NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCE
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATION

POSTGRADUATE PROGRAM
"DATA SCIENCE AND INFORMATION TECHNOLOGIES"
SPECIALIZATION
“BIOINFORMATICS – BIOMEDICAL DATA SCIENCE”

MASTER THESIS

Flexible single-cell RNAseq data analysis pipelines using MLscAN

George A. Koliopanos

Supervisor: **Elias S. Manolakos,** Professor, Department of Informatics and Telecommunication, National and Kapodistrian University of Athens

ATHENS

DECEMBER 2021

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ
ΔΙΑΤΜΗΜΑΤΙΚΟ ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
"ΕΠΙΣΤΗΜΗ ΔΕΔΟΜΕΝΩΝ ΚΑΙ ΤΕΧΝΟΛΟΓΙΕΣ ΠΛΗΡΟΦΟΡΙΑΣ"
ΕΙΔΙΚΕΥΣΗ
“ΒΙΟΠΛΗΡΟΦΟΡΙΚΗ – ΕΠΙΣΤΗΜΗ ΒΙΟΙΑΤΡΙΚΩΝ ΔΕΔΟΜΕΝΩΝ”

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ευέλικτες ροές ανάλυσης δεδομένων single-cell RNAseq με χρήση του MLscAN

Γεώργιος Α. Κολιοπάνος

Επιβλέπων: **Ηλίας Σ. Μανωλάκος,** Καθηγητής, Τμήμα Πληροφορικής και Τηλεπικοινωνιών, Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών

ΑΘΗΝΑ

ΔΕΚΕΜΒΡΙΟΣ 2021

MASTER THESIS

Flexible single-cell RNAseq data analysis pipelines using MLscAN

George A. Koliopanos
SRN: DS2.18.0007

SUPERVISOR: **Elias S. Manolakos,** Professor, Department of Informatics and Telecommunication, National and Kapodistrian University of Athens

EXAMINATION COMMITEE: **Elias S. Manolakos,** Professor , Department of Informatics and Telecommunication, National and Kapodistrian University of Athens
Ema Anastasiadou, Investigator - Assistant Professor Level, Biomedical Research Foundation of the Academy of Athens (BRFAA)
Dimitris Konstantopoulos, Postdoctoral Researcher, Biomedical Sciences Research Center "Alexander Fleming"

December 2021

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ευέλικτες ροές ανάλυσης δεδομένων single-cell RNAseq με χρήση του MLscAN

Γεώργιος Α. Κολιοπάνος
Α.Μ.: DS2.18.0007

ΕΠΙΒΛΕΠΩΝ: **Ηλίας Σ. Μανωλάκος,** Καθηγητής, Τμήμα Πληροφορικής και Τηλεπικοινωνιών, Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών

ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΤΡΟΠΗ:
Ηλίας Σ. Μανωλάκος, Καθηγητής, Τμήμα Πληροφορικής και Τηλεπικοινωνιών, Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών
Έμα Αναστασιάδου, Ερευνητής Γ', Ίδρυμα Ιατροβιολογικών Ερευνών Ακαδημίας Αθηνών
Δημήτρης Κωνσταντόπουλος, Μεταδιδακτορικός Ερευνητής, Ερευνητικό Κέντρο Βιοϊατρικών Επιστημών "Αλέξανδρος Φλέμινγκ"

Δεκέμβριος 2021

ABSTRACT

The single-cell RNA sequencing technology (scRNA-seq) was introduced to overcome its predecessor's, bulk RNAseq, low resolutions limitations. By providing gene expression profiles at the level of individual cells, scRNA-seq enables us to detect rare cell subpopulations, offering unique insights into fundamental cell interaction mechanisms in developmental and cancer biology. Many specialized data analysis tools have emerged to extract information from large and noisy scRNA-seq datasets. They aim to reconstruct a dataset's "epigenetic landscape" by discerning cell states and/or inferring trajectory networks. However, very few support an unbiased exploration of the large model space for capturing that landscape based on probabilistic machine learning.

MLscAN (Machine Learning for Single-Cell ANalytics) is a set of methods and a corresponding R-package developed by our group employing unsupervised machine learning single-cell data analysis based on Gaussian Mixture Models. Without any prior knowledge, by using only a preprocessed expression matrix of a scRNA-seq dataset, MLscAN can discover cell-states and infer state transitions using a probabilistic approach. A distinct feature of the MLscAN pipeline is that it partitions state transitions into consecutive phases (micro-states), identifies the "key-genes" governing the transition, and reconstructs Gene Regulatory Networks for every micro-state. MLscAN was initially built with the "naïve users" (with limited expertise in computational biology or R programming) in mind providing an automated end-to-end pipeline and extensive visualization for interpreting the results of every stage. However, it has gradually evolved to allow advanced users to customize a run invoke alternative processing methods and import results from other tools in nearly every step of the MLscAN analysis.

The main objective of this graduate thesis was to enhance MLscAN's versatility and flexibility by improving the integration of external results into the computational pipeline. The second objective was to develop methods to isolate and analyze separately "mixed states" that may emerge from GMM. These states have large variance and may encapsulate many small yet potentially significant cell subpopulations that may contribute interesting hypotheses into how the landscape of states may be structured if properly handled. Finally, particular emphasis was placed on demonstrating the MLscAN capabilities and versatility using representative and instructive use cases based on non-trivial real-world datasets.

SUBJECT AREA: single-cell RNA-seq data analysis, bioinformatics, unsupervised machine learning, probabilistic modeling

KEYWORDS: single-cells, RNA sequencing, state transitions, epigenetic landscape, trajectory inference, gene regulatory networks, R package

ΠΕΡΙΛΗΨΗ

Οι τεχνολογίες single-cell RNA- sequencing (scRNA-seq) εισήχθηκαν για να μπορέσουν να ξεπεραστούν οι περιορισμοί που δημιουργούσε η προγενέστερη τεχνολογία bulk RNA-seq. Παρέχοντας μας ενα γονιδιακό προφίλ έκφρασης σε επίπεδο single-cell, το scRNA-seq μας δίνει τη δυνατότητα να ανιχνεύουμε σπάνιους κυτταρικούς υποπληθυσμούς, προσφέροντας σημαντικές γνώσεις για τους θεμελιώδεις μηχανισμούς αλληλεπίδρασης των κυττάρων στην αναπτυξιακή βιολογία και την έρευνα για τον καρκίνο. Πολλά εξειδικευμένα εργαλεία ανάλυσης δεδομένων έχουν αναπτυχθεί για την εξαγωγή πληροφοριών από μεγάλα και θορυβώδη δεδομένα scRNA-seq.Τα εν λόγω πακέτα στοχεύουν στην ανακατασκευή ενός «επιγενετικού τοπίου» διακρίνοντας καταστάσεις κυττάρων ενώ ενα μέρος αυτών εξάγει και τροχιές μεταξύ των καταστάσεων. Ωστόσο, πολύ λίγα πακέτα παρέχουν μια αμερόληπτη εξερεύνηση του μεγάλου χώρου των μοντέλων για την αποτύπωση αυτού του τοπίου με βάση την πιθανοτική μηχανική μάθηση.

Το MLscAN (Machine Learning for Single-Cell ANAlytics) είναι ένα σύνολο μεθόδων που αναπτύχθηκε από την ομάδα μας στην γλώσσα προγραμματισμού R για ανάλυση δεδομένων single-cell χρήσημοποιώντας μη εποπτευόμενη μηχανική μάθηση με βάση τα Gaussian Mixture Models. Χωρίς καμία προηγούμενη γνώση, χρησιμοποιώντας μόνο προεπεξεργασμένα δεδομένα γονιδιακής έκφρασης ενός συνόλου δεδομένων scRNA-seq, το MLscAN μπορεί να ανακαλύψει κυτταρικές καταστάσεις και να εξάγει μεταβάσεις μεταξύ των καταστάσεων χρησιμοποιώντας μια πιθανοτική προσέγγιση. Ένα ξεχωριστό χαρακτηριστικό του MLscAN είναι ότι διαχωρίζει τις μεταβάσεις καταστάσεων σε διαδοχικές φάσεις (μικρο-κατάσταση), προσδιορίζει τα «γονίδια-κλειδιά» που διέπουν τη μετάβαση και αναδομεί τα ρυθμιστικά δίκτυα γονιδίων για κάθε μικροκατάσταση. Το MLscAN κατασκευάστηκε αρχικά για τον «αρχάριο χρήστη» (με περιορισμένη τεχνογνωσία στην υπολογιστική βιολογία ή τον προγραμματισμό σε R) παρέχοντας μια αυτοματοποιημένη αναλυση και εκτενή οπτικοποίηση για την ερμηνεία των αποτελεσμάτων κάθε σταδίου. Ωστόσο, έχει εξελιχθεί σταδιακά για να επιτρέπει στους προχωρημένους χρήστες να προσαρμόζουν την ανάλυση τους και να εισάγουν αποτελέσματα από άλλα εργαλεία σε σχεδόν κάθε βήμα της ανάλυσης του.

Ο κύριος στόχος αυτής της διπλωματικής εργασίας ήταν να ενισχύσει το MLscAN βελτιστοποιώντας την ενσωμάτωση εξωτερικών αποτελεσμάτων. Ο δεύτερος στόχος ήταν η ανάπτυξη μεθόδων για την απομόνωση και την ανάλυση «μεικτών καταστάσεων» που μπορεί να προκύψουν από το GMM. Αυτές οι καταστάσεις έχουν μεγάλη διακύμανση και είναι πιθανό να περικλείουν πολλούς μικρούς αλλά δυνητικά σημαντικούς κυτταρικούς υποπληθυσμούς που μπορεί να συνεισφέρουν σε ενδιαφέρουσες υποθέσεις στο πώς δομείται το «επιγενετικό τοπίο» εάν αντιμετωπιστούν σωστά. Τέλος, δόθηκε ιδιαίτερη έμφαση στην επίδειξη των δυνατοτήτων και της ευελιξίας του MLscAN χρησιμοποιώντας κατάλληλα αντιπροσωπευτικά και επιμορφωτικά παραδείγματα που βασίζονται σε πραγματικά σύνολα δεδομένων.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: ανάλυση δεδομένων scRNA-seq, βιοπληροφορική, μη εποπτευόμενη μηχανική μάθηση, πιθανοτικά μοντέλα

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: μεμονωμένα κύτταρα, αλληλούχιση RNA, μεταβάσεις κυτταρικής κατάστασης, επιγενετικό τοπίο κυτταρικών καταστάσεων, αναγνώριση κυτταρικών τροχιών, ρυθμιστικά δίκτυα γονιδίων, πακέτο R

ACKNOWLEDGEMENTS

I would like to thank my professor and thesis advisor Elias *S Manolakos*, for the great opportunity he gave me to be involved in this project, the watchfulness and availability he provided to discuss and brainstorm in all the steps of the project. I want also to acknowledge the effort he put on making sure this manuscript is of high quality.

I would also like to thank the examination committee, Dr. *Ema Anastasiadou* of BFRAA and Dr. *Dimitris Konstantopoulos* of the Fleming Institute, both experts in life sciences, for their comments that helped improve the final manuscript.

Special thanks go to my colleague *Arsenios Chatzigeorgiou*, with whom we were the main developers of the MLscAN team for the duration of this work, and our cooperation and mutual understanding has been exceptional.

I want to thank also every other member of the MLscAN team for their contributions: Dr. *Panos Tsakanikas* and Dr. *Dimitris Manatakis* for working with Prof. Manolakos to give shape to the initial idea of MLscAN, but also for their helpful insights in later development. Mrs *Efi Malesiou,* the initial developer of the MLscAN R package, whose work was the cornerstone of every later MLscAN improvement. Mrs *Rania Theologi* for proposing the need for *TrajecoryPath* function and Mr. *Nikolas Kalavros* for his help on vignette writing and the suggestions he provided.

This thesis would not be possible if we did not have access to GRNET Knossos resources for analyzing a vast number of sizable datasets.

Table of Contents

LIST OF FIGURES

1. INTRODUCTION

1.1 Thesis scope

Paul Ehrlich, in his Nobel lecture (1908) [1] while talking about the cell concept he quoted:

“For this concept is the axis around which the whole of the modern science of life revolves”

Even a hundred years later, it remains one of the main subjects of interest in biomedical research. A cell is widely considered to be the smallest structural and functional unit of an organism and it presents a great diversity of forms. This diversity can be explained by looking at the proteome of the different cells, which is a translation of the transcriptome.

RNA sequencing was invented to quantitatively assess the transcriptome of cells and understand how biological function is related to gene expression levels [2]. This revolutionary biotechnology can capture a snapshot of the gene expression of cells [3]. The first RNA sequencing technology was “bulk” RNA seq, in which a population of cells is analyzed to obtain an average gene expression profile of the entire cell mixture. However, average gene expression does not reveal the role of individual cell types in the mixture. This lack of resolution does not allow us to study developmental or cancer-related processes as we cannot observe the gradual differentiation and interaction of cells participating in an evolving biological process.

Single-cell RNA sequencing was introduced in the early 2010s to overcome these limitations [4]. Single-cell RNA seq (scRNA seq) isolates the cells in a mixture and provides gene expression profiles for each individual cell. It allows us to detect rare subpopulations and investigate intracellular state transitions. Single-cell RNA seq has already played a significant role in cancer research since it enabled us to study the interaction of tumor cells with different immune cells present in the same microenvironment and decipher their role [5]. Also, it has contributed dramatically to developmental biology by offering the ability to describe the continuity of cells maturing processes and their dynamics [6].

The emergence of scRNA-seq created the need for specialized analysis tools that can process effectively and efficiently large single-cell datasets. Since single-cell expression data are noisier than bulk RNA seq data [7], better quality control and preprocessing techniques have been developed. Using the preprocessed data, bioinformaticians, with the help of machine learning, developed techniques to discern cell states (clusters) and/or discover rare cell subpopulations. Next, the transition inference field emerged to investigate the interaction between the discovered cell clusters. Transition inference approaches try to capture the stages of a biological process by importing multiple single cells snapshots from the same sample/tissue. These snapshots are then used to create a pseudotemperal ordering of the cells based on their gene expression. In a sense, every cell can be thought of as a distinct step in a continuum of cell states.

Our project, called “Machine Learning for single-cell Analytics”, or MLscAN, came to address the unbiased single-cell data analysis and transition inference problems in a solid probabilistic framework. MLscAN provides a flexible end-to-end computational pipeline that performs all stages of the downstream data analysis in an unbiased manner, i.e., without requiring any prior knowledge (e.g., cell types, gene markers etc.). Starting with a preprocessed expression matrix, MLscAN can infer the landscape of cell states, state transitions, key-genes driving each transition, and Gene Regulatory Networks capturing how the key-genes regulate all phases of each state transition. MLscAN was first created with the “naïve" users (non-computational experts with limited R expertise) in mind and thus provides automatically extensive visualization of the results at every step of the data analysis. However, in its current incarnation, it also serves the advanced users (computational biologists, R experts), giving them all the flexibility they need to explore different models and how they capture their data [8], [9]).

The main distinguishing characteristics of the MLscAN pipeline are the following:

- Unsupervised/unbiased model selection. No prior knowledge is assumed (cell types, markers etc.). Using unsupervised machine learning methods to infer cell states.
- Probabilistic statistical machine learning methods are used instead of heuristics.
- Emphasis on reconstructing the epigenetic landscape of cell states and transitions suggested by a dataset using unsupervised machine learning methods
- Emphasis on inferring pairwise state-to-state transitions and identifying the “key-genes” that drive them
- Emphasis on inferring dynamic Gene Regulatory Networks (GRN) capturing how the key-genes interact during all phases of a state transition
- Effective visualization of the results at all stages of the pipeline, with no user effort.

1.2 Thesis Objectives

This graduate thesis was conducted in the context of the MLscAN project led by Prof. Elias S. Manolakos at the National and Kapodistrian University of Athens. Mrs Efi Malesiou, as part of her thesis, has created the first version of the MLscAN R package based on methods developed earlier in Matlab and reported in [8]. As the field moves

very fast and the package was not originally tested using large datasets, some aspects of MLscAN needed improvement and adding flexibility to its pipeline that was deemed necessary. Furthermore, while we analyzed large datasets, the “mixed states” issue (see chapter 3.5) was discovered and needed to be addressed. Finally, even though the package is very suitable for the ‘naïve’ users, we needed to demonstrate how a more experienced user can take full advantage of the capabilities of MLscAN by incorporating results from external pipelines at different stages of the analysis.

With all these points in mind, the main goals of this thesis work were the following:

Goal 1: Flexible pipelines creation: In its default pipeline, MLscAN uses Principal Components Analysis (PCA) for dimensionality reduction and Generalized Mixture Modeling (GMM) for inferring cell states (clustering). However, other methods for dimensionality reduction and/or cell clustering can be more suitable for a particular dataset. We wanted users to be able to implement any dimensionality reduction or clustering method they prefer outside the package and still be able to import their results into MLscAN to explore its unique capabilities for inferring trajectories, their key-genes, and GRNs. We have made the needed interventions and present here use cases showcasing this added flexibility (see chapter 3.2).

Moreover, we improved a function that can be used for dataset exploration in the model selection space before MLscAN runs are initiated, implemented two new functions for further analysis of MLscAN results (see chapter 3.3-3.4), and improved a lot of the visualization functions in the original version.

Goal 2: Mixed States handling: During clustering of big datasets, GMM may infer cell states with very large variances that quite often correspond to a mixture of small disparate cell clusters and/or outliers. It is very important to have tools to analyze those potentially interesting “mixed states” and handle them appropriately (break them into smaller states or remove them altogether) before moving to the stage of transitions inference. We have developed such methods and also show how they can be used (see chapter 3.5).

Goal 3: Demonstration through well-selected use cases: Finally, to guide the user on how to take full advantage of MLscAN, we used four well-known datasets to demonstrate four different use cases:

- General overview of the improved MLscAN pipeline exploring many of its visualization capabilities (see chapter 4)
- How to incorporate UMAP different dimensionality reduction results into an MLscAN run (see chapter 5)
- How to handle a Mixed State that encapsulates cell subpopulations of interest (see chapter 6.1)

- How to handle a Mixed State that includes mostly outliers (see chapter 6.2)

1.3 Thesis Organization

The rest of the thesis is organized as follows:

- In Chapter 2 we review the most commonly used single-cell data analysis packages and explain what MLscAN can offer compared to them.
- In Chapter 3 we describe the MLscAN pipeline extensively and how someone can integrate results from other pipelines into MLscAN analysis. In addition, we present some external functions that were developed in this work and explain the Mixed State problem.
- In Chapter 4 we present a general overview of MLscAN and demonstrate a lot of its visualization capabilities via a complete analysis example.
- In Chapter 5 we demonstrate how a user can implement dimensionality reduction using UMAP and include these results in downstream MLscAN analysis.
- In Chapter 6, we propose two different ways of dealing with a Mixed State.
- Finally, in Chapter 7, we summarize our contributions and suggest future improvements for the MLscAN project.

2. RELATED WORK

This chapter will present an overview of representative state-of-the-art single-cell RNAseq data analysis packages. We will focus on the main ingredients of their pipelines and how MLscAN distinguishes itself relative to them.

2.1 Single-cell data analysis

The need for higher resolution increased single-cell experiments' popularity during the past decade. This trend has led to the creation of different methodologies for analyzing scRNA-seq datasets. However, single-cell data are far noisier than bulk RNA se [7] q. Therefore, the pre-processing part of the analysis usually includes the following steps: Quality Control to omit poor-quality data [7] , correction of batch effects [10], normalization [11], and data imputation [12]. The main processing usually starts with projecting the preprocessed single-cell expression profiles to a lower-dimensional space using methods such as PCA [13], Diffusion Maps [14], t-SNE [15], UMAP [16]. The reduced dimension data are then used to cluster cells and possibly infer trajectories, gene markers, etc.

Transition Inference methods aim to derive a pseudo temporal ordering of the cells along a trajectory path using genes expression profiles information. In this ordering, the position of a cell in the trajectory path is assumed to represent its relative position in the course of an evolving biological process (developmental stage, differentiation, cancer progression, etc.) [17]. There are more than 70 tools that are trying to address the trajectory Inference problem using different approaches [17].

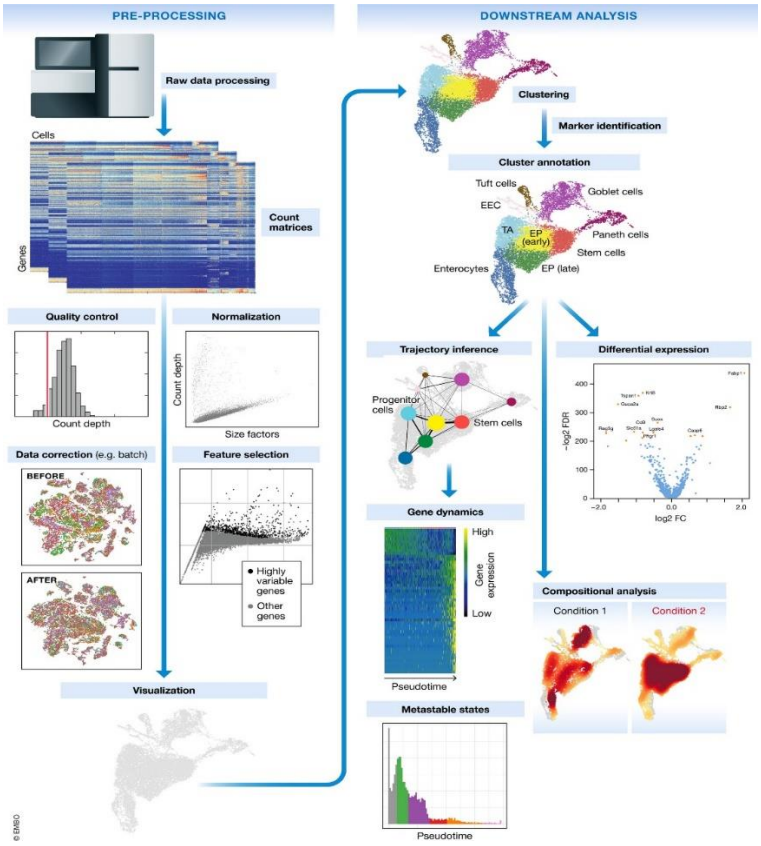

Figure 1 Example of a single-cell data analysis pipeline

2.1.1 Some popular pipelines and their features

There is a large number of packages following different approaches to address single-cell data analysis. Below we present the most popular ones and the main steps of their default computational pipelines.

Seurat pipeline [18]:

I. Normalization of data (preprocessing)

II. Feature selection

III. Dimensionality reduction PCA [13], ICA [19], t-SNE [20]

IV. Graph-based clustering

SC3 pipeline [21]:

I. Gene filtering

II. Distance matrix calculation using Euclidean, Pearson, and Spearman correlation

III. Dimensionality Reduction using PCA [13] on distance matrices

IV. K-means clustering [22]on the first distance matrix eigenvectors

V. Consensus clustering using Cluster-based Similarity Partitioning Algorithm (CSPA) [23]

Seurat and SC3 are not providing Transition Inference analysis.

Some well-known Transition Inference packages are Monocle [24], Slingshot [25], and PAGA/PAGA Tree [26].

Monocle pipeline:

I. Dimensionality reduction PCA [13] on preprocessed data

II. K-means clustering [22]

III. “Shifting” cells towards closest vertex

IV. If algorithm does not converge, repeat k-means clustering and shift cells towards closest vertex

V. Calculate pseudotime using distance from root node

Slingshot pipeline:

I. Dimensionality reduction using PCA(default: irlba [27] with 20 PCs)

II. Clustering technique: Partition Around Medoids [28]

III. Minimum spanning tree on clusters to determine the number and shape of lineages

IV. Obtain smooth representations of each lineage using simultaneous principal curves

V. Calculate pseudotime values using orthogonal projections onto the curves

PAGA/PAGA TREE:

I. Generate neighborhood graph of single-cells

II. Graph-partitioning, clustering

III. Cell ordering using distance from a root cell [random-walk-based distance]

SCANPY pipeline:

I. Preprocessing

II. Visualization using tSNE [20]

III. Dimensionality reduction using PCA [13]

IV. Louvain Clustering [29]

V. Finding markers in Louvain clusters

VI. Pseudotime analysis

TSCAN pipeline:

I. Dimensionality reduction using gene clusters instead of genes to perform PCA

II. Clustering using Gaussian Mixture Modeling [30]

III. Cluster ordering with Minimum Spanning Tree [31]

IV. Cell ordering: Each cell is projected to an edge of the cluster order

2.2 Distinct MLscAN features

MLscAN was designed to serve both the “naïve” user (non-expert in computational biology and R programming) and the more experienced user.

Unlike most packages, a call to MLscAN implements a complete end-to-end analysis pipeline with required input only a preprocessed expression matrix. It also returns a whole directory structure and an abundance of plots and files to visualize the results produced at every pipeline stage.

The default MLscAN pipeline includes the following steps:

- Dimensionality Reduction using PCA [13]
- Clustering (cell states identification) using Gaussian Mixture Modeling [13] and parsimonious best model selection
- State-to-state transitions and trajectories inference
- For each inferred trajectory
    - Partition the ordered cells into successive micro-states (phases)

    - Identify the “key-genes”, i.e., those with bimodal and mode switching expression behavior when considering the two micro-states at the trajectory ends
    - Gene Regulatory Networks (GRN) inference for micro-states
- Flexibility to import alternative dimensionality reduction and/or clustering results before producing trajectories and GRNs downstream.
- Analysis of “mixed states” (mixtures of subpopulations)

The complete end-to-end analysis with one function call also distinguishes MLscAN from the rest of the state-of-the-art single-cell analysis packages, as they require executing each step of the pipeline manually.

This automated pipeline invocation allows the inexperienced R user to get well-described results easily without being a proficient R programmer. At the same time, by providing a significant number of additional fully documented arguments, an experienced user can tune her pipeline run to probe deeper into different aspects of interest.

Gaussian Mixture Modeling (GMM) allows casting the pairwise state trajectories Inference problem in the posterior probabilities space. In MLscAN, each cell has a posterior distribution, i.e., a probability to each cluster inferred by GMM. The largest two posterior probabilities in this distribution define the state-to-state transition that a specific cell belongs to. TSCAN and SCANPY are the only algorithms following probabilistic approaches; however, they do not offer trajectories analysis into phases (micro-states) and GRNs inference for the trajectory driving mechanisms.

MLscAN considers pairwise trajectories as means to model state-to-state interactions in the “epigenetic landscape”. Pairwise MLscAN trajectories can also be “stitched together” to reconstruct long trajectory paths connecting distant states and infer cells pseudotime ordering. This makes MLscAN also compatible with more conventional TI inferences methods that do not focus on the analysis of state-to-state interactions. We should remark that the main motivation for inferring MLscAN pairwise trajectories is to decipher the regulatory mechanisms that control them and how these may evolve dynamically in the course of a trajectory (pseudotime).

In summary, MLscAN was created to provide unbiased, unsupervised, probabilistic scRNA seq data analysis in an easy to use, yet flexible, versatile pipeline, with main focus to reconstruct in posterior probabilities space a representation of the “epigenetic landscape” of states (major cellular phenotypes) and decipher the regulatory mechanisms driving local state-to-state interactions. Moreover, MLscAN promotes by default parsimonious modeling and respects the Occam’s razor principle [32] as much as possible in all its default actions.

3. THE MLSCAN PIPELINE

This chapter will briefly describe the MLscAN pipeline and explain how the user can intervene to import her external results at various stages. We will also present some MLscAN functions that we introduced to analyze the MLscAN results further. Finally, we explain what are the “mixed states”, how they emerge and suggest different approaches to handle them.

3.1 Overview - Pipeline stages

In this section, we briefly present the MLscAN pipeline and show to which steps the user can intervene.

