flufftail: Unveiling Gene Regulatory Network Dynamics through Fuzzy Clustering of Single-Cell RNAseq Data

USN: 2292
Wolfson College
19/07/2024

This dissertation is submitted for the degree of Master of Philosophy in Population Health Sciences (Health Data Science)

UNIVERSITY OF CAMBRIDGE

Abstract

Due to recent technological advances, single-cell assays are now firmly accepted as state-of-the-art for a wide range of biomedical projects; RNA-focused assays represent a cost-effective and quick proxy for both DNA and protein evaluations, as well as containing intrinsic expression signals. Clustering is pivotal in scRNA-seq for partitioning cells into distinct groups, setting the stage for analyses such as trajectory inference of cell-cell interactions. Recently, community detection algorithms emerged as superior for identifying transcriptomic-driven subpopulations.

Despite their enhanced performance, all cutting-edge community detection techniques are inherently stochastic and crisp; iterative runs result in variable partitions and interpretations even when applied to identical inputs. This variability highlights the need to replace crisp assignation with a probabilistic/fuzzy approach, i.e., fuzzy clustering, where cells can concurrently be associated with multiple clusters.

Inspired by findings from Gribben et al. [1] on biphenotypic cells in metabolic dysfunction-associated diseases, in this project I introduce *flufftail*, a comprehensive R package designed for single-cell data analysis through the angle of fuzzy clustering that aims to characterise fuzzy cells/genes and subsequently uncover insights on mechanisms that drive cellular plasticity and GRN dynamics. *flufftail* exploits the variability of standard clustering approaches by proposing a fuzzy community-detection clustering through repeated stochastic clustering. I also developed a new methodology for identifying key genes (major regulatory hubs) that drive biological transitions through fuzzy gene module clustering. Furthermore, *flufftail* presents a new approach for characterising gene regulatory network (GRN) dynamics and the evolution of regulatory interactions for major regulatory hub genes at different stages of the pseudotime ordering of cells. Importantly, a Shiny interface was developed, allowing interactive analysis with the package and democratising it to researchers from diverse backgrounds.

This marks the first exploration/exploitation of GRN dynamics across different points (or bins) of pseudotime in single-cell data, advancing our understanding of the mechanisms controlling cellular plasticity, transdifferentiation, and potential therapeutic targets.

Keywords: single-cell sequencing, clustering algorithms, element-centric clustering comparison, robustness, stability, cell identity, data-driven characterisation.

Declaration

This dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration except where specifically indicated in the text.

Word count: 12,765

Acknowledgments

I would like to express my deepest gratitude to my supervisor Dr. Irina Mohorianu, for her excellent guidance and expertise throughout the research and writing process. Her advice and support were fundamental in the completion of this project.

Furthermore, I would like to thank Andi Munteanu for the very valuable discussions and insights that significantly contributed to my research.

Lastly, I would like to thank Dr. Floris Johan Maria Roos for his support with the biological interpretations, and Professor Ludovic Vallier (Berlin Institute of Health, Germany) for providing me with the datasets I used to develop and test *flufftail*.

Contents

Chapter 1

Introduction

1.1 Background

1.1.1 Biological context. single-cell assays

The central dogma of molecular biology outlines the flow of genetic information from DNA to RNA, via transcription and subsequently to protein, via translation, consolidating RNA quantification as a reliable proxy for events both at the DNA level and with respect to protein expression [2]. Traditional bulk RNA sequencing (RNAseq) methods aggregate RNA from large populations of cells, providing an overview of gene expression across tissue(s) for a large number of genes but a limited number of samples (large m, small/medium n). While comprehensive, this approach masks the heterogeneity among individual cells by averaging gene expressions within each sample. In contrast, scRNA-seq allows for the analysis of gene expression at individual cell level, resulting in datasets with a large number of genes and a large number of samples (large m, large n). This technique can help uncover cellular heterogeneity, identify rare cell types and states, and enable a more precise understanding of biological processes [3, 4]; recent technological advances for linking transcriptomic profiling to other modalities, also enhanced the information available for proposing mechanistic hypotheses of regulatory interactions [5].

In scRNA-seq analyses, the identification of cell types is essential for all subsequent analyses; rephrased from a computational perspective, clustering is key for partitioning cells into distinct groups (cell types) using cell-specific transcriptional profiles as starting point [6], and facilitates the identification of cell markers and rare populations, also setting the stage for further analyses like trajectory inference [7]. Recently, community detection algorithms emerged as superior clustering techniques for identifying cell-driven (sub)populations [8]. A limitation damping down their superior performance for single-cell analysis, is that all cutting-edge community detection techniques are inherently stochastic [9]. Repeated runs on identical inputs,

only changing the random seed, can yield very different partitions, and variable subsequent interpretations.

1.1.2 Single-cell clustering

The methodology introduced by PhenoGraph [10] is the state-of-the-art pipeline for single-cell clustering with community detection algorithms; it is integrated into popular single-cell analysis libraries like Seurat [11], Monocle3 [12], and Scanpy [13]. The pipeline comprises three main steps: i) dimensionality reduction, ii) graph construction, and iii) graph clustering. As illustrated in an assessment framework, ClustAssess [9], decisions made at each step within the pipeline can significantly impact the robustness of the clustering.

While most clustering evaluation metrics are cluster-centric (e.g., Rand Index, adjusted Rand Index), recently, an element-centric approach was introduced, termed element-centric similarity (ECS) [14], enabling, for the first time, the evaluation of robustness of clustering for individual entries, across two partitions; the robustness across a set of partitions can be summarised by element-centric consistency (ECC), also known as frustration [14, 9]. This enables the precise identification of fuzzy entries (in this context, cells) whose clustering behaviour is inconsistent throughout several partitions. The essential goal of the partitioning step is that a proposed clustering should capture the true biological signal and not be biased by technical variation (e.g., due to changing random seed) [9]. Rephrased, over repeated clusterings with different random seeds, there is an expectation of a high median ECC, across all cells, supporting the conclusion that the partitioning of cells remains robust even under algorithmic/initialisation variations.

1.1.3 Impact of fuzzy clustering on the biological interpretation

However, the robustness assumption underlining crisp clustering has only a limited applicability on biological datasets i.e., while there exist homogenous populations of cells, it is also acknowledged that transitioning cells or perturbed cells can occur [15]. The current strategy, to exclude problematic cells, can mask biologically valid conclusions [16]. To illustrate the latter scenario, Gribben et al. [1] revealed, for the first time, the presence of biphenotypic cells, defined on Cholangiocytes and Hepatocytes, in end-stage metabolic dysfunction-associated steatotic liver disease (MASLD) and metabolic dysfunction-associated steatohepatitis (MASH) patients. These cells co-express specific markers for both cell types, form a transitioning bridge between the two cell types, and are characterised by transdifferentiating states. The impact of uncovering these cells was significant, as it altered our understanding of disease progression in chronic liver conditions i.e., the paper suggested that cellular plasticity in a diseased environ-

ment is a response to pathology rather than a regenerative process. Reaching these results using state-of-the-art methods was highly complex, relied on extensive manual curation, and involved numerous steps e.g., clustering and subclustering to differentiate between cell populations, finding specific marker genes, computing pseudotime and RNA velocity (the latter spanning different programming environments). The authors concluded by suggesting that a deeper understanding of the signals driving and controlling the plasticity could pave the way for the development of efficient and safe therapeutic strategies against chronic liver diseases.

The description of biphenotypic cells resembles the concept of fuzzy (soft) clustering, which allows the assignment of entries/cells to multiple clusters, allocating each a probability of assignment [17]. I hypothesise that following robust clustering that captures the true biological signal of the data (i.e., homogenous cells within partitions lead to high median ECC), any resulting variability in individual cells is more likely to reflect biological signal rather than technical noise. Specifically, it could indicate the presence of cells in transitional states (e.g., transdifferentiating) or cells exhibiting multiple phenotypes (e.g. biphenotypic cells [1, 18]). Identifying and further characterising these cells (i.e., getting their membership degrees, visualising their location and co-clustering behaviour) has the potential to uncover valuable insights into cellular plasticity, developmental processes, and disease mechanisms. While numerous packages have been developed for the analysis of single-cell datasets through repeated clustering (e.g., fuzzy or consensus clustering) such as SC3 [6] or scALPO [19], existing literature does not focus on the downstream impact of fuzzy clustering in terms of characterising regulatory dynamics.

The concept of (fuzzy/crisp) clustering is agnostic on input. Its usage can be altered to fit the task of identifying also gene modules i.e., groups of genes with similar expression profiles that are functionally related and co-regulated [20]. While few packages such as Monocle3 [12] used community-detection clustering to identify gene modules, no prior literature exists on using fuzzy community-detection clustering to identify genes involved across multiple modules. Furthermore, Raza et al.[21] briefly discusses the effects of fuzzy genes, identified through fuzzy clustering, on gene regulatory networks (GRNs), providing a structured representation of causality/directionality of gene interactions with nodes as genes and edges as regulatory relationships. However, no bioinformatic packages currently support or build on such analyses. I hypothesise that identifying genes involved in multiple modules (i.e., fuzzy genes) that also have high connectivity in the GRN could be particularly interesting, as targeting them may cause a ripple effect [22, 23], propagating changes throughout the network. Identifying these driver genes (also known as major regulatory hubs [24]), especially for transitioning/fuzzy cells, followed by investigating the GRN dynamics of those hub genes across the transition, could help better understand the signals controlling the appearance of plasticity. This represents a step closer to identifying potential therapeutic targets, as Gribben et al. hinted in their concluding remarks.

Chapter 2

Aims

My project aims to develop the methodology and an R package that facilitate fuzzy clustering to identify and characterise fuzzy cells/genes and subsequently uncover insights on mechanisms that drive cellular plasticity and GRN dynamics. The *flufftail* framework is applicable to all single-cell datasets (i.e., it is agnostic to the quantified modality). The objectives for the project are the following:

1. **Method & Implementation:** Assess state-of-the-art community-detection clustering methods from a new perspective i.e., the variability between runs. Utilising their stochasticity, introduce a fuzzy community detection clustering through repeated stochastic clustering; Programatically develop functions that support both cell- and gene-level fuzzy analyses.

2. **Implementation:** Develop functions within the package to characterise the "fuzziness" of assignations. This includes:

    - Computing and visualising consensus matrices to depict co-clustering behaviours.
    - Analysing fuzzy entries (cells/genes) by visualising their locations (e.g., on a UMAP), calculating their membership degrees to show cluster affiliations, and predicting their most probable clusters through majority voting.

3. **Method & Implementation:** Development of a methodology for the identification of major regulatory hubs based on fuzzy gene modules and high-covariation with many other genes; Programatically develop functions to identify these hub genes in a data-driven way.

4. **Method & Implementation:** Explore GRN dynamics (i.e., variation in strength of interactions and topology) and the ordered dynamics of regulatory interactions of hub genes at different stages of a biological process (snapshots at pseudotime trajectory). This is the first time the dynamics of GRNs are explored at the single-cell level.

5. **Implementation:** Develop code to programmatically build an R shiny interface that would provide interactive access to all the analysis steps that researchers could do with the package using scripting. This would enable rapid experimentation/interaction with the data and hypothesis generation while also providing access for scientists with diverse expertise (e.g., wet-lab researchers as well as bioinformaticians).

Chapter 3

Methodology

This Chapter begins with a brief discussion of the deviations from the project proposal. I then explain the setup used to develop the framework and provide essential background on pre-processing and state-of-the-art community-detection clustering. Next, I focus on the extension of community-detection algorithms through stochastic repeated clustering and the identification of fuzzy cells. Finally, I describe the novel methodologies developed for identifying major regulatory hubs and exploring GRN dynamics.

3.1 Deviations from the proposal

While the main focus and purpose of the project remain unchanged, several adjustments have been applied in response and as a result of reflection on the feedback received; further developments are also the result of further research and interactions with my supervisor. Next, I will highlight and justify these amendments:

1. One of my primary objectives mentioned in the proposal was "creating a novel fuzzy community detection algorithm". I have since rephrased it to be more precise (see objective 1 - Chapter 2). While I am not developing a new clustering algorithm in the classical sense (i.e., structurally novel), I am utilizing the stochasticity of community-detection clustering to assess fuzziness. Through repeated clustering with different random seeds, followed by a reconciliation of the results, I am extending the applicability of standard community-detection clustering algorithms to be used in a fuzzy clustering context. The novelty lies in offering the bioinformatics community a streamlined approach for fuzzy community-detection clustering, but more importantly, in how I use these algorithms in the *flufftail* framework comprising post-clustering subsequent steps.

2. In the proposal, I focused more on the downstream impact (pseudotime, differentially

expressed genes) affected by including and excluding fuzzy cells. While I included a version of this task in the current iteration by checking the robustness of pseudotime after removing the top x% of fuzzy cells (Sections 3.5.1 & 4.2.5), my focus has now shifted to harnessing the information that the fuzziness enables which I believe is more powerful and informative than the simple identification and exclusion of cells. This is especially exemplified by identifying major regulatory hubs in a data-driven way using fuzzy gene module clustering (Sections 3.5.2 & 4.2.7), a concept I hadn't considered when I first started working on the project. Due to this shift, I also changed the title of the dissertation to more accurately reflect its content.

3. I indicated in the proposal that I would like to examine the GRN dynamics; however I initially considered this from the perspective of calculating general GRNs for each identified cluster (not specifying specific transcription factors) and examining how the inclusion of a fuzzy cell in multiple clusters, thus being used to infer multiple GRNs, could change/ modulate the dynamics compared to a crisp approach. After conducting more research and devising the idea of identifying major regulatory hubs using fuzzy gene module clustering, the concept evolved to include checking the GRN dynamics of the identified hubs (setting them as transcription factors) at different snapshots/bins across the pseudotime transition of two clusters with adjacent pseudotime distributions (Sections 3.6 & 4.2.8). This enables a deeper understanding of the dynamics and impact of the genes that influence the transition across different points of a biologically dynamic process.

4. Following discussions with my supervisor, we decided against focusing on the mathematical link between ECC and entropy (the third objective in the proposal). Initially, plots generated on subsamples of cells suggested a potential mathematical relationship; however subsequent experiments with a larger datasets and higher number of clusters revealed a non-negligible variation in ECC scores for similar entropy values (Appendix C), suggesting that while there is a clear negative correlation, an exact mathematical link might be more challenging at this point. Moreover, upon reflecting on feedback from the assessors, I concluded that focusing on this link would divert resources and attention from the central aims of the project, which are critical to its success.

Appendix B includes a response to the feedback received, highlighting how I addressed the assessors' suggestions.

3.2 Setup, pre-processing steps, and clustering stability

This section overviews the setup (computing environment, datasets) used for developing the framework and gives essential background for the pre-processing steps and current state-of-the-

art community detection clustering.

3.2.1 Setup used for developing and testing of *flufftail*

Code was developed and analyses were run on R version 4.2.1 on a MacBook Pro with the following specifications: Apple M3 Pro chip, 18 GB of memory, running macOS Sonoma 14.1. The package will be summarised in a future CRAN package named *flufftail*. The final code, submitted along with the report, is separated into two folders:

1. **Flufftail:** includes all the source code developed for the package.
2. **Analysis:** includes all the scripts and vignettes developed to write this report and to test the package.

The package was developed and tested on subsets of a single-nuclei RNA sequencing (snRNA-seq) dataset by Gribben et al. [1] for metabolic dysfunction-associated steatotic liver disease (MASLD) and metabolic dysfunction-associated steatohepatitis (MASH). The full dataset comprises 99,809 cells and 30,117 genes across 47 patients/samples. The subsets were provided by the authors as Seurat objects.

To develop the package, the *Immune* subset was used, which includes 9,239 cells and 24,539 genes across three cell types: 5,755 Lymphocytes, 3,014 Macrophages, and 470 Neutrophils. These cells comprise 2,442 end-stage, 1,097 healthy control, 906 MASLD, 4,530 MASH without cirrhosis, and 264 MASH with cirrhosis. Once developed, the package was evaluated using the *End-stage* Seurat object used by the authors to demonstrate plasticity between Hepatocytes and Cholangiocytes, consisting of 25,527 cells and 29,653 genes. All samples were from end-stage patients, and the cell types included 23,097 Hepatocytes and 2,430 Cholangiocytes. All pre-processing and clustering value selection steps were performed by the authors (using ClustAssess [9]) and are described in their paper [1]. The scientific goals with the data were to develop a framework applicable to any single-cell data and to validate the approach by comparing the results obtained with the package to the published results in the paper.

To demonstrate the generalisability of the package, the framework was also applied to a publicly-available scATAC-seq dataset (multiome PBMC) from the SeuratData [25] package. It contains 11,907 cells and 108,377 chromatin accessibility regions across various cell types, including different types of monocytes, T cells, B cells, natural killer cells, and dendritic cells.

3.2.2 Pre-processing

A vital pre-processing step in scRNA-seq analysis is quality control, which removes low-quality cells and genes (i.e., technical noise) by assessing gene counts, unique molecular identifiers (UMIs), and mitochondrial gene expression percentages [26]. Effective thresholds exclude cells with insufficient reads and low-expression genes, often determined through quality-control plots. Additionally, high read counts might indicate doublets, requiring stringent count depth thresholds to identify and eliminate them [26]. The Seurat objects were provided with filtering in place by the authors.

Gene expression levels were normalized and scaled using SCTransform [27], which employs Pearson residuals from regularised negative binomial regression to minimise technical noise while preserving biological variability.

Dimensionality reduction is crucial for clustering high-dimensional data, as it efficiently computes distances while retaining essential information. Principal component analysis (PCA) is the most widely used technique, projecting data onto principal components that capture the most variance. Truncation methods like Irlba [28] typically retain the first 30 to 50 principal components for a balance of information preservation and computational efficiency. However, PCA may not effectively represent complex, non-linear structures typical in biological data. To capture these complexities, non-linear methods like Uniform Manifold Approximation and Projection (UMAP) maintain both local and global patterns, providing a comprehensive representation of the data's intrinsic structure. Research suggests that PCA is better for summarising scRNA-seq data, while t-SNE and UMAP are better for visualising the data [26, 29], as relying solely on 2D embeddings can misinterpret cellular relationships [30]. Efficient feature selection is essential before performing dimensionality reduction. Ideal methods prioritize genes that reflect biological variations across, rather than within, subpopulations and minimize noise [15, 31]. Common strategies include identifying highly variable (HV) or most abundant (MA) genes. Appropriate feature sets for the dimensionality reduction setting were identified using ClustAssess.

3.2.3 State-of-the-art community-detection clustering & stability of partitions

In this section the current state-of-the-art community-detection algorithms are explained, followed by an explanation of the PhenoGraph pipeline, element-centric consistency and the need to optimise parameter values before clustering.

3.2.3.1 Community detection algorithms

Community-detection algorithms treat single-cell datasets as networks, with cells as nodes and their similarities as weighted edges. The goal is to cluster nodes with denser connections among themselves while optimising an objective function that defines the relationship between intra- and inter-cluster edge density, directly characterising the quality of the clustering. Common objective functions include modularity and RB:

- **Modularity [32]**: $Q = \frac{1}{2m} \sum_c \left(m_c - \frac{K_c^2}{4m} \right)$; where m represents the total weight of all edges, m_c denotes the weight of the internal edges within cluster c, and K_c is the sum of the weighted degrees of the nodes in cluster c.

- **Reichardt and Bornholdt (RB) [33]:** $Q = \sum_c \left(m_c - \gamma \frac{K_c^2}{4m} \right)$; where γ represents the resolution parameter used to control the number of clusters in the final partition.

The Louvain algorithm [34] is a state-of-the-art community detection method using an iterative greedy technique. It starts by assigning each node to its own cluster and follows two main steps: the local moving heuristic and graph shrinking. During the local moving heuristic, nodes may be reassigned to improve partition quality, typically evaluated via modularity, until no further improvements can be made. The second step involves shrinking the graph by replacing communities with super-nodes and recalculating edge weights as the sum of inter-cluster weights. This process repeats until no changes occur between iterations, indicating convergence. Despite its greedy nature, the Louvain algorithm efficiently achieves high-quality clustering with an average time complexity of $O(n\ log\ n)$, where n is the number of nodes.

Recent enhancements to the Louvain algorithm include Louvain-with-multi-level refinement [35], Smart Local Moving (SLM) [36], and the Leiden algorithm [37]. Louvain-with-multi-level refinement adds a phase between iterations to refine clustering by reassessing node positions within and across hierarchy levels. SLM improves clustering by iterating over communities, creating a subnetwork for each, and using the local moving heuristic to optimise community structures. SLM can improve results after multiple runs and in contrast with Louvain (and its refined variant) is optimal concerning movements of sets of nodes from one community to another. The Leiden algorithm introduces a refinement phase post-modularity optimisation to split and refine clusters, ensuring well-connected communities and higher quality clusters, particularly for larger networks.

All state-of-the-art community detection algorithms are stochastic, producing different partitions with different random seeds.

3.2.3.2 Phenograph pipeline

The PhenoGraph [10] pipeline which is adepted by Seurat includes three main steps: i) dimensionality reduction (Section 3.2.2), ii) graph construction, and iii) graph clustering.

A key aspect of the PhenoGraph pipeline is translating single-cell data into a graph that reflects phenotypic connections among cells, with edges quantifying similarity [10]. The final graph is computed in two steps. First, for each node, the k-nearest neighbours are identified using a distance metric (e.g., Euclidean or cosine). Edges are then drawn between each node and its k-nearest neighbours, forming a directed graph due to the asymmetry in neighbourhood relationships. This results in N sets, each representing the k-neighbourhood for a cell. A weighted graph can be constructed using the Jaccard similarity index (JSI) to evaluate the overlap between the k-neighborhoods:

$$W_{ij} = \frac{|v(i) \cap v(j)|}{|v(i) \cup v(j)|}$$

where $v(i)$ and $v(j)$ are the k-neighborhoods of cells i and j, respectively. Weights range from 0 to 1, with 0 indicating significant divergence and 1 indicating substantial overlap [10]. The resulting graph is known as a shared nearest neighbor (SNN) graph.

The final step involves identifying clusters using a community detection algorithm. The original PhenoGraph paper recommends the Louvain approach as the default. Seurat [11] also provides implementations of Louvain with multi-level refinement, SLM, and Leiden.

3.2.3.3 Assessing clustering stability in an element-wise fashion

Gates et al. [14] recently proposed the element-centric similarity (ECS), a paradigm shift from cluster-centric evaluation to individual elements assessment.

ECS constructs a cluster affiliation graph, a bipartite graph with one vertex set representing elements and the other representing clusters. An edge connects an element to a cluster if the element is a member of that cluster. The cluster-induced element graph, representing relationships between elements sharing at least one cluster, is derived by projecting the cluster affiliation graph onto the element set. Edge weights between two elements are proportional to their shared memberships.

Personalized PageRank (PPR), or affinity matrix, is used to analyze the cluster-induced element graph. The PPR of a node i to others is computed as:

$$p_i = (1 - \alpha)v_i + \alpha p_i W,$$

where v_i is a one-hot encoded N-vector with 1 at the ith entry, α controls the influence of overlapping and hierarchical clusters, and W is the weighted adjacency matrix of the cluster-induced element graph.

For disjoint partitions, the PPR formula simplifies to:

$$p_{ij} = \begin{cases} 0, & \text{if } i \text{ and } j \text{ are not members of the same cluster,} \\ \frac{\alpha}{|C_\beta|}, & \text{if } i, j \in C_\beta, \\ 1 - \alpha + \frac{\alpha}{|C_\beta|}, & \text{if } i = j. \end{cases}$$

The ECS score for each element is calculated using the $L1$ distance between the affinity matrices of two clusterings (A and B):

$$S_i(A, B) = 1 - \frac{1}{2\alpha} \sum_{j=1}^{N} \left| p_{ij}^A - p_{ij}^B \right|.$$

The overall ECS score is the average of individual scores. The element-centric consistency (ECC), or frustration, is calculated for a set of clusterings $R = \{R_1, \ldots, R_T\}$ to assess the consistency for element v_i.

$$\frac{2}{T(T-1)} \sum_{j=1}^{T} \sum_{k=1}^{j-1} S_i(R_k, R_j),$$

where T represents the number of clusters. Identifying elements with low ECC across numerous partitions that generally exhibit high median ECC (stable) can reveal specific samples that share characteristics with multiple clusters (i.e., fuzzy cells).

The ClustAssess [9] package provides optimised R and Python implementations of ECS and ECC.

3.2.3.4 Finding optimal parameter values using ClustAssess

In a traditional setting, most users might proceed with results derived from the default random seed, neglecting potential instability. However, as the ClustAssess [9] paper highlighted, there is substantial variability across different seeds across resolutions. This indicates that relying on a single, arbitrary clustering could lead to unreliable partitions which can profoundly impact downstream analyses, including differential expression analysis, trajectory inference, and cell-type annotation.

Apart from the community-detection algorithms being stochastic, decisions at each step of the

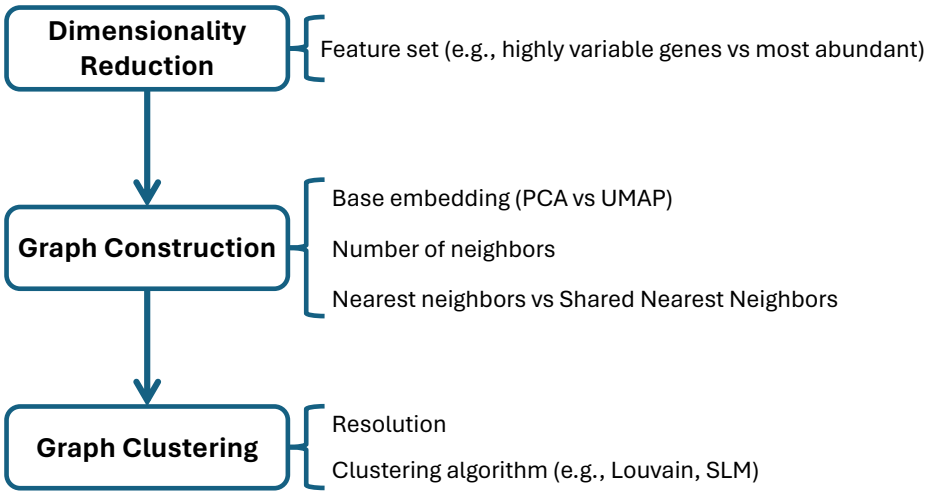


Figure 3.2.1: Decisions required at each step of the PhenoGraph pipeline. ClustAssess helps identify optimal parameter values for each of these decisions.

PhenoGraph pipeline can significantly impact the clustering outcome (Figure 3.2.1). Before performing dimensionality reduction, the feature set, including the type (HV/MA) and the number of features, must be decided. For graph construction, choices include the base embedding (PCA vs UMAP), the number of neighbours, and whether to use a weighted graph (SNN) or nearest neighbours (NN). Finally, for community detection, the clustering algorithm and the resolution value, which determines the number of clusters, must be selected.

ClustAssess [9] using ECC evaluates the robustness of clustering outcomes against technical variations, such as changes in the random seed and can be used to determine an optimal range of parameters for all the decisions affecting the clustering, ensuring that true biological signal is accurately reflected in the results.

3.3 Fuzzy Clustering

Fuzzy clustering allows the assignation of data-points to multiple clusters, allocating a probability to each assignment [17]. This facilitates the identification of transitional/intermediate cells, often overlooked by traditional crisp methods and matches developmental biology, where cells often exist in a continuum of states [38]. Current research on fuzzy community-detection

algorithms is sparse, with most contributions comming from Gutierrez et al. [39, 40, 41]. Their work, however, diverges from traditional fuzzy clustering; instead, it mainly focuses on adapting the Louvain algorithm to consider additional information (they refer to this as fuzzy measures) about the relation among the nodes that are not present in the structure.

3.3.1 New approach on community detection fuzzy clustering

As discussed in the introduction, cells often exist in a continuum of states (i.e., a discrete assignation of cells does not always accurately reflect their biological state). Owing to their stochasticity, repeatedly running the current state-of-the-art community-detection algorithms n times can yield n different partitions that might not converge to the same conclusion. If we try to characterise the fuzziness in such a scenario, our results would have biological and technical variability. However, if we first optimise the signal using a tool like ClustAssess [9] and get a highly reliable and reproducible set of partitions, any remaining variability is more likely to reflect genuine biological heterogeneity rather than technical noise and could thus be highly informative. To develop fuzzy community-detection clustering, a stochastic repeated clustering approach was followed, where we are repeatedly running the algorithm with the same input but different random seed.

