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

# Acknowledgements

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

# Contents

# Nomenclature

| | |
|---|---|
| CART | Classification And Regression Trees |
| DNA | Deoxyribonucleic Acid |
| GRN | Gene Regulatory Network |
| HCA | Human Cell Atlas |
| IM | Importance Measure |
| ML | Machine Learning |
| mRNA | Messenger RNA |
| NI | Network Inference |
| RF | Random Forests |
| RNA | Ribonucleic Acid |
| TF | Transcription Factor |

CHAPTER 1

# Introduction

**Abstract:**

Partially adapted from:
**Cannoodt, R.***, Saelens, W.*, and Saeys, Y. Computational methods for trajectory inference from single-cell transcriptomics. *European Journal of Immunology* 46, 11 (2016), 2496–2506. doi:10.1002/eji.201646347.
Saelens, W.*, **Cannoodt, R.***, Todorov, H., and Saeys, Y. A comparison of single-cell trajectory inference methods. *Nature Biotechnology* 37, 5 (2019), 547–554. doi:10.1038/s41587-019-0071-9.
Todorov, H., **Cannoodt, R.**, Saelens, W., and Saeys, Y. Network Inference from Single-Cell Transcriptomic Data. *Gene Regulatory Networks* (2019), 235–249. doi:10.1007/978-1-4939-8882-2_10.
* Equal contribution

1

## 1.1 The cell

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 allow it to adapt to changes in its environment continuously. It can be very challenging to comprehend the cellular response to a signal due to the dynamic nature of these processes. A reductionist approach to understanding a complex biological system is to study the biochemical components which it is comprised of [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. Observing the biomolecular insides of cells in this manner can ultimately provide fundamental insights into the processes that govern these cells and help uncover novel approaches for diagnosing and treating disease. Every coin has its flip side, however, and in this case, it is that the amount of data generated from these experiments is not analysable by hand.

For example, the Human Cell Atlas (HCA) consortium [2] 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 [3], and this number is likely to increase well above one hundred million.

HT: I feel like you're only going to focus on preprocessing, when reading this section. Maybe add a second paragraph mentioning that separating technical and biological noise from actual biological processes in the data is not the only challenge, and that there is also a necessity of new computational tools to start generating more complex models, integrating the information from single cell high sequencing experiments in a much more refined way?
RC: Literally but succinctly state what the exact contributions of the thesis are, this is still very vague.

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. Namely, the data contains high levels of noise arising from inherent biomolecular stochasticity in the cells and from the experimental profiling techniques uses, as well as batch effects arising from differences between donors and labs [4]. Biologists thus turn to computer scientists[1] to develop new tools to tackle these problems and help biologists extract meaningful biological insights from the data.

This work makes incremental contributions to the field in order to be able to address the aforementioned problems in a more comprehensive context. This chapter first introduces several key concepts in both cell biology and computer science, upon which the remainder of this work relies. Afterwards, the research objectives and main contributions of this work are outlined.

### 1.1.1 The origins 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 origins 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. The magnitude of this discovery is reflected in our language and culture alike; with sayings such as "It's in your DNA.", or usage of its shape in countless illustrations or artworks (Figure 1.1).

RC: Use crispr-cas to assert the impact of the discovery of DNA, not the FSVM spiral

Even so, a widely-accepted hypothesis states that life (or cells) did not originate from DNA, but

[1] or computational biologists turn to themselves

**Figure 1.1: A prominent display of the double helix shape at the VIB FSVM building.**

instead was kicked off 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 perform the chemical reactions required to sustain themselves (Figure 1.2). Only later did cells develop the ability to use DNA and proteins to self-sustain in a process commonly referred to as the Central Dogma.

use this reference? [7]

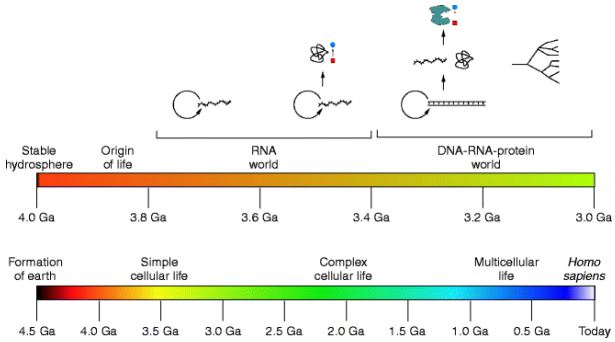



**Figure 1.2: RNA world.** The postulated rise and fall of the RNA world during the evolution of life, from early self-replicating RNA to complex, RNA-controlled metabolism, to the invention of translation, followed by diversification of all modern branches of life. Image from Horning (2011) [8].

### 1.1.2 The Central Dogma

HT: feels like an enumeration, should make it more into a story

The Central Dogma is a set of processes present which govern the general flow of genetic information in almost all existent living cells. In short, it states that DNA codes for RNA, which in turn codes for proteins. In this work, we assume the main processes involved in the Central Dogma are replication, transcription, splicing and translation (Figure 1.3).

**Replication** is the process of duplicating DNA, which allows a cell to divide such that the resulting daughter cells retain complete copies of its genetic information. DNA consists of four different so-called nucleobases named adenine (A), cytosine (C), guanine (G), and thymine (T). Each strand of the double helix structure of DNA is a linear chain of nucleobases. The two strands are held together by hydrogen bonds since adenine can form three hydrogen bonds with thymine and cytosine can form two hydrogen bonds with guanine. Each nucleobase can only bond with one other nucleobase, making the strands complementary to one another. Since one strand can be deduced from its complementary strand, DNA can be represented by a sequence of letters, for example, "ACTCGGTTTAGCA".

A stretch of DNA that contains a genetic blueprint for a particular molecule is called a gene, and the collection of an organism's genes is called its genome. **Transcription** is the process of synthesising

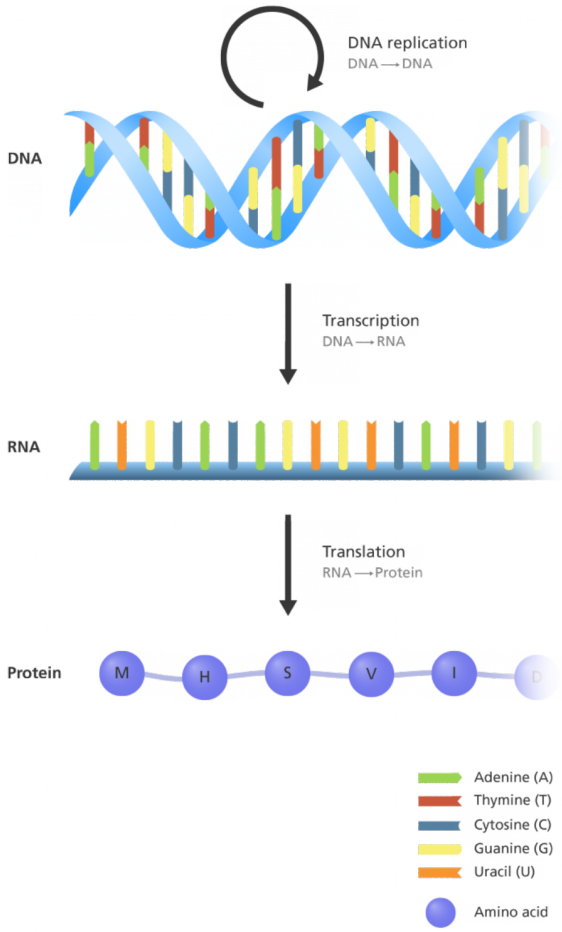



**Figure 1.3: Central Dogma.**

an RNA molecule from a gene, and the resulting molecule is called a transcript. RNA is similar to DNA but differs in several ways; most notably all thymine nucleobases are replaced with uracil (U), and RNA molecules consist of only one strand. Due to its single-strandedness, RNA is less stable and will break down faster. Single-strandedness also allows some types of RNA (e.g. transfer RNA, ribosomal RNA) to form more complex three-dimensional structures by having certain regions bind to other regions of the strand. This work only considers messenger RNA (mRNA), which are transcribed from protein-coding genes, meaning that the mRNAs can result in the production of particular proteins. A protein-coding gene consists of alternating sections called introns and exons. Exonic regions contain code for what a protein should be made up of, whereas the intronic regions can contain information on how to assemble the pieces.

**RNA splicing** is a process that occurs in almost all organisms and results in the removal of intronic regions in the mRNA molecules, resulting in a mature mRNA. Splicing allows to create multiple variants of the same product, which can affect the enzymatic properties or localisation of the resulting product [9].

During **translation**, a chain of amino acids is synthesised from a mature mRNA transcript. Every three nucleobases are translated into one of 21 different amino acids. The resulting chain of amino acids is folded up into a protein, the structure of which is determined by the sequence of different amino acids in the chain. In turn, its structure determines the functionality of the protein, which includes catalysing biochemical reactions, providing structure, and transportation of molecules.

### 1.1.3 Cell types

The functionality provided by a cell is defined (mostly) by the proteins of which it consists. One common approach to trying to understand the functionality of a cell is to observe which molecules are present in the cell and to associate those molecules with functionality.

*Homo sapiens* like to categorise everything they encounter and so too have they conceptualised groups of cells called "cell types" according to their functionality. The concept of cell types eases reasoning about all aspects of biology, for instance, which cell types turn into (differentiate) or communicate with which other cell types, or how a cell type responds to a specific stimulation. Cells can 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.

do not only discuss differentiation, but instead any kind of developmental process; and mention 'developmental trajectory'.

Cell differentiation is not an instant process; it is a continuous process in which a cell gradually produces the biochemical machinery required in order to fulfil a particular task. In this regard, it makes sense not only to reason about cell types but also about the transition states between cell types and the dynamic processes involved therein.

are dynamic processes defined? a dynamic process is just a biochemical reaction

### 1.1.4 Cell dynamics and gene regulation

If cells are dynamic entities and can gradually produce the molecules needed to acquire new functionality, what is the process by which this happens? The mechanism by which this happens is called gene regulation. Some proteins (or other molecules such as micro RNAs) are capable of determining the rate at which a gene is transcribed (transcription rate). Such proteins are called transcription factors (TFs), and the genes they regulate are called their targets. Typically, one TF will regulate the transcription rate of many targets.

TODO: download database and calculate some statistics?

Production of a specific molecule might require multiple cascades of gene regulation. The collection of all gene regulatory interactions between transcription factors and targets is called a gene regulatory network (GRN). Studying the active parts of a cell's gene regulatory network can thus reveal which dynamic processes are taking place.

TODO: explain regulation mechanisms such as transcription factor binding sites?

### 1.1.5 Profiling single cells

This sections goes greatly into detail about IHC and cytometry. Can the merging of the 'single-cell' and 'omics' be explained without going so much into detail?

In order to understand a biological process, it is often quite helpful to be able to profile (i.e. observe) the biomolecular components involved therein. The single-cell "omics" technologies which we have at our fingertips today originated from the convergence of two different fields, *"single-cell"* and *"omics"*.

The earliest approaches for measuring the abundance of particular molecules in *single cells* used the preferred instrument of every stereotypical biologist: the microscope. Since it was developed by Coons et al. in 1941, immunohistochemistry (IHC) has been instrumental in visualising antigen-antibody proteins [10]. In many multicellular organisms, antibodies and antigens serve as crucial

communication tools as part of the organism's immune system. A cell can present a particular type of antigen on its cell surface, which allows a particular type of antibody to bind to it.

Rephrase the microscope sentence

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 light excitation. Using different colours (wavelengths) allows measuring expression levels of different antibodies simultaneously. Characterising cells in a 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). While modern implementations of IHC improve the throughput drastically by using robots to automate the image acquisition and computer software to automate cell segmentation, the procedure is still labour intensive as the robots and computer software still needs to be kept in check.

Flow cytometry [11] is a technique which circumvents imaging and segmentation issues by having a steady stream of cells run through a laser and measuring the amount of light scattered from those cells. Flow cytometry technology enables to measure protein expression levels for millions of cells and tens of different antibodies.

Since 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 qPCR, FISH). All of these single-cell (non-omics) technologies are limited by the number of different molecules they could measure, however; and thus required handpicking the molecules of interest before performing an experiment, making the experiment biased towards the preconceptions of the experimenter.

On the other side of the spectrum are the so-called "omics" technologies. "Omics"[2] is a collective term for profiling all molecules of a particular type in a high-throughput manner. There are many types of "omics", but the most commonly used are the following. In genomics, all of an organism's genes are studied – its whole genome. Transcriptomics and proteomics study the organisms RNA transcripts and proteins, respectively. A notable downside of traditional omics technologies is that in order to capture enough material an ensemble of cells needs to be profiled, and thus only the average expression levels are returned; thereby granting the technology the name "bulk" omics. If a subset of these cells contains unique patterns in expression levels, this pattern will be masked in the bulk population and is thus undetectable. Specific examples of omics technologies are next-generation sequencing, which can be used to determine the DNA sequence of an organism, and RNA sequencing, which profiles the sequences of RNA transcripts. By mapping the sequences of RNA transcripts to genes in the organisms DNA, a gene expression profile can be obtained.

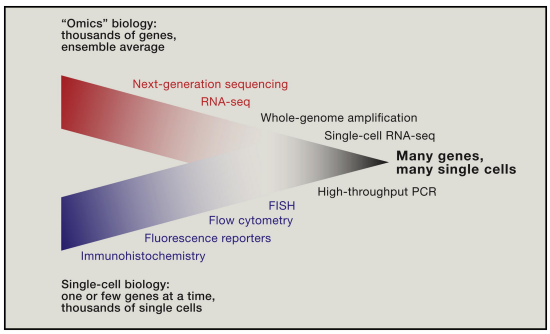
Demonstrate the masking effect of bulk analyses.

Transformative technological advances in microvolume sequencing allowed Tang et al. to analyse the transcriptome at single-cell resolution [13], thereby bringing single-cell biology and omics together to create single-cell omics (Figure 1.4A). During the decade that followed, the number of single-cell omics technologies has skyrocketed, allowing to profile tens of thousands of cells (Figure 1.4B) and measuring other levels of information such as proteomic expression levels (Figure 1.4C).

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 Human Cell Atlas. The HCA consortium has set out to redefine all human cell types in terms of their gene expression and location, and the developmental trajectories connecting the different cell types. As part of this endeavour, the consortium will likely profile the whole transcriptomes tens or even hundreds of millions of cells.

[2] The etymology of "omics" is quite interesting [12].

A.




C.

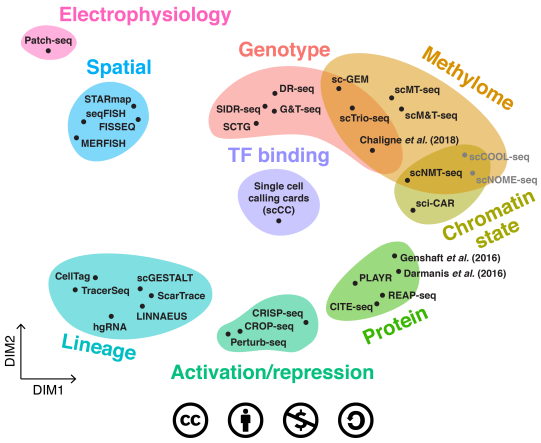



**Figure 1.4:** A. **Convergence of "Omics" Biology and Single-Cell Biology.** Technology that allows researchers to obtain genome-wide information from single cells is extending the boundaries of a field that has thus far been limited to the analyses of a select gene in eukaryotes. Image from Junker and van Oudenaarden (2014) [14]. B. C. scmultiomics [15].

## 1.2 Computational tools

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 such as the Human Cell Atlas. The HCA consortium has set out to redefine all human cell types in terms of their gene expression and location, and the developmental trajectories which connect the different cell types. As part of this endeavour, the consortium will perform single-cell omics on tens or even hundreds of millions of cells.

Single-cell omics permits new types of analyse but also come with hitherto unseen data characteristics, the combination of which poses exciting new challenges for the computational community to tackle (Figure 1.5A)[16, 17, 18]. These challenges include:

- normalisation: separating biological noise from technical noise,
- dimensionality reduction: providing a visual and informative overview of a given dataset,
- trajectory inference: identifying and characterising transitions between different cellular states, and
- gene regulatory network inference: inferring regulatory interactions between transcription factors across individual cells.

Make a better connection to the subsections below

### 1.2.1 Normalisation

TODO

### 1.2.2 Dimensionality reduction

Single-cell omics datasets typically have too many dimensions (features) in order to be easily interpretable by humans and even by most computational tools. Dimensionality reduction (DR) methods transform high-dimensional data into a meaningful representation with fewer dimensions. It is important to note that its usage depends on the target audience: for humans – to visualise data in a 2-D

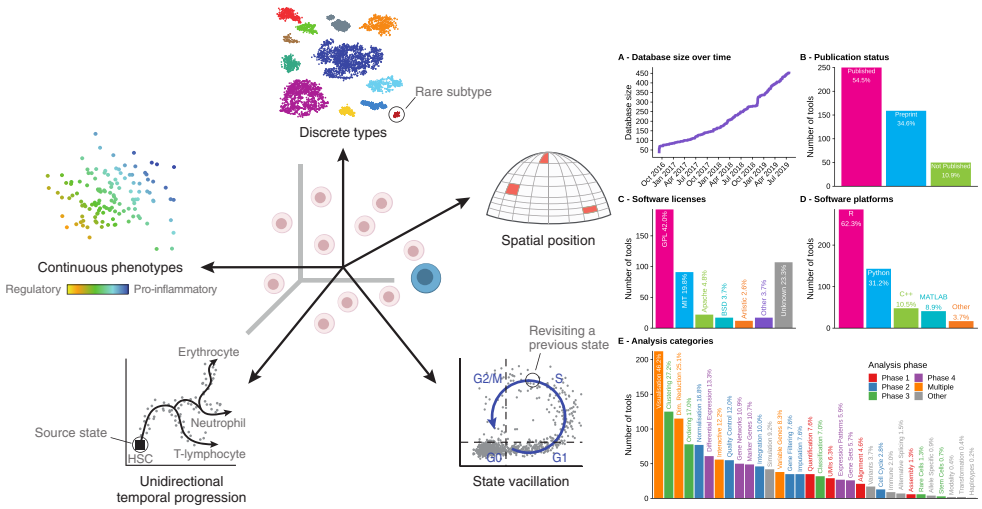



**Figure 1.5: A. Single-cell omics allows for many new types of computational approaches.** Figure adapted from Wagner et al. (2016) [19]. B. Zappia et al. (2018) [20]

plane to aid with interpretation by humans, or for computers – to construct a denser representation of the data such that it mostly contains the same information but with fewer dimensions.

There are many ways of classifying DR methods [21], but this work will use the following main categories: feature projection-based and manifold learning. Projection-based DR methods aim to perform a linear transformation of the data while preserving the pairwise distances between samples as much as possible. Examples of commonly used projection-based DR methods in single-cell omics are PCA and MDS. Manifold learning methods are methods which 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-SNE, Diffusion Maps and UMAP.

This section is not very interestingly written

### 1.2.3 Trajectory inference

Single-cell omics data provide new opportunities for studying cellular dynamic processes, such as the cell cycle, cell differentiation and cell activation [22, 23]. Trajectory inference (TI) is a new category of computational tools used to offer an unbiased and transcriptome-wide understanding of a dynamic process [22, 24].

The dataset can be a single snapshot of a mixture of cells in different stages, or a set of samples collected at different time points (Figure 1.6A). Typically, TI methods first analyse similarities between cells, optionally infer the topology of the underlying process, and finally order cells along that trajectory (Figure 1.6B). The second step can be optional, as some methods assume a specific topology beforehand. TI methods allow the identification of new subsets of cells, delineation of a differentiation tree, and characterisation of the main driver genes along a state transition (Figure 1.6C). Current applications of TI focus on specific subsets of cells, but ongoing efforts to construct transcriptomic catalogs of whole organisms [25, 26, 27] underline the urgency for accurate, scalable [28, 29] and user-friendly TI methods.

Could still expand this section with pieces from the EJI paper, though it needs to be adapted strongly.

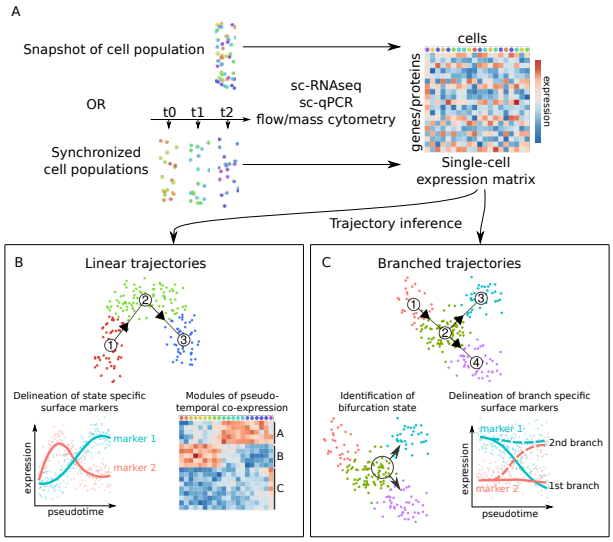



**Figure 1.6:** Applications of single-cell trajectory inference methods. (A) Single-cell omics data appropriate for TI can be both obtained from an unsynchronised population of single cells (snapshot data) but also from synchronised cell populations. (B) UPDATE! (C) UPDATE!

### 1.2.4 Gene regulatory network inference

Gene regulatory network inference, or network inference (NI) for short, is a type of computational analysis where thousands of transcriptomic profiles are analysed together in order to infer the regulatory interactions between transcription factors and genes. This topic already received much attention with the advent of bulk omics (before single-cell omics). These efforts culminated in several DREAM competitions assessing the performance of 29 different NI methods [30, 31].

After the last DREAM competition, it seemed that interest in NI methodology had declined. After all, NI on bulk omics profiles suffered from several crucial issues. As mentioned previously, bulk profiles are generated by pooling together the RNA transcripts of a supposedly homogeneous population of thousands of cells. Since the expression values are averaged over the whole population, incorrect assumptions on the homogeneity of the pooled cells may lead to the masking of relevant expression patterns in rare cell populations (Figure 1.7). Besides, NI methods rely on a diverse set of time-series and perturbation experiments in order to reliably identify causal regulatory interactions. Such experiments are expensive and time-consuming, and an inaccurate selection of time points might result in crucial intermediate stages being missed.

The advent of single-cell omics has made scientists wonder whether now is the time to revisit network inference [16]. One of the main advantages of single-cell omics is the ability to quantify the exact cellular state of thousands of cells per experiment. The heterogeneity between cells caused by naturally occurring biological randomness [32] can be exploited to infer regulatory interactions between TFs and their target genes at much lower costs (see Figure 1.7). In this setting, heterogeneity in the cell population eases network inference, rather than mask condition-specific expression patterns in regulatory interactions.

## 1.3 Research context and objectives

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 cellu-

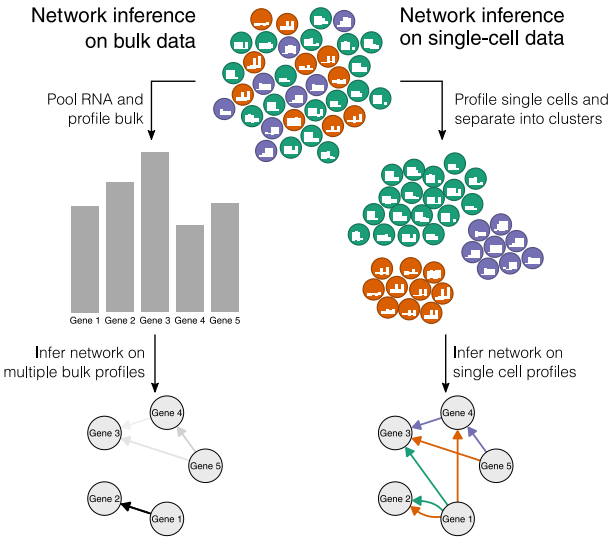



**Figure 1.7:** Bulk expression data return the average expressions of genes among large numbers of cells. In order to infer regulatory networks from this type of data, multiple bulk profiles (resulting from time series or perturbation experiments) are required. On the other hand, sequencing the transcriptome at the single-cell level uncovers the high variability among cells, providing the necessary information to infer gene regulatory networks directly.

lar development or intercellular communication. Hundreds of new software tools were developed [20] 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 majority of the newly developed computational tools were not quantitatively and comparatively evaluated. Rather, they relied on anecdotal evidence to demonstrate its usefulness. This issue is not the result of the tool developer's malevolence, but instead of the lack of data required to perform such comprehensive benchmarks.

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.

In this work, we aim to speed up scientific progress in single-cell omics by providing end-users with guidelines on how to use which state-of-the-art tools, and by providing developers with the necessary tools to assert a mimimum performance criterion when introducing new methods.

We develop benchmarking strategies for assessing the performance of novel computational tools constrained by low availability of real data (Chapter 2). *In silico* simulations of individual cells are used to help kick-start emerging domains much more safely and allow anticipation of future technological developments by already developing computational tools.

We apply this strategy to perform a comparison of TI methods (Chapter 3). Trajectory inference is one of the largest categories of all the novel single-cell omics tools, yet a comprehensive and quantitative study of the advantages and disadvantages of the numerous tools was hitherto lacking.

We introduce a novel TI method specialised in inferring linear trajectories (Chapter 4). Despite linear TI being the most simple but commonly used form of trajectory inference, the benchmark demonstrated that most TI methods are not capable of producing accurate models of linear datasets.

We conclude by inventing a new type of approach aimed at inferring the GRN of individual cells

(Chapter 5). Due a lack of real data which can be used as a gold standard dataset, we use *in silico* single-cell data to quantitatively evaluate the performance.



## 1.4 List of contributions

### 1.4.1 First-author publications

- **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.
- **Cannoodt R** *, Saelens W *, Saeys Y. dyngen: Simulating developing single cells. In preparation.
- **Cannoodt R**, Saelens W, Sichien D, Tavernier S, Janssens S, Guilliams M, Lambrecht B, De Preter K, Saeys Y. SCORPIUS: Fast, accurate, and robust single-cell pseudotime. In preparation.
- **Cannoodt R** *, Saelens W *, Saeys Y. dyno: A toolkit for inferring, visualising, and interpreting trajectories. In preparation.
- **Cannoodt R**, Saelens W, Saeys Y. bred: Inferring single cell regulatory networks. In preparation.

*: Equal contribution.

### 1.4.2 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, ..., **Cannoodt R**, Saelens W ..., 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 Dec;20(1):125.
- Lorenzi L, ..., **Cannoodt R**, ..., Mestdagh P. The RNA-Atlas, a single nucleotide resolution map of the human transcriptome. In preparation.
- 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 Bram, ..., **Cannoodt R**, ..., Saeys Y, Aerts S. A scalable SCENIC workflow for single-cell gene regulatory network analysis. Submitted to Nature Protocols.

### 1.4.3 Open-source software

As part of this work, many open-source software packages were created and many others were contributed to (Table 1.1).

Packages that were created as part of this work are hosted on Github under the username rcannood[3] or the dynverse organisation[4]. As part of our standard development practices, we automate execution of unit tests and writing extensive documentation to ensure the code complies with CRAN policy before submission. We aim to submit all other packages to CRAN as well.