The main steps of the fully automated MLscAN pipeline [8] are shown below:

1. **Dataset input**

   Importing a preprocessed gene expression matrix

2. **Dimensionality Reduction**

   Gene filtering and PCA analysis

3. **Model selection**

   States (clusters) identification using Gaussian Mixture Modeling

4. **Trajectories extraction**

   Pairwise state-to-state trajectories inference

5. **Micro-States identification**

   Partitioning each trajectory to micro-states

6. **Key Genes identification**

   Identification of the key-genes driving each trajectory

7. **Gene Regulatory Networks inference**

   GRN inference for each trajectory micro-state.

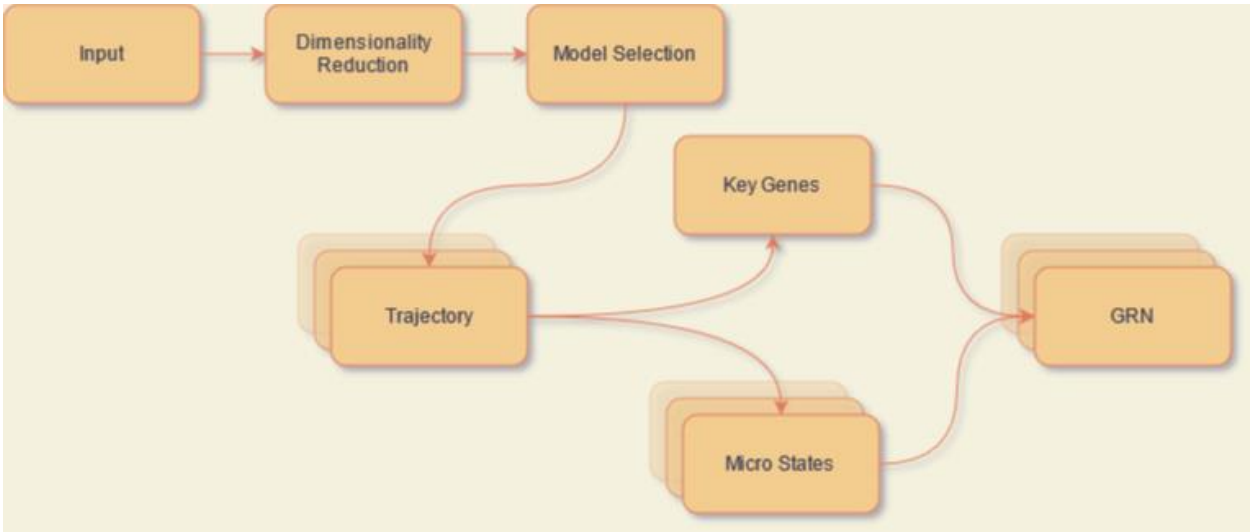

Figure 2 Overview of the MLscAN Pipeline. Since there are multiple pairwise trajectories per model, we have multiple instances of trajectory class per model. Accordingly in Micro States and GRN

A more detailed description of the pipeline is provided below:

Required Input: The only required input for running the MLscAN pipeline is the preprocessed single-cell gene expressions matrix resulting from scRNA sequencing. Any desired data transformation, preprocessing, cells, or genes filtering, should be performed before MLscAN initiation. The expected matrix should have the format cells (rows) by genes (columns).

Dimensionality Reduction: Analyzing big and complex datasets is often the case with single-cell experiments. Those two characteristics can increase processing time and add noise. To overcome those problems, MLscAN finds the 500 most variable genes of the dataset using Seurat's function Seurat::FindVariableFeatures [18]. The next step is to use PCA to reduce dimensions. Default PCA method in MLscAN is *prcomp* [33]. However, when both dimensions of the input expression matrix exceed 100, the *irlba* PCA method [27] is used as default, significantly reducing computational time.

MLscAN selects the number of PCs to be used by calculating the "knee-point" of the variance explained per PC using the algorithm described below developed by our group.

Knee-point algorithm to determine the number of PCs in PCA

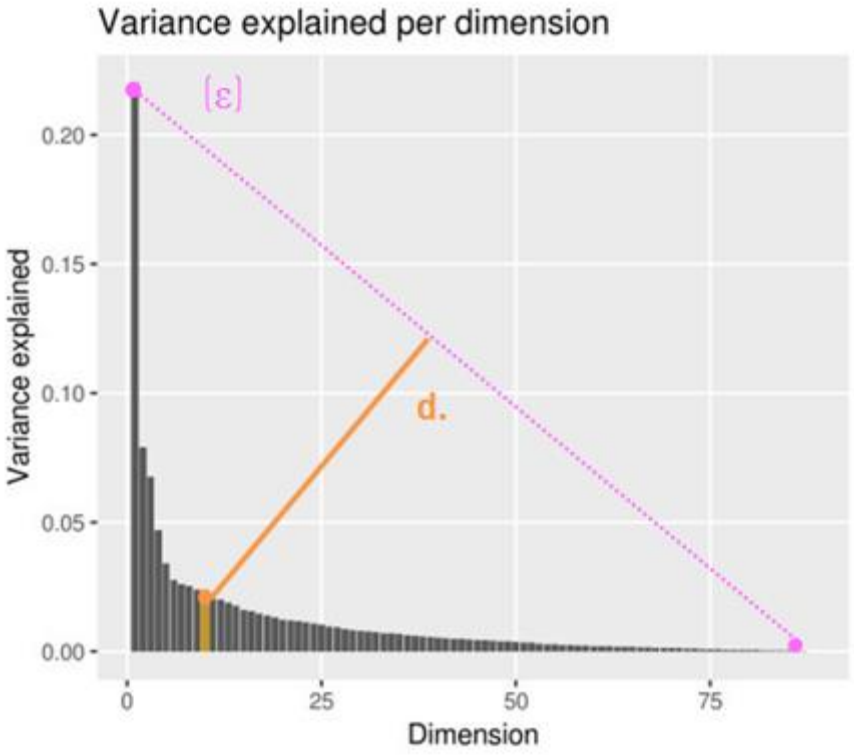

Figure 3 Plot that depicts the knee-point method for choosing number of dimensions.

Given a sequence of points (n, V_n) on the dimension vs. variance explained plane (see Figure 3), where *n* is the current number of PCs examined and V_n is the extra variance explained when using *n* PCs instead of $n-1$, $1 \leq n \leq N$, the knee-point *k* is selected as the number of PCs for which the distance *d* of point *(k, V_k)* to the line connecting the two extreme points (*1, V_1)* and *(N, V_N)* is minimized (see Figure 3).

Model selection: MLscAN uses unsupervised Gaussian Mixture Modeling (GMM) [30] to identify cell clusters in dimensionality reduced data. As it is using the *mclust* package [34], MLscAN considers a range for the number of states (default [2:9]) and the gaussian component types. The default is to consider all the 14 “models names” (covariance matrix structures) supported by *mclust*. MLscAN then finds in a parsimonious manner the best model (number of states, model name combination) by applying the developed by our team “deltaBIC” (□BIC) algorithm.

This □BIC algorithm uses the Bayesian Information Criterion (BIC) score of the different examined GMM models. Specifically, it considers the difference in BIC values among the largest-BIC GMM models as the number of states increases from low to high values in the defined range. Briefly, for a given number of states *n*, the absolute BIC difference (deltaBIC) of the largest-BIC GMM model to the largest-BIC GMM models for the

previous *(n-1)* and the next (*n+1)* number of states is computed and compared to a threshold. In the end, the simplest model (Occam's razor principle) having the smallest number of states and deltaBIC values to its left and right "neighbors" below the threshold is identified and considered the "best" parsimonious model.

Trajectories inference: The best GMM model produces a posterior probabilities matrix (cells x posterior probabilities to inferred states), allowing us to cast trajectories inference in posterior probabilities space. For MLscAN, a trajectory A-to-B, connecting a "start" state (A) to a "destination" state (B), consists of the subset of cells in the matrix whose two largest posterior probabilities are for states A and B, ordered in decreasing probability to the start state A. For a trajectory to be considered valid, (i) the total number of its cells should be at least 6, and (ii) at least 3 cells should belong to each one of the two states defining the trajectory. Sometimes we refer to the start (destination) states also as the "ground" ("landing") states, respectively.

Transitions and their propensity: If a trajectory does not meet the above two conditions, it is still considered as a state transition. A transition is characterized by its "strength" or "propensity". The propensity of an (A,B) transition is the sum of two ratios: the ratio of state A's cells having second-highest posterior to state B, and the ratio of state B's cells having second-highest posterior to state A [8].

For example, let's assume that there are three states overall in the best GMM model: A, B, and C. Moreover, 90% of cells in state A have their second-highest posterior probability in state B, and 60% of cells in state B have their second-highest posterior probability in state A. In that case, the propensity of the transition (A, B) will be 1.5 (=0.9 + 0.6). Obviously, the maximum value for a transition propensity is two (2), and that happens when all cells of state A "look towards" (have second-highest posterior for) state B, and in addition, all cells of state B "look towards" state A. Intuitively, the larger the percentages of cells of the two interacting states (A, B) that "look towards" each other in posterior probabilities space, the higher the support in the model for the existence of the specific state transition. This support is what the transition propensity ties to estimate. If all cells of a transition happen to belong to one of the two interacting states (say to state A) the transition (A, B) is called unidirectional (A-to-B transition).

Trajectory Micro-states: Micro-states (m-states) are non-overlapping consecutive subsets of the ordered trajectory cells. Typically, we can identify three successive micro-states (phases) that partition the cells of an A-to-B trajectory. They are called: the *ground m-state*, at the beginning with cells departing from the start state A, the *transitional m-state*, in the middle with cells mid-way in the trajectory, and the *landing m-state*, at the end with cells arriving in the destination state B. MLscAN uses a splines-based algorithm and the posterior probability curve of the ordered trajectory cells to the start state A to determine the micro-state boundaries. For an m-state to be considered valid it should contain at least two (2) cells. When the algorithm cannot extract three valid micro-states, then two m-states, called the start and the destination m-states are defined, including the

trajectory cells belonging (having highest posterior) to the start and the destination states, respectively.

Key-genes identification: As a trajectory models a one-way biological process, “key-genes” of a trajectory are those genes that govern this underlying biological process dynamics. MLscAN detects “key-genes” based on two properties: (i) They exhibit bimodal expression along the trajectory, and (ii) switch expression mode (low-to-high or the opposite) as cells “move” from the ground to the landing micro-states. MLscAN has its own algorithm that checks for those conditions [8] but also supports other algorithms that look for genes with differential expression between the ground and landing m-states. The additional supported methods are currently: t-test, MAS T [35], edgeR [36], which fits the counts using a Negative Binomial distribution, and SwitchDE [37], which identifies switch-like behavior by fitting a sigmoid curve on the gene expression data. While any method can be used alone, MLscAN also offers a helper function that implements a majority voting scheme after calling all of them. In this case, a candidate gene must be voted by at least 3 out of the 5 currently supported methods to be a key-gene. Voting helps select fewer key-genes for which we are more confident at the expense of increased computation time.

Gene Regulatory Networks inference: In MLscAN a Gene regulatory Network (GRN) is a directed graph with nodes the key-genes capturing the regulatory relationships between gene-regulators and gene-targets [7]. MLscAN generates a GRN for each micro-state of a trajectory using the GENIE3 algorithm [8] that is based on bootstrapping [38] and Random Forests [39] to infer the regulatory relationships for each key-gene target.

3.2 Adding flexibility - Integration with external processing

Even though MLscAN is a pipeline that performs by default automatically all stages of the above-described analysis, it also allows a user to import results produced outside the package easily and at all its pipeline stages.

Importing Dimensionality Reduction results: In single-cell experiments, it is common to use tSNE [20] or UMAP [16] to visualize high-dimensional data. However, we can also use tSNE [20] and UMAP [16] dimensionality reduced data to perform clustering using GMM in MLscAN. In complex datasets, this may help GMM to identify the correct clusters. Another common approach is using PCA [40] output as input in UMAP [16] and tSNE [20] to reduce computation time. The user may import a dimensionality reduced data matrix to MLscAN produced using any method she chooses using the dimRedData argument. We provide a use case demonstrating this flexibility in Chapter 5.

GMM initialization: There are cases where GMM cannot provide the appropriate clustering for a particular dataset and other clustering techniques may outperform GMM. Therefore, a user may want to provide already available clustering results and is interested in examining the trajectories MLscAN produces with them. We have added this flexibility allowing a user to initialize MLscAN clustering with some known class labels for

the cells used as initial conditions for GMM. By default, GMM initialization is conducted automatically by *mclust* using hierarchical agglomerative clustering [34].

To perform an initialized MLscAN run, the user should provide in an argument called modelInit a named vector; its names are the corresponding cells, and its values are their initial states. MLscAN uses this named vector to produce an initial posterior probabilities matrix of all cells [rows] with the value of 1 to their declared state [column] and 0 to the rest of the states that is used as the initial value for GMM. We provide and discuss a use case example using GMM initialization with known clustering results in Chapter 6.

Importing a Posterior Probabilities matrix: The user may even import a posterior probabilities matrix using the argument modelPostProbs. This option can be helpful if the user wants to use posterior results from a previous MLscAN analysis that stopped at an earlier stage without producing trajectories and wishes to continue the run without repeating the same analysis. Also, if the user wants to import a posterior probabilities matrix calculated using a different method than GMM and still wants to produce trajectories and GRNs using MLscAN.

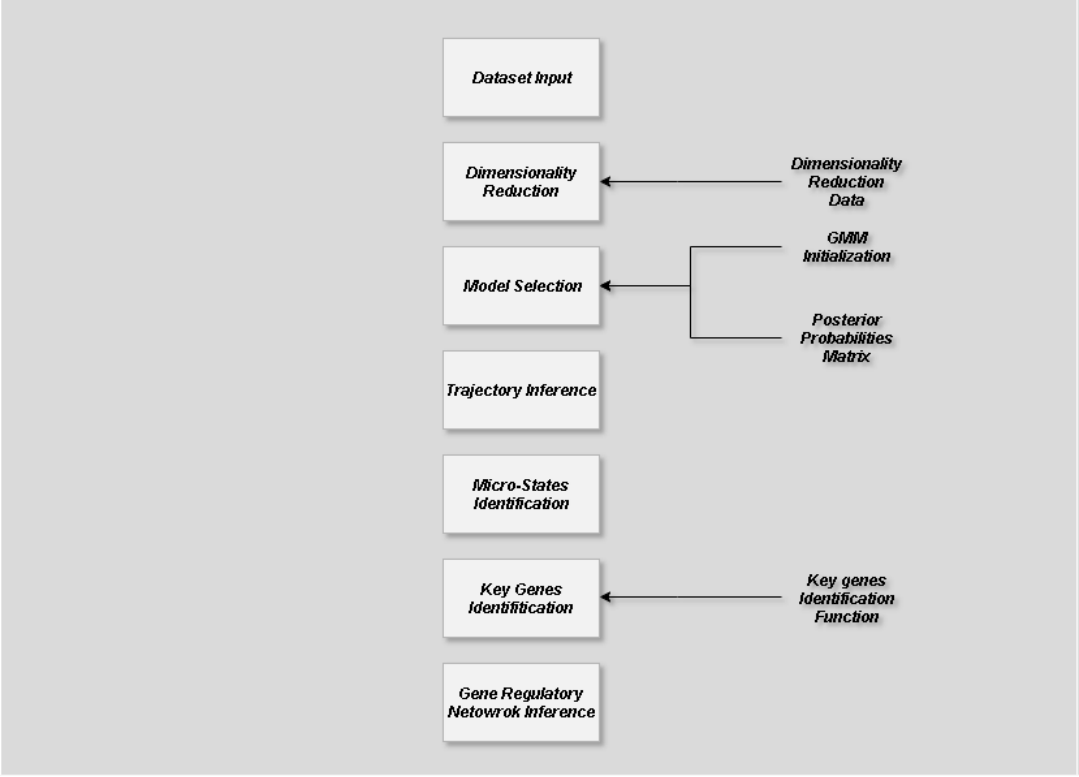

Figure 4 . Overview of the MLscAN Pipeline pointing out the stages of the analysis where a user can intervene and import his data or functions

The figure above summarizes the steps in the MLscAN analysis where a user can import either their results or a certain function to customize his run according to his will.

It is essential to remark that the MLscAN object can provide the results of all stages of the analysis after they have been produced using the proper MLscAN getter function.

Also, MLscAN allows partial runs that do not go all the way to producing trajectories and/or GRNs that are time-consuming steps before the user has fully explored the models space and has settled on the model that best fits their dataset. The possible options are:

- Stop at model: Do not continue to trajectory inference
- Stop at trajectory: Do not continue to micro states and key genes identification

3.3 SD plot analysis

It is advisable to avoid splitting cells into too many or too few clusters, as this will lead to introducing noise in the data, which will carry over through the pipeline and affect the results. MLscAN contains functionality to aid users in varying the number of PCs and see how that affects the number of inferred states before they commit to a model. This is possible through a function called plotSD. PlotSD needs as input the preprocessed gene expression matrix of the dataset we plan to analyze with MLscAN. The user also needs to define the range of the number of PC's and the model name of the GMM models she desires to explore. The function then applies dimensionality reduction, estimates the number of states using GMM, and reports the cumulative variance explained as the number of dimensions increases. It also locates the "knee-point" of the variance per PC curve and suggests that point as a possible good solution to try first for the given model name. Below, we present an example of the plotSD function application to the pancreatic cells dataset used in Chapter 4 [41].The plot suggests using five (5) PCs leading to a model with three (3) states when using the model name EEV in GMM.

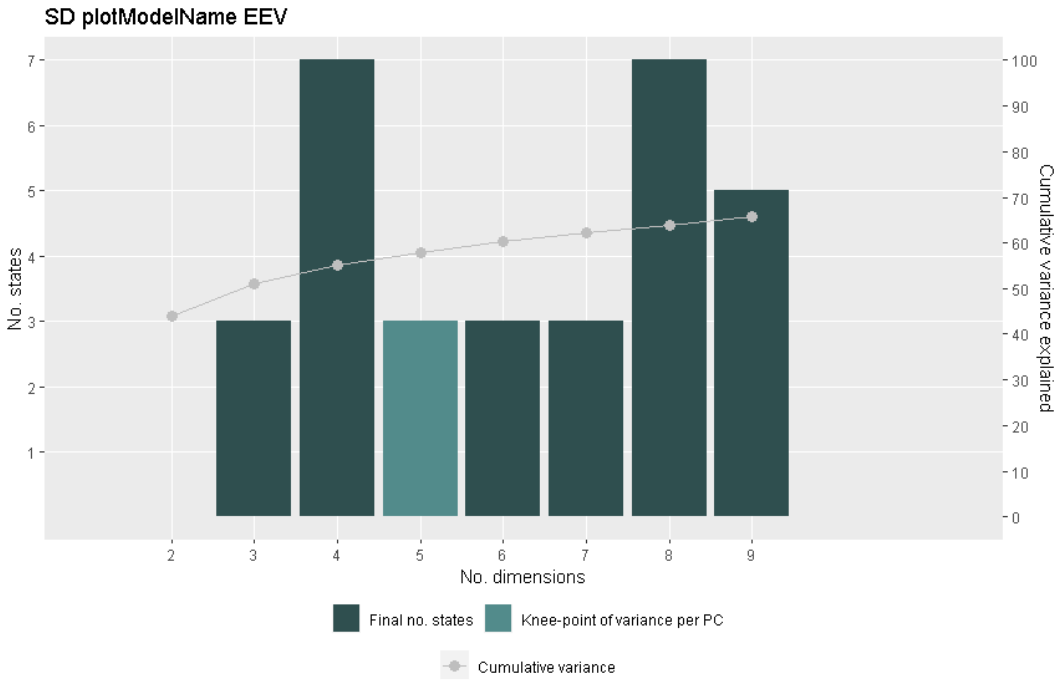

Figure 5 SD plot depicting PCs, States, and cumulative variance explained

3.4 Results analysis flexibility

During this thesis, we introduced two functions for further analysis of the MLscAN results. These functions are not called during the MLscAN pipeline execution. In contrast, they use the produced MLscAN run object as input to provide additional information on demand. In the future, more functions of this nature (post-run analysis) will be added to the package.

3.4.1 Trajectory paths

Most trajectory inference packages have as a final goal to create a pseudo time cell ordering [24]. Even though MLscAN focuses on inferring and analyzing state-to-state transitions, we have created a function that assigns pseudotime to cells. TrajectoryPath is an external function that can produce cell ordering for a certain trajectory path. It takes as input an MLscAN object and the name of a valid trajectory path and returns a matrix with three columns: cell name, cell order, cell cluster. To decide cell ordering, it uses the posterior probabilities for each state-to-state trajectory in the path. We provide an example of using this function in the use case of Chapter 6.1.

3.4.2 Cluster markers analysis

In case a user does not have prior knowledge about the types of cells in the analyzed dataset, we provide a function that identifies clusters markers. The function *FindClustersMarkers* requires as input an MLscAN object and a vector of states (default all inferred states). It provides a list containing the most differentiated genes for each state in regards to the rest of the states in the list. For this particular function implementation, we use Scran's findMarkers function [42]. We provide an example of using FindClustersMarkers in the use case of Chapter 5.

3.5 Mixed States

After conducting a plethora of MLscAN runs using datasets of different sizes, we detected that the best GMM model might occasionally create a false state with a very big variance in PC space parsing through the other states. We name this overreaching cluster a "mixed state" because it usually corresponds to a mixture of smaller states (clusters) that parsimonious modeling refused to split as it promotes "simpler" solutions. If not properly handled, mixed states may cause problems in downstream trajectories analysis since all the other states may be "attracted" to a mixed state.

3.5.1 MLscAN Mixed States identification

MLscAN detects potential mixed states by calculating the variance for the first and second dimensions after dimensionality reduction for each inferred cluster (state) and comparing them with the corresponding variance for the whole dataset. Suppose that a cluster's variance in either one of the first two dimensions is bigger than the entire dataset's variance for the same dimension. In that case, the cluster is characterized as a potential mixed state, and its name will end with a "#" character.

It is also possible to stop the analysis when a possible mixed state appears by using the option MLscANIgnoreMixedState= FALSE. Moreover, MLscAN isolates the mixed states in a separate output directory and provides plots to visualize their cells in low dimensions to get a visual impression of the situation.

3.5.2 Mixed State handling approaches

There are two orthogonal approaches in handling a mixed state, removing it or further analyzing it.

First, if a mixed state contains a very small percentage of the total cells, it may be a state of outliers. In this case, the user can easily remove those cells and either rerun MLscAN or start an MLscAN run initialized with the original MLscAN run's clusters minus the mixed state cluster. We provide an example of this procedure in the use case of Chapter 6.1.

Otherwise, if a mixed state consists of more cells or the user wants to analyze it anyway, MLscAN can come to the rescue for that purpose. For example, the mixed state cells can be easily isolated, and MLscAN can be called to analyze them alone. Then the user can see if some interesting cell subpopulations have emerged that warrant more attention. In that case, it is possible to incorporate them easily in a final MLscAN run and give them the chance to contribute to the epigenetic landscape's inference. By combining the clustering results of the original MLscAN run and the mixed state alone MLscAN run, we can initialize a final MLscAN run that may provide better-resolved clusters of different sizes without increasing too much the number of states (keeping in mind parsimonious modeling) while getting more informative trajectories. The use case of Chapter 6.2 provides a detailed example of this split-then-integrate approach.

We can also use the above strategies in combination and recursively when processing large datasets if an MLscAN run detects more than one mixed states. To the best of our knowledge, identifying mixed states is a unique feature of MLscAN, and properly handling them is an added recent feature and a contribution of this thesis work.

4. USE CASE - MLSCAN WORKFLOW

In this section, we present the main workflow of the MLscAN R package for single-cell RNA seq data analysis to demonstrate its core capabilities. We show and explain the different results and plots that the package produces to help the user interpret the results obtained at different stages of the MLscAN computational pipeline.

In this use case, we use the Gene Expression Omnibus data under the accession GSE83139. Briefly, the authors of this study perform scRNA-seq on pancreatic islet cells of all four types (α, β, and PPY) from diabetic and non-diabetic donors [41]. This dataset serves well the purpose of illustrating the main capabilities of MLscAN as the authors have already provided a curated label for each cell based on signature genes' expression.

In the analysis that follows, we will use only the β-cells taken from three categories: diabetic adults, non-diabetic adults, and children. The initial expression matrix contained 88 β-cells with 19.949 genes. The preprocessed expression matrix we used here consists of only 86 cells (we removed two outlier cells) and 123 genes. The genes were selected by filtering the 19.949 initial genes using a two-part generalized linear model (GLM), which explicitly considers stochastic dropout and bimodal expression provided by package MAST [35], combined with a t-test to discover differentially expressed genes.

4.1 First MLscAN run

```
res <- MLscAN(exprData=exprs, #Expression matrix, mandatory

        MLscANCellFeatures=cellFeatures, #Cell features, optional
        kgGenesSelFun=kg_voting(), #Function to select key genes using
                voting(helper)among 5 methods
        modelNumStates = c(2:7), #Selected range of number of States to be
                 considered in GMM based model selection
        modelStateNameMode="mostFreqPerState", #Inferred States naming scheme
        MLscANOutMode = "all") # Produce all plots
```

```
##
## Creating the MLscAN object...
## Performing dimensionality reduction...
## Creating the model...
## Forming the sub-populations...
## Possible mixed state(s): child
## Creating the trajectories...
## Generating the output files...
```

We start with a near default MLscAN run. In this run, only two arguments assume non-default values for the possible number of states and the method we employ to detect the “key genes” for inferred trajectories.

We use a different range of states from the default [2:9] to save computation time since we know that the expected number of states will be less than 9.

Beyond its default method, MLscAN supports some more well-established techniques in the literature to select “key genes” for inferred trajectories and build Gene Regulatory Networks (GRNs). Those methods are: t-test, MAST [35], edgeR [36], which fits the counts using a Negative Binomial distribution, and SwitchDE [37], which identifies switch-like behavior by fitting a sigmoid curve on the gene expression data. While any one method can be used alone, MLscAN can also implement a majority voting scheme after calling all of them. In that case, a gene is considered a “key-gene” only if chosen (voted) by at least 3 out of the 5 currently supported methods. This can help de-clutter the output by selecting key genes for which we are more confident. So, in our current analysis, we selected this voting scheme.

