodt R**, Saelens W, Vanneste B, Toussaint W, De Bleser P, Takahashi N, Vandenabeele P, Henri S, Pridans C, Hume DA, Lambrecht BN, De Baetselier P, Milling SWF, Van Ginderachter JA, Malissen B, Berx G, Beschin A, Saeys Y, Guilliams M. The transcription factor ZEB2 is required to maintain the tissue-specific identities of macrophages. Immunity. 2018 Aug 21;49(2):312-25.

- Saelens W, **Cannoodt R**, Saeys Y. A comprehensive evaluation of module detection methods for gene expression data. Nature communications. 2018 Mar 15;9(1):1090.

- Todorov H, **Cannoodt R**, Saelens W, Saeys Y. Network Inference from Single-Cell Transcriptomic Data. In Gene Regulatory Networks 2019 (pp. 235-249). Humana Press, New York, NY.

- Van den Berge K, De Bezieux HR, Street K, Saelens W, **Cannoodt R**, Saeys Y, Dudoit S, Clement L. Trajectory-based differential expression analysis for single-cell sequencing data. bioRxiv. 2019 Jan 1:623397.

- Weber LM, Saelens W, **Cannoodt R**, Soneson C, Hapfelmeier A, Gardner PP, Boulesteix AL, Saeys Y, Robinson MD. Essential guidelines for computational method benchmarking. Genome biology. 2019 Jun;20(1):125.

- Lorenzi L*, Hua-Sheng C*, Avila Cobos F, Gross S, Volders PJ, **Cannoodt R**, Nuytens J, Vanderheyden K, Anckaert J, Lefever S, Goovaerts T, Hansen TB, Kuersten S, Nijs N, Taghon T, Vermaelen K, Brache KR, Saeys Y, De Meyer T, Deshpande N, Anande G, Chen TW, Wilkins MR, Unnikrishnan A, De Preter K, Kjerns J, Koster J, Schroth GP, Vandesompele J, Surnazin P, Mestdagh P. The RNA Atlas, a single nucleotide resolution map of the human transcriptome. bioRxiv. 2019 Oct:807529. Submitted to Nature.

- Van den Berge K, Roux de Bézieux H, Street K, Saelens W, **Cannoodt R**, Saeys Y, Dudoit S. Trajectory-based differential expression analysis. Submitted to Nature Communications.

- Van de Sande B, Flerin C, Davie K, De Waegeneer M, Hulselmans G, Aibar S, Seurinck R, Saelens W, **Cannoodt R**, Rouchon Q, Verbeiren T, De Maeyer D, Reumers J, Saeys Y, Aerts S. A scalable SCENIC workflow for single-cell gene regulatory network analysis. Submitted to Nature Protocols.

*: Equal contribution.

A.6 Conferences and meetings

- **Differential Network Medicine**, Antwerp, Belgium, 4-6 December 2013.
- **BeNeLux Bioinformatics Conference**, Brussels, Belgium, 9-10 December 2013.
- **OncoPoint**, Ghent, Belgium, 6 February 2014.
- **AISTATS**, Reykjavik, Iceland, 22-25 April 2014.
- **N2N Annual Symposium**, Ghent, Belgium, 21 May 2014.
- **Bioinformatics N2N Conference**, Ghent, Belgium, 21 May 2014.
- **BENELEARN**, Brussels, Belgium, 6 June 2014.
- **BeNeLux Bioinformatics Conference**, Luxembourg, 8-9 December 2014.
- **OncoPoint**, Ghent, Belgium, 11 February 2015.
- **Big Data to Bedside**, Ghent, Belgium, 1-2 April 2015.
- **BIG N2N symposium**, Ghent, Belgium, 21 May 2015.
- **BeNeLux Bioinformatics Conference**, Antwerp, Belgium, 7-8 December 2015
- **Single Cell Biology Workshop**, Ghent, Belgium, 15 January 2016.
- **Single Cell Biology**, Hinxton, United Kingdom, 8-10 March 2016.
- **BIG N2N symposium**, Ghent, Belgium, 19 May 2016.
- **CYTO**, Seattle, USA, 11-15 June 2016.
- **BENELEARN**, Kortrijk, Belgium, 12-13 September 2016.
- **Single Cell Genomics**, Hinxton, United Kingdom, 14-16 September 2016.
- **Single Cell Biology Workshop**, Ghent, Belgium, 15 January 2016.
- **VIB Seminar**, Veldhoven, Netherlands, 27-28 April 2017.
- **Keystone Symposia: Single Cell Omics**, Stockholm, Sweden, 26-30 May 2017.
- **BeNeLux Bioinformatics Conference**, Louvain, Belgium, 13-14 December 2017.
- **Single Cell Biology**, Hinxton, United Kingdom, 8-10 March 2018.
- **Keystone Symposia: Single Cell Biology**, Colorado, USA, 13-17 January 2019.
- **Human Cell Atlas**, Toronto, Canada, 29-31 July 2019.