We also helped maintain or extend other packages on Github, CRAN or Bioconductor on which our software depends. This includes help speed up parts of the dependency (slingshot), adding new functionality (devtools, ParamHelpers), fixing bugs (proxyC, rlang, monocle, splatter, slingshot), becoming a maintainer of orphaned packages (diffusionMap, princurve, GillespieSSA), and extending the documentation (devtools, mlr, remotes). Several of these package receive millions of downloads per year (devtools, remotes, rlang).

[3] https://github.com/rcannood?tab=repositories
[4] https://github.com/dynverse?tab=repositories

**Table 1.1: 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 2019-08-01 and 2019-09-10. CRAN download statistics are retrieved from the Rstudio CRAN mirror only; other CRAN mirrors do not track download statistics. For Github repositories, no download statistics could be retrieved.

| Name | Role | Host | Downloads per year | Description |
|---|---|---|---|---|
| babelwhale | aut | CRAN | 3996 | Interacting with Docker and Singularity containers |
| diffusionMap | aut | CRAN | 21'361 | Implements diffusion map method of data parameterization, including creation and visualization of diffusion map |
| dynbenchmark | aut | Github | | Pipeline for benchmarking trajectory inference methods |
| dyndimred | aut | CRAN | 5511 | 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 | 3084 | 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 | 5657 | Common functionality for the dynverse packages |
| dynwrap | aut | Github | | A common format for trajectories |
| GillespieSSA | aut | CRAN | 7546 | Gillespie's Stochastic Simulation Algorithm (SSA) |
| GillespieSSA2 | aut | CRAN | 6506 | Gillespie's Stochastic Simulation Algorithm for Impatient People |
| gng | aut | Github | | An Rcpp implementation of the Growing Neural Gas algorithm |
| incgraph | aut | CRAN | 3175 | Incremental graphlet counting for network optimisation |
| lmds | aut | Github | | Landmark Multi-Dimensional Scaling |
| princurve | aut | CRAN | 26'991 | Fits a principal curve in arbitrary dimension |
| proxyC | aut | CRAN | 117'484 | Computes proximity in large sparse matrices |
| qsub | aut | CRAN | 3193 | Running commands remotely on gridengine clusters |
| SCORPIUS | aut | CRAN | 4772 | Inferring developmental chronologies from single-cell RNA sequencing data |
| ClusterSignificance | | Bioc | 803 | Assess if class clusters in dimensionality reduced data representations have a separation different from permuted data |
| devtools | ctb | CRAN | 3'775'350 | Tools to make developing R packages easier |
| merlot | ctb | Github | | A method for reconstructing lineage-tree topologies from scRNA-seq data |
| mlr | ctb | CRAN | 142'605 | Machine Learning in R |
| monocle | ctb | Bioc | 35'240 | Clustering, differential expression, and trajectory analysis for single-cell RNA-Seq |
| ParamHelpers | ctb | CRAN | 109'408 | Helpers for Parameters in Black-Box Optimization, Tuning and Machine Learning |
| pseudogp | ctb | Github | | Probabilistic pseudotime for single-cell RNA-seq |
| Rdimtools | ctb | CRAN | 7367 | Dimension Reduction and Estimation Methods |
| remotes | ctb | CRAN | 3'704'594 | R package installation from remote repositories, including GitHub |
| rlang | ctb | CRAN | 11'470'763 | Functions for base types and core R and tidyverse features |
| SCope | ctb | Github | | Visualization of large-scale and high dimensional single cell data |
| slingshot | ctb | Bioc | 11'643 | Tools for ordering single-cell sequencing |
| splatter | ctb | Bioc | 3741 | Simple simulation of single-cell RNA sequencing data |
| URD | ctb | Github | | URD reconstructs transcriptional trajectories underlying specification or differentiation processes in the form of a branching tree from single-cell RNAseq data |
| wishbone | ctb | Github | | Identify bifurcating developmental trajectories from single-cell data |

# CHAPTER 2

## dyngen: simulating single cells

**Abstract:**

2

## 2.1 Introduction

Continuous technological advancements to high-throughput profiling of single cells are having profound effects on how researchers can validate biological hypotheses. For example, single-cell RNA sequencing (scRNA-seq) directly resulted in the development of a new type of computational method called trajectory inference (TI). By profiling the transcriptomics profiles of developing cells, TI methods attempt to reconstruct and characterise the underlying dynamic processes [24]. While early experimental technologies allowed to profile one single modality (e.g. DNA sequence, RNA or protein expression), recent developments permit profiling multiple modalities simultaneously.

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 [33]. While this falls outside the reach of current experimental technologies, *in silico* simulations of single cells would allow developing the next wave of computational techniques in anticipation of new experimental technologies.

A few generators of scRNA-seq profiles have already been developed (e.g. splatter [34], powsimR [35], PROSSTT [36] and SymSim [37]). These can be used to evaluate the performance of computational tools, and to explore their strengths and weaknesses. A limitation of directly simulating a scRNA-seq profile (instead of a single cell) is that extending the simulation to other aspects of the cell – such as tracking the full history of molecular states – becomes difficult.

We introduce dyngen, a multi-modality simulator of single cells and their dynamics (Figure 2.1). dyngen was initially developed as part of a comprehensive benchmark of TI methods [38] but has since been extended to be applicable in a much broader context. We demonstrate its flexibility by simulating three different types of biological experiments, and using these simulations to develop new benchmarking techniques for computational tools.

Our simulator draws inspiration from a simulator for bulk transcriptomics data, GeneNetWeaver [39, 31], but with two key improvements to make it work for single-cell data. First, instead of simulating continuous systems using stochastic differential equations (SDE), we simulate individual molecules and their reactions using derivatives of Gillespie's Stochastic Simulation Algorithm (SSA) [40]. Such a simulation better captures the stochasticity when low number of molecules are present [40], as is the case in single-cells [41]. Processes such as transcriptional bursting may be difficult to represent in a continuous model, but develop naturally in single-molecule simulations [42]. A second improvement is that we include several ways to construct a GRN such that it mimics a dynamic process of interest, such as cell differentiation into multiple cell types.

We demonstrate dyngen's flexibility by simulating numerous different types of biological experiments, and using these simulations to develop new benchmarking techniques for computational tools.

## 2.2 Results

dyngen simulates the transcriptomic changes of a cell over time using a model of gene regulation. Throughout this section, a simple simulation of a cell undergoing a cyclic process is used to illustrate key strengths of dyngen (Figure 2.2). This example only comprises of a single cell containing 5 genes, but dyngen can easily scale up to thousands of simulations containing thousands of genes.

GRN is defined nowhere

In dyngen, a cell consists of a set of molecules, the abundance of which are affected by a set

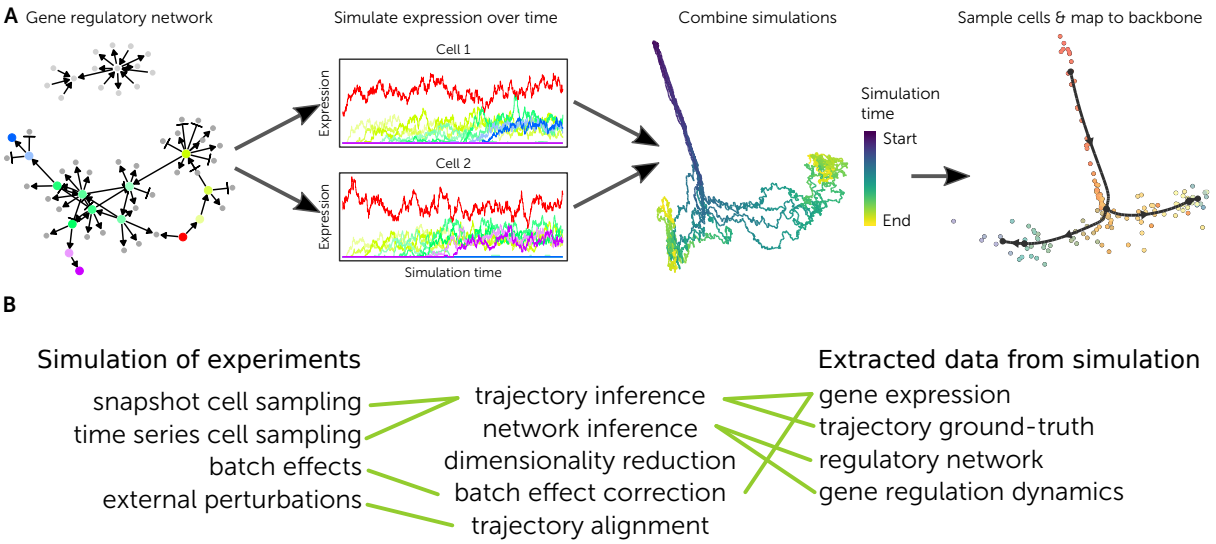



**Figure 2.1:** Showcase of dyngen functionality. Remove B? *Yes please*

of reactions: transcription, splicing, translation, and degradation (Figure 2.2A). These reactions are determined from a predefined set of gene regulatory interactions (Figure 2.2B), henceforth referred to as a gene regulatory network (GRN). The likelihood of a reaction occurring at any given point in time is defined by the GRN and by the abundance of molecules involved each reaction.

One of dyngen's main advantages is that through careful engineering of the GRN, different cellular developmental processes can be simulated. Different GRNs can result in branching, converging, cyclic, or even disconnected developmental topologies. Multiple simulations with slightly different GRNs can emulate rewiring events in disease or perturbation experiments. Multiple simulations with some slight perturbations can be used to replicate batch effects.

could use figure; one with GRNs of different topologies, another with rewiring events

Another advantage is that 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.

explain what is meant by 'sampling' and 'profiling technique' a little better

The final main advantage is that by making alterations to the simulation pipeline, multiple types of experiments (sampling technique or profiling technique) can be simulated. By default, dyngen supports snapshot experiments (uniformly sampling from an asynchronous dynamic process) and time-series experiments (sampling cells from different intervals in the simulation). It is possible to implement other experimental protocols, such as sampling the same cell at regular intervals.

show that dyngen output resembles real data using e.g. countsimQC?

add result figures pertaining use cases

## 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

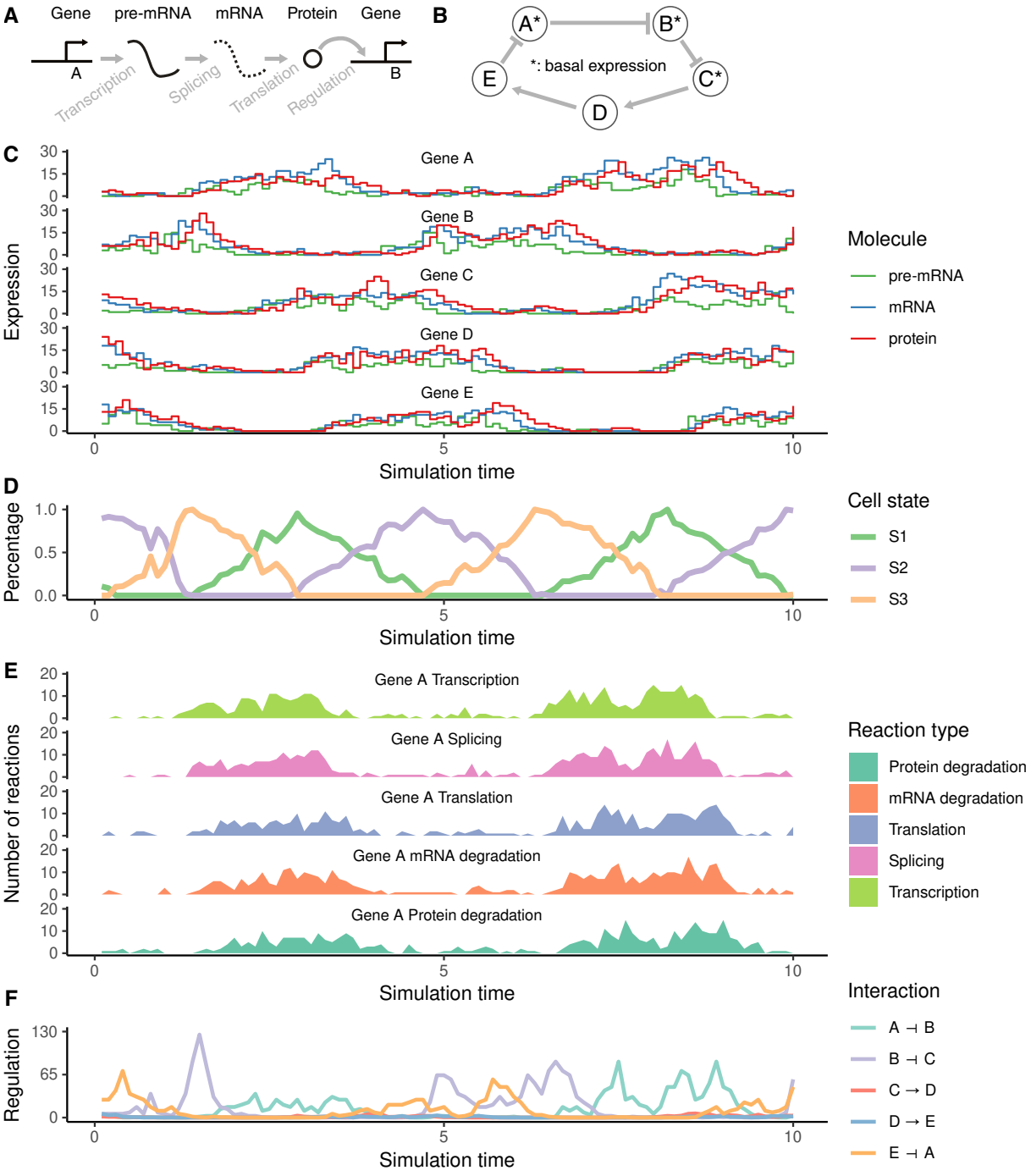



**Figure 2.2:** Showcase of dyngen functionality. A time resolution of 0.1 was used, but this can be increased or decreased without effect on performance of the execution of the simulation. **TODO: perhaps it's better to replace Figure 2.2 with one subfigure for each of the paragraphs in this text.**

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 [39, 43].

Adding batch effects to snapshot simulations of linear (or even branching) trajectories allows evaluating trajectory alignment methods – which attempt to map two or more trajectories onto each other. Adding perturbations to the GRN allows evaluating the performance of differential network inference methods – which predict differential regulatory interactions between two or more groups of profiles. Sampling a cell at a particular time point and once more at a later time point allows evaluat-

ing the performance of RNA velocity approaches – which predict the future state of a cell by looking at differences in pre-mRNA and mRNA abundance levels.

dyngen ultimately also 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. Similarly, it could also be used to compare which experimental technique will likely produce the most accurate result. For example, is it possible to infer directionality of regulatory interactions from snapshot experiments only, or are time series or knockdown experiments a necessity in order to infer high-quality regulatory networks?

Currently, dyngen focuses on simulating cells as standalone entities. Future developments include extending the framework to simulate multiple cells in a virtual environment. Allowing cells to receive and react to environmental and intercellular stimuli would enable simulating essential cellular processes such as cell division and migration.

also adding more types of molecules, e.g. protein complex, small rnas, PPI

## 2.4 Methods

The method section is REALLY rough at this stage.

The workflow to generate *in silico* single cell data consists of six main steps (Figure 2.3).

### 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 [44] strongly repress each other, forcing one of the two to become inactive [45]. Such mutual antagonism can be modelled and simulated [46, 47]. Although the two-gene model is simple and elegant, the reality is frequently more complex, with multiple genes (grouped into modules) repressing each other [48].

In dyngen, the user defines the behaviour of the simulation by defining how sets of genes, called modules, are regulating eachother. A module may have basal expression, which means that pre-mRNA of the 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.

Several examples of module networks are given (Figure 2.4). 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 eachother 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).

mention strength and cooperativity.

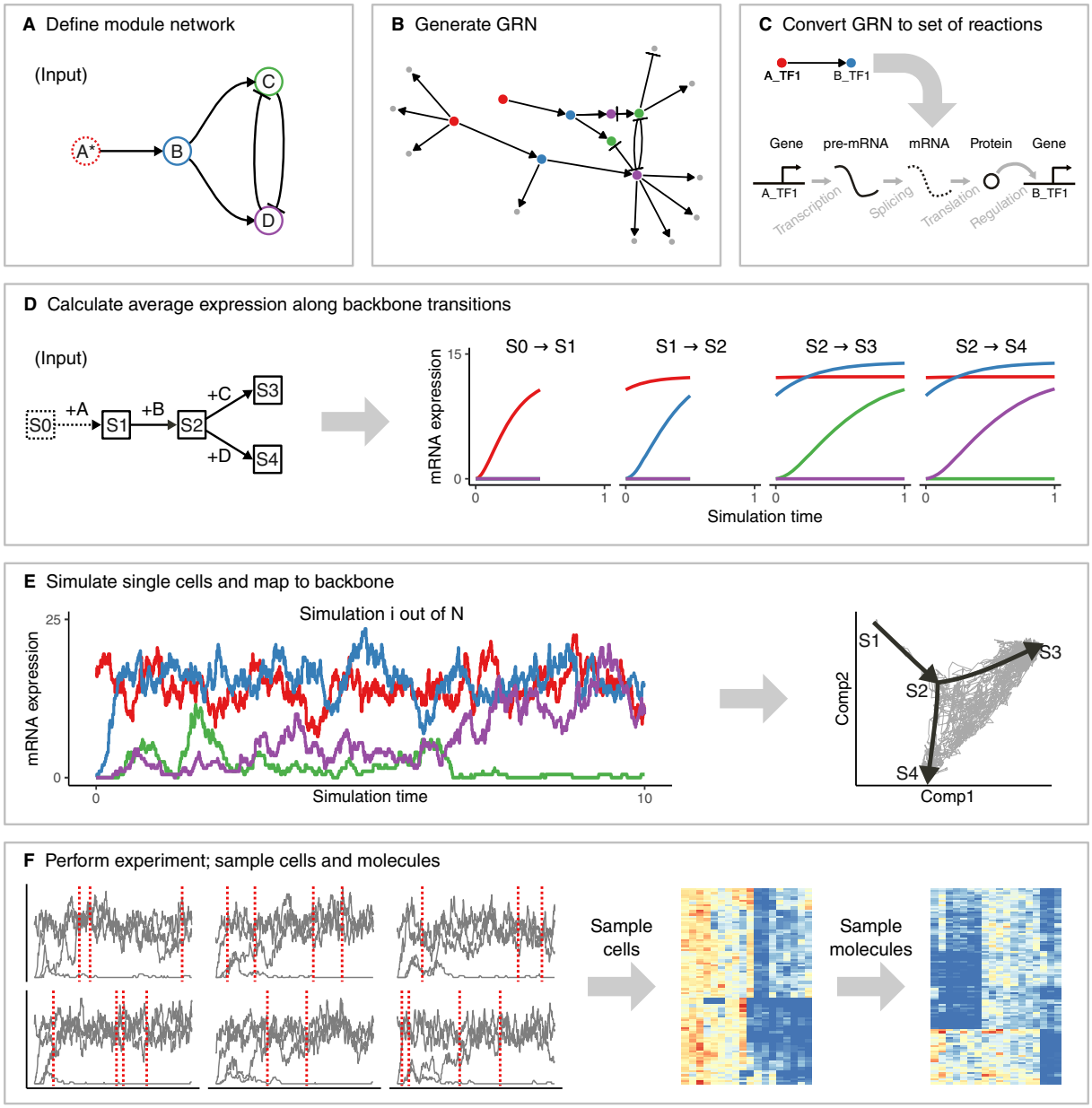



**Figure 2.3: The workflow of dyngen is comprised of six main steps.** **A:** The user needs to specify the desired module network or use a predefined module network. **B:** Each gene in a module is is regulated by one or more transcription factors from the upstream module. Additional target genes are 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 mocules at each time step are extracted. Each time step is mapped to a point in the backbone. **F:** Multiple cells are sampled from each simulation. Molecules are sampled from each cell.

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 2.3D). 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

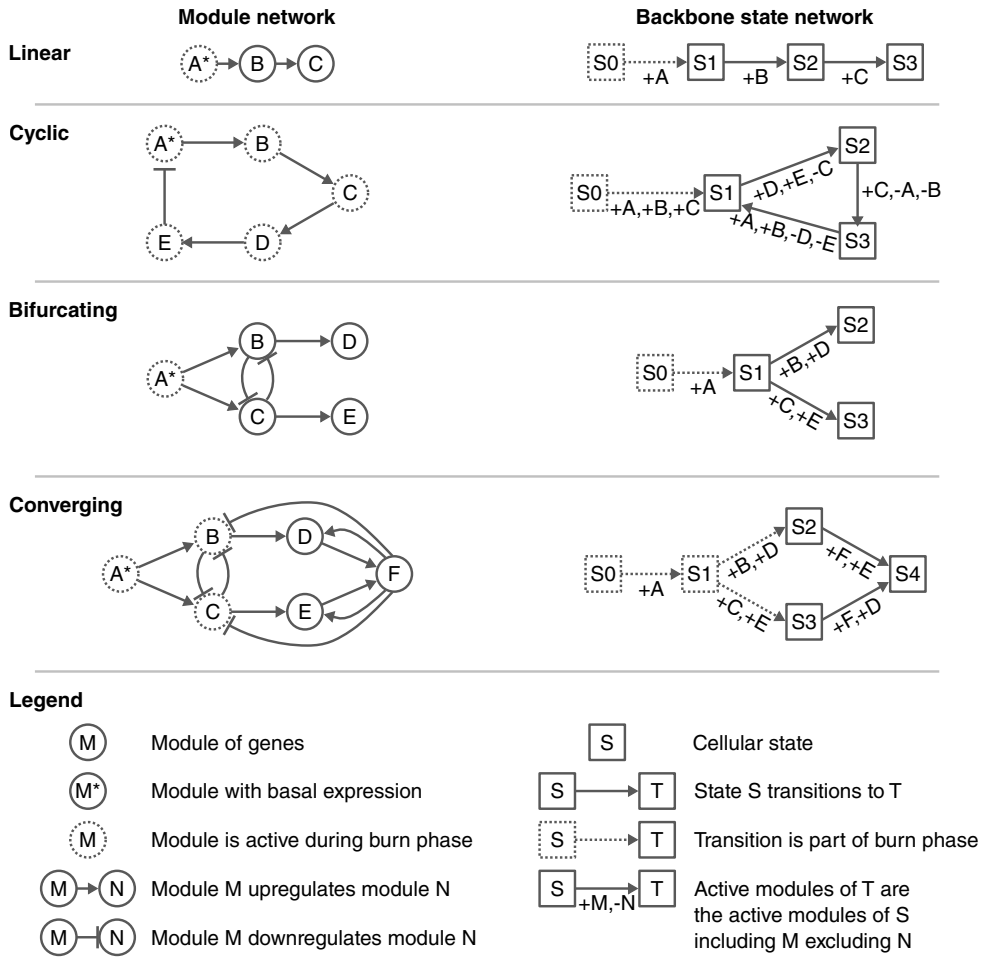



**Figure 2.4:** Example module networks

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 2.5). 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 2.5A). 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 milestone networks and corresponding state networks (Figure 2.5B). 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 2.5C) 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

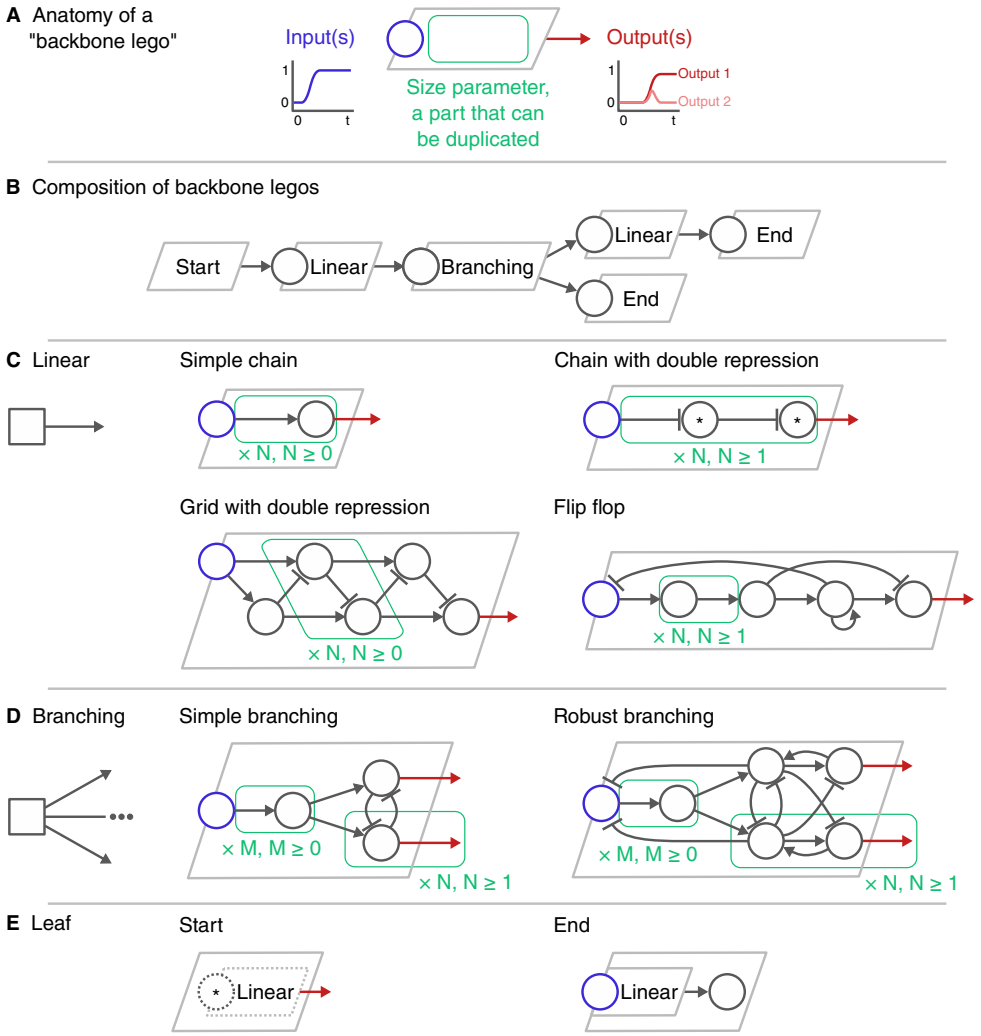



**Figure 2.5:** Backbone lego

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 2.5D) 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 being upregulated other than the one that has been chosen.

A **leaf BBL** (Figure 2.5E) 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 Generate gene regulatory network

Welcome to the beginning of the end

#### Generate the transcription factor network

Parameters:

- Number of TFs to generate $num\_tfs$
- Minimum TFs per module $min\_tfs$, default = 1
- Number of regulatory interactions per module interaction $num\_tf\_ints$, default = 2

Procedure:

- Divide $num\_tfs$ TFs amongst modules in backbone such that each module has at least $min\_tfs$ TFs.
- For every TF in a particular module $M$, connect that TF with $num\_tf\_ints$ TFs for each upstream module of $M$.
- The strength and cooperativity of interactions created this way are defined by the module network.

