# Interactive Analysis of Single-Cell RNA-Sequencing Data

Yue Zhang

A dissertation submitted in partial fulfillment of the requirements
for the degree

Doctor of Philosophy

University of Washington

2022

Reading Committee:

Georg Seelig, Chair

Sreeram Kannan

Su-In Lee

Hannaneh Hajishirzi

Program Authorized to Offer Degree:

Computer Science and Engineering

University of Washington

**Abstract**

Interactive Analysis of Single-Cell RNA-Sequencing Data

Yue Zhang

Chair of the Supervisory Committee:

Georg Seelig

Analysis of single cell RNA sequencing (scRNA-seq) datasets is a complex and time-consuming process, requiring both biological knowledge and technical skill. With the rapid growth in scRNA-seq datasets and the increasing number of biological applications, there is a need to simplify and systematize this process. In order to systematize and simplify this process, we introduce three tools to aid : UNCURL, CellMeSH, and UNCURL-App. UNCURL is a preprocessing tool that assists in dimensionality reduction and clustering of scRNA-seq data. CellMeSH is a database and associated query method that allows for the easy identification of cell types present in single cell data. UNCURL-App is an online GUI-based interactive web app which integrates multiple stages of the data analysis pipeline into a single interface, including dimensionality reduction, clustering, differential expression, and cell type identification. Together, these tools encompass all stages of the scRNA-seq cell type annotation pipeline into a single user-friendly interface.

# Contents

# Chapter 1

# Introduction

Single cell RNA sequencing (scRNA-seq) has become an essential and ubiquitous tool for exploring the diversity of cell types in multicellular organisms. Progress in experimental technology development has driven rapid growth in the number of scRNA-seq datasets (Chen et al., 2018; Svensson et al., 2019), with a search for "single-cell RNA-seq" on NCBI GEO returning tens of thousands of results. Over little more than a decade, scRNA-seq experiments progressed from first proof-of-principle demonstrations using a handful of cells (Tang et al., 2009) to the construction of "cell atlases" that enumerate all of the cell types present in an organ or organism (Han et al., 2018; Cao et al., 2017; Regev et al., 2017; Consortium, 2018; Tasic et al., 2016; Rosenberg et al., 2018).

However, while experimental approaches have become higher throughput and more widely available, it remains challenging to map experimentally determined single cell transcriptomes to biologically meaningful cell types. Given the very large throughput in cell number and the high complexity of many of the systems under investigation, reliable data analysis has become the main bottleneck of the scRNA-seq workflow.

The overall goal of this project is to improve the process scRNA-seq analysis by better integrating prior knowledge into the analysis process, and by making the analysis workflow more accessible by reducing the requirements for coding and command-line usage. To do so, this paper will describe a number of novel methods for analyzing single-cell RNA-seq data in a data-driven way. First, we will discuss UNCURL, a method for preprocessing scRNA-seq

data based on a known distribution. Then, we will discuss CellMeSH, a method for identifying cell types from single-cell data using information from the scientific literature. Finally, we will discuss UNCURL-App, an interactive web-based tool that encompasses the whole pipeline in a user-friendly package.

In this chapter, we will begin with a brief history of experimental methods for scRNA-seq, followed by an overview of the various stages of the computational pipeline involved in scRNA-seq data analysis.

## 1.1 Experimental Methods For Single-Cell RNA-Sequencing

Single cell RNA sequencing (scRNA-seq) is a technology for determining the gene expression profiles of individual cells. Many different methods for scRNA-seq have been developed in the past decade, but the inputs and outputs remain similar. All of these methods take a sample of single cells, and produce, via sequencing of the cDNA, a set of mRNA reads that are labeled with their originating cells. These reads are then aligned to the genome in order to produce a gene-cell matrix, showing the expression levels of every gene in every cell in the sample. This matrix is then used for downstream computational analysis.

The first experimental method involving whole-transcriptome mRNA sequencing of a single cell was in Tang et al. (2009). In this paper, the transcriptomes of several mouse blastomeres and oocytes were sequenced. This was followed by Islam et al. (2011), where 96 mouse embryonic stem and fibroblast cells were sequenced using their "single-cell tagged reverse transcription" method. Subsequent methods have greatly increased the number of cells that can be analyzed into the tens or hundreds of thousands.

Many high-throughput single cell sequencing methods are based on microfluidic technologies, where individual cells are isolated into tiny volumes of liquid in order to perform the various reactions necessary for sequencing. These include the Drop-seq method of Macosko et al. (2015), which used nanoliter droplets to sequence over 44k cells, and the droplet-based method developed by 10X Genomics (Zheng et al., 2017), which was used to sequence over 68k cells. There are also methods that do not rely on microfluidics: Microwell-seq (Han et al., 2018) uses a single plate with microwells that isolate up to $10^5$ cells, and SPLiT-seq (Rosenberg et al., 2018) uses

combinatorial barcoding to distinguish cells, also allowing for sequencing of tens of thousands of cells.

One major area of difference between the various scRNA-seq methods is how the reads are generated. Some methods, such as SMART-seq (Ramsköld et al., 2012), allow for sequencing along the full length of the transcripts, and fragment the transcripts for sequencing. Other methods (Macosko et al., 2015; Zheng et al., 2017; Rosenberg et al., 2018) only sequence along the 3' end of the mRNA transcript, and attach a UMI, or unique molecular identifier, to the end of each mRNA transcript, prior to PCR amplification. The UMI allows for easier quantification of mRNA levels as the UMI counts should correspond directly to the mRNA counts (Islam et al., 2014; Smith et al., 2017).

With all of the sequencing methodologies, technical noise and variation continue to be prevalent in scRNA-seq data. The set of transcripts that are sequenced will always be a fraction of the mRNA molecules present in each cell; many molecules will not be captured by the sequencing process - "dropouts" is the term for this phenomenon (Chen et al., 2019). This often leads to many or most genes not being detected in any given cell. Many computational methods have been developed to address this issue, including UNCURL, as well as many of the tools described in the following section.

There have been numerous biological applications of scRNA-seq. Many have focused on the identification of cell subpopulations in various tissues, whether to identify novel cell types or to build a "cell atlas" of all cell types present in a tissue (Chen et al., 2018). Such experiments have been done in the brain (Zeisel et al., 2015; Tasic et al., 2016; Rosenberg et al., 2018), blood cells (Villani et al., 2017), and other tissue types. In a similar vein, scRNA-seq has been used to create cell type atlases of various tissues (Tasic et al., 2016) or even at the level of the whole organism. Cao et al. (2017) describes a cell atlas of the whole roundworm *C. elegans*, and Plass et al. (2018) describes a cell atlas of the planarian *Schmidtea mediterranea*. There have been a number of cell type atlases of the mouse (*Mus musculus*) (Consortium, 2018; Han et al., 2018). The Human Cell Atlas is a project to identify all cell types present in humans (Regev et al., 2017).

Another common application for scRNA-seq is in lineage analysis. The goal here is usually

to place the cells along some trajectory of development or differentiation, in order to study the dynamics of gene expression during these processes. This often depends on computationally aligning sequenced cells onto a "pseudotime" trajectory, which is an ordering of cells inferred from the data (Trapnell et al., 2014). Lineage analysis has been applied to domains such as B-cells (Bendall et al., 2014), neuronal development (Hanchate et al., 2015; Rosenberg et al., 2018), and even the development trajectories of tissues across whole multicellular organisms (Plass et al., 2018; Cao et al., 2019).

## 1.2 Computational Methods For Analyzing scRNA-Seq Data

A wide range of computational tools have been developed to guide and assist each step in the analysis workflow from preprocessing (Pierson and Yau, 2015; Wang et al., 2017; Mukherjee et al., 2018; Dijk et al., 2018), to clustering (Kiselev et al., 2017; Lin et al., 2017; Zhang et al., 2018a; Sun et al., 2019; Diaz-Mejia et al., 2019), data integration through batch effect correction (Kiselev et al., 2018; Haghverdi et al., 2018) and cell type annotation (Pliner et al., 2019; Srivastava et al., 2018; Alavi et al., 2018).


Figure 1.1: This is an illustration of one of the common scRNA-seq data analysis pipelines, leading to cell type annotation.

Most applications of scRNA-seq rely on a few common computational steps, as described in Andrews et al. (2021) and Chen et al. (2019) (Figure 1.1). The first step is usually a sequence alignment that maps short reads onto the genome, using tools such as STAR (Dobin et al., 2012). The sequence alignments are then used to produce a *gene expression matrix*, showing the expression levels of every gene in each cell. In UMI-based sequencing methods, the collapsed UMI counts can be used directly, while for full-length sequencing methods, there has to be an additional quantification step to transform the raw read counts into actual gene expression values, using tools such as RSEM or CuffLinks (Li and Dewey, 2011; Trapnell et al., 2010; Chen et al., 2019).

The gene expression matrix is the input to almost all downstream analysis pipelines. After this matrix is generated, the next step is usually preprocessing to deal with data quality issues. Cells might be filtered to remove low-quality cells, including cells with too-low read counts or cells that have too-high proportions of mitochondrial genes. Sometimes, imputation is done to deal with dropouts, which involves computational methods that fill in the zero values of the gene expression matrix (often, between 50 to 90 percent of the matrix values are zero) (Gong et al., 2018). Data normalization is done to deal with variation in read counts per cell - this often involves dividing all gene expression values by a cell-specific factor, so that all cells will end up with the same total gene expression level. If the data is comprised of multiple technical replicates (or "batches"), batch effect correction is often done to eliminate the confounding technical variation so that only biological variation remains.

The next steps are usually dimensionality reduction and clustering. Since the gene expression matrix is of high dimensionality, often with tens of thousands of cells by tens of thousands of genes, dimensionality reduction is necessary in order to create a human-interpretable view of the data. Clustering is done in order to detect potential distinct cell types or states. Following clustering, differential gene expression is calculated to identify the highly expressed genes in each cluster. These highly expressed genes are then used to search through the literature in order to identify the cell types.

Many preprocessing tools have been developed to deal with scRNA-seq data, often performing some combination of imputation, normalization, probabilistic modeling, and dimensionality

reduction. Some of these methods, such as ZIFA (Pierson and Yau, 2015) and ZINB-WAVE (Risso et al., 2018), use a probabilistic distribution to model scRNA-seq data. ZIFA uses a zero-inflated normal distribution, while ZINB-WAVE uses a zero-inflated negative binomial distribution. Other preprocessing methods, such as MAGIC (Dijk et al., 2017) and SIMLR (Wang et al., 2017), use a nearest-neighbor graph method to try to infer the relationships between cells and perform imputation, without any specific assumed data distribution.

Most existing methods for data preprocessing rely almost exclusively on unsupervised learning and do not incorporate useful and commonly available prior information, such as bulk gene expression data or cell type specific marker genes, to guide the analysis process.

Following data preprocessing, dimensionality reduction is commonly done using common methods such as PCA, tSNE (Maaten and Hinton, 2008), or umap (McInnes and Healy, 2018). These are general-purpose methods not specifically developed for scrNA-seq data. Clustering can be done on the processed data using general methods like k-means or the Louvain or Leiden methods for community detection in graphs (Blondel et al., 2008; Traag et al., 2018), or using any number of methods developed specifically for scRNA-seq (Kiselev et al., 2017; Lin et al., 2017; Zhang et al., 2018a). Some of these methods are compared in the following chapter.

There are many methods for assessing differential expression and identifying marker genes, including methods developed for microarray or bulk RNA-seq data Robinson et al. (2010); Anders and Huber (2010), methods developed specifically for single-cell data Finak et al. (2015); Kharchenko et al. (2014), and more general, long-standing statistical tests such as the t-test or the Wilcoxon signed-rank test. A large number of these methods are compared in Soneson and Robinson (2018), which shows that the general-purpose methods such as the t-test perform as well or better than specialized methods on many metrics, and are much more computationally efficient.

There are also a number of integrated analysis frameworks that combine several of these tasks into one package: two popular such packages are scanpy, written in Python (Wolf et al., 2018), and Seurat, written in R (Satija et al., 2015). These tools integrate the pre-processing, clustering, and visualization steps, often combining some of the packages listed above. However, these tools are typically restricted to command-line usage and require programming knowledge,

hindering the accessibility of scRNA-seq analysis.

Some web-based platform for single-cell RNA-seq analysis include scQuery (Alavi et al., 2018), Granatum (Zhu et al., 2017), and Cumulus (Li et al., 2020). ScQuery allows for users to upload scRNA-seq datasets (that have been pre-processed in a given format), and can identify cell types using a pre-trained neural network. It is somewhat limited in its interactivity; it does not allow cluster assignments or cell type labels to be changed by the user. Granatum and Cumulus also allow users to upload scRNA-seq datasets to their platform and perform analyses on them, but they do not have the built-in ability to identify known cell types.

A number of tools have been developed for cell type annotation of scRNA-seq data. Among these methods, many are based on directly mapping single cells cells from the gene-cell matrix onto an annotated single-cell or bulk RNA-seq dataset by means of some kind of similarity search or machine-learned model. These include (Kiselev et al., 2018; Srivastava et al., 2018; Pliner et al., 2019; Hou et al., 2019). Some of these tools contain their own curated cell type databases, while others require the end user to provide a candidate database. CellAtlasSearch (Srivastava et al., 2018) is an online tool that uses locality sensitive hashing to query for cell types, and contains its own curated database of reference cell types. scMatch (Hou et al., 2019) and scmap (Kiselev et al., 2018) can work locally or online, use Pearson and Spearman correlation, and contain links to downloadable reference cell types.