4.2 The run summary

```
print(res) #Check out the summary output
```

```
An S4 object of class `MLscAN`
```

```
An S4 object of class `MLscAN`

##### GENERAL INFORMATION ####
- Initial expression data: 86 cells x 123 genes
- Expression data used: 86 cells x 123 genes
- Mixed States: 1 (child#), mixed state cells: 18 (20.9%)
- No. dimensions of dimensionality reduction results used: 5
** Variance explained: 55.9%
** Dim. names: PC1, PC2, PC3, PC4, PC5

- Confusion matrix (%) - state:
                type `adult` type `T2D` type `child`
state `T2D`       9.090909  90.909091      0.00000
state `child#`    5.555556   0.000000     94.44444
```

state `adult` 97.826087 2.173913 0.00000

- Confusion matrix (%) - type:

	state `T2D`	state `child#`	state `adult`
type `adult`	4.166667	2.083333	93.750000
type `T2D`	95.238095	0.000000	4.761905
type `child`	0.000000	100.000000	0.000000

CELL TYPES - GROUND TRUTH

- Type `adult`: 48 cells (55.8%)

** 2 cells in state `T2D` (9.1% of the state)

** 1 cells in state `child#` (5.6% of the state)

** 45 cells in state `adult` (97.8% of the state)

- Type `T2D`: 21 cells (24.4%)

** 20 cells in state `T2D` (90.9% of the state)

** 0 cells in state `child#` (0.0% of the state)

** 1 cells in state `adult` (2.2% of the state)

- Type `child`: 17 cells (19.8%)

** 0 cells in state `T2D` (0.0% of the state)

** 17 cells in state `child#` (94.4% of the state)

** 0 cells in state `adult` (0.0% of the state)

CELL STATES - INFERRED

- 3 states

** State names: T2D, child#, adult

- State `T2D`: 22 cells (25.6% of all cells)

** 22 cells (100.0%) in the sub-population of the state

** 2nd largest a posteriori probability:

---- state `child#`: 0 cells (0.0%)

---- state `adult`: 22 cells (100.0%)

- State `child#`: 18 cells (20.9% of all cells)

** 18 cells (100.0%) in the sub-population of the state

** 2nd largest a posteriori probability:

---- state `T2D`: 15 cells (83.3%)

---- state `adult`: 2 cells (11.1%)

- State `adult`: 46 cells (53.5% of all cells)

** 46 cells (100.0%) in the sub-population of the state

** 2nd largest a posteriori probability:

---- state `T2D`: 45 cells (97.8%)

---- state `child#`: 1 cells (2.2%)

TRAJECTORIES - INFERRED

- Trajectory `T2D-to-adult`

** 67 cells (77.9%)

** No. micro-states: 3

** No. micro-states: 3

---- ground micro-state: 20 cells (29.9%), valid GRN: TRUE

---- transitional micro-state: 15 cells (22.4%), valid GRN: TRUE

---- landing micro-state: 32 cells (47.8%), valid GRN: TRUE

** Valid key-genes: TRUE

---- Key-genes default method: No

---- No. key-genes: 19 (15.4% of all genes)

---- Key-genes: HLA-G,HLA-J,PRSS3P2,PRSS1,CLPS,REG1B,HLA-F,CTRB2,MT1E,CPB1,PRSS2,TMEM37,C9orf16,HLA-L,CELF2,OLFM4,ZNF880,REG1A,RPL19P12

- Trajectory `adult-to-T2D`

** 67 cells (77.9%)

** No. micro-states: 3

** No. micro-states: 3

---- ground micro-state: 24 cells (35.8%), valid GRN: TRUE

---- transitional micro-state: 23 cells (34.3%), valid GRN: TRUE

---- landing micro-state: 20 cells (29.9%), valid GRN: TRUE

** Valid key-genes: TRUE

---- Key-genes default method: No

---- No. key-genes: 19 (15.4% of all genes)

---- Key-genes: HLA-G,PRSS3P2,HLA-J,REG1B,PRSS1,CLPS,TMEM37,HLA-F,PRSS2,CTRB2,CELF2,CPB1,MT1E,ZNF880,HLA-H,OLFM4,HLA-L,REG1A,C9orf16

By invoking a print on the MLscAN object we obtain a human-friendly overview of the unbiased data analysis. It includes summary of information about the dimensionality reduction, namely the number of Principal Components selected, and the total variance explained.

Furthermore, it provides a confusion matrix comparing the curated cell types with the cell states inferred using unsupervised Generalized Mixture Modeling (GMM) based clustering. We observe that each state (cluster) is populated mainly by one cell type, and each inferred state is named after the cell type that is most frequently (more than 70%) represented in the cells of that state.

A distinguishing characteristic of MLscAN is that it not only infers state-to-state trajectories but also partitions the cells of a trajectory into three consecutive subsets that we call “micro-states” [8]. We can think of a micro-state as a phase of the biological progression modeled by an MLscAN state transition. Each cell's micro-state is decided based on the posterior probabilities of being in each one of the two states defining the trajectory. For example, a cell belongs to the “ground micro-state” if its posterior probability to the “departing”, or “ground”, state of a trajectory is much larger than the posterior to the “destination”, or “landing”, state of the trajectory, and vice versa. There may also exist a transitory micro-state, including cells with these two posterior probabilities close in value.

Moreover, MLscAN also identifies the “key-genes” of each state-to-state trajectory, i.e., genes with a markedly differential expression in the cells of the ground and landing micro-states.

A unique feature of MLscAN is that it also infers Gene Regulatory Networks (GRNs) of the key-genes for each micro-state of a trajectory. These GRNs provide insights into how the dynamic regulatory mechanisms governing a state transition evolve with pseudo-time.

While MLscAN automatically produces many useful plots, special attention is given here only to some of the more informative ones. Plots can be generated automatically using the argument MLscANOutMode and setting it to “all” or “plots”. However, MLscAN also provides functionality to suppress plotting or generate individual plots on demand.

4.3 The MLscAN produced outputs

An MLscAN run produces by default an output split into six main folders with many plots to visually assess the outcomes of the various analysis stages of the computational pipeline at different levels of detail. In brief, these folders are the following:

Expr_plots folder to help the user understand the statistical characteristics of the gene expression profiles matrix of the analyzed dataset,

DimRed_plots folder, providing different views of the dimensionality reduction stage results,

States_plots folder, providing a simple recap of the GMM clustering process for every inferred state,

Types_plots folder correlating the curated cell types, as provided by the user, to the inferred by MLscAN states and transition states,

Traj_plots folder analyzing each trajectory, down to its micro-states, key-genes, and inferred GRNs.

Overview_plots, providing a top-level summary of the whole MLscAN run.

Users are encouraged to browse the Overview_plots folder before drilling down to the other directories, starting from the dimensionality reduction plots, state and type plots, and finally, the trajectory plots. To get a sense of the analyzed expression matrix characteristics, the Expr_plots are also useful. To keep the discussion confined, we will present here examples of only a subset of the produced plots.

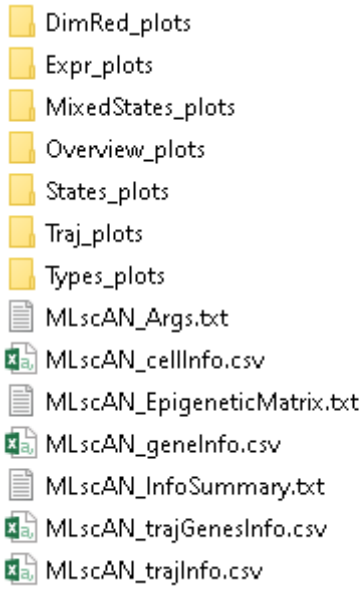

Figure 6 Picture of MLscAN output directory

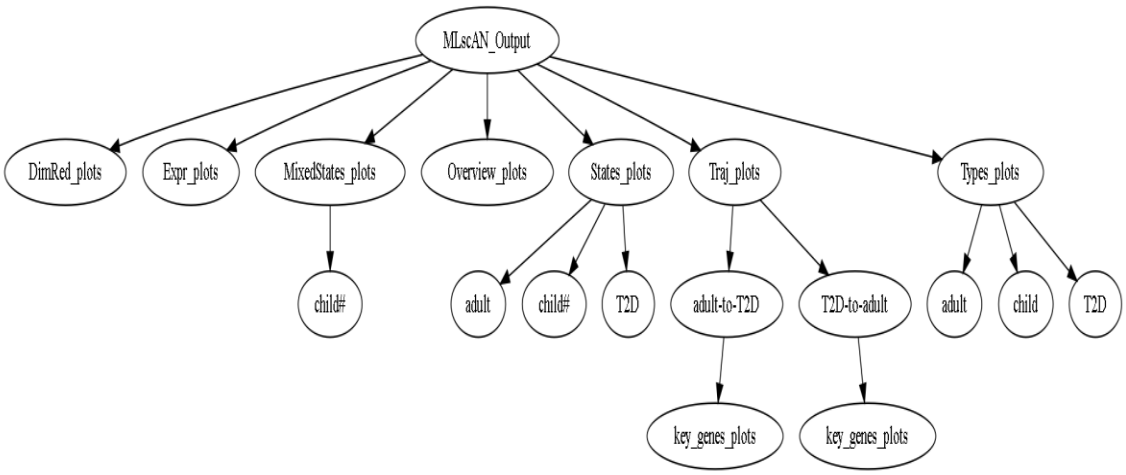

Figure 7 Tree with depicting all folders and subfolder of an MLscAN output

The first picture is a snapshot of the MLscAN output directory created for the example analysis below. In the second picture, we can see a tree containing all folders of MLscAN output with their respective subfolders created by a complete MLscAN run. We can observe in States_plots subdirectories for all three states and in the Traj_plots subdirectories for the two inferred trajectories.

As we move forward we will present representative examples from the plethora of plots that MLscAN generates. Not much emphasis will be placed to the biological interpretation of the results, as the purpose of this vignette is to provide an overview of the MLscAN analysis pipeline. While most of the plots shown here are generated automatically, the code snippets we provide can also be used to create them individually.

4.4 Characteristics of the gene expression matrix

```
plotExprBoxplot(exprData(res),
        output_filename = "tmp_plot1.png")
```

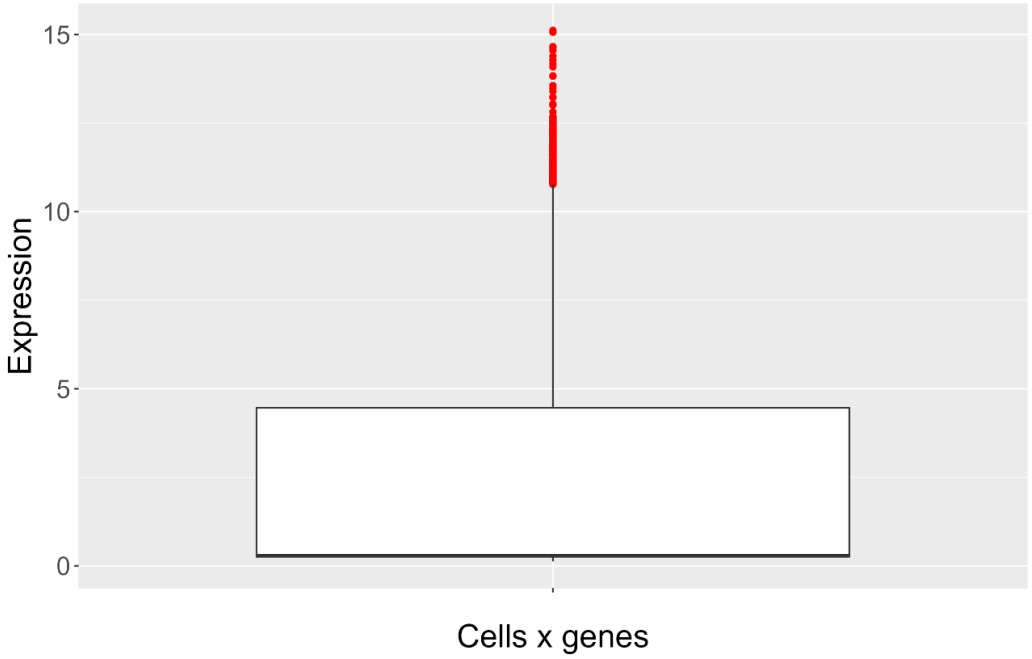

Figure 8. Gene expression boxplot

The plots in this section characterize the expression matrix and provide a useful overview of its statistical characteristics. For example, the boxplot above shows that most log-transformed gene expression matrix entries have a low value. It also explains why a log transformation was used for this dataset. MLscAN uses GMM to infer cell states which assumes that cell subpopulations in the dataset follow a normal distribution. Substantial differences in the data values (many orders of magnitude in un-logged data) can lead to spurious results.

```
plotExprMinMeanMax(exprData(res),
            output_filename = "tmp_plot2.png")
```

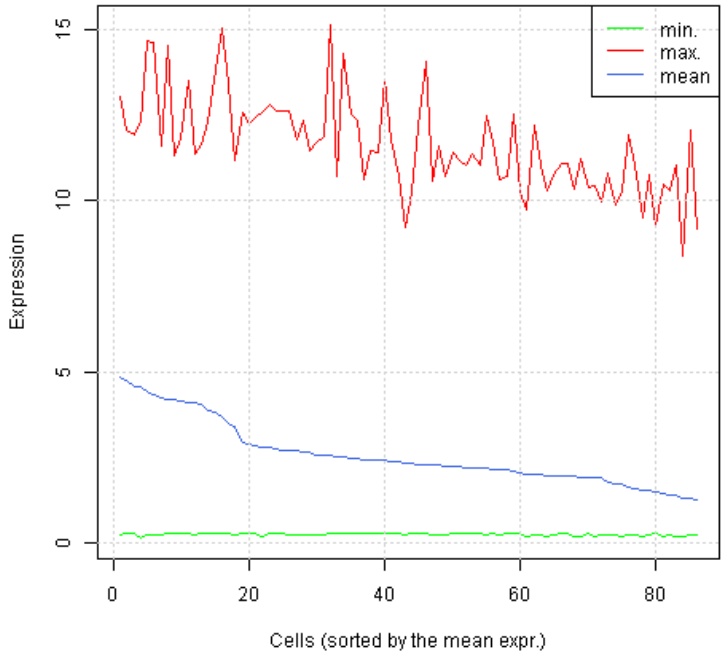

Figure 9 Min, mean, and max gene expression values per cell

In this plot, cells are sorted in descending order of their mean expression value, and the maximum, minimum, and mean gene expression values are shown. We can see that some cells have markedly lower mean gene expression than others, which might indicate differences in sequencing depth.

```
plotExprMeanSD(exprData(res),
        title = "Mean and SD of expression per cell",
        output_filename="tmp_plot3.png")
```

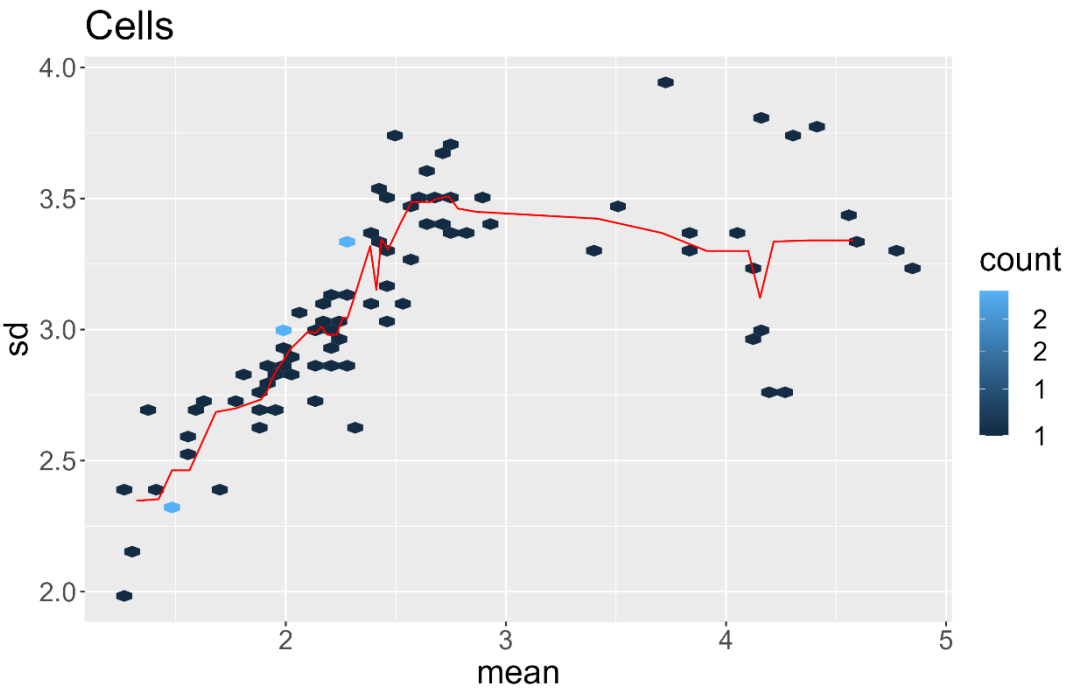

Figure 10 Mean vs. Standard Deviation per cell

The mean gene expression vs. the standard deviation per cell is provided in the above plot. We can see that the standard deviation levels are on the same order of magnitude as the mean levels. Since overdispersion is not that prominent, it is not problematic to use GMM downstream, which assumes a normal distribution for its components.

```
plotExprHeatmap(exprData(res),
                save = TRUE,
                saveDir = ".",
                z_scores = FALSE)
```

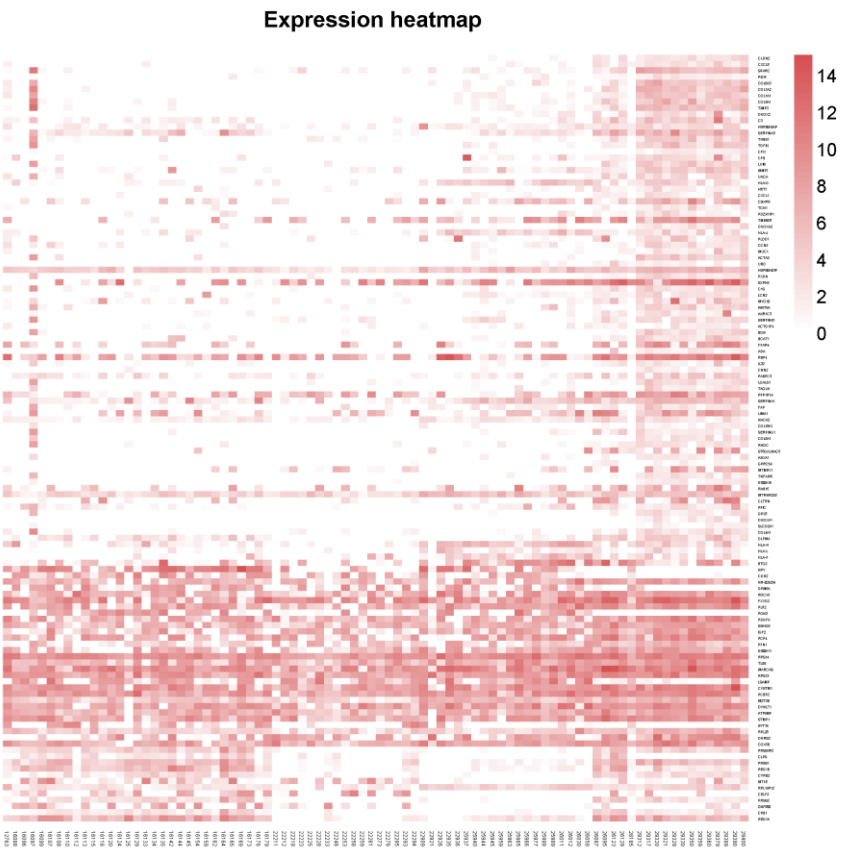

Figure 11 Gene expression heatmap

The above heatmap shows a sizeable difference in gene expression for many genes within the cells, with a notable “backbone” of genes (rows) with high expression across all cells (columns).

Suppose the user prefers to inspect the relative expression values in the heatmap. In that case, z-score normalization can be performed by MLscAN automatically by setting the z_scores argument in the plotExprHeatmap function to TRUE. The produced heatmap will then depict the z-scores of the transformed gene expressions matrix.

4.5 Dimensionality reduction

```
plotVarianceComb(res, #MLscAN object
                 save = TRUE, #Save file
                 fileOnly = TRUE, #Save file, don't display
                 saveDir = ".") #Where to save?
```

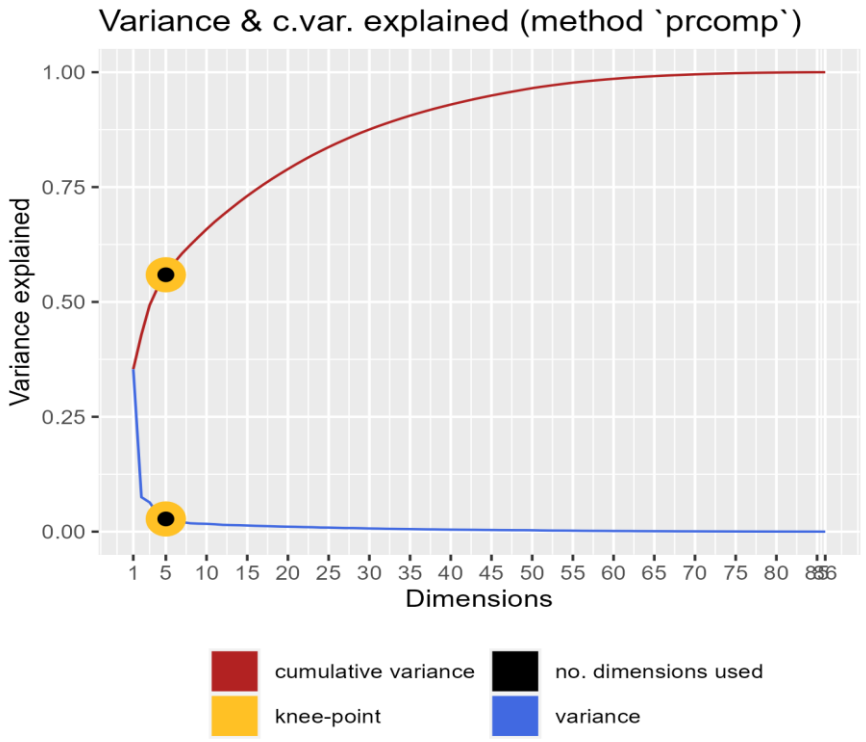

Figure 12 Visualizing the PCA results

The above plot is a mainstay in any analysis involving Principal Components Analysis. It shows the variance explained per PC component (blue line) and the cumulative variance explained (red line) in the same plot. Additionally, it highlights the number of PCs MLscAN suggests using based on the "knee-point" method (default) and the number of dimensions the user has selected (which is the suggested by MLscAN in this case). Moreover, we can see in the title that the SVD method used was *prcomp* [33]. MLscAN also supports using the *ilrba* PCA method [27] that is faster for very large datasets.

```
plotDimRed(res, #MLscAN object
        dim1="PC1", #Which dimension to plot?
        dim2="PC2", #Which dimension to plot?
        feature="cellType",#Annotate cells based on ground truth cell type
        save = TRUE, #Save to file
        saveDir = ".", #Where to save?
        fileOnly = TRUE) #Save file, don't display
```

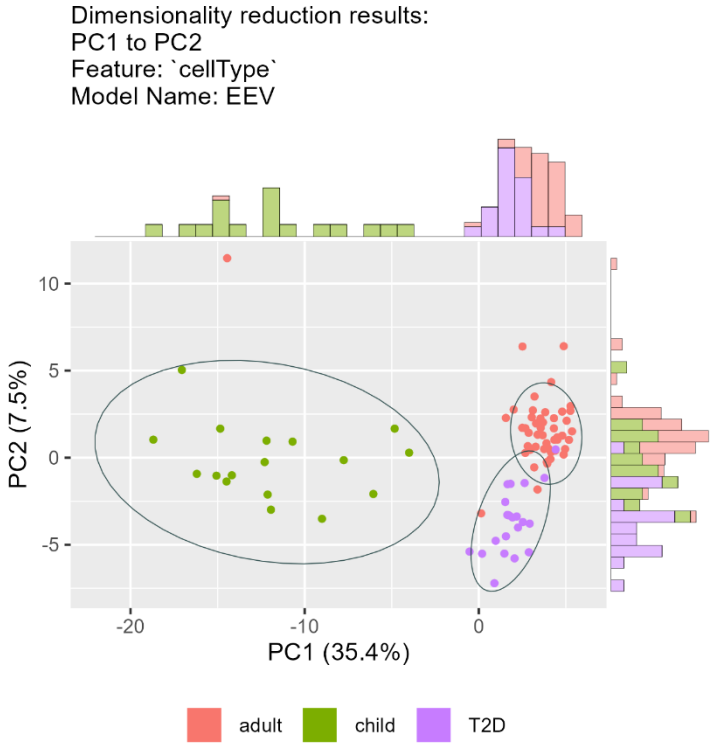

Figure 13 PC1 vs. PC2 with cells colored by cell type (ground truth)

We can see from the above plot that the cell types (ground truth) are mapped nicely into the states (inferred using unsupervised ML). So, for example, PC1 seems to capture age information as it separate child cells from adult normal and T2D cells, while PC2 separates the adult normal from the adult T2D cells.

```
plotDimRedPairs(res, #MLscAN object
        dims=paste0("PC", seq(3)), #Can be done with >2 dims
        save = TRUE, #Save plot to file
        saveDir = ".", #Where to save?
        fileOnly = TRUE) #Save file, don't display
```

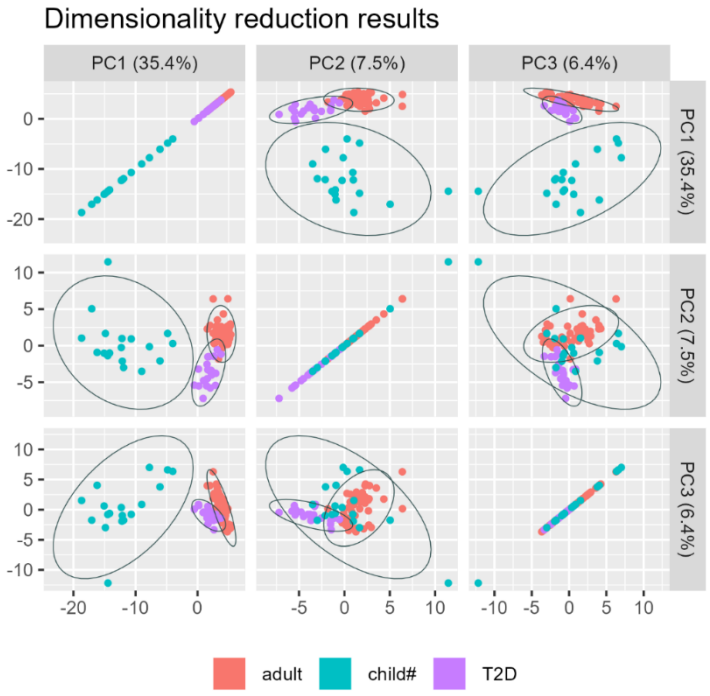

Figure 14 Cells on PC1, PC2, and PC3 dimensions colored by cell type

More detailed plots can be generated with the *plotDimRedPairs* function, showing all PCs pair-wise and how the cells (samples) are projected to them. This can help us understand what aspect of the data each PC is capturing.

4.6 Unsupervised model selection

MLscAN also creates an overview plot showing the Bayesian Information Criterion (BIC) score of different candidate models it examined. This helps the user understand how its unbiased, unsupervised, parsimonious modeling approach arrives at a model selection.

```
plotBIC(res, #The MLscAN object
      save = TRUE, #Save the image (png)
      saveDir = ".", #Where to save?
      showModelNames = TRUE, # Show BIC plot for all GMM model names
      fileOnly = TRUE) #Save in file, don't display
```

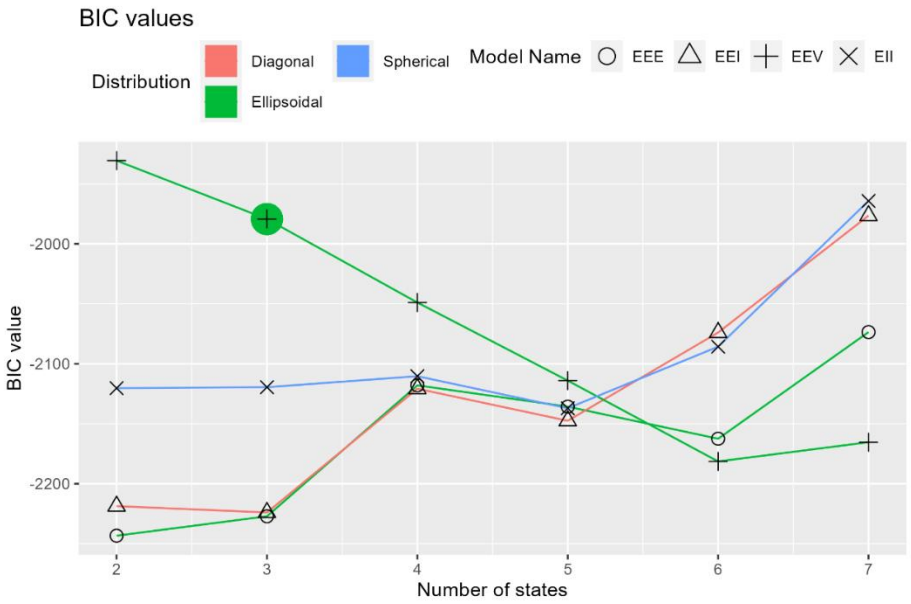

Figure 15 BIC value of different GMM models as the number of inferred states (clusters) increases

MLscAN follows by default an unbiased unsupervised parsimonious model selection approach. It advocates using the simplest possible model (with the smallest number of states) that explains (fits) the data reasonably well following the Occam's razor principle [43]. Of course, the users may override the default method and use alternative approaches if they so desire.