The `run_fuzzy_clustering` function was developed, which initially runs ClustAssess's [9] `assess_clustering_stability`. This function repeatedly executes community detection clustering using Seurat's `FindClusters` with different random seeds, resolutions, and clustering algorithms and then groups the results by configuration (algorithm, resolution, k). To effectively address the issue of label switching—where corresponding clusters are assigned different labels across runs—a label reconciliation process was developed. This label reconciliation process, outlined in Algorithm 1, operates on the extracted list of partitions grouped by configuration to achieve consistent labelling. The main steps are the following:

1. **Identification of the reference partition:** The partition with the highest frequency[1] of occurrence is selected as the reference.

2. **Construction of the cost matrix:** While iterating through the partitions, a contingency table is first created to determine the intersection counts between clusters of the current partition and those of the reference partition. Using the JSI as a measure, we calculate the JSI for each cluster pair between the two partitions. The cost matrix is then derived by subtracting the JSI from 1, transforming it into a dissimilarity score where higher values indicate lower similarity.

[1] ClustAssess's `assess_clustering_stability` function quantifies how frequently the specific partition occurs, by counting how many times it is exactly replicated in the configuration.

3. **Reconciling labels:** The Hungarian algorithm is applied to the cost matrix to determine the optimal matching of cluster labels between the current and reference partitions.

Once we have the reconciled partitions grouped by configuration the membership degree, hard clusters, and consensus matrix can be calculated.

Algorithm 1 Reconcile Labels Across Partitions

```
function RECONCILE_LABELS(partitions)
    labels_list ← list of labels from each partition in partitions
    reference_partition ← labels_list[1]
    adjusted_labels_list ← list containing reference_partition

                                    ▷ Reconcile the labels in other partitions with the reference
    for i ← 2 to length of labels_list do
        cost_matrix ← create_cost_matrix(labels_list[i], reference_partition)
        optimal_match ← hungarian_algorithm(cost_matrix)
        adjusted_labels_list[i] ← adjust_labels(labels_list[i],optimal_match)
    end for
    return adjusted_labels_list
end function
```

3.3.2 Membership degree and Hard Clusters

Algorithm 2 Get Degree of Membership

```
1: function GET_DEGREE_OF_MEMBERSHIP(partitions_list, sample_names)
2:     membership_matrix ← empty matrix
3:     count ← ZEROMATRIX(num_samples, num_clusters)

       ▷ Count how many times each sample gets assigned to each cluster across partitions
4:     for each partition in partitions_list do
5:         for each sample in partition do
6:             Increment count[sample][cluster]
7:         end for
8:     end for
                                          ▷ Calculate probabilities of assignment
9:     total_partitions ← length of partitions_list
10:    for each sample, cluster pair do
11:        membership_matrix[sample][cluster] ← count[sample][cluster] / total_partitions
12:    end for
13:    return membership_matrix
14: end function
```

For each data point x_i, we track its assignment to various clusters across all runs. Specifically, for each cluster j, we count the number of times x_i is assigned to cluster j across multiple executions. This frequency, denoted as n_{ij}, quantifies x_i's association with cluster j. The degree of membership u_{ij} of data point x_i in cluster j is subsequently calculated as the proportion of times x_i is assigned to cluster j out of the total number of runs (M). Mathematically, this is expressed as:

$$u_{ij} = \frac{n_{ij}}{M}$$

This formulation results in a membership matrix $U = [u_{ij}]_{N \times C}$, where N is the number of data points and C is the number of clusters. The resultant membership matrix offers insights into the stability of cluster assignments and the presence of transitional states within the data.

To determine the most probable cluster for each data point x_i, we can identify the cluster j for which the membership degree u_{ij} is maximal for each x_i. Mathematically, the hard cluster assignment c_i for each data point x_i is given by:

$$c_i = \arg\max_j u_{ij}$$

The membership degree calculation was implemented through the `get_degree_of_membership` function which takes a list of partitions (from the same configuration) and outputs a membership

degree matrix. The core logic of this function is presented in Algorithm 2.

Hard cluster assignments are determined using the `get_hard_clusters` function (Algorithm 3) which takes a membership degree matrix as input and assigns each sample to the cluster with its highest membership percentage.

Algorithm 3 Get Hard Clusters

```
1: function GET_HARD_CLUSTERS(degree_matrix)
2:     hard_clusters ← []                                  ▷ Empty list
3:     for each row in degree_matrix do
4:         hard_clusters[row] ← ARGMAX(degree_matrix[row])
5:     end for
6:     return hard_clusters
7: end function
```

3.3.3 Consensus matrix

In 2017, Kiselev et al. introduced SC3 [6], a method that computes a consensus matrix from multiple k-means clustering solutions and subsequently clusters this matrix into k groups via complete-linkage hierarchical clustering. Their approach demonstrated that consensus clustering significantly enhances the robustness and accuracy of clustering, but also visualisation of the matrix allows for the assessment and visualisation of clustering stability (and subsequently fuzziness) by analysing the co-clustering of samples among multiple solutions.

In *flufftail*'s implementation, the computation of the consensus matrix starts with the execution of the `run_fuzzy_clustering` function. This returns a set of partitions, which are organized according to the algorithm used, resolution, and number of clusters. For each partition in a given configuration, an $n \times n$ co-association matrix is created, assigning a value of 1 if items i and j belong to the same cluster, and 0 otherwise. These matrices are then aggregated and normalized to yield the final matrix, which represents the proportion of times each pair of points are assigned to the same cluster. This frequency, denoted as s_{ij}, is calculated as:

$$s_{ij} = \frac{1}{M} \sum_{m=1}^{M} \delta_m (x_i, x_j)$$

where $\delta_m (x_i, x_j)$ equals 1 if x_i and x_j are in the same cluster in partition m, and 0 otherwise.

The consensus matrix is computed using a hybrid R-C++ approach for efficiency, with the core logic implemented in C++. Algorithm 4 outlines the main logic.

Algorithm 4 Compute Consensus Matrix: Pseudocode for C++ (0-indexed).

```
1: function COMPUTE_CONSENSUS_MATRIX(clusterings)
2:     num_samples ← ROWS(clusterings)
3:     consensus_matrix ← ZEROMATRIX(num_samples, num_samples)

                    ▷ Calculate co-clustering behaviour for each pair of samples across iterations
4:     for k ← 0 to COLS(clusterings) − 1 do
5:         for i ← 0 to num_samples − 1 do
6:             for j ← 0 to num_samples − 1 do
7:                 if clusterings[i, k] = clusterings[j, k] then
8:                     consensus_matrix[i, j] ← consensus_matrix[i, j] + 1
9:                 end if
10:            end for
11:        end for
12:    end for
13:    consensus_matrix ← consensus_matrix / COLS(clusterings)          ▷ Normalise
14:    return consensus_matrix
15: end function
```

3.4 New approach for identifying fuzzy cells

Fuzzy cell prevalence at specific cluster boundaries can indicate varying degrees of overlap and shared characteristics between clusters. The location of the most fuzzy cells can thus help pinpoint areas where cluster boundaries are less distinct, suggesting plasticity zones within the clustered data.

We are considering three methods for identifying the most fuzzy cells: cells with low ECC values across different iterations, cells with high entropy on their co-association with other cells from the consensus matrix, and cells with high entropy on their membership degree matrix.

3.4.1 Low ECC values across iterations

After multiple executions of the clustering algorithm, we observe a variety of slightly different partitions, attributable to inherent stochasticity. As detailed in Section 3.2.3.3, the ECC metric can be computed across a set of partitions. According to the authors [14], higher ECC values signify a more consistent assignment of data points to the same cluster across different runs. Given ECC's focus on individual data elements rather than clusters, we can identify *both* regions and specific cells with high levels of inconsistency/fuziness by examining the lower values in the resulting ECC vector.

3.4.2 High entropy on co-association from the consensus matrix

An alternative approach developed applies Shannon's entropy from information theory on each cell's consensus matrix row[2] to calculate the variability in each cell's co-clustering behaviour:

$$H(p_i) = -\sum_j p_{ij} \log(p_{ij})$$

where p_{ij} represents the frequency that cell i co-clusters with cell j. Cells exhibiting high entropy indicate higher instability, reflecting significant inconsistency in their co-clustering behaviour. Appendix D.1 shows that extremely fuzzy cells (highest entropy) have co-clustering probabilities between 0.25 and 0.5 with most other cells, signalling inconsistent co-clustering behaviour. Conversely, cells exhibiting the lowest entropy feature co-clustering probabilities that are densely concentrated around 0 and 1.

3.4.3 High entropy on the membership degree matrix

The entropy approach can also be applied to the membership degree matrix where entropy for each row of U quantifies the uncertainty in a data point's cluster membership. The entropy H for a data point's memberships is given by:

$$H(u) = -\sum_j u_{ij} \log(u_{ij})$$

where u_{ij} is the membership degree of data point i in cluster j. High entropy values indicate that a data point is spread across multiple clusters, suggesting a lack of clear affiliation. Conversely, low entropy values indicate strong affiliation to a particular cluster. This is empirically confirmed by the experiments presented in D.2.

The `calculate_entropy` function was developed which computes Shannon's entropy for each row of an input matrix.

[2]due to the consensus matrix being symmetric, this could equivalently be applied to columns.

3.5 Novel identification of major regulatory hubs using fuzzy clustering

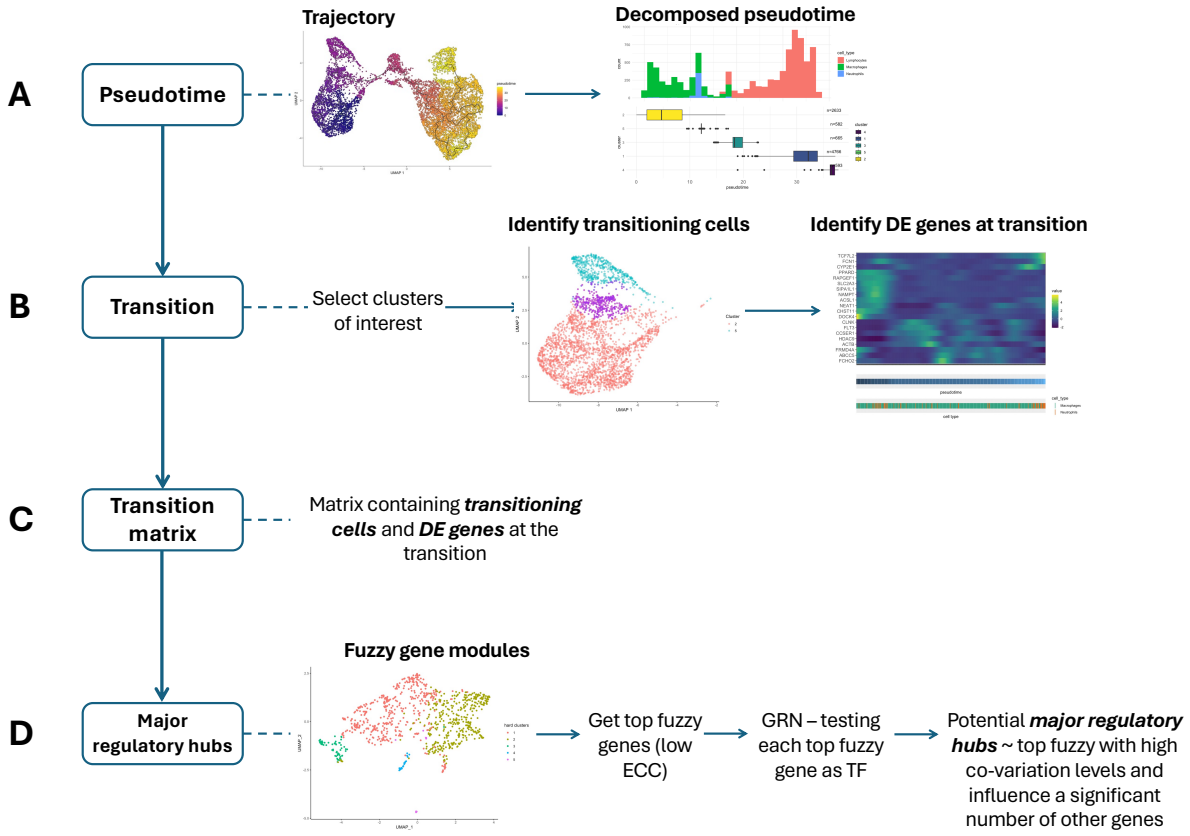


Figure 3.5.1: Overview of the developed process for identifying potential major regulatory hubs through fuzzy clustering.

Major regulatory hub genes are central elements within GRNs that control the expression of numerous target genes [24], playing critical roles in various biological processes, including development, differentiation, and disease regulation [42]. These hub genes, often transcription factors, are essential for maintaining the intricate balance of gene interactions and ensuring the proper functioning of cellular mechanisms [43]. Moreover, they serve as promising targets for therapeutic interventions, as modulating these genes can significantly impact disease pathways and improve treatment outcomes [44].

Figure 3.5.1 provides an overview of the developed process for identifying major regulatory hubs. The following subsections discuss in detail the implementation of this approach.

3.5.1 Repurposing pseudotime for the identification of transitioning clusters

The process starts by first computing the pseudotime trajectory. In biological processes such as differentiation, cells transition through a continuum of states. Understanding cell fate from single-cell data is challenging because the data represents static snapshots at specific time points. Pseudotime analysis (or trajectory inference) methods order cells along these trajectories by gene expression similarities, aiding in the understanding of cell development and disease progression.

Trajectory inference approaches include Slingshot [45], which performs well for simple topologies, and RaceID/STEMID [46] which is suitable for complex trajectories [15]. The most popular package offering trajectory inference is Monocle3 [12]. It starts the process by first projecting cells onto a lower-dimensional space using UMAP. It then clusters similar cells using Louvain and forms 'supergroups' by merging adjacent groups. Finally, it identifies potential cell trajectories by learning a principal graph from the reduced graph embedding, which is a branching curve representing the core structure of the dataset. Monocle3 employs reverse graph embedding (RGE) to learn both the principal graph and a function mapping low-dimensional trajectory points back to the original high-dimensional space [47].

In *flufftail*, trajectory inference can be performed using the `get_trajectory_graph_nodes` wrapper function. It first converts a Seurat object to a Monocle3 cell data set (*cds*) object, then learns the trajectory using Monocle3's `learn_graph` function, and identifies graph nodes with more than two edges, which can represent branching points in the trajectory, indicating key decision points in the process. The pseudotime ordering is subsequently computed by specifying a node with more than two edges and calling the `order_cells` function.

An alternative approach has been developed for ordering the trajectory graph by first converting the Seurat object to *cds* and then using the `order_cells_by_start_end_genes` helper function. This function allows the user to specify start genes, end genes, and threshold quantiles of expression, and processes the *cds* by selecting cells that express predefined start and end genes above specific quantile thresholds. This is especially beneficial as biologists often want more control over where the pseudotime process should start and end. Each gene's threshold is individually calculated based on its expression distribution, requiring a cell to express the gene above a certain quantile of its overall expression to qualify for pseudotime analysis. The cells are then filtered through their UMAP embeddings, and a principal graph is constructed to delineate sub-trajectories between the identified start and end cells. Additionally, the function allows for a specified number of relax genes, determining how many genes can fall below their quantile thresholds in both start and end gene sets. This added flexibility allows the inclusion of cells in pseudotime analysis, even if some genes do not meet the expression thresholds.

Once the pseudotime trajectory has been calculated, we could compare the pseudotime values obtained using all the cells with those obtained when the top $x\%$ of fuzzy cells are excluded (complement). If the ordering is preserved, this serves as a robustness check, indicating that the true biological signal was captured when computing the pseudotime and that it is robust. Furthermore, the pseudotime can be decomposed, enabling a comparison between the clusters and cell types and also highlighting the sequence of cluster transitions during the cell differentiation process (Figure 3.5.1 A). The `get_decomposed_pseudotime_plot` function was developed to take the *cds* object with the learned trajectory and visualise the distributions of cell types and hard clusters along the pseudotime axis.

3.5.2 Novel characterisation of cell continuity using fuzzy genes

By isolating cells located at the transition area between two clusters of interest that have adjacent pseudotime distribution and analysing their differential expression patterns, we can identify a set of differentially expressed (DE) genes with dynamic expression changes that orchestrate the transition between the clusters (Figure 3.5.1 B). Programmatically, this can be done by subsetting the *cds* and thus the pseudotime trajectory with cells located at the transition and then calling Monocle3's `graph_test` function. The top DE genes can be identified by sorting in descending order the Moran's I test value returned (spatial autocorrelation test). Importantly, gene set enrichment analysis (GSEA) can then be performed using `gprofiler2` to identify enriched pathways by the DE genes.

Constructing a "transition matrix" that includes the transitioning cells and the identified DE genes (Figure 3.5.1 C) serves as the input for gene module clustering, which can then provide deeper insights into the regulatory mechanisms driving the transition. Co-expression modules can infer regulatory connections between transcription factors and potential target genes [48, 49]. However, a standard gene module clustering approach assigns each gene to a single module. By computing gene module partitions multiple times (using different random seeds), fuzzy clustering techniques, as detailed in Section 3.3 can be applied. This allows each gene to be associated with multiple modules, reflecting its involvement in various biological processes [50, 20]. This probabilistic approach indicates that genes participating in multiple biological modules (i.e., fuzzy genes), are likely key drivers of the dynamic changes observed in pseudotime ordering and related cellular processes [50].

In *flufftail*, the `gene_module_fuzzy_clustering` function was developed, taking the transition matrix, desired algorithm, resolution sequence, and number of iterations as input. It returns the gene module clustering partitions, ECC, degree matrix, and hard clusters, grouped by the clustering settings (algorithm, resolution, k).

3.5.3 New approach for the identification of major regulatory hubs

Algorithm 5 Get summary of connections per gene in GRN

function GET_SUMMARY_CONNECTIONS_PER_GENE(grn, top_fuzzy_list)
  $target\_stats \leftarrow \{\}$ ▷ Empty dictionary
  for each unique $target$ in grn **do**
    $t\_data \leftarrow$ subset of grn where target gene = $target$
    $target\_stats[target].med \leftarrow$ median of $t\_data.importance$
    $target\_stats[target].q3 \leftarrow$ 75th percentile of $t\_data.importance$
  end for
  $table\_conn \leftarrow []$ ▷ Empty list
  for each $gene$ in top_fuzzy_list **do**
    $sub\_net \leftarrow grn[grn.TF = gene]$
    $sub\_net\_stats \leftarrow$ join sub_net with $target\_stats$ on target
    $pct\_above\_med \leftarrow \text{mean}(sub\_net\_stats.imp > sub\_net\_stats.med)$
    $pct\_above\_q3 \leftarrow \text{mean}(sub\_net\_stats.imp > sub\_net\_stats.q3)$
    $summary \leftarrow \{gene, pct\_above\_med, pct\_above\_q3\}$
    Append $summary$ to $table\_conn$
  end for
  return $table\_conn$
end function

Fuzzy genes that have high co-variation levels and influence a significant number of other genes through strong regulatory interactions can be identified as potential major regulatory hub genes and their widespread influence makes them prime candidates for targeted therapeutic interventions. To identify gene hubs, we can construct a GRN and test each fuzzy gene (e.g., top 10% ranked by lowest ECC) as a transcription factor (TF) and all DE genes as potential target genes.

GRN inference methods are categorised into information theory (IT) and machine learning (ML) approaches. IT-based methods rely on mutual information to assess pairwise gene interactions, with popular algorithms including CLR [51] and ARACNE [52]. ML-based approaches use tree-based implementations. The initial and still state-of-the-art model that achieved the best performance on a DREAM challenge dataset is GENIE3 [53], a random forests (RF) based approach that uses variable importance to assess co-variation. More recently, GRNBoost2 [54] was introduced as a regression-based GRN inference method based on the GENIE3 concept, training a tree-based regression model for each gene to predict its expression using candidate transcription factors (TFs). GRNBoost2 employs gradient boosting, building models by additively combining weak learners (shallow decision trees) with a regularized stochastic variation and early stopping based on out-of-bag improvement estimates. This self-tuning mechanism ensures each regression model contains only the necessary number of trees, preventing unnecessary computational workload. GRNBoost2 achieves comparable performance to GENIE3 on

the DREAM benchmark while offering significant speed-ups. Due to its efficiency and high performance, the GRNBoost2 method was selected for GRN inference. As GRNBoost2 is a Python package (offered by the Arboreto [54] package), the reticulate [55] package in R is used to interface between the two languages. The input is an expression matrix and a list of TFs.

The method returns a table containing three columns: TF, target, and Importance, indicating how strongly the TF covaries with the expression of the target gene. To identify fuzzy genes that influence a significant number of target genes, we first calculate the median and third quartile (Q3) Importance for each target gene across all its TFs (referred to as target-specific Importance). To identify TF genes that are highly involved in regulatory dynamics (Figure 3.5.1 D), we count, for each TF, the percentage of target genes for which it has an Importance value exceeding that target gene's median and third quartile Importance. This process is implemented in the `get_summary_connections_per_gene` function, which takes the GRNBoost2 output and a list of top fuzzy genes, and returns a dataframe with the statistics for each gene in the list. Descendigly sorting the top fuzzy genes by this metric can identify key genes driving the dynamics. The logic is outlined in Algorithm 5.

3.6 First approach for exploring GRN dynamics on single-cell datasets

Once potential hub genes are identified, we delve deeper into the study of GRN dynamics of these hubs at various pseudotime stages. By analysing the interactions and changes occurring between network topologies at specific points, such as within cluster A, at the midpoint of the transition, and within cluster B, we gain profound insights into the evolution of regulatory dynamics.

The `get_GRN` function was developed to infer GRNs from an expression matrix and a list of *a priori* curated TFs using GRNBoost2 (chosen for its efficiency and performance).

To investigate the dynamics/transition between two homogeneous clusters, in the current setup, three GRNs can be constructed. Each GRN is based on an expression matrix containing the DE genes but differs in its cell selection:

1. A percentage of cells around the centroid of Cluster A (homogeneous cluster)
2. The selected transitioning cells
3. A percentage of cells around the centroid of Cluster B (homogeneous cluster)

This approach allows us to observe which genes are most influenced by the regulatory hubs over the pseudotime ordering, providing a dynamic snapshot of how regulatory mechanisms shape

the GRNs throughout the differentiation process. Understanding the mechanistic details of the GRN rewiring process is essential for selecting targets for drug testing [56, 57] and for focused perturbations such as those induced by antisense oligonucleotides (ASOs) [58, 59].

To visualise key GRN interactions, the `visualise_GRN` function was developed. This function accepts a GRNBoost2 output table, a quantile threshold to filter interactions by the distribution of *Importance* and a specified number of top interactions to display. Additionally, it identifies and visualises interactions involving common targets among the selected hubs. The `create_combined_GRN` function takes three GRNBoost2 visualisation networks returned by the `visualise_GRN` function and highlights the common and specific interactions of each step using a colour scheme, highlighting the dynamic changes.

Chapter 4

Results

As part of this dissertation, I developed *flufftail*—a comprehensive R package for single-cell data analysis, agnostic of modality, supporting in-depth biological research and data-driven hypothesis generation through a multi-faceted approach.

First, the stochasticity of community-detection algorithms was harnessed. Using a repeated stochastic clustering approach and developing functions for reconciling and consolidating the partitions, fuzzy clustering of cutting-edge clustering algorithms is provided in a streamlined and efficient manner. The computation of membership degrees, assessment of hard clusters, summarisation of the consensus matrix, and the identification of fuzzy cells/genes using three approaches (ECC, entropy on consensus, and entropy on degree matrix) enables an in-depth characterisation and visualisation of fuzziness (objectives 1 & 2).

A new methodology was developed for the identification of major regulatory hubs in a data-driven way (objective 3). This approach relies on identifying fuzzy DE genes across the transition area of two clusters that also have a high covariation (assessed using tree-based regressions) with a large percentage of other DE genes that are influencing the transition.

The project proposed and implemented a new approach to characterise GRN dynamics and the evolution of regulatory interactions across an ordering such as pseudotime (objective 4). I computed the GRNs for "major regulatory hub" genes for three different bins across the transition between two clusters; using the information from adjacent pseudotime distributions, the dynamics (i.e., strength of interaction and topology) of the networks can be examined as the process evolves. This is the first time GRN dynamics across the pseudotime process have been studied for single-cell data and represents a significant step forward in understanding the signals controlling the plasticity and transdifferentiation of cells.

A key advantage of the package/framework is that it facilitates interactive analyses via an R Shiny interface (objective 5), enabling easier and faster exploration and extending its utility to

a broad range of researchers, with varied skills and expertise.

In this chapter, the current clustering stability of the state-of-the-art approaches is showcased, highlighting the necessity to characterise the observed variability. Then, the *flufftail* framework and the novel methodologies developed are illustrated from the scope of the R Shiny interface. Finally, the performance of similar analyses through scripting using the package functions are briefly discussed.

4.1 Clustering stability using state-of-the-art approaches. Limitations

This section uses the *Immune* subset to showcase the stability/variability of current state-of-the-art approaches, making a case for optimising the input. It also includes and comments on the results of applying the ClustAssess pipeline to the *Immune* Seurat object.

4.1.1 Stability without tuning

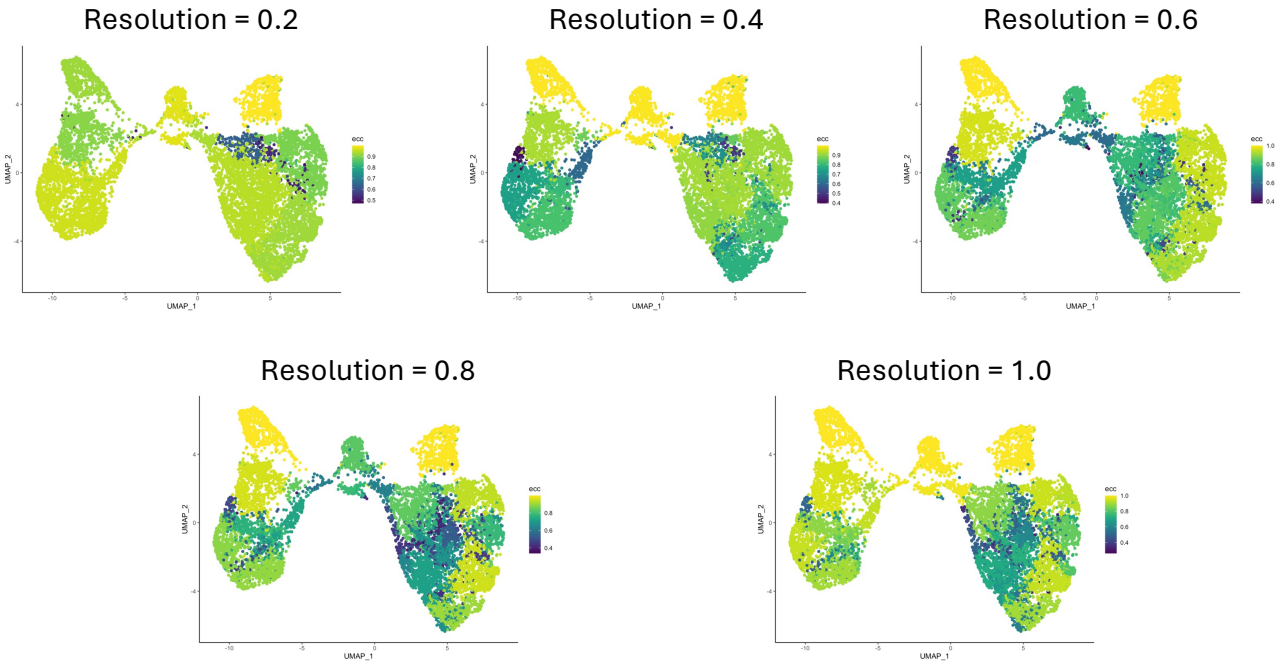


Figure 4.1.1: ECC represented on the colour gradient, across 30 repetitions of Louvain community detection across different resolution values. The different panels illustrate the effect of increased resolution (i.e. increased number of clusters); I note the presence of increased fuzziness on the central bridge, and on ridges in either left or right islands. This variability in transcriptomic signature is in line with the biological interpretation.

Figure 4.1.1 illustrates the stability of partitions using ECC across different resolution values (and therefore different number of clusters) obtained when applying the Louvain algorithm (Seurat implementation), over 30 runs/iterations with different random seeds. The darker-coloured points represent cells with lower ECC and thus considerable instability, highlighting the variability inherent in community detection methods.

These results underscore: 1) the intrinsic stochasticity in state-of-the-art community detection algorithms and the subsequent information loss (e.g., cells frequently switching clusters, marker genes underlining the biological role of the clustered cells) that could be leveraged in a fuzzy clustering context; and 2) without optimising the input (using ClustAssess) both technical and biological variability co-exist.

4.1.2 Applying ClustAssess

Table 4.1.1 summarises the values tried for each step of the PhenoGraph pipeline using ClustAssess on the *Immune* Seurat object.