There are also some existing methods for cell type annotation that use a list of differentially expressed genes to query cell type databases; these methods include (Chen et al., 2013; Stachelscheid et al., 2013; Zhang et al., 2018b; Hänzelmann et al., 2013; Franzén et al., 2019). There are at least two databases of cell types and gene markers designed for use with scRNA-seq: CellMarker (Zhang et al., 2018b) and PanglaoDB (Franzén et al., 2019). CellMarker was manually curated by searching through the literature. PanglaoDB is based on existing scRNA-seq experiments that are annotated with cell types.

## 1.3 Goals and Proposed Work

In Lähnemann et al. (2020), "Mapping single cells to a reference atlas" is described as one of the top challenges in single cell data science. Creating a better pipeline for cell type identification

is the primary purpose of our work. Currently, the process of assigning sequenced cells to cell types is a multi-step process that requires the user to make decisions based on their judgment, because the ground truth about abundance and identity of cell types in an experiment of interest is usually unknown. The sequence of tasks described in the previous section requires users who have both technical proficiency and knowledge of the underlying biology, and has many ad-hoc steps.

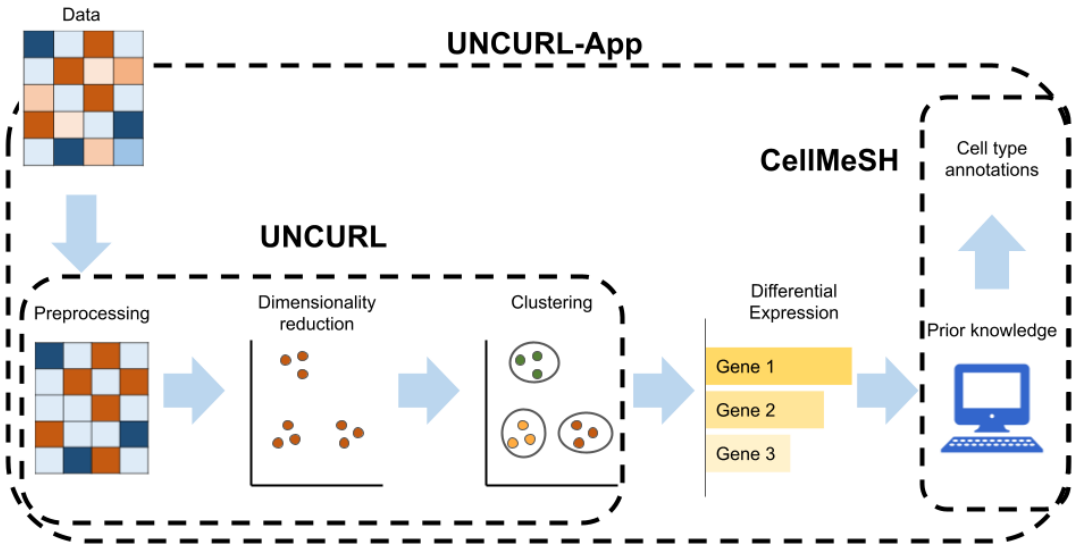



Figure 1.2: This shows an overview of the scRNA-seq data analysis pipeline, overlaid with the methods proposed herein, and how they fit in the pipeline. UNCURL is a method for preprocessing scRNA-seq data that can also be used for clustering and dimensionality reduction. CellMeSH is a database and method for identifying cell types. UNCURL-App is an interactive web-based tool that encapsulates the entire process.

Our goal here is to improve the process scRNA-seq analysis by better integrating prior knowledge into the analysis process, and by making the analysis workflow more accessible by reducing the requirements for coding and command-line usage. To do so, this paper will describe set of novel methods that comprise a pipeline for analyzing single-cell RNA-seq data. First, we will discuss UNCURL (Mukherjee et al., 2018), a method for preprocessing scRNA-seq data based on a known distribution. Next, we will discuss CellMeSH (Mao et al., 2022), a method for identifying cell types from single-cell data using information from the scientific literature. Finally, we will discuss UNCURL-App (Zhang et al., 2020), an interactive web-based tool that

encompasses the whole pipeline in a user-friendly package. Together, these tools encompass the scRNA-seq analysis pipeline, from data preprocessing and visualization to cell type annotation.

# Chapter 2

# UNCURL

## 2.1 Introduction

UNCURL (Mukherjee et al., 2018) is a preprocessing framework for scRNA-seq data that attempts to estimate the true transcriptomic state of the cells prior to the sampling effect of RNA-seq. The preprocessed data from UNCURL can then be directly used as input by most major unsupervised learning algorithms commonly used in the context of scRNA-seq data.

The two issues that UNCURL was designed to address are: 1) the specific distributional nature of scRNA-seq data caused by the sampling effect of sequencing, and 2) the inability of other methods to use existing prior knowledge directly in the analysis.

The main technical contribution of UNCURL is a generalized non-negative matrix factorization (NMF) that explicitly accounts for the most likely sampling distribution of the dataset. Furthermore, UNCURL incorporates an accelerated parallelizable optimization method tailored for sparse input data, so as to handle datasets with millions of cells efficiently.

UNCURL exploits the low-dimensional nature of the true biological state matrix; it assumes that each cell is in a convex combination of a few archetypal cell-states. Under this assumption, the true state matrix can be expressed as a product of an archetypal main state matrix, M, comprising of gene-expression in the archetypal states, and a matrix of mixing coefficients, W, a cluster-by-cell matrix for which each column sums to 1.

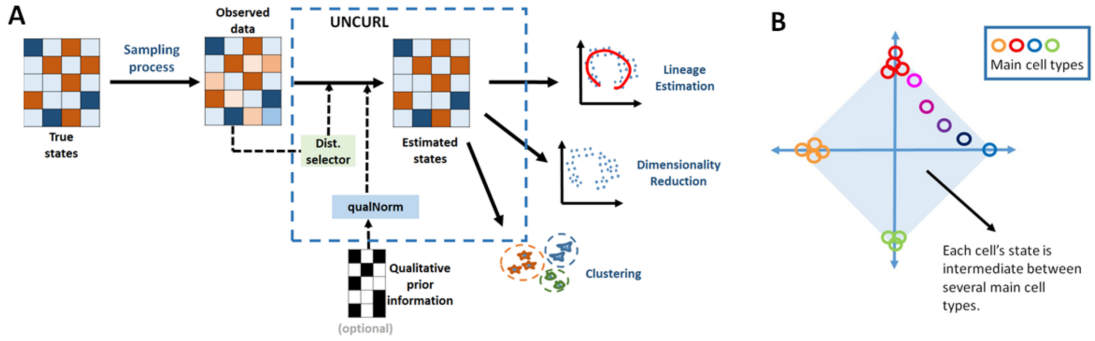



Figure 2.1: A) The primary input for UNCURL is the highly sampled single cell sequenced data and optionally any prior information that is known about the specific dataset. UNCURL then converts the observed sampled data to an estimated version of the true data using a novel sampling model aware matrix factorization. This can then be used in downstream unsupervised learning tasks. B) The convex mixture of cell states assume that all cell states lie in the convex hull spanned by a few extreme cell types.

## 2.2 Methods

We assume that we are provided with a data matrix $X \in \mathbb{R}^{n \times d}$, where $n$ is the number of genes and $d$ is the number of cells, and $k$, the number of cell types. Let $X_{g,c}$ denote the count measured for gene $g$ in cell $c$, and let $X^{\text{true}}_{g,c}$ be the relative abundance of gene $g$ in cell $c$. We assume that the true matrix has a non-negative decomposition into two factors, i.e., $X^{\text{true}} = M \times W$. Here, $M \in \mathbb{R}^{n \times k}$ is the matrix of cell type means of the $k$-archetypal states with $M_{g,j}$ denoting the expression of gene $g$ in archetype $k$. $W \in \mathbb{R}^{k,d}$ is the mixture parameter matrix which stores each cell as a convex combination of the archetypes, i.e., $W_{j,c}$ is the contribution of archetype $j$ to cell $c$. Thus $W$ satisfies $\mathbf{1}^k W_{j,c} = 1$ and $W_{j,c} \geq 0$.

Since we do not observe directly the relative abundance but only a sampled version, we assume that there is a channel that connects the true abundance to the observed abundance, $\mathbb{P}(x|x^{\text{true}}, \theta)$, i.e, given a value of $x^{\text{true}}$ there is a certain distribution on $x$ with some parameters $\theta$. Note that we have not used a subscript for $g$ and $c$ to emphasize that the same distribution is used for all genes. We give three examples here, but our framework works with general distributions: (1) $\mathbb{P}(x|x^{\text{true}})$ is a Gaussian distribution with mean $x^{\text{true}}$ and a fixed variance 1 (say). (2) $\mathbb{P}(x|x^{\text{true}})$ is a Poisson distribution with mean $x^{\text{true}}$. (3) $\mathbb{P}(x|x^{\text{true}})$ is such that $\log(x)$ is a Gaussian with mean $\log(x^{\text{true}})$ and variance 1. Our goal now is to maximize the

log-likelihood of the observed data matrix $X$, by finding the optimal $M$ and $W$. Note that $\mathbb{P}(X|M,W,\Theta) := \prod_{g,c} \mathbb{P}(X_{g,c}|X^{\text{true}}_{g,c})$.

This problem is non-convex but the sub-problems of estimating either $M$ or $W$ with the other matrix fixed are convex problems for many common sampling distributions, including the ones mentioned above. We thus utilize an alternating maximization algorithm to estimate these model parameters as follows:

$W = \text{argmax}_W \log(\mathbb{P}(X|M,W,\Theta))$ subject to $W_{j,c} \geq 0$, and
$M = \text{argmax}_M \log(\mathbb{P}(X|M,W,\Theta))$ subject to $M_{g,j} \geq 0$.

We repeat these two steps iteratively until convergence or until a maximum number of iterations has been reached. Once converged, we normalize the columns of $W$ to sum to 1 to ensure the condition $\mathbf{1}^k W_{j,c} = 1$ is satisfied. We note that each of the steps is convex for many distributions and can be solved by gradient descent. In the case of the Poisson model, we use a custom alternating minimization approach using the NoLips algorithm (Bauschke et al., 2016) to optimize the Poisson Log-Likelihood with additional modifications to allow for faster computation on sparse matrices and the ability to parallelize the computation.

Pseudocode for the overall state estimation process is shown in Section 2.2.1, and for the NoLips algorithm is shown in Section 2.2.2.

### 2.2.1 State estimation algorithm

**Algorithm 1** State estimation with the Probabilistic Convex Mixture Model

**function** ESTIMATE-STATE($X, k, maxiters, \epsilon$)

$W \leftarrow$ Init-W($X, k$)

$M \leftarrow$ Init-M($X, W, k$)

**for** $iter \leftarrow 1...maxiters$ **do**

Update $W$ to minimize $-\log P(X \mid M, W)$, subject to $W$ nonnegative

Update $M$ to minimize $-\log P(X \mid M, W)$ , subject to $M$ nonnegative

**if** $M$ and $W$ changed less than $\epsilon$ this iteration **then**

**return** $M, W$ normalized

**end if**

**end for**

**return** $M, W$ normalized

**end function**

### 2.2.2 Sparse NoLips Algorithm

**Algorithm 2** One round of NoLips optimization for W

**function** NoLips-Estimate-W($X, M, W, k, \epsilon$)

  **for** $c \leftarrow 1...numcells$ **do**

    $\lambda \leftarrow 1/(2\sum_j X[j,c])$

    $ci \leftarrow [0]^k$

    **for** $g \leftarrow 1...numgenes$ **do**

      $mw \leftarrow X[g,c]/(M[g,:] * W[:,c])$

      **for** $j \leftarrow 1...k$ **do**

        $ci[j] \leftarrow ci[j] + mwM[g,j]$

      **end for**

    **end for**

    **for** $j \leftarrow 1...k$ **do**

      $W \leftarrow \max(0, W[j,c]/(1 + \lambda W[j,c](\sum_l M[l,j] - ci[j])))$

    **end for**

  **end for**

  **return** $W$

**end function**

The optimization step for $M$ proceeds identically, with $X^T$, $W^T$, and $M^T$ as the parameters to the above algorithm.

By default, Init-W($X, k$) simply performs a k-means clustering on D reduced with truncated SVD, and assigns 0.75 to the highest cluster for each cell and 0.25 divided among the remaining clusters. Init-M($X, W, k$) sets each column of M to be the mean of all cells assigned to that cluster.

W and M can also be initialized manually, or with the output of qualNorm or a different clustering/dimensionality reduction algorithm.

### 2.2.3 State estimation for different distributions

While UNCURL can easily be extended to different sampling distributions, here we have limited ourselves to three of the most common ones: Gaussian, log-normal and Poisson. It is easy to see that the state estimation problem for the Gaussian distribution, if variances are treated as uniform, is identical to the Non-Negative Matrix Factorization (NMF) problem. Thus, we utilize standard NMF solvers for this distribution followed by the column normalization of $W$ as stated above. Similarly, for log-normal data, transforming the data as $Y = \log(1 + X)$ makes the transformed dataset Gaussian distributed and allow us to again use NMF solvers.

### 2.2.4 Distribution selection

UNCURL has a set of possible distributions to choose from. In order to select the distribution to use automatically, we implemented a method using fit error (Fig 2.2). First, we fit the different distributions for each gene using maximum likelihood. Then, we compute the distance between the empirical distribution and each of the fitted distributions. This test is similar to the root-mean-square statistic for goodness of fit (Perkins et al., 2011). The distribution with the minimum such distance is considered the best fit. Finally, we output the 'best estimation fraction' vector which captures the fraction of genes for which each distribution is the best-fit distribution. This is meant to be a guideline for the selection of the sampling distribution during the matrix factorization process.