#### Generate targets

Parameters:

- Number of target genes $num\_targets$
- Name of a FANTOM5 GRN
- Damping factor $damping$, default = 0.05
- Target resampling $resamp$, default = $+\infty$
- Maximum in-degree $max\_in\_degree$, default = 5

Procedure:

- Download the FANTOM5 GRN if not already available
- Randomly map TFs to the regulators in the GRN
- Compute page rank from the selected regulators in the GRN with damping factor $damping$
- Perform weighted sample of $\min(num\_targets, resamp)$ targets weighted by the page rank
- Select subgraph induced by the TFs and the sampled targets in the GRN, remove edges to the TFs
- Add subgraph to current TF-target network
- If less than $num\_targets$ have been sampled in this way, go back to step 2.
- Remove regulatory interactions if a target has more than $max\_in\_degree$ edges.

**Generate housekeeping genes**

Parameters:

- Number of housekeeping genes $num\_targets$
- Name of a FANTOM5 GRN
- Target resampling $resamp$, default = $+\infty$
- Maximum in-degree $max\_in\_degree$, default = 5

Procedure:

- Use same FANTOM5 GRN
- Subsample GRN such that each gene has a maximum in-degree of $max\_in\_degree$
- Perform breadth-first-search from a random gene in the GRN, select $\min(num\_targets, resamp)$ first genes encountered
- Select subgraph induced by the sampled housekeeping genes in the GRN
- Add subgraph to current TF-target network
- If less than $num\_targets$ have been sampled in this way, go back to step 2.

### 2.4.3 Convert gene regulatory network to a set of reactions

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.

We define the abundance levels of pre-mRNA, mRNA and protein of gene $G$ as $w_G$, $x_G$ and $y_G$ respectively. Five reactions affect the abundance levels of these molecules: transcription, splicing, mRNA degradation, translation, and protein degradation. The effects and propensity functions of these reactions are defined in Table 2.1.

**Table 2.1: Reactions affecting the abundance levels of pre-mRNA $w_G$, 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 | $\emptyset \rightarrow w_G$ | $\text{wpr}_G \times \frac{\text{ba}_G - \text{ind}_G^{\lvert R_G^+ \rvert} + \prod_{H \in R_G^+} (\text{ind}_G + \text{reg}_{G,H})}{\prod_{H \in R_G^+} (1 + \text{reg}_{G,H})}$ |
| Splicing | $w_G \rightarrow x_G$ | $\text{wsr}_G \times w_G$ |
| mRNA degradation | $x_G \rightarrow \emptyset$ | $\text{xdr}_G \times x_G$ |
| Translation | $x_G \rightarrow w_G + y_G$ | $\text{ypr}_G \times x_G$ |
| Protein degradation | $y_G \rightarrow \emptyset$ | $\text{ydr}_G \times y_G$ |

TODO: add step-by-step derivation of transcription formula

**Table 2.2: Parameters defined for the calculation of reaction propensity functions.**

| Parameter | Symbol | Definition |
|---|---|---|
| Transcription rate | $\text{wpr}_G$ | $\in N(100, 20),\ \geq 10$ |
| Splicing rate | $\text{wsr}_G$ | $\in N(10, 2),\ \geq 2$ |
| mRNA degradation rate | $\text{xdr}_G$ | $\in N(5, 1),\ \geq 2$ |
| Translation rate | $\text{ypr}_G$ | $\in N(5, 1),\ \geq 2$ |
| Protein degradation rate | $\text{ydr}_G$ | $\in N(3, 0.5),\ \geq 1$ |
| Interaction strength | $\text{str}_{G,H}$ | $\in 10^{U(0,2)}$ * |
| Interaction cooperativity | $\text{co}_{G,H}$ | $\in U(0.5, 2)$ * |
| Independence factor | $\text{ind}_G$ | $\in [0, 1]$ * |
| TF concentration at half-maximal binding | $\text{hmy}_H$ | $= 0.5 \times \frac{\text{wpr}_H \times \text{ypr}_H}{\text{xdr}_H \times \text{ydr}_H}$ |
| Regulation activity | $\text{reg}_{G,H}$ | $= \left(\text{str}_{G,H} \times \frac{\text{y}_H}{\text{hmy}_H}\right)^{\text{co}_{G,H}}$ |
| Basal expression | $\text{ba}_G$ | $= \begin{cases} 1 & \text{if } \text{R}_G^+ = \emptyset \\ 0.0001 & \text{if } \text{R}_G^- = \emptyset \text{ and } \text{R}_G^+ \neq \emptyset \text{ *} \\ 0.5 & \text{otherwise} \end{cases}$ |

*: unless already defined when $G$ is a TF.

### 2.4.4 Compute average expression along backbone transitions

this is basically an ODE, though we never reference ODEs anywhere. Maybe this is ok, this simplifies the section a bit?

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 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) 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 $j$th reaction for the current state of the simulation.

SSA simulations are notoriously slow. We use GillespieSSA2 which contains many optimisations such as translating and compiling all the propensity functions to C++ and implementations of approximations of SSA which allows to trigger many reactions simultaneously at each iteration.

The framework 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

The cellular state of each timepoint in the SSA simulation is mapped to the state network of the backbone by calculating the 1NN between a state vector in the simulation and the average expression levels along transitions.

### 2.4.6 Simulate experiment

From the SSA simulation we obtain the abundance levels of all the molecules at the different time points. 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

Cells can be sampled from an unsynchronised population of single cells (snapshot) or at multiple time points in a synchronised population (time series).

**Snapshot** Cells are just sampled randomly from the different time points in the simulation.

**Time series** The timeline of the simulations is cut up into chunks. From several of these chunks, cells are sampled. For each cell it is known at which time point it was sampled.

#### Sample molecules

- From real dataset, look at the number of transcripts that was captured per cell. Library size $ls_i$ of cell $i$ is samples from this distribution.
- Capture rate of each molecule type $j$ is drawn from $cr_j \in N(1, 0.05)$
- For each cell $i$, a multinomial distribution is used to draw $ls_i$ molecules from molecule type $j$ with probability $cr_j \times ab_{i,j}$ with $ab_{i,j}$ the molecule abundance level of molecule $j$ in cell $i$.

### 2.4.7 Example runs of predefined backbones

#### Linear

#### Bifurcating

#### Cycle

#### Branching

(and binary tree, consecutive bifurcating, trifurcating)

**Converging**

**Bifurcating converging**

**Bifurcating cycle**

**Bifurcating loop**

**Disconnected**

### 2.4.8 Example use cases

**Trajectory alignment**

From discussion: Adding batch effects to snapshot simulations of linear (or even branching) trajectories allows evaluating trajectory alignment methods – which attempt to map two or more trajectories onto each other.

**Differential network inference**

From discussion: Adding perturbations to the GRN allows evaluating the performance of differential network inference methods – which predict differential regulatory interactions between two or more groups of profiles.

**RNA velocity**

From discussion: Sampling a cell at a certain time point and once more at a later time point allows evaluating the performance of RNA velocity approaches – which predict the future state of a cell by looking at differences in pre-mRNA and mRNA abundance levels.

**Perturbation experiment**

# CHAPTER 3

## dynbenchmark: A comparison of single-cell trajectory inference methods

**Abstract:** Trajectory inference approaches analyze genome-wide omics data from thousands of single cells and computationally infer the order of these cells along developmental trajectories. Although more than 70 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. 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. Based on these results, we develop a set of guidelines to help users select the best method for their dataset. Our freely available data and evaluation pipeline (benchmark.dynverse.org) will aid in the development of improved tools designed to analyze increasingly large and complex single-cell datasets.

Adapted from:
Saelens, W.*, **Cannoodt, R.***, Todorov, H., and Saeys, Y. A comparison of single-cell trajectory inference methods. *Nature Biotechnology* 37, 5 (2019), 547–554. doi:10.1038/s41587-019-0071-9.

* Equal contribution

## 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 [22, 23]. Such dynamic processes can be modeled computationally using trajectory inference (TI) methods, also called pseudotime analysis, which order cells along a trajectory based on similarities in their expression patterns [49, 24, 50]. The resulting trajectories are most often linear, bifurcating or tree-shaped, but more recent methods also identify more complex trajectory topologies, such as cyclic [51] or disconnected graphs [52]. TI methods offer an unbiased and transcriptome-wide understanding of a dynamic process[22], thereby allowing the objective identification of new (primed) subsets of cells [53], delineation of a differentiation tree [54, 55] and inference of regulatory interactions responsible for one or more bifurcations [28]. Current applications of TI focus on specific subsets of cells, but ongoing efforts to construct transcriptomic catalogs of whole organisms [25, 26, 27] underline the urgency for accurate, scalable [28, 29] 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 1). Indeed, in several repositories listing single-cell tools, such as omictools.org [56], the 'awesome-single-cell' list [57] and scRNA-tools.org [58], 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 [59, 53, 60, 61] or bifurcating trajectories [62, 63]) or through parameters provided by the user [64, 65]. These methods therefore mainly focus on correctly ordering the cells along the fixed topology. More recent methods also infer the topology [66, 67, 52], 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 [62, 68, 69, 65, 70, 24, 71, 72, 52], 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.

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 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

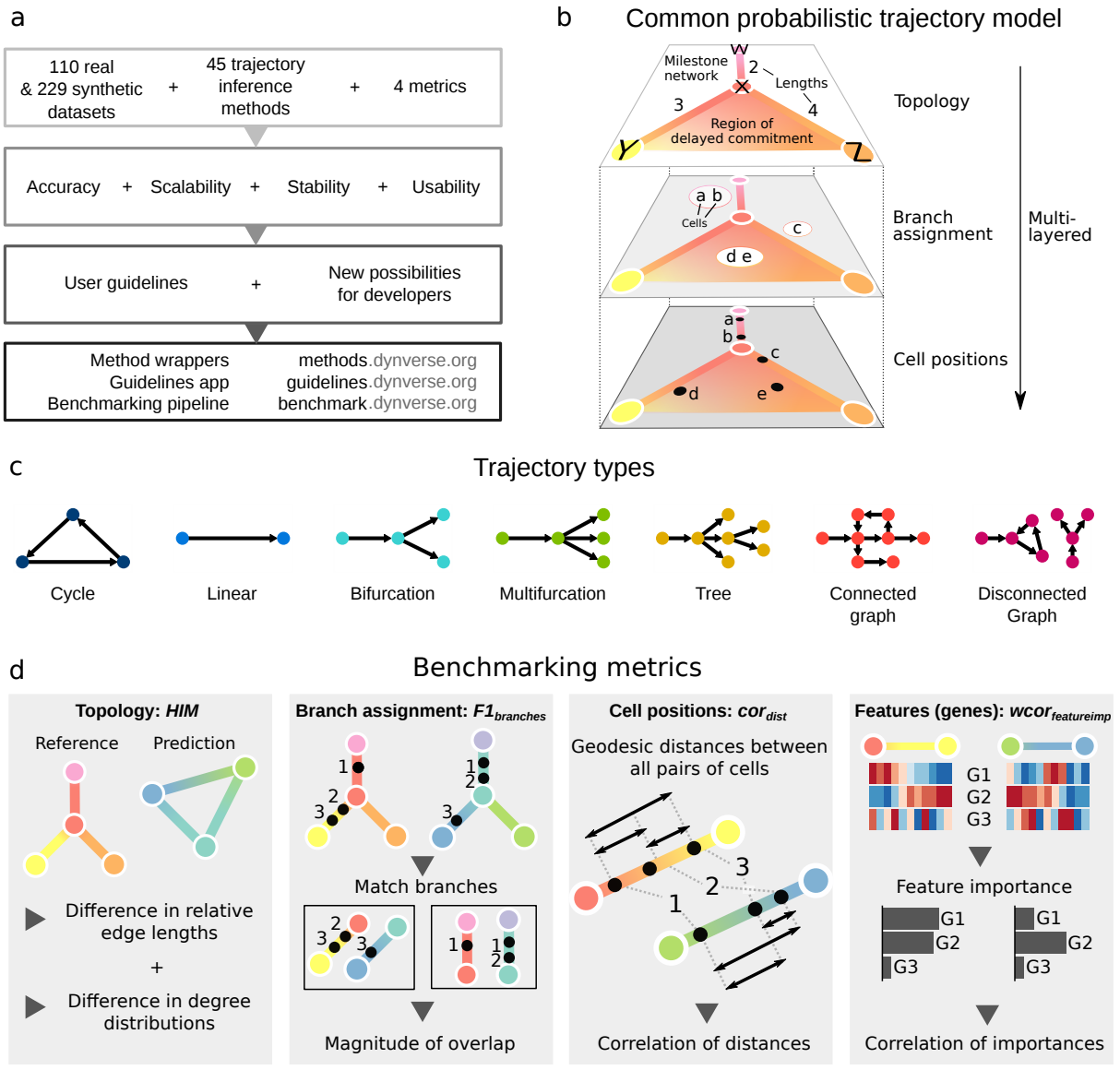



**Figure 3.1: Overview of several key aspects of the evaluation.** **a**, A schematic overview of our evaluation pipeline. **b**, To make the trajectories comparable to each other, a common trajectory model was used to represent reference trajectories from the real and synthetic datasets, as well as 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.

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

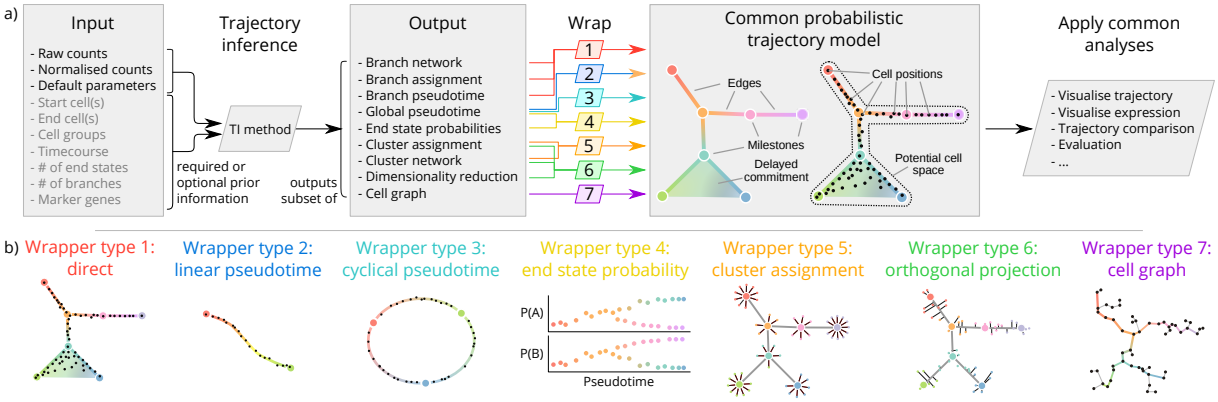



**Figure 3.2: A common interface for TI methods.** **a** The input and output of each TI method is standardized. As input, each TI method receives either raw or normalized counts, several parameters, and a selection of prior information. After its execution, a method uses one of the seven wrapper functions 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.

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, Slingshot and SCORPIUS, performing well across the board (Figure 3.3b). We will discuss each evaluation criterion in more detail (Figure 3.4 and Supplementary Fig. 2), 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 (Supplementary Note 1). Based on an analysis of their robustness and conformity to a set of rules (Supplementary Note 1), 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 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 [73]. 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 [39]. 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 [34].

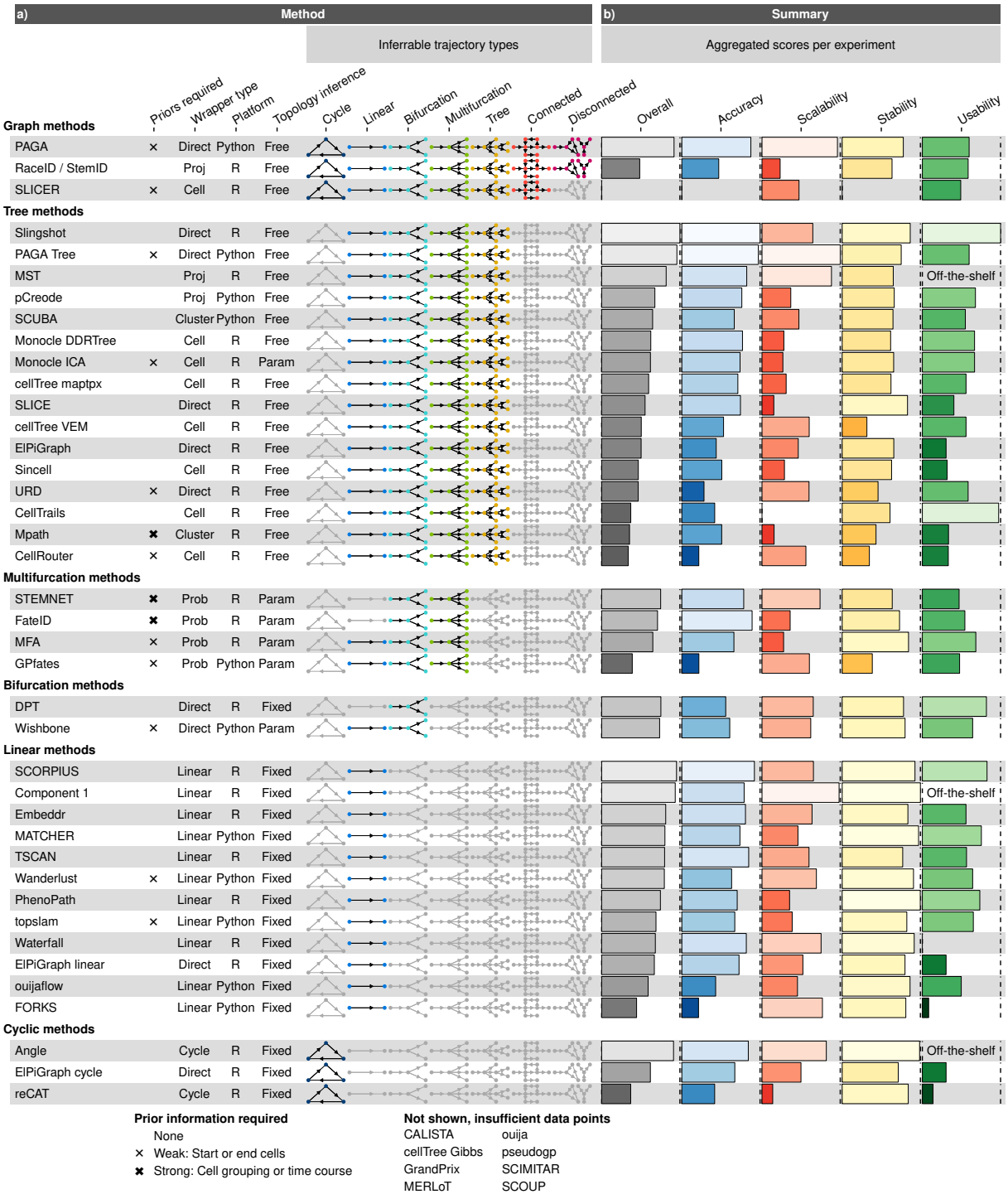



**Figure 3.3: A characterization of the 45 methods evaluated in this study and their overall evaluation results. a**, We characterized the methods according to the wrapper type, their required priors, whether the inferred topology is constrained by the algorithm (fixed) or a parameter (param), and the 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 Fig. 2.

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

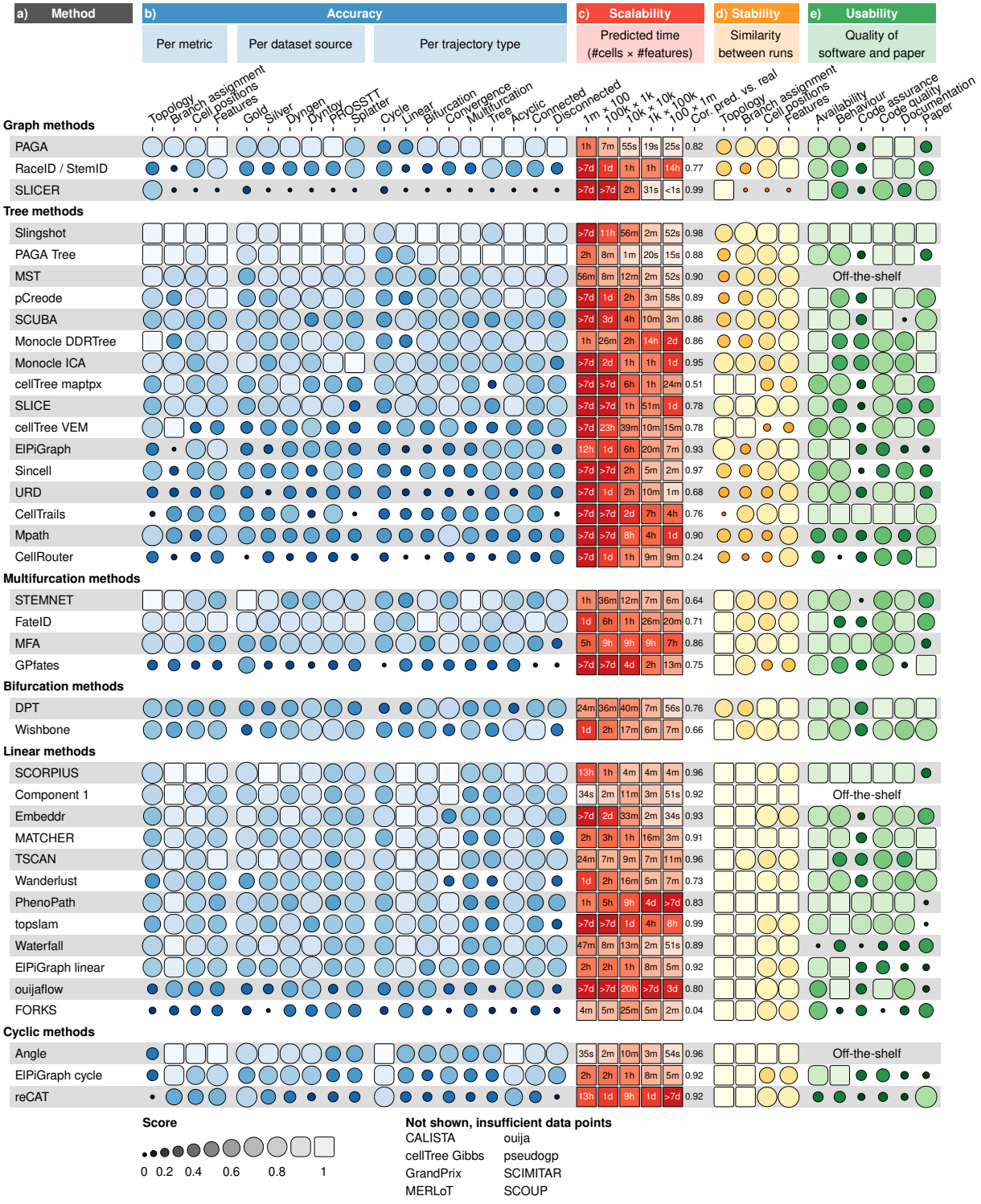



**Figure 3.4: Detailed results of the four main evaluation criteria: accuracy, scalability, stability and usability. a,** The names of the methods, ordered as in Figure 3.3. **b**, Accuracy of trajectory inference methods across metrics, dataset sources and dataset trajectory types. The performance of 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.

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 [74]. Moreover, the recent application of TI methods on multi-omics single-cell data also showcases the increasing demands on the number of features [75]. 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 [76] (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, Supplementary Fig. 5), 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 Fig. 2a).

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 [74]. Running

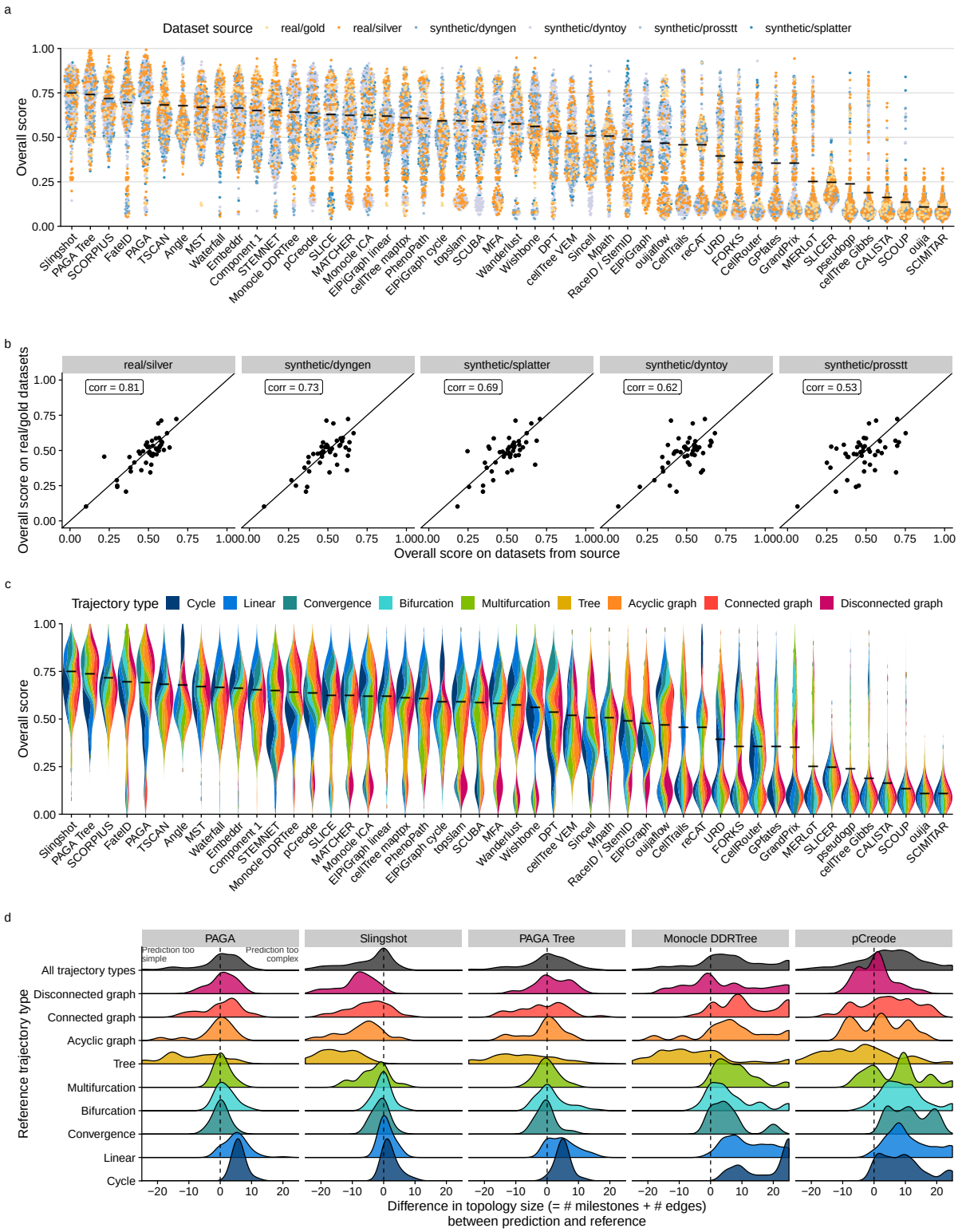



**Figure 3.5: Accuracy of trajectory inference methods.** **a** Overall score for all methods across 339 datasets, colored by the source of the datasets. Black line indicates the mean. **b** Similarity between the overall scores of all dataset sources, compared to real datasets with a gold standard, 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.