To that end, we identify the "best" GMM model using MLscAN "deltaBIC" method. Briefly, with the number of states increasing, the absolute BIC difference (deltaBIC) of the largest-BIC (best performing) GMM model to the largest-BIC models obtained for the previous and the following number of states is computed and compared to a threshold. In the end, the simplest model with the smallest number of states having deltaBIC values to both its neighbors below the threshold is identified. This is considered by MLscAN as the "best" model in the parsimonious "deltaBIC sense" and is marked in the plot. In our case, we see that the GMM model with name EEV, having only three components (states) with ellipsoidal covariance matrix shape is the optimal.

The above plot provides a visualization of all BIC values for every GMM model type (model name, i.e., covariance matrix structure) and number of states considered in model selection process, also highlighting the covariance structure family each considered model belongs to.

```
plotStatesComposition(res, #MLscAN object
        feature="cellType", #Annotate cells using ground truth cell types
        save = TRUE, #Save to file
        saveDir = ".", #Where to save?
        fileOnly = TRUE) #Save in file, don't display
```

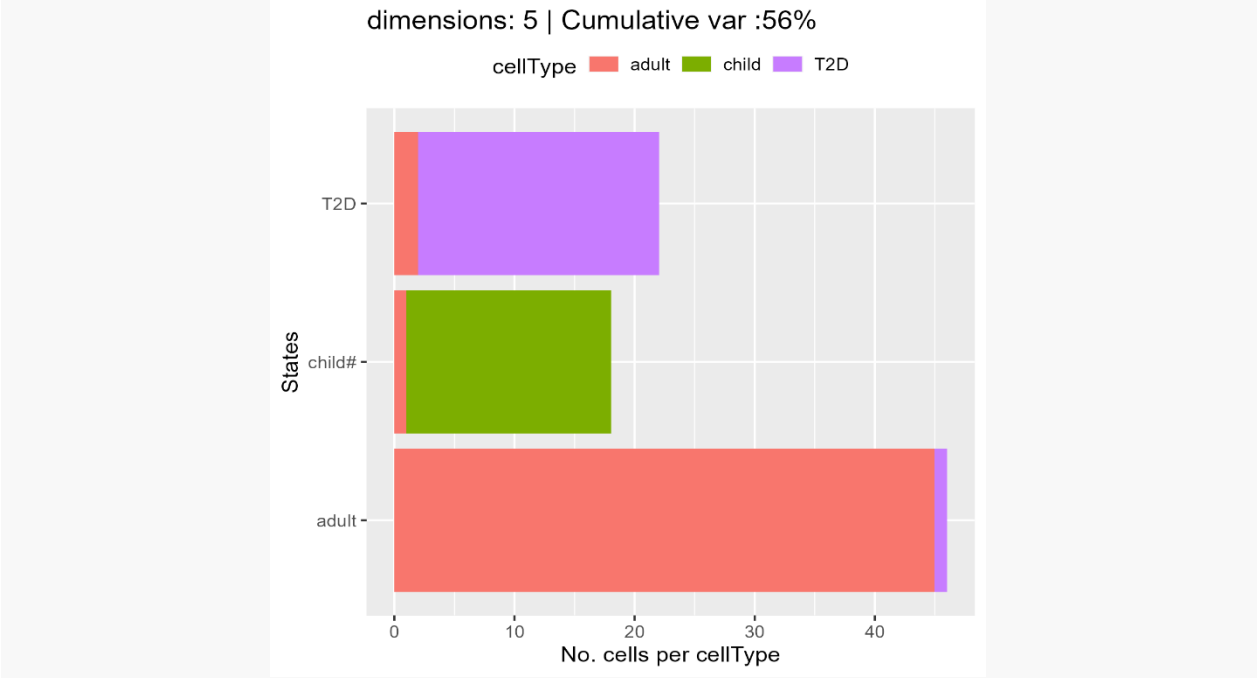

Figure 16 States Composition plots colored using cell type

The State composition plot above is the most efficient way to depict the MLscAN clustering results obtained by using the "best" model. We see that the unsupervised clustering results are very close to the ground truth since the inferred states consist mainly of cells of the same type.

Having a model resulting in good clustering is vital before we continue to state-to-state trajectories inference.

4.7 Transitions and Trajectories

```
plotTransitions(res, #MLscAN object
                save = TRUE, #Save plot to file
                saveDir = ".", #Where to save?
                fileOnly = TRUE) #Save file, don't display
```

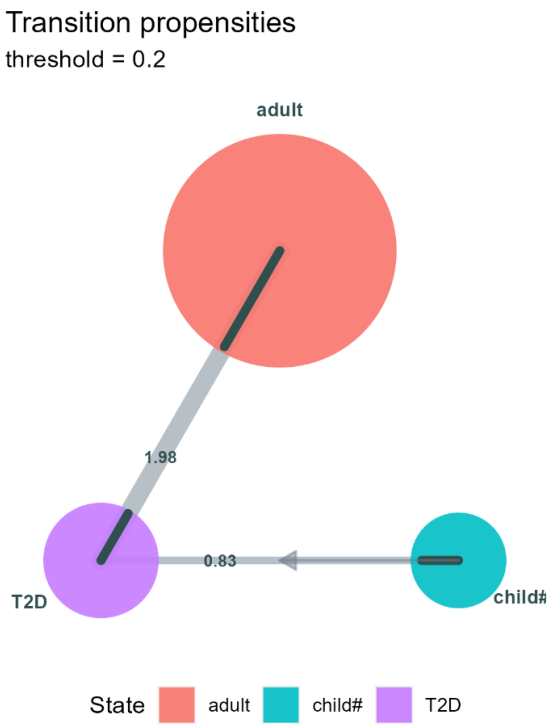

Figure 17 Transitions and their propensities

MLscAN not only infers cell states (major cellular phenotypes) in an unsupervised manner, but also infers state-to-state transitions suggested by the data and estimates their “strengths” (propensities) in an unbiased manner. For a specific transition to be depicted in this plot, its propensity should surpass a set threshold (default value = 0.2).

In the inferred transitions plot, each state is represented as a circle of radius increasing with the number of cells assigned to it (i.e., cells having the highest posterior probability for that state). A transition (grey line) exists between two states if there exist a set of cells whose two highest posterior probabilities (resulting from the GMM clustering) are in those two states. This is depicted by the edge connecting a pair of nodes in the plot. The edge’s weight (and line width) depends on the percentage of cells in the two states having this property. By adding those two percentages, the propensity value of the transition is calculated.

For example, say there are 3 states overall in a model: A, B, and C. Moreover, 90% of c state A cells have their second-highest posterior probability in state B, and 60% of the state B cells have their second-highest posterior probability in state A. In that case, the propensity of the transition will be 1.5 (=0.9 + 0.6). Obviously, the largest possible propensity value is 2. The edge line connecting the centers of the two circles representing states A and B is black at its extremes. The length of this black portion grows with the percentage of the cells of state A that have their second-highest posterior probability value in state B, the other state of the pair. If that percentage is, say, 50%, then half of the radius length of the circular node for state A is depicted as black.

In our case, almost all state “adult” cells have their second posterior to state “T2D” and vice versa. So the aligned radii of both state circles are darkened almost to their full length, indicating that the transition propensity value is very close to 2.

We also see a “unidirectional” transition (with arrow) from the “child” to the “T2D” state with a propensity 0.83. That means 83% of the child state cells have second-highest posterior to the T2D state, but no cell of T2D has a second posterior to the child state.

It is interesting that in our example, MLscAN could not detect a direct path from the child to the normal adults' state, but rather this forward evolution path passes through the T2D adults' state. This supports the theory that T2D may be a “remembered” state, crossed as cells evolve from the child to the adult state. According to this theory, the transition from adult to T2D state could possibly be considered as a later in life activation of a de-differentiation process along that remembered path [41]!

```
plotTrajectories(res,
                 save = TRUE,
                 saveDir = ".",
                 fileOnly = TRUE) #Save file, don’t display
```

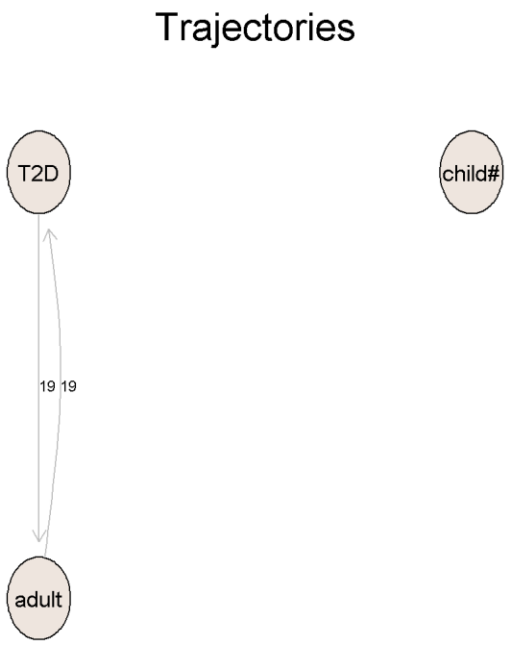

Figure 18 Trajectories between pairs of states and their key-genes

MLscAN imposes some limitations on which transitions can result in trajectories. For a transition A-to-B to be a trajectory, it should have at least 3 cells belonging to each of the

two states A, B. If a transition has cells that belong only to one state (is unidirectional) then it is not considered a valid trajectory. In our example, the transition child→T2D is unidirectional, as the arrow indicates in the transitions plot, and therefore it did not produce a trajectory.

```
plotMStates(res,
        mode="num", #X axis is absolute frequency of cells,not percentage
        save = TRUE,
        saveDir = ".",
        fileOnly = TRUE) #Save to file, don't display
```

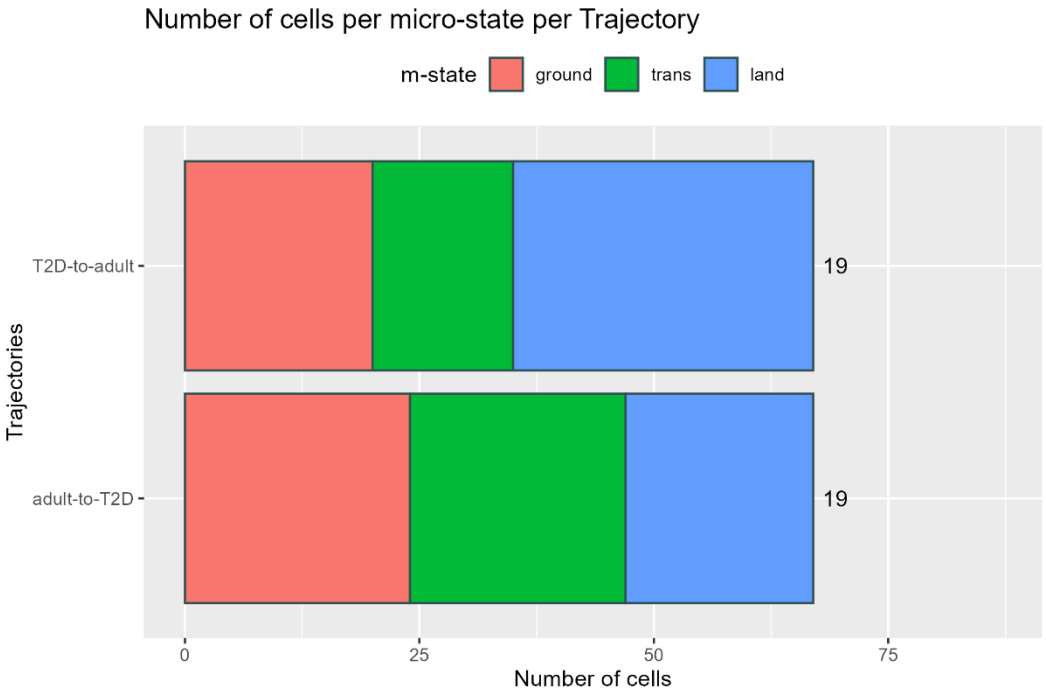

Figure 19 Trajectory micro-states

The bar plot above shows the proportion of cells that belong in each micro-state for each trajectory. Moreover, to the right of each bar is the numbers of key-genes identified for the corresponding trajectory. The two trajectories connecting the two states may have different micro-state boundaries and key-genes in general. For example, in the plot above, we see that the middle transitory micro-state (green) has more cells in the adult-to-T2D trajectory than in the opposite direction T2D-to-adult trajectory. Also, both trajectories have 19 key-genes, which may not be identical. Moreover, the inferred GRNs for the micro-states of the two trajectories may be very different since they model the regulation of different biological processes.

```
plotAlluvialState(res,
            save = TRUE,
            saveDir = ".",
            fileOnly = TRUE)
```

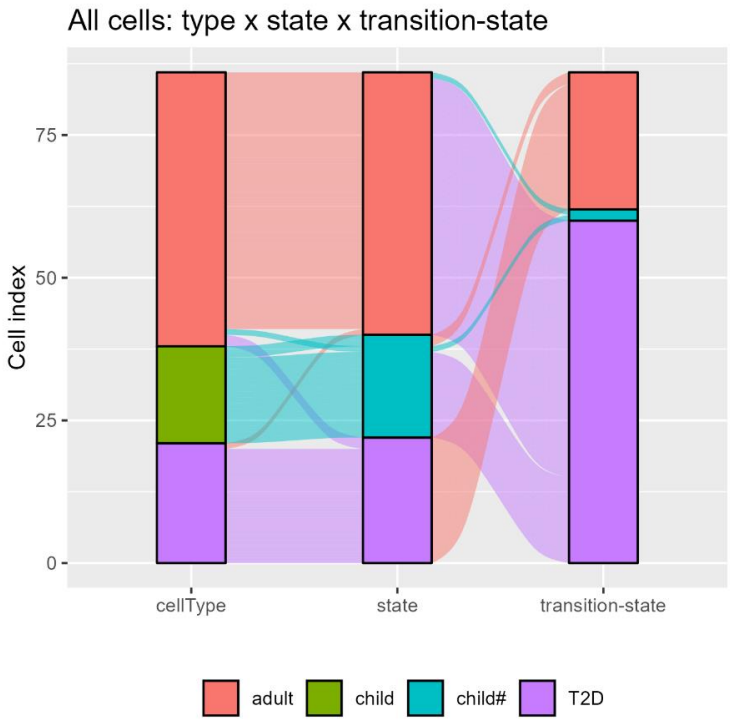

Figure 20 Alluvial plot relating states to cell types (ground truth) and next (transition) states

The alluvial plot above helps visualizing the correlation between the MLscAN inferred cell states (middle column) with the known cell-types (ground truth) on the left and the inferred transition (next) states on their right. We observe that the inferred states match very well to the cell types, indicating a near-perfect clustering. Moreover, each state has a strongly preferred transition (next) state, but more than one state may have a preference to the same transition state. For example, T2D is the most popular transition state having almost all cell of the other two states "looking towards" it. This is also evident visually by observing the black proportion of the radii of the adult and child state circles in the transitions plot; they are both large and "looking towards" the T2D state circle.

Moreover, we observe that a few adult cells have "child" as their transition state, but this transition is not depicted in the transitions plot. This is so because the percentage of these adult cells is very small, leading to a transition propensity lower than the set threshold (0.2).

4.8 Trajectories Analysis

```
plotProbTraj(res, #MLscAN object
        traj="adult-to-T2D", #Which trajectory
        save = TRUE, #Save plot as image
        saveDir=".", #Where to save?
        fileOnly = TRUE) #Save in file, don't display
```

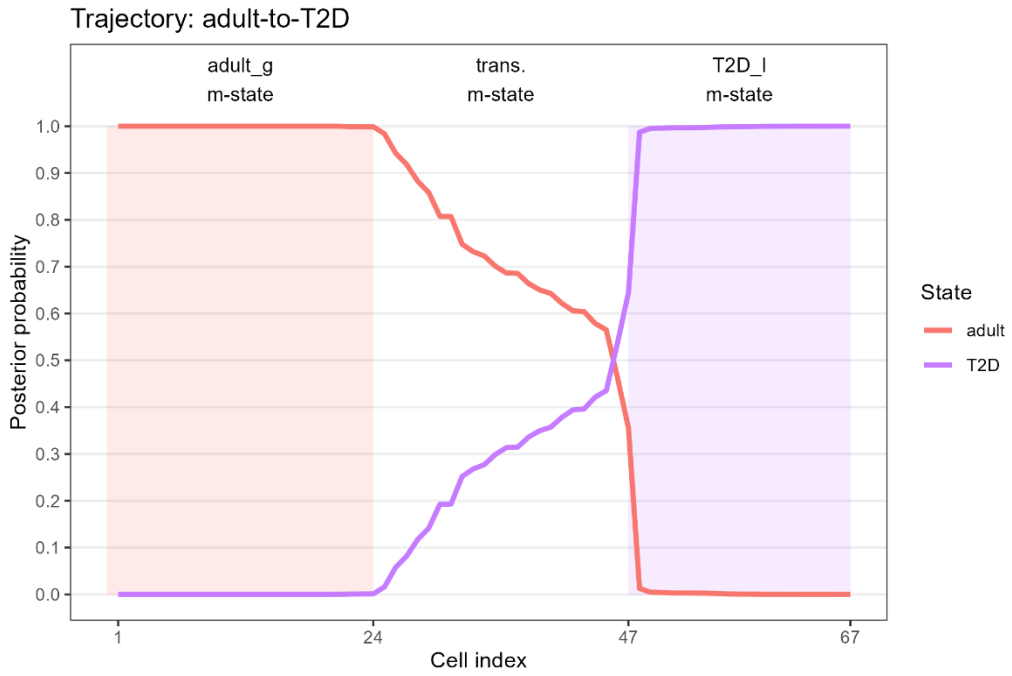

Figure 21 Posterior probabilities of trajectory cells

A set of cells belongs to a state-to-state trajectory, e.g., adult-to-T2D, if their two highest posterior probabilities are in those two states. As shown in the figure above, the 67 trajectory cells form a list organized in decreasing order of their posterior probability to the first or "departing" (or "ground") state (red curve) as we move from left to right. Moreover, MLscAN uses an algorithm to determine the posterior probabilities thresholds that partition a trajectory into consecutive micro-states, called "ground", "transitory", and "landing" m-states (going from left to right).

These micro-state regions are visualized on the plot above using color shading. We see that as cells transition from the adult ground m-state to the T2D landing m-state, their posterior to the adult state is strictly non-increasing (red curve) while their posterior to the T2D state is in most cases increasing (purple curve).

```
plotCircleTraj(res, #MLscAN object
        traj="adult-to-T2D", #Which trajectory
        save = TRUE, #Save plot as image
        saveDir=".", #Where to save?
        fileOnly = TRUE) #Save in file, don't display
```

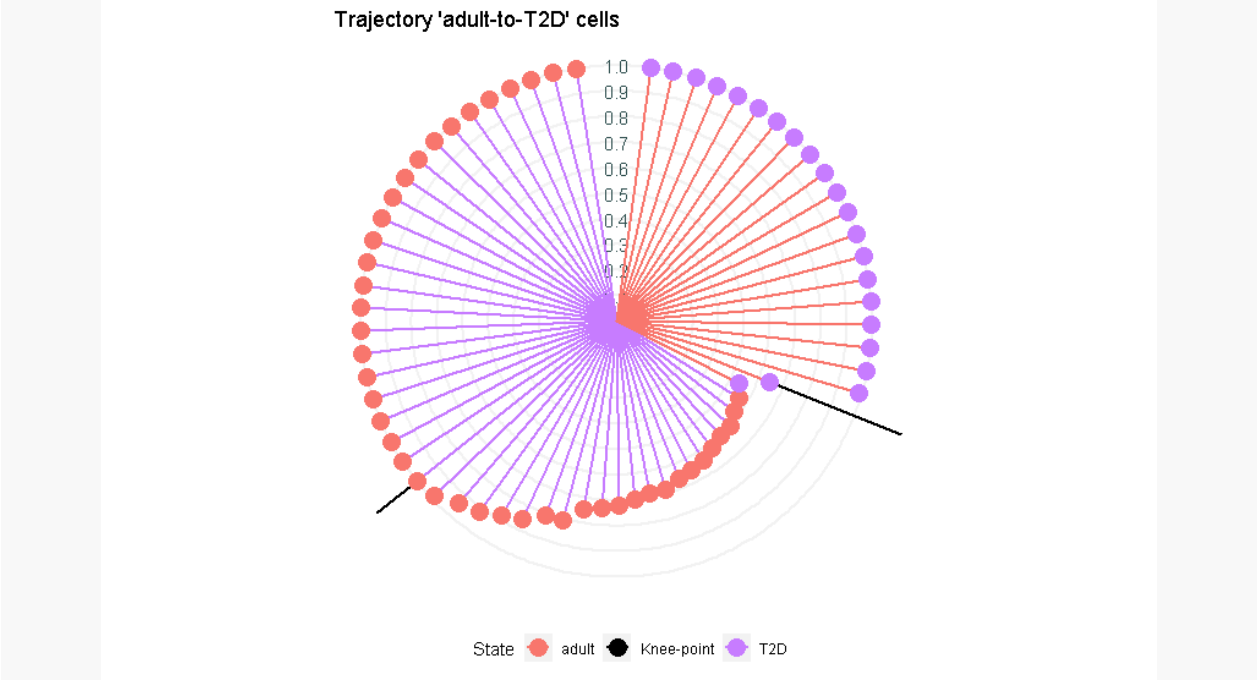

Figure 22 The circle trajectory plot displays information about the trajectory cells

The above circular plot is another way to visually capture the cells of a trajectory as an ordered list of cells in posterior probability space. Every cell is a node on the trajectory circle, colored according to its state (highest posterior) and connected to the circle's center by a line colored according to its transition state (2nd posterior). As we move on the circle counterclockwise, the posterior probability (radius length) of cells (nodes) with the same color to their state decreases.

The black radius extenders mark the trajectory knee points (micro-state boundaries. The first is the boundary between the ground and the transitory micro-states. The second marks the boundary between the transitory and the landing micro-states. We see that most cells belong to the ground and landing micro-states and are those cells used to determine via bimodality and differential expression tests the key-genes driving a state-to-state transition along an inferred trajectory.

```
plotViolinOverlayTraj(res,
                      traj="adult-to-T2D",
                      genes=keyGenes(res, traj="adult-to-T2D"),
                      save = TRUE,
                      saveDir = ".",
                      fileOnly = TRUE)
```

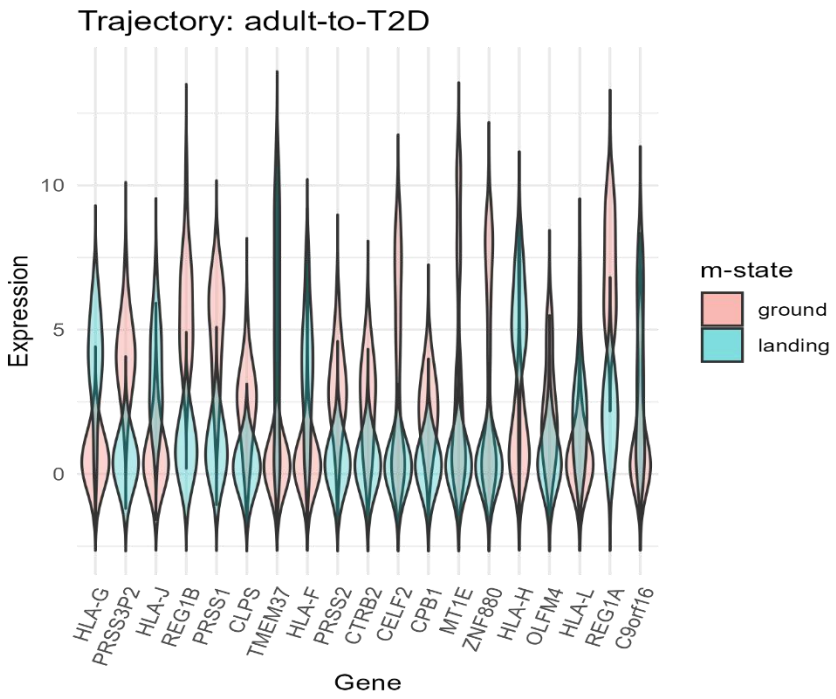

Figure 23 Bimodal gene expression of key-genes along the trajectory

The plot above shows expression violin plots for the 19 key-gene of the adult-to-T2D trajectory. A quick examination of the plot reveals that the key-genes have (a) bimodal expression and (b) mode-switching behavior along the trajectory. Namely, their expression is mostly high in the ground micro-state and mostly low in the landing micro-state, or vice versa. Therefore, the key-genes exhibit a switching behavior (High-to-Low, or Low-to-High) as cells move along the trajectory connecting two states.

```
plotBarExprTraj(res, #MLscAN object
        traj="adult-to-T2D", #Trajectory to show
        gene="HLA-G", #Gene to highlight
        save = TRUE, #Save plot to file
        saveDir = ".", #Where to save?
        fileOnly = TRUE) #Save in file, don't display
```

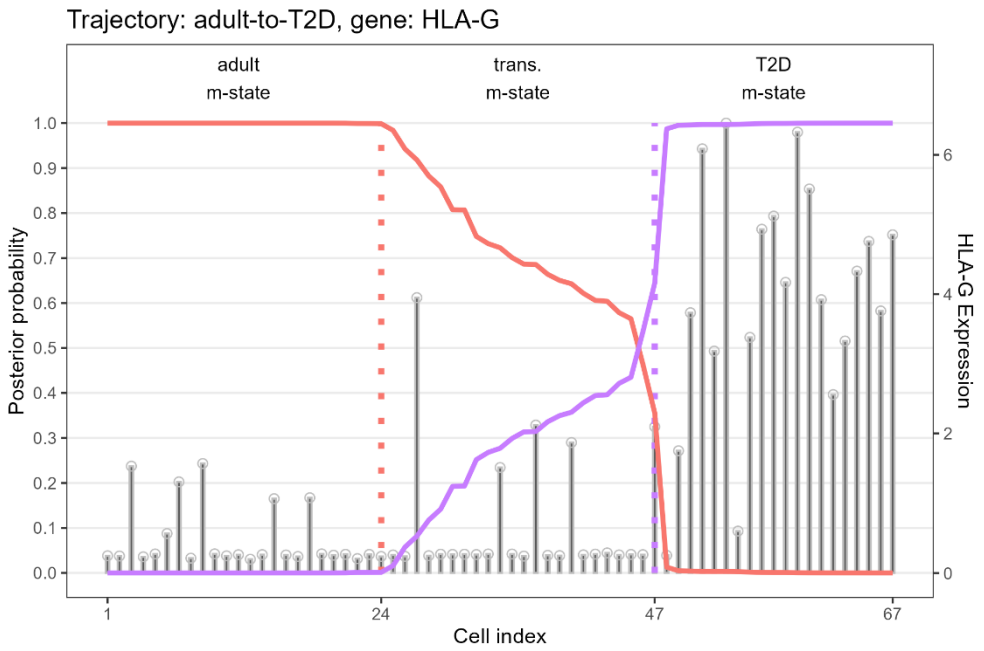

Figure 24 Gene expression switches modes along a trajectory

For example, looking at the behavior of gene HLA-G along the trajectory, we notice a stark difference in expression patterns between micro-states (separated, by vertical dotted lines). The adult ground m-state cells have strikingly lower gene expression than the T2D landing m-state cells. This is also reflected in part to the cells' posterior probabilities. Higher gene expression is correlated with a higher posterior probability to the landing state in this case. In summary, we observe a clear OFF-to-ON expression switching pattern of the HLA-G gene along the trajectory, which makes it a "key-gene".

```
plotHeatmapTraj(res, #MLscAN object
            traj="adult-to-T2D", #Trajectory to show
            save = TRUE,
            saveDir = ".",
            z_score = FALSE)
```

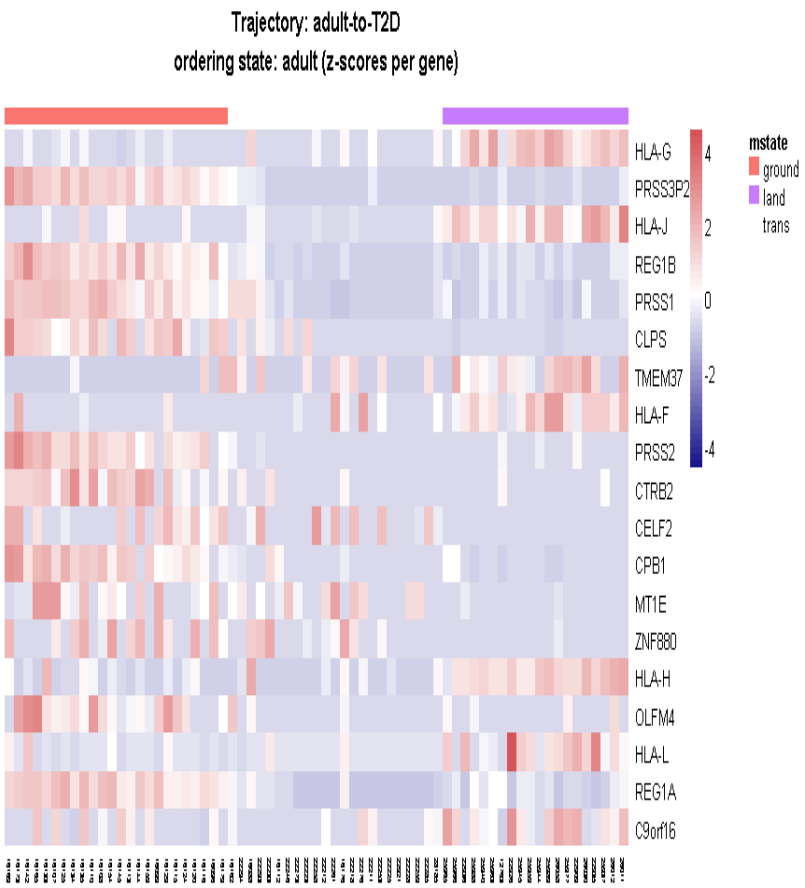

Figure 25 The heatmap of key-genes expression for the trajectory cells (z-scores)

Heatmaps also assist in the visualization of expression data patterns along trajectories. In the key-genes expression heatmap above, the cells are organized from left to right based on their relative position along the trajectory (pseudo-time), and the micro-state boundaries are marked. Once again, we can confirm the bimodality of gene expression and the switching of dominant mode as we move from the ground to the landing m-state, with some key-genes having more pronounced patterns than others.