Pipeline step	**Decision**	**Values tried**
Dim. reduction	Feature set	HV & MA - 1000, 2000, 3000
Graph construction	Base embedding Number of neighbours Graph type	PCA, UMAP 5 to 30 (5 increments) NN, SNN
Graph clustering	Clustering algorithm Clustering resolution	Louvain, Louvain-refined, SLM, Leiden 0.1 to 1.0 (0.1 increments)

Table 4.1.1: Data-driven options assessed for the *Immune* dataset, using the ClustAssess framework.

Figure 4.1.2 panels A and B correspond to the feature set and graph type analysis. Panels C and D demonstrate the clustering algorithm analysis.

While *flufftail* and fuzzy clustering are built on the stochasticity of the community-based clustering approaches, and thus benefit from variability, we should aim to harness this variability within an otherwise reliable clustering configuration that captures the true signal of the data (i.e., users should avoid selecting parameter values that result in highly unreliable results, since the variability would be a weighted average of technical and biologically valid variation). After evaluating the results from the ClustAssess analysis, the following settings have been chosen for the *Immune* dataset and applied using Seurat to obtain the final Seurat object:

- Feature set: HV 3000
- Base embedding: PCA

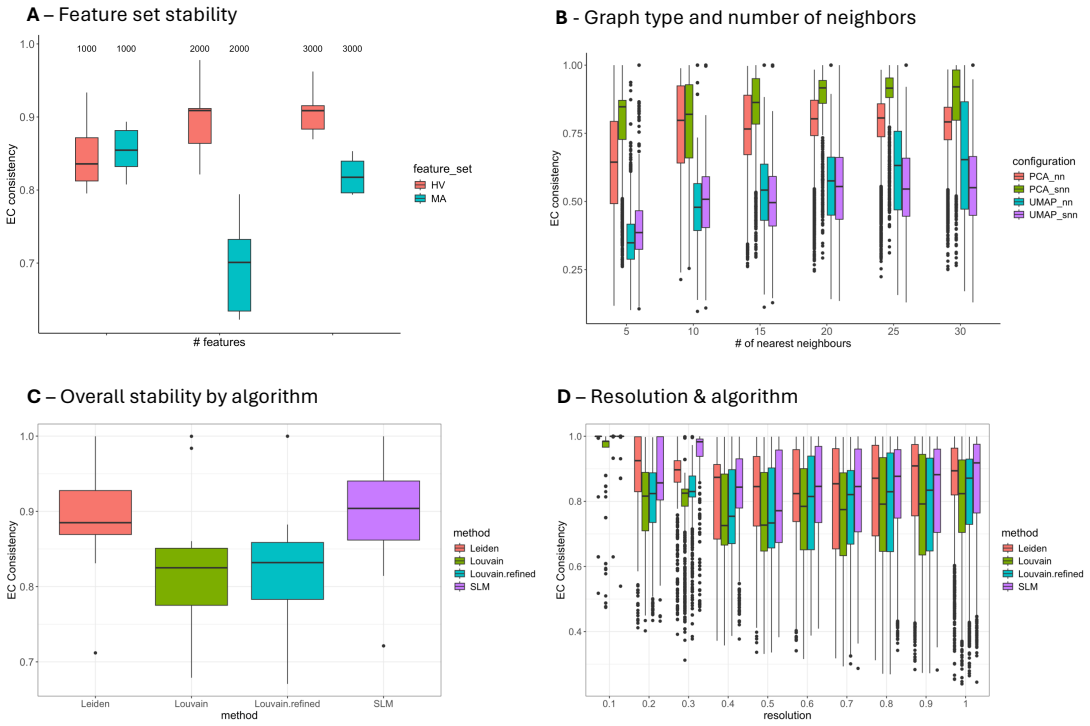


Figure 4.1.2: ClustAssess stability assessment results: (A) ECC at different number of HV & MA genes; (B) ECC across different embedding, graph types (NN vs SNN) and number of neighbours. (C) Overall ECC stability by algorithm across all the resolutions. (D) ECC stability by resolution and algorithm.

- Number of neighbours: 20
- Graph type: SNN
- Clustering algorithm: SLM

By using an assessment framework like ClustAssess, we identify instabilities and pinpoint their locations. However, no current tool exists that enables the characterisation (i.e., membership degrees, co-clustering stability, impact of the removal of fuzzy cells on the pseudotime ordering) of the fuzzy cells coming from a community-detection algorithm; importantly, the downstream impact of this variability on the characterisation of the dynamics driving the transition between two clusters displaying some fuzziness between them is now possible using *flufftail*.

4.2 *flufftail* framework

The first subsection briefly explains how to create an R object required to run the Shiny interface. The following sections detail each tab in the app, highlighting the novel analyses

enabled by the package. I am not interpreting the results in detail, as the primary goal of this section is to demonstrate the method's capabilities in a generalisable way.

4.2.1 Setup

We define the *fuzzy_object* as the required R object for constructing the R Shiny application. This object includes membership degree matrices, hard clusters, ECC values, and additional information such as cell types and sample names. Each clustering configuration on this object is organized by a specific combination of algorithm, resolution, and k value. There are two primary methods for creating this object:

1. **Manual:** When the clustering configuration is already known and the adjacency matrix has been precomputed (e.g., using Seurat's `Findneighbours`), the `run_fuzzy_clustering` function from *flufftail* should be used. The `prepare_fuzzy_object_for_shiny_manual` can then be used to create the *fuzzy_object*.
2. **Using ClustAssess object:** Perform ClustAssess's [9] automatic stability assessment (`automatic_stability_assessment`). This allows identifying desired configurations for all the steps affecting community detection clustering (e.g., embedding type, number of nearest neighbours, adjacency matrix graph type) [60] before executing the fuzzy analysis. The resulting object can then be passed to `prepare_fuzzy_object_for_shiny_after_CA` to create the *fuzzy_object*.

Detailed instructions on creating the Shiny app can be found in the "creating_shiny_app" vignette. For the following examples, the *fuzzy_object* was created using the manual method for the *Immune* dataset. The fuzzy clustering partitions were obtained by running `run_fuzzy_clustering` over 50 iterations using SLM, resolution 0.12[1], and k=5.

4.2.2 Exploration tab

The first tab of the Shiny application allows researchers to identify which clustering configuration exhibits fuzziness in a region of interest. Researchers can interactively select a specific region on the UMAP plot, prompting the application to generate a detailed statistical summary for the cells in that area across all configurations. This includes:

- 5+1 ECC figure summary

[1]Resolution range tested was between 0.02 and 0.2, incrementing by 0.02 each time. The range was selected after checking resolution and cluster correspondence and discussing with a biologist.

- Number of samples within the selection classified as ECC outliers (defined as those with an absolute z-score exceeding 3).
- The percentage of ECC outliers in that area compared to the outliers across all cells.

A clustering configuration with a high median ECC and a high percentage of ECC outliers in the selected area is especially interesting for further exploration.

4.2.3 Fuzzy Clustering tab

In the sidebar of the fuzzy clustering tab, researchers can select their desired configuration (algorithm, resolution, k) for subsequent analysis. Once selected, the tab displays UMAP plots that depict the hard clusters and cell types, allowing for direct comparison of clusters and annotations at the chosen settings (Figure 4.2.1 A). The tab also includes an interactive table displaying the membership degree matrix and allows researchers to compute, subsample, and visualise the consensus matrix (Figure 4.2.1 B & C).

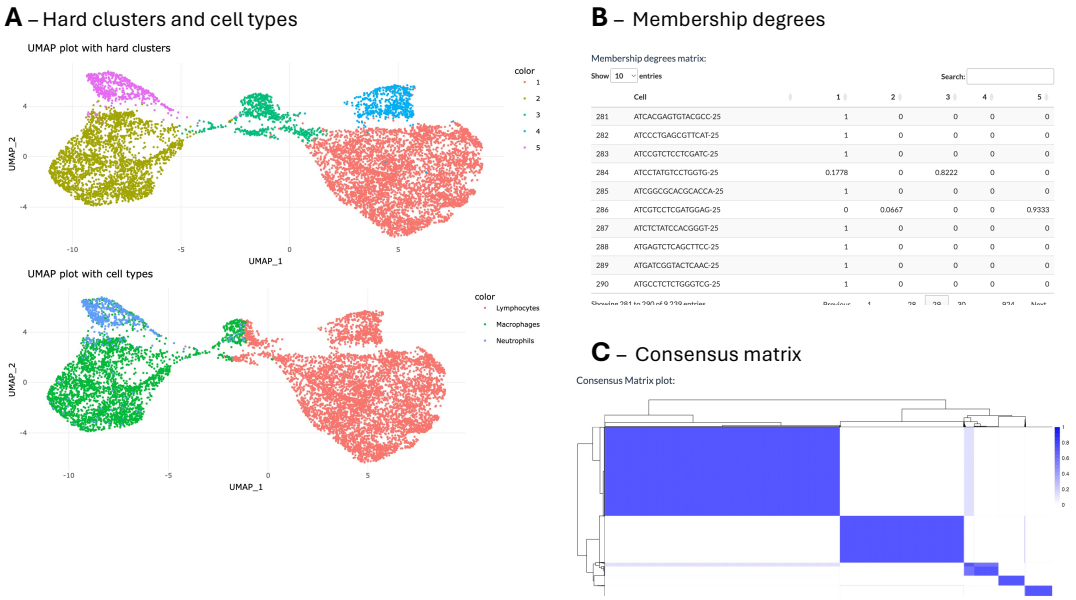


Figure 4.2.1: Main functionalities of the Fuzzy Clustering tab: (A) UMAP plots illustrating hard clusters and cell types, enabling direct comparison of clusters and annotations; (B) Interactive table displaying membership degrees; (C) Visualisation of the consensus matrix.

4.2.4 Fuzziness Exploration tab

This tab facilitates the identification of highly fuzzy cells through three metrics: ECC, entropy on consensus, and entropy on the degree matrix (Figure 4.2.2 A). The UMAP plots displaying the metrics can be zoomed in and cells can be highlighted interactively. Furthermore, researchers can use a sidebar slider to identify and select the top $x\%$ of fuzzy cells sorted by their preferred metric. In Figure 4.2.2 B, the top 2% of cells with the lowest ECC, which are primarily located at the transitions between clusters 2 and 3, as well as 3 and 5, are highlighted. This functionality aids in distinguishing the areas with the highest fuzziness and permits an analytical comparison between the ECC distributions of the top-selected fuzzy cells and the remaining (complement) cells (Figure 4.2.2 C).

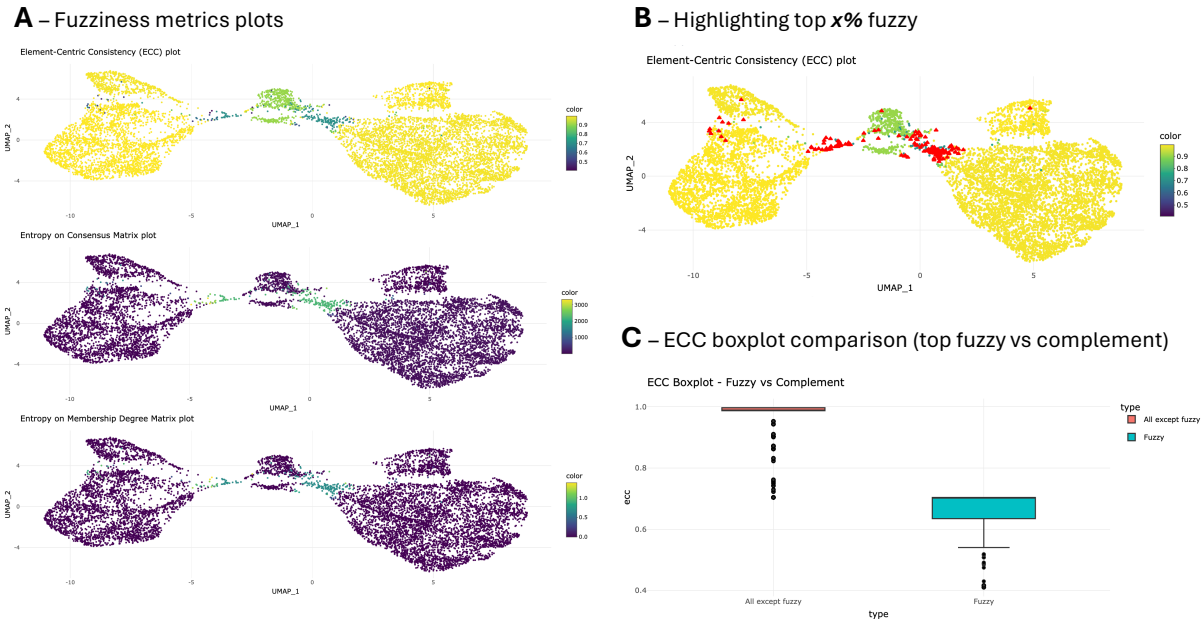


Figure 4.2.2: Main functionalities of the Fuzziness Exploration tab: (A) Fuzziness metrics plots displaying ECC, entropy on consensus, and entropy on the degree matrix; (B) Top x% of fuzzy cells highlighted; (C) ECC boxplot comparison of top fuzzy cells versus the complement.

4.2.5 Pseudotime Analysis tab

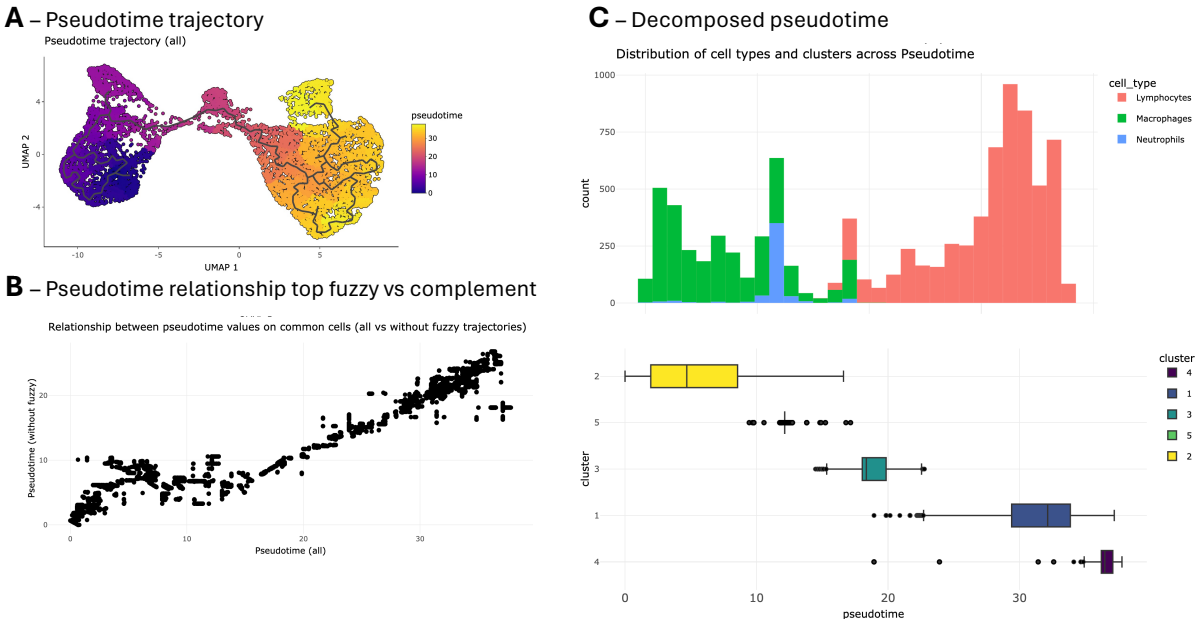


Figure 4.2.3: Main functionalities of the Pseudotime Analysis tab: (A) Pseudotime analysis trajectory; (B) Relationship of pseudotime values between the two trajectories (all cells vs excluding top fuzzy); (C) Decomposed pseudotime - showing the distribution of cell types and clusters across the pseudotime.

Researchers first must select a subset of cells (in this case, all cells were selected) to analyse by interactively dragging their mouse across the UMAP plot. Once cells are selected, pseudotime trajectories can be computed interactively using one of two methods:

1. **Monocle3 standard trajectory computation**: If start and end genes are not known, the trajectory can be computed using the standard approach of Monocle3:
    - First, the principal graph is learned.
    - A plot is then generated, displaying nodes with more than two connections (degrees).
    - Users can select a root node from a dropdown menu to initiate the pseudotime ordering.
2. **User-specified start and end genes**: If users have prior knowledge of start and end genes, which can be provided by a biologist or identified using the ClustAssess Shiny app [9], they can specify:
    - **Start and end genes:** These are critical for pseudotime ordering and can be selected in *flufftail* by typing gene names in a multi-selection box.
    - **Distribution threshold:** This sets the quantile threshold for each gene in the start and end gene sets. A cell must express the gene above this quantile of its overall

expression to be considered for pseudotime analysis. The threshold is calculated individually for each gene.

- **Number of relax genes:** This specifies the maximum number of genes that can fall below their respective quantile thresholds in a cell's expression profile for both the start and end gene sets. This flexibility allows for the inclusion of cells in the pseudotime analysis even if a certain number of genes don't meet the expression threshold.

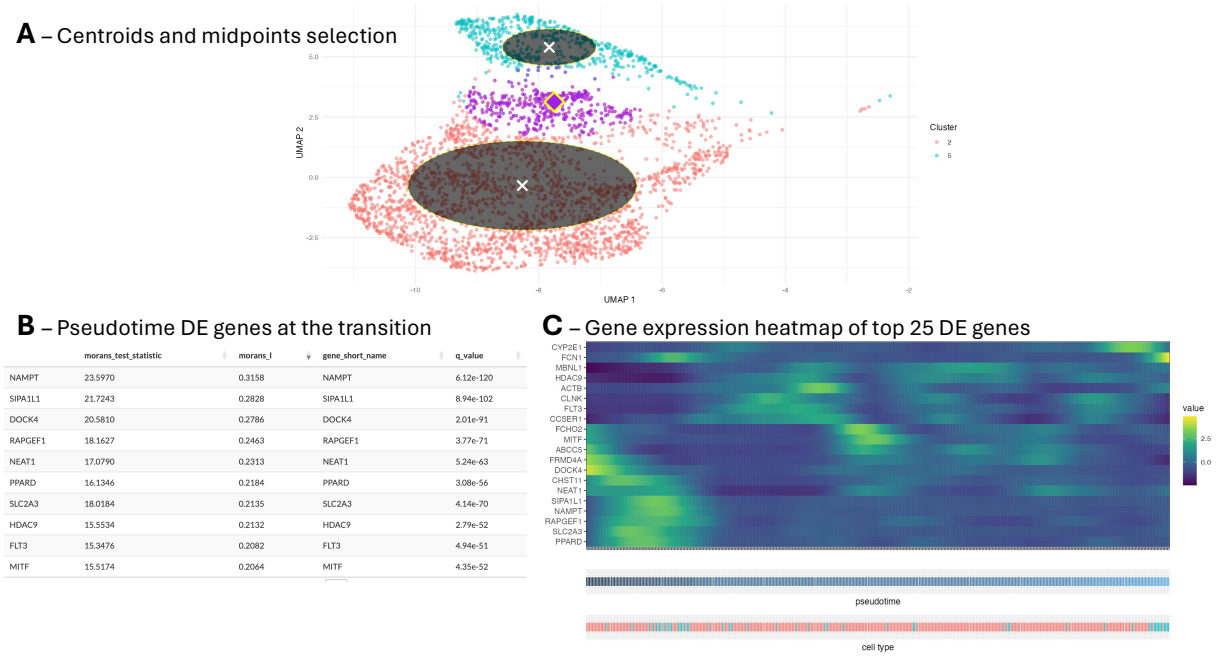
In this example, the pseudotime was computed using the standard approach. If the user has previously identified the top $x\%$ of fuzzy cells in the Fuzziness Exploration tab (Section 4.2.4), the application proceeds to learn two distinct pseudotime trajectories: one including all cells (Figure 4.2.3 A) and another excluding the identified fuzzy cells. This facilitates a nuanced comparison of the two trajectories by visualising both trajectory plots alongside a point plot that illustrates the correlation of pseudotime values among common cells (Figure 4.2.3 B). An alignment close to the diagonal suggests that the biological signal was robust, indicating that the cell ordering remained unbiased after removing the fuzzy cells. In this case, the ordering was largely preserved but showed some instability at the early stages of the pseudotime trajectory. If no top fuzzy cells are selected in the Fuzziness Exploration tab, the analysis is performed on all cells, providing a single trajectory.

The decomposed pseudotime plot (Figure 4.2.3 C) is an essential component of the developed methodology. It visualises the distribution of cell types and clusters along the pseudotime axis (derived from the trajectory learned on all cells). This visualisation is crucial for elucidating the sequence of cluster transitions throughout the progression of a biological process, providing insights into the temporal dynamics of cellular differentiation or state changes.

4.2.6 Differentially expressed genes tab

Once researchers identify two clusters transitioning from one to another they can use the DE genes tab to pinpoint genes that are differentially expressed during the transition. Subsequently, gene module clustering will be based on these findings.

The researcher must select the two clusters from a dropdown list, and three centroids will appear: two at the centres of the clusters and one at the midpoint between them. For the two cluster centroids, the percentage of points around each centroid to consider can be specified. For the midpoint centroid, representing the cells at the transition between the two clusters, the centroid's location can be adjusted (X, Y translation), and the number of cells to consider can be selected using sliders. Figure 4.2.4 A illustrates the analysis between clusters 2 and 5, which were selected because they largely correspond to the transition from Macrophages to



	morans_test_statistic	morans_I	gene_short_name	q_value
NAMPT	23.5970	0.3158	NAMPT	6.12e-120
SIPA1L1	21.7243	0.2828	SIPA1L1	8.94e-102
DOCK4	20.5810	0.2786	DOCK4	2.01e-91
RAPGEF1	18.1627	0.2463	RAPGEF1	3.77e-71
NEAT1	17.0790	0.2313	NEAT1	5.24e-63
PPARD	16.1346	0.2184	PPARD	3.08e-56
SLC2A3	18.0184	0.2135	SLC2A3	4.14e-70
HDAC9	15.5534	0.2132	HDAC9	2.79e-52
FLT3	15.3476	0.2082	FLT3	4.94e-51
MITF	15.5174	0.2064	MITF	4.35e-52

Figure 4.2.4: Main functionalities of the differentially expressed (DE) genes tab: (A) Plot of the two chosen clusters and their centroids, with 30% of cells selected around the cluster centroids and 1,600 cells around the midpoint (transition) centroid; (B) Interactive table displaying DE genes across the pseudotime at the transitioning cells around the midpoint centroid; (C) Heatmap of gene expression across the pseudotime of the top 25 DE genes.

Neutrophils. The selection included 40% of cells around each cluster's centroid and 400 cells around the midpoint centroid.

Genes that are driving the transition can be identified and presented in an interactive table (Figure 4.2.4 B) by subsetting the trajectory to include only the transitioning cells (purple colour) and then identifying DE genes using Monocle3's `graph_test` function. Thresholds for the q-value and Moran's I test can be set using a slider, giving the researcher control over the level of strictness they want to apply. The tab also allows for GSEA on the DE genes.

The gene expression of the top DE genes identified across the pseudotime can be visualized in a heatmap. Figure 4.2.4 C shows the gene expression heatmap of the top 25 DE genes, enabling the identification of genes that are most expressed at different points of the transition.

4.2.7 Gene Modules tab

Fuzzy gene module clustering can be performed on the pseudotime DE genes at the transition cells. The researcher can choose the resolution sequence, community-detection algorithm, and the number of iterations for the fuzzy gene module clustering. Once the clustering finishes, a table reporting the number of partitions and a 5+1 figure ECC summary for each configuration (resolution, k) is displayed. The user can select their desired resolution and k to obtain the

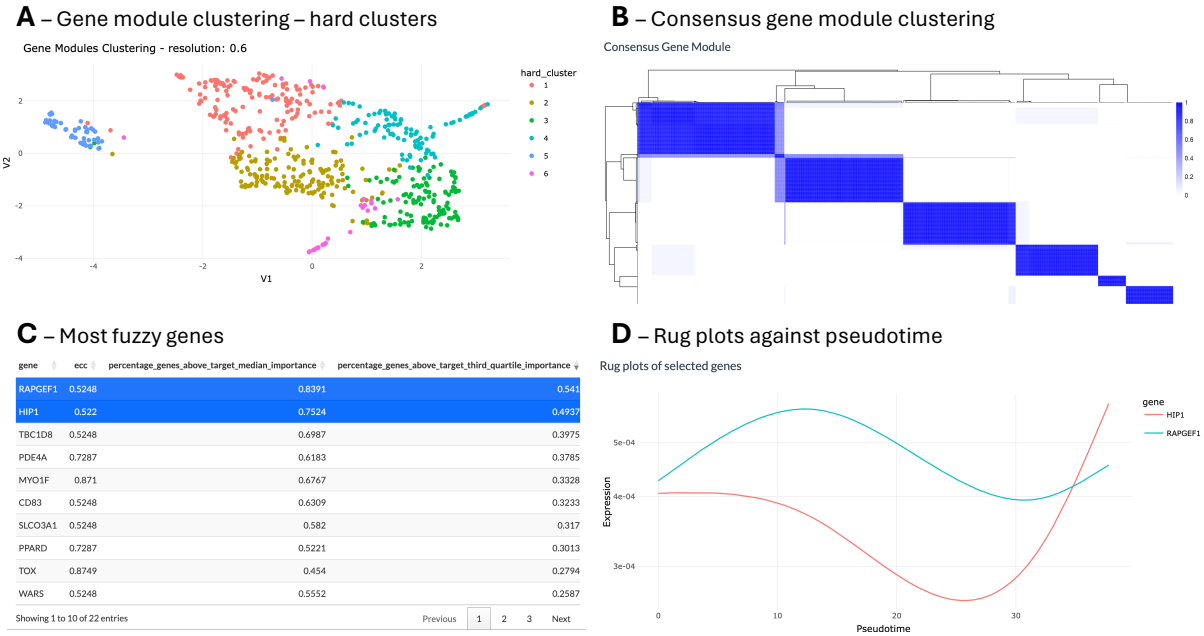


Figure 4.2.5: Main functionalities of the Gene Modules tab: (A) Hard cluster plots at resolution = 0.8 and $k = 4$; (B) Consensus matrix visualisation of the gene module clustering; (C) Interactive table containing top fuzzy cells, sorted by number of co-varied genes with which the gene of interest has Importance value greater than the global third quartile of importance; (D) Rug plots of selected genes against the pseudotime.

hard clusters, consensus matrix (Figure 4.2.5 A & B), membership degree, and ECC plots.

The tab also enables the identification of major regulatory hubs (Section 3.5) in a data-driven way by finding fuzzy genes that have high co-variation of expression with a high percentage of the DE genes and returning them in an interactive table (Figure 4.2.5 C). When selecting a gene from that table its expression across the pseudotime is plotted (Figure 4.2.5 D). Considering the data and existing literature, interesting genes can be fixed as major regulatory hubs and subsequent inference and examination of the GRNs in the next tab will be based on them.

4.2.8 GRN inference tab

From this tab, GRNs for the cells around the three centroids (Section 4.2.6) can be calculated, specifying as GRNBoost2 TFs the major regulatory hubs chosen at the Gene Modules tab. The genes in each input matrix are the DE genes at the transition, identified earlier. The resulting GRNs for the three time-points/locations are presented in a table. The GRNs can also be visualised individually at each step (Figure 4.2.6 A) and combined (Figure 4.2.6 B), allowing for the examination of changes in topology and interaction strength across the transitions.

This is significant as the whole framework, through an interactive and data-interrogating approach, leads to these detailed snapshots of GRNs across the transition, focusing on genes that

influence the transition the most. This has great potential in unveiling regulatory dynamics and advancing researchers' understanding of the signals controlling the appearance of plasticity.