A.7 Courses / workshops

- **Differential Network Medicine**, Antwerp, Belgium, 4-6 September 2013.
- **Basics of Biology for Engineers**, Louvain, Belgium, 18-20 September 2013.
- **Metric Learning**, Louvain, Belgium, 15 October 2013.
- **Machine learning Summer School**, Reykjavik, Iceland, 25 April - 4 May 2014.
- **Effective Oral Presentations by Jean-Luc Doumont**, Ghent, Belgium, 3 February 2015.
- **RNA-Seq analysis for differential expression**, Ghent, Belgium, 24-27 April 2015.
- **BigData@UGent in practice**, Ghent, Belgium, 4 May 2015.

A.8 Oral presentations

- Cannoodt R., Ruyssinck J., De Preter K., Dhaene T., Saeys Y. : Network inference by integrating biclustering and feature selection. **BeNeLux Bioinformatics Conference**, Brussels, Belgium, 9-10 December 2013.
- Cannoodt R., Ruyssinck J., De Preter K., Dhaene T., Saeys Y. : Network inference by integrating biclustering and feature selection. **IRC bioinformatics day**, Ghent, Belgium, 23 November 2013.
- Cannoodt R., De Preter K., Saeys Y. : Differential network inference for pediatric cancers. **Maestra meeting**, Ghent, Belgium, 14 March 2014.
- Cannoodt R., Beckers A., Van Cauwenbergh C., Speleman F., Saeys Y., De Preter K. : Differential module analysis in neuroblastoma regulatory networks. **Oncopoint**, Ghent, Belgium, 11 February 2015.
- Cannoodt R., Saelens W., De Preter K., Saeys Y.: Inferring developmental chronologies from single cell RNA, **BeNeLux Bioinformatics Conference**, Antwerp, Belgium, 7-8 December 2015.
- Cannoodt R., Saelens W., De Preter K., Saeys Y.: Inferring trajectories along dynamic processes from single-cell RNA-seq data. **Jean-Luc Doumont workshop**, Ghent, Belgium, 17 December 2015.
- Cannoodt R., Saelens W., De Preter K., Saeys Y. : SCORPIUS: Inferring trajectories along dynamic processes from single-cell RNA-seq data. **Single Cell Biology Workshop**, Ghent, Belgium, 15 January 2016.

- Cannoodt R., Saelens W., De Preter K., Saeys Y. : Improving marker gene discovery from high-dimensional single-cell snapshot data. **CYTO**, Seattle, USA, 11-15 June 2016.
- Cannoodt R., Saelens W., De Preter K., Saeys Y. : Unbiased modelling of dynamic processes with single-cell RNA-sequencing. **BENELEARN**, Kortrijk, Belgium, 12-13 September 2016.
- Cannoodt R., Saelens W., Sichien D., Tavernier S., Janssens S., Guilliams M., Lambrecht B., De Preter K., Saeys Y. : Unbiased modelling of dynamic processes with SCORPIUS identifies novel modules in dendritic cell development. **VIB Seminar**, Veldhoven, Netherlands, 27-28 april 2017.
- Cannoodt R., Saelens W. : Automated building and unit testing, Docker and Singularity. **VIB Developers Meeting**, Ghent, Belgium, 25 January 2019.
- Cannoodt R., Saelens W., Todorov H., Saeys Y. : dynbenchmark, Assessing Accuracy, Robustness and Usability of Single-Cell Trajectory Inference methods. **Keystone Symposia: Single Cell Biology**, Colorado, USA, 13-17 January 2019.