While all trajectories and their micro-states can be examined and their GRNs plotted, the “adult-to-T2D” trajectory will be the only one presented here for brevity. MLscAN automatically provides a complete analysis and several plots for each inferred trajectory in the respective directory with no extra effort from the user. Users are encouraged to explore these plots thoroughly to glean valuable nuggets of information.

```
plotDotTraj(res, #MLscAN object
            traj="adult-to-T2D", #Trajectory to show
            genes= keyGenes(res, traj="adult-to-T2D"), #All key-genes
            save = TRUE, #Save plot to file
            saveDir = ".", #Where to save?
            fileOnly = TRUE) #Save file, don't display
```

Figure 26 Dot plot of key-genes expression for the trajectory cells at the different micro-states

The trajectory dot plot is a different way to summarize information about key-genes expression in the trajectory's micro-states. The bigger the size of the dot, the higher the mean expression of the key-gene in an m-state. Also, by looking at the dot's color, we can assess the key-gene's variability (standard deviation).

4.9 Gene Regulatory Networks

```
plotGRN(res, #MLscAN object
      traj=" adult-to-T2D", #Trajectory selected
      mstate="ground", #Which micro-state's GRN to plot?
      save = TRUE, #Save plot to file
      saveDir = ".", #Where to save?
      fileOnly = TRUE) #Output in only file
```

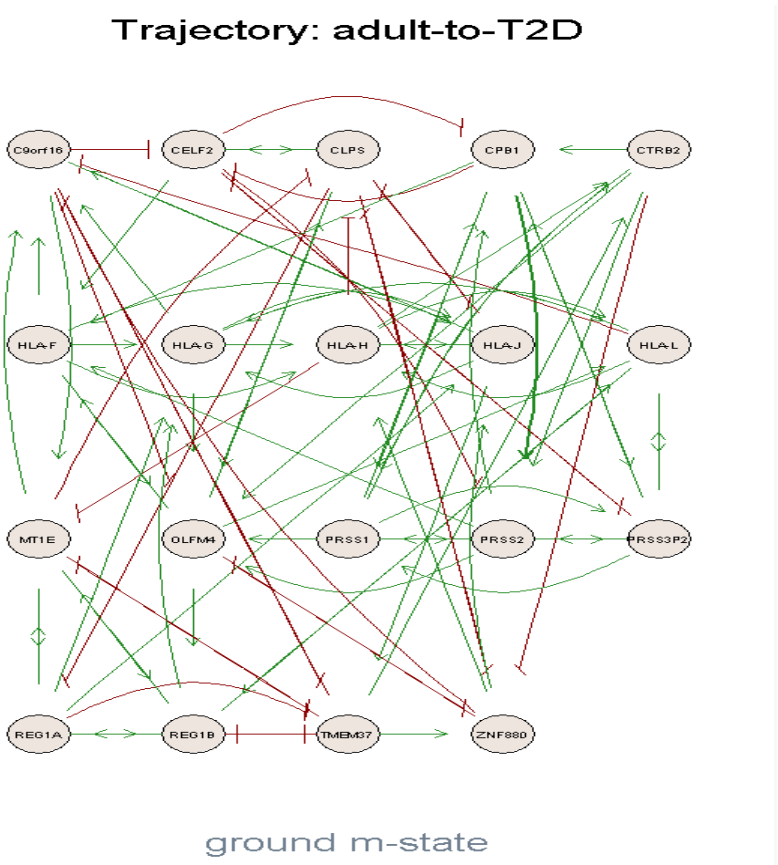

Figure 27 Inferred GRN. Green (red) edges indicate activation (inhibition)

The *plotGRN* function can be used to visualize any GRN, for any trajectory and micro-state of interest. MLscAN creates GRNs using default parameters and the *GENIE3* algorithm [44], but users may explore the space of available parameters if so inclined.

```
plotGRN(res, #MLscAN object
      traj="adult-to-T2D", #Trajectory to plot GRN for
      mstate="land", #Which micro-state's GRN to plot
      save = TRUE, #Save plot to file
      saveDir = ".", #Where to save?
      fileOnly = TRUE) #Save file, don't display
```

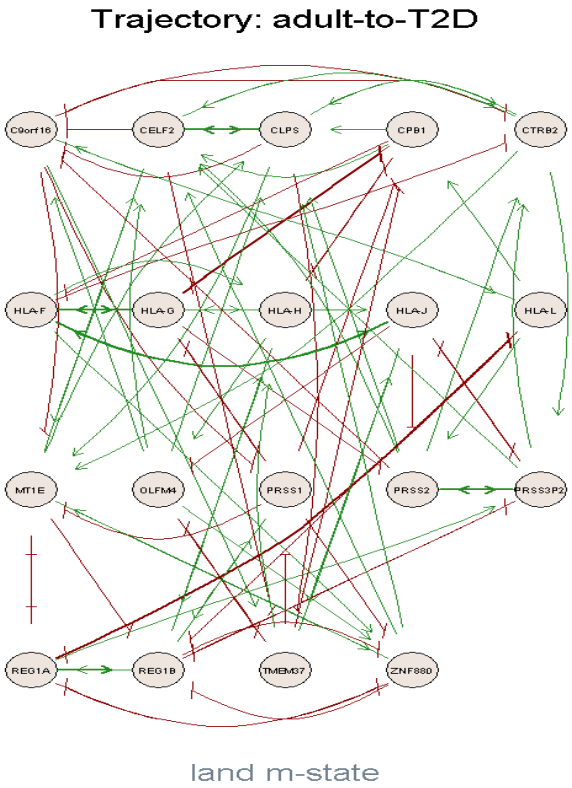

Figure 28 Constructing GRN for trajectory micro-states

The notion of micro-states was introduced in MLscAN because gene expression patterns may change dynamically along a trajectory (pseudo-time), and the mode of regulation of the same key-genes might be markedly different. This can be vitally important in many cellular processes and has been shown to occur very often in development, cancer, and many critical cellular functions [45].

In fact, a simple visual comparison reveals that the regulation mode of the same key-genes is different in the ground (early) and the landing (late) micro-states of the adult-to-T2D trajectory. MLscAN allows users to probe deeper and investigate differential regulation along the path of micro-states progression for any state-to-state trajectory of their interest and extract more insights from their datasets.

MLscAN integrates dimensionality reduction, unbiased unsupervised model selection, trajectories inference, key-genes identification, partitioning trajectories to micro-states, and GRNs inference down to the micro-state level, all in an easy-to-use pipeline and R package.

```
plotGRNHeatmap(res, #MLscAN object
        traj="adult-to-T2D", #Trajectory to plot
        mstate="ground", # Micro-state to plot
        save = TRUE, #Save plot to file
        saveDir = ".", #Where to save?
        fileOnly = TRUE) #Save in file, don't display
```

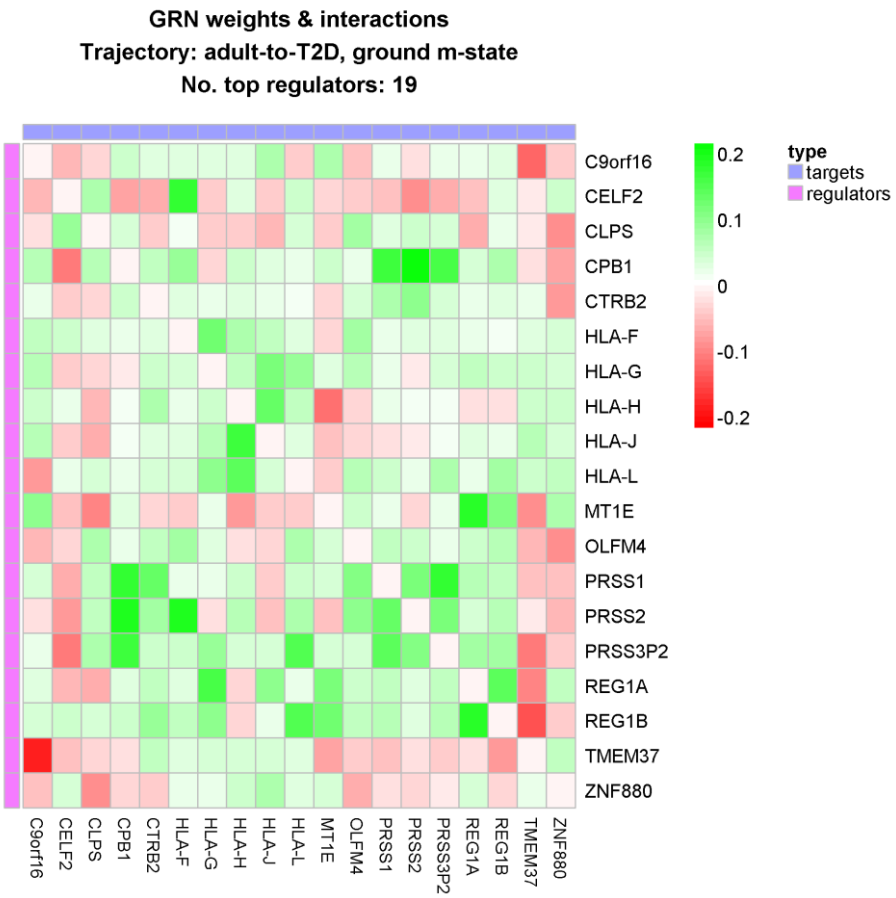

Figure 29 Visualizing the GRN weights between regulators and their targets

```
plotGRNHeatmap(res, #MLscAN object
        traj="adult-to-T2D", #Trajectory to plot
        mstate="ground", #Micro-state to plot
        save = TRUE, #Save plot to file
        saveDir = ".", #Where to save?
        fileOnly = TRUE) #Save file, don't display
```

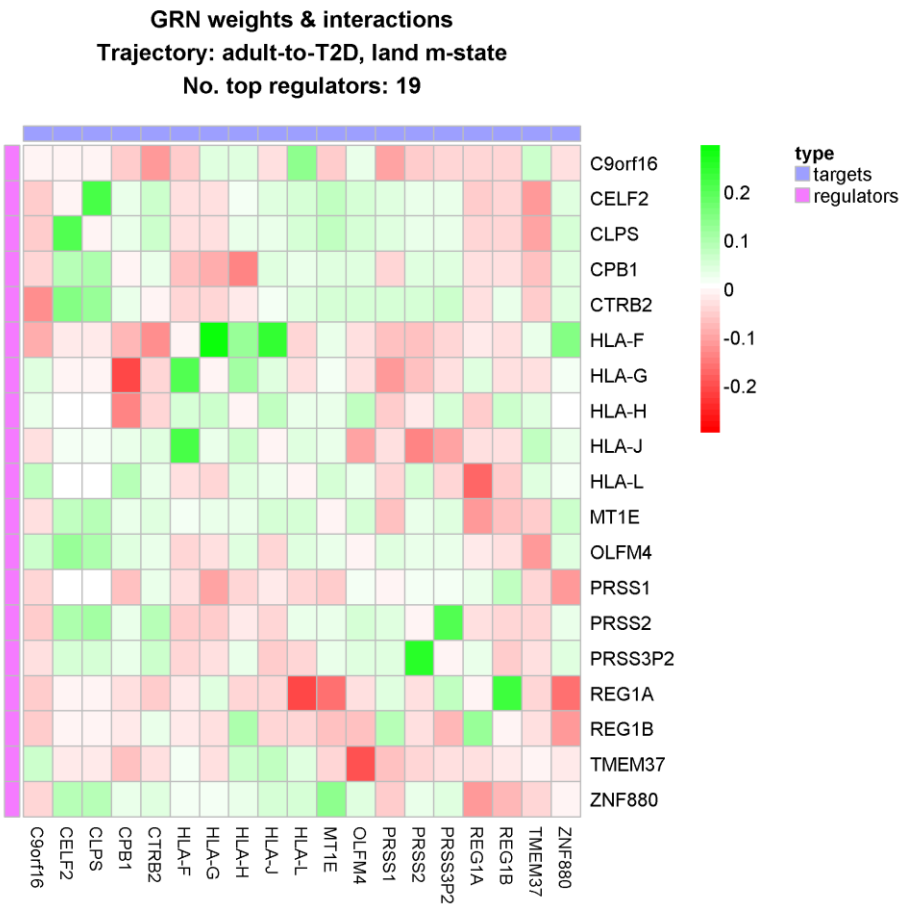

Figure 30 Visualizing the GRN weights between regulators and their targets

Large GRN plots may be messy, but the above function allows plotting the inferred GRN weights between the key-genes in matrix format, revealing hidden regulation trends between pairs of genes. For example, for any particular gene target (column), we can check its regulators (rows), see if they are positive (excitatory) or negative (inhibitory), and how strong their influence is on the target gene. Users can also quickly obtain the weight values themselves by using *grnWeights* for any trajectory and micro-state of interest. Larger weights correspond to more strongly supported by the data regulatory relationships [44].

Using the weights produced by GENIE3 [44], MLscAN creates for each target gene a list of its top regulator key-genes ordered by the absolute value of their weights. The user may define how many of the top regulators she wants to present for each target gene. The rest of the regulators are colored grey.

Of course, the inferred GRNs may differ significantly between the ground and landing micro-states of the same trajectory, as is the case in this example.

4.10 Mixed States

You may have observed that the “child#” state contains a “#” at the end of its name. This is happening because MLscAN has flagged this state as a potential “Mixed State”. Mixed states are sets of cells with an unusually large variance relative to other inferred states. Their existence may indicate the existence of interesting cell subpopulations or the existence of outlier cells and may cause problems in downstream trajectories analysis. We will see examples of problematic Mixed States and how MLscAN can be used to deal with them in Chapter 6. However, the child# Mixed state is harmless in this example as it consists mainly of one cell type.

In conclusion, this overview demonstrated that MLscAN integrates dimensionality reduction, unbiased and unsupervised model selection, trajectories inference, key-genes identification, partitioning of trajectories into micro-states, and GRNs inference down to the micro-state level, all in an easy-to-use flexible computational pipeline. Moreover, it provides excellent visuals to interpret the results of every pipeline stage.

5 USE CASE - DIMENSIONALITY REDUCTION USING UMAP

The purpose of this use case is to demonstrate the flexibility MLscAN provides to its users who may want to explore using alternative dimensionality reduction methods not supported by the package. For example, this need may arise if they seek to improve the cell clustering if the results of PCA [40] a linear method, cannot properly represent the dataset.

For this use case we will use the cell cycle Buettner dataset [46] containing mouse embryonic cells in 3 different cell cycle stages (G1, S, G2M) identified using flow cytometry. The dataset contains 264 cells and 6812 genes.

We will begin by trying a default MLscAN run.

5.1 Using an alternative dimensionality reduction method

In the run shown below the following MLscAN parameters assumed default values:

modelNumStates: Denotes the range of the number of states we want MLscAN to consider in model selection. The default range is [2:9].

modelStatesSelFun: Function used to determine the number of states of the “best” model. Default value: δBIC function, alternative build-in option: maxBIC function.

modelModelNames: These are the types of covariance matrix structures of the GMM components we want MLscAN to consider in the search for the “best” model that fits the dimensionality reduced data. By default, MLscAN will try all 14 model types available.

dimRedMethod: Denotes the PCA method to be used by MLscAN. If the dataset is not small, having more than 100 genes or cells, MLscAN by default uses the *irlba* [27] SVD method with 50 dimensions.

dimRedTopN: It is the number of the most variable genes to be used in PCA. The default value is 1000.

```
#MLscAN run
cell_cycle_run <- MLscAN(exprData=expressD, #Expression Matrix


        MLscANCellFeatures=cellFeat, #Cell Features used in plots
        MLscANColors=coloring, #Defined colors for specific cell features
        MLscANStopAt="model", #Stop the analysis at model selection, do not
                produce trajectories
        MLscANOutMode="no", #Do not produce output files
        modelStateNameMode = "mostFreqPerState" #an inferred state is named
                 according to the most represented (70% or more) cell type.
)
##
## Creating the MLscAN object...
## Performing dimensionality reduction...
## Creating the model...
```

```
## Forming the sub-populations...
## Possible mixed state(s): 1
```

MLscAN selected the GMM model name VEI with 3 states (components) as the “best” model. First, let’s visualize the composition of the dimensionality reduced data.

```
plotDimRed(golden_mlscan,feature="cellType")
```

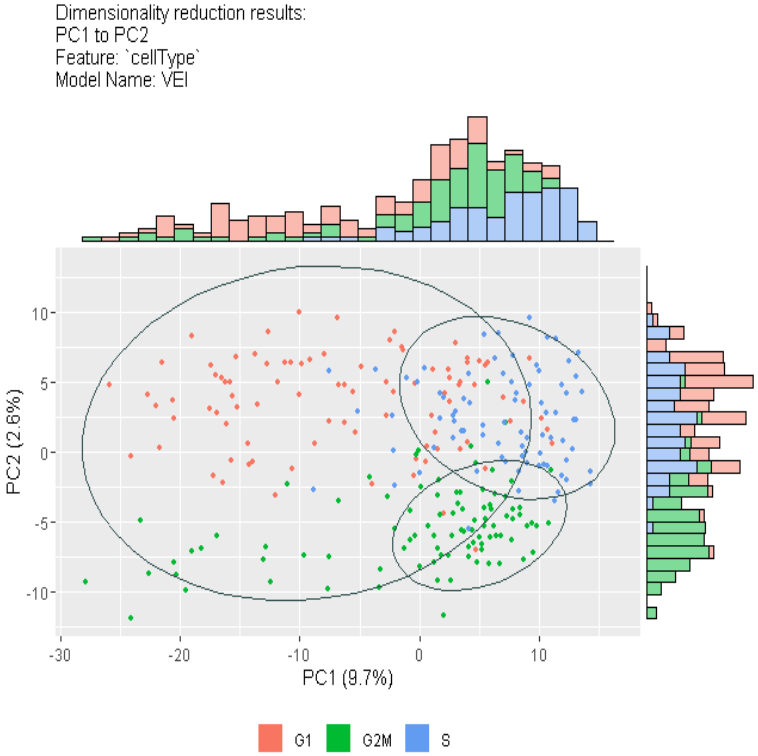

Figure 31 Inspect the PC1 vs. PC2 results with cells colored by cell type (ground truth)

```
plotStatesComposition(golden_mlscan,feature="cellType")
```

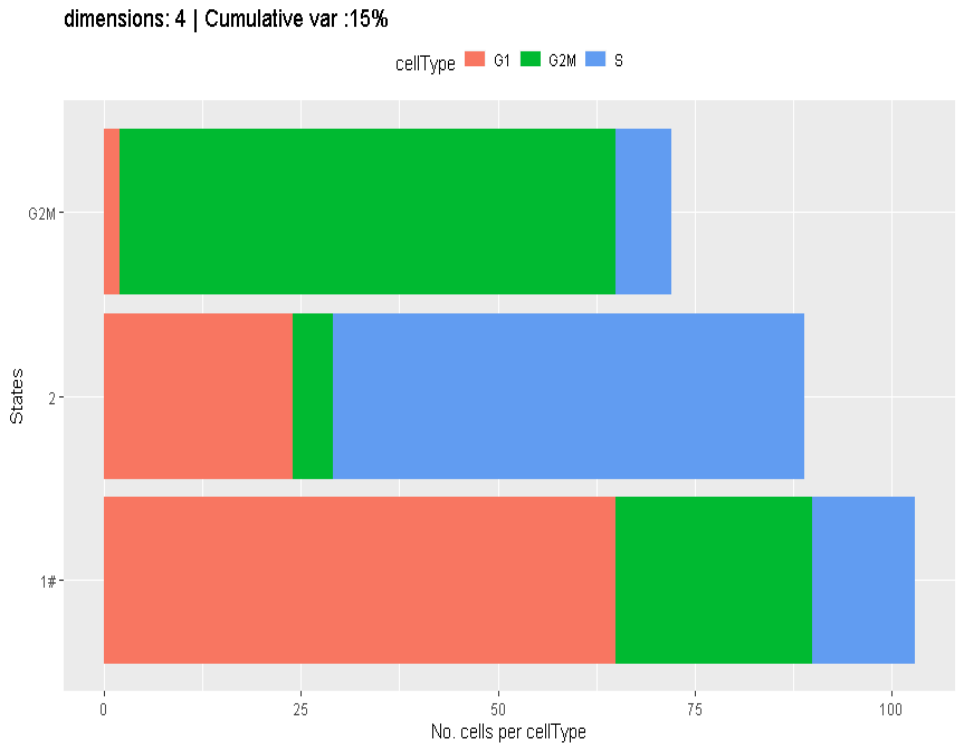

Figure 32 Inferred states Composition plots colored by cell type (cell cycle stage)

The results show that clustering is far from optimal. Although the best model has as many states as the cell types, when examining the PCA plot, we notice that it is hard to distinguish the cell cycle stages because they are mixed.

In cases like this, PCA may not be appropriate for representing our dataset in low dimensions. Therefore, one way to improve the clustering results is to try an alternative dimensionality reduction technique. MLscAN allows to do that outside the package, using any method of interest, and import the results as input in the MLscAN run. In this specific case, we will use UMAP [47] with ten dimensions.