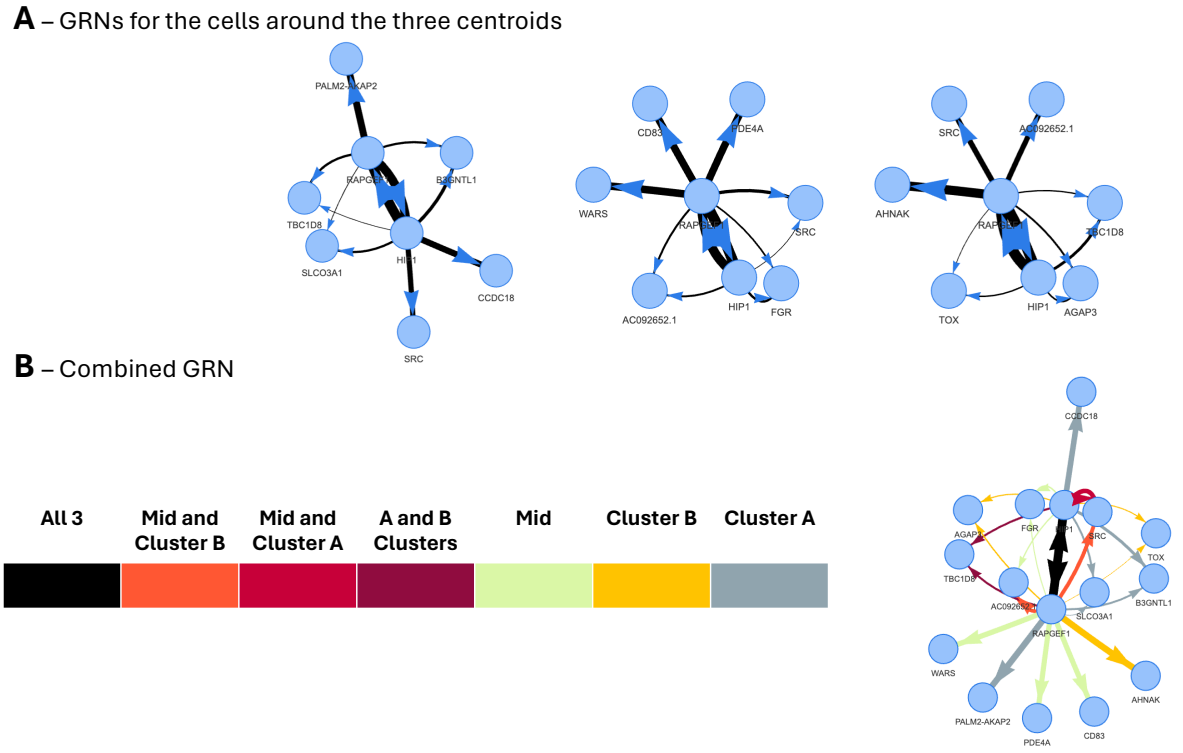


Figure 4.2.6: GRN plots at the GRN inference tab: (A) GRNs at the three different locations: cells around centroid of Cluster A, cells around centroid of midpoint, and cells around centroid of cluster B; (B) Combined GRN, highlighting the differences and commonalities between the three GRNs.

4.3 Direct scripting

Function	Purpose
`run_fuzzy_clustering`	Perform repeated clustering for each configuration, reconcile the labels and group the results by (algorithm, resolution, k).
`fuzzy_clustering_analysis`	Perform full fuzzy clustering analysis. Returning partitions, membership degrees, hard clusters, and the ECC for each tested configuration.
`get_consensus_matrix`	Constructs a consensus matrix given a list of partitions.
`get_top_fuzzy_ranked`	Get top fuzzy genes given a dataframe containing cells/-genes and a fuzziness metric. Eligible rows are those with ECC values below the 75th percentile and for entropy above the 25th percentile, capturing the top respective percentages of these eligible entries.
`get_pseudotime_trajectory_inference`	Returns the pseudotime trajectory given starting, ending genes, and number of genes to relax.
`get_decomposed_pseudotime_plot`	Plots the distribution of cell types and clusters across pseudotime (decomposed pseudotime).
`get_transition_cells`	Calculates the midpoint centroid between two clusters and selects the closest points to the midpoint.
`identify_DE_genes_at_the_transition`	Identifies differentially expressed genes at the transition area selected.
`gene_modules_fuzzy_clustering`	Performs fuzzy gene modules clustering.
`get_summary_connections_per_gene_df`	Returns a summary statistics for each TF containing information about the degree of its covariation with the target genes.
`get_GRN`	Creates a gene regulatory network using GRNBoost2 given an expression matrix and a list of transcription factors (TFs).
`visualise_GRN`	Visualise GRNBoost2 output.

Table 4.3.1: Summary of the purpose of the most important functions in the *flufftail* package.

flufftail also enables programmers and bioinformaticians to interact directly with its features via scripting, allowing manual, reproducible execution of each step outlined in the Shiny app. They can control every aspect of the process by invoking specific functions within an R script, tailoring the analysis to their exact specifications. The main functions of the package are presented in Table 4.3.1.

The documentation (docs folder) includes detailed information on the package's functions and instructions (including vignettes) on how to use the package.

Chapter 5

Discussion

In the Discussion Chapter, I focus on illustrating and discussing the usefulness of *flufftail* while relating the analysis to the objectives and aim proposed by the authors in Gribben et al. [1]. On this subset of cells, which highlighted the existence of biphenotypic cells for the first time, I recapitulate their conclusions and show a data-driven approach for a full characterisation of the biphenotypic cells. To further underline the generalisability of *flufftail*, I illustrate its applicability across different modalities by applying it to the scATAC-seq dataset. Moreover, to assess scalability, I benchmark the package on variable number of cells derived from the Gribben et al dataset. I conclude with an outline of current limitations and future opportunities.

5.1 *flufftail* case study on the Gribben et al. dataset

Parameter name	**Value**
Feature set for PCA	3,000 highly variable genes
Graph type	Shared nearest neighbours identified on 50 principal components
Batch effect correction	Harmony, with default parameters except for θ which was minimised (set to 0)
UMAP calculation	Default parameters, except min.dist which was set to 0.8 and n.neighbours which was set to 20.

Table 5.1.1: Parameters used in the analysis. Descriptions taken from [1].

This section presents an application of *flufftail*, on the *End-stage* samples from Gribben et al. [1]. Essential pre-processing steps, such as quality control, normalization, dimensionality

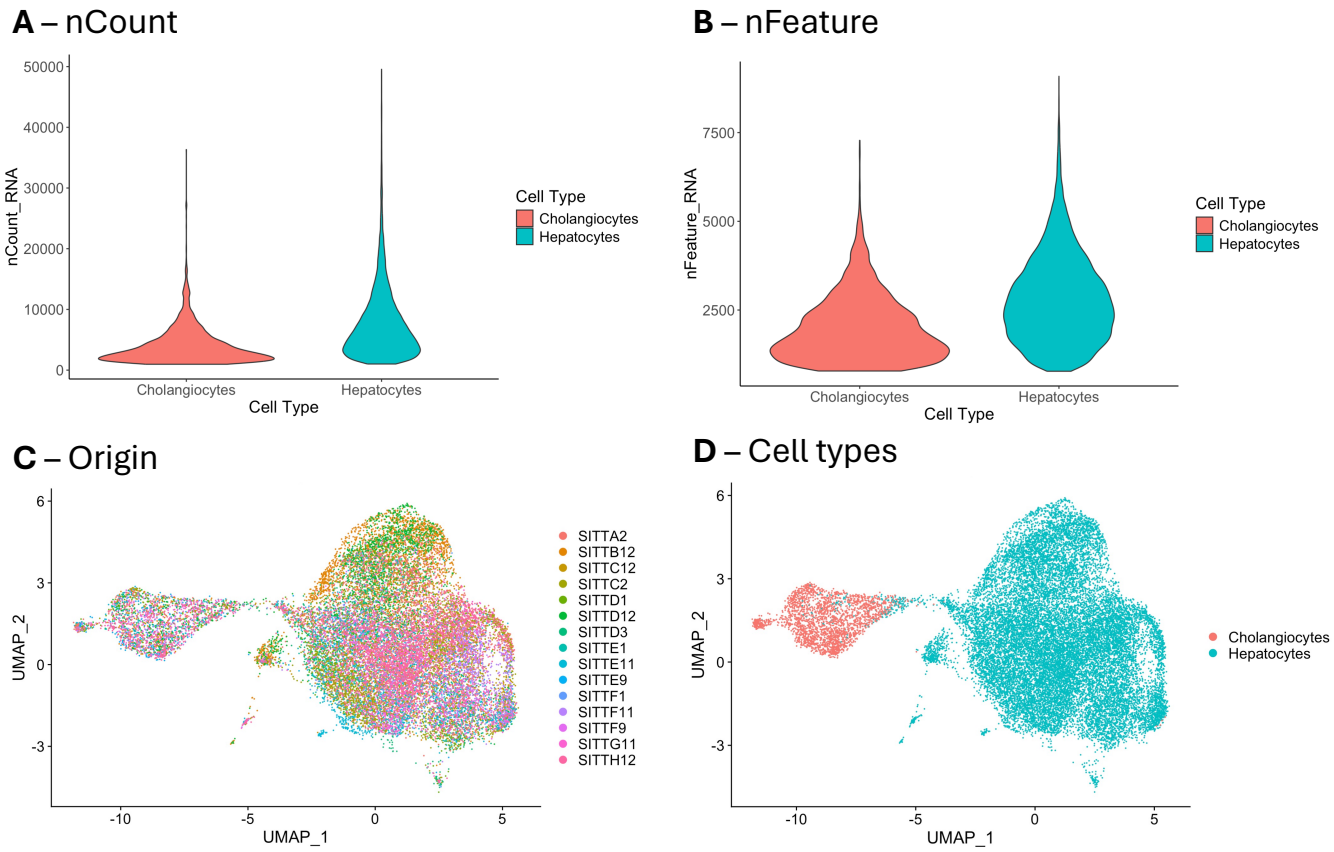


Figure 5.1.1: Dataset characteristics summary: (A) Distribution of total molecules (UMIs) in Hepatocytes and Cholangiocytes (nCount); (B) Number of unique genes in Hepatocytes and Cholangiocytes (nFeature); (C) UMAP coloured by data sources; (D) UMAP coloured by cell types.

reduction, feature selection, and batch correction, were applied by the authors and will not be reiterated in detail as the focus here is illustrating the package. The selected parameters for the analysis, as determined by the authors who also used ClustAssess, are detailed in Table 5.1.1.

Figure 5.1.1 summarises the main characteristics of the dataset. The top panels display the distributions of total molecules (UMIs) and unique genes, with Hepatocytes consistently exhibiting higher counts of both compared to Cholangiocytes. The Cholangiocytes, however, show a broader distribution at lower UMI values, indicating a higher frequency of cells with reduced RNA content. The significant overlap and mixing of cells from different origins (Panel C) suggests that the batch correction and integration processes were effective, preserving biological variability without overcorrection. Panel D illustrates distinct transcriptional profiles between Cholangiocytes and Hepatocytes; notably, there is a bridging region where the two cell types mix, suggesting the existence of transitional states (intermediate cells). The isolated Hepatocyte group at the bottom might represent a distinct sub-cell type, still classified as Hepatocyte. For our analysis, we focused exclusively on the larger island linked to the transition,

hence the Seurat object was subsampled, encompassing 23,835 cells and 29,653 genes.

The analysis script "application_gribben_et_al.R" can be found in the Scripts folder. A lighter version (excluding pre-processing steps) with explanations is available as an R markdown in the Vignettes folder ("tutorial_gribben_et_al.Rmd"). A supplementary video was also created demonstrating the application of the framework to perform a similar analysis on the subset through the Shiny app ("flufftail_demo.mp4"[1]).

5.1.1 Fuzzy cell clustering

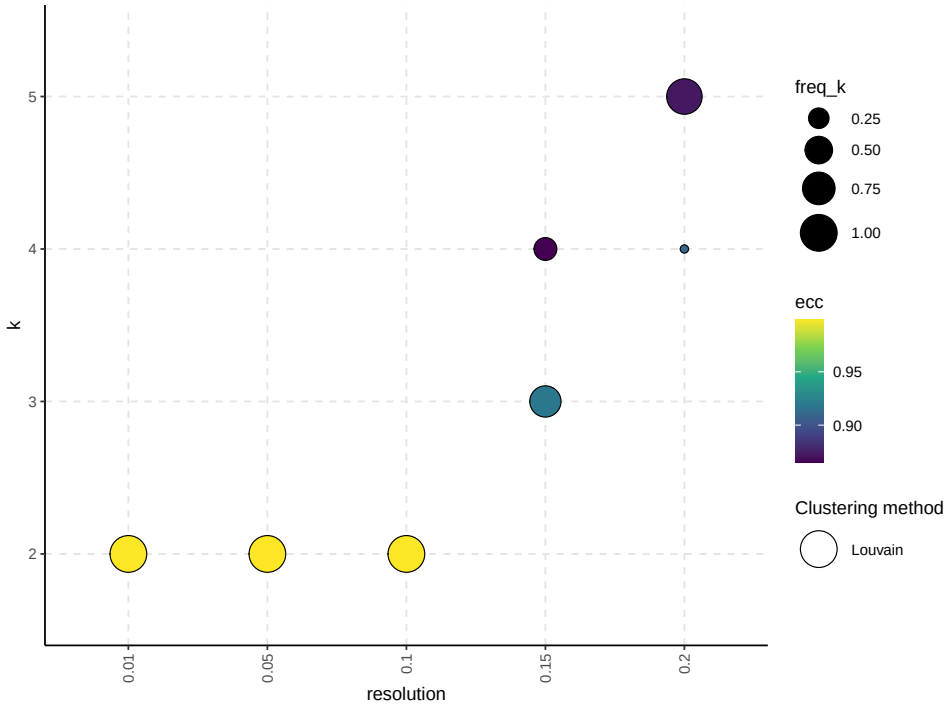


Figure 5.1.2: The plot visualises partitions generated across various resolutions. Each point's colour gradient illustrates the ECC, with darker colours indicating higher values. The size of each point corresponds to the percentage of partitions at a given resolution that resulted in (k) clusters.

The `fuzzy_clustering_analysis` function was used to compute membership degrees, hard clusters, and ECC for each tested configuration given the adjacency matrix, number of iterations (50), algorithm (Louvain[2]), and resolution sequence (0.01, 0.05, 0.1, 0.15, and 0.2).

Figure 5.1.2 illustrates the partitioning outcomes, with the colour gradient representing ECC values from 0.85 to 0.95. The y-axis indicates the number of clusters (k), and the x-axis represents resolution values. Symbol sizes reflect *freq_k*, the frequency of partitions at specific

[1]Also available at this link: `https://www.youtube.com/watch?v=C0JJMdQvRew`
[2]The Louvain algorithm was chosen as it is the algorithm the authors used in their analysis.

resolutions yielding (k) clusters, with larger symbols indicating a higher frequency. Table 5.1.1 enumerates the total count of observed partitions for each resolution and (k) combination.

The results are summarised as follows:

- At resolutions of 0.01, 0.05, and 0.1, high stability is observed with average ECC scores near 1, consistently resulting in two clusters over 50 iterations. Since all three resolutions yield the same number of clusters and similar configurations, we will proceed with the smallest resolution (0.01), following the Keep-it-Simple principle.
- At a resolution of 0.15, ECC scores for three and four clusters range between 0.85 and 0.9, with three clusters occurring more frequently than four.
- At a resolution of 0.2, five clusters appear most frequently (46 out of 50 cases), though four clusters also emerge (4 instances), with ECC scores ranging from 0.85 to 0.9.

Resolution	k	**Partitions**
0.01	2	50
0.05	2	50
0.1	2	50
0.15	3	34
0.15	4	16
0.2	4	4
0.2	5	46

Table 5.1.2: Number of partitions in each resolution and k configuration

Figure 5.1.3 presents consensus matrices for configurations with the highest partition counts at each resolution, subsampled to 3,000 cells for visualisation. At a resolution of 0.01 and $k = 2$, clustering shows high stability with a predominantly blue diagonal, indicating strong consensus and distinct separation between the two groups. At resolution 0.15 with $k = 3$, the plot displays moderate variability, yet maintains a relatively high consensus. At resolution 0.2 with $k = 5$, the matrix shows greater variability and less consistent co-clustering, as indicated by mixed intensities within clusters and more prevalent blue in off-diagonal sections.

Gribben et al. [1] aimed to characterise the bridge between Cholangiocytes and Hepatocytes and identify biphenotypic cells (i.e., fuzzy cells) found at the transition between the two cell types. In our fuzzy clustering analysis, the only clustering configuration resulting in two clusters is at a resolution of 0.01 (Figure 5.1.4 A). Despite the high stability of the consensus matrix shown in Figure 5.1.3, the application of the entropy method to both the consensus and degree matrices, along with the calculation of the ECC, pinpointed fuzzy cells at the juncture between

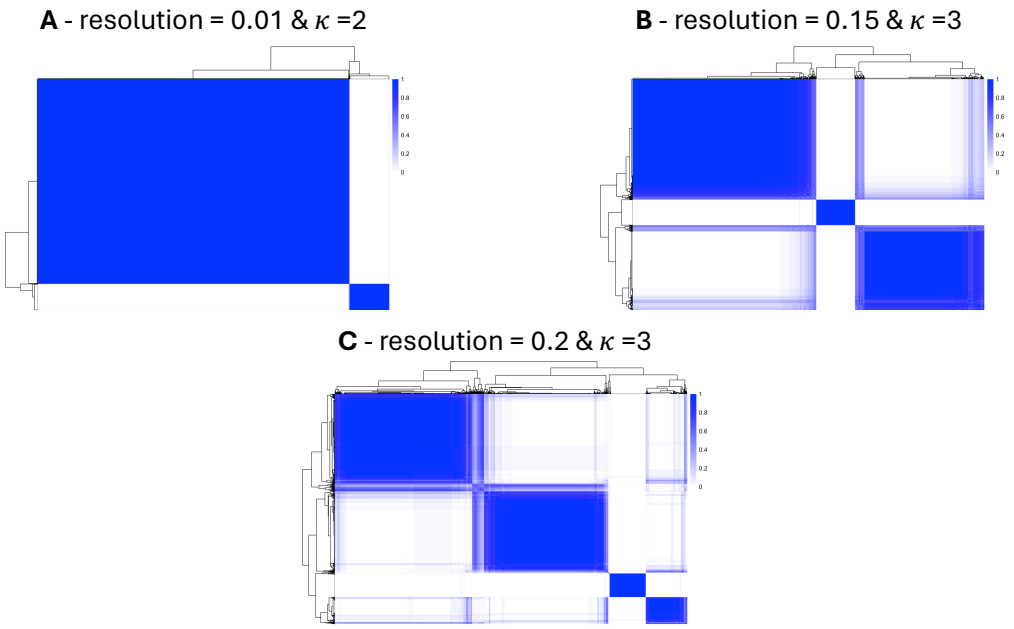


Figure 5.1.3: Subsampled consensus matrix (3,000 cells) for the configurations resulting in the most partitions for each resolution.

the two clusters, as depicted in Figure 5.1.4. This demonstrates the effectiveness of identifying fuzzy cells using *flufftail* even in highly stable configurations. By calculating the membership degree, we can further quantify the extent of fuzziness for each of these cells (objectives 1 & 2), providing a nuanced understanding of their membership characteristics.

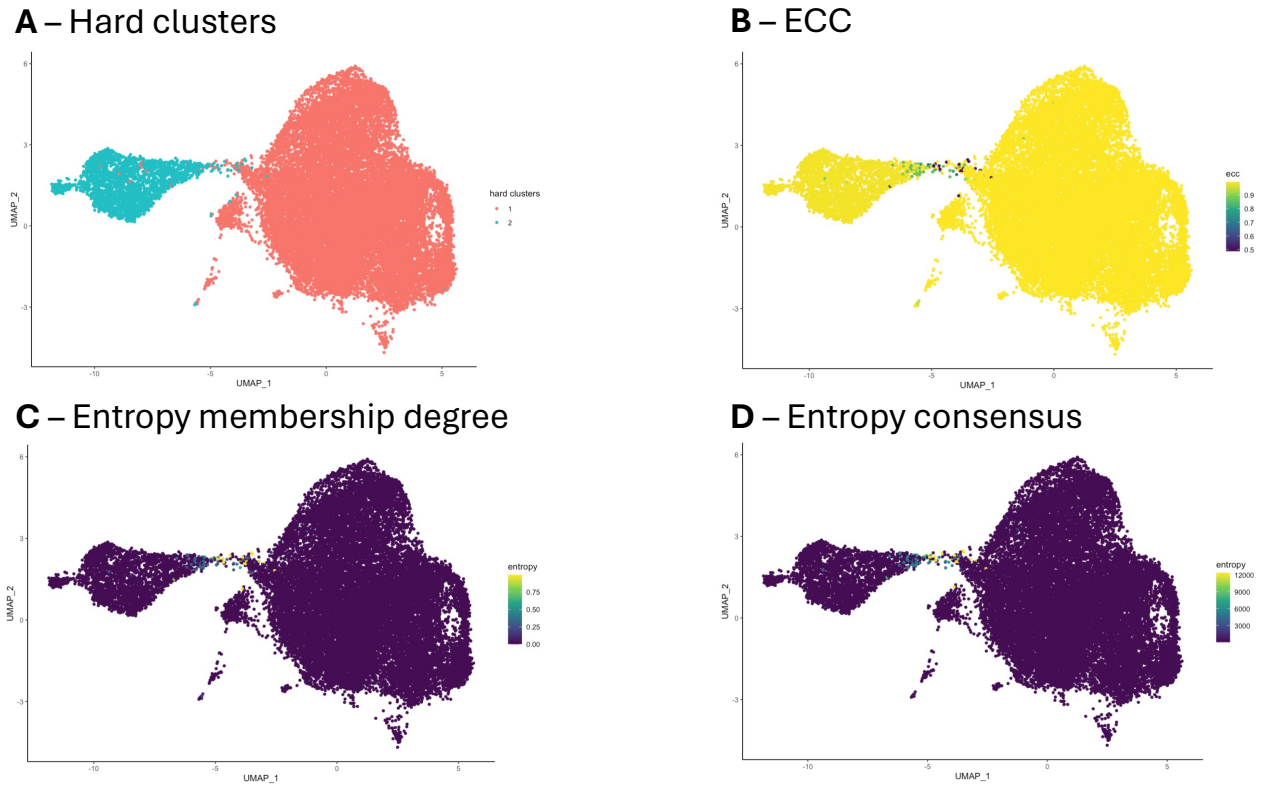


Figure 5.1.4: Resulting UMAPs from the fuzzy clustering analysis on the selected configuration: (A) Hard clusters; (B) ECC; (C) Entropy applies on the membership degree matrix; (D) Entropy applied on the consensus matrix.

5.1.2 Transitional cells

Pseudotime trajectory analysis was conducted using starting and end point subsets of manually curated genes. SERPINE1, CYP2B6, INSIG1, and HMGCS1 were designated as start genes, while CFTR, RALYL, KRT7, and ITGB8 as end genes. The `get_pseudotime_trajectory_inference` function was employed with a quantile threshold of 0.85 and the number of relax genes set to 1 for both start and end gene sets (Section 4.2.5).

Figure 5.1.5 showcases the pseudotime trajectory in panel A and its stability when the top 2% of fuzzy cells were removed in panel B. The comparison between the two pseudotime orderings, before and after excluding the fuzzy cells, lies on the diagonal, confirming that the trajectory inference is robust. The decomposed pseudotime plot (panel C) shows that cells in cluster 1 exhibit lower pseudotime values, in contrast to the broader distribution observed in cluster 2. This early predominance of Hepatocytes delineates the transition over time from cluster 1 (Hepatocytes) to cluster 2 (Cholangiocytes). The `get_transition_cells` function was then employed to select 40% of cells surrounding the centroids of both clusters, along with 1,600 cells around the midpoint centroid (Figure 5.1.5 D). Panel E shows the top 25 DE genes and their gene expression across the transition, with a greater number of genes expressed

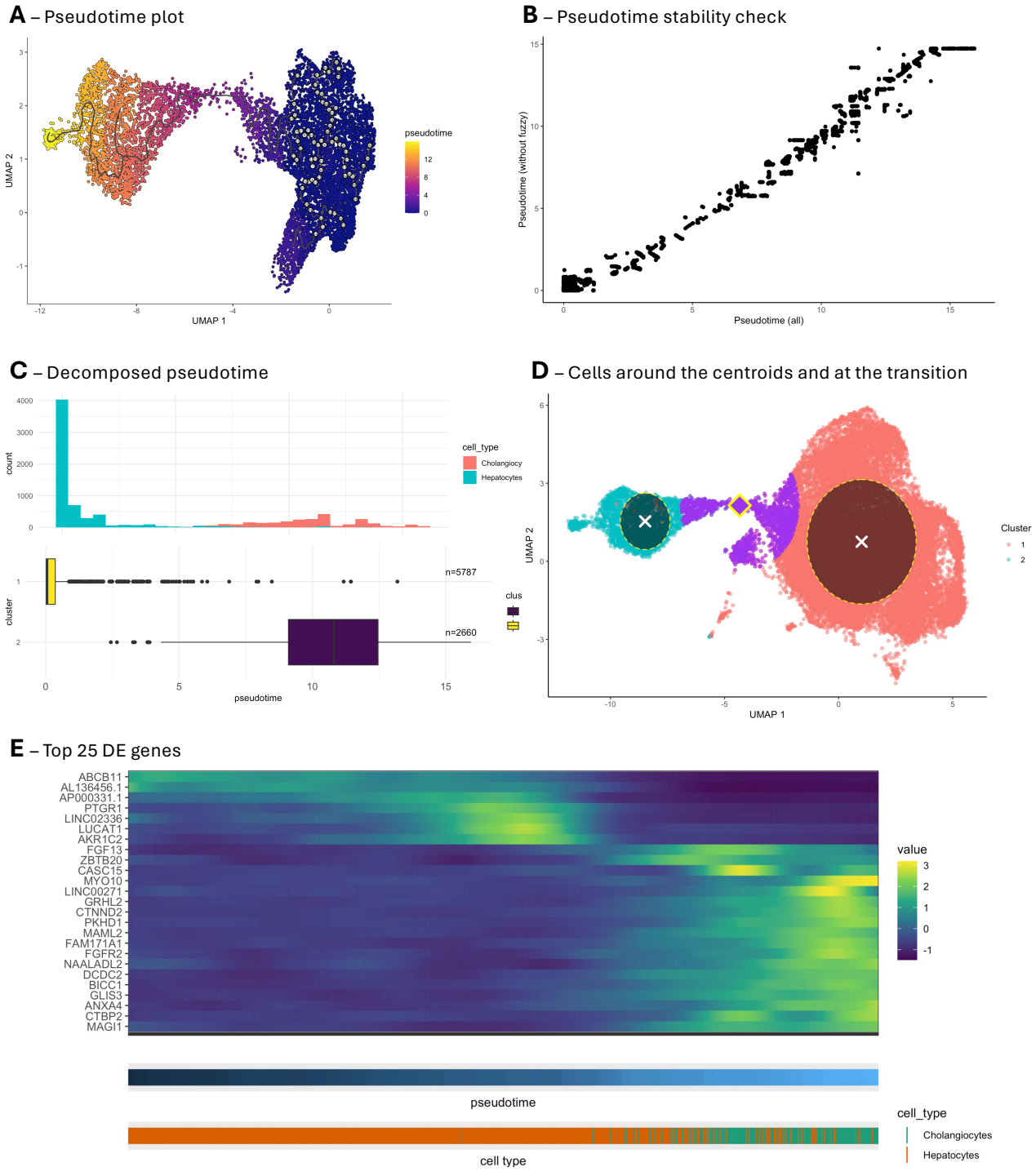


Figure 5.1.5: Pseudotime analysis. (A) Trajectory showing the ordering of cells; (B) Relationship of pseudotime values between the two trajectories (all vs without top fuzzy cells); (C) Decomposed pseudotime visualising the distribution of cell types and clusters across the pseudotime; (D) UMAP visualising the cells selected around the two clusters and midpoint centroid; (E) Heatmap showing the top 25 DE genes across the transition.

predominantly as the transition progresses towards Cholangiocytes, despite the smaller number of Cholangiocyte cells present at the transition area selected.

GSEA of the DE genes ($q<0.05$) using gprofiler2 [61] revealed enrichment in various Gene Ontology terms, including cell differentiation. In KEGG pathways, there was significant enrichment for alcoholic liver disease, insulin resistance, and most importantly, the PI3K–AKT pathway, which was identified by Gribben et al. [1] and is strongly associated with obesity and metabolic syndrome—both closely linked to MASLD [62, 63].

5.1.3 Fuzzy gene module clustering and hub genes

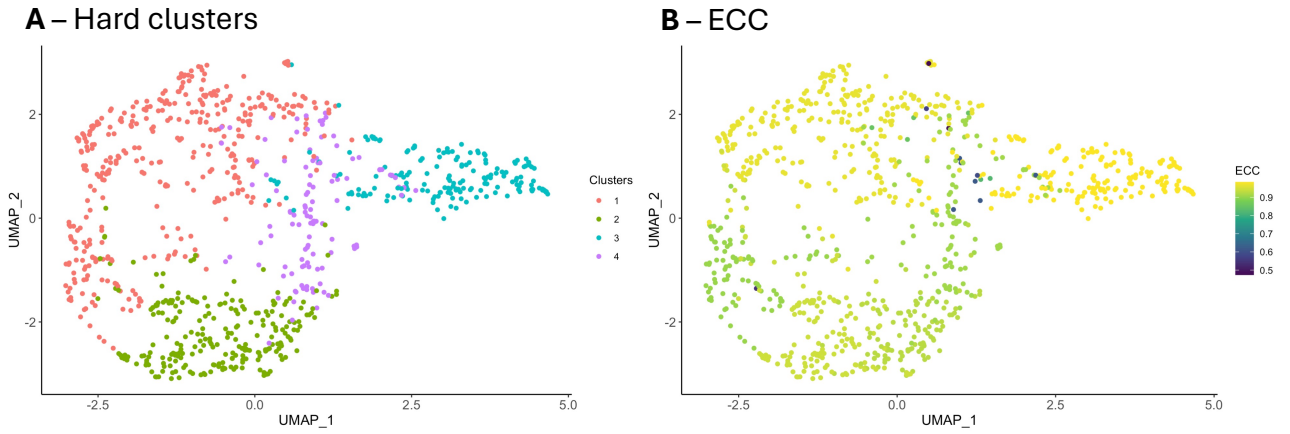


Figure 5.1.6: Gene modules fuzzy clustering: (A) Hard clusters after majority voting; (B) ECC results.