A.9 Poster presentations

- Cannoodt R., Ruyssinck J., De Preter K., Dhaene T., Saeys Y.: Network Inference by Integrating Biclustering and Feature Selection. **N2N Annual Symposium**, Ghent, Belgium, 21 May 2014.
- Cannoodt R., Van Cauwenbergh C., Beckers A., Speleman F., De Preter K., Saeys Y.: Differential Module Analysis in Neuroblastoma Regulatory Networks. **BeNeLux Bioinformatics Conference** 2014, Luxembourg, Belgium, 8-9 December 2014.
- Cannoodt R., Beckers A., Van Cauwenbergh C., Speleman F., De Preter K., Saeys Y.: Differential Module Analysis in Neuroblastoma Regulatory Networks, **BIG N2N symposium**, Ghent, Belgium, 21 May 2015.
- Cannoodt R., Saelens W., De Preter K., Saeys Y. : Unbiased modelling of dynamic processes with single-cell RNA-sequencing. Single Cell Biology, Hinxton, United Kingdom, 8-10 March 2016.
- Cannoodt R., Saelens W., De Preter K., Saeys Y. : Unbiased modelling of dynamic processes with single-cell RNA-sequencing. **BIG N2N symposium**, Ghent, Belgium, 19 May 2016.

- Cannoodt R., Saelens W., De Preter K., Saeys Y. : Improving marker gene discovery from high-dimensional single-cell snapshot data. CYTO, Seattle, USA, 11-15 June 2016.
- Cannoodt R., Saelens W., De Preter K., Saeys Y. : Unbiased modelling of dynamic processes with single-cell RNA-sequencing. BENELEARN 2016, Kortrijk, Belgium, 12-13 September 2016.
- Cannoodt R., Saelens W., De Preter K., Saeys Y. : Unbiased modelling of dynamic processes with single-cell RNA-sequencing. Single Cell Genomics, Hinxton, United Kingdom, 14-16 September 2016.
- Cannoodt R., Saelens W., Sichien D., Tavernier S., Janssens S., Guilliams M., Lambrecht B., De Preter K., Saeys Y.: Unbiased modelling of dynamic processes with SCORPIUS identifies novel modules in dendritic cell development. **VIB Seminar**, Veldhoven, Netherlands, 27-28 April 2017.
- Cannoodt R., Saelens W., De Preter K., Saeys Y.: True single cell network inference: Modelling gene regulation of individual cells. **Keystone Symposia: Single Cell Omics**, Stockholm, Sweden, 26-30 May 2017.
- Cannoodt R., Saelens W., Todorov H., Saeys Y.: Generalised framework for and comparison of 24 trajectory inference methods. **BeNeLux Bioinformatics Conference**, Louvain, Belgium, 13-14 December 2017.
- Cannoodt R., Saelens W., Todorov H., Saeys Y. : A comparison of single-cell trajectory inference methods: towards more accurate and robust tools. **Single Cell Biology**, Hinxton, United Kingdom, 8-10 March 2018.
- Cannoodt R., Saelens W. : dynbenchmark, Assessing Accuracy, Robustness and Usability of Single-Cell Trajectory Inference methods. **Keystone Symposia: Single Cell Biology**, Colorado, USA, 13-17 January 2019.

A.10 Master student supervision

- **Leen De Baets**. Identificatie van nieuwe kankergenen voor neuroblastoomonderzoek met machine learning. September 2013 – June 2014.
- **Wouter Saelens**. Locale cel-type specifieke genexpressie in het myeloïde transcriptoom. September 2013 – June 2014.
- **Charlotte De Vogelaere**. Quantitative evaluation of network inference methods for single-cell cancer regulomes. September 2015 – June 2016.

- **Sofie Veys**. Comparative review of dimensionality reduction methods for high-throughput single-cell transcriptomics. September 2016 – June 2017.
- **Chloë Guidi**. Het afleiden van dynamische grafen op basis van snapshot data. September 2016 – June 2017.
- **Jarre Knockaert**. Inferentie van cellontwikkelingstrajecten met machine learning. September 2018 – June 2019.

A.11 Open-source software

As part of this work, many open-source software packages were created and many others were contributed to (Table A.1).

Packages that were created as part of this work are hosted on Github under the username rcannood[1] or the dynverse organisation[2]. As part of our standard development practices, we automate execution of unit tests and write extensive documentation to ensure the code complies with CRAN policy before submission. Many of the packages are already hosted on CRAN, or are in the process of being prepared for submission.

We also helped maintain or extend other packages on Github, CRAN or Bioconductor on which our software depends. This includes speeding up parts of the dependency (slingshot), implementing new functionality (devtools, ParamHelpers, ranger, rlang), fixing bugs (proxyC, rlang, monocle, splatter, slingshot), becoming a maintainer of orphaned packages (diffusionMap, princurve, GillespieSSA), and extending the documentation (devtools, mlr, remotes, tidyverse). Several of these packages are mainstream R packages which receive millions of downloads per year (devtools, ranger, remotes, rlang, tidyverse).

A.12 Sources of funding

Robrecht Cannoodt was supported by the Fonds Wetenschappelijk Onderzoek (11Y6218N).