### 2.2.5 Initialization for state estimation

Since state estimation is a non-convex problem, its result depends greatly on the initialization. Two commonly used methods for NMF initialization are based on k-means and SVD (singular value decomposition), respectively (Langville et al., 2006; Boutsidis and Gallopoulos, 2008). In UNCURL, we have two ways to initialize the state estimation: (1) distribution-specific k-means initialization, and (2) truncated SVD + k-means based initialization. We describe our distribution specific K-means in the rest of the section, particularly for the Poisson case.

In the following sub-sections we develop an explicit framework to utilize prior biological information to initialize the matrix factorization. This relies on clustering each gene into clusters

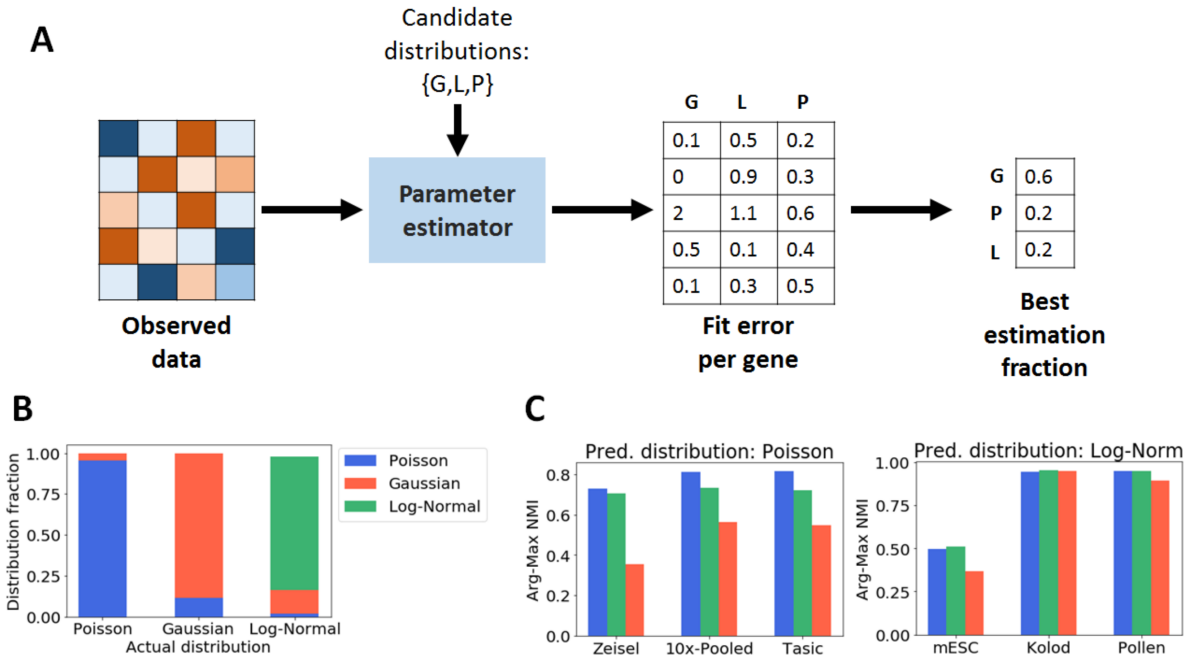



Figure 2.2: Selecting the best sampling distribution for a dataset from a set of distributions using Distribution Selector. A) Overview of Distribution Selector. B) 'Best Estimation Fraction' correctly identifies distributions of synthetic datasets. C) Comparison on different single cell datasets show that using predicted distribution leads to the highest cluster purity (measured using arg-max NMI).

of high and low expression, which is done using k-means clustering for Gaussian and log-normal (after log-transformation) distributions. Here we outline a similar procedure for the Poisson distribution that allows our approach to be consistent across different distributions along with an approach to initialize the clustering.

### Poisson k-means++

k-means++ (Arthur and Vassilvitskii, 2007) is a well known seeding method for the k-means clustering algorithm, which tries to identify $k$ points in the data with the highest mutual separation. While popular, its use of Euclidean distances between points makes it a poor fit for non-Gaussian distributed data. To use a similar approach for Poisson sampled data, we use a new distance metric. One intuitive distance would be the Poisson log-likelihood with one data point as the mean (say $y$) and the other as the point being considered (say $x$). Let ($ll_p(x|y)$

be defined as the log likelihood of a Poisson distribution with mean $y$. However, $ll_p(x|y)$ is not symmetric, which is required for a distance measure. We create a normalized version of this function which satisfies all properties of a distance measure:

$$\begin{aligned} d(x,y) &= ll_p(x|x) + ll_p(y|y) - (ll_p(x|y) + ll_p(y|x)) \\ &= (x-y)\log(\frac{x}{y}) \end{aligned}$$

This distance is based on the observation that value of $ll_p(x|y)$ is maximum when $x = y$. Thus, the $d(x, y)$ quantity measures the distance from the maximum value log-likelihood value for both $x$ and $y$ (for the sake of symmetry). This distance then replaces the Euclidean distance used in the standard implementation of k-means++.

**Poisson k-means**

Poisson k-means is very similar to the classical k-means clustering with the difference being in the underlying distribution of the data; it is essentially the hard EM algorithm applied with a Poisson distribution. As with state estimation, we assume we are provided the expected number of cell types $k$ and the data matrix $X \in \mathbb{R}^{n\times d}$. The first step of the algorithm involves calculating the Poisson log-likelihood for each cell given a set of means ($M \in \mathbb{R}^{n\times k}$ ), representing $k$ cell types and then assigning each cell to the cell type for which it has the highest log-likelihood. The logarithm of the probability that cell $c$ with observed counts $X_{\cdot c}$ is sampled from the type $i$ with gene expression $M_{\cdot i}$ is then $ll_p(c|i) = -\sum_j [M]_{ji} + [X]_{jc}\log([M]_{ji})$.

This is called the E step of the algorithm, which partitions the cells into distinct types. This is followed by the M step, where we find the means for each cell type that maximize the log-likelihood. For the case of the Poisson distribution, this is simply the arithmetic mean of the data given by:

$$M_{g,i} = \frac{1}{|S_i|}\sum_{c\in S_i} X_{g,c} \quad \forall g$$

Here, $S_i$ is the set of cell indices for which the log-likelihood is highest for the $i$th mean. The M step creates a new estimate of the means, which are then used to redo the E step. This

procedure is repeated till convergence or until a maximum number of iterations.

### 2.2.6 Qualitative semi–supervision with QualNorm

In many applications of scRNA-seq, there is a wealth of prior knowledge. For example, there may be FISH images or bulk gene expression data measured through microarrays or RNA-seq. Alternatively, marker genes may be known for a subset of cell types. Two major issues in using such information are the incompatibility between different data types (e.g. FISH images or microarray data with RNAseq data), and variability between experiments using the same technique (e.g. bulk RNA-seq batch effects). We develop a method to specifically account for such variations in order to leverage this prior information. A key benefit with our method is that since the prior information is leveraged in UNCURL preprocessing, it can boost the performance of downstream methods not designed to utilize such prior information.

A basic problem that we need to solve is in deciding how to incorporate such differing type of information into our framework. Our hypothesis is that even though the particular quantitative measurements may not transfer well to scRNA-seq, the qualitative information inherent in such a dataset can be exploited. To do that, we first convert the available prior information into a binary matrix, that can then be imported into our method. The binarization is motivated by the observation that cell type specific genes often have bimodal expression patterns: they are high in certain cell types and low in most others (Shmulevich and Zhang, 2002). In some cases, the prior biological knowledge available may be marker information which is already in a binary format. In other cases, where prior biological knowledge is available as bulk gene expression or other real-valued measurements, we first binarize the gene expression by thresholding around the central value for each differentially expressed gene. Differentially expressed genes are identified using DESeq2 (Love et al., 2014) on the bulk expression data using one-vs-all different expression analysis. In our experiments, we have used the Poisson version of t-test (Gu et al., 2008) to identify genes that are composed of two separate distributions and have limited the input to only include up to 25 'ON' genes per cell type with the highest p-values.

The inputs to the framework are the following: 1) A single cell sequenced data matrix $X \in \mathbb{R}^{n \times d}$, 2) the number of cell types expected in the data, $k$ and 3) a binary matrix of

dimension $B \in \{0,1\}^{n_0 \times k_0}$, where $n_0$ is the number of genes for which the information is provided and $k_0$ is the number of cell types for which the information is provided. We note that the number of cell types $k_0$ for which prior information is available can be lesser than the number of cell-types $k$, and the number of genes $n_0$ for which prior information is available can be lesser than total number of genes in the data $n$.

Now, the main algorithmic problem is how to utilize the matrix $B$ in solving the state-estimation problem. The obvious approach is to utilize the matrix as an initialization for state estimation. However there are three bottlenecks in our problem. (1) The matrix is binary, not real valued and it is unclear how to utilize such a matrix. (2) Information about every cell type is not available, i.e., $k_o \leq k$. (3) Information is not available about every gene, i.e., $n_o \leq n$. We deal with these issues sequentially.

Suppose we have a binary prior information matrix $B$ such that $n_0 = n$ and $k_0 = k$. We wish to convert this into a real valued matrix $M_0$ of the same size, where the expression is in the same scale as the scRNA-seq data $X$. To do this, we use the data matrix $X$ to calculate the high and low levels for each gene by clustering the observed expression values for that gene into two clusters and mapping the high and low values of cluster centers to 1 and 0 respectively. We note that in the case that $n_0 \leq n$ and $k_0 \leq k$, still the same method can be utilized to obtain the real-valued matrix $M_0$ of size $n_0 \times k_0$.

Next, we take up the issue that information is not available about all genes, i.e., $n_0 < n$. The basic idea that we exploit is the following: the knowledge even of some genes is sufficient to cluster all observed cells into $k_0$ types, using a distribution specific clustering (for example, the Poisson k-means algorithm described earlier). The $k_0$ cluster centers in dimension $n$ can then be used as the means of these clusters, thus effectively giving us an updated $M_0$ matrix of size $n \times k_0$.

Finally, we proceed to the issue that only some cell-types are specified, i.e., $k_0 \leq k$. In case that $k_0 = k$, we have an initialization for all the matrix $M_0$, which can be used an initialization for the alternating maximization algorithm. In case that $k_0 < k$, we do one round of distribution specific k-means++ with the first $k_0$ components initialized by the known data and the rest obtained as maximally distant points from the known points. This returns us the means for

all the $k$ components, which we update as $M_0$ of dimension $n \times k$. This matrix of predicted means $M_0$ is now used to initialize the various downstream algorithms, in particular serving as an initialization for maximizing $W$ in the alternating optimization.

## 2.3 Results

### 2.3.1 Distribution selection

To verify the accuracy of the distribution selection method, we first generated three synthetic datasets using different distributions (Poisson, Gaussian and log-normal). Each gene in the synthetic dataset has a mean that was randomly chosen between 0 and 1, with a constant variance for all genes for the Gaussian and log-normally distributed datasets.

As seen in Figure 2.2, the distribution selector correctly predicts the dominant distribution for each of the synthetic datasets. On the real datasets, we do not know the underlying distribution; however, we can check which distribution performs best on the downstream task of clustering. We check whether the predicted distribution leads to the highest accuracy. The predicted clusters are identified by assigning each cell to the highest weight class in the cell type fraction matrix, W. The cluster purity is then measured using Normalized Mutual Information (NMI) between the predicted clustering and the true cell types in the data and is seen to be highest for the distributions that are predicted to be dominant distribution according to distribution selector as seen in Fig 2.2C. In general, the Poisson distribution is seen to be a better fit for count or UMI data, while the log-normal distribution is a better fit for normalized (FPKM, RPKM, TPM) data.

Having identified the sampling distribution for a dataset, the state estimation procedure factorizes the data into two matrices, $M$ and $W$ using the distribution, as described in the Methods section. The product of these factorized matrices then can be treated as an estimate of the 'true state' matrix and used for all subsequent downstream learning tasks.

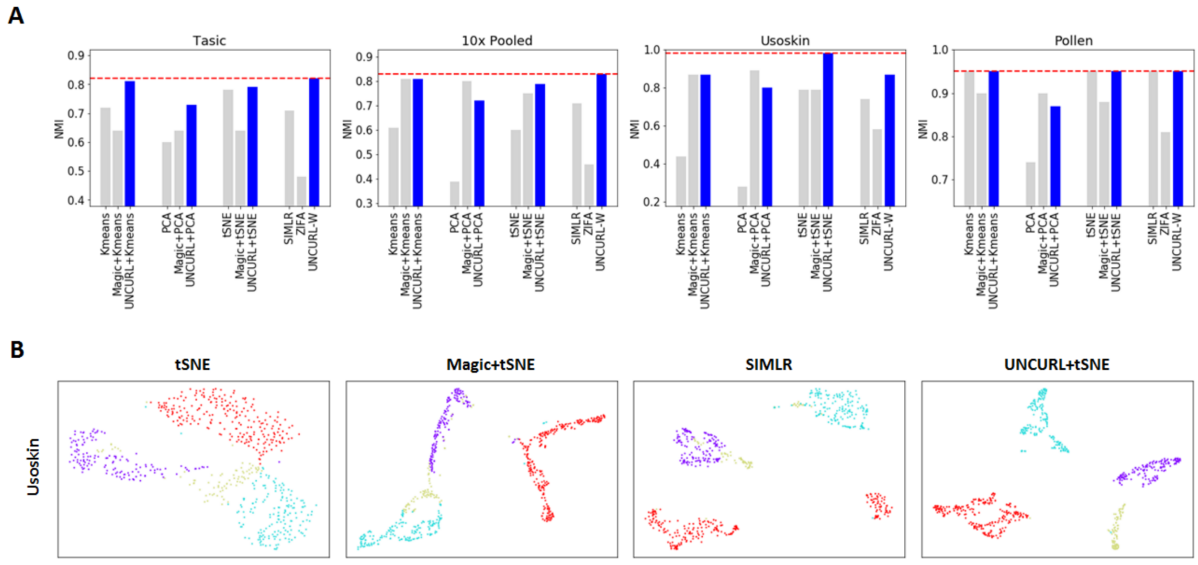



Figure 2.3: Preprocessing with UNCURL leads to improved visualization and clustering performances. A) Comparison of various clustering approaches with and without preprocessing on different scRNA-seq datasets. B-C) Different 2D visualizations of the Usoskin and Tasic datasets respectively.