The Moran's I threshold significantly impacts the number of genes returned by the graph test, as shown in Appendix E. For conducting fuzzy gene module clustering, a Moran's I threshold above 0.03 was selected to reduce the number of genes while still retaining enough to capture interesting patterns. This resulted in a transition matrix (Section 3.5) comprising 909 genes and 1,600 cells. Fuzzy gene module clustering was performed using `gene_modules_fuzzy_clustering` over 50 iterations. The SLM algorithm was applied, testing resolutions from 0.1 to 1.0 in increments of 0.1.

The partitions at resolution=0.5 and k=4 were chosen for further analysis as they showed high median ECC (0.9446) but also some fuzziness (min ECC = 0.4734, Q1 ECC = 0.9297). Figure 5.1.6, visualises the resulting clusters and ECC. Cells near the intersection of clusters 1, 3, and 4 have the lowest ECC with the genes with the lowest ECC located at the cluster boundaries.

To identify major regulatory hubs, a GRN was constructed testing each of the top 5% fuzzy genes (lowest ECC) as GRNBoost2 TFs. The `get_summary_connections_per_gene` function was then called to get the summary statistics about which are most involved in the dynamics. The findings are detailed in Table 5.1.3, which lists the percentage of target genes for each TF, where the Importance co-variation exceeds the target-specific median and Q3 Importance (Section 3.5). Importantly, each of these genes has been corroborated by the scientific literature as being linked to liver conditions or MASLD/MASH, thereby strongly validating the effectiveness and relevance of the developed methodology (objective 3).

After consulting with a biologist[3] the BICC1 and CTNND2 genes (top 2 in the table) were selected as potential regulatory hub genes. Importantly, the authors found CTNND2 to be a Cholangiocyte marker and BICC1 to be a Cholangiocyte-like Hepatocyte marker, i.e., involved

[3]Dr. Floris Johan Maria Roos, Cambridge Stem Cell Institute, University of Cambridge.

in plasticity.

Gene	% > Median	% > Q3	References
BICC1	94%	82%	[64, 65]
CTNND2	94%	79%	[66]
PKHD1	96%	78%	[67]
FGF13	85%	70%	[67]
DCDC2	86%	65%	[68, 69]
PZP	63%	26%	[70, 71, 72]
ALAS1	54%	22%	[73]
SAA2	48%	21%	[74, 75]
UGDH	55%	20%	[76]
PRKCE	50%	20%	[77, 78, 79, 80]

Table 5.1.3: Top 10 fuzzy genes ranked by their expression co-variation with a high percentage of differentially expressed (DE) genes. The table displays the percentages of target genes for which the transcription factor (TF) exceeds the global median and third-quartile (Q3) of target-specific Importance distributions. Also provides references linking each of the genes to liver or MASLD/MASH.

5.1.4 First framework facilitating the study of GRN dynamics for chracterising regulatory interactions on major regulatory hubs

Using the `get_GRN` function, three GRNs were constructed at three different areas (centroids around clusters and cells at the transition). BICC1 and CTNND2 were used as GRNBoost2 TFs, with all other DE genes serving as target genes.

Figure 5.1.7 (A-C) shows the top seven interactions of the three GRNs and the common targets between the two TFs with an importance value above the Q3 of the network's overall importance distribution. Panel D highlights that most interactions at each step are unique, indicating considerable changes in top gene covariation through the transition. Notably, the interaction between CTNND2 and PROM1 is common to the GRNs of clusters 1 and 2 but absent in the transition GRN. CTNND2 being a target gene of BICC1 is common across all three GRNs. Conversely, BICC1 is a target of CTNND2 only in the transition and second cluster GRNs.

Panel 5.1.7 E shows that the two genes have very high expression after the transition to Cholangiocytes. This is also reflected in the GRNs, as the GRNs at the transition and cluster 2 have a greater number of highly covariant genes.

This highlights the kind of analysis that *flufftail* enables. By interrogating the data in a data-

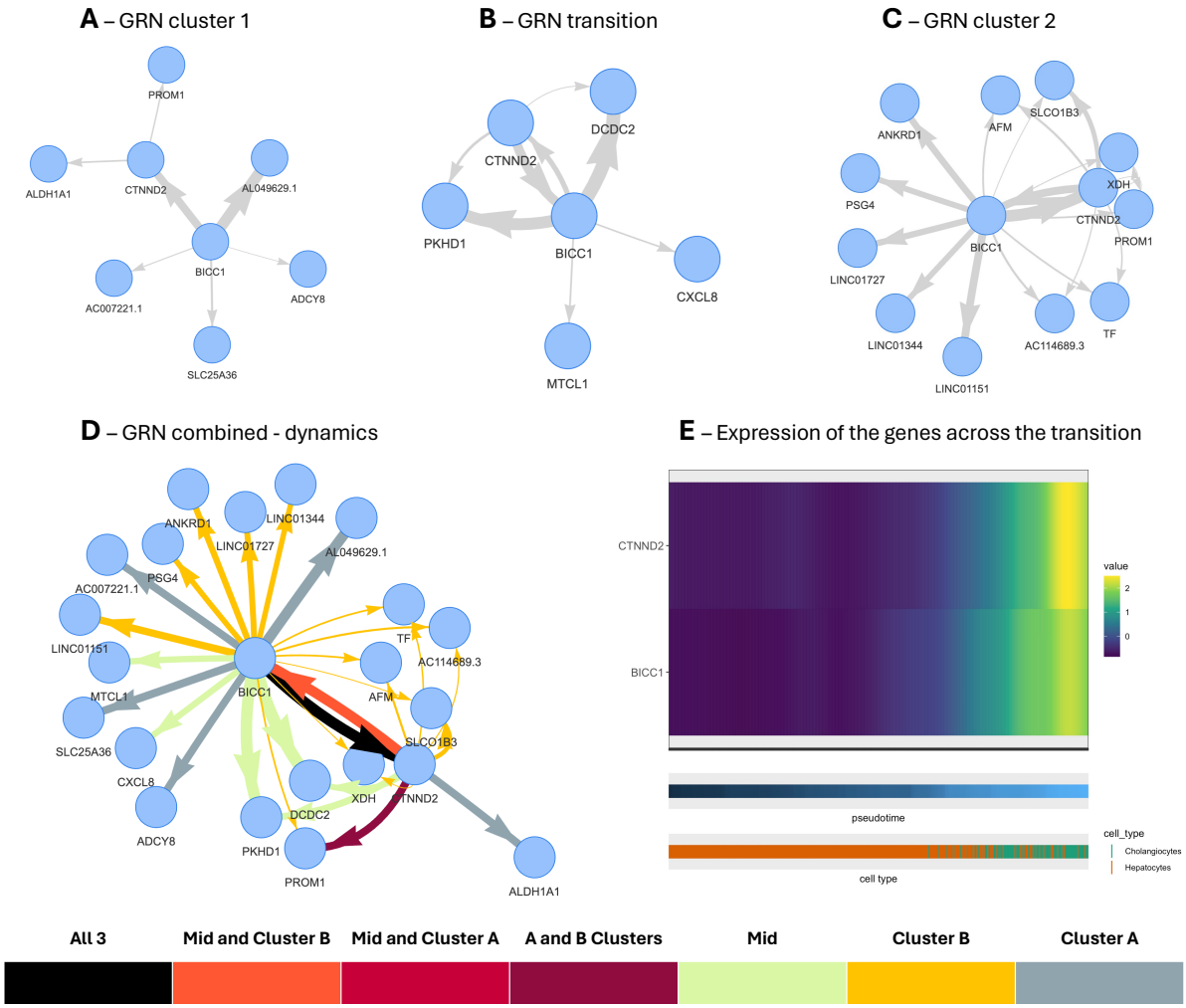


Figure 5.1.7: GRN dynamics of the two hub genes selected. (A) GRN using DE genes and cells around the centroid of cluster 1; (B) GRN using cells around midpoint centroids; (C) GRN using cells around the centroid of cluster 2; (D) Combined GRN showing the dynamics; (E) expression of the two genes across the transition.

driven fuzzy way and combining that with GRN inference, we were able to uncover how the regulatory dynamics of genes highly involved in the transdifferentiation process evolve (main aim & objective 4).

5.2 *flufftail* is agnostic across different modalities

Since *flufftail* internally uses Seurat's clustering and Monocle3 pseudotime functions, it is agnostic to the modality and can be used across a range of single-cell data types. This includes scRNA-seq, single-cell ATAC sequencing (scATAC-seq), and RNA+ATAC multiome data.

To demonstrate this, the package was applied on a publicly available scATAC-seq dataset

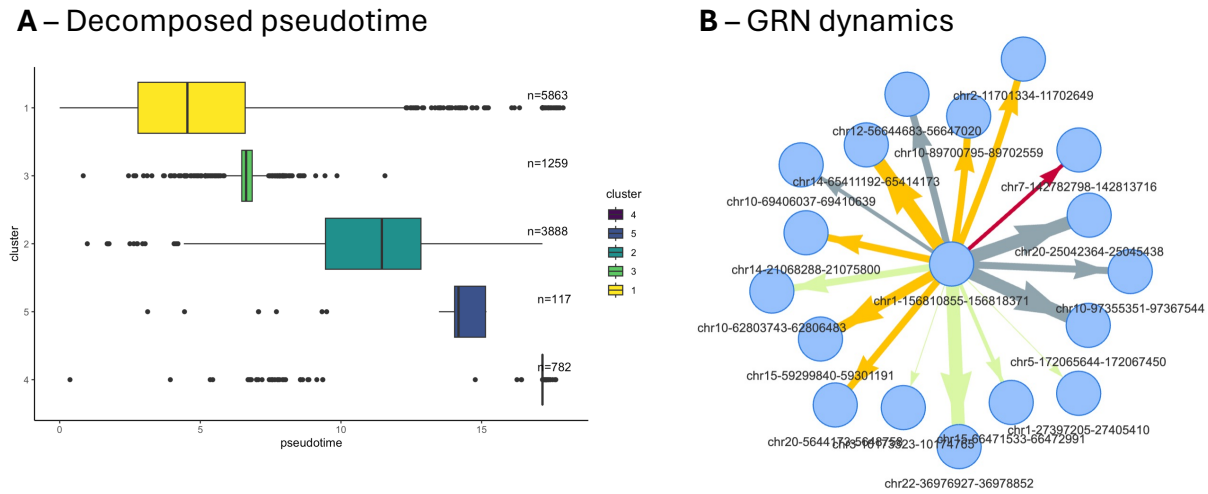


Figure 5.2.1: *flufftail* can be applied to different modalities. Plots showing application to an scATAC dataset. (A) Decomposed pseudotime on clusters; (B) GRN dynamics inferred for a major regulatory hub.

(multiome PBMC) from the SeuratData package [25] and the full analysis was successfully conducted. Figure 5.2.1 shows the cluster transitions following the decomposed pseudotime (panel A) and the GRN dynamics of the major regulatory hub chromatin accessibility region identified using the framework (panel B).

The full analysis is available on the "ATAC.Rmd" vignette.

5.3 Benchmarking of the *flufftail* framework

Due to a ransomware viral attack in February 2024 that caused server outages at the Addenbrooke's campus from February (25/02/2024) until today (19/07/2024), the benchmarking process was significantly affected.

Nevertheless, I put forward my best efforts to benchmark the main fuzziness-related algorithms developed for the package, `fuzzy_clustering_analysis` and `get_consensus_matrix` using the computer setup described in Section 4.2.1. The dataset used was the *End-stage* Seurat object, featuring the larger island with 23,835 cells and 29,653 genes (described in Section 5.1.1). The dataset was subsampled to sizes of 2,000, 5,000, 10,000, 15,000, 20,000, and 23,835 cells to assess performance.

The reported times are median execution times in seconds after executing each operation 10 times using the bench[4] package.

[4]Official GitHub repository: `https://github.com/r-lib/bench` - accessed 02/07/2024

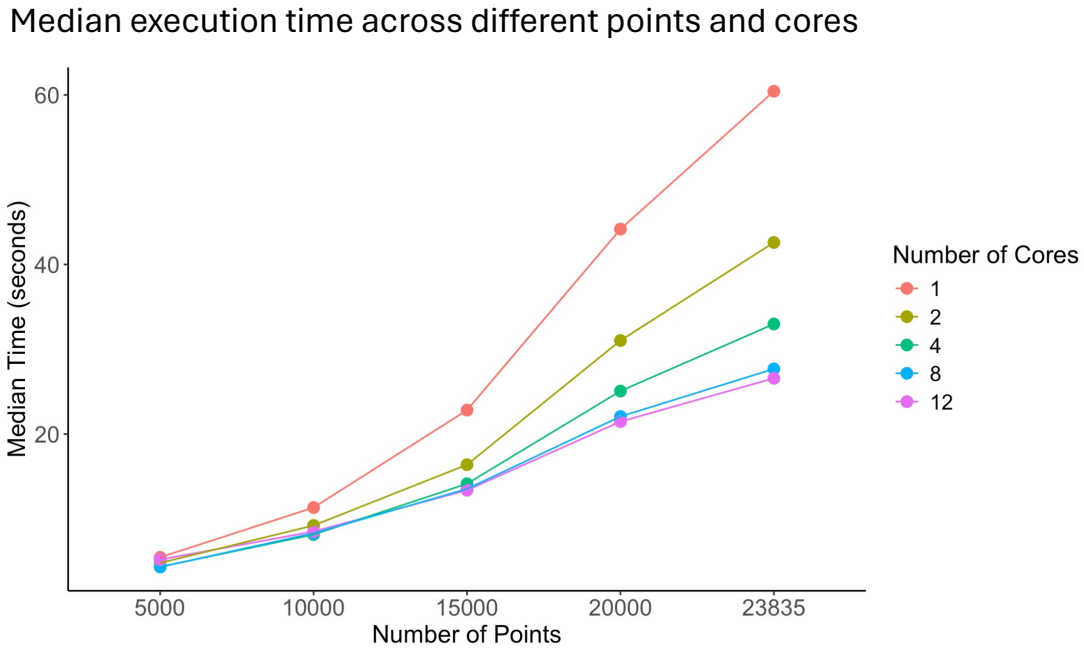


Figure 5.3.1: Median execution time across different points and cores.

5.3.1 Benchmarking fuzzy clustering analysis

Figure 5.3.1 illustrates the relationship between sample size, parallelization (core count), and execution time of the fuzzy clustering function using the Louvain algorithm with 30 repetitions. The analysis reveals several key insights:

1. **Non-linear scaling with sample size:** As the number of samples increases, execution time also rises non-linearly. Importantly, the rate of change is less steep when using more cores.

2. **Highly parallelisable:** Increasing the number of cores significantly reduces median execution times across all sample sizes, with the most notable improvements seen in larger sample sizes.

3. **Parallelisation bottleneck:** The performance gain when increasing from 8 to 12 cores is less pronounced, likely due to increased parallelization management overhead.

4. **Effective on limited resources:** Even with a single core on a personal computer, the analysis completes in under a minute, highlighting *flufftail*'s efficiency and accessibility for users with limited computational resources.

Table 5.3.1 demonstrates that increasing the number of repetitions results in a linear increase in execution time (R^2=0.997), as tested using the Louvain algorithm with 0.5 resolution on the full dataset.

Repetitions	Median execution time (s)
10	4.97
30	12.10
50	20.19
70	27.40
90	36.73

Table 5.3.1: Median execution times for fuzzy clustering analysis across different number of repetitions, based on 10 benchmark repeats. Ran sequentially.

To compare the performance of the four algorithms, the subsampled dataset with 10,000 samples, a resolution of 0.5, and 8 cores was used. As shown in Table 5.3.2, Louvain, Louvain-refined, and SLM exhibit very fast execution times across all repetitions. However, the Leiden algorithm is significantly slower, posing limitations for analysing large datasets. This slowdown occurs because Seurat calls a Python implementation of Leiden via the reticulate package[5], whereas the other algorithms were implemented by the Seurat authors in C++. This process introduces considerable overhead, especially during multiple repeats of the algorithm, as in our case.

Repetitions	Time (seconds)			
	Louvain	Louvain-refined	SLM	Leiden
10	1.05	1.05	1.26	120
30	1.52	1.55	1.75	321.85
50	2.23	2.24	2.49	537.27

Table 5.3.2: Median execution times for fuzzy clustering analysis of 10,000 samples across 8 cores using the four different algorithms.

5.3.2 Benchmarking consensus matrix

The computation of the consensus matrix was assessed by comparing our implementation with the widely used SC3 package. For each sample size evaluated, 30 distinct partitions were generated and utilized to construct the consensus matrices.

The consensus matrices produced by the two methods were equivalent across all tested scenarios, thereby confirming the correctness of the implementation. Figure 5.3.2 presents the median execution times in seconds for each algorithm across various subsample sizes from the dataset. The *flufftail* implementation consistently outperforms SC3 in terms of speed, demonstrating approximately 3.5 times faster processing speeds across all sample sizes. This can be attributed to an optimization in *flufftail* that leverages OpenMP for parallel processing. By parallelising

[5]Discussion: `https://github.com/satijalab/seurat/discussions/6754`

Samples	Flufftail			SC3		
	Time (s)	Memory (MB)	GC (s)	Time (s)	Memory (MB)	GC (s)
5,000	0.27	193	0.42	0.96	192	0.12
10,000	0.99	768	0.68	3.9	765	0.065
15,000	2.4	1,680	0.11	8.0	1,680	0.031
20,000	4.4	2,990	0.095	16	2,980	0.016
23,835	7.7	4,240	0.089	28	4,240	0.0071

Table 5.3.3: Comparison of *flufftail* and SC3 performance metrics. Time is reported in seconds, memory allocation in megabyte (MB), and number of garbage collections per second.

the outer loop, the computation efficiently distributes iterations across available threads using static scheduling, which divides iterations approximately equally among threads.

Both methods allocate nearly identical amounts of memory across each sample size, as detailed in Table 5.3.3. Notably, despite its faster speed, *flufftail* exhibits a higher garbage collection frequency per second than SC3, particularly at smaller sample sizes. This may be linked to the parallelisation strategy which might result in more frequent, thread-local memory allocations and deallocations. Future optimisations could explore techniques to balance the benefits of parallelisation with more efficient memory management.

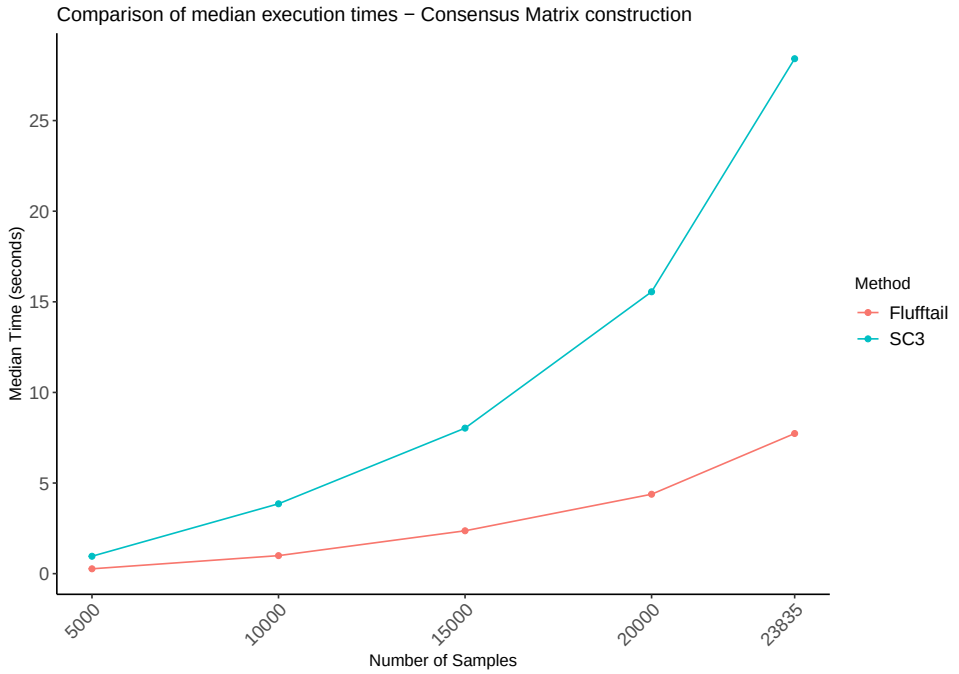


Figure 5.3.2: Median execution times (out of 10 repeats) to compute consensus matrices using the *flufftail* and SC3 packages across varying sample sizes. It demonstrates *flufftail*'s consistent performance advantage over SC3 as sample sizes increase.

5.4 Current limitations and future opportunities

While the *flufftail* package represents a step forward in terms of data-driven single-cell analysis from a fuzzy perspective, there are still several current limitations and exciting future improvements.

Selecting the most suitable clustering configuration (algorithm, resolution, k) for fuzzy clustering analysis is a non-trivial task that users might struggle with, potentially affecting the results returned by the framework. I recommend using ClustAssess before employing *flufftail* to identify an optimal configuration that achieves two key objectives: first, it should yield a high median ECC (e.g., above 0.95), indicating consistent capture of true biological signals across repeated clustering; second, it is expected to retain some degree of fuzziness in localised areas, reflecting genuine biological uncertainty, that would allow us to harness the fuzziness. Rephrased, from a computational perspective, the *flufftail* framework should be used for the identification of genuine (and interpretable) variable expression signatures, and not capture technical variation derived from sub-optimal parameters.

flufftail currently permits the independent analysis of individual modalities, e.g., it is possible to apply the pipeline first to scRNA-seq and then to scATAC-seq. However, it currently does not support an integrated analysis of multi-modal data from the outset. Integrating multi-modal data analyses would enable a more holistic overview of the interdependencies between different types of biological assays, potentially uncovering additional crucial insights that could only emerge from a combined analysis.

Moreover, while fuzzy clustering at the gene module level is central to identifying genes driving the dynamics, fuzzy clustering at the cell level currently focuses on identifying the locations and degrees of fuzziness. This approach assists the user in selecting transitional clusters/identifying areas of plasticity and examining the stability of the pseudotime trajectory; yet future updates could further exploit the fuzzy cells by integrating the RNA velocity information e.g. using a package like scVelo [81]. This could provide further insights into the directionality/ causality of transdifferentiating cells.

Studies show that the Leiden algorithm is currently the preferred approach for single-cell clustering [82, 37]. However, as discussed in 5.3.1, the Leiden algorithm in the Seurat package is significantly slower than the other three options. Due to the repeated nature of *flufftail*, analysing very large datasets with Leiden requires significantly more time and computational resources compared to the other approaches. For subsequent versions of the framework, the Leiden algorithm could be replaced by calling a different R-native package such as leidenbase [83] or writing an independent implementation of the algorithm.

Currently, using *flufftail* we can effectively compare GRN dynamics of hub gene(s) at three key

locations: cells around the centroid of Cluster A, cells at the transition, and cells around the centroid of Cluster B. This allows us to capture significant/driving regulatory changes across a transition of two clusters of interest. This capability could be further enhanced by developing a data-driven heuristic to identify the number and localization of additional suitable bins. Such an approach would enable a more detailed examination of GRN dynamics, providing deeper insights into regulatory changes at various points across the pseudotime.

In a future update, I could experiment with applying different clustering techniques (e.g., hierarchical, spectral) to the consensus matrix to identify the optimal approach for finding robust clusters. Currently, we are computing the consensus matrix but not structuring the output e.g., by performing additional clustering. Instead, we are using it to identify fuzzy cells (entropy approach) and visualise co-clustering behaviour. This extension would enhance the applicability of the package, providing it also with a robust clustering identification angle. Other minor updates could include an updated colour scheme (or reversing the existing one) for the entropy plots, as I recognise that it is currently challenging to distinguish fuzziness.

Finally, as previously mentioned, this project faced substantial challenges due to a ransomware attack that blocked access to the computing infrastructure at the Addenbrooke's campus from the end of February until today (19/07/2024). This disruption hindered my ability to perform extensive benchmarking. Soon, I plan to conduct additional extensive benchmarking and further validate more functions of the package with larger and more diverse datasets.

Chapter 6

Conclusions

The project resulted in the successful implementation of a package and an app that provide a robust, reproducible, scalable, and generalizable framework permitting the identification and characterization of fuzzy cells and genes. Positive feedback from the authors of the Gribben et al. [1] paper, corroborated with the interesting biological hypotheses resulting from applying this package to other datasets, represents a strong driver to further expand the work and structure the outputs in a manuscript.

From a clustering perspective, *flufftail* provides the bioinformatics community with a streamlined method for fuzzy clustering on community-detection algorithms and an optimized implementation of consolidation into a consensus matrix. The decomposed pseudotime feature significantly enhances the identification of transitions between clusters and transdifferentiation between cell types. Furthermore, through differential expression analysis on specific trajectory subsets, genes driving transition dynamics can be identified.

The package also supports fuzzy gene module clustering. Applying this to the DE gene set allows the identification of fuzzy genes involved in various biological processes at the transition area. *Flufftail* can be used to quantify the fuzziness of genes, which, in turn, can be linked through high covariation of expression with other genes within and across pathways, thus validating that the fuzzy genes are likely to act as major regulatory hubs driving the transition dynamics.

Another key feature of *flufftail* is the analysis of GRN dynamics and the evolution of regulatory interactions for selected hub genes at different points in pseudotime (other ordering of the cells is possible, thanks to the modular setup of the framework). This provides a snapshot of the dynamics across the transition, offering a deeper understanding of the dynamics driving disease progression or other biological processes linked to phenotypes. By identifying genes accelerating the disease at each stage, this analysis has the potential to inform precision medicine approaches, targeting these genes with drugs or nucleic acids.

Moreover, to strengthen communication between wet- and dry-lab researchers, a highly interactive R Shiny interface was developed, enabling rapid experimentation and assisting scientists in hypothesis generation. The interface democratizes this type of analysis, making it accessible to researchers with diverse backgrounds who, using their domain-specific knowledge, can extract focused hypotheses and insights from high-throughput datasets using *flufftail*.

The framework also proposes deeper analyses of single-cell datasets, going beyond the current static lists of differentially expressed genes, and bringing our understanding one step closer to assessing the dynamics and causality of regulatory interactions, a task that is one of the grand challenges of the decade [84].

Bibliography

[1] Christopher Gribben, Vasileios Galanakis, Alexander Calderwood, Eleanor C. Williams, Ruben Chazarra-Gil, Miguel Larraz, Carla Frau, Tobias Puengel, Adrien Guillot, Foad J. Rouhani, Krishnaa Mahbubani, Edmund Godfrey, Susan E. Davies, Emmanouil Athanasiadis, Kourosh Saeb-Parsy, Frank Tacke, Michael Allison, Irina Mohorianu, and Ludovic Vallier. Acquisition of epithelial plasticity in human chronic liver disease. *Nature*, 630(8015):166–173, 2024.

[2] Francis Crick. Central dogma of molecular biology. *Nature*, 227(5258):561–563, 1970.

[3] Quy H. Nguyen, Nicholas Pervolarakis, Kevin Nee, and Kai Kessenbrock. Experimental considerations for single-cell rna sequencing approaches. *Frontiers in Cell and Developmental Biology*, 6, 2018.

[4] Steven S. Potter. Single-cell rna sequencing for the study of development, physiology and disease. *Nature Reviews Nephrology*, 14(8):479–492, 2018.

[5] Alev Baysoy, Zhiliang Bai, Rahul Satija, and Rong Fan. The technological landscape and applications of single-cell multi-omics. *Nature Reviews Molecular Cell Biology*, 24(10):695–713, 2023.

[6] Vladimir Yu Kiselev, Kristina Kirschner, Michael T Schaub, Tallulah Andrews, Andrew Yiu, Tamir Chandra, Kedar N Natarajan, Wolf Reik, Mauricio Barahona, Anthony R Green, and Martin Hemberg. Sc3: consensus clustering of single-cell rna-seq data. *Nature Methods*, 14(5):483–486, March 2017.

[7] Wouter Saelens, Robrecht Cannoodt, Helena Todorov, and Yvan Saeys. A comparison of single-cell trajectory inference methods. *Nature Biotechnology*, 37(5):547–554, April 2019.