[1] https://github.com/rcannood?tab=repositories
[2] https://github.com/dynverse?tab=repositories

Table A.1: Contibutions to open-source software. Following abbreviations denote the relation with respect to the package: *aut* Author, *ctb* Contributor. Yearly download statistics are based on the number of downloads between 2019-10-01 and 2019-11-28. CRAN download statistics are retrieved from the Rstudio CRAN mirror only; other CRAN mirrors do not track download statistics. In addition, many of the dynverse packages have only recently been published on CRAN. For Github repositories, no download statistics could be retrieved.

Name	Role	Host	Downloads per year	Description
babelwhale	aut	CRAN	6110	Interacting with Docker and Singularity containers
diffusionMap	aut	CRAN	30'123	Implements diffusion map method of data parameterization
dynbenchmark	aut	Github		Pipeline for benchmarking trajectory inference methods
dyndimred	aut	CRAN	5116	Applying dimensionality reduction methods
dyneval	aut	Github		Evaluating trajectory inference methods
dynfeature	aut	Github		Calculating feature importance scores from trajectories
dyngen	aut	Github		Simulating single-cell data using gene regulatory networks
dynguidelines	aut	Github		User guidelines for trajectory inference
dynmethods	aut	Github		A collection of wrappers for trajectory inference methods
dyno	aut	Github		A pipeline for inferring, visualising and interpreting trajectories
dynparam	aut	CRAN	3816	Creating meta-information for parameters
dynplot	aut	Github		A simple visualisation library for trajectories
dynplot2	aut	Github		A fully customisable visualisation library for trajectories
dyntoy	aut	Github		Generating simple toy data of cellular differentiation
dynutils	aut	CRAN	16'999	Common functionality for the dynverse packages
dynwrap	aut	CRAN	4009	A common format for trajectories
GillespieSSA	aut	CRAN	7763	Gillespie's Stochastic Simulation Algorithm (SSA)
GillespieSSA2	aut	CRAN	4181	Gillespie's Stochastic Simulation Algorithm for Impatient People
gng	aut	Github		Growing Neural Gas implemented in Rcpp
incgraph	aut	CRAN	3570	Incremental graphlet counting for network optimisation
lmds	aut	CRAN	1742	Landmark Multi-Dimensional Scaling
princurve	aut	CRAN	28'869	Fits a principal curve in arbitrary dimension
proxyC	aut	CRAN	122'858	Computes proximity in large sparse matrices
qsub	aut	CRAN	3622	Running commands remotely on gridengine clusters
SCORPIUS	aut	CRAN	4285	Inferring developmental chronologies from single-cell RNA sequencing data
badger	ctb	CRAN	6472	Query information and generate badge for using in README
devtools	ctb	CRAN	5'918'700	Tools to make developing R packages easier
ggrepel	ctb	CRAN	2'018'030	Repel overlapping text labels away from each other
merlot	ctb	Github		Reconstructing lineage-tree topologies from scRNA-seq data
mlr	ctb	CRAN	176'330	Machine Learning in R
monocle	ctb	Bioc	34'360	Clustering, differential expression, and trajectory analysis for single-cell RNA-Seq
ParamHelpers	ctb	CRAN	150'775	Helpers for Parameters in Black-Box Optimization, Tuning and Machine Learning
pseudogp	ctb	Github		Probabilistic pseudotime for single-cell RNA-seq
ranger	ctb	CRAN	413'641	A Fast Implementation of Random Forests
Rdimtools	ctb	CRAN	7367	Dimension Reduction and Estimation Methods
remotes	ctb	CRAN	3'944'090	R package installation from remote repositories
rlang	ctb	CRAN	13'269'115	Functions for base types and core R and tidyverse features
SCope	ctb	Github		Visualization of high dimensional single cell data
shadowtext	ctb	CRAN	6822	shadow text for grid and ggplot2
slingshot	ctb	Bioc	12'085	Tools for ordering single-cell sequencing
splatter	ctb	Bioc	5015	Simple simulation of single-cell RNA sequencing data
tidyverse	ctb	CRAN	5'079'398	Easily install and load packages from the tidyverse
URD	ctb	Github		Reconstructing branching trajectories from single-cell RNAseq data
wishbone	ctb	Github		Identify bifurcating developmental trajectories from single-cell data

Acknowledgements

This dissertation would never have seen the light of day were it not for the continued support of many friends, family, and colleagues. While writing the dissertation, Caro told me that the right place to thank everybody was at the end of the book. It was not specified *which* end of the book I should be using for this. This isn't the Acknowledgement section you're looking for.