### 2.3.2 Clustering and visualization

Clustering and dimensionality reduction are common downstream tasks for scRNA-seq data. These are both unsupervised tasks: clustering involves dividing the cells into different cell types based on similarity of gene expression patterns, while dimensionality reduction involves creating a low-dimensional view of the data for visualization or clustering. Both tasks are useful in identifying cell sub-populations in an unlabelled dataset.

Two of the most commonly used tools for scRNA-seq data are PCA and tSNE (Maaten and Hinton, 2008). While these algorithms have different underlying mechanisms for converting the high dimensional information to typically 2 or 3 dimensions, they assume Gaussian or t-distributions for the dataset, which is often not reflective of the actual sampling distribution of the data. To alleviate such issues, specialized tools have been developed for scRNA-seq data such as SIMLR (Wang et al., 2017) and ZIFA (Pierson and Yau, 2015), which explicitly account for the distribution of scRNA-seq datasets. While ZIFA explicitly accounts for zero inflation, it does not account for the full sampling distribution. SIMLR uses multiple Gaussian kernels to fit the

| Dataset | tSNE + kmeans | SIMLR + kmeans | ZIFA + kmeans | Magic + tSNE + kmeans | UNCURL+ argmax | UNCURL+ tSNE + kmeans |
|---|---|---|---|---|---|---|
| 10x_pooled_full | 0.65 | 0.59 | N/A | 0.54 | **0.83** | 0.56 |
| 10x_PBMC | 0.42 | 0.42 | N/A | 0.31 | **0.57** | 0.38 |
| Zeisel | 0.72 | **0.77** | 0.42 | 0.53 | 0.73 | 0.67 |
| Zeisel_sub | 0.55 | 0.58 | 0.43 | 0.37 | **0.64** | 0.61 |
| Tasic | 0.78 | 0.71 | 0.48 | 0.64 | **0.82** | 0.79 |
| mESC | 0.38 | **0.81** | 0.41 | 0.3 | 0.51 | 0.54 |
| Kolod | 0.99 | 0.99 | 0.55 | **1.0** | 0.95 | **1.0** |
| Pollen | **0.95** | **0.95** | 0.81 | 0.88 | **0.95** | **0.95** |
| Usoskin | 0.79 | 0.74 | 0.58 | 0.79 | 0.87 | **0.98** |

| Dataset | kmeans | PCA + kmeans | Magic + kmeans | Magic + PCA + kmeans | UNCURL+ kmeans | UNCURL+ PCA + kmeans |
|---|---|---|---|---|---|---|
| 10x_pooled_full | 0.34 | 0.59 | 0.71 | 0.71 | **0.85** | 0.7 |
| 10x_PBMC | 0.4 | 0.25 | 0.07 | 0.07 | **0.57** | 0.5 |
| Zeisel | 0.59 | 0.44 | 0.49 | 0.48 | 0.74 | 0.55 |
| Zeisel_sub | 0.43 | 0.43 | 0.42 | 0.37 | **0.68** | 0.5 |
| Tasic | 0.72 | 0.6 | 0.64 | 0.64 | 0.81 | 0.73 |
| mESC | 0.4 | 0.32 | 0.29 | 0.3 | 0.62 | 0.56 |
| Kolod | 0.92 | 0.65 | **1.0** | **1.0** | 0.97 | 0.97 |
| Pollen | **0.95** | 0.74 | 0.91 | 0.9 | **0.95** | 0.87 |
| Usoskin | 0.44 | 0.28 | 0.89 | 0.89 | 0.87 | 0.8 |

Table 2.1: NMI results for various preprocessing/clustering methods on different datasets. In most datasets, a method using UNCURL was either the best method, or tied as the best method.

data without having an explicit sampling model. While these tools lead to improvements over PCA/tSNE, we demonstrate that using UNCURL as a preprocessing tool can greatly improve the performances of these common dimensionality reduction tools and often even make them better than specialized tools such as ZIFA/SIMLR.

UNCURL can be used for clustering in several ways. The simplest clustering method is to assign as the cluster $c_i = argmax_j W_{j,i}$ for cell $i$, where $W$ is inferred from state estimation. We call this method UNCURL-W. In addition, clustering can benefit from UNCURL preprocessing by running tSNE or PCA and then k-means on the output of UNCURL.

We demonstrate the utility of using UNCURL as a preprocessing tool by comparing the

clustering performance of various methods with and without preprocessing with UNCURL, along with clustering after dimensionality reduction with ZIFA and SIMLR. Additionally we compare with another preprocessing tool, Magic (Dijk et al., 2017). Performance was measured using the NMI between true and predicted clusters with the selected preprocessing and clustering methods, as done in Wang et al. (2017). As seen in Fig 2.3A, UNCURL improves the performance of k-means and PCA on all four datasets (a more comprehensive set of results can be seen in Table 2.1), and it improves the performance of tSNE on most datasets. Moreover, in all four datasets the top performing approach (showed with the red dotted line) is an UNCURL preprocessed approach.

To investigate the effect of UNCURL preprocessing on visualization, we run tSNE on the Usoskin et al. (2015) and the Tasic et al. (2016) datasets, which are FPKM and Count valued respectively. We compare against the unprocessed tSNE visualization, tSNE after preprocessing using Magic and visualization using SIMLR (which is closely related to tSNE). The Usoskin dataset (Fig 2.3B) comprises of sensory neuronal cells from four principal neuronal types. We see that while all other approaches end up grouping one or more cell types together, UNCURL leads to complete separation of all principal neuronal types.

The Tasic dataset (Figure 2.3) is comprised of cells from mouse cortex tissue, with 49 cell types, including neuronal and non-neuronal cells. Using UNCURL along with tSNE, we were able to identify 49 distinct clusters that corresponded very strongly to the cell types identified previously. For the same dataset, the performance of tSNE without preprocessing and Magic preprocessing was seen to be significantly worse, with many cell types being grouped together.

To investigate the effect of UNCURL preprocessing on visualization, we run tSNE on the Usoskin et al. (2015) and the Tasic et al. (2016) datasets, which are FPKM and Count valued respectively. We compare against the unprocessed tSNE visualization, tSNE after preprocessing using Magic and visualization using SIMLR (which is closely related to tSNE). The Usoskin dataset (Fig 2.3B) comprises of sensory neuronal cells from four principal neuronal types. We see that while all other approaches end up grouping one or more cell types together, UNCURL leads to complete separation of all principal neuronal types.

The Tasic dataset is comprised of cells from mouse cortex tissue, with 49 cell types, including

neuronal and non-neuronal cells. Using UNCURL along with tSNE, we were able to identify 49 distinct clusters that corresponded very strongly to the cell types identified previously. For the same dataset, the performance of tSNE without preprocessing and Magic preprocessing was seen to be significantly worse, with many cell types being grouped together.

An important point to note here is that in each of these datasets, the true number of cell types ($k$) was known and used during state estimation. However, this may not always be the case, so robustness to varying values of $k$ is desirable. We tested the robustness of clustering NMI to $k$ on various datasets (Figure 2.4) and found that UNCURL is quite robust to overestimation of $k$ (to within a factor of 2), but that underestimation can lead to worse results.

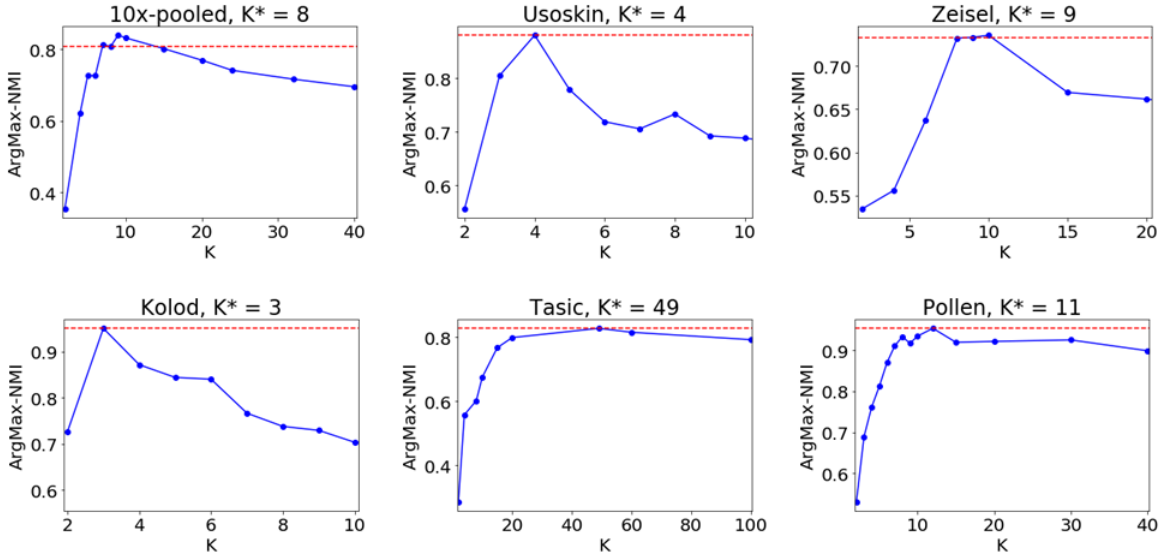



Figure 2.4: Robustness of NMI to choice of K. For a number of different datasets, we compared the NMI of arg-max(W) for different choices of K, where K* is the "correct" number of clusters. We see that while the NMI of UNCURL peaks around the true value of K, for most datasets, increasing it further leads to only a slight loss in precision. This shows that UNCURL is quite robust to the choice of K.

### 2.3.3 Prior knowledge improves UNCURL

To demonstrate the utility of prior biological knowledge, we look at a dataset with available qualitative information in the form of bulk RNA-seq data obtained from different experimental conditions, and consider the effect of semi-supervision on visualization with tSNE. Specifically, we focus on the a subset of the data from Zeisel et al. (2015), comprised of five non-pyramidal

cell types: oligodendrocytes, astrocytes, interneurons, microglia and endothelial cells. An upper bound on the performance with semi-supervision information is obtained when we feed the aggregate means of the true clusters (the means of all cells with each of the ground-truth labels) as the initialization. We compare this with semi-supervision using the output of QualNorm, bulk means and unsupervised preprocessing. In order to test the validity of our QualNorm framework, we compare the performance with aggregate-mean initialization to the performance obtained when we process these aggregate means through the QualNorm framework. In Fig 2.5B, the four initialization methods are compared, and it is seen that while semi-supervision with aggregate means and QualNorm means lead to a clear separation of cell types and an improved over unsupervised preprocessing, initialization with bulk means leads to worse visualization for this dataset. A similar experiment with the 10x pooled dataset (Figure 2.6) also leads to a qualitative improvement in tSNE/PCA based visualizations and improvement in clustering purity.

First, we simulate the scenario where we have information available about a subset of cell types by using different number of known means (generated by binarizing the public bulk data and passing it through the QualNorm) and generating the other means using our version of the distribution informed k-means++ algorithm. We then calculate NMI between predicted and true clusters (using arg-max of W) to quantitatively measure the performance of state estimation. As seen in Fig 2.5C, we observe that increasing the number of known means leads to improvement in accuracy. Moreover, we also see that prior information about even a subset of cell types is usually enough to improve the performance over the completely unsupervised case.