[8] Lijia Yu, Yue Cao, Jean Y. H. Yang, and Pengyi Yang. Benchmarking clustering algorithms on estimating the number of cell types from single-cell rna-sequencing data. *Genome Biology*, 23(1), 2022.

[9] Arash Shahsavari, Andi Munteanu, and Irina Mohorianu. Clustassess: tools for assessing the robustness of single-cell clustering. February 2022.

[10] Jacob H. Levine, Erin F. Simonds, Sean C. Bendall, Kara L. Davis, El-ad D. Amir, Michelle D. Tadmor, Oren Litvin, Harris G. Fienberg, Astraea Jager, Eli R. Zunder, Rachel Finck, Amanda L. Gedman, Ina Radtke, James R. Downing, Dana Pe'er, and Garry P. Nolan. Data-driven phenotypic dissection of aml reveals progenitor-like cells that correlate with prognosis. *Cell*, 162(1):184–197, 2015.

[11] Tim Stuart, Andrew Butler, Paul Hoffman, Christoph Hafemeister, Efthymia Papalexi, William M. Mauck, Yuhan Hao, Marlon Stoeckius, Peter Smibert, and Rahul Satija. Comprehensive integration of single-cell data. *Cell*, 177(7):1888–1902.e21, 2019.

[12] Junyue Cao, Malte Spielmann, Xiaojie Qiu, Xingfan Huang, Daniel M. Ibrahim, Andrew J. Hill, Fan Zhang, Stefan Mundlos, Lena Christiansen, Frank J. Steemers, Cole Trapnell, and Jay Shendure. The single-cell transcriptional landscape of mammalian organogenesis. *Nature*, 566(7745):496–502, February 2019.

[13] F. Alexander Wolf, Philipp Angerer, and Fabian J. Theis. Scanpy: large-scale single-cell gene expression data analysis. *Genome Biology*, 19(1), 2018.

[14] Alexander J. Gates, Ian B. Wood, William P. Hetrick, and Yong-Yeol Ahn. Element-centric clustering comparison unifies overlaps and hierarchy. *Scientific Reports*, 9(1), 2019.

[15] Lukas Heumos, Anna C. Schaar, Christopher Lance, Anastasia Litinetskaya, Felix Drost, Luke Zappia, Malte D. Lücken, Daniel C. Strobl, Juan Henao, Fabiola Curion, Hananeh Aliee, Meshal Ansari, Pau Badia-i Mompel, Maren Büttner, Emma Dann, Daniel Dimitrov, Leander Dony, Amit Frishberg, Dongze He, Soroor Hediyeh-zadeh, Leon Hetzel, Ignacio L. Ibarra, Matthew G. Jones, Mohammad Lotfollahi, Laura D. Martens, Christian L. Müller, Mor Nitzan, Johannes Ostner, Giovanni Palla, Rob Patro, Zoe Piran, Ciro Ramírez-Suástegui, Julio Saez-Rodriguez, Hirak Sarkar, Benjamin Schubert, Lisa Sikkema, Avi Srivastava, Jovan Tanevski, Isaac Virshup, Philipp Weiler, Herbert B. Schiller, and Fabian J. Theis. Best practices for single-cell analysis across modalities. *Nature Reviews Genetics*, 24(8):550–572, 2023.

[16] Mohammad Lotfollahi, Yuhan Hao, Fabian J. Theis, and Rahul Satija. The future of rapid and automated single-cell data analysis using reference mapping. *Cell*, 187(10):2343–2358, 2024.

[17] Jose Valente de Oliveira and Witold Pedrycz, editors. *Advances in Fuzzy Clustering and its Applications*. John Wiley & Sons, Nashville, TN, 2007.

[18] Marco De Zuani, Haoliang Xue, Jun Sung Park, Stefan C. Dentro, Zaira Seferbekova, Julien Tessier, Sandra Curras-Alonso, Angela Hadjipanayis, Emmanouil I. Athanasiadis, Moritz Gerstung, Omer Bayraktar, and Ana Cvejic. Single-cell and spatial transcriptomics analysis of non-small cell lung cancer. *Nature Communications*, 15(1), 2024.

[19] Simone G. Riva, Brynelle Myers, Paolo Cazzaniga, Francesca M. Buffa, and Andrea Tangherloni. Consensus clustering strategy for cell type assignments of scrna-seq data. In *2023 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*. IEEE, 2023.

[20] Wouter Saelens, Robrecht Cannoodt, and Yvan Saeys. A comprehensive evaluation of module detection methods for gene expression data. *Nature Communications*, 9(1), 2018.

[21] Khalid Raza. Fuzzy logic based approaches for gene regulatory network inference. *Artificial Intelligence in Medicine*, 97:189–203, 2019.

[22] Chang H. Seo, Jeong-Rae Kim, Man-Sun Kim, and Kwang-Hyun Cho. Hub genes with positive feedbacks function as master switches in developmental gene regulatory networks. *Bioinformatics*, 25(15):1898–1904, 2009.

[23] Monique G. P. van der Wijst, Dylan H. de Vries, Harm Brugge, Harm-Jan Westra, and Lude Franke. An integrative approach for building personalized gene regulatory networks for precision medicine. *Genome Medicine*, 10(1), 2018.

[24] Irina Mohorianu, Emily K. Fowler, Tamas Dalmay, and Tracey Chapman. Control of seminal fluid protein expression via regulatory hubs in drosophila melanogaster. *Proceedings of the Royal Society B: Biological Sciences*, 285(1887):20181681, 2018.

[25] Marlon Stoeckius, Christoph Hafemeister, William Stephenson, Brian Houck-Loomis, Pratip K Chattopadhyay, Harold Swerdlow, Rahul Satija, and Peter Smibert. Simultaneous epitope and transcriptome measurement in single cells. *Nature Methods*, 2017.

[26] Malte D. Luecken and Fabian J. Theis. Current best practices in single-cell rna-seq analysis: A tutorial. *Molecular Systems Biology*, 15(6), 2019.

[27] Christoph Hafemeister and Rahul Satija. Normalization and variance stabilization of single-cell rna-seq data using regularized negative binomial regression. *Genome Biology*, 20(1), 2019.

[28] James Baglama. *Fastpartialsingularvaluedecompositionmethod*. Handbook of Big Data, 2016.

[29] Kevin R. Moon, David van Dijk, Zheng Wang, Scott Gigante, Daniel B. Burkhardt, William S. Chen, Kristina Yim, Antonia van den Elzen, Matthew J. Hirn, Ronald R. Coifman, Natalia B. Ivanova, Guy Wolf, and Smita Krishnaswamy. Visualizing structure and transitions in high-dimensional biological data. *Nature Biotechnology*, 37(12):1482–1492, 2019.

[30] Tara Chari and Lior Pachter. The specious art of single-cell genomics. 2021.

[31] Pengyi Yang, Hao Huang, and Chunlei Liu. Feature selection revisited in the single-cell era. *Genome Biology*, 22(1), Dec 2021.

[32] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69(2), 2004.

[33] Jörg Reichardt and Stefan Bornholdt. Statistical mechanics of community detection. *Physical Review E*, 74(1), 2006.

[34] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008.

[35] Randolf Rotta and Andreas Noack. Multilevel local search algorithms for modularity clustering. *ACM Journal of Experimental Algorithmics*, 16, 2011.

[36] Ludo Waltman and Nees Jan van Eck. A smart local moving algorithm for large-scale modularity-based community detection. *The European Physical Journal B*, 86(11), 2013.

[37] Vincent A. Traag, Ludo Waltman, and Nees Jan. van Eck. From louvain to leiden: guaranteeing well-connected communities. *Scientific Reports*, 9(1), 2019.

[38] Anoop P. Patel, Itay Tirosh, John J. Trombetta, Alex K. Shalek, Shawn M. Gillespie, Hiroaki Wakimoto, Daniel P. Cahill, Brian V. Nahed, William T. Curry, Robert L. Martuza, David N. Louis, Orit Rozenblatt-Rosen, Mario L. Suvà, Aviv Regev, and Bradley E. Bernstein. Single-cell rna-seq highlights intratumoral heterogeneity in primary glioblastoma. *Science*, 344(6190):1396–1401, 2014.

[39] Inmaculada Gutiérrez, Daniel Gómez, Javier Castro, and Rosa Espínola. *A New Community Detection Algorithm Based on Fuzzy Measures*, page 133–140. Springer International Publishing, 2019.

[40] Inmaculada Gutiérrez, Daniel Gómez, Javier Castro, and Rosa Espínola. From fuzzy information to community detection: An approach to social networks analysis with soft information. *Mathematics*, 10(22):4348, 2022.

[41] Inmaculada Gutiérrez, Daniel Gómez, Javier Castro, and Rosa Espínola. *A New Community Detection Problem Based on Bipolar Fuzzy Measures*, page 91–99. Springer International Publishing, 2022.

[42] Julia Åkesson, Zelmina Lubovac-Pilav, Rasmus Magnusson, and Mika Gustafsson. Comhub: Community predictions of hubs in gene regulatory networks. *BMC Bioinformatics*, 22(1), 2021.

[43] Reut Shalgi, Daniel Lieber, Moshe Oren, and Yitzhak Pilpel. Global and local architecture of the mammalian microrna–transcription factor regulatory network. *PLoS Computational Biology*, 3(7):e131, 2007.

[44] Masaru Koido, Yuri Tani, Satomi Tsukahara, Yuka Okamoto, and Akihiro Tomida. In-depth: detection of hub genes for developing gene expression networks under anticancer drug treatment. *Oncotarget*, 9(49):29097–29111, 2018.

[45] Kelly Street, Davide Risso, Russell B. Fletcher, Diya Das, John Ngai, Nir Yosef, Elizabeth Purdom, and Sandrine Dudoit. Slingshot: cell lineage and pseudotime inference for single-cell transcriptomics. *BMC Genomics*, 19(1), 2018.

[46] Dominic Grün, Mauro J. Muraro, Jean-Charles Boisset, Kay Wiebrands, Anna Lyubimova, Gitanjali Dharmadhikari, Maaike van den Born, Johan van Es, Erik Jansen, Hans Clevers, Eelco J.P. de Koning, and Alexander van Oudenaarden. De novo prediction of stem cell identity using single-cell transcriptome data. *Cell Stem Cell*, 19(2):266–277, 2016.

[47] Xiaojie Qiu, Qi Mao, Ying Tang, Li Wang, Raghav Chawla, Hannah A Pliner, and Cole Trapnell. Reversed graph embedding resolves complex single-cell trajectories. *Nature Methods*, 14(10):979–982, 2017.

[48] Qian Zhu, Aaron K Wong, Arjun Krishnan, Miriam R Aure, Alicja Tadych, Ran Zhang, David C Corney, Casey S Greene, Lars A Bongo, Vessela N Kristensen, Moses Charikar, Kai Li, and Olga G Troyanskaya. Targeted exploration and analysis of large cross-platform human transcriptomic compendia. *Nature Methods*, 12(3):211–214, 2015.

[49] Laia Alsina, Elisabeth Israelsson, Matthew C Altman, Kristen K Dang, Pegah Ghandil, Laura Israel, Horst von Bernuth, Nicole Baldwin, Huanying Qin, Zongbo Jin, Romain Banchereau, Esperanza Anguiano, Alexei Ionan, Laurent Abel, Anne Puel, Capucine Picard, Virginia Pascual, Jean Laurent Casanova, and Damien Chaussabel. A narrow repertoire of transcriptional modules responsive to pyogenic bacteria is impaired in patients carrying loss-of-function mutations in myd88 or irak4. *Nature Immunology*, 15(12):1134–1142, 2014.

[50] Tuqyah Abdullah Al Qazlan, Aboubekeur Hamdi-Cherif, and Chafia Kara-Mohamed. State of the art of fuzzy methods for gene regulatory networks inference. *The Scientific World Journal*, 2015:1–11, 2015.

[51] Jeremiah J Faith, Boris Hayete, Joshua T Thaden, Ilaria Mogno, Jamey Wierzbowski, Guillaume Cottarel, Simon Kasif, James J Collins, and Timothy S Gardner. Large-scale mapping and validation of escherichia coli transcriptional regulation from a compendium of expression profiles. *PLoS Biology*, 5(1):e8, 2007.

[52] Adam A Margolin, Ilya Nemenman, Katia Basso, Chris Wiggins, Gustavo Stolovitzky, Riccardo Dalla Favera, and Andrea Califano. Aracne: An algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context. *BMC Bioinformatics*, 7(S1), 2006.

[53] Vân Anh Huynh-Thu, Alexandre Irrthum, Louis Wehenkel, and Pierre Geurts. Inferring regulatory networks from expression data using tree-based methods. *PLoS ONE*, 5(9):e12776, 2010.

[54] Thomas Moerman, Sara Aibar Santos, Carmen Bravo González-Blas, Jaak Simm, Yves Moreau, Jan Aerts, and Stein Aerts. Grnboost2 and arboreto: efficient and scalable inference of gene regulatory networks. *Bioinformatics*, 35(12):2159–2161, 2018.

[55] Kevin Ushey, JJ Allaire, and Yuan Tang. *reticulate: Interface to 'Python'*, 2024. R package version 1.38.0, https://github.com/rstudio/reticulate.

[56] Jessica Vamathevan, Dominic Clark, Paul Czodrowski, Ian Dunham, Edgardo Ferran, George Lee, Bin Li, Anant Madabhushi, Parantu Shah, Michaela Spitzer, and Shanrong Zhao. Applications of machine learning in drug discovery and development. *Nature Reviews Drug Discovery*, 18(6):463–477, 2019.

[57] José Jiménez-Luna, Francesca Grisoni, and Gisbert Schneider. Drug discovery with explainable artificial intelligence. *Nature Machine Intelligence*, 2(10):573–584, 2020.

[58] Jinkuk Kim, Sijae Woo, Claudio M. de Gusmao, Boxun Zhao, Diana H. Chin, Renata L. DiDonato, Minh A. Nguyen, Tojo Nakayama, Chunguang April Hu, Aubrie Soucy, Ashley Kuniholm, Jennifer Karlin Thornton, Olivia Riccardi, Danielle A. Friedman, Christelle Moufawad El Achkar, Zane Dash, Laura Cornelissen, Carolina Donado, Kamli N. W. Faour, Lynn W. Bush, Victoria Suslovitch, Claudia Lentucci, Peter J. Park, Eunjung Alice Lee, Al Patterson, Anthony A. Philippakis, Brad Margus, Charles B. Berde, and Timothy W. Yu. A framework for individualized splice-switching oligonucleotide therapy. *Nature*, 619(7971):828–836, 2023.

[59] Marlen C. Lauffer, Willeke van Roon-Mom, and Annemieke Aartsma-Rus. Possibilities and limitations of antisense oligonucleotide therapies for the treatment of monogenic disorders. *Communications Medicine*, 4(1), January 2024.

[60] Andi Munteanu. Automatic clustassess pipeline. generating the clustassess shiny-app, 2024.

[61] Liis Kolberg, Uku Raudvere, Ivan Kuzmin, Jaak Vilo, and Hedi Peterson. gprofiler2– an r package for gene list functional enrichment analysis and namespace conversion toolset g:profiler. *F1000Research*, 9 (ELIXIR)(709), 2020. R package version 0.2.3.

[62] Mengwei Li, Xiaowei Chi, Ying Wang, Sarra Setrerrahmane, Wenwei Xie, and Hanmei Xu. Trends in insulin resistance: insights into mechanisms and therapeutic strategy. *Signal Transduction and Targeted Therapy*, 7(1), 2022.

[63] Zobair Younossi, Quentin M. Anstee, Milena Marietti, Timothy Hardy, Linda Henry, Mohammed Eslam, Jacob George, and Elisabetta Bugianesi. Global burden of nafld and nash: trends, predictions, risk factors and prevention. *Nature Reviews Gastroenterology amp; Hepatology*, 15(1):11–20, 2017.

[64] Yalan Zhu, He Zhang, Pengjun Jiang, Chengxia Xie, Yao Luo, and Jie Chen. Transcriptional and epigenetic alterations in the progression of non-alcoholic fatty liver disease and biomarkers helping to diagnose non-alcoholic steatohepatitis. *Biomedicines*, 11(3):970, 2023.

[65] Yue Hu and Jun Zhou. Identification of key genes and functional enrichment analysis of liver fibrosis in nonalcoholic fatty liver disease through weighted gene co-expression network analysis. *Genomics amp; Informatics*, 21(4):e45, 2023.

[66] Shigeki Nakagawa, Lan Wei, Won Min Song, Takaaki Higashi, Sarani Ghoshal, Rosa S. Kim, C. Billie Bian, Suguru Yamada, Xiaochen Sun, Anu Venkatesh, Nicolas Goossens, Gretchen Bain, Gregory Y. Lauwers, Anna P. Koh, Mohamed El-Abtah, Noor B. Ahmad, Hiroki Hoshida, Derek J. Erstad, Ganesh Gunasekaran, Youngmin Lee, Ming-Lung Yu, Wan-Long Chuang, Chia-Yen Dai, Masahiro Kobayashi, Hiromitsu Kumada, Toru Beppu, Hideo Baba, Milind Mahajan, Venugopalan D. Nair, Michael Lanuti, Augusto Villanueva, Angelo Sangiovanni, Massimo Iavarone, Massimo Colombo, Josep M. Llovet, Aravind Subramanian, Andrew M. Tager, Scott L. Friedman, Thomas F. Baumert, Myron E. Schwarz, Raymond T. Chung, Kenneth K. Tanabe, Bin Zhang, Bryan C. Fuchs, and Yujin Hoshida. Molecular liver cancer prevention in cirrhosis by organ transcriptome analysis and lysophosphatidic acid pathway inhibition. *Cancer Cell*, 30(6):879–890, 2016.

[67] Fu Chen, Yong Zhou, Zhiyuan Wu, Yunze Li, Wenlong Zhou, and Yong Wang. Integrated analysis of key genes and pathways involved in nonalcoholic steatohepatitis improvement after roux-en-y gastric bypass surgery. *Frontiers in Endocrinology*, 11, 2021.

[68] Yuxiang Lin, Jianxing Zhang, Xiaoli Li, Dezhu Zheng, Xiurong Yu, Yichu Liu, Fenghua Lan, and Zhihong Wang. Biallelic mutations in dcdc2 cause neonatal sclerosing cholangitis in a chinese family. *Clinics and Research in Hepatology and Gastroenterology*, 44(5):e103–e108, 2020.

[69] Qing-Qing Liu, Jing Chen, Tao Ma, Wei Huang, and Cui-Hua Lu. Dcdc2 inhibits hepatic stellate cell activation and ameliorates ccl4-induced liver fibrosis by suppressing wnt/-catenin signaling. *Scientific Reports*, 14(1), 2024.

[70] Maria Ryaboshapkina and Mårten Hammar. Human hepatic gene expression signature of non-alcoholic fatty liver disease progression, a meta-analysis. *Scientific Reports*, 7(1), 2017.

[71] Naga Chalasani, Xiuqing Guo, Rohit Loomba, Mark O. Goodarzi, Talin Haritunians, Soonil Kwon, Jinrui Cui, Kent D. Taylor, Laura Wilson, Oscar W. Cummings, Yii-Der Ida Chen, and Jerome I. Rotter. Genome-wide association study identifies variants associated with histologic features of nonalcoholic fatty liver disease. *Gastroenterology*, 139(5):1567–1576.e6, 2010.

[72] Vishnubhotla Venkata Ravi Kanth. Pooled genetic analysis in ultrasound measured non-alcoholic fatty liver disease in indian subjects: A pilot study. *World Journal of Hepatology*, 6(6):435, 2014.

[73] Hamideh Dehghan, Alireza Ghasempour, Mahboobeh Sabeti akbar abad, Zahra Khademi, Mahsa Sedighi, Tannaz Jamialahmadi, and Amirhossein Sahebkar. *An update on the therapeutic role of RNAi in NAFLD/NASH*, page 45–67. Elsevier, 2024.

[74] Zi-Ying Yuan, Xing-Xin Zhang, Yu-Jing Wu, Zhi-Ping Zeng, Wei-Min She, Shi-Yao Chen, Yuan-Qing Zhang, and Jin-Sheng Guo. Serum amyloid a levels in patients with liver diseases. *World Journal of Gastroenterology*, 25(43):6440–6450, 2019.

[75] Bin Jiang, Dongdong Wang, Yunfu Hu, Wenxuan Li, Fengjiang Liu, Xudong Zhu, Xiaoyu Li, Hanwen Zhang, Hui Bai, Qing Yang, Xiuna Yang, Jingjing Ben, and Qi Chen. Serum amyloid a1 exacerbates hepatic steatosis via tlr4-mediated nf-b signaling pathway. *Molecular Metabolism*, 59:101462, 2022.

[76] Tao Zhang, Na Zhang, Jing Xing, Shuhua Zhang, Yulu Chen, Daichao Xu, and Jinyang Gu. Udp-glucuronate metabolism controls ripk1-driven liver damage in nonalcoholic steatohepatitis. *Nature Communications*, 14(1), 2023.

[77] Max C. Petersen, Anila K. Madiraju, Brandon M. Gassaway, Michael Marcel, Ali R. Nasiri, Gina Butrico, Melissa J. Marcucci, Dongyan Zhang, Abudukadier Abulizi, Xian-Man Zhang, William Philbrick, Stevan R. Hubbard, Michael J. Jurczak, Varman T. Samuel, Jesse Rinehart, and Gerald I. Shulman. Insulin receptor thr1160 phosphorylation mediates lipid-induced hepatic insulin resistance. *Journal of Clinical Investigation*, 126(11):4361–4371, 2016.

[78] Kasper W. ter Horst, Pim W. Gilijamse, Ruth I. Versteeg, Mariette T. Ackermans, Aart J. Nederveen, Susanne E. la Fleur, Johannes A. Romijn, Max Nieuwdorp, Dongyan Zhang, Varman T. Samuel, Daniel F. Vatner, Kitt F. Petersen, Gerald I. Shulman, and Mireille J. Serlie. Hepatic diacylglycerol-associated protein kinase c translocation links hepatic steatosis to hepatic insulin resistance in humans. *Cell Reports*, 19(10):1997–2004, 2017.

[79] Markus Ahrens, Ole Ammerpohl, Witigo von Schönfels, Julia Kolarova, Susanne Bens, Timo Itzel, Andreas Teufel, Alexander Herrmann, Mario Brosch, Holger Hinrichsen, Wiebke Erhart, Jan Egberts, Bence Sipos, Stefan Schreiber, Robert Häsler, Felix Stickel, Thomas Becker, Michael Krawczak, Christoph Röcken, Reiner Siebert, Clemens Schafmayer, and Jochen Hampe. Dna methylation analysis in nonalcoholic fatty liver disease suggests distinct disease-specific and remodeling signatures after bariatric surgery. *Cell Metabolism*, 18(2):296–302, 2013.

[80] Varman T. Samuel, Zhen-Xiang Liu, Xianqin Qu, Benjamin D. Elder, Stefan Bilz, Douglas Befroy, Anthony J. Romanelli, and Gerald I. Shulman. Mechanism of hepatic insulin resistance in non-alcoholic fatty liver disease. *Journal of Biological Chemistry*, 279(31):32345–32353, 2004.

[81] Volker Bergen, Marius Lange, Stefan Peidli, F. Alexander Wolf, and Fabian J. Theis. Generalizing rna velocity to transient cell states through dynamical modeling. *Nature Biotechnology*, 38(12):1408–1414, August 2020.

[82] Angelo Duò, Mark D. Robinson, and Charlotte Soneson. A systematic performance evaluation of clustering methods for single-cell rna-seq data. *F1000Research*, 7:1141, July 2018.

[83] Brent Ewing. leidenbase: R and c/c++ wrappers to run the leiden find$_p artition()function, February$2022.

[84] David Lähnemann, Johannes Köster, Ewa Szczurek, Davis J. McCarthy, Stephanie C. Hicks, Mark D. Robinson, Catalina A. Vallejos, Kieran R. Campbell, Niko Beerenwinkel, Ahmed Mahfouz, Luca Pinello, Pavel Skums, Alexandros Stamatakis, Camille Stephan-Otto Attolini, Samuel Aparicio, Jasmijn Baaijens, Marleen Balvert, Buys de Barbanson, Antonio Cappuccio, Giacomo Corleone, Bas E. Dutilh, Maria Florescu, Victor Guryev, Rens Holmer, Katharina Jahn, Thamar Jessurun Lobo, Emma M. Keizer, Indu Khatri, Szymon M. Kielbasa, Jan O. Korbel, Alexey M. Kozlov, Tzu-Hao Kuo, Boudewijn P.F. Lelieveldt, Ion I. Mandoiu, John C. Marioni, Tobias Marschall, Felix Mölder, Amir Niknejad, Alicja Rączkowska, Marcel Reinders, Jeroen de Ridder, Antoine-Emmanuel Saliba, Antonios Somarakis, Oliver Stegle, Fabian J. Theis, Huan Yang, Alex Zelikovsky, Alice C. McHardy, Benjamin J. Raphael, Sohrab P. Shah, and Alexander Schönhuth. Eleven grand challenges in single-cell data science. *Genome Biology*, 21(1), February 2020.

Appendix A

Proposal

Fuzzy vs Crisp clustering of single-cell RNAseq expression data and its potential on revealing the dynamics of Gene Regulatory Networks

Word count: 2,194

1. Introduction

1.1 scRNA-seq, clustering usefulness, pre-processing considerations, and evaluation

Single-cell RNA sequencing (scRNA-seq) reshaped our understanding of cellular diversity and process dynamics in complex tissues by revealing individual cell transcriptional profiles [1]. Its significance lies in exposing variability within and between cells in heterogenous populations, identifying new cell states, enhancing our understanding of cell types [22-24].

In scRNA-seq analyses, clustering is key for partitioning cells into distinct groups based on transcriptional profiles, identifying unique cell-markers and rare populations [2], setting the stage for further analyses like trajectory inference [39]. Due to technical variations/noise, analyses rely on thorough pre-processing, including quality-control, normalization, feature selection, dimensionality reduction, and batch correction [59]. Appendix 1 details standard pre-processing steps and methodologies.

Standard clustering evaluation metrics like (adjusted) Rand Index (RI/ARI) are cluster-centric [59]. Recently, Element-Centric similarity (ECS) was introduced [17], focusing on individual elements instead of clusters, computing similarity scores per element and averaging these for an overall partition-score. ECS avoids common biases (see Appendix 2 for details) deriving from variable cluster sizes [17,11].

1.2 Common scRNA-seq clustering approaches, stochasticity and the need for a fuzzy approach

Various clustering methods are commonly employed to partition cells into cell-types. A popular approach is k-means [33], which iteratively identifies a user-specified number (k) of cluster centroids by minimizing a distance (usually squared Euclidean distance) between cells and their closest centroid. Several scRNA-seq clustering tools utilising k-means include SAIC [34] and RaceID [35]. However, k-means fails to effectively identify rare cell types and can have convergence issues due to its assumptions of hyperspheric clusters and similar sizes [61]. Another approach, hierarchical clustering, progressively merges (agglomerative) or splits (divisive) cells into clusters, revealing hierarchical relationships without distribution assumptions [61]. The method is computationally demanding and prone to local optima [61]. Tools using hierarchical clustering include CIDR [36], pcaReduce [37], and SINCERA [38].

Community detection algorithms emerged as superior clustering techniques for identifying cellular subpopulations [14]. Treating single-cell datasets as networks, with cells as nodes and their similarities as weighted edges, their goal is to cluster cells, uncovering groups of transcriptionally related cells [3]. The state-of-the art community detection algorithms include Louvain [4], a two-step approach, focused on greedily optimizing an objective function to establish optimal graph partitioning. Enhancements to Louvain include multi-level refinement [5], Smart Local Moving (SLM) [6], and the newer Leiden algorithm [7].

All cutting-edge community detection techniques are inherently stochastic [11], with repeated runs on identical data, with varying random seeds, yielding different results, compromising the robustness and reproducibility of clustering. Steps like dimensionality reduction, graph construction, and feature selection also contribute to the instability. ClustAssess [11] employs ECS for evaluating stable cluster configurations and pertinent parameters. This aids in data-driven decisions regarding dimensionality reduction (UMAP/PCA), graph construction methods (nearest-neighbours/shared-nearest-neighbours), and clustering algorithms (Leiden/SLM).

These approaches uniquely assign each cell to specific clusters (crisp). Their stochastic nature provides an opportunity to identify ambiguously classified cells, across different seed values, thereby uncovering the nuanced, dynamic nature of cell populations.

1.3 Fuzzy/consensus clustering

Fuzzy clustering algorithms allow the assignation of data-points to multiple clusters, allocating a probability to each assignment [16]. This facilitates the identification of transitional/intermediate cells, often overlooked by traditional crisp methods. This matches developmental biology, where cells often exist in a continuum of states [40].