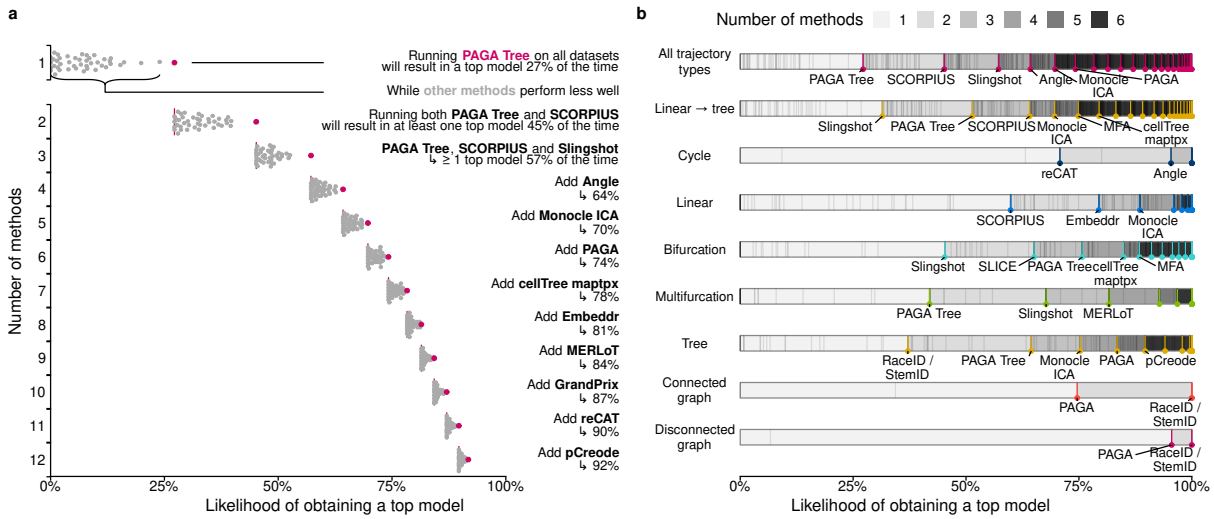



**Figure 3.6: Complementarity between different trajectory inference methods.** **a**, We assessed the likelihood for different combinations of methods to lead to a 'top model' (defined as a model with an overall score of at least 95% of the best model) when applied to all datasets. **b**, The likelihood 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.

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 Fig. 2c). 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

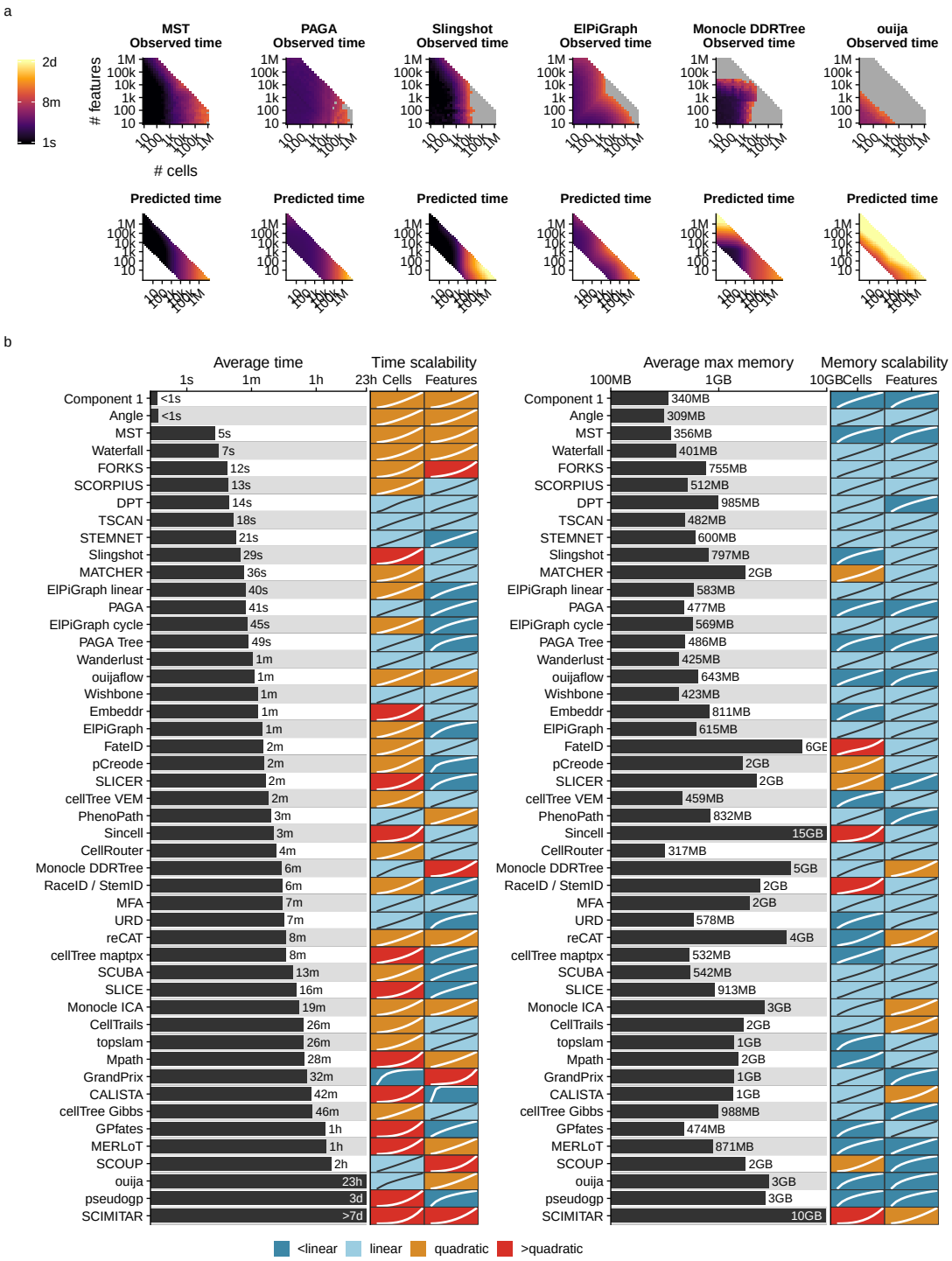



**Figure 3.7: Scalability of trajectory inference methods.** **a** Three examples of average observed running times across five datasets (left) and the predicted running time (right). **b** Overview of the scalability results of all methods, ordered by their average predicted running time from (a). We 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.

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 [77]. 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:un Installation issues seem to be quite general in bioinformatics [78] 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 Supplementary Fig. 6). While recent methods had a slightly better quality score than older methods, several quality aspects were consistently lacking for the majority of the methods (Supplementary Fig. 6 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 [79, 80]. 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.

Based on the results of our benchmark, we propose a set of practical guidelines for method users (Figure 3.8 and 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 3.8. 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 app (guidelines.dynverse.org). This app 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

**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.

| Aspect | Items | References |
|---|---|---|
| **Availability** | | |
| Open source | (1) Method's code is freely available (2) The code can be run on a freely available platform | [81, 79, 77, 82, 80, 83, 84] |
| Version control | The code is available on a public version controlled repository, such as Github | [81, 79, 77, 82, 80, 83] |
| 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, ... | [81, 82, 84, 83] |
| Dependencies | (1) Dependencies are clearly stated in the tutorial or in the code (2) Dependencies are automatically installed | [77, 82, 80, 85] |
| License | (1) The code is licensed (2) License allows academic use | [81, 77, 82, 80, 83, 84] |
| 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 | [83] |
| **Code quality** | | |
| Function and object naming | (1) Functions/commands have well chosen names (2) Arguments/parameters have well chosen names | [79, 82] |
| 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 | [79, 82, 80] |
| Code duplication | Duplicated code is minimal | [79, 82] |
| Self-contained functions | The method is exposed to the user as self-contained functions or commands | [86, 77, 83] |
| 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 | [81, 85] |
| **Code assurance** | | |
| Unit testing | Method is tested using unit tests | [81, 79, 86, 82, 83] |
| Unit testing | Tests are run automatically using functionality from the programming language | [81, 79, 86, 82, 83] |
| Continuous integration | The method uses continuous integration, for example on Travis CI | [87, 82, 80, 83] |
| Code coverage | (1) The code coverage of the repository is assessed. (2) What is the percentage of code coverage | |
| **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 | [79, 82, 80, 83, 84] |
| 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. | [88] |
| 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) | [82, 83, 84, 85, 89] |
| 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 | [79, 77, 82, 83, 85] |
| Inline documentation | Inline documentation is present in the code | [79, 77, 82, 83, 85] |
| Parameter transparency | All important parameters are exposed to the user | [77] |
| **Behaviour** | | |
| Seed setting | The method does not artificially become deterministic, for example by setting some (0.5) or a lot (1) of seeds | [90] |
| 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 | [80] |
| 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 | [85, 91, 92] |
| 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 | [93, 94] |
| Evaluation of robustness | The paper assessed method robustness (to eg. noise, subsampling, parameter changes, stability) in one (0.5) or several (1) ways | [85, 93, 89, 94] |

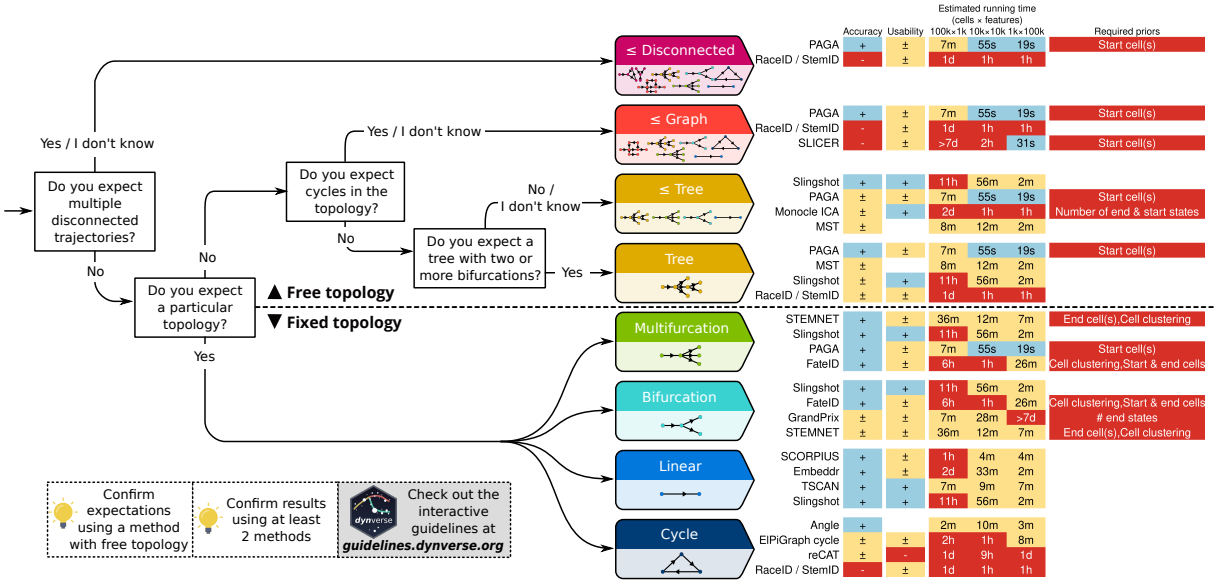



**Figure 3.8: Practical guidelines for method users.** As the performance of a method mostly depends on the topology of the trajectory, the choice of TI method will be primarily influenced by the user's existing knowledge about the expected topology in the data. We therefore devised a set of practical 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.

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 [28] or finding modules of genes [95]. 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 3.9). 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 [96], 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 3.9a,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 3.9c,d). Furthermore, higher standards for code assurance and documentation could help in adopting these tools across the single-cell omics

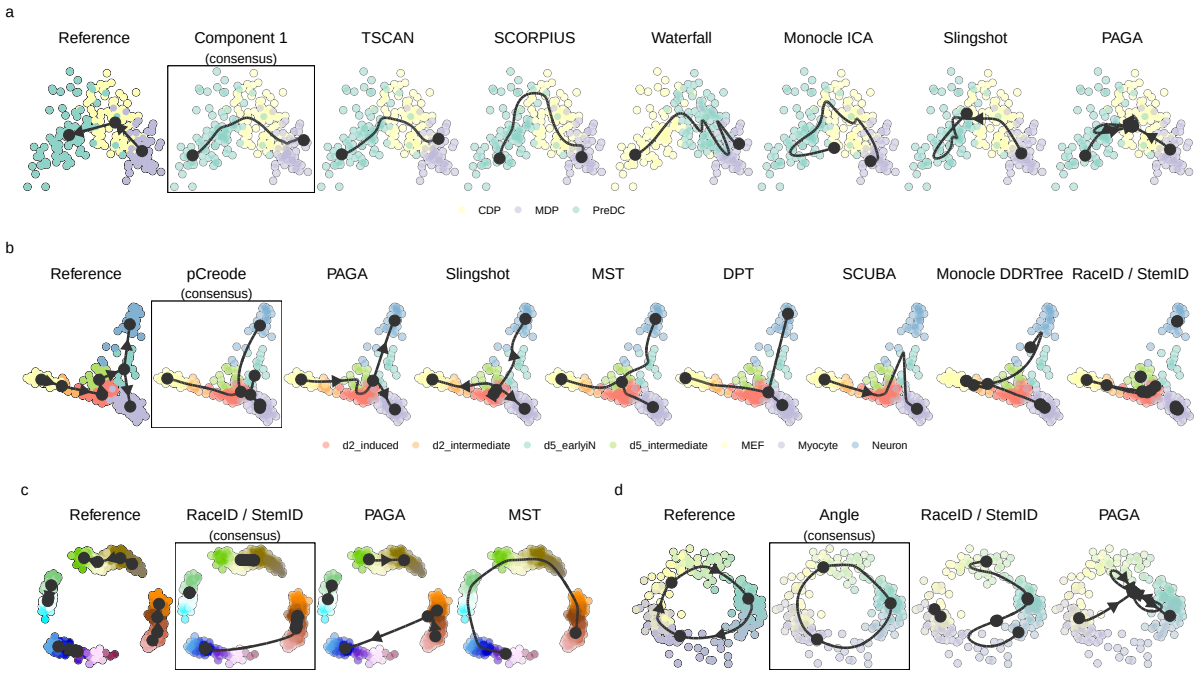



**Figure 3.9: Demonstration of how a common framework for TI methods facilitates broad applicability using some example datasets.** Trajectories inferred by each method were projected to a common dimensionality reduction using multidimensional scaling. For each dataset, we also calculated a 'consensus' 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.

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 [97] 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 uncertainty 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 1) by searching the literature for 'trajectory inference' and 'pseudotemporal ordering', and based on two existing lists found online: https://github.com/seandavi/awesome-single-cell [57] and https://github.com/agitter/single-cell-pseudotime

[98]. 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 $\times$ 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 uniformily discretized 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 [99, 100].
- **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 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 [36]
- **Splatter**: simulations of non-linear paths between different expression states [34]

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 [34]. 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 (https://github.com/dynverse/dyngen) workflow to generate synthetic data is based on the well established workflow used in the evaluation of network inference methods [39, 31] 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 extendable. We simulated a total of 110 datasets, with 11 different topologies.

*Network generation*

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 genes [44] strongly repress each other, forcing one of the

two to become inactive [45]. Such mutual antagonism can be modelled and simulated [46, 47]. Although such a two-gene model is simple and elegant, the reality is frequently more complex, with multiple genes (grouped into modules) repressing each other [48].

To simulate certain trajectory topologies, we therefore designed module networks in which the cells follow a particular trajectory topology given certain parameters. Two module networks generated linear trajectories (linear and linear long), one generated a bifurcation, one generated a convergence, one generated a multifurcation (trifurcating), two generated a tree (consecutive bifurcating and binary tree), one generated an acyclic graph (bifurcating and converging), one generated a complex fork (trifurcating), one generated a rooted tree (consecutive bifurcating) and two generated simple graph structures (bifurcating loop and bifurcating cycle). The structure of these module networks is available at https://github.com/dynverse/dyngen/tree/master/inst/ext_data/modulenetworks.

From these module networks we generated gene regulatory networks in two steps: the main regulatory network was first generated, and extra target genes from real regulatory networks were added. For each dataset, we used the same number of genes as were differentially expressed in the real datasets. 5% of the genes were assigned to be part of the main regulatory network, and were randomly distributed among all modules (with at least one gene per module). We sampled edges between these individual genes (according to the module network) using a uniform distribution between 1 and the number of possible targets in each module. To add additional target genes to the network, we assigned every regulator from the network to a real regulator in a real network (from regulatory circuits [101]), and extracted for every regulator a local network around it using personalized pagerank (with damping factor set to 0.1), as implemented in the page_rank function of the *igraph* package.

*Simulation of gene regulatory systems using thermodynamic models*

To simulate the gene regulatory network, we used a system of differential equations similar to those used in evaluations of gene regulatory network inference methods [31]. In this model, the changes in gene expression ($x_i$) and protein expression ($y_i$) are modeled using ordinary differential equations [39] (ODEs):

$$\frac{dx_i}{dt} = \underbrace{m \times f(y_1, y_2, ...)}_{\text{production}} - \underbrace{\lambda \times x_i}_{\text{degradation}}$$

$$\frac{dy_i}{dt} = \underbrace{r \times x_i}_{\text{production}} - \underbrace{\Lambda \times y_i}_{\text{degradation}}$$

where $m$, $\lambda$, $r$ and $\Lambda$ represent production and degradation rates, the ratio of which determines the maximal gene and protein expression. The two types of equations are coupled because the production of protein $y_i$ depends on the amount of gene expression $x_i$, which in turn depends on the amount of other proteins through the activation function $f(y_1, y_2, ...)$.

The activation function is inspired by a thermodynamic model of gene regulation, in which the promoter of a gene can be bound or unbound by a set of transcription factors, each representing a certain state of the promoter. Each state is linked with a relative activation $\alpha_j$, a number between 0 and 1 representing the activity of the promoter at this particular state. The production rate of the gene is calculated by combining the probabilities of the promoter being in each state with the relative activation:

$$f(y_1, y_2, ..., y_n) = \sum_{j \in \{0,1,...,n^2\}} \alpha_j \times P_j$$

The probability of being in a state is based on the thermodynamics of transcription factor binding. When only one transcription factor is bound in a state:

$$P_j \propto \nu = \left(\frac{y}{k}\right)^n$$

where the hill coefficient $n$ represents the cooperativity of binding and $k$ the transcription factor concentration at half-maximal binding. When multiple regulators are bound:

$$P_j \propto \nu = \rho \times \prod_j \left(\frac{y_j}{k_j}\right)^{n_j}$$

where $\rho$ represents the cooperativity of binding between the different transcription factors.

$P_i$ is only proportional to $\nu$ because $\nu$ is normalized such that $\sum_i P_i = 1$.

To each differential equation, we added an additional stochastic term:

$$\frac{dx_i}{dt} = m \times f(y_1, y_2, ...) - \lambda \times x_i + \eta \times \sqrt{x_i} \times \Delta W_t$$
$$\frac{dy_i}{dt} = r \times x_i - \Lambda \times y_i + \eta \times \sqrt{y_i} \times \Delta W_t$$

with $\Delta W_t \sim \mathcal{N}(0, h)$.

Similar to GeneNetWeaver [39], we sample the different parameters from random distributions, defined as follows. $e$ defines whether a transcription factor activates (1) or represses (-1), as defined within the regulatory network network.

$$r = \mathcal{U}(10, 200)$$
$$d = \mathcal{U}(2, 8)$$
$$p = \mathcal{U}(2, 8)$$
$$q = \mathcal{U}(1, 5)$$
$$a_0 = \begin{cases} 1 & \text{if } |e| = 0 \\ 1 & \text{if } \forall x \in e,\ x = -1 \\ 0 & \text{if } \forall x \in e,\ x = 1 \\ 0.5 & \text{otherwise} \end{cases}$$
$$a_i = \begin{cases} 0 & \text{if } \exists x \in e_i,\ x = -1 \\ 1 & \text{otherwise} \end{cases}$$
$$s = \mathcal{U}(1, 20)$$
$$k = y_{max}/(2 * s),$$
$$\text{where } y_{max} = r/d \times p/q$$
$$c = \mathcal{U}(1, 4)$$

We converted each ODE to an SDE by adding a chemical Langevin equation, as described in [39]. These SDEs were simulated using the Euler–Maruyama approximation, with time-step $h = 0.01$ and noise strength $\eta = 8$. The total simulation time varied between 5 for linear and bifurcating datasets, 10 for consecutive bifurcating, trifurcating and converging datasets, 15 for bifurcating converging

datasets and 30 for linear long, cycle and bifurcating loop datasets. The burn-in period was for each simulation 2. Each network was simulated 32 times.

*Simulation of the single-cell RNA-seq experiment*

For each dataset we sampled the same number of cells as were present in the reference real dataset, limited to the simulation steps after burn-in. These cells were sampled uniformly across the different steps of the 32 simulations. Next, we used the Splatter package [34] to estimate the different characteristics of a real dataset, such as the distributions of average gene expression, library sizes and dropout probabilities. We used Splatter to simulate the expression levels $\lambda_{i,j}$ of housekeeping genes $i$ (to match the number of genes in the reference dataset) in every cell $j$. These were combined with the expression levels of the genes simulated within a trajectory. Next, true counts were simulated using $Y'_{i,j} \sim \text{Poisson}(\lambda_{i,j})$. Finally, we simulated dropouts by setting true counts to zero by sampling from a Bernoulli distribution using a dropout probability $\pi^D_{i,j} = \frac{1}{1+e^{-k(\ln(\lambda_{i,j})-x_0)}}$. Both $x_0$ (the midpoint for the dropout logistic function) and $k$ (the shape of the dropout logistic function) were estimated by Splatter.

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

*Gold standard extraction*

Because each cellular simulation follows the trajectory at its own speed, knowing the exact position of a cell within the trajectory topology is not straightforward. Furthermore, the speed at which simulated cells make a decision between two or more alternative paths is highly variable. We therefore first constructed a backbone expression profile for each branch within the trajectory. To do this, we first defined in which order the expression of the modules is expected to change, and then generated a backbone expression profile in which the expression of these modules increases and decreases smoothly between 0 and 1. We also smoothed the expression in each simulation using a rolling mean with a window of 50 time steps, and then calculated the average module expression along the simulation. We used dynamic time warping, implemented in the dtw R package [102, 103], with an open end to align a simulation to all possible module progressions, and then picked the alignment which minimised the normalised distance between the simulation and the backbone. In case of cyclical trajectory topologies, the number of possible milestones a backbone could progress through was limited to 20.

**dyntoy**

For more simplistic data generation ("toy" datasets), we created the dyntoy workflow (https://github.com/dynverse/dyntoyhttps://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 [36], 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 $a \sim U(0.01, 0.1)$, $branch\text{-}tol_{intra} \sim U(0, 0.9)$ and $branch\text{-}tol_{inter} \sim U(0, 0.9)$. 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 centered 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 [34] 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 [104]. 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 [97]. 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 Supplementary Note 1 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 (Supplementary Note 1). 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 [105] 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 $\gamma$, which was fixed at 0.1 to make the scores comparable between datasets. We illustrate the metric and discuss alternatives in Supplementary Note 1.

#### The F1 between branch assignments

To compare branch assignment, we used an F1 score, also used used for comparing biclustering methods [95]. 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 Supplementary Note 1.

#### 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 Supplementary Note 1, 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 Supplementary Note 1.

#### 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 [106]), 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 Supplementary Note 1.

#### 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 Supplementary Note 1.

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 [76], 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:

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

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 [87] 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.c

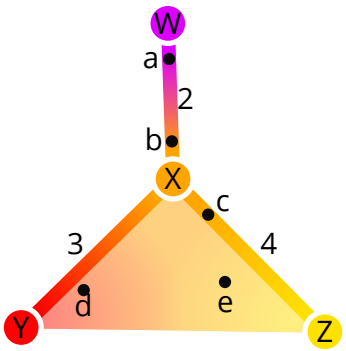
developed using the R shiny package. This app will also filter the methods, among other things, depending on the predicted running time and memory requirements, the prior information available and the preferred execution environment (using the dynmethods package or standalone).

### 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 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 3.10).




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

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).

### 3.5.1 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 [107], 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 [108]. 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) [105] which was adopted from the R nettools package (https://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 [109], 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 [110], 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 $\gamma$ 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 3.11a). While conceptually very different, the *edgeflip* and *HIM* still produce similar scores (Figure 3.11b). The *HIM* tends to punish the detection of cycles, while the *edgeflip* is more harsh for differences in the number of bifurcations (Figure 3.11b). The main difference however is that the *HIM* takes into account edge lengths when comparing two trajectories, as illustrated in (Figure 3.11c). 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 3.12. These clusters of cells can then be compared using one of the many external cluster evaluation measures [95]. 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:

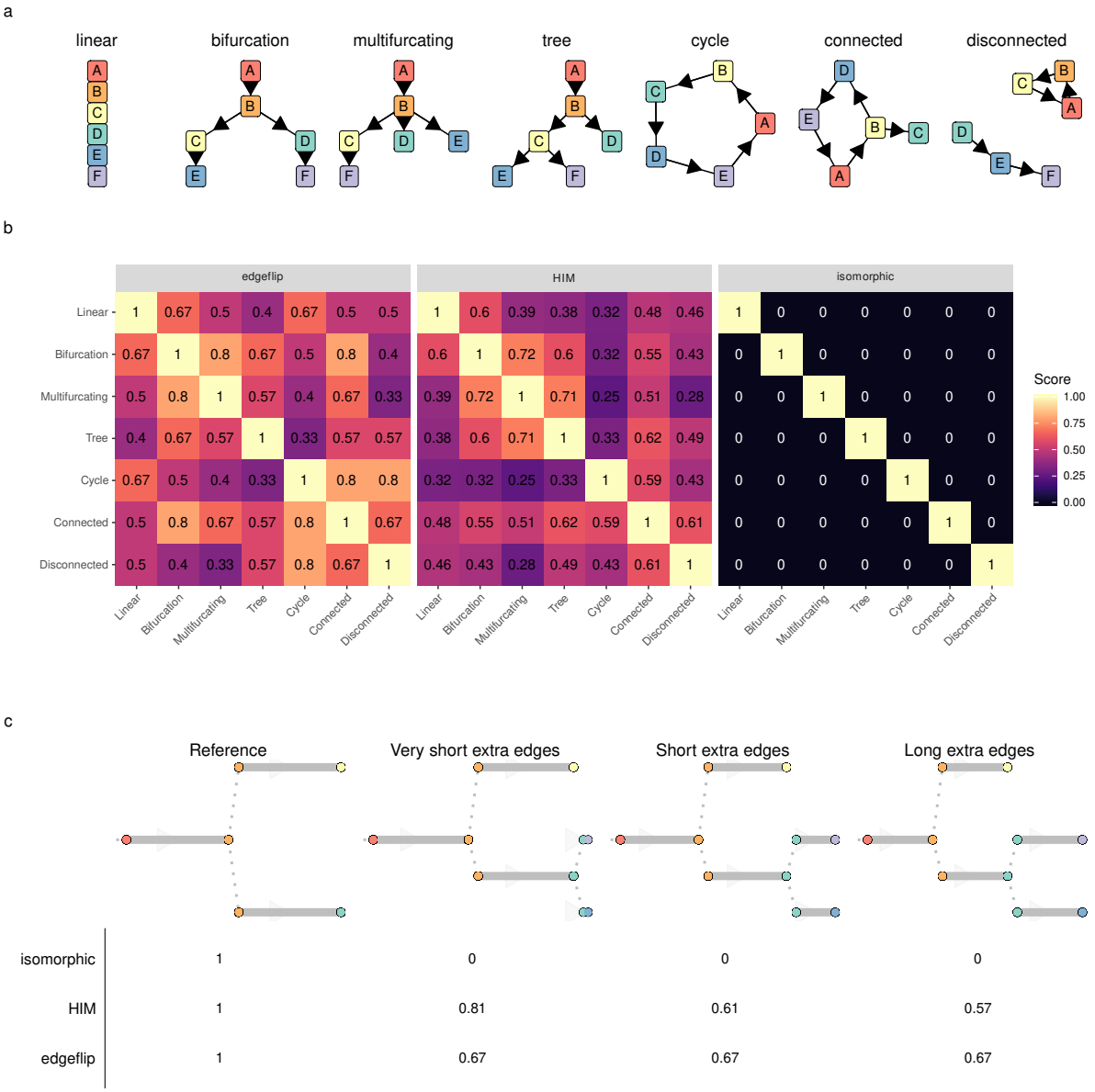



**Figure 3.11: Showcase of three metrics to evaluate topologies: *isomorphic*, *edgeflip* and *HIM* (a)** The used topologies. **(b)** The scores when comparing each pair of trajectory types. (c) Four datasets in which aan extra edge is added and made progressively longer. This shows how the HIM can take into account edge lengths.