External UMAP run

First, we will use the UMAP package [16] to obtain the dimensionality reduction matrix, and then we will use the dimRedData argument to import this matrix to the next MLscAN run

```
cell_cycle_UMAP <- umap::umap(expressD,n_components=10)
```

For this run MLscAN uses by default δBIC criterion for model selection and use all model Names available for its model exploration.

```
cell_cycle_UMAP_mlscan<- MLscAN(exprData=expressD, #Expression Matrix
    MLscANCellFeatures=cellFeat, #Cell features vector
    MLscANStopAt="model", #Analysis stop at model selection, no Trajectories
```

```
produced
MLscANOutMode="no", #Do not produce any output file
MLscANColors=coloring, #Use specific colors for the cell features
dimRedData=cell_cycle_UMAP$layout, #Use UMAP dimensionality reduction results
modelStateNameMode = "mostFreqPerState", #Naming method to use for the states
kgGenesSelFun= kg_voting() #Use voting to determine the key genes for the inferred state-to-state trajectories
 )
```

```
##
## Creating the MLscAN object...
## Performing dimensionality reduction...
## Creating the model...
## Forming the sub-populations...
## Creating the trajectories...
```

```
plotBIC(cell_cycle_UMAP_mlscan,showModelNames = TRUE)
```

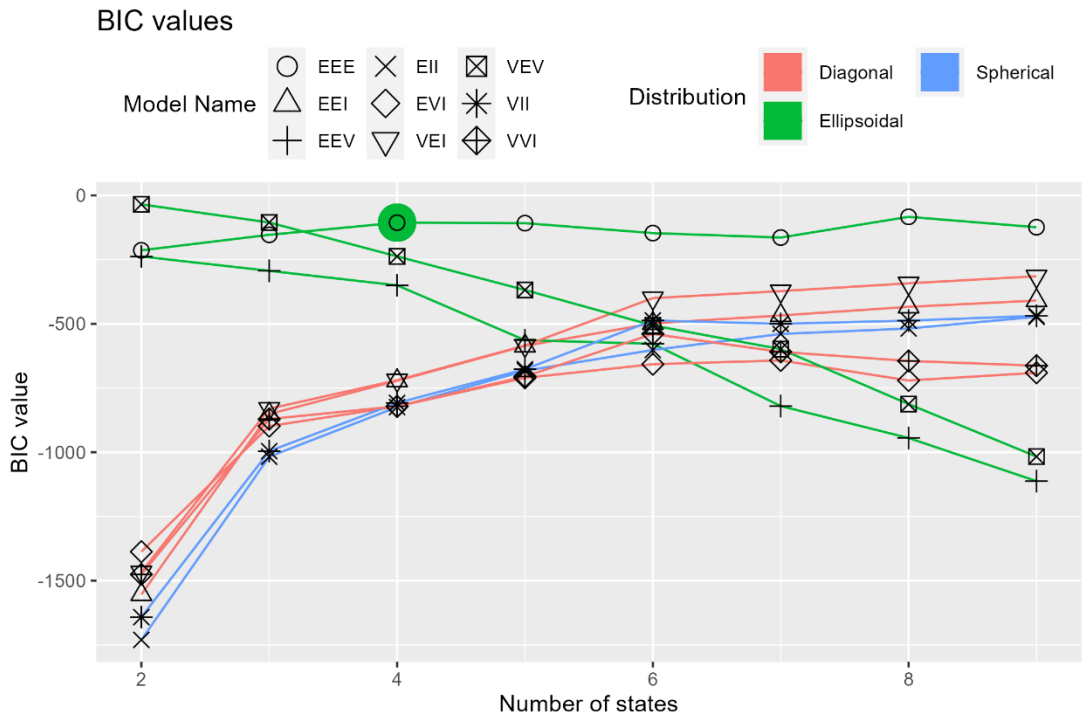

Figure 33 BIC values of different models considered

The above plot provides a visualization of the BIC values versus the number of states for every GMM model name (covariance matrix type) considered by MLscAN in model selection. It also reminds us of the covariance matrix structure of each model using a specific color. MLscAN selected as “best” the EEE model name with four states using the

default parsimonious δBIC criterion. This combination also happens to be very close to the maximum BIC value in this example.

```
plotDimRed(cell_cycle_UMAP_mlscan,feature="cellType")
```

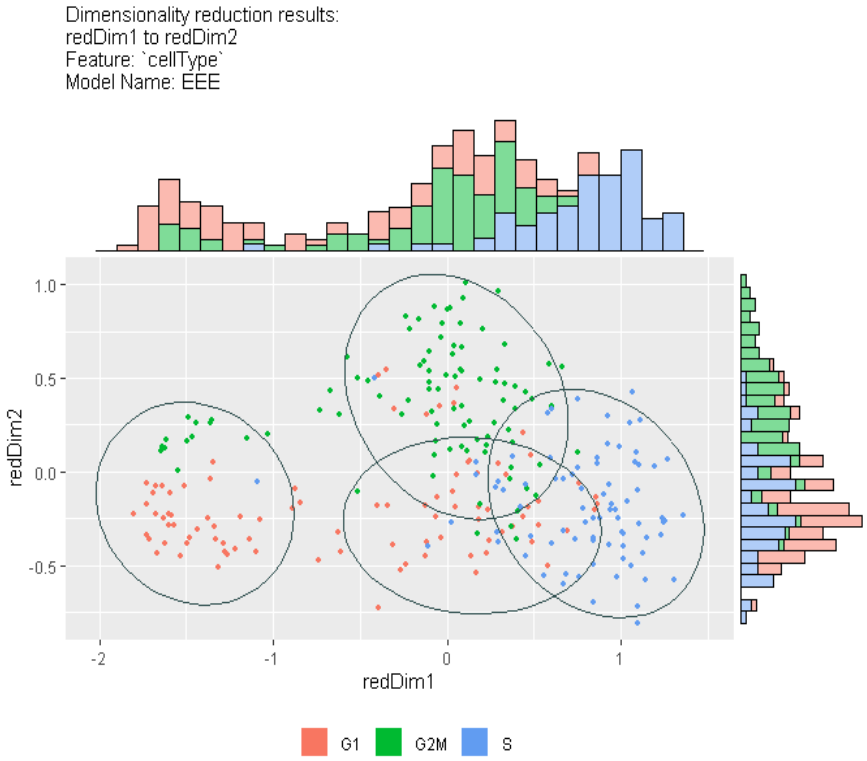

Figure 34 UMAP dim1 vs. UMAP dim2 with cells colored by cell type (ground truth)

We observe that the inferred cell states are less mixed at UMAP dimensions 1 and 2. So we expect a better clustering result.

```
plotStatesComposition(cell_cycle_UMAP_mlscan,feature="cellType)
```

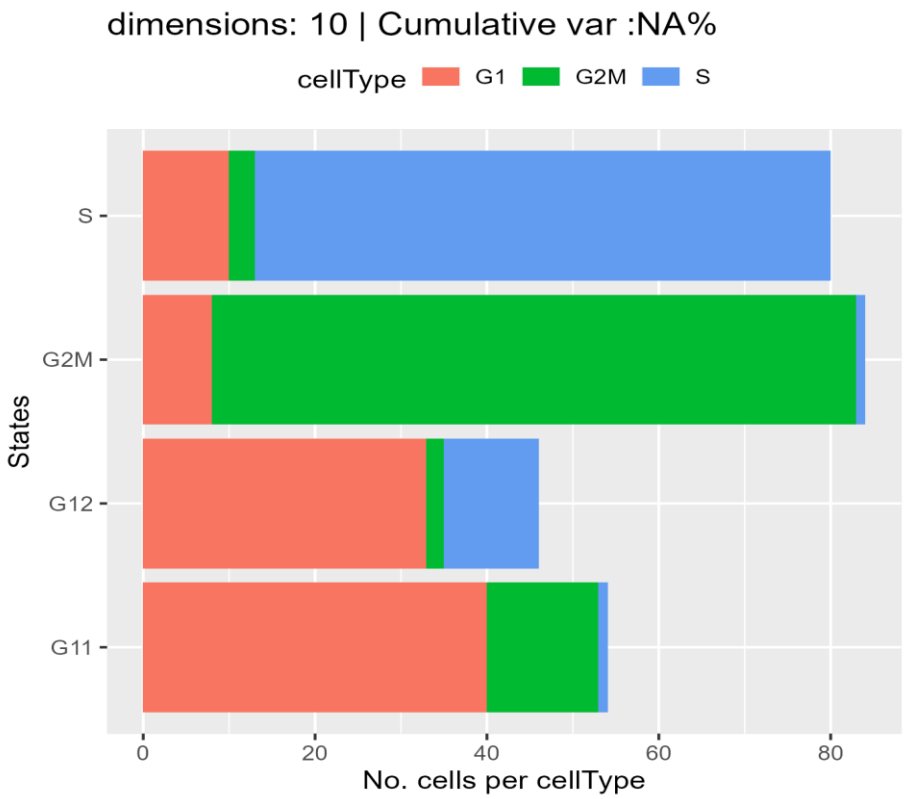

Figure 35 States Composition plots colored using cell type (cell cycle stage)

MLscAN with default parameters detects four states when using UMAP dimensionality reduced data as input. The unsupervised parsimonious model selection suggests the existence of two sub-clusters for cell stage G1. All inferred states consist of cells of one cell type (cell stage) by at least 70%. Even though the states are not entirely “pure”, we can see that importing the UMAP results to MLscAN improved the clustering results for this specific dataset.

plotTransitions*(cell_cycle_UMAP_mlscan)*

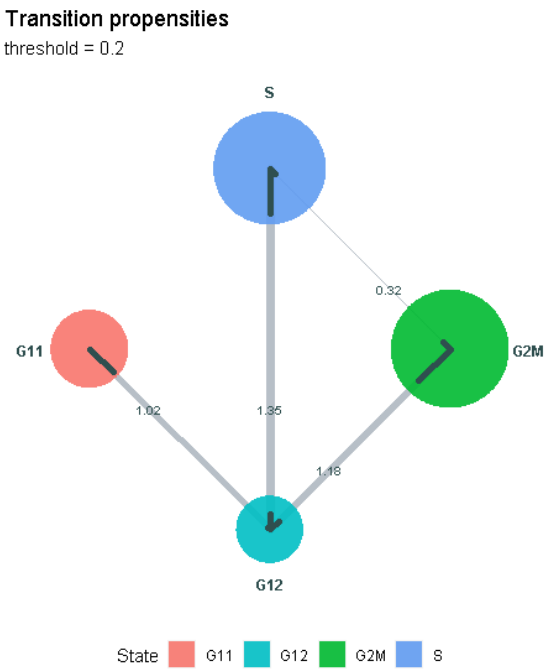

Figure 36 Inferred state transitions

The above plot shows the state transitions with propensities that surpass the default threshold (0.2). We observe the expected cell cycle relations between G12, S, and G2M. We also see that almost all cells of G11 are “looking towards” G12 as their transition state.

plotTrajectories*(cell_cycle_UMAP_mlscan)*

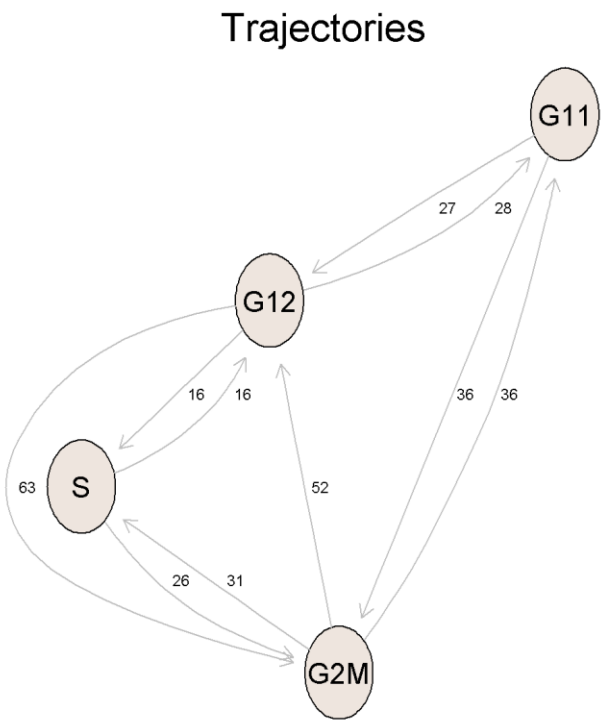

Figure 37 Inferred trajectories and their key-genes

In the trajectories plot, we can see the triangle with vertices G12, S, and G2M also observed in plotTransitions. However, we also see an interaction between G11 and G2M that is missing in the plot of transitions. This is so because it is a low propensity transition. In an attempt to extract new knowledge, MLscAN analyses all possible state-to-state trajectories, even those that the data support weakly as possible state transitions. The number next to a trajectory arrow is the number of key-genes that are driving the trajectory. MLscAN used voting here to detect key-genes, i.e., it has applied all five differential expression methods it supports, and then took the majority.

5.2 Post MLscAN run analysis – Gene cluster markers

MLscAN also provides functions allowing the user to explore more the MLscAN's run results.

FindClustersMarkers() is an external MLscAN function, which provides the user with the markers of each state compared to the other indicated states (default is all states). Also, the plotDimRed function allows the user to color the cells based on a selected gene's expression (e.g., a marker gene).

plotDimRed*(cell_cycle_UMAP_mlscan)*

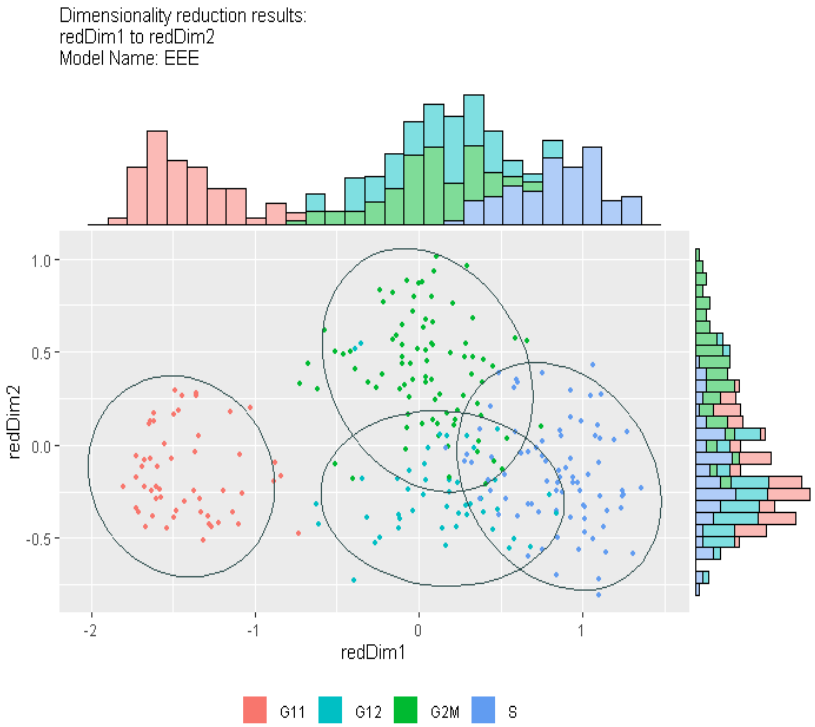

Figure 38 UMAP dim1 vs. UMAP dim2 with cells colored by cell states

The plot above presents a dimensionality reduction plot colored by cell states. In the three dimensionality reduction plots shown below, cells are colored based on the expression of one of the state markers with respect to the S state. We have isolated the respected pairs of states in three plots to make the expression differences more visible.

*Gene_markers <- FindClusterMarkers(*cell_cycle_UMAP_mlscan,states=c("G11", S"*))*

plotDimRed*(cell_cycle_UMAP_mlscan,gene="Iah1",selected_states=c("G11","S"))*

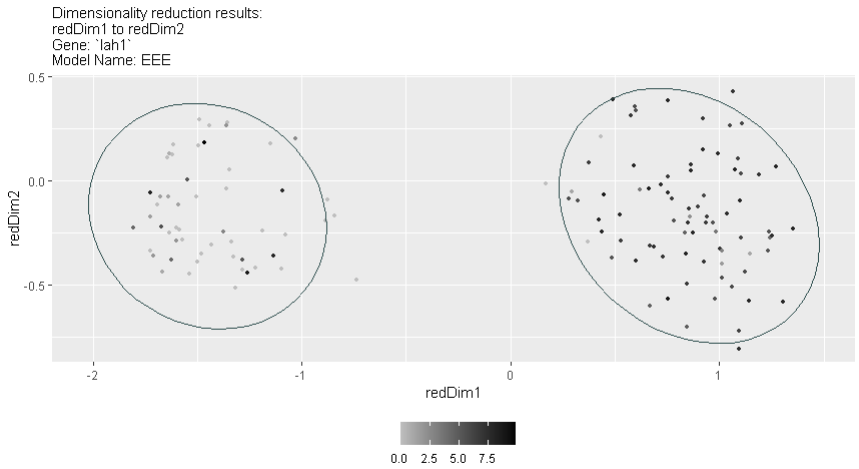

Figure 39 UMAP dim1 vs. UMAP dim2 with cells colored by gene Iah1 [G11 to S Marker] expression

*Gene_markers <- FindClusterMarkers(*cell_cycle_UMAP_mlscan,states=c(“G2M”, S”)*)*

plotDimRed*(cell_cycle_UMAP_mlscan,gene="Enpp3"",selected_states=c(“G2M”,”S”))*

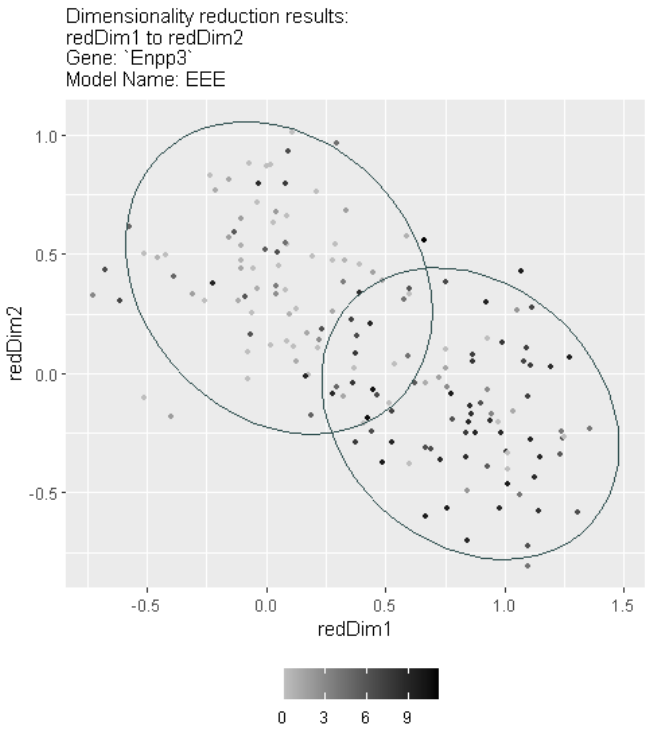

Figure 40 UMAP dim1 vs. UMAP dim2 with cells colored by gene Enpp3 [G2M to S Marker] expression

*Gene_markers <- FindClusterMarkers(*cell_cycle_UMAP_mlscan,states=c(“G12”, S”)*)*

```
plotDimRed(cell_cycle_UMAP_mlscan,gene="Stk17b",selected_states=c("G12","S"))
```

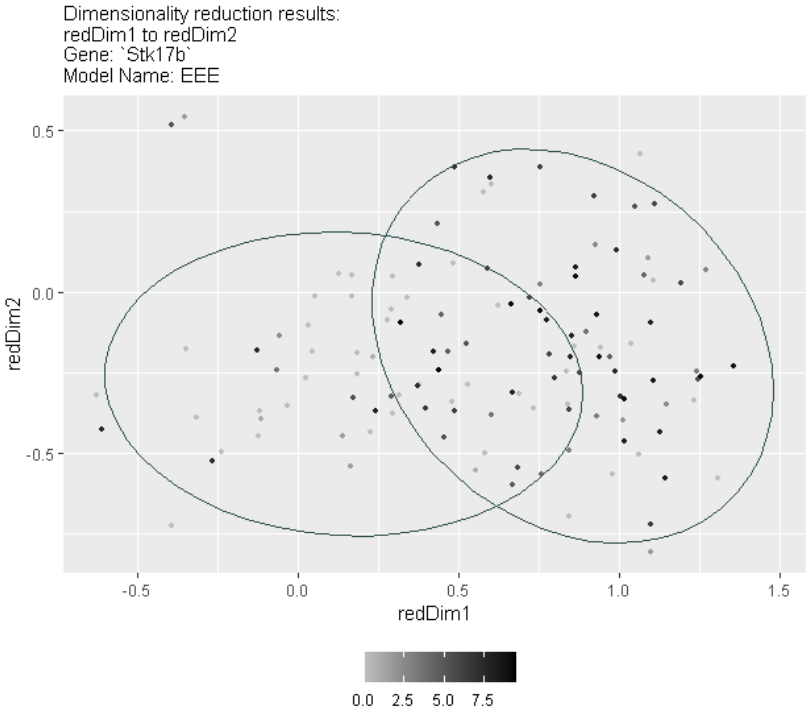

Figure 41 UMAP dim1 vs. UMAP dim2 with cells colored by gene Stk17b [G12 to S Marker] expression

6 MLSCAN MIXED STATE ISSUE

6.1 MLscAN Mixed State issue approach: Mixed State analysis

We will present here an approach for dealing with the Mixed State issue. For this purpose, we will use the Hayashi 2018 dataset [48]with scRNA-seq data derived from mouse Embryonic Stem Cells (mESC) collected at 0, 12, 24, 48, and 72 hours after the induction of cell differentiation into Primitive Endoderm cells. We have used the same dataset before in [9]to demonstrate how exploiting MLscAN's flexible model selection capabilities leads to a representative clustering approximating the ground truth faithfully. Since here the focus is on mixed states handling, we will not repeat the steps performed in [9]but rather build on top of that analysis.

6.1.1 MLscAN Mixed State analysis

The run performed in [9] that we will complement here used for unsupervised GMM-based model selection the default range of 2 to 9 states and the default δBIC parsimonious modeling approach._The rest of the arguments were set as follows:

```
MLscAN_Obj <- MLscAN(exprData= expressD,#Expression Matrix
                     MLscANCellFeatures= cellFeat,#Cell Features
                     modelStateNameMode= "mostFreqPerState",#Naming method for inferred states
                     dimRedMethod = "prcomp",#PCA method to use
                     dimRedTopN= 1000,#number of most variable genes to use in PCA
                     modelModelNames= "Diagonal",#Consider only models with diagonal covariance strutctures (i.e., EEI,VEI,EVI,VVI)in model selection
                     MLscANStopAt= "model",#Analysis stop at model, do not produce trajectories
                     MLscANOutMode= "no",#Do not produce any output files
                     MLscANColors = coloring #Use provided colors for the cell features
)
plotDimRed(mixed_state_mlscan,feature="cellType")
```

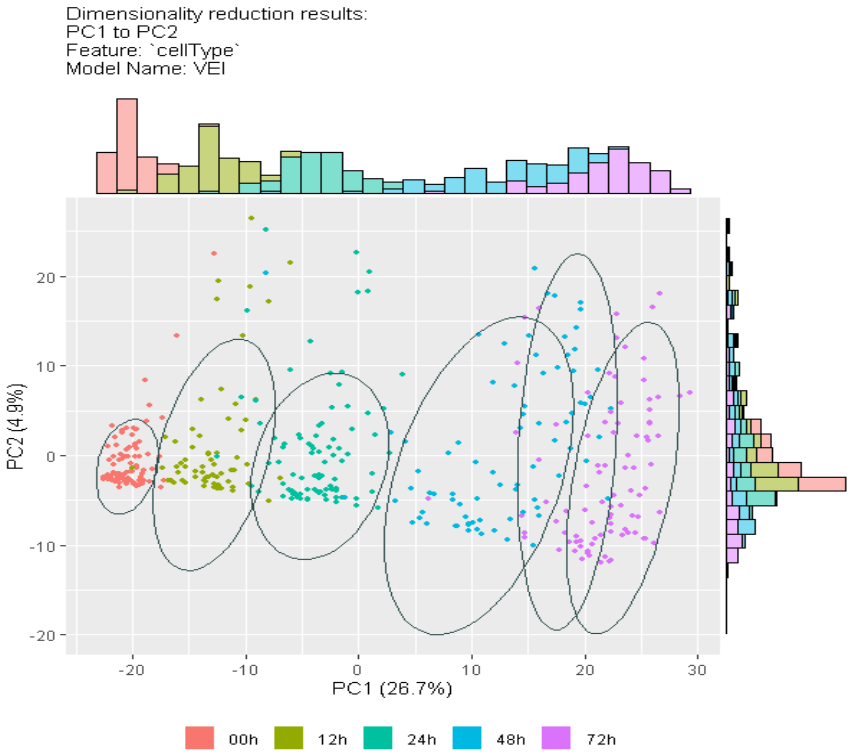

Figure 42 Inspect the PC1 vs. PC2 results with cells colored by cell type (ground truth)

```
plotStatesComposition(MLscAN_Obj,feature="cellType")
```

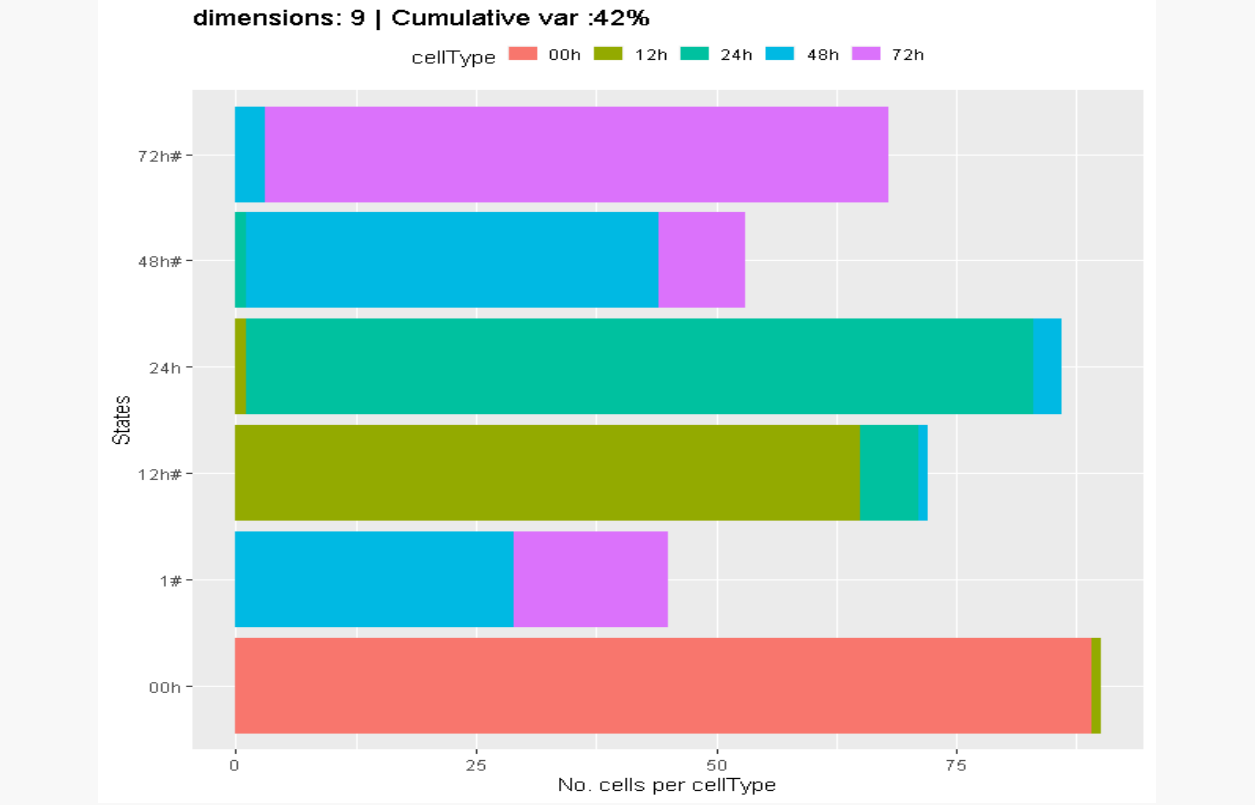

Figure 43 States Composition plots colored using cell type

The above dimensionality reduction and states composition plots give us a clear picture of the clustering results. We can see four Mixed States, named 1#, 72h#, 12h#, 48h# (names ending with a # character). The last three consist of cells of one cell type (cell stage) by at least 70%, so we will not consider them further in the analysis. In contrast, 1# is a large variance state consisting of a mixture of 48h and 72h type cells, suggesting the possible existence of sub-populations. MLscAN can help us investigate this possibility by analyzing further this particular mixed state cells.

6.1.2 Mixed state analysis run

We need to run MLscAN analysis using only the 1# mixed state cells to examine the sub-population hypothesis. To create and analyze the mixed state's dataset in isolation, we need to apply the following steps in R.

1. Isolate the cells of the identified mixed state
2. Obtain their expression data
3. Remove the genes that have only zero expression values from the new expression data matrix
4. Obtain the cell features data for mixed states' specific cells.
5. Run MLscAN with those inputs.