SC3 [13] showed that a consensus-based method integrating various clustering solutions may substantially improve clustering robustness/accuracy, as the bias of any single configuration is reduced [41]. Its dependency on k-means limits its scalability, particularly for datasets larger than 5,000 cells[1], a concern given that average scRNA-seq datasets in 2020 exceeded 50,000 cells [42]. scALPO [15] proposes another consensus/ensemble approach, combining different algorithms (like k-means, agglomerative, Leiden) in a pseudo-voting system for cell type assignment. Additionally, Gutierrez et al. [12] proposed a Louvain model extension for fuzzy network data, requiring extra information, independent of the graph's structure, on node affinity.

1.4 Pseudotime, GRNs, and research gaps

Recently, scRNA-seq research evolved from identifying differential gene expression (DE/DEG) to investigating the causality/directionality in regulatory dynamics. Central to this are the inference of pseudotime trajectories and gene regulatory networks (GRNs). The former arranges cells along a hypothetical timeline, reflecting developmental progression and is pivotal for characterising dynamic cellular processes [43]. GRNs provide a structured representation of causality/directionality of gene interactions, with nodes representing genes and edges summarising regulatory relationships [44]. Nodes with high connectivity are of particular interest as targeting them may cause a ripple-effect, propagating changes throughout the network and altering phenotypes [45-48].

While I note an increase in applying consensus clustering approaches to scRNA-seq data, existing literature does not focus on evaluating the downstream impact of unstable cells on the interpretation of dynamics of regulatory interactions. I hypothesize that fuzzy assignation of cells will impact the interpretation of pseudotime trajectories in terms of patterns of DE genes, potentially revealing intermediate states not observable on crisp partitions. The degree of fuzziness can be used to identify bottlenecks in GRN dynamics, capturing essential transitional phases in disease progression that would have otherwise been missed in a crisp setting.

[1] The authors also presented a hybrid method for clustering large datasets: it samples 5000 cells for SC3 clustering, then trains a support vector machine with these labels to classify the remaining cells. However, when N>>5,000 there is a substantial risk that the sampled distribution will differ significantly from the full distribution.

2. Aims

This project aims to develop and deploy an R package proposing a novel clustering framework, based on a fuzzy paradigm. The optimised implementation of the algorithm will benefit from a Shiny app interface, facilitating an easy, interactive data mining by wet-lab researchers, including visualisations assisting and enhancing the understanding of cellular dynamics/causality.

My primary objectives are:

1) Creating a novel fuzzy community detection algorithm, with direct applications to scRNA-seq clustering.

2) Development of functions for assessing the replacement of crisp with fuzzy clustering, particularly on pseudotime trajectory inference and GRN predictions.

3) Mathematically formulating a link between element-centric consistency (ECC) and entropy to enhance the evaluation of clustering outcomes within a fuzzy framework.

4) Creating an easy-to-use, interactive R shiny interface.

3. Methods

Figure 1 outlines the proposed project workflow. Detailed component descriptions follow in subsequent sections.

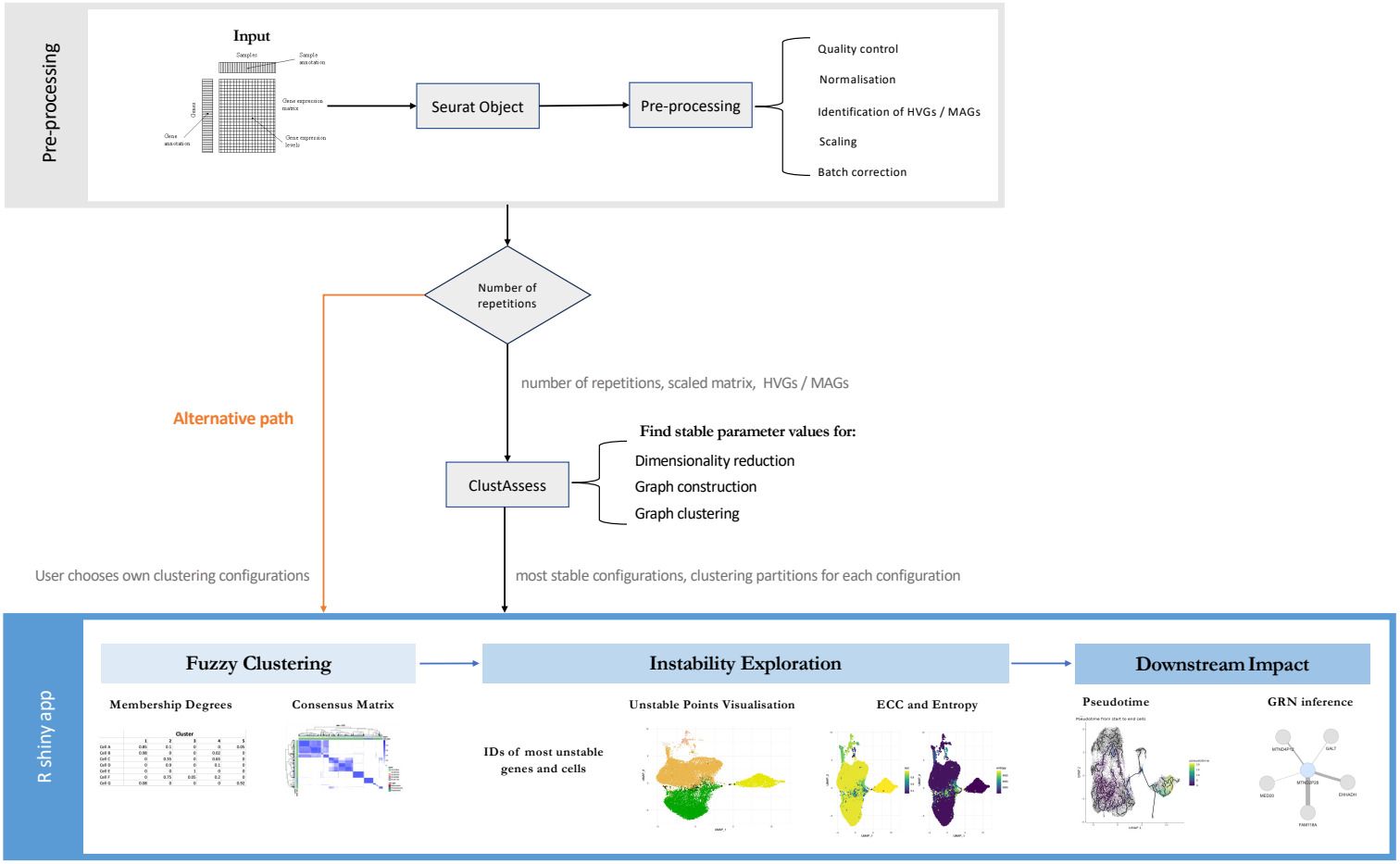


Figure 1: Flowchart summarising the proposed framework.

3.1 Data and Setup

This study will use a scRNA-seq non-alcoholic fatty liver disease (NAFLD)/non-alcoholic steatohepatitis (NASH) *H. sapiens* dataset subset, with 15,000 cells (across various conditions) and 30,000 genes. The count matrix is provided and is used to generate a Seurat object. Metadata with additional information (e.g. batch, cell-types) is available.

Code is developed and analyses run on R version 4.2.1 using a high-performance server (dual CPU @2.2GHz, 32 cores per CPU, 768G RAM, Linux Ubuntu). The code will be submitted on GitHub.

Table 1 lists essential libraries, their functions, and roles.

Table 1: Main libraries to be used during the development of the R package and shiny app. Each accompanied by an explanation of their purpose in development.

Library	Description	Purpose in development
Seurat [53] (version 5.0.1)	Designed for QC, analysis, and exploration of scRNA-seq data.	Load and pre-process (normalisation, scaling, identification of important features, dimensionality reduction) the scRNA-seq data. Seurat clustering functions will be used internally for the development of the alternative path fuzzy clustering algorithm.
Harmony [51] (version 0.1)	Single-cell data integration.	Correct batch effect - deriving from the different batches in the dataset.
ClustAssess [11] (version 1.0.0)	Assessing the robustness of partitions in a data-driven way using ECS/ECC.	Derive robust clustering configurations, on a predefined number of iterations, to be used for the fuzzy clustering.
Monocle3 [54] (version 1.3.1)	Analysis of scRNA-seq data to uncover cellular trajectories and differential expression.	Pseudotime trajectory inference.
SCENIC [56-58] (version 1.1.2)	Inferring gene regulatory networks from single-cell RNA-seq data.	Gene regulatory network inference.
Shiny [55] (version 1.8)	Package that makes it easy to build interactive web applications directly from R.	Development of an interactive web interface.

3.2 Pre-processing

The Seurat object will undergo quality control, normalization, identification of highly-variable genes (HVGs) and most-abundant genes (MAGs), scaling, and dimensionality reduction using PCA and UMAP. Harmony [51] will be used for the correction of batch effects. Table 2 summarizes the technical details for each pre-processing step.

Table 2: An overview of functions to be used for each pre-processing step.

Process	Step	Technical details/Function to be used
Quality Control	Eliminating cells with mitochondrial reads	Identify cells with mitochondrial reads using `PercentageFeatureSet()` and eliminate them using `subset()`. Both functions are from Seurat.
	Eliminating cells with few detected genes	`subset()` from Seurat, with a condition on the *nFeature_RNA* metadata. Filtering cells that have gene counts over 2500 or less than 200.
Normalisation	SCTransform	`SCTransform()` from Seurat, due to the nature of the 10x genomics data [9].
Feature Selection	HVGs identification	`FindVariableFeatures()` from Seurat. Top 2000 HVGs will be identified - common approach (see Appendix 1).
	MAGs identification	Manually sort the genes based on their expression level using base R `order()` function. Top 2000 MAGs will be identified – common approach (see Appendix 1).
Dimensionality Reduction	PCA UMAP	`RunPCA()` from Seurat. `RunUMAP()` from Seurat.
Batch Correction	Harmony correction	`RunHarmony()` from Harmony. The function takes a Seurat object. A theta value needs to be specified which determines how strong the correction is.

3.3 Identification of stable clustering configuration

Selecting a stable clustering configuration is an important preliminary step as it enhances the reliability of detecting unstable entries in an otherwise stable configuration.

ClustAssess proposes optimal values for resolution, graph type (kNN/sNN), feature selection, dimensionality reduction (UMAP/PCA), and clustering algorithms (Louvain/SLM/Leiden). Resulting partitions are exported as an R object.

3.4 Shiny app creation

A function will be developed to accept clustering partitions from ClustAssess, the pre-processed scaled matrix, and HVGs/MAGs as input and generate a Shiny app with three main tabs: fuzzy clustering, instability exploration, and downstream impact analysis.

3.4.1 Fuzzy clustering

ClustAssess returns partitions containing clustering results from the selected algorithm run multiple times (e.g., 1000) with different seeds.

Users may select clustering configuration partitions, guided by the app on the most stable configurations. From the selected configuration partitions, a consensus matrix is constructed displaying the frequency of co-clustering (cells assigned to the same cluster) between sample pairs over all the iterations. Values in the matrix will range from 0, signifying no joint clustering in any iteration, to 1, denoting consistent co-clustering in every run. Membership degrees for each cell will be calculated based on the frequency of a cell's association with each of the clusters over multiple iterations e.g. if cell A is grouped with cluster 1 in 600 of 1000 iterations, its membership degree for cluster 1 will be 0.6.

A custom function will be developed to enable users to compute partitions using their preferred algorithm (Louvain/SLM/Leiden - using Seurat's `FindClusters()`) and clustering configurations independently, which will then be used to compute the consensus matrix (alternative path - Figure 1). The consensus and membership degree matrices computation process remains the same.

3.4.2 Instability exploration

3.4.2.1 Identification of most unstable points

Users will be able to select the number of unstable points for analysis. Unstable points can be effectively identified by analysing their membership degrees. The most unstable entries are those with minimal differences between the highest and second-highest membership degrees, indicating almost equal probabilities of association with two clusters.

Equation 1: Entropy. Will be used to identify most unstable points.

$$H(X) = -\sum_{i=1}^{n} p_i \log_2 p_i$$

To detect instability across more than two clusters, I propose using entropy (Equation 1) on each cell's consensus matrix entry. High entropy suggests significant inconsistency in its co-clustering with others, whereas low entropy indicates consistent co-clustering behaviour.

3.4.2.2 Visualisations

Unstable points will be visualised on a UMAP, for which metadata will be available i.e. original clusters, ECC, entropy. This overview will enable a closer evaluation of regions of notable instability.

3.4.3 Downstream Impact

The following subsections discuss my plan about pseudotime trajectory and GRN inference. There is some uncertainty about the precise methodology I will use, which I hope to resolve as the project progresses. The topics will also be covered in the Applied Machine Learning Module.

3.4.3.1 Pseudotime trajectory

Custom functions will be developed using Monocle3 for dynamic analyses of cell trajectories, focusing on the impact of unstable cells identified in the previous step. This analysis will include:

- Assessing the implications of including/excluding unstable cells on the pseudotime, providing insights into disease progression.
- Identifying DE genes in stable versus unstable cell populations to understand causal regulatory interactions.

3.4.3.2 GRNs prediction

GRNs will be inferred using the SCENIC framework. GRNs will be generated for specific clusters or small groups of cells along the pseudotime axis to observe the evolution of regulatory interactions. Cell fuzziness enables the inclusion of additional cells within each cluster (those in transitional or ambiguous states) as input, providing a more comprehensive and accurate depiction of cellular dynamics.

3.5 Extensions

3.5.1 Element-centric consistency and entropy link

Frustration, or ECC, is a generalisation of ECS that measures how uniformly elements are categorized within a set of clusterings [17]. For a series of clusterings denoted as $R=\{R_1,\ldots,R_T\}$, the individual frustration for a specific element v_i is determined by Equation 2. S_i is the similarity (L1 distance) for that point in the two clusterings [17].

Equation 2: Element-centric consistency.

$$\frac{1}{\binom{T}{2}} \sum_{j=2}^{T} \sum_{k=1}^{j-1} S_i(\mathcal{R}_k, \mathcal{R}_j)$$

I will attempt to mathematically define the relationship between ECC and entropy, with the aim of ultimately producing a refined version of ECC for the fuzzy setting. Uncertainty exists about the existence of a link and my ability to prove it. However, I believe that the lectures and skills gained from Advanced Biostatistics for HDS and Bayesian Statistics will be highly beneficial.

4. Discussion

4.1 Fuzzy clustering

A membership degree matrix provides a precise characterization of each cell's identity and helps identify transitional cells, which are essential for understanding disease progression, and treatment responses [62][2]. Visualising the unstable cells enhances the interpretation by providing a context. However, the identification of unstable cells may overcomplicate the analysis/interpretation or potentially overshadow other significant cellular behaviours, especially if a high proportion of cells are classified as transitional. The number of clusters, dictated by the resolution, directly affects the identification of transitional points as they can only be between clusters; lower resolution values may uncover transitions between cell types; higher resolutions concentrate on nuances within cell states. Therefore, selecting a resolution value relies on a biologically informed selection process. I introduced ClustAssess as an intermediate step to help the researcher make decisions about values like resolution, considering the most stable results. Additionally, the accuracy/convergence of the results is contingent on the number of algorithm repetitions.

Fuzzy clustering's robustness is a notable advantage as through its repetitive nature it can yield more stable results. Yet, running the package can become computationally demanding with extremely large datasets, and scalability may require further optimization.

4.2 Downstream Impact

The inclusion/exclusion of fuzzy cells in pseudotime trajectory inference may result in different trajectories, uncovering transitional pathways or intermediate states, not apparent with only stable cells. Furthermore, it can also influence the profile of DE genes, potentially identifying those involved in state transitions. In developing GRNs, considering fuzzy cells in all their respective clusters could yield a more detailed view of cellular dynamics, especially in crucial transitional phases of disease progression. Ultimately better understanding of GRNs can allow future personalised treatments by targeting specific genes within the GRNs, that drive disease progression. The input data's quality and the choice of pre-processing parameters are critical and can skew the outcomes.

4.3 Entropy and ECS Link

Combining ECC and entropy would create a unified mathematical framework that encapsulates the stability and variability of clustering assignments, offering deeper insights into cell transition dynamics and optimizing clustering algorithms.

4.4 Shiny App Development

The Shiny app would facilitate access to an interactive (with real-time updates following parameter adjustments) interface that wet-lab researchers can use to perform in-depth clustering analysis and better understand cellular dynamics and causality. A potential limitation derives from heavy computational tasks slowing down interactivity, especially with large datasets.

[2] Paper is currently under review in Nature. I had access to it through my supervisor.

Possible risks and contingency plans are shown on Table 3.

Table 3: Overview of possible risks and discussion of contigency plans to mitigate them.

Risk	Effect	Likelihood	Severity	Contingency plan
Limited time for software development and dissertation writing	Inability to complete all content	Medium	High	• Develop a detailed project timeline with allocated time for each phase. • Prioritize features and tasks based on importance and impact. • Begin dissertation writing early, documenting each section upon its completion while progressing with the next phase of software development. • If necessary, adjust project scope to ensure the completion of critical tasks.
Insufficient Computing Resources	Inability to develop the software fast and test it on extremely large datasets	Low	Moderate	• Discuss with my supervisor and systems administrator and request access to more/better performance servers.
Computer failure or file corruption	Loss of important software development /dissertation writing	Low	High	• Regular backups on multiple platforms (cloud storage, external drives, servers). • Use version control systems like GitHub for code and document versioning.
Coding errors	No/Incorrect results	Moderate	Moderate	• Adhere to the DRY (Don't Repeat Yourself) principle to minimize redundant code. • Implement unit testing and code review processes. • Maintain a well-documented codebase for easier troubleshooting.
Illness	Delayed Progress	Low	Low	• Include a 2-week period for contingency issues in the timeline just in case.

5. Timeline

TASK	Progress
Research Plan	100%
Preliminary work	
Access to data	100%
Getting familiar with Seurat, ClustAssess, Monocle3, Scenic, Shiny	100%
Studying community-detection algorithms (Louvain, SLM, Leiden)	100%
Main Functions/Methods Development	
Pre-processing wrapper functions	100%
Fuzzy clustering algorithm development	100%
Fuzzy clustering alternative path	0%
Pseudotime functions	0%
GRN functions	0%
Evaluating package code so far and testing	0%
Maths	
Try to find link between Entopy and ECS	0%
R shiny app	
Main initial tab	0%
Instability exploration tab	0%
Downstream impact tab	0%
Polishing app	0%
Dissertation write-up	
Main report writing	0%
Buffer zone	0%

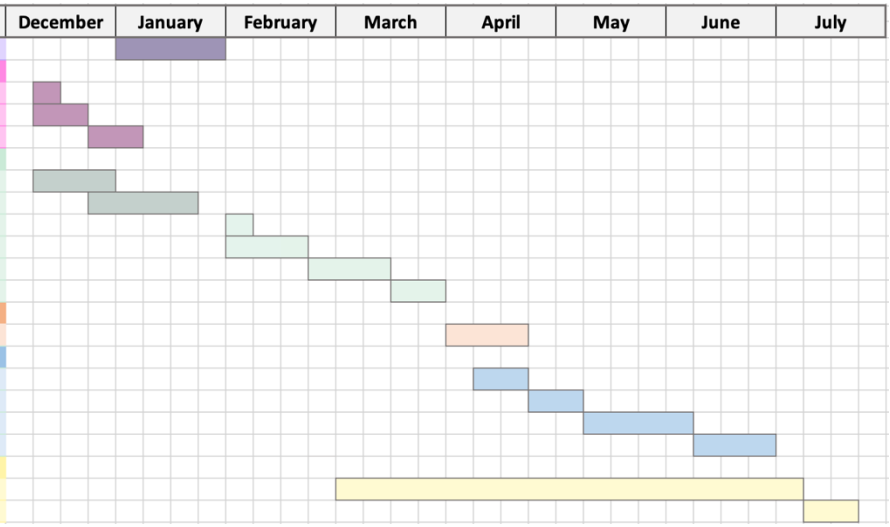


Figure 2: Timeline of the project

References:

[1] Chen G, Ning B, Shi T. Single-cell RNA-seq technologies and related computational data analysis. Frontiers in Genetics. 2019;10. doi:10.3389/fgene.2019.00317

[2] Kiselev VY, Kirschner K, Schaub MT, Andrews T, Yiu A, Chandra T, et al. SC3: Consensus clustering of single-cell RNA-seq data. Nature Methods. 2017;14(5):483–6. doi:10.1038/nmeth.4236

[3] Lee MY, Li M. Integration of multi-modal single-cell data. Nature Biotechnology. 2023 May 25; doi:10.1038/s41587-023-01826-4

[4] Blondel VD, Guillaume J-L, Lambiotte R, Lefebvre E. Fast unfolding of communities in large networks. Journal of Statistical Mechanics: Theory and Experiment. 2008;2008(10). doi:10.1088/1742-5468/2008/10/p10008

[5] Rotta R, Noack A. Multilevel local search algorithms for modularity clustering. ACM Journal of Experimental Algorithmics. 2011;16. doi:10.1145/1963190.1970376

[6] Waltman L, van Eck NJ. A smart local moving algorithm for large-scale modularity-based community detection. The European Physical Journal B. 2013;86(11). doi:10.1140/epjb/e2013-40829-0

[7] Traag VA, Waltman L, van Eck NJ. From Louvain to Leiden: Guaranteeing well-connected communities. Scientific Reports. 2019;9(1). doi:10.1038/s41598-019-41695-z

[8] Butler A, Hoffman P, Smibert P, Papalexi E, Satija R. Integrating single-cell transcriptomic data across different conditions, technologies, and species. Nature Biotechnology. 2018;36(5):411–20. doi:10.1038/nbt.4096

[9] Choudhary S, Satija R. Comparison and evaluation of statistical error models for scrna-seq. Genome Biology. 2022;23(1). doi:10.1186/s13059-021-02584-9

[10] Levine JH, Simonds EF, Bendall SC, Davis KL, Amir ED, Tadmor MD, et al. Data-driven phenotypic dissection of AML reveals progenitor-like cells that correlate with prognosis. Cell. 2015;162(1):184–97. doi:10.1016/j.cell.2015.05.047

[11] Shahsavari A, Munteanu A, Mohorianu I. ClustAssess: Tools for assessing the robustness of single-cell clustering. 2022; doi:10.1101/2022.01.31.478592

[12] Gutiérrez I, Gómez D, Castro J, Espínola R. A new community detection algorithm based on fuzzy measures. Intelligent and Fuzzy Techniques in Big Data Analytics and Decision Making. 2019;133–40. doi:10.1007/978-3-030-23756-1_18

[13] Stuart T, Butler A, Hoffman P, Hafemeister C, Papalexi E, Mauck WM, et al. Comprehensive integration of single-cell data. Cell. 2019 Jun;177(7). doi:10.1016/j.cell.2019.05.031

[14] Yu L, Cao Y, Yang JY, Yang P. Benchmarking clustering algorithms on estimating the number of cell types from single-cell RNA-sequencing data. Genome Biology. 2022;23(1). doi:10.1186/s13059-022-02622-0

[15] Riva SG, Myers B, Cazzaniga P, Buffa FM, Tangherloni A. Consensus clustering strategy for cell type assignments of scRNA-Seq Data. 2023 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB). 2023; doi:10.1109/cibcb56990.2023.10264908

[16] Oliveira JV, Pedrycz W. Advances in fuzzy clustering and its applications. Chichester: Wiley; 2007.

[17] Gates AJ, Wood IB, Hetrick WP, Ahn Y-Y. Element-centric clustering comparison unifies overlaps and hierarchy. Scientific Reports. 2019;9(1). doi:10.1038/s41598-019-44892-y

[18] Qiu X, Mao Q, Tang Y, Wang L, Chawla R, Pliner HA, et al. Reversed graph embedding resolves complex single-cell trajectories. Nature Methods. 2017 Aug 21;14(10):979–82. doi:10.1038/nmeth.4402

[19] Saelens W, Cannoodt R, Todorov H, Saeys Y. A comparison of single-cell trajectory inference methods. Nature Biotechnology. 2019;37(5):547–54. doi:10.1038/s41587-019-0071-9

[20] Wang Z, Ding H, Zou Q. Identifying cell types to interpret scrna-seq data: How, why and more possibilities. Briefings in Functional Genomics. 2020;19(4):286–91. doi:10.1093/bfgp/elaa003

[21] Wang J, Xia J, Tan D, Lin R, Su Y, Zheng C-H. SCHFC: A hybrid fuzzy clustering method for single-cell RNA-seq data optimized by natural computation. Briefings in Bioinformatics. 2022;23(2). doi:10.1093/bib/bbab588

[22] Haque A, Engel J, Teichmann SA, Lönnberg T. A practical guide to single-cell RNA-sequencing for Biomedical Research and Clinical Applications. Genome Medicine. 2017;9(1). doi:10.1186/s13073-017-0467-4

[23] Zhang Y, Wang D, Peng M, Tang L, Ouyang J, Xiong F, et al. Single-cell RNA sequencing in cancer research. Journal of Experimental & Clinical Cancer Research. 2021;40(1). doi:10.1186/s13046-021-01874-1

[24] Guo X, Chen L. From G1 to M: A comparative study of methods for identifying cell cycle phases. Briefings in Bioinformatics. 2024;25(2). doi:10.1093/bib/bbad517

[25] Satija R, Farrell JA, Gennert D, Schier AF, Regev A. Spatial reconstruction of single-cell gene expression data. Nature Biotechnology. 2015;33(5):495–502. doi:10.1038/nbt.3192

[26] Jiang L, Chen H, Pinello L, Yuan G-C. Giniclust: Detecting rare cell types from single-cell gene expression data with Gini Index. Genome Biology. 2016;17(1). doi:10.1186/s13059-016-1010-4

[27] Finak G, McDavid A, Yajima M, Deng J, Gersuk V, Shalek AK, et al. MAST: A flexible statistical framework for assessing transcriptional changes and characterizing heterogeneity in single-cell RNA sequencing data. Genome Biology. 2015;16(1). doi:10.1186/s13059-015-0844-5

[28] Butler A, Hoffman P, Smibert P, Papalexi E, Satija R. Integrating single-cell transcriptomic data across different conditions, technologies, and species. Nature Biotechnology. 2018;36(5):411–20. doi:10.1038/nbt.4096

[29] Haghverdi L, Lun AT, Morgan MD, Marioni JC. Batch effects in single-cell RNA-sequencing data are corrected by matching mutual nearest neighbors. Nature Biotechnology. 2018;36(5):421–7. doi:10.1038/nbt.4091

[30] Andrews TS, Hemberg M. Identifying cell populations with scRNASeq. Molecular Aspects of Medicine. 2018;59:114–22. doi:10.1016/j.mam.2017.07.002

[31] McInnes L, Healy J, Saul N, Großberger L. UMAP: Uniform manifold approximation and projection. Journal of Open Source Software. 2018;3(29):861. doi:10.21105/joss.00861

[32] Hafemeister C, Satija R. Normalization and variance stabilization of single-cell RNA-seq data using regularized negative binomial regression. Genome Biology. 2019;20(1). doi:10.1186/s13059-019-1874-1

[33] Hartigan JA, Wong MA. Algorithm as 136: A K-means clustering algorithm. Applied Statistics. 1979;28(1):100. doi:10.2307/2346830

[34] Yang L, Liu J, Lu Q, Riggs AD, Wu X. SAIC: An iterative clustering approach for analysis of single cell RNA-Seq Data. BMC Genomics. 2017;18(S6). doi:10.1186/s12864-017-4019-5

[35] Grün D, Lyubimova A, Kester L, Wiebrands K, Basak O, Sasaki N, et al. Single-cell messenger RNA sequencing reveals rare intestinal cell types. Nature. 2015;525(7568):251–5. doi:10.1038/nature14966

[36] Lin P, Troup M, Ho JW. CIDR: Ultrafast and accurate clustering through imputation for single-cell RNA-seq data. Genome Biology. 2017;18(1). doi:10.1186/s13059-017-1188-0

[37] Zurauskienė J, Yau C. PCAREDUCE: Hierarchical clustering of single cell transcriptional profiles. BMC Bioinformatics. 2016;17(1). doi:10.1186/s12859-016-0984-y

[38] Guo M, Wang H, Potter SS, Whitsett JA, Xu Y. Sincera: A pipeline for single-cell RNA-seq profiling analysis. PLOS Computational Biology. 2015;11(11). doi:10.1371/journal.pcbi.1004575

[39] Saelens W, Cannoodt R, Todorov H, Saeys Y. A comparison of single-cell trajectory inference methods. Nature Biotechnology. 2019;37(5):547–54. doi:10.1038/s41587-019-0071-9