Then, to test the effect of having information about only a subset of the genes, we chose different number of top cell type specific genes (measured by one-vs-all differential expression) as initialization points for UNCURL and tested their effect on the purity of the clusters. Furthermore, we also tested the effectiveness of three different semi-supervision strategies namely, 1) true cluster centers (generated by taking aggregate means with the known true labels), 2) bulk means and 3) QualNorm means. It is seen in Fig 2.5D, that while the true cluster centers lead to almost perfect estimation of cluster membership, the QualNorm means lead to better accuracy than using the bulk data for most subset sizes, which we attribute to biases inherent

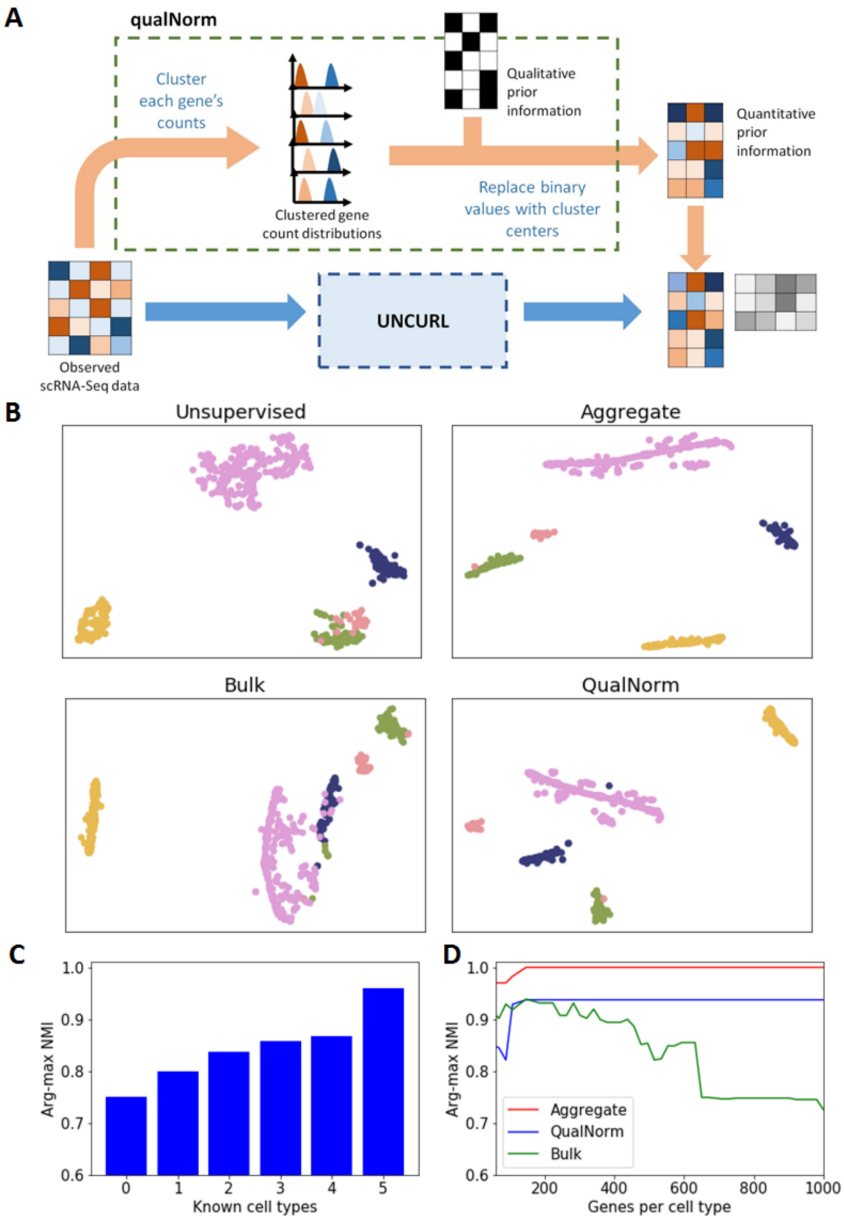



Figure 2.5: Semi-supervision with prior biological information can further improve the performance of UNCURL. A) An illustration of the 'qualNorm' framework to convert qualitative prior information into good initialization points for unsupervised learning algorithms. B) Visualization using tSNE with unsupervised, aggregate, bulk and QualNorm semi-supervision on a subset of the Zeisel dataset (containing 1672 cells and 5 cell types). C) Comparison of improvement in purity with prior information about different number of cell types. D) Comparison of clustering purity with prior information about different number of cell type specific genes.

to different sequencing methods. This effect becomes more pronounced as the number of genes being considered increases. An interesting observation here is that information about a few cell

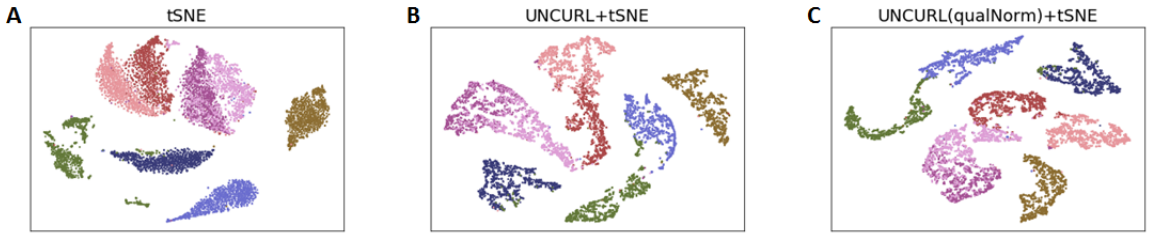



Figure 2.6: Effect on tSNE based visualization for 10x pooled dataset using different initialization strategies. A) tSNE without pre-processing. B) Unsupervised UNCURL + tSNE. C) QualNorm semi-supervised UNCURL + tSNE.

type specific genes is enough to very high NMI values, even when the information is qualitative.

While the results in these two cases of missing information highlight the flexibility of the 'QualNorm' framework to handle different amounts of provided prior information, they also demonstrate how having even a little additional information is enough to improve unsupervised learning results significantly.

### 2.3.4 Lineage estimation

One biologically important application of scRNA-seq is lineage estimation, which is often used to study the dynamics of various genes during cell differentiation or development. Lineage inference aims at identifying smooth continuous manifolds in which cells lie in order to study the gradual change in gene expression during various developmental processes. While there exist several common tools that exist for this purpose (Trapnell et al., 2014; Welch et al., 2016; Qiu et al., 2017), most tools operate directly on the sampled observed data. Additionally, most lineage estimation tools do not allow for the incorporation of any prior biological data beyond selecting the subset of genes to use for lineage inference. Here we demonstrate that the use of UNCURL preprocessing can lead to the estimate of cleaner lineages as well as allow for the incorporation of qualitative prior information into the lineage inference process.

Since it is not possible to obtain a ground truth ordering of cells, we first study the effect of UNCURL preprocessing on simulated datasets. We created two separate synthetic lineages (a linear and a branched lineage), and sampled using a Poisson distribution. For both datasets, we preprocessed the datasets using UNCURL and Magic and tested three commonly used lineage inference tools (Monocle, Monocle2 and Slicer) on both preprocessed and unprocessed datasets

(Figures 2.7 - 2.9).

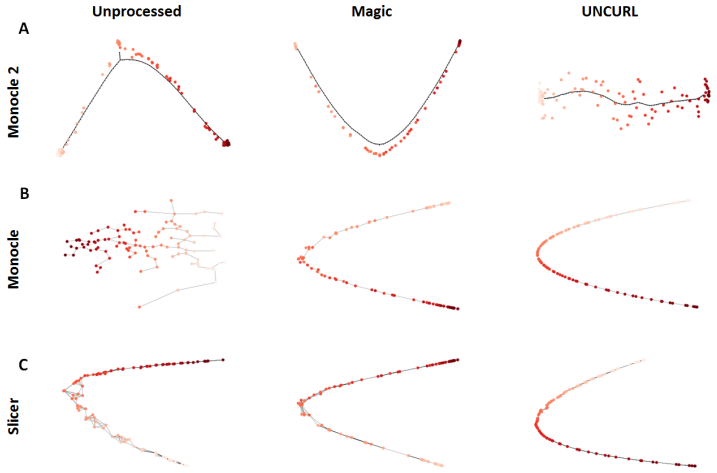



Figure 2.7: Comparison of lineage estimation on a synthetic linear lineage containing 100 cells. Comparison of different algorithms and different three different pre-processing methods namely A) Unprocessed, B) Magic pre-processed, C) UNCURL pre-processed. Cells are colored by true progress along the lineage.

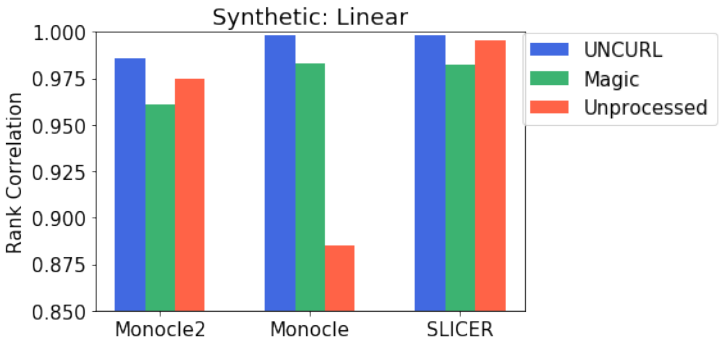



Figure 2.8: Rank correlation (with true pseudotime) of the pseudotime estimated by different lineage estimation algorithms using different pre-processing methods.

For the linear dataset, a good measure of lineage accuracy is the rank correlation of the true ordering of the cells with the pseudotime (an arbitrary metric that is commonly used to measure progress along a trajectory). We find that UNCURL improved the rank correlation of all three methods compared to both the unprocessed data and Magic, whereas preprocessing using Magic lead to worse performance for both Monocle2 and Slicer. For the branched data, there isn't a similarly simple way to quantify lineage estimates so we looked at the 'branch purity' (a measure of how well cells have been ordered into the right branches). Even in this case we found that the

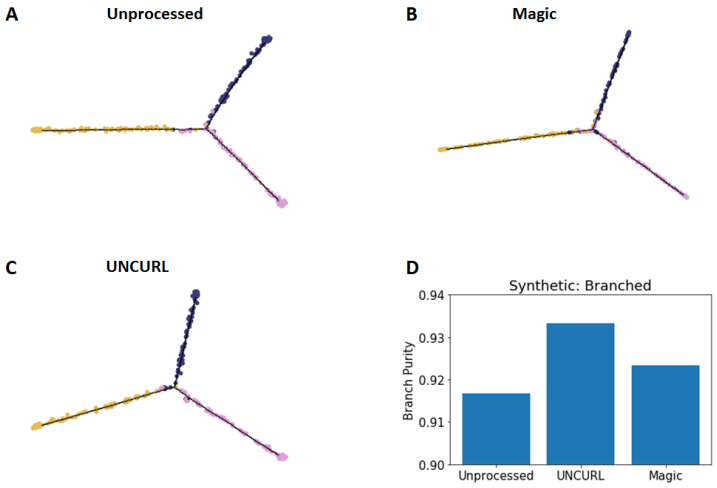



Figure 2.9: Estimated lineage along a branched trajectory (containing 300 cells, 100 per branch) using Monocle2 after different pre-processing methods. A) Unprocessed, B) UNCURL pre-processed, C) Magic pre-processed. D) Comparison of branch purity measured using identified branches and actual branches.

branch purity obtained in Monocle2 after preprocessing with UNCURL was higher than both unprocessed and preprocessing using Magic.

We now look at real biological data with some amount of ground truth data available. We specifically focus on the dataset of Hanchate et al. (2015) which contains cell types comprised of four stages of olfactory neurogenesis along a linear differentiation path. The cell types were identified using markers specific to different stages of development. While this information itself is of the form of qualitative prior information, using the marker genes used to label the cells would make the problem trivial for UNCURL. Hence, we instead simulate a bulk dataset by using the true labels of the dataset, which is then binarized to generate qualitative marker information for all sufficiently expressed genes (same as used in the original paper). We then use this to use both the qualNorm initialization as well as unsupervised initialization to preprocess the dataset using UNCURL. We then perform lineage inference using Monocle2, Monocle and Slicer on the unprocessed data, UNCURL preprocessed data and Magic preprocessed data. As seen in Fig 2.10 A-C, QualNorm initialized UNCURL leads to consistent improvements of the lineage inference algorithms when measured by the amount of overlap between the known cell types in the lineage graphs. On the other hand, while preprocessing using Magic seems to lead to qualitatively similar lineages for Monocle2 and Slicer, it also leads to the estimation of a

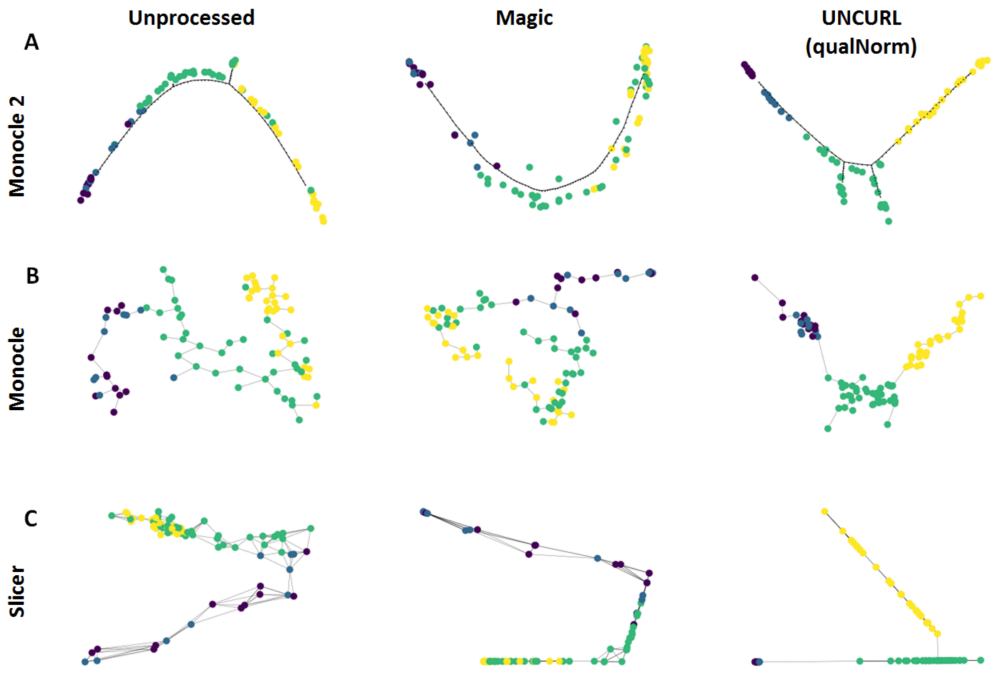



Figure 2.10: Semi-supervised preprocessing using UNCURL can dramatically improve the performance of lineage inference algorithms. A-C) Lineages inferred by Monocle2, Monocle and Slicer respectively with no-preprocessing, Magic preprocessed and semi-supervised UNCURL preprocessed for the dataset of Hanchate et al. (2015) (containing 85 cells from 4 cell types). The lineages obtained after preprocessing using semi-supervised UNCURL lead to much clearer separation of known cell types in the predicted lineages.

major non-existent branch in the case of Monocle (Fig 2.10).