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

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

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

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

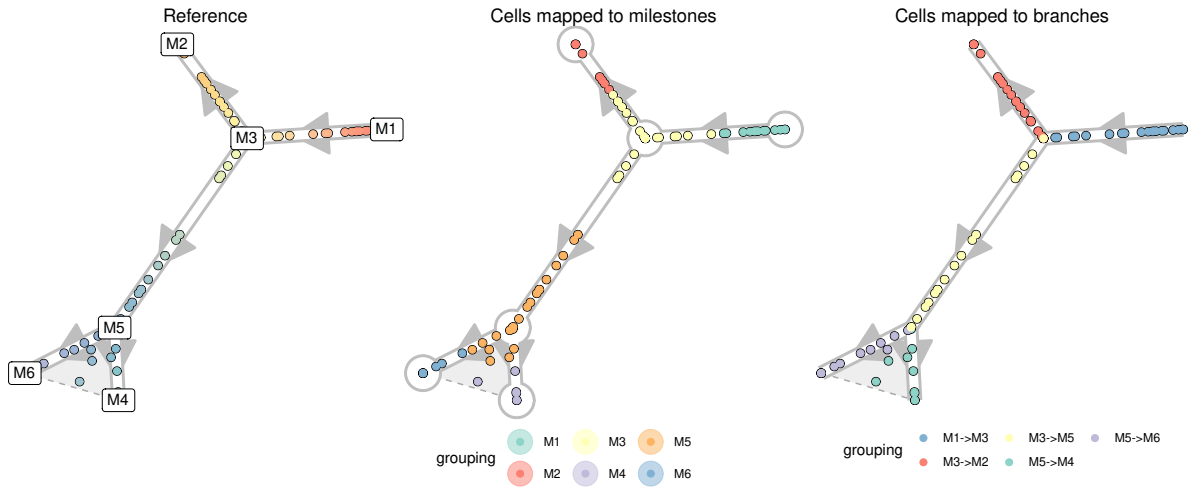
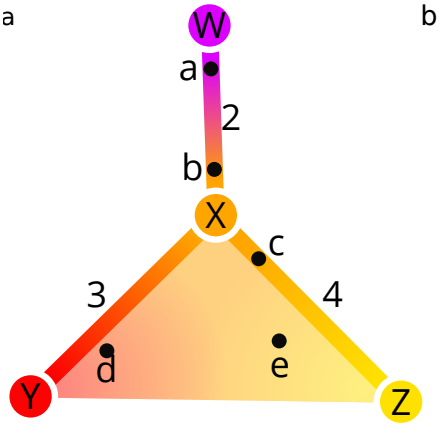



**Figure 3.12: Mapping cells to their closest milestone or branch for the calculation of the $F1_{milestones}$ and $F1_{branches}$.** To calculate the $F1_{milestones}$, cells are mapped towards the nearest milestone, i.e. the milestone with the highest milestone percentage. For the $F1_{branches}$, the cells are mapped to the closest edge.

### $cor_{dist}$: 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.

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

d

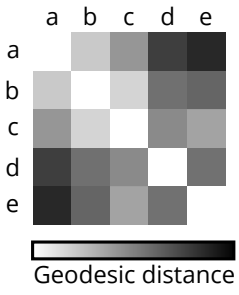



**Figure 3.13: The calculation of geodesic distances on a small example trajectory. a)** A toy example containing four milestones (W to Z) and five cells (a to e). **b)** The corresponding milestone network, milestone percentages and regions of delayed commitment, when the toy trajectory is converted to the common trajectory model. **c)** The calculations made for calculating the pairwise geodesic distances. **d)** A heatmap representation of the pairwise geodesic distances.

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 3.13:

- **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 3.14a). 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 3.14a). 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 3.15) 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.

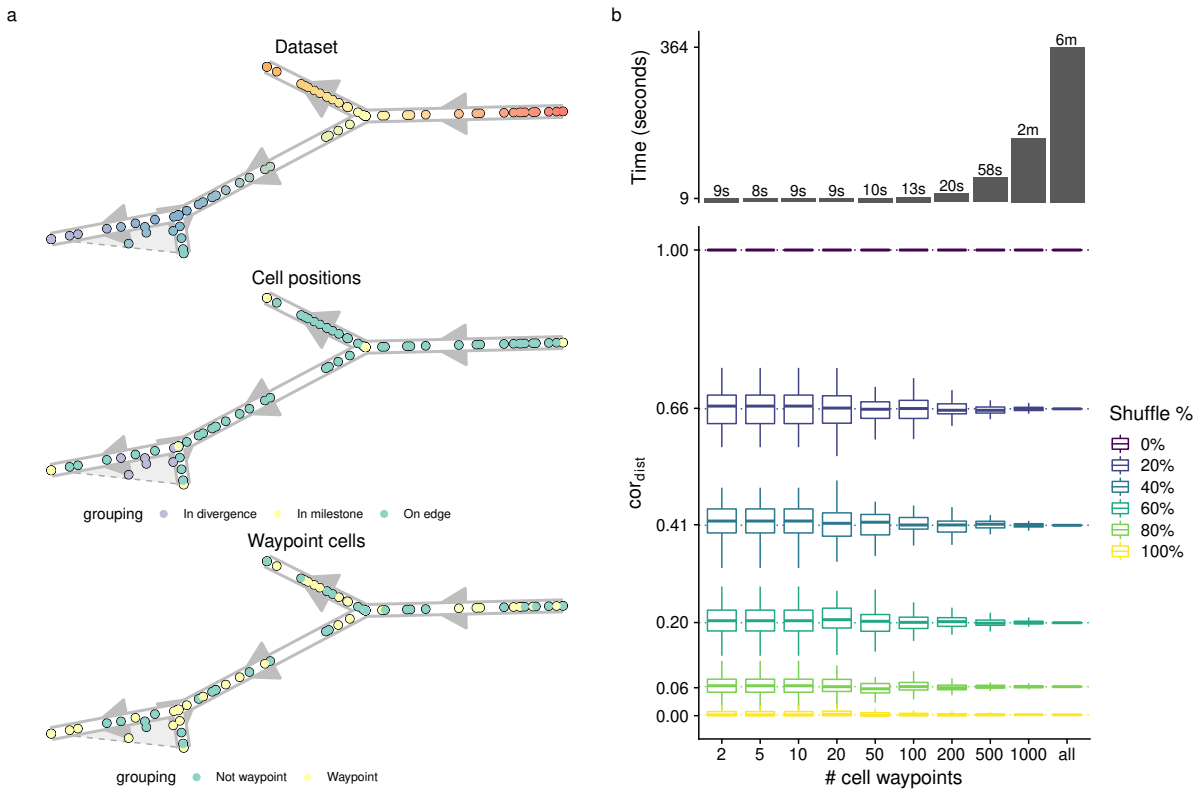



**Figure 3.14: Determination of cell waypoints a)** Illustration of the stratified cell sampling using an example dataset (top). Each milestone, edge between two milestones and region of delayed commitment is seen as a collection of cells (middle), and the number of waypoints (100 in this case) are divided 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.

Specifically, we implemented two metrics which predict the milestone percentages from the reference by using the predicted milestone percentages as features (Figure 3.16). We did this with two regression methods, linear regression (*lm*, using the R lm function) and Random Forest (*rf*, implemented in the *ranger* package [106]). 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.

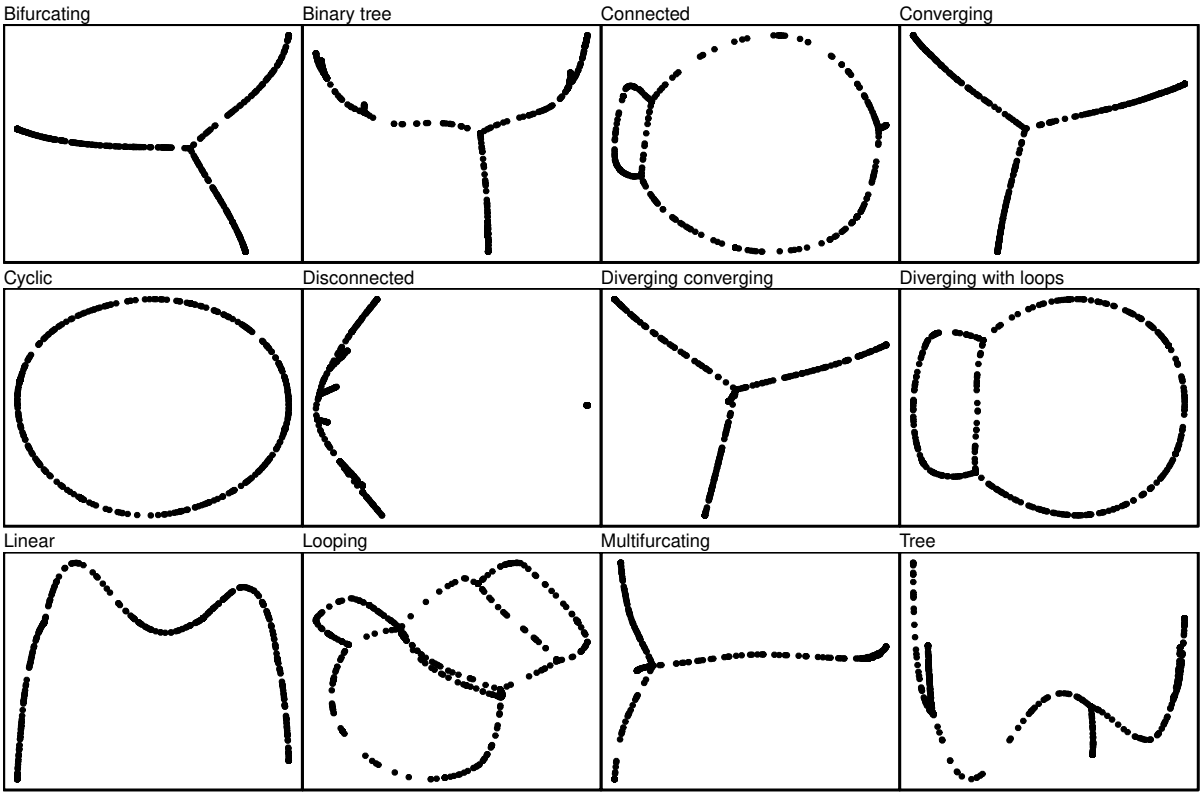



**Figure 3.15: Determination of cell waypoints.** We generated different toy trajectory datasets with varying topologies and calculated the geodesic distances between all cells within the trajectory. We then used these distances as input for classical multidimensional scaling. This shows that the geodesic distances do not only contain information regarding the cell's positions, but also information on the lengths and wiring of the topology.

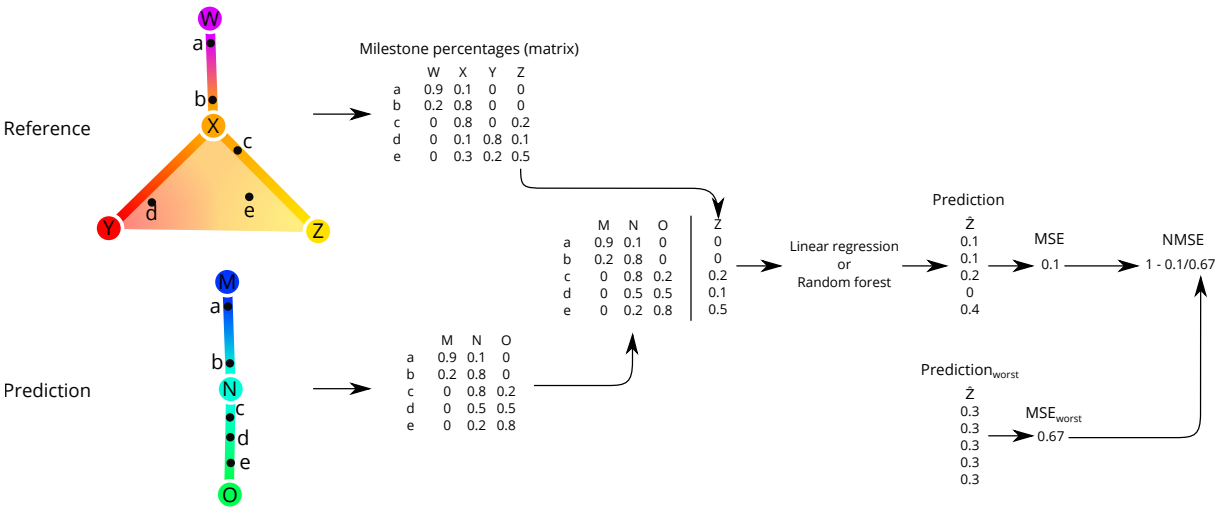



**Figure 3.16: The calculation of $NMSE_{lm}$ distances on a small example trajectory.** The milestone percentages of the reference are predicted based on the milestone percentages of the prediction, using regression models such as linear regression or random forests. The predicted trajectory is then scored by comparing the mean-squared error (MSE) of this regression model with the baseline MSE where the prediction is the average milestone percentage.

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 [106], 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 3.17. 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.

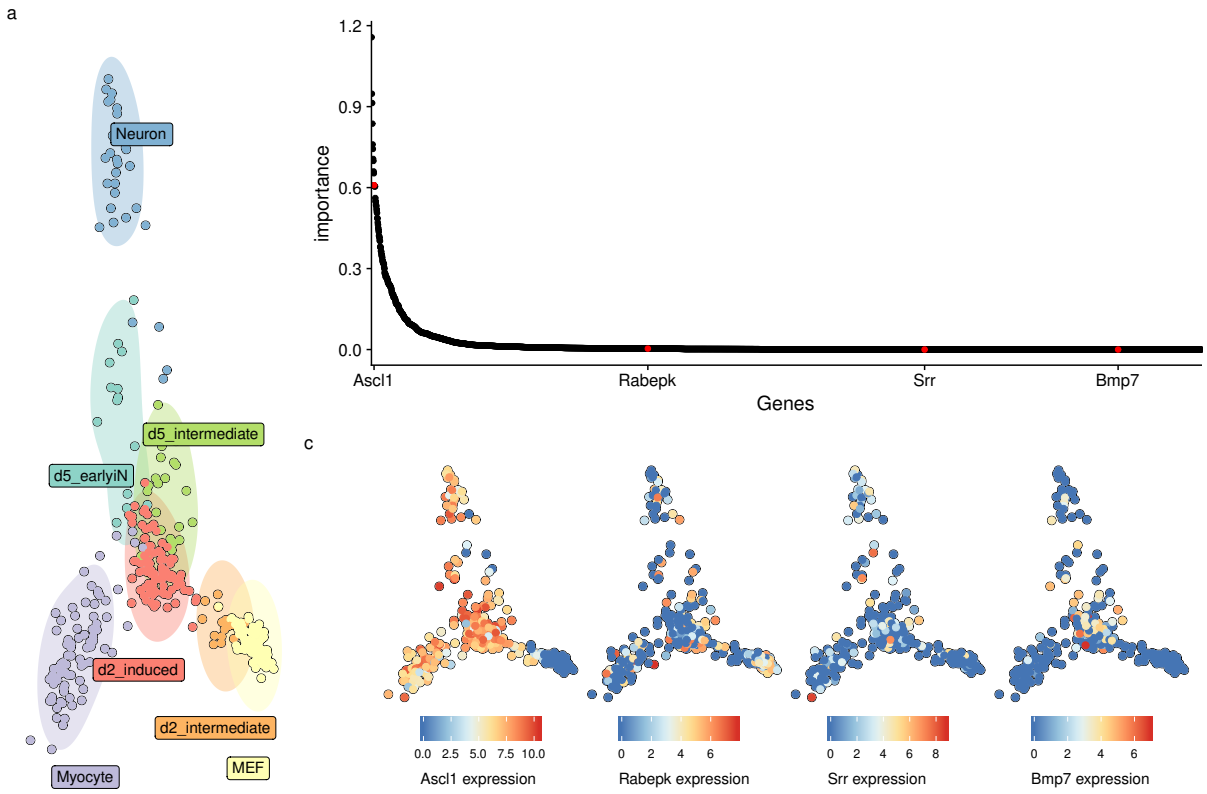



**Figure 3.17: An illustration of ranking features based on their importance in a trajectory.** (a) A MDS dimensionality reudction of a real dataset in which mouse embryonic fibroblasts (MEF) differentiate into Neurons and Myocytes. (b) The ranking of feature importances from high to low. The majority 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.

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 3.18. 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 3.18 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 3.19).

### 3.5.2 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 bench-

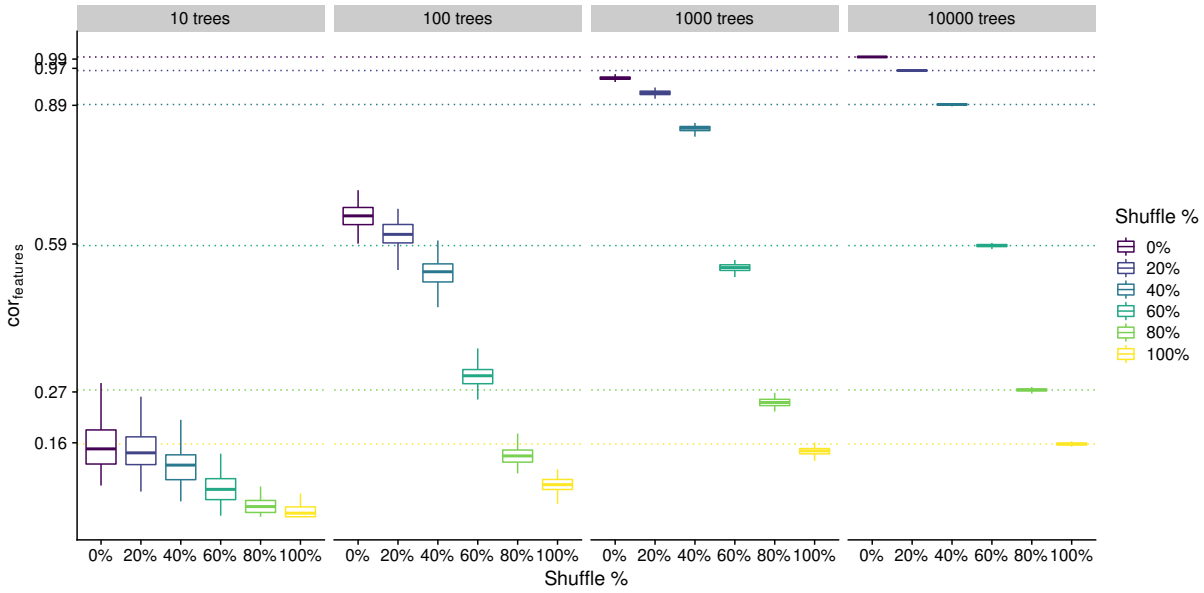



**Figure 3.18: Effect of the number of trees parameter on the accuracy and variability of the $cor_{features}$.** We used the dataset from Figure 3.17 and calculated the $cor_{features}$ after shuffling a percentage of cells.

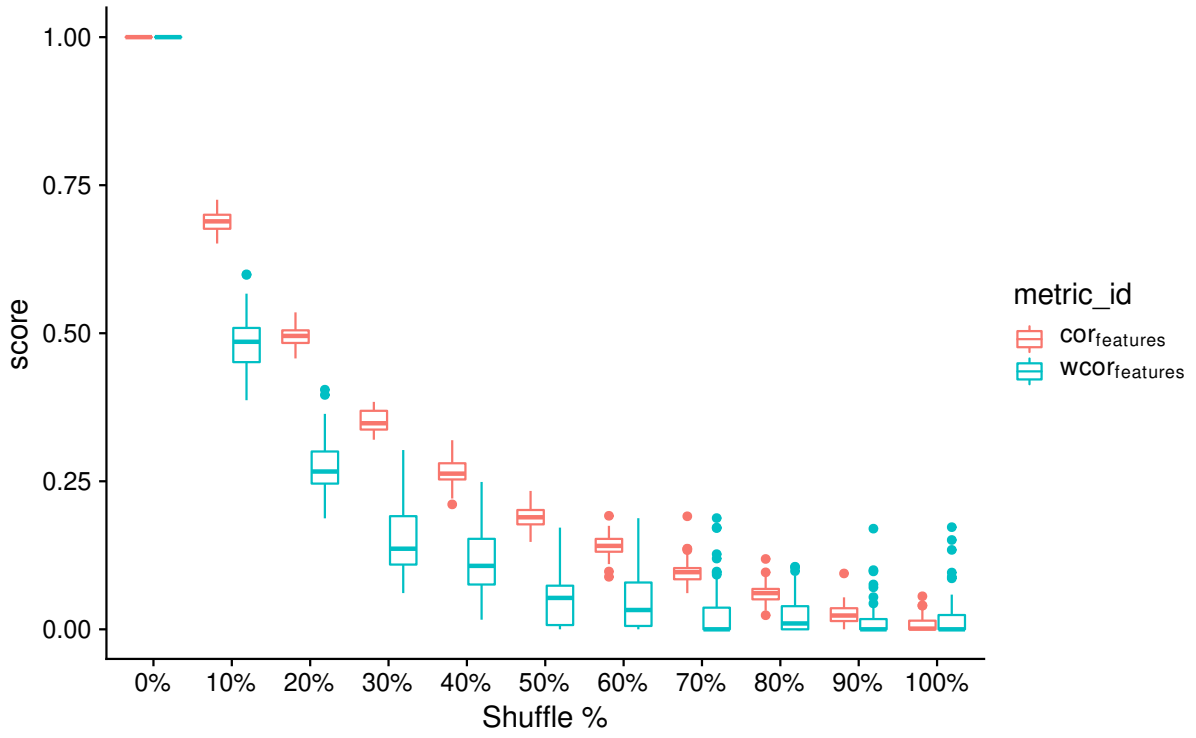



**Figure 3.19: Effect of weighting the features based on their feature importance in the reference.** We used the same dataset as in Figure 3.17, and calculated the $cor_{features}$ after shuffling a percentage of cells.

marking. 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

**Table 3.2:** 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 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

- 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 3.2. 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.

### 3.5.3 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 3.20. 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 3.20). 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 3.21.

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 3.22a). 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 3.22b). Finally, the scores are aggregated over the different trajectory types again using a arithmetic mean (Figure 3.22c).

### 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

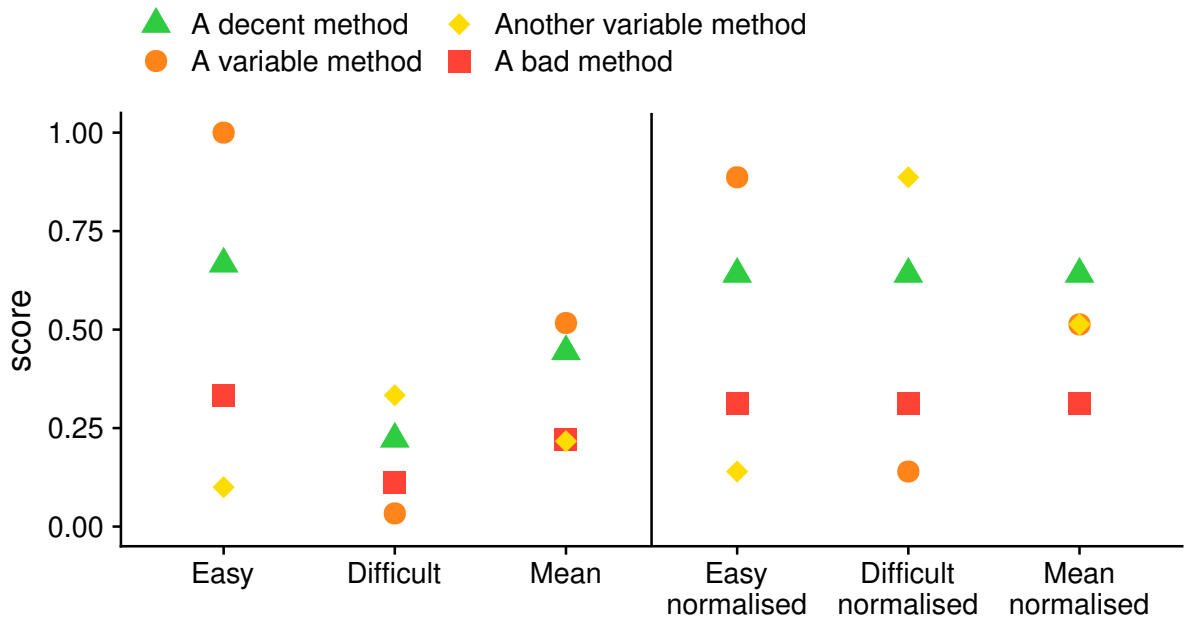



**Figure 3.20: An illustration of how the difficulty of a dataset can influence the overall ranking.** A decent method, which consistently ranks high on an easy and difficult dataset, does not get a high score when averaging. On the other hand, a method which ranks high on the easy dataset, but very low on the difficult dataset does get a high score on average. After normalising the scores (right), this problem disappears.

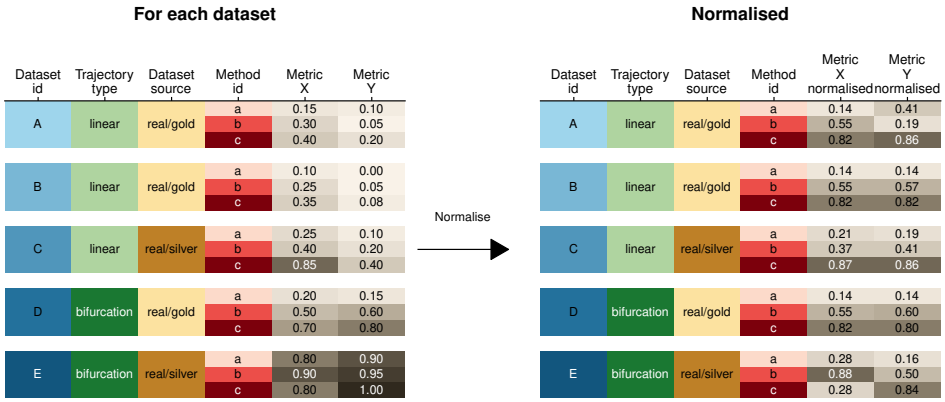



**Figure 3.21: An example of the normalisation procedure.** Shown are some results of a benchmarking procedure, where every row contains the scores of a particular method (red shading) on a particular dataset (blue shading), with a trajectory type (green shading) and dataset source (orange shading).