These steps are executed by the code shown below:

```
MLscANObj <- MLscAN_Obj # Name of MLscAN object
state <- "1#" # name of Mixed State

# Isolate the cells of the mixed state
state_cells <- stateCells(MLscANObj,state=state)[[1]]

#Obtain the expression data for specific cells
state_exprData <- exprData(MLscANObj,cells = state_cells)

# Remove the genes that have only 0 expression values from the new expression data matrix
state_new_expr_data <- state_exprData[,colSums(state_exprData)!=0]

# Obtain the dimensionality Reduction Data for specific cells
state_dimRedData <- dimRedData(object = MLscANObj,cells = state_cells)

# Obtain the cell Features Data for specific cells
state_CellFeatures <- cellFeatures(object = MLscANObj, cells = state_cells)
```

The next step is to use MLscAN to analyze the mixed state cells in isolation using the default δBIC method for GMM-based model selection and considering the default range of 2 to 9 states.

```
# MLscAN run for the cells of Mixed State
mixed_state_mlscan <- MLscAN(exprData= state_new_expr_data,#Expression Matrix
                             MLscANCellFeatures= state_CellFeatures,#Cell Features
                             modelStateNameMode= "mostFreqPerState",#Naming method for inferred states
                             dimRedMethod = "prcomp",#The PCA method to use
                             dimRedTopN= 1000,#Number of most variable genes to use in PCA
                             modelModelNames= modelModelNames(MLscANObj),#Use the same model name (VEI) as in the initial run
                             MLscANStopAt= "model",#Stop the analysis at the model creation, do not produce trajectories
                             MLscANOutMode= "no",#Do not produce any output files
                             MLscANColors = coloring #Use specific colors selected for certain cell features
)
```

```
##
## Creating the MLscAN object...
```

```
## Performing dimensionality reduction...
## Creating the model...
## Forming the sub-populations...
## Possible mixed state(s): 48h2
```

```
plotDimRed(mixed_state_mlscan,feature="cellType")
```

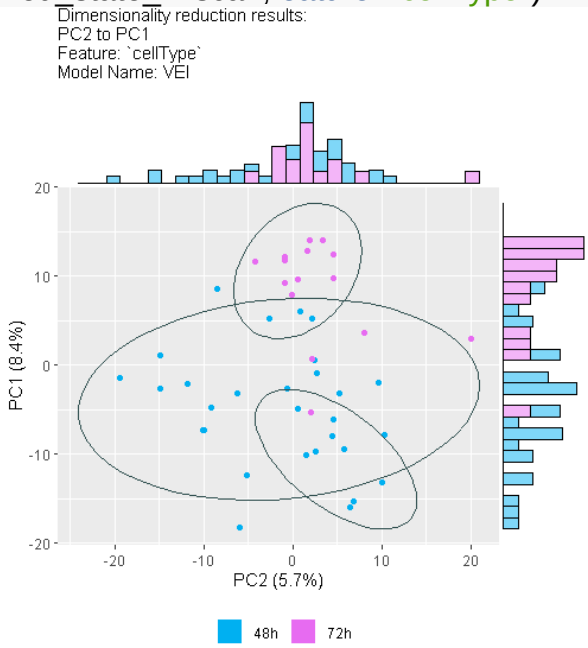

Figure 44 Inspect the PC1 vs. PC2 results with cells colored by cell type (ground truth)

```
plotStatesComposition(mixed_state_mlscan,feature="cellType")
```

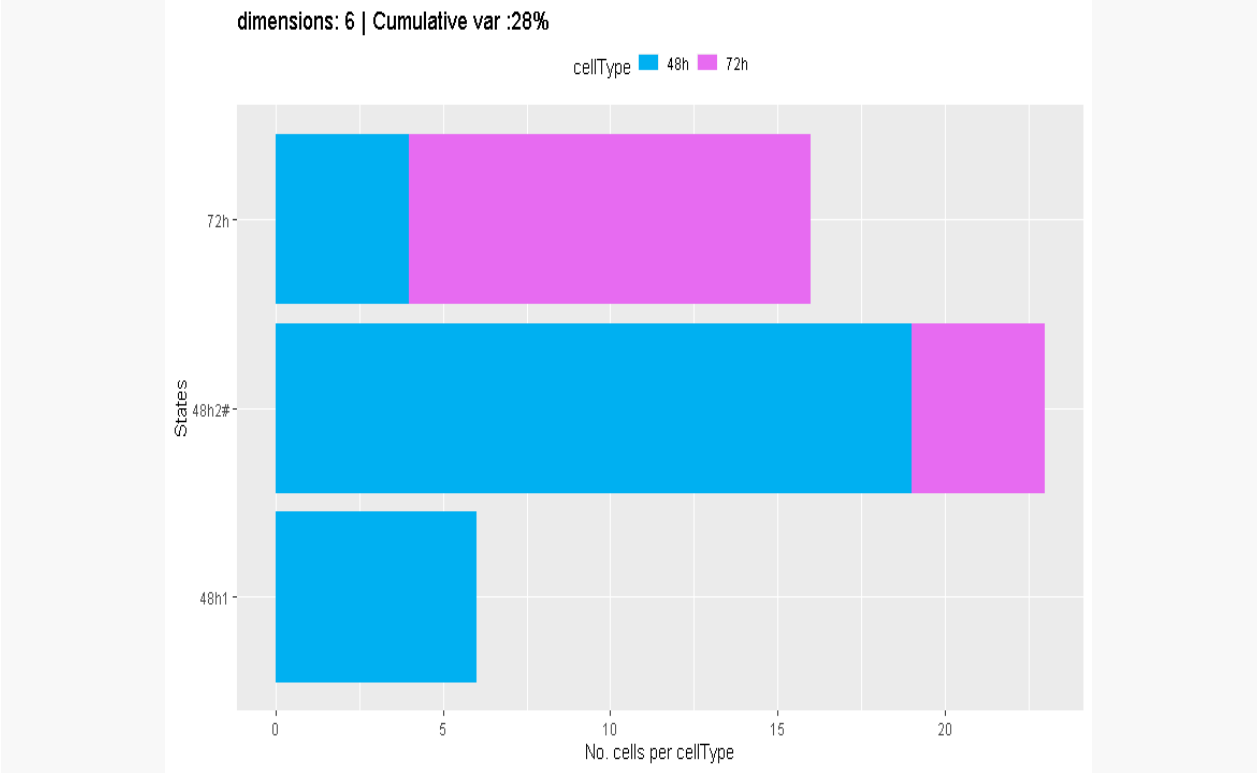

Figure 45 States Composition plots colored using cell type

The MLscAN run of the mixed state in isolation reveals three sub-populations, two containing mainly 48h type cell and one containing mainly 72h type cells.

6.1.3 MLscAN run combining the results of the two previous runs (Initial and Mixed State analysis)

Our next action will be to use the five clusters [1# excluded] that the initial MLscAN run produced and the three clusters that the mixed state run gave us and use the cell class assignments as initial values to conduct a third but constrained (initialized) MLscAN run. To initialize the new MLscAN run, we need to create a vector with the cell names and the states the cells belong to.

Below, we present the steps for creating the initialization vector.

```
#The initialization vector for the next MLscAN run, Contains clusters found in the initial run combined with the mixed state run sub-clusters.

#Obtain states and names of cells from the initial run
 init_vector <- as.vector(modelMAPState2(MLscANObj)[,1])
 names(init_vector) <- rownames(modelMAPState2(MLscANObj))

#Obtain states from the mixed state run
 mixed_state_subclusters <- modelMAPState2(mixed_state_mlscan)

#Replace 1# state with the appropriate state created from the mixed staterun
```

```
  for (subcluster in unique(mixed_state_subclusters)){

subcluster_cells <-
rownames(mixed_state_subclusters[which(mixed_state_subclusters[,1] ==
subcluster),])


init_vector[which(names(init_vector) %in% subcluster_cells) =  paste0(subcluster,"_ms")

}

#Find the most variable 500 genes. We will use them to restrict key genes exploration to reduce computation time

top_500_var_genes <- colnames(getTopNExprData(exprData = new_exprData,topN = 500))
```

We skip the model selection step through initialization as GMM is applied with a given number of states and a specific model name. MLscAN, in this case, just applies the Expectation-Maximization algorithm using the given parameters to derive posterior probability distributions for the cells and infer trajectories.

```
#Initialized MLscAN run
hayashi_ms_analysis_init <- MLscAN( exprData= expressD, #Expression Matrix
                    MLscANCellFeatures= cellFeat, #Cell Features
                    modelInit= init_vector, #Initialization vector
                    modelStateNameMode= "mostFreqPerState",  #Naming method for the inferred states
                    dimRedMethod = "prcomp", #PCA method to use
                    dimRedTopN= 1000, #Number of most variable genes to use in PCA
                    modelModelNames= modelModelNames(MLscANObj), #The initial run model name(VEI)
                    kgGenesSelFun= kg_voting(), #Key genes determined by voting method
                    MLscANOutMode= "no", #Do not produce any output files
                    keyGenesVector= top_500_var_genes, #Use specific genes for key genes discovery
                    MLscANColors = coloring #Use specific colors for certain cell features
)
##
## Creating the MLscAN object...
## Performing dimensionality reduction...
## Creating the model...
```

Forming the sub-populations...
Possible mixed state(s): 12h 48h3 48h2 72h1 72h2
Creating the trajectories...

```
plotDimRed(hayashi_ms_analysis_init,feature="cellType")
```

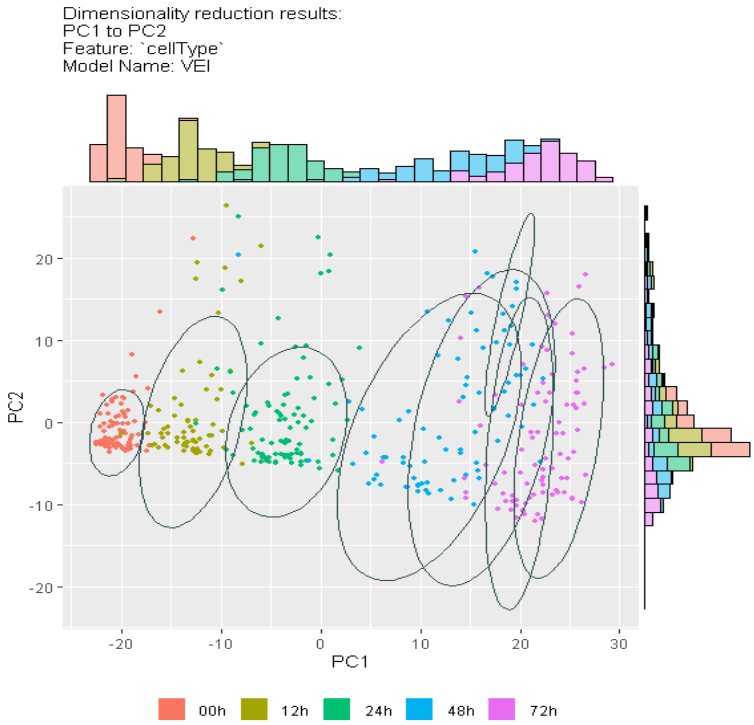

Figure 46 Inspect the PC1 vs. PC2 results with cells colored by cell type (ground truth)

```
plotStatesComposition(hayashi_ms_analysis_init,feature="cellType”)
```

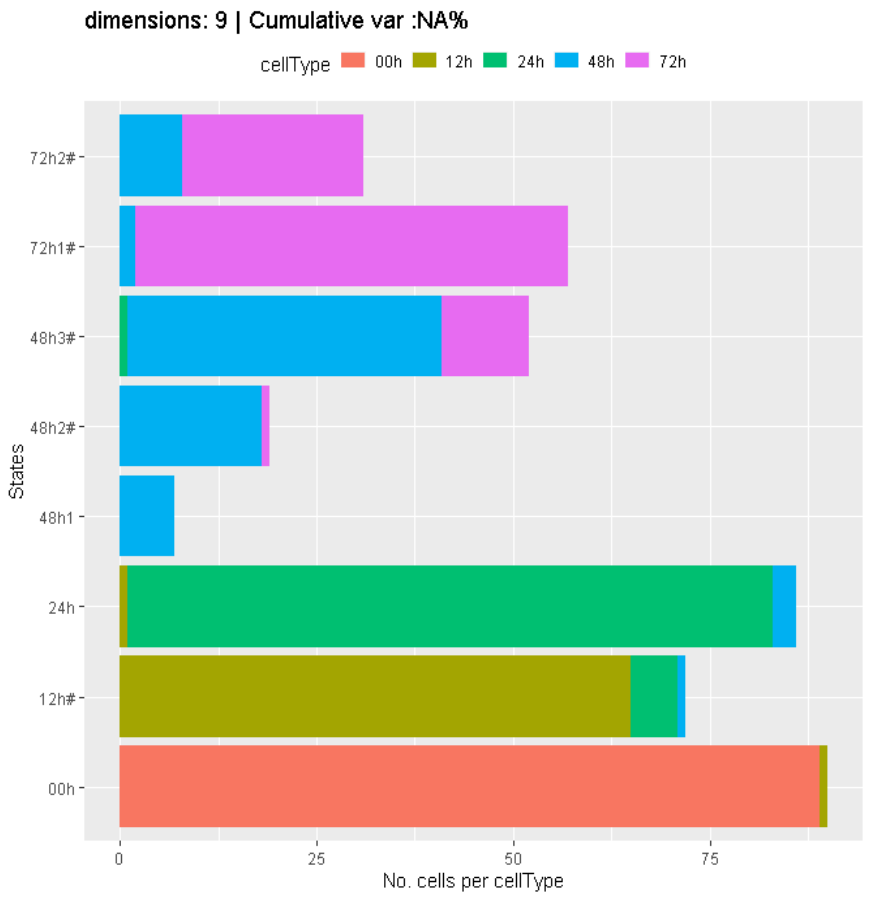

Figure 47 States Composition plots colored using cell type

MLscAN detects two possible subpopulations for 72h and three for 48h cell types. However, all subpopulations (states) consist of cells of mostly one type (cell stage) by at least 70%.

```
plotTrajectories(hayashi_ms_analysis_init)
```

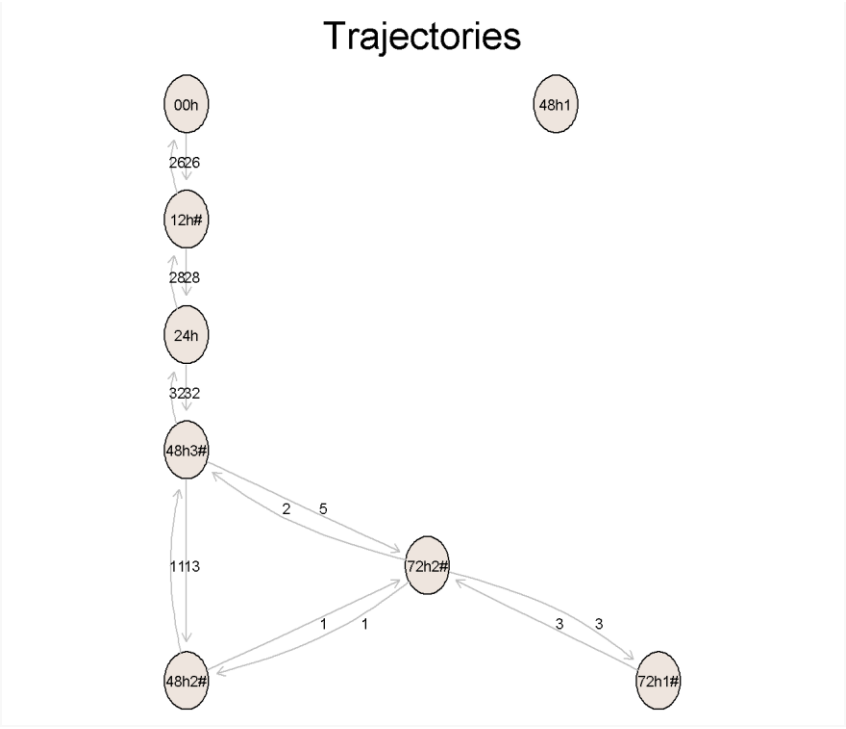

Figure 48 Inferred trajectories and their key-genes

```
plotTransitions(hayashi_ms_analysis_init)
```

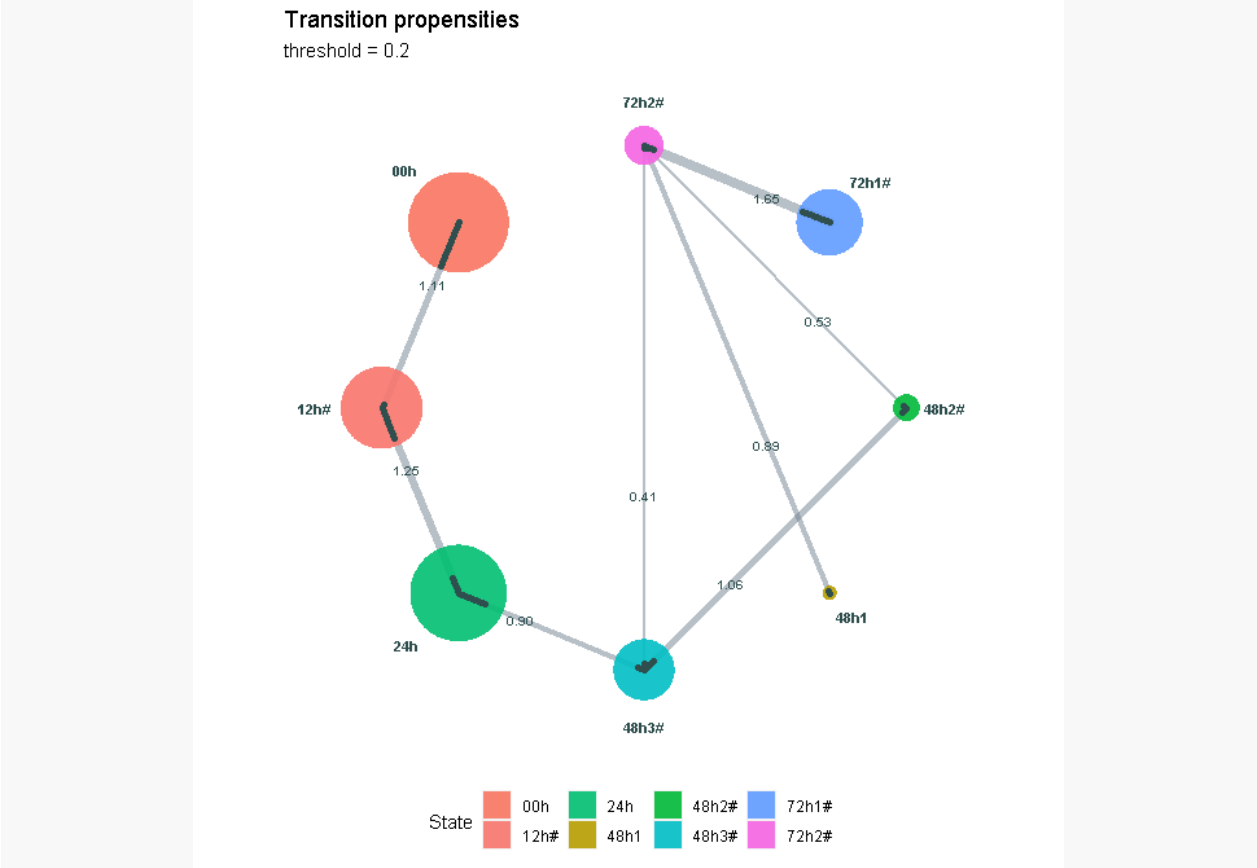

Figure 49 Inferred state transitions

Moreover, MLscAN infers a logical trajectories network that follows the time progression of the prior information, as two 48h states are connected with the intermediate 72h2# state, which connects to the primary 72h1# state. The isolated 48h1 state contains only a few cells “looking towards” the 72h2# state. Further bioinformatic analysis of these sub-populations and their interactions revealed by MLscAN in an unbiased and unsupervised manner may provide new insights. Such analysis is, however, beyond the scope of this document.

6.1.4 Pseudo-time meta-analysis

Many trajectory inference packages have the final goal to create a pseudo time ordering using their trajectories [24].MLscAN was created with the main goal to discover cell sub-populations, assess their state-to-state interactions and their dynamic gene regulatory mechanisms in an unbiased manner. However, using an external function called Trajectory path MLscAN can also generate pseudotime- like cell ordering along a particular multi-state trajectory path using the trajectories cells’ first posteriors. The function takes as input the MLscAN object and the name of a valid trajectory path. Below we provide an example of this function using the trajectory path 00h-12h#-24h-48h3#-48h2#, i.e., the path with the higher transition propensities in the transitions plot.

traj_path <- TrajectoryPath(MLscANObj = hayashi_ms_analysis_init,trajectory = "00h-12h#-24h-48h3#-48h2#")

Cell Name	Cell Position	Cell State
RamDA_mESC_00h_E05	1	00h
RamDA_mESC_00h_F08	2	00h
RamDA_mESC_00h_G08	3	00h
RamDA_mESC_00h_B03	4	00h
RamDA_mESC_00h_B04	5	00h
RamDA_mESC_00h_C11	6	00h
RamDA_mESC_00h_F06	7	00h
RamDA_mESC_00h_F11	8	00h
RamDA_mESC_00h_D10	9	00h
RamDA_mESC_00h_E11	10	00h
RamDA_mESC_00h_D03	11	00h
RamDA_mESC_00h_D06	12	00h
RamDA_mESC_00h_B05	13	00h
RamDA_mESC_00h_H02	14	00h
RamDA_mESC_00h_C06	15	00h
RamDA_mESC_00h_C08	16	00h
RamDA_mESC_00h_A05	17	00h
RamDA_mESC_00h_B06	18	00h
RamDA_mESC_00h_H01	19	00h
RamDA_mESC_00h_F12	20	00h

Figure 50 Part of the matrix produced by TrajectoryPath

6.2 MLscAN Mixed State issue approach: Mixed State removal

6.2.1 Introduction

We will now present an example where a detected mixed state is an outlier state and show how to remove it from downstream analysis. We will use Engel's natural killer T single cells dataset [49], consisting of 187 thymic NKT cells and 6774 genes. In their research, Engel's group characterized thymic NKT cells into four subsets called NKT0, NKT1, NKT2, and NKT17 that were highly divergent, despite their antigen similarity, with many gene-expression and epigenetic differences.

6.2.2 The MLscAN run

As we have discussed, MLscAN uses by default parsimonious modeling based on the δBIC method to decide the "best" GMM model type and its number of states in an

unbiased manner. However, one may also want to explore using the maximum BIC model, which usually has more states, including mixed states with few cells.

In the following MLscAN run we explore the maxBIC option. For dimensionality reduction, the run uses the 1000 most variable genes and the irlba [27] PCA method with 100 PCs by default.

```
# MLscAN run

NKT_mlscan <- MLscAN(exprData=expressD,#Expression Matrix

    MLscANCellFeatures= cellFeat,#Cell Features
    modelStateNameMode = "mostFreqPerState",#Naming method for the inferred st
    ates
    MLscANColors=NKT_colors,#Use provided colors for cell features
    modelStatesSelFun=maxBICState, #Select model name and number of state   us
    ing maximum BIC
    MLscANStopAt="traj", #Analysis stop at trajectories, do not produce key genes a
    nd GRNs
    MLscANOutMode="no" #Do not produce any output files


))

##
## Creating the MLscAN object...
## Performing dimensionality reduction...
## Creating the model...
## Forming the sub-populations...
## Possible mixed state(s): 1
## Creating the trajectories...

plotDimRed(NKT_mlscan,feature="cellType")
```

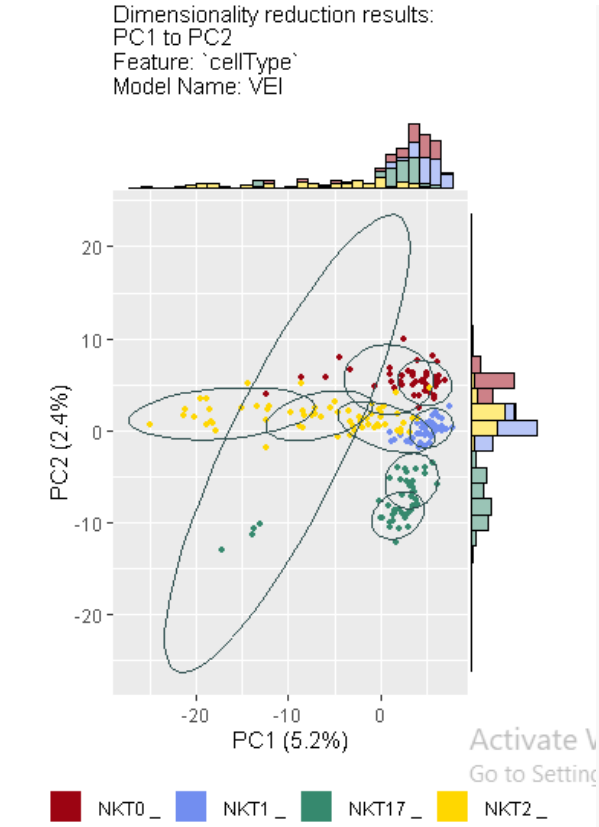

Figure 51 Inspect the PC1 vs. PC2 results with cells colored by cell type (ground truth)

By inspection, we can easily detect a mixed state (1#) with a very large PC2 variance in the dimensionality reduction plot. Because of that, we wait to observe transitions between the mixed and most of the other states.

```
plotStatesComposition(NKT_mlscan,feature="cellType")
```

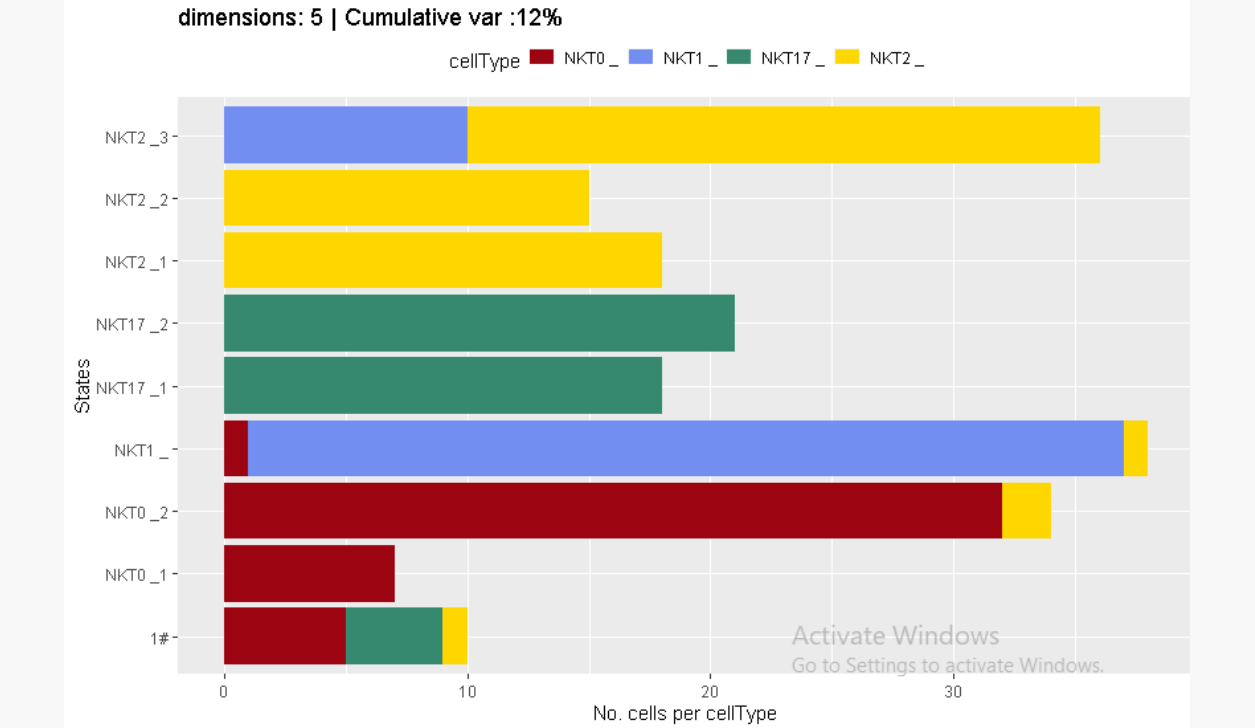

Figure 52 States Composition plots colored using cell type

As we can see in the above states composition plot, MLscAN finds three NKT2, two NKT17, two NKT0, and one NKT1 sub-clusters. It also creates one mixed state (1#) with few cells that consists of NKT0, NKT17, and NKT2 type cells. The mixed state has 10 cells out of 187 total cells (5.1%). Since the mixed state is heterogeneous and contains a very small proportion of the total cells we consider it an outlier and will try to remove it.

```
plotTransitions(NKT_mlscan)
```

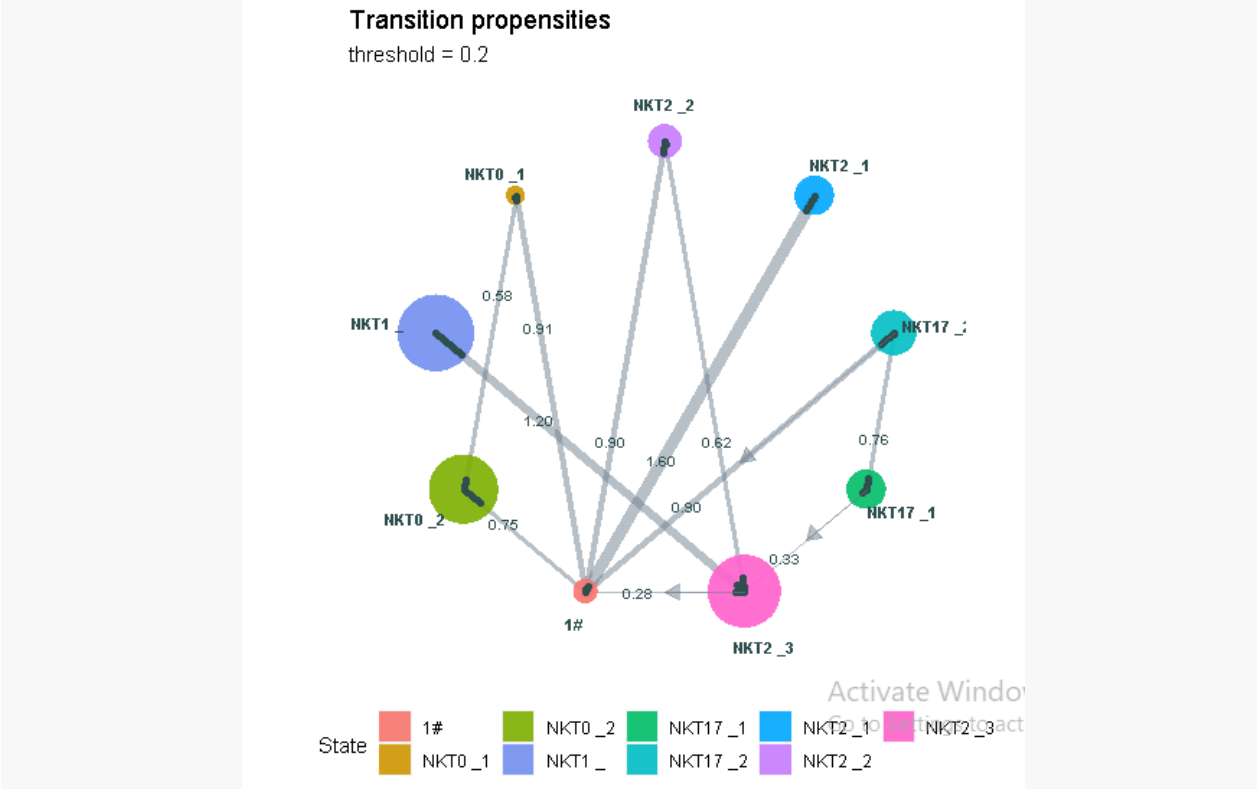

Figure 53 Inferred state transitions

Before doing so, by examining the transition plot above, we can see that nearly every other state wants to interact with the mixed state. This is expected because this state casts a wide net in the posterior probabilities space due to its very large variance. Therefore, it is justified to remove it as this transition network is not realistic due to that artifact.