### 2.3.5 Scalability

UNCURL is capable of running on larger datasets comprising of up to millions of cells. UNCURL uses a fast public NMF package (Pedregosa et al., 2011) for the log-normal and Gaussian distributions, while using SPNoLips (Sparse-Parallel-NoLips - Section 2.2.2), which is a custom implementation based on the NoLips algorithm (Bauschke et al., 2016) for the Poisson distribution. Both implementations are capable of using sparse matrices as input for memory and runtime advantages, and are parallelizable.

The runtime of UNCURL is $O(n_p k)$, where $n_p$ is the number of nonzero elements in the input matrix $X$, and $k$ is the number of cell types. This means that UNCURL scales linearly in the number of cells in the dataset. To deal with the dependence on $k$, one possibility is to

use UNCURL hierarchically: first run it with a small $k$ on the entire dataset, then partition the dataset based on the assigned clusters, and run UNCURL on the subsets.

The computational performance of UNCURL on various datasets compares favorably to that of other methods as seen in Figure 2.11. The runtime of UNCURL is usually less than that of the other comparable methods for clustering tasks on most datasets as seen in Figure 2.11, with more comprehensive results shown in Table 2.3.5. The memory usage was lower than that of comparable methods such as SIMLR and Magic. UNCURL's performance is best on sparser datasets, where more entries are zero, since the NoLips update function only uses nonzero values of the data matrix. Runtime comparisons with Magic and ZIFA are limited because Magic's memory usage is quadratic in the number of cells and ZIFA is slow compared to other algorithms, making it impractical to run on the largest datasets.

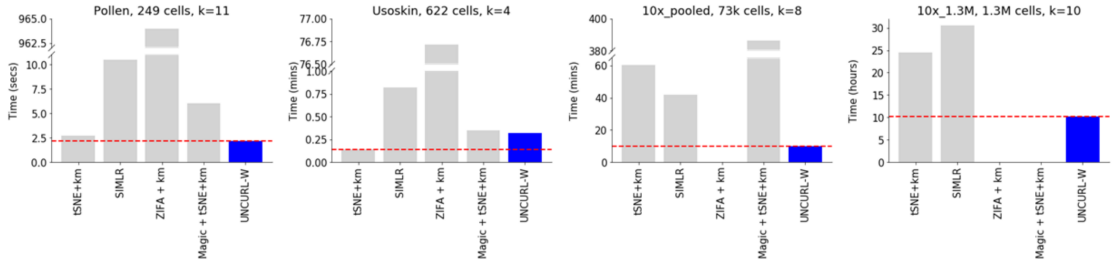



Figure 2.11: Timing comparison of different clustering approaches for various scRNA-seq datasets. UNCURL is faster than other approaches on most datasets of various sizes. Moreover, unlike methods like ZIFA and Magic, UNCURL is scalable for large datasets. A more comprehensive comparison can be found in Table 2.3.5.

| Dataset | Distribution | Cells | K | tSNE + kmeans | SIMLR | ZIFA + kmeans | Magic + tSNE + kmeans | UNCURL |
|---|---|---|---|---|---|---|---|---|
| 10x_pooled | Poisson | 73233 | 8 | 3615 | 2504 | N/A | 23188 | **591** |
| 10x_PBMC | Poisson | 68579 | 10 | 4051 | 2733 | N/A | 9670 | **490** |
| Zeisel | Poisson | 3005 | 9 | 120 | 184 | 3040 | 93 | **62** |
| Zeisel_sub | Poisson | 3005 | 9 | 134 | 149 | 12778 | 96 | **36.5** |
| 10x_1.3M | Poisson | 1.3 mil | 10 | 87982 | 109887 | N/A | N/A | **36869** |
| Tasic | Poisson | 1629 | 49 | **24** | 27 | 2734 | 56 | 144 |
| mESC | LogNorm | 182 | 3 | 2 | 6.73 | 287 | 3 | **0.5** |
| Kolod | LogNorm | 704 | 3 | 9 | 84.75 | 2401 | 13 | **2** |
| Pollen | LogNorm | 249 | 11 | 2.7 | 10.5 | 964 | 6 | **2.2** |
| Usoskin | Poisson | 622 | 4 | **8.3** | 49.29 | 4603 | 21 | 19 |

Table 2.2: Comparison of run times of various algorithms on different single cell RNA-Seq datasets.

## 2.4 Discussion

In this section, we introduced a preprocessing framework for scRNA-seq data. Our framework, UNCURL, uses the estimated sampling distribution of scRNA-seq data together with a convex mixture model assumption to estimate a true state matrix from observed scRNA-seq data. UNCURL further includes a computational framework, qualNorm, which can be used to incorporate prior biological knowledge into an improved estimate of the true state matrix.

By comparing against several benchmarking datasets, we demonstrated that preprocessing using UNCURL leads to superior separation of cell types in reduced dimensions as well as higher cluster purity for clustering tasks compared to prior tools. We further showed that semi-supervision using different types of prior information can lead to further improvement in accuracy of the learning tasks. Furthermore, we demonstrate that semi-supervised preprocessing using UNCURL allows the incorporation of prior information in even lineage estimation tasks. UNCURL scales to large datasets and typically runs faster than prior methods, particularly on large and sparse datasets. The run time for UNCURL scales linearly with the number of cell types in the dataset, but it may be possible to further reduce run-time using a hierarchical strategy.

UNCURL is an efficient preprocessing framework for several unsupervised and semi-supervised learning tasks, but it still has some limitations. Tools that already specifically account for the sampling effect of single cell sequencing may not benefit from UNCURL pre-processing. Another potential shortcoming is the need to know or at least have a good estimate of the true number of cell types. This is a limitation common to many approaches for single cell data analysis, and we are currently working on incorporating biological prior information into selecting the optimal number of cell types in a dataset. Another limitation is that the semi-supervision framework converts all prior information into a binary format, although it can be easily generalized to multi-level quantizations. While this constraint is reasonable for truly cell type specific genes, it still limits the amount of prior information that can be used. For instance, many genes that are not strictly cell type specific might also contain useful information that cannot currently be utilized. Future work can be aimed at expanding the semi-supervision framework to incorporate more diverse qualitative information.

# Chapter 3

# CellMeSH

## 3.1 Introduction

We will describe CellMeSH, which is a database and query method that can be used to identify the cell types present in an scRNA-seq dataset. There are two key innovations in CellMeSH. First of all, CellMeSH builds its database in a scalable and automatic way, by automatically linking cell types and gene names from the literature. Second, to address the challenges of publication bias and potential errors in the gene-cell associations, we develop a novel probabilistic query method using maximum likelihood estimation.

### 3.1.1 Previous Work

There exist a number of tools that can be used for cell type identification, some specifically designed for single-cell methods, some more general. These methods are described in detail in Table 3.1.

First, there are a number of *cluster-level* methods that annotate clusters with cell types using the differentially expressed genes in the cluster. They work by querying cell type databases for similar genes (Chen et al., 2013; Stachelscheid et al., 2013; Zhang et al., 2018b; Hänzelmann et al., 2013; Franzén et al., 2019). The databases are collected either from a few specific studies Chen et al. (2013); Stachelscheid et al. (2013), from manual literature surveys Zhang et al. (2018b); Hänzelmann et al. (2013), or from scRNA-seq experiments which have their clustered

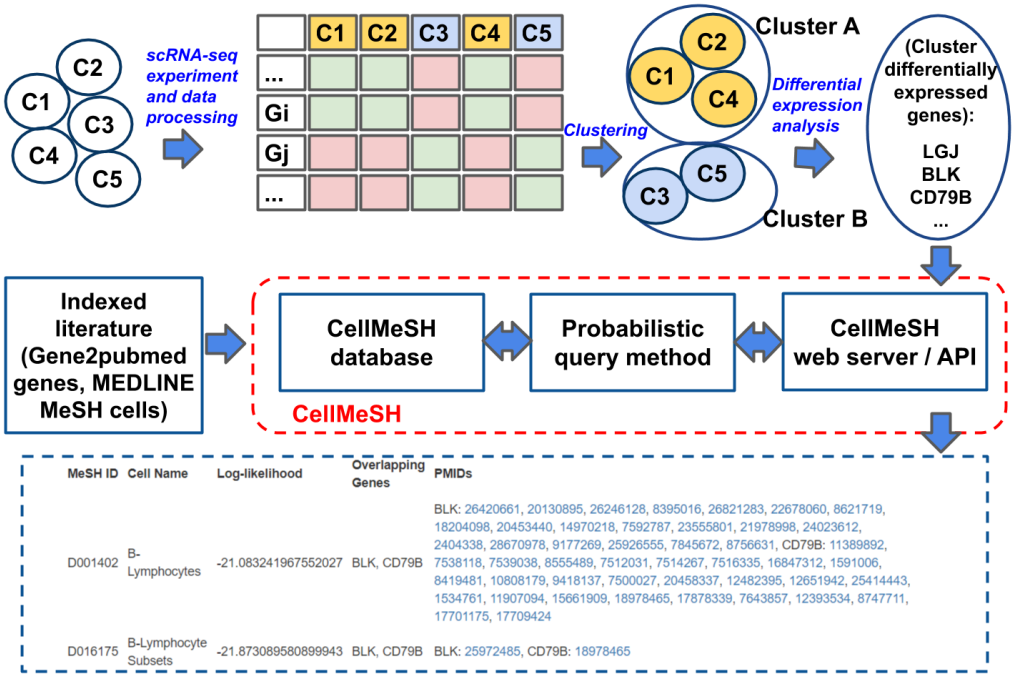



Figure 3.1: **CellMeSH Overview.** CellMeSH enables automated annotation of cell types, which is usually the last step of scRNA-seq analysis. CellMeSH can be accessed through a web server and API. The input to CellMeSH are the differentially expressed genes of clustered cells and optional prior knowledge of cell types, the outputs are ranked candidate cell types for each cluster, and for each candidate cell the log-likelihood score (Equation 3.2), the overlapping genes and the related literature resources. CellMeSH relies on a database created by linking the Gene2pubmed genes and MEDLINE MeSH cell types. The database is queried by a probabilistic method based on maximum likelihood estimation.

cells pre-annotated according to cell type markers Franzén et al. (2019). The database query mechanisms can return a list of unsorted cell types Stachelscheid et al. (2013); Zhang et al. (2018b); Franzén et al. (2019) or a list of cell types sorted by a measure of statistical significance given the query genes, either based on Fisher's exact test Fisher (1935); Chen et al. (2013) or a Kolmogorov-Smirnov test Hänzelmann et al. (2013). A common issue for these cell type identification methods is that their databases are not comprehensive, and it is also laborious to update and expand them.

Another line of recent work directly predicts the cell types on the *cell level*, using the gene-cell matrix, rather than using clusters (Butler et al., 2018; Alavi et al., 2018; Kiselev et al., 2018; Aran et al., 2019; Hou et al., 2019; Ma and Pellegrini, 2019; Tan and Cahan, 2019; Pliner et al., 2019; Zhang et al., 2019b) 3.1. However, these methods typically require either existing

annotated gene expression profiles Butler et al. (2018); Kiselev et al. (2018); Aran et al. (2019); Alavi et al. (2018); Hou et al. (2019); Ma and Pellegrini (2019); Tan and Cahan (2019) or hand-curated cell-type marker-gene files Pliner et al. (2019); Zhang et al. (2019b) as prior knowledge. Many of these methods follow a machine learning approach, by first training a model on existing datasets, and then using the trained model to either label the input gene expression vector with a reference cell type Ma and Pellegrini (2019); Tan and Cahan (2019); Pliner et al. (2019), or to project the input gene expression vector onto an embedding vector and then match that embedding vector with the reference cell type that has the most similar embedding Alavi et al. (2018); Kiselev et al. (2018); Aran et al. (2019). Some of these methods follow a more statistical approach (without training), by annotating the input gene expression vector with the reference cell type that has the highest correlation (Butler et al., 2018; Kiselev et al., 2018; Aran et al., 2019; Hou et al., 2019) or maximum a posteriori estimation score (Zhang et al., 2019b) for the input.

None of these methods are able to annotate cell types that have not been seen in prior experiments (Butler et al., 2018; Alavi et al., 2018; Kiselev et al., 2018; Aran et al., 2019; Hou et al., 2019; Ma and Pellegrini, 2019; Tan and Cahan, 2019) or absent from the marker file, which typically contains only a small number of known cell types (Pliner et al., 2019; Zhang et al., 2019b).

## 3.2 Methods

### 3.2.1 Database Construction

CellMeSH was created by combining two existing literature indices: the MEDLINE citation index (noa, 2019c), which contains publication abstracts with associated metadata, and the gene2pubmed database (Maglott et al., 2007), which contains a mapping of genes to publications.

The MEDLINE citations index is a bibliographic database containing around 30 million references (as of 2021) to biomedical and life science journal articles, including to most articles in NCBI PubMed. The key metadata from MEDLINE that we are interested in are the associated Medical Subject Headings, or MeSH terms (noa, 2019b), a subset of which represent cell types.