- 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 $cor_{dist}$

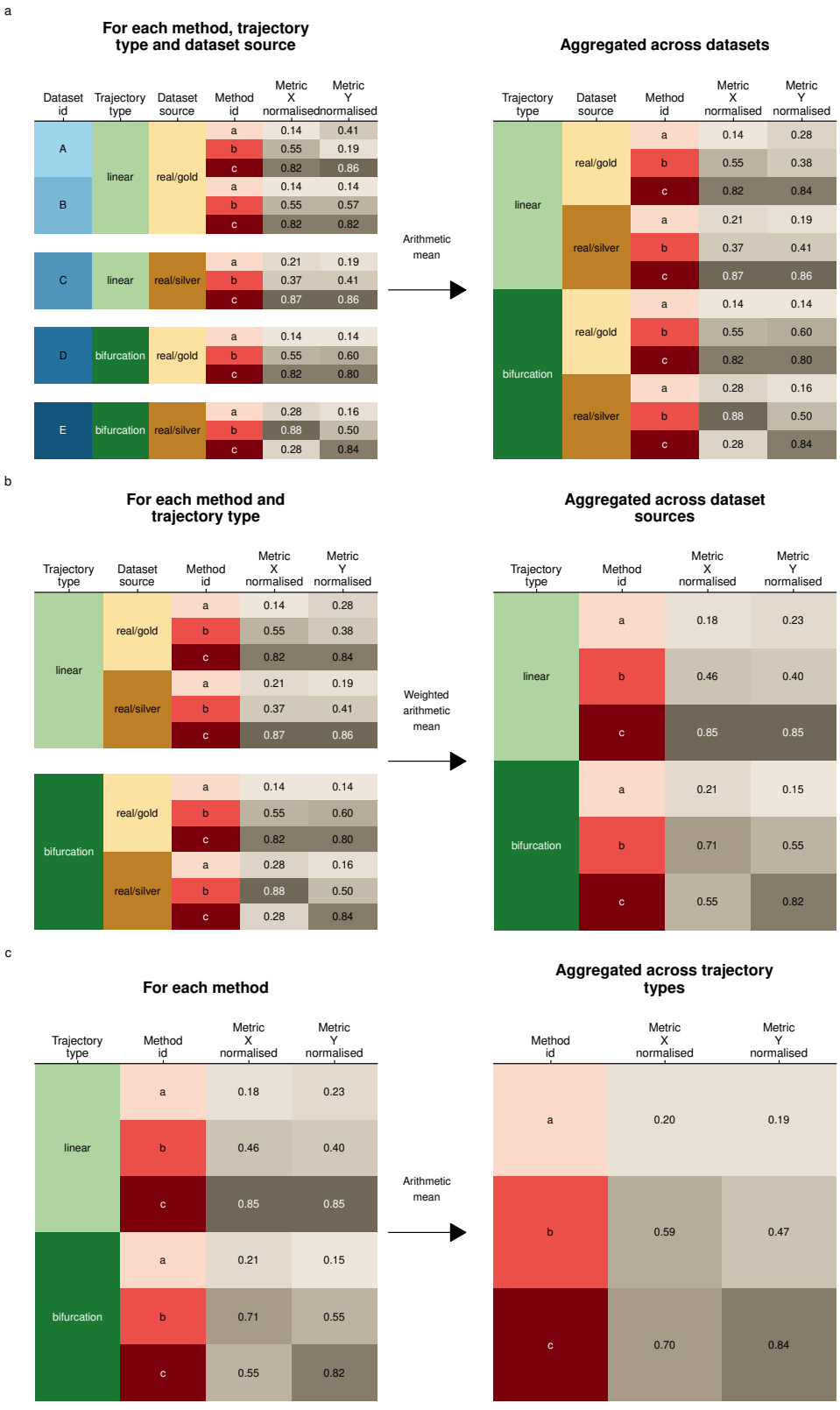



**Figure 3.22: An example of the aggregation procedure.** In consecutive steps we aggregated across **(a)** different datasets with the same source and trajectory type, **(b)** different dataset sources with the same trajectory type (weighted for the correlation of the dataset source with the real gold dataset source) and **(c)** all trajectory types.

- 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 3.23) for a method was thus defined as:

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

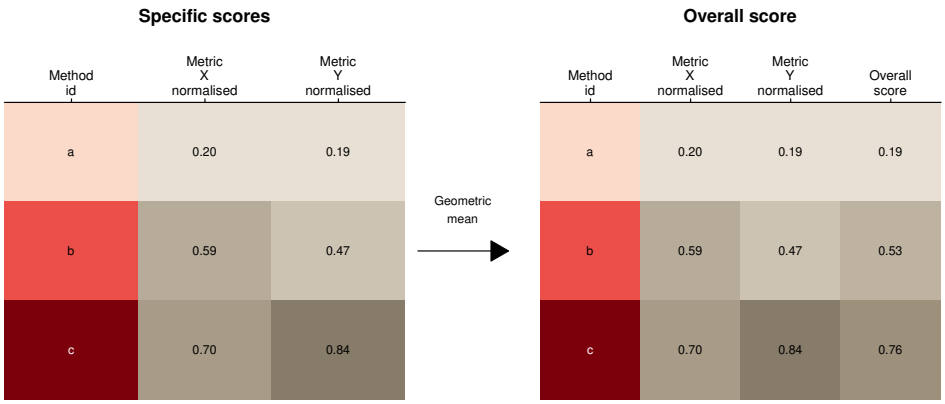



**Figure 3.23: An example of the averaging procedure.** For each method, we calculated the geometric mean between its normalised and aggregated scores

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).

# CHAPTER 4

## SCORPIUS: Fast, accurate, and robust single-cell pseudotime

**Abstract:** Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Adapted from:
**Cannoodt, R.**, Saelens, W., Sichien, D., Tavernier, S., Janssens, S. Guilliams, M., Lambrecht, B., De Preter, K., and Saeys, Y. SCORPIUS improves trajectory inference and identifies novel modules in dendritic cell development. *Journal* vol, issue (2019), page–page. doi:10.1101/079509v2.

## 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 [23, 22]. 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 [24].

In linear TI, also sometimes called pseudotemporal ordering, the user assumes that the dynamic process of interest is linear and is interested how gene expression changes along the dynamic process. Linear TI is 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 [38], 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 [53].

## 4.2 Results

In essence, SCORPIUS reduces the dimensionality of the dataset using Multi-Dimensional Scaling (MDS) [111], and derives a smooth curve that goes through the middle of the dataset using principal curves [112] (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 [113]. 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.

### 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 [38]. 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 segments (e.g. 100), again the execution time of the principal curve algorithm is reduced significantly (Figure 4.3B).

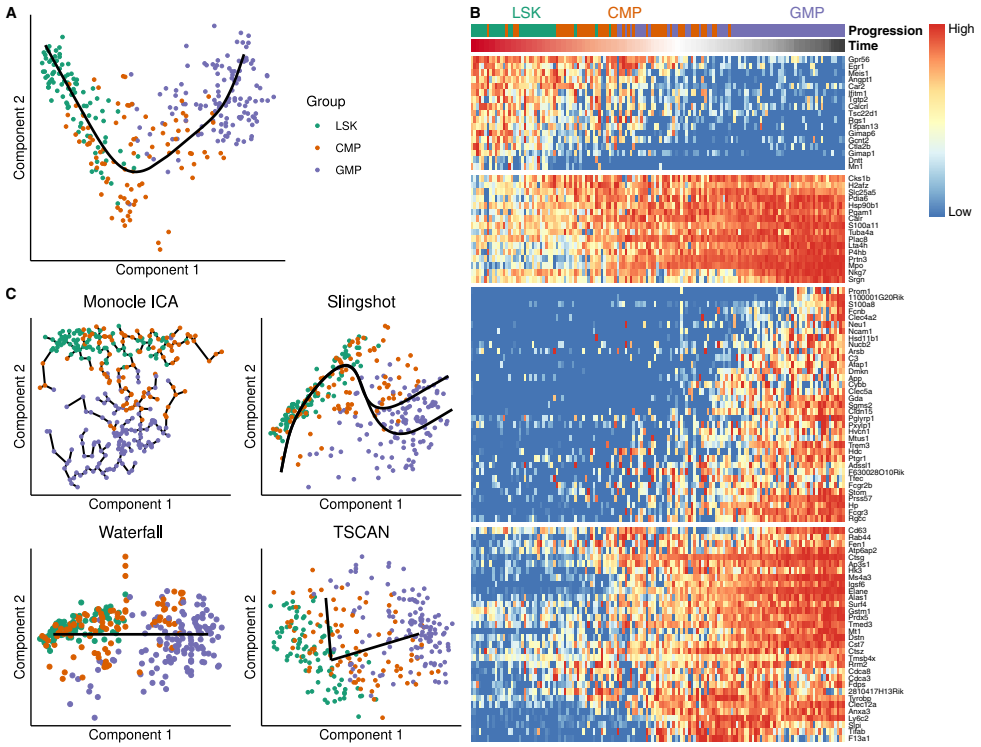



**Figure 4.1: A:** SCORPIUS derives a smooth curve that passes through the middle of the dataset. **B:** Prioritising genes in function of the pseudotemporal ordering allows easier interpretation of the dynamic process at hand. **C:** Low accuracy predictions are a result of false positive branches or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

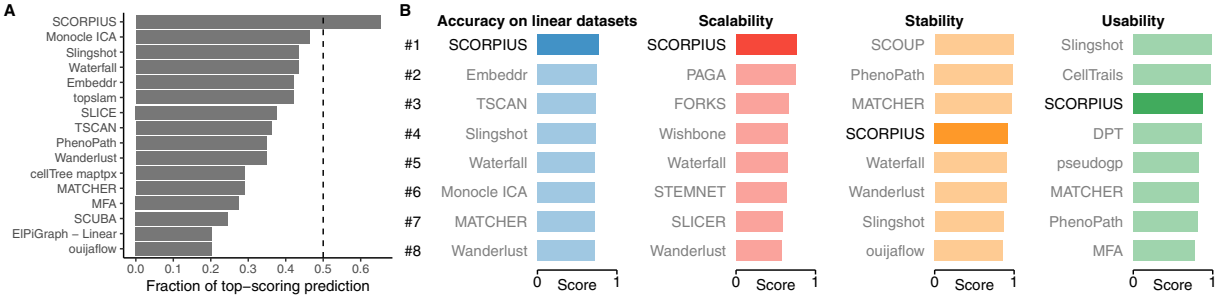



**Figure 4.2: SCORPIUS outperforms 44 TI methods in inferring linear trajectories. A:** It is the only method to produce top-scoring predictions on more than 50% of linear datasets. A predicted model is considered top-scoring if its accuracy is larger than 95% of the maximum accuracy obtained by any 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.

Remove kmeans initialisation as it does not improve performance? See Figure 4.3C.

### 4.2.2 Functional modules in dendritic cell development

Applying SCORPIUS to a dataset of dendritic cell (DC) progenitors [53] 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 [114]. The dataset contains 57 Monocyte and Dendritic cell Progenitors (MDPs), 95 Common Dendritic cell Progenitors (CDPs) and 96 Pre-Dendritic Cells (PreDCs). SCOR-

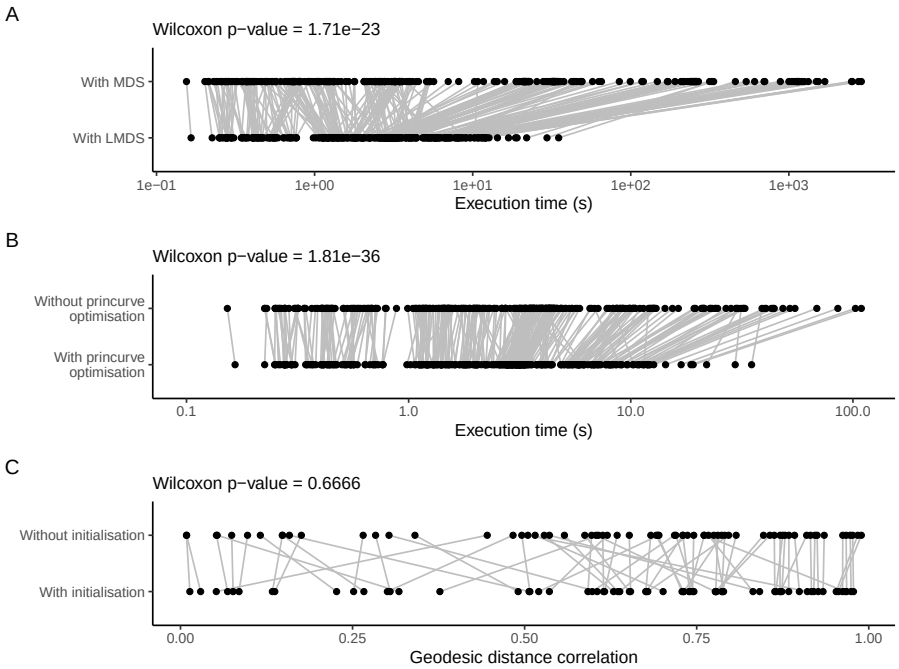



**Figure 4.3: Comparison of the effect of different optimisations made in SCORPIUS.** The optimisations in the dimensionality reduction (**A**) and principal curves (**B**) steps significantly reduce the execution times of SCORPIUS. Initialising the principal curve does not yield significant improvements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

PIUS 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 [116]. 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 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, [117, 118]), and presenting antigens is one of the core responsibilities of a DC (module 6, [119]).

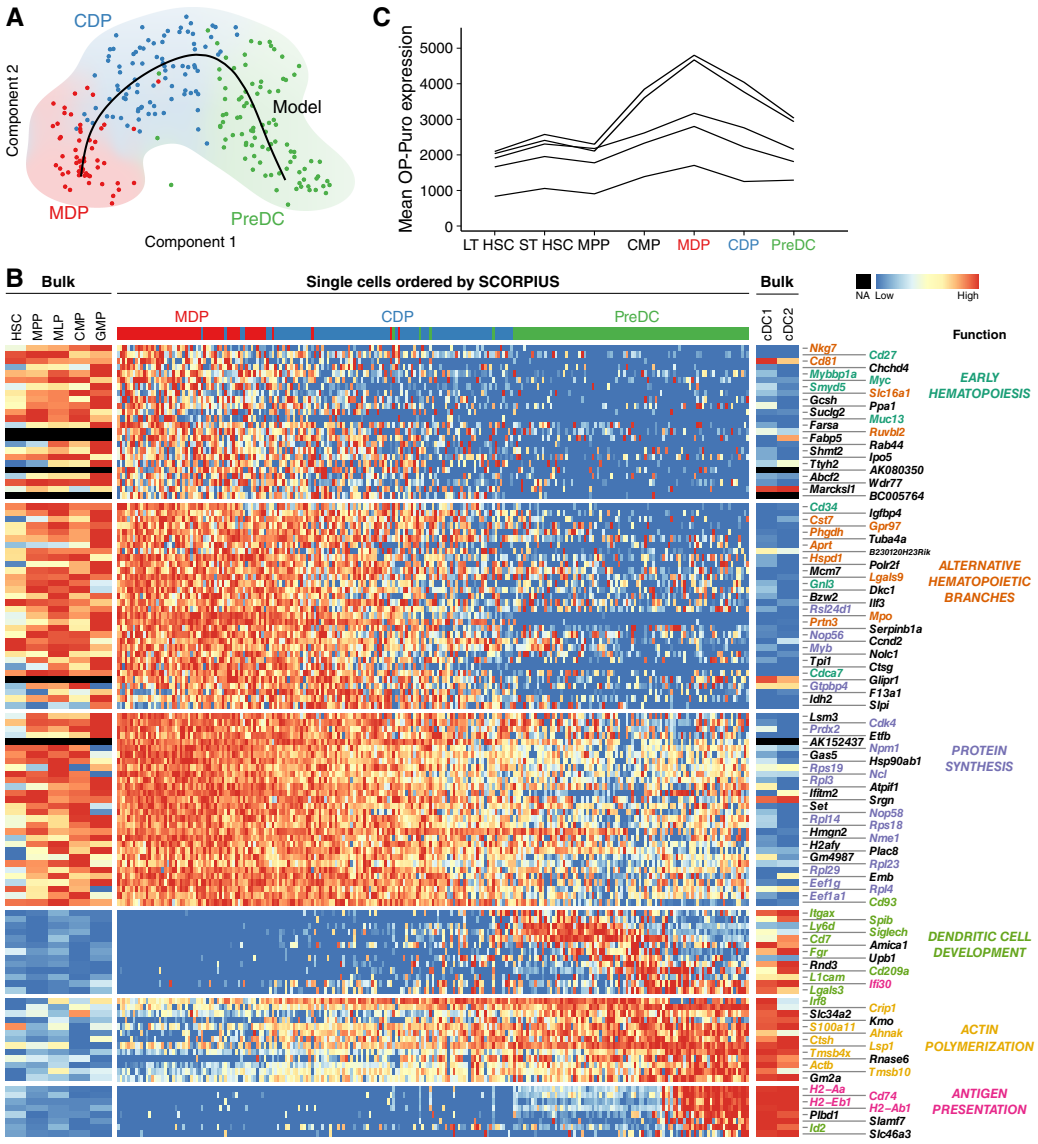



**Figure 4.4: SCORPIUS sheds new, data-driven light on dendritic cell development.** **A:** SCORPIUS creates an accurate model for DC development from scRNA-seq data. **B:** These genes are clustered into six gene modules. Each module is responsible for different aspects of DC development. Bulk microarray expression for up- and downstream stages of dendritic cell development [115] 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 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.

Something about what you can now do with SCORPIUS?

## 4.4 Methods

Todo: Add sections for each of the figures in the results section

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) [111], Principal Curves [112], and Random Forests [113]. 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).

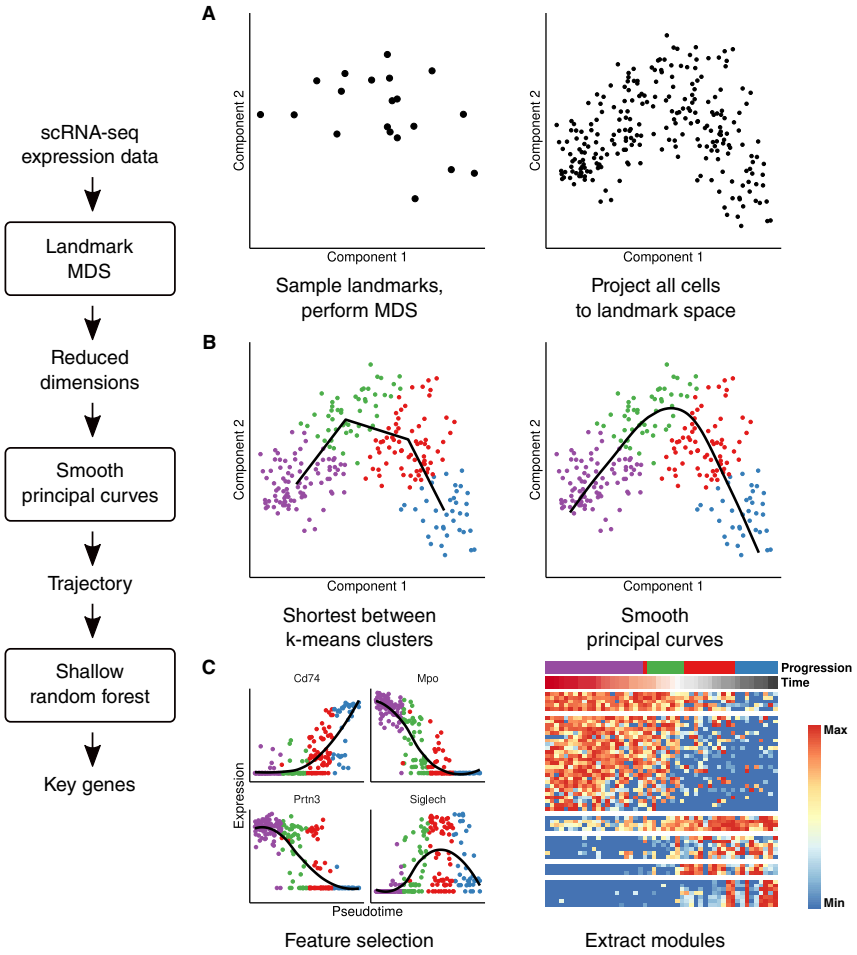



**Figure 4.5: SCORPIUS consists of three main steps.** **A:** Landmark MDS reduces the dimensionality of a small set of randomly sampled cells called landmarks. Afterwards, all other cells are projected to the landmark space. **B:** Smooth principal curves are used to pseudotemporally order the cells. 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 [120] is an extension of classical Torgerson MDS [111]. Classical MDS requires the computation of the distance matrix between all pairs of cells, which does 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 [112]. 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 [121] 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 [106].

### 4.4.5 Datasets and benchmark results

The results of the benchmark analysis were obtained directly from the benchmark of 45 TI methods [38], available at github.com/dynverse/dynbenchmark_results. The accuracy, scalability, stability and usability metrics are described by Saelens et al. [38]. All datasets were obtained from a repository of single-cell omics datasets containing a trajectory hosted on Zenodo record number 1443566 [122].

### 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/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 $70\,\mu\mathrm{m}$ 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).

$7 \times 10^6$ 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, $2.5\,\mu\mathrm{M}$ 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 on 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.

# CHAPTER 5

## bred: Inferring single cell regulatory networks

**Abstract:** Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Adapted from:
**Cannoodt, R.**, Saelens, W., and Saeys, Y. Inferring Single Cell Regulatory Networks. *Journal* vol, issue (2019), page–page. doi.

5

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

# CHAPTER 6

# Discussion

# 6

Note: This text was written very quickly and is only meant as a temporary coathanger.

This chapter reflects on the impact of this work on the field, referring back to the research objectives that were defined in Chapter 1.

dyngen is a succes, as we created a simulator of single cell data that is already used to evaluate trajectory[38], trajectory alignment[123] and single cell network inference[124].

dynbenchmark was meant to guide users to better practices for applying trajectory inference methods, and guide developers to adopt better practices for developing accurate and robust TI methods. The first part we succeeded in, as the guidelines are commonly disseminated in manuscripts [125, 126], courses [127, 128], and slides shown during keynote caffeine refuelling sessions [129]. However, we do not observe an increase in developers of TI methods performing quantitative benchmarks. We will discuss this in more detail in Chapter 7.

dynbenchmark was a benchmark of TI methods developed over the course of 4 years involving writing software to run 45 different error-prone TI methods with a common interface, downloading and processing hundreds of single cell datasets, developing novel metrics for comparing ground truth and predicted trajectories, writing a simulator for synthetic single cell data. Along the way, we have learned a lot about how to benchmark computational methods. We share our vision and guidelines on benchmarking computational methods in Chapter 8.

Something about SCORPIUS.

Something about bred.

Something about dyno.

Something about gng?

# CHAPTER 7

# Self-assessment in trajectory inference

7

Many articles introducing novel trajectory inference (TI) tools lack quantitative assessment of the accuracy of the method. Instead, they rely on anecdotal evidence to demonstrate their added value. A brief review of 75 articles reveals that only about 37% contain a self-assessment (Figure 7.1A,B). Peer-reviewed articles fared even worse, self-assessing in only 34% of cases (n=55), whereas articles first published as a pre-print self-assess in 43% of cases (n=39).

The number of datasets used and methods compared against is also below expectations (Figure 7.1C,D). Only three TI articles feature a comparison of at least 5 methods using 5 datasets or more [130, 131, 132].

While self-assessments are universally biased in favour of the authors[97] (intentially or not), it is dangerous and unusual to publish a computational tool without quantitatively demonstrating its performance compared to state-of-the-art methods. Indeed, a recent comparison of 45 TI methods demonstrated that most methods perform worse than a few baseline methods constructed by combining simple off-the-shelf algorithms such as PCA, $k$-means and MST[38].

In this perspective, we hypothesise that low self-assessment rates are primarily caused by a lack of a standardised problem definition, readily available benchmarking datasets, and suitable metrics. We elaborate on these causal reasons, and provide viable solutions for performing TI benchmarks more easily.

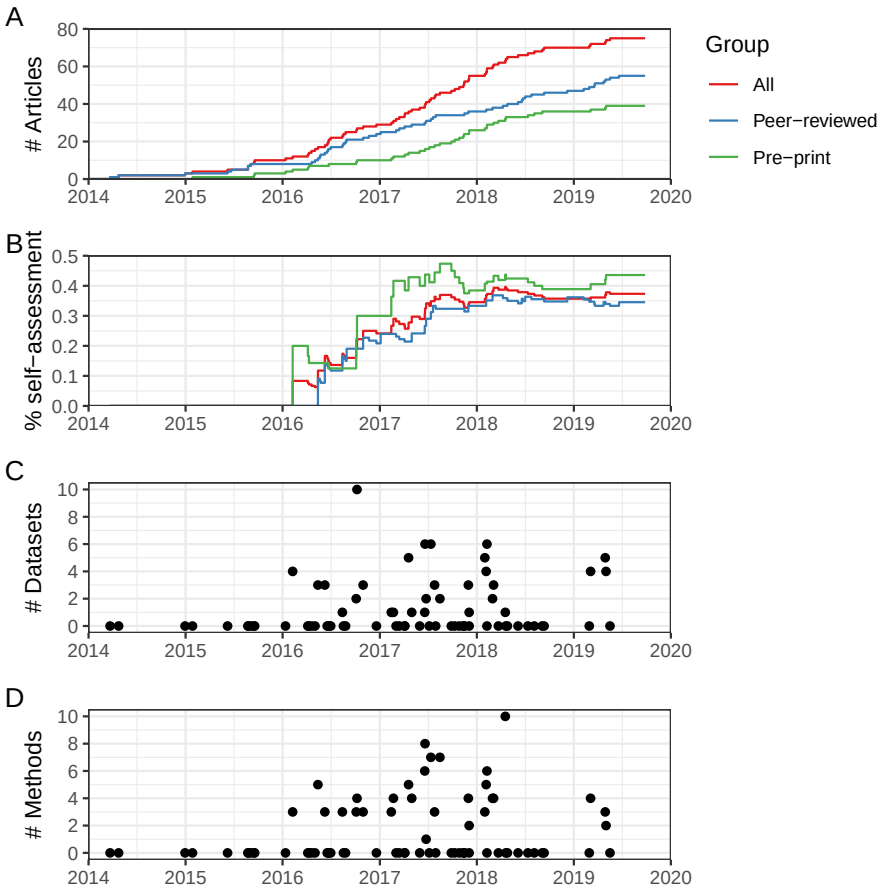



**Figure 7.1: Less than half of all TI articles perform quantitative self-assessment. A:** Since 2016, the number of TI articles has been increasing rapidly. Note that TI methods with both a pre-print and a peer-reviewed article only count once in the overall tally. **B:** Less than 50% of articles feature a self-assessment. Peer-reviewed articles self-assess only in 34% of cases. **C:** The number of datasets used in each benchmark is low. **D:** The number of methods (inclusing itself) evaluated is low.

## 7.1 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 (Figure 7.2A). For example, a method might infer linear or cyclic trajectories, or predict the probability of a cell ending up in one of several end states.