6.2.3 Mixed State removal

We describe below the steps for isolating the mixed state cells and removing them from the initial dataset using appropriate MLscAN getter functions.

1. Obtain all cells
2. Obtain the cells of a certain state (mixed state)
3. Take the rest of the cells (all cells - mixed state cells)
4. Obtain the expression data for specific cells (rest of cells)
5. Obtain the dimensionality Reduction Data for specific cells (rest of cells)
6. Remove genes that have only zero expression values from the new expression matrix

7. Obtain the cell Features Data for specific cells (rest of cells)
8. Run MLscAN with those inputs (rest of cells).

```
MLscANObj <- NKT_mlscan  # Name of MLscAN object
state <- "1#" # name of the Mixed State

#Setting up the arguments for MLscAN  run with MLscAN getters removing all mixed sta
te cells
#1st step: Obtain all cells
all_cells <- cellNames(MLscANObj)


#2nd step: Obtain the cells of a certain State (mixed state)
state_rm_cells <-stateCells(object = MLscANObj,state = state )[[1]]


#3rd step: taking the rest of the cells
state_cells <- all_cells[which(!(all_cells %in% state_rm_cells))]


#4th step: Obtain expression data for specific cells (rest of cells)
state_exprData <- exprData(MLscANObj,cells = state_cells)


#5th step: Remove the genes having only 0 expression values from the new expression
data matrix
new_expr_data <- state_exprData[,colSums(state_exprData)!=0]


#6th step: Obtain dimensionality Reduction Data for specific cells (rest of cells)
state_dimRedData <- dimRedData(object = MLscANObj,cells = state_cells)


#7th step: Obtain cell Features Data for specific cells (rest of cells)
state_CellFeatures <- cellFeatures(object = MLscANObj, cells = state_cells)

#Find the most variable 500 genes. We will use them to restrict key genes exploration in this set
to reduce computation time

top_500_var_genes <- colnames(getTopNExprData(exprData = new_exprData,topN =
500))


NKT_remove_mixed_state<-MLscAN(exprData =new_expr_data,#Expression Matrix

    MLscANOutMode = "no",#Do not produce any output files
    modelStateNameMode = "mostFreqPerState",#Naming method for states
    MLscANCellFeatures = new_CellFeatures,#Cell Features
```

```
    MLscANColors=NKT_colors,#Use provided colors for certain cell features
    dimRedData=state_dimRedData,#Dimensionality reduction Data
    modelModelNames=modelModelNames(MLscANObj),#Use the initial run GMM model name(VEI)
    kgGenesSelFun= kg_voting(), #Key genes determined by voting
    keyGenesVector= top_500_var_genes #Search for key genes among the 500 most variable genes
)
##
## Creating the MLscAN object...
## Performing dimensionality reduction...
## Creating the model...
## Forming the sub-populations...
## Creating the trajectories...

plotDimRed(NKT_remove_mixed_state,feature="cellType")
```

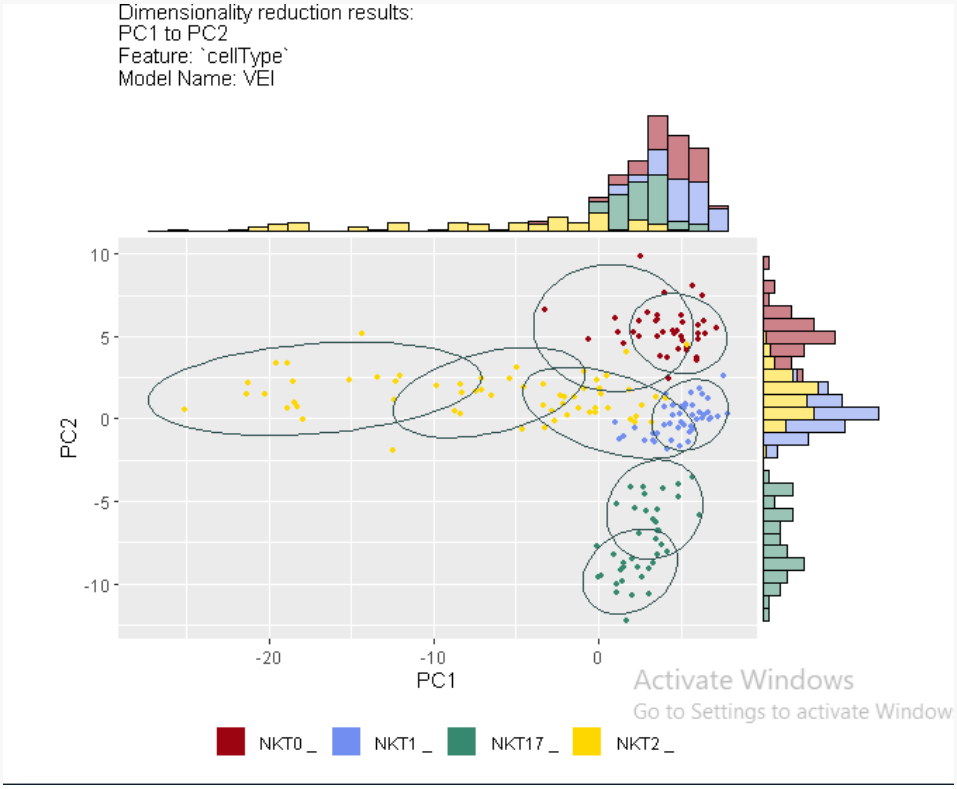

Figure 54 Inspect the PC1 vs. PC2 results with cells colored by cell type (ground truth)

```
plotStatesComposition(NKT_remove_mixed_state,feature="cellType")
```

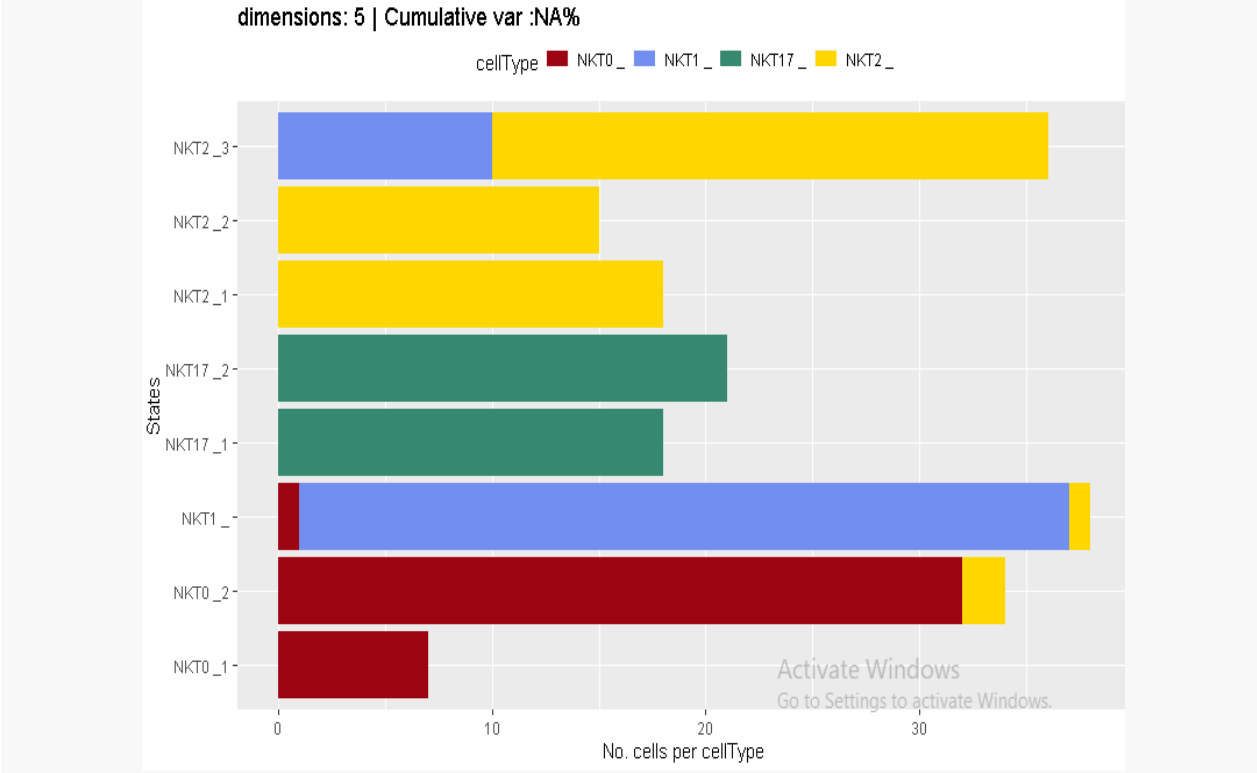

Figure 55 States Composition plots colored using cell type

Removing the mixed state has improved the clustering as seen by the two plots above.

```
plotTrajectories(NKT_remove_mixed_state)
```

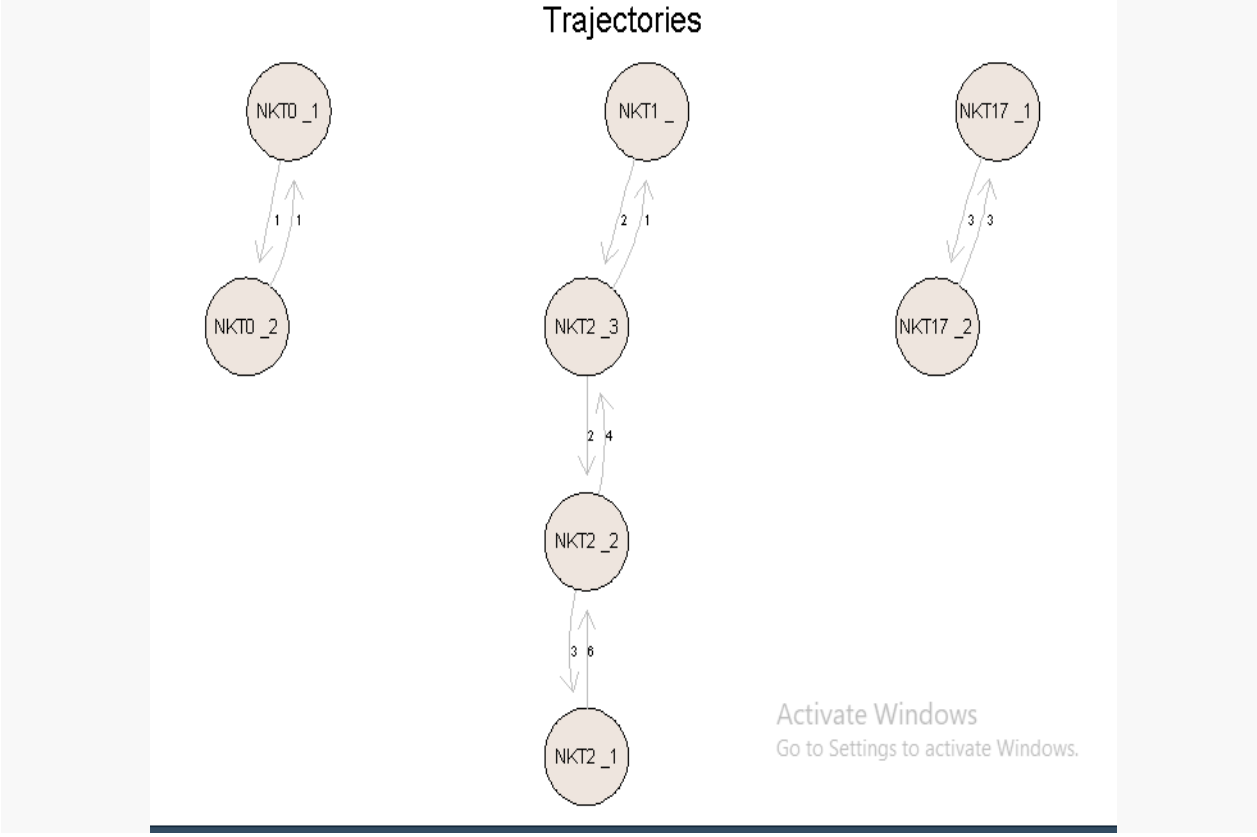

Figure 56 Inferred trajectories and their key-genes

```
plotTransitions(NKT_remove_mixed_state)
```

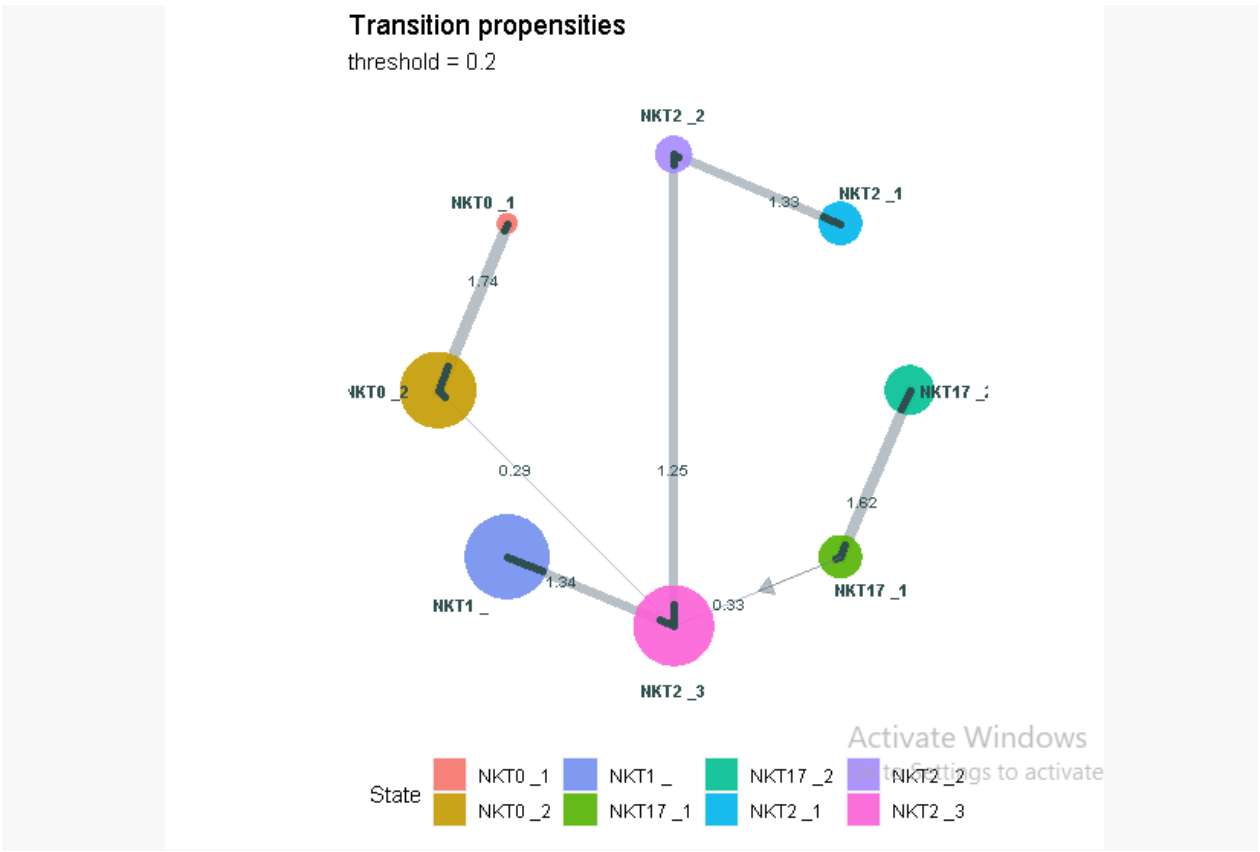

Figure 57 Inferred state transitions

Removing the mixed allows MLscAN to create a transition network with way better quality. We know from Engels' analysis [49] that NKT0, which contains precursor cells, is closer to NKT2 cells than the other cell subsets (even if it looks like it is a weak connection between those states). This seems to be confirmed by our transition plot. Also, we may conclude that NKT2 is an intermediate subset as it connects to all the other subsets. Further investigation is needed to assess if there is a biological meaning to the existence of subclusters and their network of state-to-state trajectories.

7. CONCLUSIONS AND FURTHER RESEARCH

Improving MLscAN to become a more flexible and user-friendly tool was the primary goal of this thesis. To this end, even though MLscAN was originally designed to provide an easy to use end-to-end computational analysis pipeline to the “naïve” user (non-computational expert), we now also provide to the more experienced user the ability to customize her run, incorporate external results, and experiment with model selection, while still remaining unbiased and using a probabilistic framework.

Another important contribution to the MLscAN pipeline development was detecting the Mixed States and implementing ways to handle them properly in MLscAN. Finally, we put a lot of emphasis on demonstrating all the previous contributions and MLscAN’s capabilities, versatility, and flexibility through well-designed use cases.

Specifically, Chapter 4 explores the MLscAN workflow, presenting its capabilities and navigating through the many different plots the package can create automatically. We used a pancreatic cells dataset from diabetic and non-diabetic donors [41] for this purpose.

Chapter 5 provides instructions on integrating alternate dimensionality reductions data to the MLscAN pipeline and how this can improve the analysis for specific datasets. For this case, we selected Buettner’s dataset (Buettner F, 2015) containing mouse embryonic cells in the different cell cycle stages.

Finally, we presented two approaches for dealing with the issue of the mixed states in Chapter 6. Using Hayashi’s dataset [48], we break a mixed state into its subpopulations and then repeat the analysis to include them into the epigenetic landscape before inferring trajectories. On the other hand, using Engel’s dataset [49], we show how to remove a mixed state of outlier cells. Both techniques can be used recursively, as needed.

In the fast pacing field of single-cell analytics, MLscAN may need some additions in the near future. Extending the communication between MLscAN and well-known single-cell analysis packages like SingleCellExperiments [50] and be able to use R objects created by these packages as input, will be vital to help inexperienced users implement effortlessly their expression data and corresponding metadata, minimizing even less the minimum programming skills needed to use MLscAN. Also, MLscAN can become even more user-friendly by providing a graphical user interface environment through Shiny apps [51].

Following the latest Single Cell analytics studies, we can observe an increased use of RNA velocity data [52] for improving Trajectory Inference approaches. We believe integrating RNA velocity to single-cell data analysis may offer excellent insights.

Moreover, the combination of different kinds of data modalities other than RNA-seq, such as proteomics and metabolomics, will be significant for overcoming many of the limitations we face in single-cell RNA sequencing.

Integrating multi-omics data modalities and combining bioinformatics and machine learning approaches seems to be the next big step in the long journey of revealing the biological mechanisms of cells. Furthermore, new tools research will undoubtedly support multidisciplinary efforts to help us understand and exploit the function and dynamics of complex biological systems for therapeutic and biotechnological purposes.

REFERENCES

[1] P. Ehrlich, Nobel Lecture, Tue. 23 Nov 2021.

[2] A. Weber, Discovering New Biology through Sequencing of RNA, Plant Physiology, November 2015.

[3] Z. Wang, RNA-Seq: a revolutionary tool for transcriptomics, Nature reviews. Genetics, 2009.

[4] E.Pennisi, The biology of genomes. Single-cell sequencing tackles basic and biomedical questions, Science, 2012.

[5] Zhang, Single-cell RNA sequencing in cancer research, J Exp Clin Cancer, 2021.

[6] M. Jonathan A Griffiths, Using single-cell genomics to understand developmental processes and cell fate decisions, 2018.

[7] T. Aleksandra A. Kolodziejczyk, The Technology and Biology of Single-Cell RNA Sequencing, Molecular Cell, 2015.

[8] P. Tsakanikas,E. Manolakos , Machine learning methods to reverse engineer dynamic gene regulatory networks governing cell state transitions, bioRxiv, 2018.

[9] A. P. Chatzigeorgiou, "MLscAN a tool for implementing flexible pipelines for Single-cell data analysis using unsupervised Machine learning methods," *Master thesis,* 2021.

[10] G. Chen, "Single-Cell RNA-Seq Technologies and Related Computational Data Analysis," *Frontiers in Genetics,* 2019.

[11] C. KendziorskI, "Design and computational analysis of single-cell RNA-sequencing experiments," *Genome Biology,* 2016.

[12] S. Zhang, "Comparison of Computational Methods for Imputing Single-Cell RNA-Sequencing Data," *Transactions on Computational Biology and Bioinformatics,* 2020.

[13] W. Ratajczak, "Principal components analysis (PCA)," *Computers & Geosciences,* 1993.

[14] S. Lafon, "Diffusion maps," *Applied and Computational Harmonic Analysis,* 2006.

[15] G. Hinton, "Visualizing Data using t-SNE," *Journal of Machine Learning Research,* 2008.

[16] L. McInnes, UMAP: Uniform Manifold Approximation and Projection, Journal of Open Source Software, 2018.

[17] W. Saelens, "A comparison of single-cell trajectory inference methods," *Nat Biotechnol,* 2019.

[18] T. Stuart, "Comprehensive Integration of Single-Cell Data," *Cell,* 2019.

[19] Wang, " Independent component analysis-based dimensionality reduction with applications in hyperspectral image analysis.," *Geoscience and Remote Sensing,* 2006.

[20] G. Hinton, "Visualizing Data using t-SNE," *Journal of Machine Learning Research,* 2008.

[21] Kiselev VY, "SC3: consensus clustering of single-cell RNA-seq data," *Nat Methods,* 2017.

[22] J. B. MacQueen, " Some Methods for classification and Analysis of Multivariate Observations. Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability.," *University of California Press,* 1967.

[23] Faisal Saeed, " Combining Multiple Individual Clusterings of Chemical Structures Using Cluster-Based Similarity Partitioning Algorithm," 2012.

[24] Trapnell C, "Nat Biotechnol," 2014.

[25] Street, "Slingshot: cell lineage and pseudotime inference for single-cell transcriptomics," *BMC Genomics ,* 2018.

[26] Wolf, " PAGA: graph abstraction reconciles clustering with trajectory inference through a topology preserving map of single cell," *Genome Biol ,* 2019.

[27] J. Baglama, IRLBA: Fast Partial Singular Value Decomposition Method, Handbook of Big Data, 2016.

[28] R. P. Kaufman, "Partitioning Around Medoids (Program PAM). In Finding Groups in Data," 1990.

[29] Blondel, J.-L. Guillaume, R. Lambiotte and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics: Theory and Experiment,* 2008.

[30] D. Reynolds, Gaussian Mixture Models, Encyclopedia of Biometrics, 2009.

[31] O. Bor˚uvka, *O jistém problému minimálním,* 1926.

[32] J. Schaffer, "What Not to Multiply Without Necessity," *Australasian Journal of Philosophy,* 2015.

[33] R. Development Team. A language and environment for statistical computing, 2015.

[34] L. Scrucca, mclust 5: Clustering, Classification and Density Estimation Using Gaussian Finite Mixture Models, 2016.

[35] Finak, "MAST: a flexible statistical framework for assessing transcriptional changes and characterizing heterogeneity in single-cell RNA sequencing data," *Genome Biol,* 2015.

[36] Robinson MD, "edgeR: a Bioconductor package for differential expression analysis of digital gene expression data," *Bioinformatics,* 2010.

[37] C. Y. Kieran R Campbell, "switchde: inference of switch-like differential expression along single-cell trajectories," *Bioinformatics,* 2017.

[38] P. J. Lavrakas, "Bootstrapping," *Encyclopedia of Survey Research Methods,* 2008.

[39] T. Ho, "Random decision forests," *Proceedings of 3rd international conference on document analysis and recognition,* 1995.

[40] I. Jolliffe, "Principal component analysis. Springer, New York," 1986.

[41] Wang YJ, "Single-Cell Transcriptomics of the Human Endocrine Pancreas. Diabetes," 2016 .

[42] Lun ATL, "A step-by-step workflow for low-level analysis of single-cell RNA-seq data with Bioconductor," 2016.

[43] R. K. Standish, "Why Occam's razor.," *Foundations of Physics Letters ,* 2004.

[44] Sophie A Harrington, The Wheat GENIE3 Network Provides Biologically-Relevant Information in Polyploid Wheat G3 Genes|Genomes|Genetics, 2020.

[45] S. X. Deng AC, "Dynamic gene regulatory network reconstruction and analysis based on clinical transcriptomic data of colorectal cancer," *Math Bioscience Eng,* 2020 .

[46] Buettner F, "Computational analysis of cell-to-cell heterogeneity in single-cell RNA-sequencing data reveals hidden subpopulations of cells," *Nat Biotechnol,* 2015.

[47] L. McInnes, "UMAP: Uniform Manifold Approximation and Projection," *Journal of Open Source Software,* 2018.

[48] T. Hayashi, "Single-cell full-length total RNA sequencing uncovers dynamics of recursive splicing and enhancer RNAs," *Nat Commun,* 2018.

[49] Engel I, "Innate-like functions of natural killer T cell subsets result from highly divergent gene programs," *Nat Immunol,* 2019 .

[50] A. Amezquita, "Orchestrating single-cell analysis with Bioconductor," *Nature Methods,* 2020.

[51] R Development Team, "RStudio: Integrated Development Environment for R," *PBC,* 2020.

[52] La Manno, "RNA velocity of single cells," *Nature,* 2018.