Table 3.1: **Existing Works for Cell-type Identification.** They can differ in multiple aspects: the Type (if the work provides a webserver, a general method, or a user-side software), the Target (if the work can be used to annotate cell types for clusters or for cells directly), the Query Input (if a gene set or gene expressions will be used to represent the query cell), the Query Method (if the results are retrieved by enumeration, statistical tests, or machine learning inferences), the Database (from what sources the gene-cell knowledge come from) and the Query Output (how the retrieved cell types can be provided).

| Name | Type | Target | Query Input | Query Method | Database | Query Output |
|---|---|---|---|---|---|---|
| CellMeSH | webserver | cluster | gene set | maximum likelihood estimation | indexed literature (MEDLINE noa (2015) MeSH noa (2019b) cells and Gene2pubmed Maglott et al. (2007) genes) | ranked list of cell types |
| CellFinder (2013) Stachelscheid et al. (2013) | webserver | cluster | gene set | enumeration | mainly from existing ontologies, augmented by text mining of literature with manual verification | list of cell types |
| CellMarker (2018) Zhang et al. (2018b) | webserver | cluster | gene set (actually a single gene) | enumeration | manual literature survey | list of pairs of tissue and cell type |
| PanglaoDB (2019) Franzén et al. (2019) | webserver | cluster | gene set | enumeration | scRNA-seq experiments (cell clusters were pre-annotated according to hand-curated cell-type markers) | list of labeled cell clusters, and a barplot for frequencies of the labels |
| Enrichr (2013) Chen et al. (2013) | webserver | cluster | gene set | modified Fisher exact test | Mouse and Human Gene Atlases Su et al. (2004), CCLE Barretina et al. (2012) and NCI-60 Weinstein (2006) | ranked list of cell types |
| Hypergeometric testFisher (1935) | method | cluster | gene set | Fisher's exact test | need manual compile of cell and marker genes | ranked list of cell types based on raw enrichment scores |
| GSVA (2013) Hänzelmann et al. (2013) | method | cluster | gene expression | Kolmogorov Smirnov random walk statistic | need manual compile of cell and marker genes | ranked list of cell types based on raw enrichment scores |
| scQuery (2018) Alavi et al. (2018) | webserver | cell (or optionally cluster) | gene expression (or optionally gene set) | nearest neighbor of neural embedding | scRNA-seq experiments from GEO and ArrayExpress | ranked list of cell types |
| Seurat (v1 2015, v2 2018) Satija et al. (2015); Butler et al. (2018)) | software (R) | cluster, or cell (label transfer) | gene set, or gene expression | for clusters, manual annotation; for cells, canonical correlation analysis and alignment | for clusters, knowledge of marker genes; for cells, scRNA-seq experiment datasets | labeled single cells |
| scMap (2018) Kiselev et al. (2018) | software (R) | cell | gene expression | search query cells for nearest centroids (using Pearson, Spearman, and cosine distances) or sub-centroid approximations (using consine distance) of reference cells | scRNA-seq experiments | labeled single cells |
| SingleR (2019) Aran et al. (2019) | software (R) | cell | gene expression | correlation search (Spearman coefficient) and iterative fine tuning | Single-cell or bulk RNA-seq experiments | labeled single cells |
| scMatch (2019) Hou et al. (2019) | software (Python) | cell | gene expression | correlation (Spearman or Pearson) | annotated gene expression data | ranked list of cell types based on raw scores |
| ACTINN (2019) Ma and Pellegrini (2019) | software (Python) | cell | gene expression | neural network | annotated gene expression data | ranked list of cell types based on raw scores |
| SingleCellNet (2019) Tan and Cahan (2019) | software (R) | cell | gene expression | random forest classification Ho (1995) | annotated gene expression data | labeled single cells |
| Garnett (2019) Pliner et al. (2019) | software (R) | cell | gene expression | supervised classification | need manual compile of cell and marker genes | labeled single cells |
| cellassign (2019) Zhang et al. (2019b) | software (R) | cell | gene expression (critical to use only marker genes) | maximum a posteriori estimation using EM algorithm Dempster et al. (1977) | cell and marker genes from bulk RNA-seq experiments or literature | ranked list of cell types |

There are 570 terms under the heading "Cell", around 475 of which represent actual cell types and not chromosomes, cell structures, and the like. Cell types are not species-specific, as the MeSH ontology does not distinguish cell types with respect to species. However, when predicting cell types given species-specific genes, cell types that do not exist in the target species are likely to have low scores and are unlikely to be selected.

For each cell type from MeSH, we found all publications where they occur. This gives us around 3 million references in total.

Next, for each target species (human and mouse in this case), we further filter these references to keep only the references containing the species-specific genes from Gene2pubmed, a database that links standardized NCBI genes (Maglott et al., 2007) with PubMed articles. Gene2pubmed currently references $20,164$ human and $27,322$ mouse genes. Species-specific filtering results in a reduced dataset of around $300,000$ articles for human and around $209,000$ articles for mouse. Each article is therefore associated with a set of NCBI genes, as well as with a set of MeSH cell types.

We next construct distinct tables for each species. A gene and a cell type are considered to co-occur if there is at least one article that is associated with the cell type in MEDLINE and with the gene in Gene2pubmed. For example, in the article with PubMed ID p=1591006, we have indexed gene $g$ = "CD79B" (from Gene2pubmed) and indexed MeSH cell types $c1$ = "B-Lymphocytes" and $c2$ = "Hematopoietic Stem Cells" (from MEDLINE), meaning that $g$ co-occurs with both $c1$ and $c2$ in $p$. In the tables, each gene is a row and each cell type is a column, and the entry denotes a list of articles in which the gene co-occurs with the cell type.

For human, 3.8% of all possible (20,164×570) gene-cell pairs have non-zero counts, and around 300,000 PubMed articles each contain at least one pair. For mouse, 2.4% (27,322×570) gene-cell pairs have non-zero counts, and around 209,000 PubMed articles each containing at least one pair.

The process of constructing the database is shown in Figure 3.2.

The CellMeSH database is larger in scale than the hand-curated CellMarker (Zhang et al., 2018b) and PanglaoDB (Franzén et al., 2019) databases. CellMarker contains around 8900 genes, 340 cell types and $< 1\%$ co-occurring gene-cell pairs for human; the numbers are around

7200, 310 and $< 1\%$ for mouse. In PanglaoDB, there are around 4600 genes, 178 cell types and 1% co-occurring gene-cell pairs for both human and mouse.

In order to search this database, we use the top $k$ genes identified in a given cell cluster using differential expression. Searching this database can be done using a hypergeometric test (or Fisher's exact test) for the overlap between the input gene set and the query gene set (Fisher, 1935), or the probabilistic method described below.

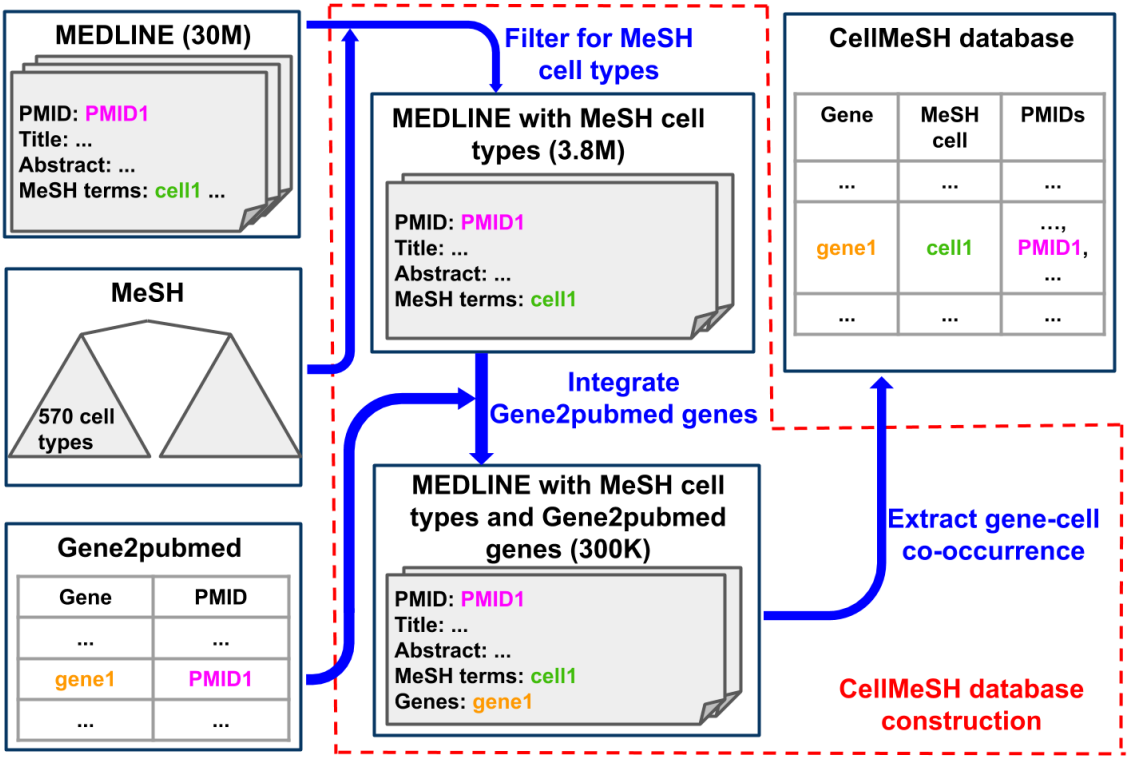



Figure 3.2: **CellMeSH Database Construction.** The construction is species-specific. Here we illustrate the construction for the human database. We start with the 30 million MEDLINE references, and keep the ones containing MeSH cell types (there are 3.8 million such references). We further filter away the references not having human genes in Gene2pubmed, after which 300 thousand MEDLINE references remain. Each remaining MEDLINE reference $p$ contains several MeSH cell types $\{c\}$ and several genes $\{g\}$, and we append $p$ to each $(g, c)$ pair in the final CellMeSH database.

### 3.2.2 Probabilistic query method

There are two major issues with using a literature-derived database. The first issue is publication bias. Some genes or cell types are studied much more than others, which means that there are

more publications and thus more associations containing those genes or cell types. The second issue is noise in the gene-cell type mapping. The CellMeSH database is inherently noisy, as it links genes and cell types at an article level, and the simple fact of an article mentioning a cell type and a gene together does not imply that the gene serves as a marker for the cell type. This leads to potentially spurious associations between genes and cell types.

First, we can address the issue of publication bias by applying TF-IDF (Term Frequency-Inverse Document Frequency) (Wikipedia, 2020c) which is a re-weighting method commonly used in Natural Language Processing (Manning, 2008; Rajaraman, 2011), and by applying column normalization. Let $w_C(g)$ denote the weight, which is the number of co-occurrences of gene $g$ in the cell type $C$. Using TF-IDF transformation, the new weight is given by $w_C(g) \leftarrow w_C(g) \times \log \frac{N_C}{K_g}$ where $N_C$ is the total number of available cell types in the database, and $K_g$ is the total number of cell types with non-zero weights for gene $g$, i.e. $K_g = \sum_C 1_{C:w_C(g)>0}$. TF-IDF addresses the publication bias of genes since the transformation appropriately deweights commonly occuring genes (since, for these genes, $K_g$ is larger). After TF-IDF transformation, the weight is further adjusted by column normalization: $w_C(g) \leftarrow \frac{w_C(g)}{\sum_{g' \in C} w_C(g')}$. Column normalization addresses the publication bias of cell types since the transformation appropriately deweights the genes occurring in common cell types.

We then query the weight-adjusted database using a probabilistic method, which is designed to address the issue of noise from spurious gene-cell associations. Our query method takes input of $w_C(g)$ which is the adjusted weight of gene $g$ in cell type $C$. The method also takes input of a query $Q$ which is a list of genes. The method outputs the database cell types sorted by their significance to the query.

Our probabilistic query method assumes the following generative model for the observed query data (based on which the inference is performed): (1) a cell type is first chosen (with a uniform prior probability) (2) associated with the cell type is a probability distribution on the genes given by $p(g|C)$. A natural model for $p(g|C)$ is to take it to be proportional to the weight $w_C(g)$. (3) However, the previous model ensures that only genes with non-zero weight for the cell type will be present in the query - this need not be the case in our noisy dataset. To model this noise, we assume that with probability $1 - \alpha$ the gene is sampled randomly from the list of

all genes, and with probability $\alpha$ it is sampled from the cell-type specific gene distribution (in experiments, $\alpha$ is fixed as 0.5).

We also denote the total number of genes as $N_g$ and the total number of genes with non-zero weight in cell type $C$ as $K_C$, i.e., $K_C = \sum_g 1_{g:w_C(g)>0}$. Thus the probability of picking a gene from a cell type can be written as follows:

$$P(g|C) = \begin{cases} \alpha \cdot w_C(g) & \text{if } g \in Q \cap C \\ (1-\alpha)\frac{1}{N_g - K_C} & \text{if } g \in Q \cap \bar{C} \end{cases} \tag{3.1}$$

$P(Q|C)$ is the probability that the list of query genes is obtained from a particular cell type $C$. $P(g|C)$ is the probability that we see gene $g$ in the query given that the cell type is $C$. Assuming that each gene is sampled independently, we have $P(Q|C) = \prod_{g \in C} P(g|C)$.

To predict the cell type for a set of query genes (e.g. differentially expressed genes, denoted as $Q$) associated with a cluster, we look at a list of candidate cells (e.g. MeSH cell types) and for each candidate cell $C$ we calculate its log-likelihood score $L(Q|C)$ as follows:

$$L(Q|C) = \log P(Q|C) = \sum_g \log P(g|C) \tag{3.2}$$

Candidate cells are then sorted based on their log-likelihood scores. We can then use maximum likelihood estimation to predict the cell type $\hat{C}^*$ that maximizes our chance of seeing the query:

$$\begin{aligned} \hat{C}^* &= \text{argmax}_C \log P(Q|C) \\ &= \text{argmax}_C \sum_g \log P(g|C) \end{aligned} \tag{3.3}$$

### 3.2.3 Using prior knowledge

In practice, it is possible or even likely that some prior knowledge, for example the cell types expected in a given tissue of origin, is available for a set of queries. Many cell-level annotation methods require this kind of prior knowledge. For example, Seurat (Butler et al., 2018) needs an

existing annotated gene expression profile that contains cell types that are similar to the query cells for label transfer, and Garnett (Pliner et al., 2019) utilizes a manually curated cell-type marker-gene file which includes only a small number of cell types that are expected to cover the queries.