As a result, it becomes harder to discover similar methods to compare against, as certain articles might only show up with search terms such as "pseudotemporal ordering", "lineage trees" or "fate bias". For the discoverability of a new TI method, it is therefore essential to use the term "trajectory inference", or at least list it as one of the keywords.

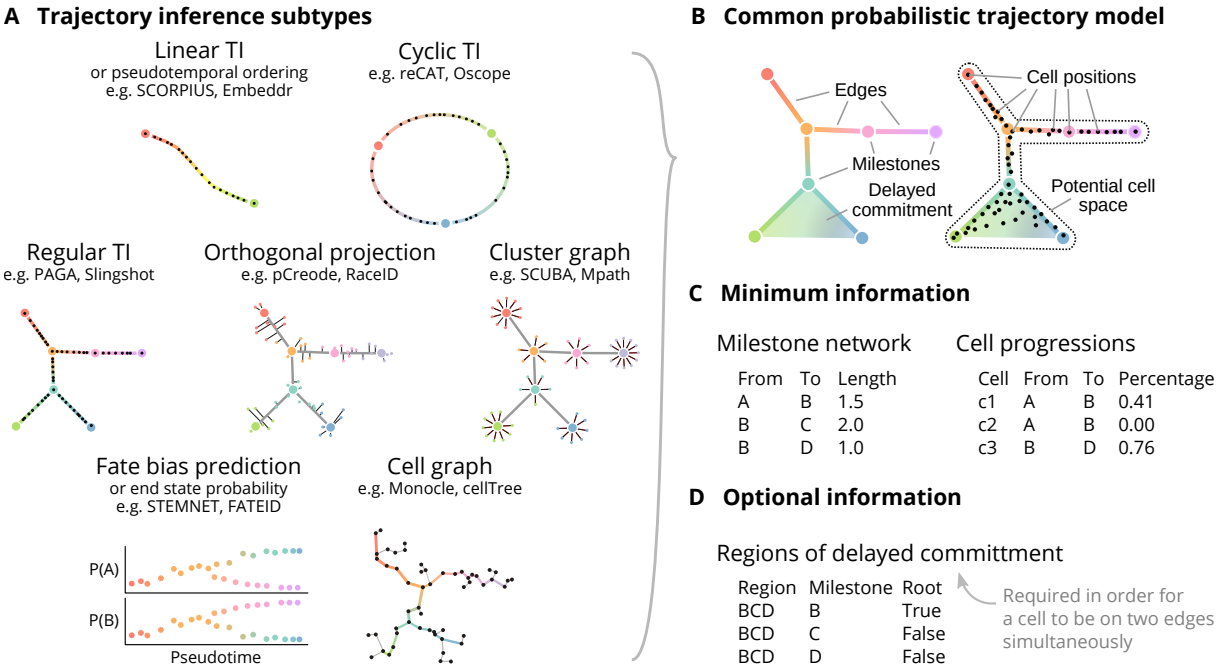



**Figure 7.2: Different forms of trajectory inference. A:** All TI methods can be categorised in one of seven subtypes in terms of its produced output [38]. **B:** Each of these can be translated into a common format, allowing easier comparison of multiple trajectories. **C:** The minimum information required to describe a trajectory in this way is the milestone network – representing transitions between cellular states – and the cell progressions – representing the positions of cells along the transitions. **D:** Optionally, regions of delayed commitment can be defined. A region of delayed commitment contain multiple transitions starting from the same milestone. This allows a TI method to assign probabilities on how likely a cell is part of one of these transitions.

A more significant and harder to solve problem is that the data formats produced by different methods varies greatly. This makes visualising and comparing multiple trajectories difficult, as different output types cannot be compared directly. The most commonly used and general is one where cells are positioned along a set of edges connecting milestones ("Regular TI", Figure 7.2A). By adding an extension to regular TI to allow for cells to be part of three or more cellular states, thereby a cell to delay its commitment toward a particular end state (Figure 7.2B).

By adding this extension, all TI subtypes can easily be converted into the common format. Implementations of these conversions can be found in `dynwrap`[**dyno**]. Using this standardised format allows developing reusable software for visualising and comparing trajectories from different TI methods.

In practice, this format consists of two data structures: the milestone network specify transition between cell states, and the cell progressions specify how far along each cell has progressed along a transition (Figure 7.2C). In addition, regions of delayed commitment need to be specified, if any (Figure 7.2D).

examples? elaborate? citation?

7

## 7.2 Benchmarking datasets

Another hurdle in benchmarking trajectory inference methods is collecting datasets to benchmark against. Before 2018, there were only a handful of datasets containing complex trajectories (Figure 7.3).

When real data is scarce, synthetic data is often used to evaluate computational methods, either standalone (n=5) or to complement real data (n=7). Most synthetic data is generated by the authors themselves (n=8), whereas some reuse datasets generated by others (n=3) or use one of the readily available simulators (n=2). To avoid introducing self-assessment bias in a benchmark, it is recommended to use readily available simulators if they fit the requirements. Examples are dyntoy [38], dyngen [**dyngen**], splatter [34], and PROSSTT [36].

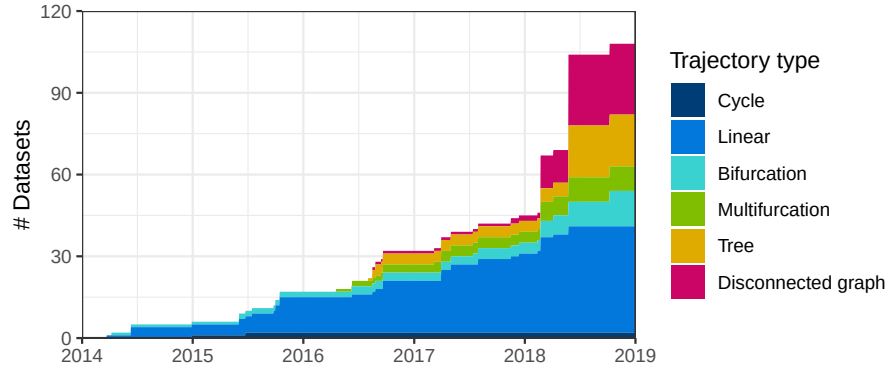



**Figure 7.3: An overview collection of real TI benchmarking datasets in function of their publication date and the topology of the trajectory.** These datasets are readily available on Zenodo[122].

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 to evaluate 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 evaluation 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.

Several authors use mainly real datasets to evaluate their method, though only few use more than four datasets (n=7). By now, already hundreds of suitable real datasets are available from GEO and ArrayExpress (Figure 7.3). Downloading and pre-processing them requires a significant time investment, but by processing the datasets once and storing them in a single repository they can be reused for multiple purposes.

Readers are welcome to reuse (and extend) the 110 real and 229 synthetic datasets used in our comparison of TI methods. The datasets are hosted on Zenodo[122] and the scripts to process them on GitHub[1]. Note that the ground truths of the datasets are represented using the common data structures format in the previous section.

[1] github.com/dynverse/dynbenchmark/tree/master/scripts/01-datasets

## 7.3 Multiple metrics

The largest problem, by far, seems to be a lack of standardised metrics to evaluate TI methods. None of the benchmarks use a metric directly aimed at comparing the topology or pairwise orderings of two trajectories.

Instead, most benchmarks (n=26) employ an ordering metric (e.g. pearson correlation) the pseudotime[2] of the predicted trajectory to ground truth information such as the cell type or time of sampling. Several benchmarks (n=5) use a clustering metric (e.g. ARI) to compare a cell's transition assignment or milestone assignment to the ground truth cell type. In four cases, multiple executions were compared to evaluate the robustness of the predictions. In one particular instance, the pseudotime correlation metric was adapted to be suitable for comparing rooted trees[67]. In another, an internal measure is used to quantify the smoothness of gene expression along the pseudotime [133].

While each of these metrics provide some insight the correctness of cell orderings or cell clusterings, they do not evaluate the correctness of the prediction's topology. Using only a pseudotime-based metric to evaluate non-linear trajectories is particularly risky, as it

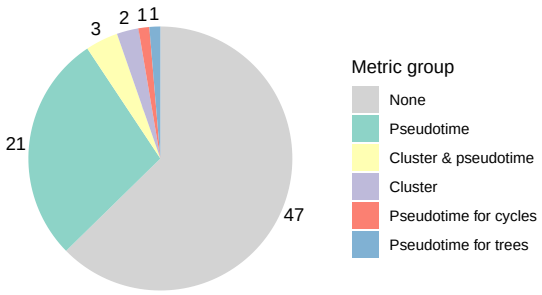



**Figure 7.4**

## 7.4 Further guidelines

[2]The pseudotime value of is calculated by computing its distance from a pre-defined or inferred start cell.

# CHAPTER 8

## Essential guidelines for computational method benchmarking

**Abstract:** 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 benchmark 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. Here, we summarize key practical guidelines and recommendations for performing high-quality benchmarking analyses, based on our experiences in computational biology.

Adapted from:
Weber, L. M., Saelens W., **Cannoodt, R.**, Soneson, C., Hapfelmeier, A., Gardner, P. P., Boulesteix, A., Saeys, Y., and Robinson, M. D.. Essential guidelines for computational method benchmarking. *Genome Biology* 20, 125 (2019). doi:10.1186/s13059-019-1738-8.

## 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 [20]. 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 [134, 135].

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 [134, 136, 89, 137, 138, 97, 93]; surveys of design practices followed by existing benchmarks [138]; the importance of neutral benchmarking studies [135]; principles for the design of real-data benchmarking studies [139, 140] and simulation studies [141]; the incorporation of meta-analysis techniques into benchmarking [142, 143, 144, 145]; the organization and role of community challenges [146, 147]; and discussions on benchmarking design for specific types of methods [148, 149]. More generally, benchmarking may be viewed as a form of meta-research [150].

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.

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.

### 8.1.1 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. [151, 152, 153, 154, 155]); (ii) neutral studies performed to systematically compare methods for a certain analysis, either conducted directly by an independent group (e.g. [38, 156, 157, 95, 158, 159, 160, 161, 162, 163, 164, 165]) or in collaboration with method authors (e.g. [166]); or (iii) those organized in the form of a community challenge, such as those from the DREAM [167, 168, 169, 43, 170], FlowCAP [171, 172], CASP [173, 174], CAMI [175], Assemblathon [176, 177], MAQC/SEQC [178, 179,

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.

180], and GA4GH [181] 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 benchmark should be approximately equally familiar with all included methods, reflecting typical usage of the methods by independent researchers [135]. 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 [139].

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 [151]).

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.

### 8.1.2 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 [38] or Table 1 in [158]). 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.

### 8.1.3 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 [182]). 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 [157]; for DNA methylation, correlation patterns among neighbouring CpG sites should be investigated [183]; for comparing mapping algorithms, error profiles of the sequencing platforms should be considered [184]. 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.

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 [152]), or against a current widely accepted method or gold standard (e.g. manual gating to define cell populations in high-dimensional cytometry [158, 171], or fluorescence *in situ* hybridization to validate absolute copy number predictions [137]). 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 [185, 94]. 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 [186] in RNA-sequencing experiments (e.g. [180, 187]), (ii) large-scale validation of gene expression measurements by quantitative polymerase chain reaction (e.g. [180]), (iii) using genes located on sex chromosomes as a proxy for silencing of DNA methylation status (e.g. [155, 188]), (iv) using fluorescence-activated cell sorting to sort cells into known sub-populations prior to single-cell RNA-sequencing (e.g. [157, 27, 189]), or (v) mixing different cell lines to create pseudo-cells [190]. 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 [151, 191, 192]). 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 [171] 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. [193]). 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.

### 8.1.4 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 [38, 194, 195]. In addition, for [38], 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. [95]). 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 [89, 196].

Software versions can also influence results, especially if updates include major changes to methodology (e.g. [197]). 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.

### 8.1.5 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 [156, 158] and by others [160, 171, 198]). 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 [199, 200]. 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 [151, 157]. 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 [201]); examples from our work include [151, 152, 154, 202].

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. [168, 203] used correlation, while [174] 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.

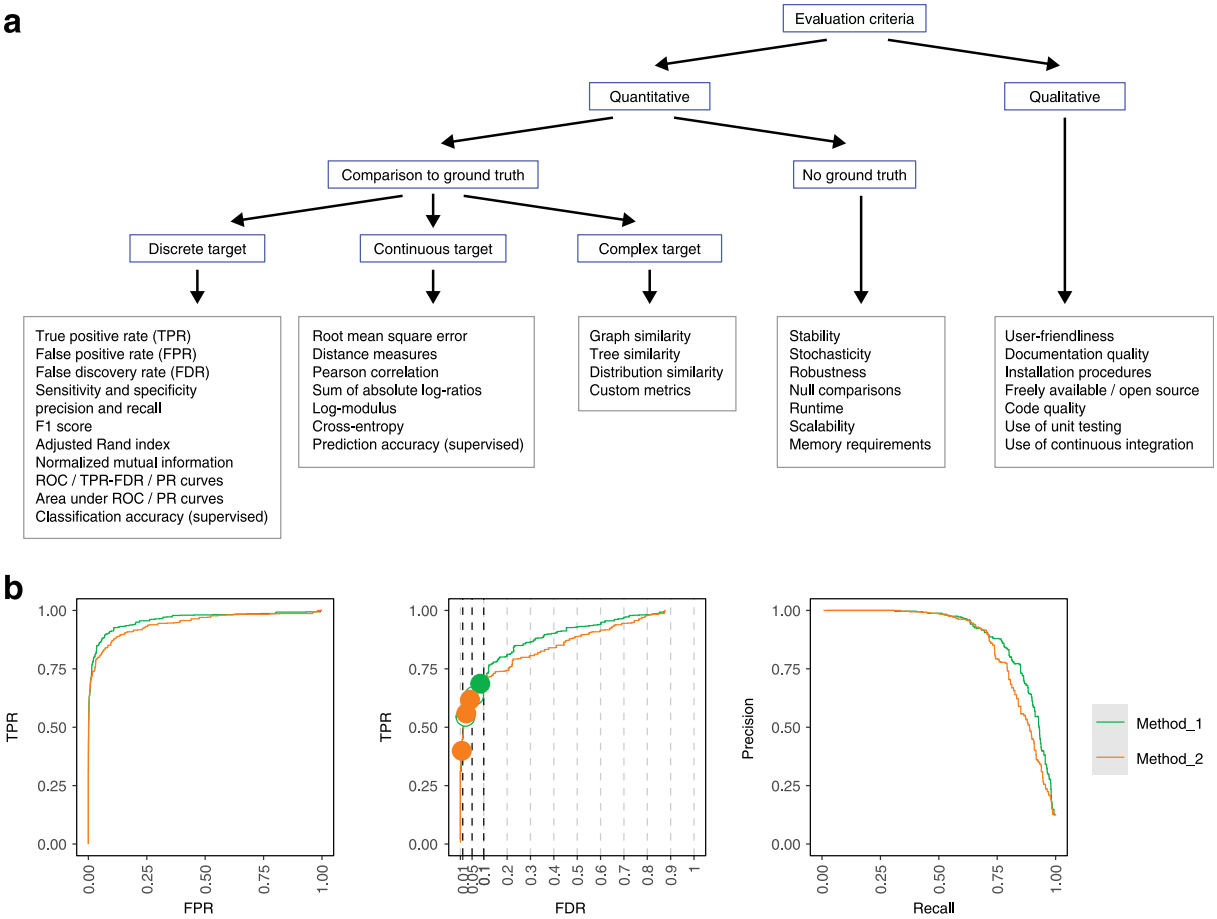



**Figure 8.2: Summary and examples of performance metrics.** **A:** Schematic overview of classes of frequently used performance metrics, including examples (*boxes outlined in gray*). **B:** Examples of popular visualizations of quantitative performance metrics for classification methods, using reference 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 [201]. *FDR* false discovery rate, *FPR* false positive rate, *PR* precision-recall, *ROC* receiver operating characteristic, *TPR* true positive rate

we defined new metrics for topologies of developmental trajectories in [38]). When designing custom metrics, it is important to assess their reliability across a range of prediction values (e.g. [204, 205]). For some metrics, it may also be useful to assess uncertainty, for example, via confidence intervals. In the context of supervised learning, classification or prediction accuracy can be evaluated by cross-validation, bootstrapping, or on a separate test dataset (e.g. [142, 172]). 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 [206]).

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 [157, 158]). 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. [38, 157, 158]). Fall-back solutions such as imputation may be considered in this case [207], 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. [151, 154, 155]). 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 [158], 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 [38] and Figure 5 in [157] from our work; and Figure 2 in [166] and Figure 6 in [159] from other authors). Identifying methods that consistently under-perform can also be useful, to allow readers to avoid these.

### 8.1.6 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 [158] and meta-genome analysis tools [203] 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 [156, 157]. 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 [208, 209]. 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.

### 8.1.7 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 [38] we used default parameters for all methods, while in [158] 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.

### 8.1.8 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 [38] and Figure 5 in [157] from our own work; as well as Figure 6 in [159]). Summary tables are also useful as a reference (e.g. [158, 171]). 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 [38]).

For extensive benchmarks, online resources enable readers to interactively explore the results (examples from our work include [38, 158], which allow users to filter metrics and datasets). Figure 3 displays an example of an interactive website from one of our benchmarks [38], 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 [203, 210]). Finally, at publication time, considering open access options will further broaden accessibility.

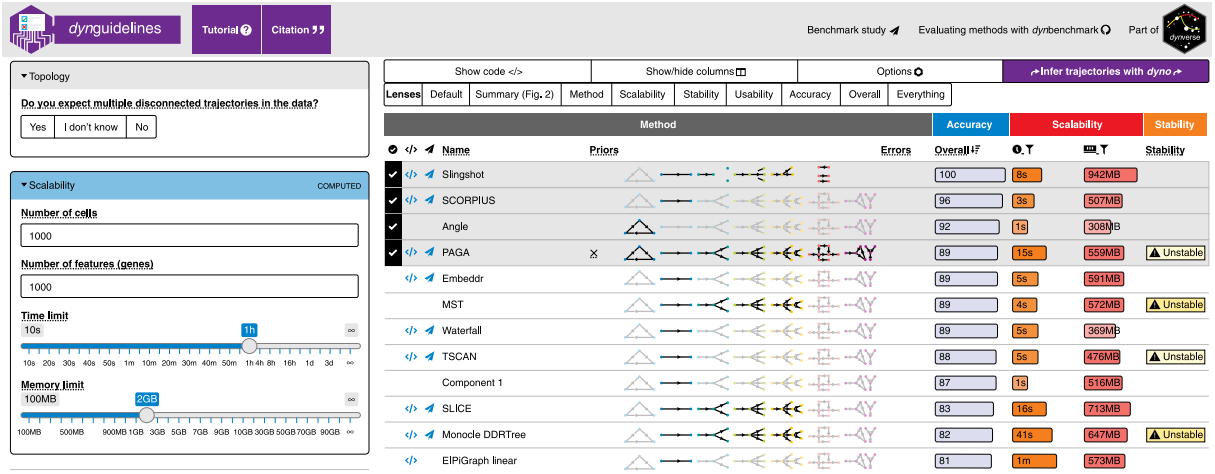



**Figure 8.3: Example of an interactive website allowing users to explore the results of one of our benchmarking studies [38].** This website was created using the Shiny framework in R.

### 8.1.9 Enabling future extensions

Since new methods are continually emerging [20], 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 [38, 156, 157, 95, 158]). In addition to raw data and code, it is useful to distribute pre-processed and/or results data (examples include [156, 157, 201] from our work and [198, 211, 212] 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. [162, 213, 214]). Continuous benchmarks, which are continually updated, are especially convenient (e.g. [215]), but may require significant additional effort.

### 8.1.10 Reproducible research best practices

Reproducibility of research findings has become an increasing concern in numerous areas of study [216]. In computational sciences, reproducibility of code and data analyses has been recognized as a useful minimum standard that enables other researchers to verify analyses [217]. Access to code and data has previously enabled method developers to uncover potential errors in published benchmarks due to suboptimal usage of methods [197, 218, 219]. Journal publication policies can play a crucial role in encouraging authors to follow these practices [220]; experience shows that statements that code and data are available on request are often insufficient [221]. 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 [222], 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 [158]).

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 [223], DataPackageR [224], workflowr [225], and Dynamic Statistical Comparisons [226]. 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 [227]. More general tools for managing computational workflows, including Snakemake [228], Make, Bioconda [229], 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 [38]). 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 [230], the Gene Expression Omnibus [231], and FlowRepository [232]), and more general data repositories (e.g. figshare, Dryad, Zenodo, Bioconductor ExperimentHub, and Mendeley Data). Customized resources (examples from our work include [157, 201]) 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 [208, 209].

An extensive literature exists on best practices for reproducible computational research (e.g. [233]). 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.

## 8.2 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 [139, 142, 143].

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

**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<br>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<br>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<br>Metrics that give over-optimistic estimates of performance<br>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<br>Subjectivity in relative weighting between multiple metrics<br>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<br>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 |

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.

# Samenvatting

# Summary

# List of Publications

# Bibliography


[1] Ingo Brigandt and Alan Love. "Reductionism in Biology". In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N Zalta. Spring 201. Published: $\backslash$url{https://plato.stanford.edu/archives/spr biology/}. Metaphysics Research Lab, Stanford University, 2017.

[2] Aviv Regev et al. "The Human Cell Atlas White Paper". In: (Oct. 2018).

[3] Human Cell Atlas consortium. *Human Cell Atlas Data Portal*. 2018.

[4] 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.

[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". en. In: *Molecular Biology of the Cell. 4th edition* (2002).

[7] Caitlin E. Cornell et al. "Prebiotic Amino Acids Bind to and Stabilize Prebiotic Fatty Acid Membranes". en. In: *Proceedings of the National Academy of Sciences* (Aug. 2019), p. 201900275. ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.1900275116.

[8] 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.

[9] Olga Kelemen et al. "Function of Alternative Splicing". In: *Gene* 514.1 (Feb. 2013), pp. 1–30. ISSN: 0378-1119. DOI: 10.1016/j.gene.2012.07.083.

[10] 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.

[11] 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.

[12] 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.

[13] Fuchou Tang et al. "mRNA-Seq Whole-Transcriptome Analysis of a Single Cell". en. In: *Nature Methods* 6.5 (May 2009), pp. 377–382. ISSN: 1548-7105. DOI: 10.1038/nmeth.1315.

[14] Jan Philipp Junker and Alexander van Oudenaarden. "Every Cell Is Special: Genome-Wide Studies Add a New Dimension to Single-Cell Biology". In: *Cell* 157.1 (Mar. 2014), pp. 8–11. ISSN: 0092-8674. DOI: 10.1016/j.cell.2014.02.010.

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

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

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

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

[19] Allon Wagner, Aviv Regev, and Nir Yosef. "Revealing the Vectors of Cellular Identity with Single-Cell Genomics". en. In: *Nature Biotechnology* 34.11 (Nov. 2016), pp. 1145–1160. ISSN: 1546-1696. DOI: 10.1038/nbt.3711.

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

[21] 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.

[22] 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.

[23] 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.

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

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

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] Daniel Marbach et al. "Revealing Strengths and Weaknesses of Methods for Gene Network Inference". In: *Proceedings of the {N}ational {A}cademy of {S}ciences* 107.14 (Apr. 2010), pp. 6286–6291. ISSN: 1091-6490. DOI: 10.1073/pnas.0913357107.

[31] 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.

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

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

[34] 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.

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

[36] 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.

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

[38] 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.

[39] 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.

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

[41] Daniel Zenklusen, Daniel R. Larson, and Robert H. Singer. "Single-RNA Counting Reveals Alternative Modes of Gene Expression in Yeast". eng. In: *Nature Structural & Molecular Biology* 15.12 (Dec. 2008), pp. 1263–1271. ISSN: 1545-9985. DOI: 10.1038/nsmb.1514.

[42] Lea Schuh et al. "Gene Networks with Transcriptional Bursting Recapitulate Rare Transient Coordinated Expression States in Cancer". In: *bioRxiv* (Jan. 2019), p. 704247. DOI: 10.1101/704247.

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

[44] 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.

[45] 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.

[46] 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.

[47] 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.

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

[49] 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.

[50] 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.

[51] 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.

[52] 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.

[53] 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.

[54] 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.

[55] 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.

[56] 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.

[57] Sean Davis et al. *Awesome Single Cell*. https://github.com/seandavi/awesome-single-cell. June 2018. DOI: 10.5281/zenodo.1294021.

[58] 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.

[59] 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.

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

[61] 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.

[62] 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.

[63] 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.

[64] 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.

[65] Hirotaka Matsumoto and Hisanori Kiryu. "{{SCOUP}}: A Probabilistic Model Based on the {{Ornstein}}$\backslash$ Process to Analyze Single-Cell Expression Data during Differentiation". In: *BMC Bioinformatics* 17 (June 2016), p. 232. ISSN: 1471-2105. DOI: 10.1186/s12859-016-1109-3.

[66] 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.

[67] 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.

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

[69] 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.

[70] 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.

[71] 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.

[72] 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.

[73] 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.

[74] 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.

[75] 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.

[76] 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.

[77] 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.

[78] 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.

[79] 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.

[80] 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.

[81] Jeff Lee. *Rpackages: {{R}} Package Development - the {{Leek}} Group Way!* Dec. 2017.

[82] Hadley Wickham. *R Packages: Organize, Test, Document, and Share Your Code*. en. "O'Reilly Media, Inc.", Mar. 2015. ISBN: 978-1-4919-1056-6.

[83] 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.

[84] 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.

[85] Mehran Karimzadeh and Michael M Hoffman. "Top Considerations for Creating Bioinformatics Software Documentation". In: *Briefings in Bioinformatics* (). DOI: 10.1093/bib/bbw134.

[86] Alex Anderson. *Writing Great Scientific Code*. Oct. 2016.

[87] 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.

[88] Vincent Driessen. *A Successful {{Git}} Branching Model*. Jan. 2010.

[89] 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.

[90] Jean Francois Puget. *Green Dice Are Loaded (Welcome to p-Hacking)*. Mar. 2016.

[91] 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.

[92] 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.

[93] 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.

[94] 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.

[95] 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.

[96] 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.

[97] 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.

[98] Anthony Gitter. *Single-Cell RNA-Seq Pseudotime Estimation Algorithm*. https://github.com/agitter/single-cell-pseudotime. June 2018. DOI: 10.5281/zenodo.1297423.

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

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

[101] Daniel Marbach et al. "Tissue-Specific Regulatory Circuits Reveal Variable Modular Perturbations across Complex Diseases". In: *Nature Methods* 13.4 (Apr. 2016), p. 366. ISSN: 1548-7105. DOI: 10.1038/nmeth.3799.

[102] 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.

[103] Paolo Tormene et al. "Matching Incomplete Time Series with Dynamic Time Warping: An Algorithm and an Application to Post-Stroke Rehabilitation". In: *Artificial Intelligence in Medicine* 45.1 (Jan. 2009), pp. 11–34. ISSN: 0933-3657. DOI: 10.1016/j.artmed.2008.11.007.

[104] 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.

[105] 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.

[106] 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.

[107] 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.

[108] 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.

[109] 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.

[110] 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.

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

[112] 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.

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

[114] 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.

[115] 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.

[116] 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.

[117] 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.

[118] 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.