[40] Patel AP, Tirosh I, Trombetta JJ, Shalek AK, Gillespie SM, Wakimoto H, et al. Single-cell RNA-seq highlights intratumoral heterogeneity in primary glioblastoma. Science. 2014;344(6190):1396–401. doi:10.1126/science.1254257

[41] Monti S, Tamayo P, Mesirov J, et al. Consensus Clustering: A Resampling-Based Method for Class Discovery and Visualization of Gene Expression Microarray Data. Machine Learning. 2003;5291–118. https://doi.org/10.1023/A:1023949509487

[42] Bouland GA, Mahfouz A, Reinders MJ. Consequences and opportunities arising due to sparser single-cell RNA-seq datasets. Genome Biology. 2023;24(1). doi:10.1186/s13059-023-02933-w

[43] Song D, Li JJ. Pseudotimede: Inference of differential gene expression along cell pseudotime with well-calibrated P-values from single-cell RNA sequencing data. Genome Biology. 2021;22(1). doi:10.1186/s13059-021-02341-y

[44] Marku M, Pancaldi V. From time-series transcriptomics to gene regulatory networks: A review on Inference Methods. PLOS Computational Biology. 2023;19(8). doi:10.1371/journal.pcbi.1011254

[45] Seo CH, Kim J-R, Kim M-S, Cho K-H. Hub genes with positive feedbacks function as master switches in developmental Gene Regulatory Networks. Bioinformatics. 2009;25(15):1898–904. doi:10.1093/bioinformatics/btp316

[46] Ha D, Kim D, Kim I, Oh Y, Kong J, Han SK, et al. Evolutionary rewiring of regulatory networks contributes to phenotypic differences between human and mouse orthologous genes. Nucleic Acids Research. 2022;50(4):1849–63. doi:10.1093/nar/gkac050

[47] van der Wijst MG, de Vries DH, Brugge H, Westra H-J, Franke L. An integrative approach for building personalized gene regulatory networks for precision medicine. Genome Medicine. 2018;10(1). doi:10.1186/s13073-018-0608-4

[48] Townes FW, Engelhardt BE. Nonnegative spatial factorization applied to spatial genomics. Nature Methods. 2022 Dec 31;20(2):229–38. doi:10.1038/s41592-022-01687-w

[49] Lun A [Internet]. Correcting batch effects in single-cell RNA-seq data [cited 2024 Jan 25]. Available from:https://www.bioconductor.org/packages/release/bioc/vignettes/batchelor/inst/doc/correction.html

[50] Tran HT, Ang KS, Chevrier M, Zhang X, Lee NY, Goh M, et al. A benchmark of batch-effect correction methods for single-cell RNA sequencing data. Genome Biology. 2020;21(1). doi:10.1186/s13059-019-1850-9

[51] Korsunsky I, Millard N, Fan J, Slowikowski K, Zhang F, Wei K, et al. Fast, sensitive and accurate integration of single-cell data with Harmony. Nature Methods. 2019;16(12):1289–96. doi:10.1038/s41592-019-0619-0

[52] Welch JD, Kozareva V, Ferreira A, Vanderburg C, Martin C, Macosko EZ. Single-cell multi-omic integration compares and contrasts features of brain cell identity. Cell. 2019;177(7). doi:10.1016/j.cell.2019.05.006

[53] Satija R, Farrell JA, Gennert D, Schier AF, Regev A. Spatial reconstruction of single-cell gene expression data. Nature Biotechnology. 2015;33(5):495–502. doi:10.1038/nbt.3192

[54] Trapnell C, Cacchiarelli D, Grimsby J, Pokharel P, Li S, Morse M, et al. The dynamics and regulators of cell fate decisions are revealed by pseudotemporal ordering of single cells. Nature Biotechnology. 2014;32(4):381–6. doi:10.1038/nbt.2859

[55] R Shiny [cited 2024 Jan 24]. Available from: https://shiny.posit.co/

[56] Aibar S, González-Blas CB, Moerman T, Huynh-Thu VA, Imrichova H, Hulselmans G, et al. Scenic: Single-cell regulatory network inference and clustering. Nature Methods. 2017;14(11):1083–6. doi:10.1038/nmeth.4463

[57] Van de Sande B, Flerin C, Davie K, De Waegeneer M, Hulselmans G, Aibar S, et al. A scalable scenic workflow for single-cell gene regulatory network analysis. Nature Protocols. 2020;15(7):2247–76. doi:10.1038/s41596-020-0336-2

[58] Bravo González-Blas C, De Winter S, Hulselmans G, Hecker N, Matetovici I, Christiaens V, et al. Scenic+: Single-cell multiomic inference of enhancers and Gene Regulatory Networks. Nature Methods. 2023;20(9):1355–67. doi:10.1038/s41592-023-01938-4

[59] Review of single-cell RNA-seq data clustering for cell-type identification and characterization

[60] Su K, Yu T, Wu H. Accurate feature selection improves single-cell RNA-seq cell clustering. Briefings in Bioinformatics. 2021 Feb 22;22(5). doi:10.1093/bib/bbab034

[61] Kiselev VY, Andrews TS, Hemberg M. Challenges in unsupervised clustering of single-cell RNA-seq data. Nature Reviews Genetics. 2019 Jan 7;20(5):273–82. doi:10.1038/s41576-018-0088-9

[62] Gribben C, Galanakis V, Calderwood A, Chazarra Gil A, Kania K, Rouhani F, et al. Acquisition of epithelial plasticity in the human liver during chronic disease progression, under revision in Nature

Appendix 1

Common approaches for each pre-processing step.

Pre-processing step	Purpose before clustering	Common approach in literature
Quality control and filtering	Eliminate low-quality cells with skewed transcriptomic profiles [53].	Filtering cells that have gene counts over 2500 or less than 200 [25, 26].
Normalisation	Reduces biases from amplification, sample differences, and other technical factors [27-29].	1) Log-normalisation: transform read counts using a logarithm with an offset (e.g. by adding one). 2) SCTransform: the Pearson residuals from regularised negative binomial regression are used as a covariate in a generalised linear model to remove the influence of technical characteristics while maintaining biological heterogeneity [32].
Feature selection	Enhances the signal-to-noise ratios, which subsequently improves the cell clustering results [60].	1) Highly variable genes (HVGs): selecting genes that show significant variation in their expression levels across different cells. Most often choosing top 1000 or 2000 genes [13]. 2) Most abundant genes (MAGs): selecting genes based on their overall expression levels, prioritizing genes that are expressed at higher levels across the dataset. Most often choosing top 1000 or 2000 genes [18].
Dimensionality reduction	scRNA-seq is inherently high dimensional [30]. Dimensionality reduction embeds each cell's high-dimensional expression profile into a low-dimensional representation, enabling visualisation and improving clustering outcomes.	1) Linear: PCA - projects a set of possibly correlated variables into a set of linearly orthogonal variables (principal components). 2) Non-linear: UMAP [31], a non-linear technique that preserves the local and global structure of the data. 3) Non-Negative matrix factorisation [48]
Batch Effect	Counter technical variations and ensure analyses reflect biological rather than technical differences.	Harmony [51], LIGER [52], and fastMNN [49].

Appendix 2

Biases in common evaluation metrics as explained in [17].

Bias	Description
Randomized membership	Comparing a partition with its duplicate after random label swaps. As swap ratio increases, similarity should decrease but not reach zero due to at least one similarity: the number and size of clusters.
Skewed cluster sizes	When comparing two partitions, their similarity is non-zero if they share identical numbers and sizes of clusters, indicating a base level of structural similarity. However, changing the clustering sizes (their skewness) should decrease he similarity between the partitions.
Number of clusters	When comparing two partitions, increasing the number of clusters in the second partition while keeping the first constant, should decrease the similarity between the partitions.
Problem of matching	The problem has three partitions: the first is fixed, the second is the result of moving x points from one cluster to another, and the third is obtained by moving the same number of points from one cluster and distributing them across multiple other clusters. The matching problem is to consider the pairs (first, third) and (first, second) as equally similar.

Appendix B

Response to feedback

I express my gratitude to the markers for their constructive feedback. Below I am detailing how I addressed/responded to their suggestions:

B.1 Marker 1:

1.1: "However the style is very terse. I would encourage a little more space given to signposting – tell the reader where you're going, and remind them along the way as you hit key markers." & "Consider being a little less brief when writing, taking your reader along a journey with you, rather than just describing key points on the way."

I agree that the proposal was quite terse. In the dissertation, I aimed to more eloquently explain concepts, frequently signpost the direction of the discussion, underline conclusions and importance of steps, and provide brief overviews before each section to guide the reader.

1.2: "Ideally the reader sees the research aims as logical steps that follow on from specific gaps identified in the introduction."

This was an excellent suggestion. In the dissertation, I tried to align the research aims more closely with the gaps underlined in the introduction, ensuring a logical flow throughout.

1.3: "While a source of data is briefly described, nothing is said of what the scientific goals are with regards these data. Is it the definition of novel cell types? Are there cells from healthy donors for comparison?"

In the dissertation, I present more details about the datasets used (including number of cells/-genes, cell types, disease stage). I also explicitly state that the scientific (methodology-focused) goal was the development of a package that can be used with any single-cell data. I then tested

the validity of underlying hypotheses and robustness of the framework using the *End-stage* subset which the authors of the Gribben et. al [1] paper extensively discussed.

Moreover, to underline its generalisability, I demonstrated the framework's applicability to other single-cell modalities using a publicly available scATAC-seq dataset as example.

1.4: "I encourage the student to step back from their project, and ask themselves whether the motivation behind the research questions is well defined, whether the planned approach is the best way to address the questions (it may well be! I'm just suggesting to take space to consider this), and what the scientific outcome will be from the project. Is it just the algorithm with the test data only there to show the method runs, or is there any particular questions that you hope to answer with the data and the new method?"

I reflected deeply on this.

The motivation for the development of *flufftail* was and is to create a streamlined methodological framework and R package that, using fuzzy clustering, can provide a deeper understanding of (biological) mechanistic interactions between cells coupled with signals driving and controlling the plasticity (concepts like DE genes describing variability and phenotypes, major regulatory hubs, dynamics of the hubs at different stages of the process can now be linked in a semantically coherent set of steps). The scientific outcome is a framework that can be applied to any single-cell dataset (i.e., agnostic to the type of measurement/assay). The project is not about conducting an analysis on a single dataset with the aim of producing biological hypotheses.

I believe that the dissertation underlines this motivation more clearly now in the introduction and aims. Furthermore, the methodology sections 3.3 3.4, and 3.5, explain in detail how I developed the approach. The Flufftail framework section (4.2) in the results provides a detailed step-by-step overview of this process. Finally, this is put to the test on a different dataset, in the discussion chapter (section 5.1.1) by explaining in detail the application of the package to the Gribben et al. [1] dataset.

1.5 "The mathematics need careful checking that terms are correctly defined"

I tried to be more precise and detailed with the mathematical formulas given (Sections 3.2.3.3, 3.4, and 3.5). I also provided detailed descriptions of the methods, including pseudocode where appropriate, to assist with the understanding and reproducibility of the developed approaches.

B.2 Marker 2:

2.1 "It was not clear to me from the introduction whether the idea would be to directly incorporate the uncertainties in the cluster assignments into subsequent

analyses, or if the idea would be to "screen out" cells whose assignments were excessively ambiguous (e.g. by thresholding an acceptable "level of ambiguity" for each cell)."

I agree that the main idea could have been more clearly stated in the introduction.

In the introduction, I strived to increase the level of detail provided e.g. *"Identifying and further characterising these cells (i.e., getting their membership degrees, visualising their location, and co-clustering behaviour) has the potential to uncover valuable insights into cellular plasticity, developmental processes, and disease mechanisms."* I also explicitly stated my hypothesis that identifying fuzzy genes, with high connectivity in the GRN, could be biologically interesting (e.g. from a drug development of personalised medicine perspective), as targeting them may cause a ripple effect in the GRN and perturb pathways of interest. I then stated that exploring the GRN dynamics of those hub genes across the pseudotime transitions could help better understand the dynamics of GRNs and assist with the identification of potential therapeutic targets.

This approach facilitates a clearer overview of the bigger picture and main questions, also assisting the reader in seeing exactly where the project is headed. This information is once again confirmed by the aim and objectives (Chapter aims). This response is related to my responses to Q1.2 and Q1.4 mentioned by the first marker.

2.2 "The objectives could have been more precise, e.g. for objective 4: to what will the R shiny app provide an interface, and what (briefly) will be the functionality provided by this app?"

Thank you for this suggestion. I addressed this comment by better separating the aims and stating whether each objective is related to methodology, implementation (coding), or both. I also increased the precision of supporting information while briefly capturing the essence of each objective.

2.3 "From what is written, it is not clear to me how the membership degree matrix will be calculated, due to the "label switching" problem (i.e., each run of the clustering algorithm may label the clusters differently)."

I appreciate this comment. Indeed, I did not address the label-switching problem when describing the proposed methodology in the proposal. I have since developed a label reconciliation process that is being been used for consolidating the partitions (grouped by algorithm, resolution, and k to ensure that they are comparable – same resolution can return partitions with different numbers of clusters) returned by the stochastic repeated clustering process (Section 3.3.1).

2.4 "I propose using entropy . . . on each cell's consensus matrix entry". Is "consen-

sus matrix" truly intended here, or should this say "membership degree matrix"? The latter would make more sense to me, i.e., we calculate the entropy associated with each cell by calculating p log p across the rows of the membership degree matrix. In any event, this ambiguity would be resolved by clarifying Equation 1, in particular: in the current context, what is i indexing, what does p_i represent, and what is X?

Thank you for the suggestion about applying the entropy to the membership degree. I also implemented this (Section 3.4.3). Moreover, I explained in detail (using mathematical arguments) the application of Shannon's entropy on both the consensus matrix and membership degree matrix. I conducted empirical experiments to demonstrate the effectiveness of these approaches which are available in Appendix D.

2.5 "Similar to Equation 1, Equation 2 could do with more explanation."

In the dissertation, I explain in more mathematical detail how ECS and ECC are calculated (Section 3.2.3.3). I also reference the ClustAssess paper [9] and stability assessment framework, which use these metrics to assess the stability of clusterings.

2.6 "attempting to define the relationship between ECC and entropy, the motivation for doing this is currently unclear"

I hypothesised a link between ECC and entropy based on some initial results obtained when plotting the entropy on consensus vs. ECC (which on small, homogenous datasets showed an almost linear relationship). After the proposal was submitted, I applied the same experiment on a larger scale and found that it would not be possible to mathematically derive an equation to go from one to the other. This is also further discussed in Appendix C.

Following discussions with my supervisor, I have since decided to focus more on the other objectives of the project, as after reflecting, I agree that this may dilute the main take-home message of the project.

2.7 "It would be great to adopt consistent terminology regarding cells that are ambiguously classified. Currently, these are referred to both as "unstable" and "fuzzy".

I agree with the suggestion. I am now using only the word fuzzy.

2.9 "My understanding is that the output of this project will enable researchers to specify a threshold on this "(in)stability" to determine whether a given cell should be included or excluded from downstream analysis, and that they will be able to vary this threshold to assess the impact on subsequent analyses/results. Is this correct? If so, I think that this could be more clearly stated"

Indeed, the original idea revolved around the exclusion of fuzzy cells.

It has since evolved towards the development of a methodological framework that can be used to harness the information provided by fuzzy clustering, use it to identify major regulatory hubs (fuzzy gene module clustering) in transitioning areas, and then characterise the GRN dynamics of these key genes across the transitions. This angle is more clearly stated now and repeatedly illustrated using several examples.

2.10 "The methods section needs tightening, particularly (but not limited to) the equations."

I acted on this. Please see my response to 1.4.

Appendix C

ECC and Entropy link - plot with dispersion

The plot (Figure C.0.1) displays a negative correlation between entropy on the consensus (x-axis) and the ECC score (y-axis) when applying Louvain across 30 iterations using resolution 0.9 and k=17 on the *Immune* dataset (described in Section 4.2.1). Furthermore, it illustrates a considerable dispersion in the data points, which indicates variability in the link between the two.

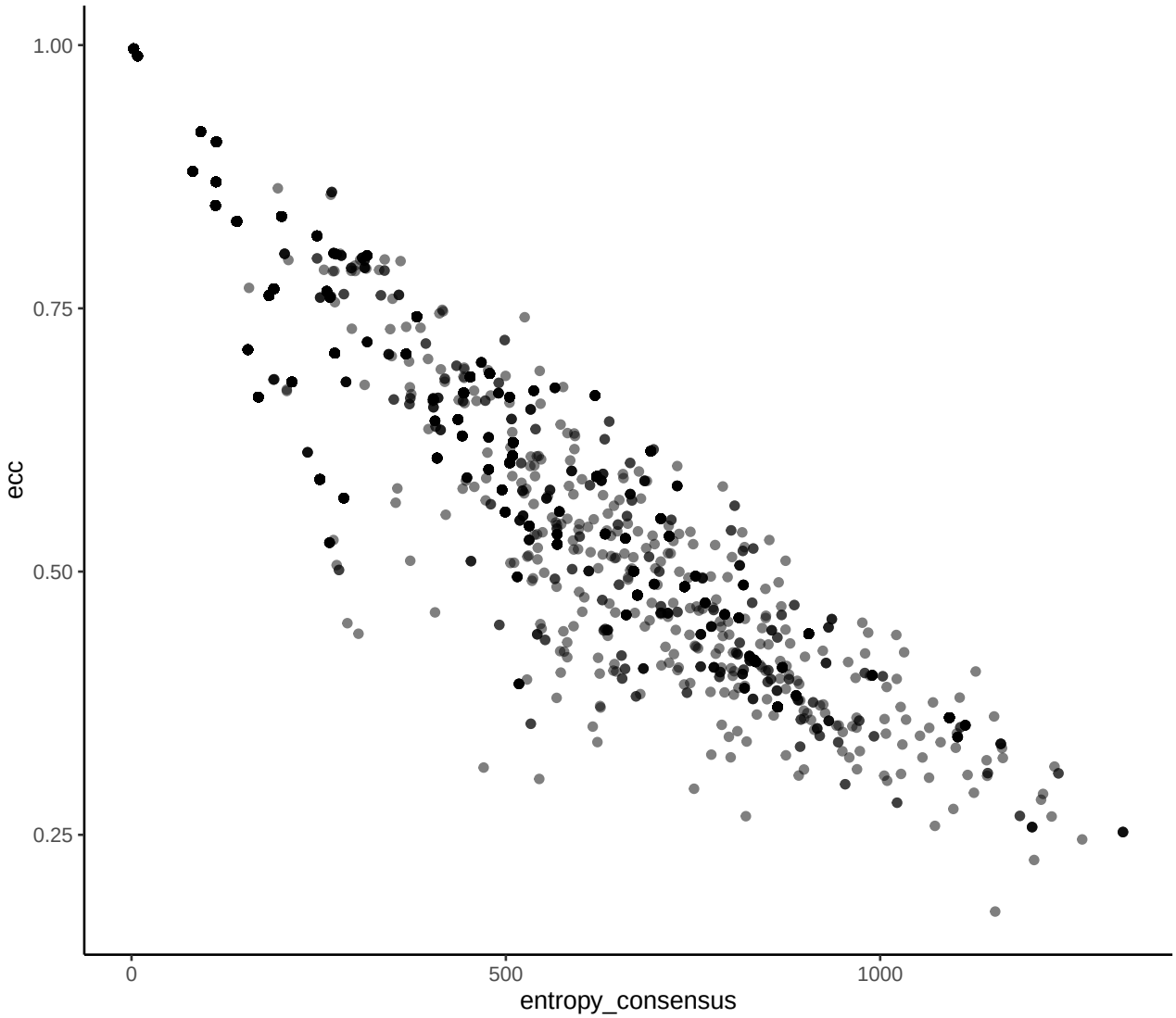


Figure C.0.1: Variation of entropy values (x-axis) vs ECC (y-axis) on a dataset comprising 9,239 cells partitioned on 17 clusters. The increased number of cells and the heterogeneity of the input leads to higher dispersion and underlines that a direct mathematical conversion between the entropy and the ECC might not be straightforward.

Appendix D

Entropy empirical experiments

This section provides additional details on experiments conducted to empirically examine the relationship between entropy values obtained from the consensus matrix and the degree matrix approaches.

The consensus and membership degree matrices of the setting at resolution=0.2, with 5 clusters (5.1.1) of the *End-stage* samples, were extracted. I used the *End-stage* dataset because it contains a much larger number of samples, thus being more informative in this examination.

D.1 Consensus matrix

To empirically examine the relationship between the distribution of co-association probability values and entropy, the entropy for each row (cell) of the consensus matrix was first calculated. The density plots of the co-association probability distributions for the top five cells with the highest unique entropy, the median unique entropy, and the lowest unique entropy were then visualised.

From D.1.1, it is observed that cells with the lowest entropy exhibit high frequency (peaks) of co-clustering probabilities at 0 and 1. This indicates that these cells consistently co-cluster with the same set of cells across multiple iterations, showing high stability. Conversely, cells with the highest entropy display peaks primarily between 0.25 and 0.5, suggesting inconsistent co-clustering behaviour with many other cells. As expected, the cells with median entropy values exhibit intermediate behaviour, representing a middle ground between high and low-entropy cells.

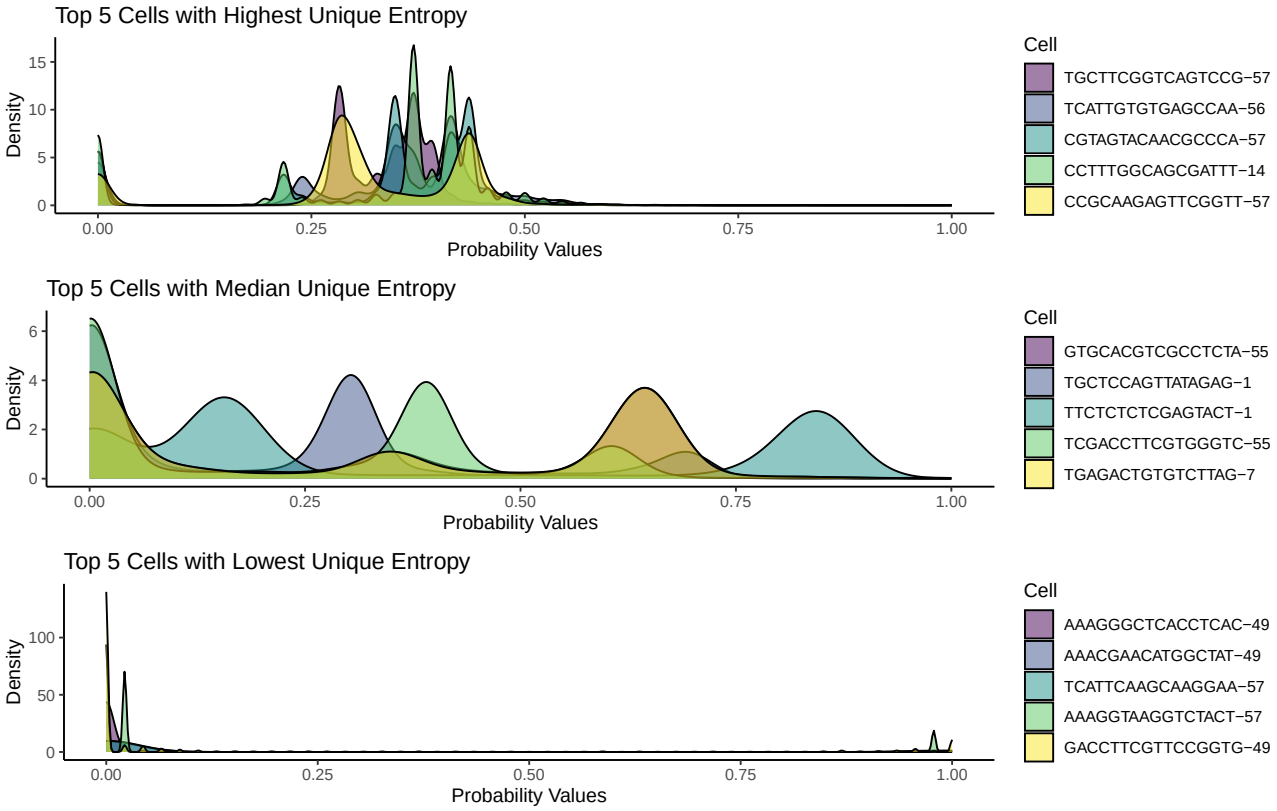


Figure D.1.1: Density plot of distribution of probability in the consensus matrix for the top 5 cells with the highest unique entropy, median unique entropy, and lowest unique entropy.

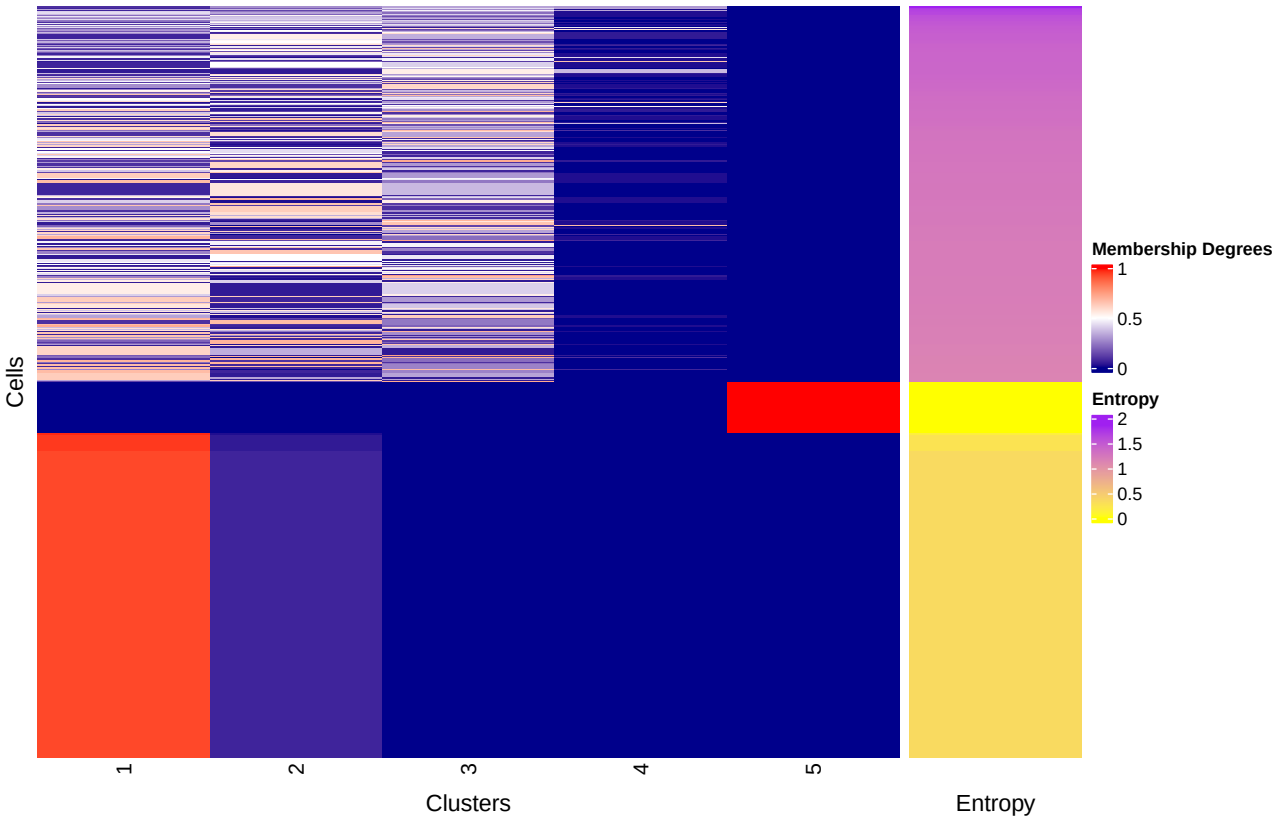


Figure D.2.1: Membership degree of top 1,000 cells with highest and lowest entropy, top and bottom respectively.

D.2 Membership Degree Matrix

The entropy for each row of the membership degree matrix was first calculated and then sorted in descending order (highest entropy). The top 1,000 cells with the highest and lowest entropy were identified, and their membership degrees were visualised as a heat map (D.2.1). As evident from the bottom part of the plot, cells with low entropy have membership degrees that exhibit very high affiliation (close to 1) with a single cluster. Cells with high entropy (top) have membership degrees spread between 2 or 3 clusters, signifying considerable instability in their association with a particular cluster.

Appendix E

Moran's I effect on number of DE genes

Table E shows the effect of applying a Moran's I greater than threshold on the results returned by the `graph_test` function from Monocle3.

Morans-I threshold	**Number of DE genes**
0.01	1,248
0.02	1,150
0.03	909
0.04	741
0.05	602

Table E.0.1: Relation between Moran's I threshold and number of DE genes returned by Monocle3 `graph_test`