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

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

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

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

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

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

[125] Atefeh Lafzi et al. "Tutorial: Guidelines for the Experimental Design of Single-Cell RNA Sequencing Studies". In: *Nature Protocols* 13.12 (Dec. 2018), pp. 2742–2757. ISSN: 1750-2799. DOI: 10.1038/s41596-018-0073-y.

[126] 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 2019), e8746. ISSN: 1744-4292. DOI: 10.15252/msb.20188746.

[127] Vladimir Kiselev et al. *Analysis of Single Cell RNA-Seq Data*. English. Cambridge, UK, May 2019.

[128] Liesbet Martens and Niels Vandamme. *Analysis of Single Cell RNA-Seq Data from 10x Genomics*. English. Ghent, Aug. 2019.

[129] Martin Hemberg. *Coffee Break during "Analysis of Single Cell RNA-Seq Data 23-24 May 2019" Workshop*. May 2019.

[130] Mayank Sharma et al. "FORKS: Finding Orderings Robustly Using K-Means and Steiner Trees". In: *bioRxiv* (June 2017), p. 132811. DOI: 10.1101/132811.

[131] Jing Guo and Jie Zheng. "HopLand: Single-Cell Pseudotime Recovery Using Continuous Hopfield Network-Based Modeling of Waddington's Epigenetic Landscape". In: *Bioinformatics* 33.14 (July 2017), pp. i102–i109. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btx232.

[132] 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.

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

[134] Anne-Laure Boulesteix et al. "On the Necessity and Design of Studies Comparing Statistical Methods". en. In: *Biometrical Journal* 60.1 (2018), pp. 216–218. ISSN: 1521-4036. DOI: 10.1002/bimj.201700129.

[135] Anne-Laure Boulesteix, Sabine Lauer, and Manuel J. A. Eugster. "A Plea for Neutral Comparison Studies in Computational Sciences". eng. In: *PloS One* 8.4 (2013), e61562. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0061562.

[136] Bjoern Peters et al. "Putting Benchmarks in Their Rightful Place: The Heart of Computational Biology". eng. In: *PLoS computational biology* 14.11 (Nov. 2018), e1006494. ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1006494.

[137] Siyuan Zheng. "Benchmarking: Contexts and Details Matter". eng. In: *Genome Biology* 18.1 (May 2017), p. 129. ISSN: 1474-760X. DOI: 10.1186/s13059-017-1258-3.

[138] Serghei Mangul et al. "Systematic Benchmarking of Omics Computational Tools". eng. In: *Nature Communications* 10.1 (Mar. 2019), p. 1393. ISSN: 2041-1723. DOI: 10.1038/s41467-019-09406-4.

[139] 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". eng. In: *BMC medical research methodology* 17.1 (Sept. 2017), p. 138. ISSN: 1471-2288. DOI: 10.1186/s12874-017-0417-2.

[140] Anne-Laure Boulesteix et al. "A Statistical Framework for Hypothesis Testing in Real Data Comparison Studies". In: *The American Statistician* 69.3 (July 2015), pp. 201–212. ISSN: 0003-1305. DOI: 10.1080/00031305.2015.1005128.

[141] Tim P. Morris, Ian R. White, and Michael J. Crowther. "Using Simulation Studies to Evaluate Statistical Methods". eng. In: *Statistics in Medicine* 38.11 (May 2019), pp. 2074–2102. ISSN: 1097-0258. DOI: 10.1002/sim.8086.

[142] Paul P. Gardner et al. "Identifying Accurate Metagenome and Amplicon Software via a Meta-Analysis of Sequence to Taxonomy Benchmarking Studies". eng. In: *PeerJ* 7 (2019), e6160. ISSN: 2167-8359. DOI: 10.7717/peerj.6160.

[143] Paul P. Gardner et al. "A Meta-Analysis of Bioinformatics Software Benchmarks Reveals That Publication-Bias Unduly Influences Software Accuracy". en. In: *bioRxiv* (Jan. 2017), p. 092205. DOI: 10.1101/092205.

[144] Evangelos Evangelou and John P. A. Ioannidis. "Meta-Analysis Methods for Genome-Wide Association Studies and Beyond". eng. In: *Nature Reviews. Genetics* 14.6 (June 2013), pp. 379–389. ISSN: 1471-0064. DOI: 10.1038/nrg3472.

[145] Fangxin Hong and Rainer Breitling. "A Comparison of Meta-Analysis Methods for Detecting Differentially Expressed Genes in Microarray Experiments". eng. In: *Bioinformatics (Oxford, England)* 24.3 (Feb. 2008), pp. 374–382. ISSN: 1367-4811. DOI: 10.1093/bioinformatics/btm620.

[146] Paul C. Boutros et al. "Toward Better Benchmarking: Challenge-Based Methods Assessment in Cancer Genomics". eng. In: *Genome Biology* 15.9 (Sept. 2014), p. 462. ISSN: 1474-760X. DOI: 10.1186/s13059-014-0462-7.

[147] Iddo Friedberg et al. "Ten Simple Rules for a Community Computational Challenge". eng. In: *PLoS computational biology* 11.4 (Apr. 2015), e1004150. ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1004150.

[148] Iven Van Mechelen et al. "Benchmarking in Cluster Analysis: A White Paper". In: *arXiv:1809.10496 [stat]* (Sept. 2018). arXiv: 1809.10496 [stat].

[149] 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". en. In: *F1000Research* 7 (Dec. 2018), p. 459. ISSN: 2046-1402. DOI: 10.12688/f1000research.14509.2.

[150] John P. A. Ioannidis. "Meta-Research: Why Research on Research Matters". eng. In: *PLoS biology* 16.3 (Mar. 2018), e2005468. ISSN: 1545-7885. DOI: 10.1371/journal.pbio.2005468.

[151] Lukas M. Weber et al. "Diffcyt: Differential Discovery in High-Dimensional Cytometry via High-Resolution Clustering". eng. In: *Communications Biology* 2 (2019), p. 183. ISSN: 2399-3642. DOI: 10.1038/s42003-019-0415-5.

[152] Malgorzata Nowicka and Mark D. Robinson. "DRIMSeq: A Dirichlet-Multinomial Framework for Multivariate Count Outcomes in Genomics". eng. In: *F1000Research* 5 (2016), p. 1356. ISSN: 2046-1402. DOI: 10.12688/f1000research.8900.2.

[153] Jacob H. Levine et al. "Data-Driven Phenotypic Dissection of AML Reveals Progenitor-like Cells That Correlate with Prognosis". eng. In: *Cell* 162.1 (July 2015), pp. 184–197. ISSN: 1097-4172. DOI: 10.1016/j.cell.2015.05.047.

[154] Xiaobei Zhou, Helen Lindsay, and Mark D. Robinson. "Robustly Detecting Differential Expression in RNA Sequencing Data Using Observation Weights". eng. In: *Nucleic Acids Research* 42.11 (June 2014), e91. ISSN: 1362-4962. DOI: 10.1093/nar/gku310.

[155] Charity W. Law et al. "Voom: Precision Weights Unlock Linear Model Analysis Tools for RNA-Seq Read Counts". eng. In: *Genome Biology* 15.2 (Feb. 2014), R29. ISSN: 1474-760X. DOI: 10.1186/gb-2014-15-2-r29.

[156] Angelo Duò, Mark D. Robinson, and Charlotte Soneson. "A Systematic Performance Evaluation of Clustering Methods for Single-Cell RNA-Seq Data". eng. In: *F1000Research* 7 (2018), p. 1141. ISSN: 2046-1402. DOI: 10.12688/f1000research.15666.2.

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

[158] Lukas M. Weber and Mark D. Robinson. "Comparison of Clustering Methods for High-Dimensional Single-Cell Flow and Mass Cytometry Data". en. In: *Cytometry Part A* 89.12 (2016), pp. 1084–1096. ISSN: 1552-4930. DOI: 10.1002/cyto.a.23030.

[159] Keegan Korthauer et al. "A Practical Guide to Methods Controlling False Discoveries in Computational Biology". eng. In: *Genome Biology* 20.1 (Apr. 2019), p. 118. ISSN: 1474-760X. DOI: 10.1186/s13059-019-1716-1.

[160] Saskia Freytag et al. "Comparison of Clustering Tools in R for Medium-Sized 10x Genomics Single-Cell RNA-Sequencing Data". eng. In: *F1000Research* 7 (2018), p. 1297. ISSN: 2046-1402. DOI: 10.12688/f1000research.15809.2.

[161] Giacomo Baruzzo et al. "Simulation-Based Comprehensive Benchmarking of RNA-Seq Aligners". eng. In: *Nature Methods* 14.2 (Feb. 2017), pp. 135–139. ISSN: 1548-7105. DOI: 10.1038/nmeth.4106.

[162] Alexander Kanitz et al. "Comparative Assessment of Methods for the Computational Inference of Transcript Isoform Abundance from RNA-Seq Data". eng. In: *Genome Biology* 16 (July 2015), p. 150. ISSN: 1474-760X. DOI: 10.1186/s13059-015-0702-5.

[163] Charlotte Soneson and Mauro Delorenzi. "A Comparison of Methods for Differential Expression Analysis of RNA-Seq Data". eng. In: *BMC bioinformatics* 14 (Mar. 2013), p. 91. ISSN: 1471-2105. DOI: 10.1186/1471-2105-14-91.

[164] Franck Rapaport et al. "Comprehensive Evaluation of Differential Gene Expression Analysis Methods for RNA-Seq Data". eng. In: *Genome Biology* 14.9 (2013), R95. ISSN: 1474-760X. DOI: 10.1186/gb-2013-14-9-r95.

[165] Marie-Agnès Dillies et al. "A Comprehensive Evaluation of Normalization Methods for Illumina High-Throughput RNA Sequencing Data Analysis". eng. In: *Briefings in Bioinformatics* 14.6 (Nov. 2013), pp. 671–683. ISSN: 1477-4054. DOI: 10.1093/bib/bbs046.

[166] Daniel Sage et al. "Quantitative Evaluation of Software Packages for Single-Molecule Localization Microscopy". eng. In: *Nature Methods* 12.8 (Aug. 2015), pp. 717–724. ISSN: 1548-7105. DOI: 10.1038/nmeth.3442.

[167] Matthew T. Weirauch et al. "Evaluation of Methods for Modeling Transcription Factor Sequence Specificity". eng. In: *Nature Biotechnology* 31.2 (Feb. 2013), pp. 126–134. ISSN: 1546-1696. DOI: 10.1038/nbt.2486.

[168] James C. Costello et al. "A Community Effort to Assess and Improve Drug Sensitivity Prediction Algorithms". eng. In: *Nature Biotechnology* 32.12 (Dec. 2014), pp. 1202–1212. ISSN: 1546-1696. DOI: 10.1038/nbt.2877.

[169] Robert Küffner et al. "Crowdsourced Analysis of Clinical Trial Data to Predict Amyotrophic Lateral Sclerosis Progression". eng. In: *Nature Biotechnology* 33.1 (Jan. 2015), pp. 51–57. ISSN: 1546-1696. DOI: 10.1038/nbt.3051.

[170] Steven M. Hill et al. "Inferring Causal Molecular Networks: Empirical Assessment through a Community-Based Effort". eng. In: *Nature Methods* 13.4 (Apr. 2016), pp. 310–318. ISSN: 1548-7105. DOI: 10.1038/nmeth.3773.

[171] 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.

[172] Nima Aghaeepour et al. "A Benchmark for Evaluation of Algorithms for Identification of Cellular Correlates of Clinical Outcomes". en. In: *Cytometry Part A* 89.1 (2016), pp. 16–21. ISSN: 1552-4930. DOI: 10.1002/cyto.a.22732.

[173] John Moult et al. "Critical Assessment of Methods of Protein Structure Prediction (CASP)-Round XII". eng. In: *Proteins* 86 Suppl 1 (Mar. 2018), pp. 7–15. ISSN: 1097-0134. DOI: 10.1002/prot.25415.

[174] John Moult et al. "Critical Assessment of Methods of Protein Structure Prediction: Progress and New Directions in Round XI". eng. In: *Proteins* 84 Suppl 1 (Sept. 2016), pp. 4–14. ISSN: 1097-0134. DOI: 10.1002/prot.25064.

[175] Alexander Sczyrba et al. "Critical Assessment of Metagenome Interpretation-a Benchmark of Metagenomics Software". eng. In: *Nature Methods* 14.11 (Nov. 2017), pp. 1063–1071. ISSN: 1548-7105. DOI: 10.1038/nmeth.4458.

[176] Dent Earl et al. "Assemblathon 1: A Competitive Assessment of de Novo Short Read Assembly Methods". eng. In: *Genome Research* 21.12 (Dec. 2011), pp. 2224–2241. ISSN: 1549-5469. DOI: 10.1101/gr.126599.111.

[177] Keith R. Bradnam et al. "Assemblathon 2: Evaluating de Novo Methods of Genome Assembly in Three Vertebrate Species". en. In: *GigaScience* 2.1 (Dec. 2013). DOI: 10.1186/2047-217X-2-10.

[178] MAQC Consortium et al. "The MicroArray Quality Control (MAQC) Project Shows Inter- and Intraplatform Reproducibility of Gene Expression Measurements". eng. In: *Nature Biotechnology* 24.9 (Sept. 2006), pp. 1151–1161. ISSN: 1087-0156. DOI: 10.1038/nbt1239.

[179] Leming Shi et al. "The MicroArray Quality Control (MAQC)-II Study of Common Practices for the Development and Validation of Microarray-Based Predictive Models". eng. In: *Nature Biotechnology* 28.8 (Aug. 2010), pp. 827–838. ISSN: 1546-1696. DOI: 10.1038/nbt.1665.

[180] SEQC/MAQC-III Consortium. "A Comprehensive Assessment of RNA-Seq Accuracy, Reproducibility and Information Content by the Sequencing Quality Control Consortium". eng. In: *Nature Biotechnology* 32.9 (Sept. 2014), pp. 903–914. ISSN: 1546-1696. DOI: 10.1038/nbt.2957.

[181] Peter Krusche et al. "Best Practices for Benchmarking Germline Small-Variant Calls in Human Genomes". en. In: *Nature Biotechnology* 37.5 (May 2019), pp. 555–560. ISSN: 1546-1696. DOI: 10.1038/s41587-019-0054-x.

[182] Charlotte Soneson and Mark D. Robinson. "Towards Unified Quality Verification of Synthetic Count Data with countsimQC". en. In: *Bioinformatics* 34.4 (Feb. 2018), pp. 691–692. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btx631.

[183] Keegan Korthauer et al. "Detection and Accurate False Discovery Rate Control of Differentially Methylated Regions from Whole Genome Bisulfite Sequencing". eng. In: *Biostatistics (Oxford, England)* 20.3 (Jan. 2019), pp. 367–383. ISSN: 1468-4357. DOI: 10.1093/biostatistics/kxy007.

[184] Ségolène Caboche et al. "Comparison of Mapping Algorithms Used in High-Throughput Sequencing: Application to Ion Torrent Data". eng. In: *BMC genomics* 15 (Apr. 2014), p. 264. ISSN: 1471-2164. DOI: 10.1186/1471-2164-15-264.

[185] 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.

[186] Lichun Jiang et al. "Synthetic Spike-in Standards for RNA-Seq Experiments". eng. In: *Genome Research* 21.9 (Sept. 2011), pp. 1543–1551. ISSN: 1549-5469. DOI: 10.1101/gr.121095.111.

[187] Daniel R. Garalde et al. "Highly Parallel Direct RNA Sequencing on an Array of Nanopores". eng. In: *Nature Methods* 15.3 (Mar. 2018), pp. 201–206. ISSN: 1548-7105. DOI: 10.1038/nmeth.4577.

[188] Fang Fang et al. "Genomic Landscape of Human Allele-Specific DNA Methylation". eng. In: *Proceedings of the National Academy of Sciences of the United States of America* 109.19 (May 2012), pp. 7332–7337. ISSN: 1091-6490. DOI: 10.1073/pnas.1201310109.

[189] Grace X. Y. Zheng et al. "Massively Parallel Digital Transcriptional Profiling of Single Cells". eng. In: *Nature Communications* 8 (Jan. 2017), p. 14049. ISSN: 2041-1723. DOI: 10.1038/ncomms14049.

[190] Luyi Tian et al. "Benchmarking Single Cell RNA-Sequencing Analysis Pipelines Using Mixture Control Experiments". eng. In: *Nature Methods* 16.6 (June 2019), pp. 479–487. ISSN: 1548-7105. DOI: 10.1038/s41592-019-0425-8.

[191] Eirini Arvaniti and Manfred Claassen. "Sensitive Detection of Rare Disease-Associated Cell Subsets via Representation Learning". en. In: *Nature Communications* 8.1 (Apr. 2017), pp. 1–10. ISSN: 2041-1723. DOI: 10.1038/ncomms14825.

[192] Guillem Rigaill et al. "Synthetic Data Sets for the Identification of Key Ingredients for RNA-Seq Differential Analysis". eng. In: *Briefings in Bioinformatics* 19.1 (Jan. 2018), pp. 65–76. ISSN: 1477-4054. DOI: 10.1093/bib/bbw092.

[193] Benedikt Löwes et al. "The BRaliBase Dent-a Tale of Benchmark Design and Interpretation". eng. In: *Briefings in Bioinformatics* 18.2 (Jan. 2017), pp. 306–311. ISSN: 1477-4054. DOI: 10.1093/bib/bbw022.

[194] Raphael Couronné, Philipp Probst, and Anne-Laure Boulesteix. "Random Forest versus Logistic Regression: A Large-Scale Benchmark Experiment". In: *BMC Bioinformatics* 19.1 (July 2018), p. 270. ISSN: 1471-2105. DOI: 10.1186/s12859-018-2264-5.

[195] Jochen Schneider et al. "Mortality Risk for Acute Cholangitis (MAC): A Risk Prediction Model for in-Hospital Mortality in Patients with Acute Cholangitis". eng. In: *BMC gastroenterology* 16 (Feb. 2016), p. 15. ISSN: 1471-230X. DOI: 10.1186/s12876-016-0428-1.

[196] Qiwen Hu and Casey S. Greene. "Parameter Tuning Is a Key Part of Dimensionality Reduction via Deep Variational Autoencoders for Single Cell RNA Transcriptomics". eng. In: *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing* 24 (2019), pp. 362–373. ISSN: 2335-6936.

[197] Jorge Vaquero-Garcia, Scott Norton, and Yoseph Barash. "LeafCutter vs. MAJIQ and Comparing Software in the Fast Moving Field of Genomics". en. In: *bioRxiv* (Nov. 2018), p. 463927. DOI: 10.1101/463927.

[198] Christian Wiwie, Jan Baumbach, and Richard Röttger. "Comparing the Performance of Biomedical Clustering Methods". eng. In: *Nature Methods* 12.11 (Nov. 2015), pp. 1033–1038. ISSN: 1548-7105. DOI: 10.1038/nmeth.3583.

[199] Takaya Saito and Marc Rehmsmeier. "The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets". eng. In: *PloS One* 10.3 (2015), e0118432. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0118432.

[200] David M. W. Powers. "Visualization of Tradeoff in Evaluation: From Precision-Recall & PN to LIFT, ROC & BIRD". In: *arXiv:1505.00401 [cs, stat]* (May 2015). arXiv: 1505.00401 [cs, stat].

[201] Charlotte Soneson and Mark D. Robinson. "iCOBRA: Open, Reproducible, Standardized and Live Method Benchmarking". eng. In: *Nature Methods* 13.4 (Apr. 2016), p. 283. ISSN: 1548-7105. DOI: 10.1038/nmeth.3805.

[202] 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. 2016). ISSN: 2046-1402. DOI: 10.12688/f1000research.7563.2.

[203] Stinus Lindgreen, Karen L. Adair, and Paul P. Gardner. "An Evaluation of the Accuracy and Speed of Metagenome Analysis Tools". eng. In: *Scientific Reports* 6 (Jan. 2016), p. 19233. ISSN: 2045-2322. DOI: 10.1038/srep19233.

[204] Alexey Gurevich et al. "QUAST: Quality Assessment Tool for Genome Assemblies". eng. In: *Bioinformatics (Oxford, England)* 29.8 (Apr. 2013), pp. 1072–1075. ISSN: 1367-4811. DOI: 10.1093/bioinformatics/btt086.

[205] Giuseppe Narzisi and Bud Mishra. "Comparing de Novo Genome Assembly: The Long and Short of It". eng. In: *PloS One* 6.4 (Apr. 2011), e19175. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0019175.

[206] Jacob Schreiber et al. "A Pitfall for Machine Learning Methods Aiming to Predict across Cell Types". en. In: *bioRxiv* (Jan. 2019), p. 512434. DOI: 10.1101/512434.

[207] Bernd Bischl, Julia Schiffner, and Claus Weihs. "Benchmarking Local Classification Methods". en. In: *Computational Statistics* 28.6 (Dec. 2013), pp. 2599–2619. ISSN: 1613-9658. DOI: 10.1007/s00180-013-0420-y.

[208] Serghei Mangul et al. "Improving the Usability and Archival Stability of Bioinformatics Software". eng. In: *Genome Biology* 20.1 (Feb. 2019), p. 47. ISSN: 1474-760X. DOI: 10.1186/s13059-019-1649-8.

[209] Serghei Mangul et al. "Challenges and Recommendations to Improve the Installability and Archival Stability of Omics Computational Tools". eng. In: *PLoS biology* 17.6 (June 2019), e3000333. ISSN: 1545-7885. DOI: 10.1371/journal.pbio.3000333.

[210] 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". eng. In: *Genome Research* 17.1 (Jan. 2007), pp. 117–125. ISSN: 1088-9051. DOI: 10.1101/gr.5890907.

[211] Nicholas A. Bokulich et al. "Mockrobiota: A Public Resource for Microbiome Bioinformatics Benchmarking". eng. In: *mSystems* 1.5 (Sept. 2016). ISSN: 2379-5077. DOI: 10.1128/mSystems.00062-16.

[212] Shane Ó Conchúir et al. "A Web Resource for Standardized Benchmark Datasets, Metrics, and Rosetta Protocols for Macromolecular Modeling and Design". en. In: *PLOS ONE* 10.9 (Sept. 2015), e0130433. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0130433.

[213] Leslie M. Cope et al. "A Benchmark for Affymetrix GeneChip Expression Measures". eng. In: *Bioinformatics (Oxford, England)* 20.3 (Feb. 2004), pp. 323–331. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btg410.

[214] Rafael A. Irizarry, Zhijin Wu, and Harris A. Jaffee. "Comparison of Affymetrix GeneChip Expression Measures". eng. In: *Bioinformatics (Oxford, England)* 22.7 (Apr. 2006), pp. 789–794. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btk046.

[215] Michael Barton. *Nucleotides · Genome Assembler Benchmarking*. http://nucleotid.es. Oct. 2014.

[216] John P. A. Ioannidis. "Why Most Published Research Findings Are False". eng. In: *PLoS medicine* 2.8 (Aug. 2005), e124. ISSN: 1549-1676. DOI: 10.1371/journal.pmed.0020124.

[217] Roger D. Peng. "Reproducible Research in Computational Science". eng. In: *Science (New York, N.Y.)* 334.6060 (Dec. 2011), pp. 1226–1227. ISSN: 1095-9203. DOI: 10.1126/science.1213847.

[218] Xiaobei Zhou and Mark D. Robinson. "Do Count-Based Differential Expression Methods Perform Poorly When Genes Are Expressed in Only One Condition?" eng. In: *Genome Biology* 16 (Oct. 2015), p. 222. ISSN: 1474-760X. DOI: 10.1186/s13059-015-0781-3.

[219] Xiaobei Zhou, Alicia Oshlack, and Mark D. Robinson. "miRNA-Seq Normalization Comparisons Need Improvement". eng. In: *RNA (New York, N.Y.)* 19.6 (June 2013), pp. 733–734. ISSN: 1469-9001. DOI: 10.1261/rna.037895.112.

[220] Benjamin Hofner, Matthias Schmid, and Lutz Edler. "Reproducible Research in Statistics: A Review and Guidelines for the Biometrical Journal". eng. In: *Biometrical Journal. Biometrische Zeitschrift* 58.2 (Mar. 2016), pp. 416–427. ISSN: 1521-4036. DOI: 10.1002/bimj.201500156.

[221] Anne-Laure Boulesteix et al. "Making Complex Prediction Rules Applicable for Readers: Current Practice in Random Forest Literature and Recommendations". eng. In: *Biometrical Journal. Biometrische Zeitschrift* 61.5 (Sept. 2019), pp. 1314–1328. ISSN: 1521-4036. DOI: 10.1002/bimj.201700243.

[222] Enis Afgan et al. "The Galaxy Platform for Accessible, Reproducible and Collaborative Biomedical Analyses: 2018 Update". eng. In: *Nucleic Acids Research* 46.W1 (Feb. 2018), W537–W544. ISSN: 1362-4962. DOI: 10.1093/nar/gky379.

[223] Patrick K. Kimes and Alejandro Reyes. "Reproducible and Replicable Comparisons Using SummarizedBenchmark". eng. In: *Bioinformatics (Oxford, England)* 35.1 (Jan. 2019), pp. 137–139. ISSN: 1367-4811. DOI: 10.1093/bioinformatics/bty627.

[224] Greg Finak et al. "DataPackageR: Reproducible Data Preprocessing, Standardization and Sharing Using R/Bioconductor for Collaborative Data Analysis". eng. In: *Gates Open Research* 2 (July 2018), p. 31. ISSN: 2572-4754. DOI: 10.12688/gatesopenres.12832.2.

[225] John Blischak, Peter Carbonetto, and Matthew Stephens. *Workflowr: A Framework for Reproducible and Collaborative Data Science*. R package version 1.4.0. 2019.

[226] G Wang, Matthew Stephens, and Peter Carbonetto. *DSC: Dynamic Statistical Comparisons*. https://stephenslab.github.io/dsc-wiki/index.html. Apr. 2016.

[227] 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.

[228] Johannes Köster and Sven Rahmann. "Snakemake–a Scalable Bioinformatics Workflow Engine". eng. In: *Bioinformatics (Oxford, England)* 28.19 (Oct. 2012), pp. 2520–2522. ISSN: 1367-4811. DOI: 10.1093/bioinformatics/bts480.

[229] Björn Grüning et al. "Bioconda: Sustainable and Comprehensive Software Distribution for the Life Sciences". eng. In: *Nature Methods* 15.7 (July 2018), pp. 475–476. ISSN: 1548-7105. DOI: 10.1038/s41592-018-0046-7.

[230] Nikolay Kolesnikov et al. "ArrayExpress Update–Simplifying Data Submissions". eng. In: *Nucleic Acids Research* 43.Database issue (Jan. 2015), pp. D1113–1116. ISSN: 1362-4962. DOI: 10.1093/nar/gku1057.

[231] Tanya Barrett et al. "NCBI GEO: Archive for Functional Genomics Data Sets–Update". eng. In: *Nucleic Acids Research* 41.Database issue (Jan. 2013), pp. D991–995. ISSN: 1362-4962. DOI: 10.1093/nar/gks1193.

[232] Josef Spidlen et al. "FlowRepository: A Resource of Annotated Flow Cytometry Datasets Associated with Peer-Reviewed Publications". en. In: *Cytometry Part A* 81A.9 (2012), pp. 727–731. ISSN: 1552-4930. DOI: 10.1002/cyto.a.22106.

[233] Geir Kjetil Sandve et al. "Ten Simple Rules for Reproducible Computational Research". eng. In: *PLoS computational biology* 9.10 (Oct. 2013), e1003285. ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1003285.