Such information is not typically used in cluster-level annotation methods (Chen et al., 2013; Alavi et al., 2018), but CellMeSH is able to leverage optional prior knowledge to refine the search space for candidate cell types. Specifically, the probabilistic query method can be restricted to a subset of the CellMeSH database consisting of MeSH cell types that belong to one or multiple subtrees within the MeSH hierarchy. For instance, when annotating the PBMC dataset (Zheng et al., 2017), we can limit search to cell types under "Blood Cells" in the MeSH ontology.

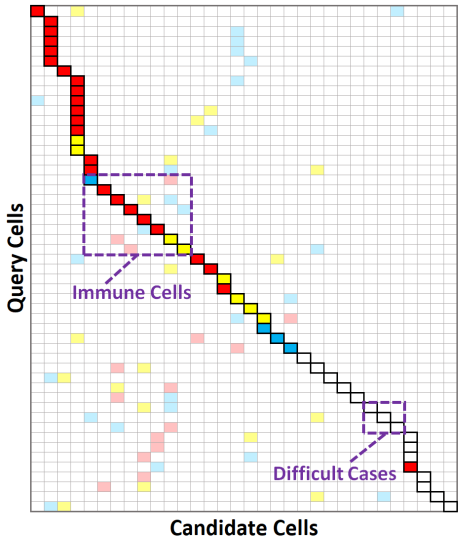
## 3.3 Results

### 3.3.1 Cell type annotation performance

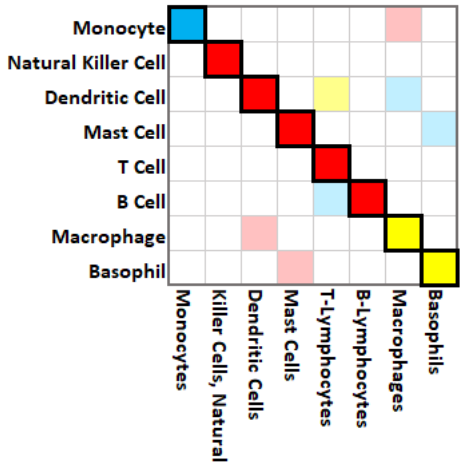
We quantified the cell-type identification performance of CellMeSH for four scRNA-seq datasets with known cell types: two Tabula Muris (TM) datasets (Consortium, 2018), the Mouse Cell Atlas (MCA) dataset (Han et al., 2018), and the human Peripheral Blood Mononuclear Cells (PBMC) dataset (Zheng et al., 2017).

In our evaluation, we used clusters and reference annotations obtained in the original papers. For each cluster, we extracted the top $n = 50$ differentially expressed genes by 1-vs-rest gene expression ratio. These genes are assumed to be marker genes of the reference cell type and are used as a query input for CellMeSH. We then queried CellMeSH using these 50 genes for each cluster and visualized the results using heatmaps that show how well the retrieved MeSH cell types agree with the reference cell type. To validate our results we manually created mappings between the reference cell types (from the original papers) and their corresponding MeSH cell types.

(a) Tabula Muris Droplet




(b) Tabula Muris Droplet - Immune Cells




(c) Top 3 Results for Difficult Cells obtained with and without using prior knowledge.

| Prior | Rank | Promonocyte | Granulocyte | Alveolar Macrophage |
|---|---|---|---|---|
| Not used | 1 | Neutrophils | Neutrophils | Macrophages |
| | 2 | T-Lymphocytes | Myeloid Cells | Macrophages, Peritoneal |
| | 3 | Macrophages | Monocytes | Dendritic Cells |

| Prior | Rank | Promonocyte | Granulocyte | Alveolar Macrophage |
|---|---|---|---|---|
| Used | 1 | T-Lymphocytes | Myeloid Cells | Macrophages |
| | 2 | Macrophages | Monocytes | Dendritic Cells |
| | 3 | Bone Marrow Cells | Granulocytes | Macrophages, Alveolar |

Correct Candidate | Supercategory of Correct Candidate | Subcategory of Correct Candidate | Related Candidate

Figure 3.3: **Annotation Results for the Tabula Muris Droplet Dataset. (a)** Annotation heatmap for queries of all cells. The y-axis represents the query cells $\{r\}$ and x-axis represents the candidate cells $\{c\}$. A bolded border for entry $(x = c, y = r)$ indicates that $c$ is the correct candidate for query $r$. The red, yellow or blue color indicates $c$ has rank 1, rank 2, or rank 3 among retrieved results for $r$; the colors are shown lighter if $c$ is not the correct candidate. **(b)** Annotation heatmap for queries of immune cells. The correct candidate cells are within the top 3 retrieved results for all queries. **(c)** The first table shows the top 3 results (prior not used) for several query cells corresponding to the uncolored border-boxes in the heatmap. For instance, for the query Granulocyte, the correct candidate "Granulocytes" is ranked 5-th (not shown) among the retrieved results. The top 2 results "Neutrophils" and "Myeloid Cells" are accurate to a certain extent because they are the subcategory and supercategory of "Granulocytes" in the MeSH tree. In the second table, by using prior knowledge we obtain better results for these queries.

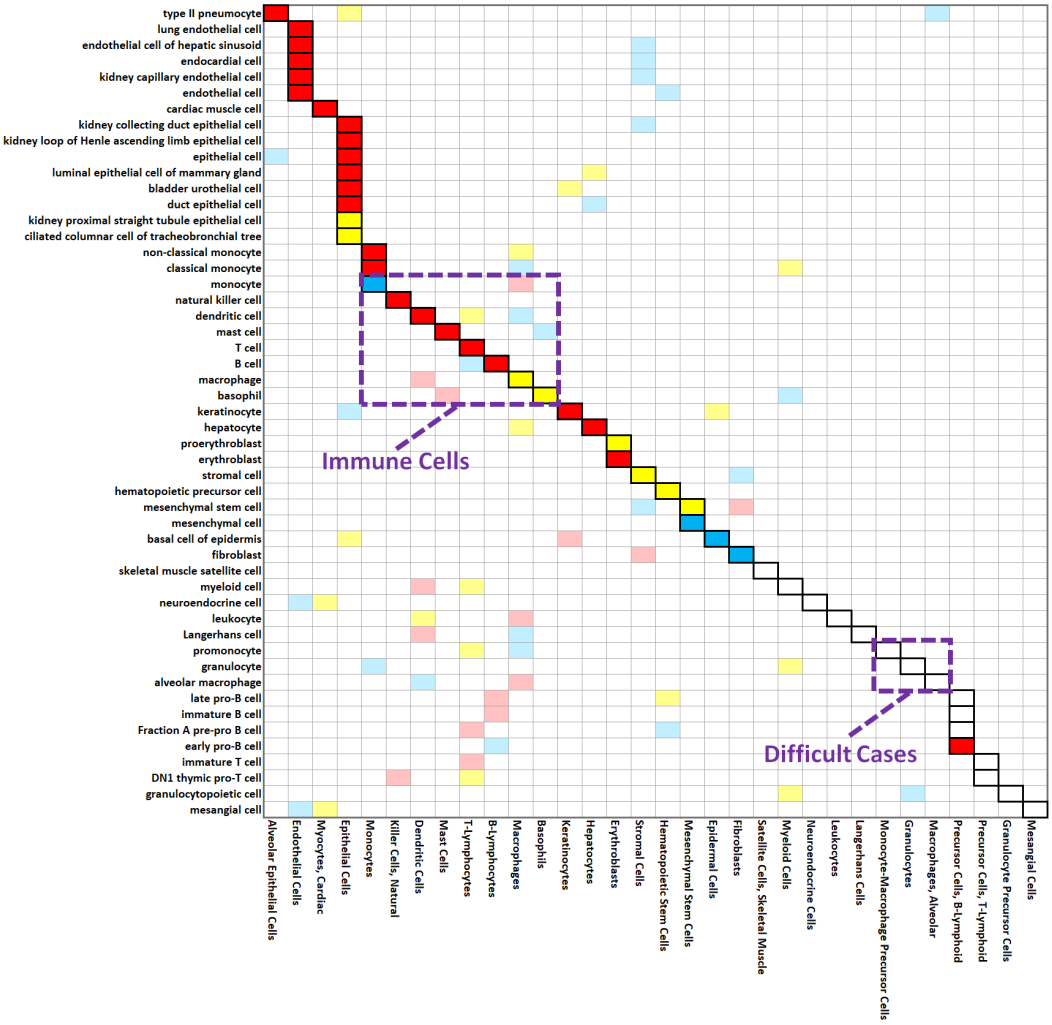



Figure 3.4: **Annotation Heatmap on the TM-Droplet Dataset.** The annotation heatmap has its y-axis representing the query cells $\{r\}$ and x-axis representing the candidate cells $\{c\}$. A border-box for entry $(x = c, y = r)$ indicates $c$ is the correct candidate for query $r$. The red, yellow or blue color indicates $c$ has rank 1, rank 2, or rank 3 among retrieved results for $r$; the colors turn lighter if $c$ is not the correct candidate.

### Tabula Muris datasets

This dataset contains cells that were captured from 20 different tissues in 3-month-old mice and that were clustered into 99 annotated cell types. The dataset has two subsets, with cells captured using a microfluidic-droplet method (denoted as the TM-Droplet dataset, containing $55,656$ cells) or by cell sorting (denoted as the TM-FACS dataset, containing $44,949$ cells). We will mostly discuss the TM-Droplet dataset (Figure 3.3) but the results for TM-FACS are

similar, as shown in Figure 3.5.

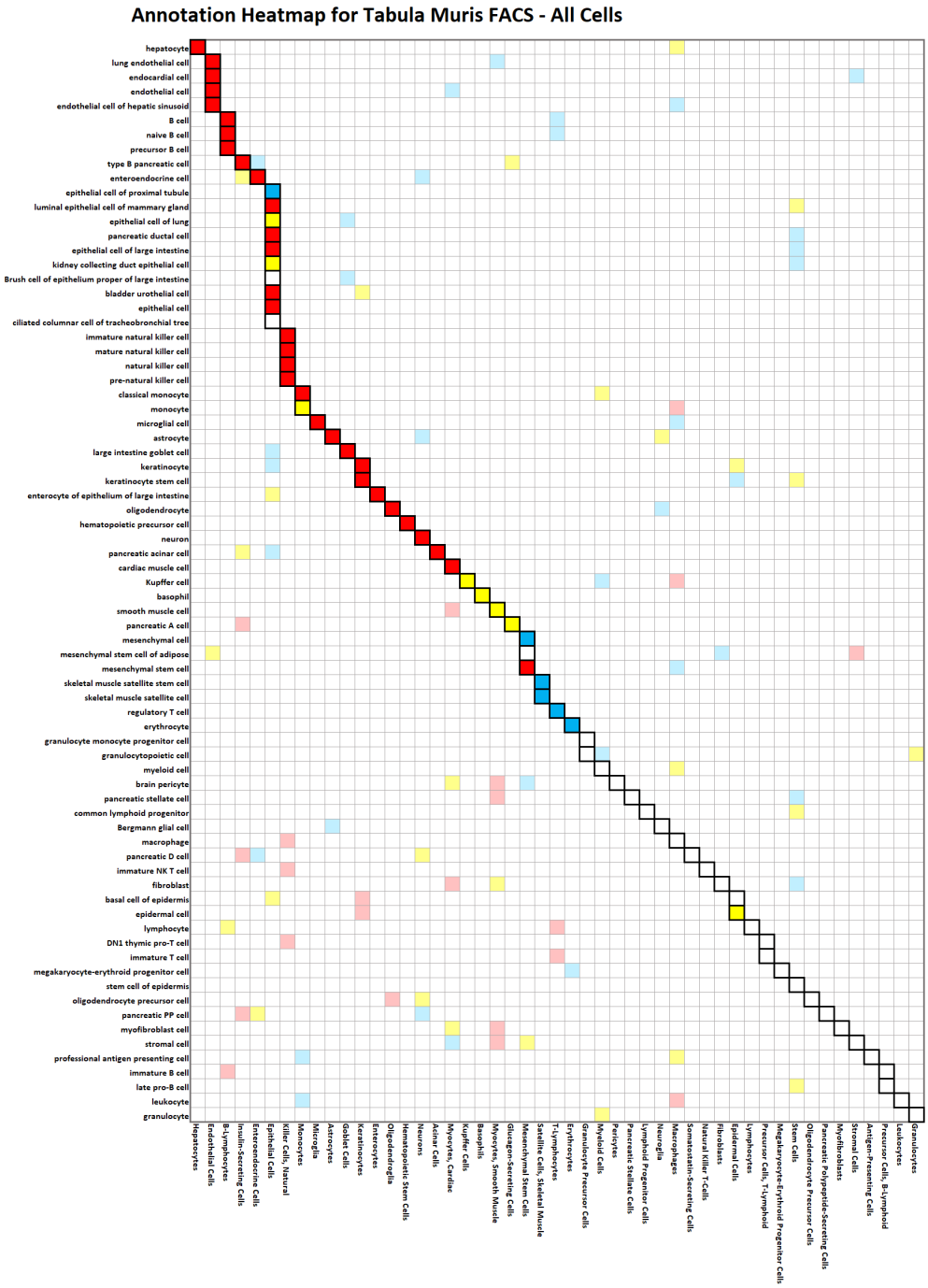



Figure 3.5: **Annotation Heatmap on TM-FACS Dataset.** The annotation heatmap has its format similar to Fig 3.4.

Annotation results for the entire TM-Droplet dataset are summarized in the heatmap shown

in Figure 3.3. See Figure 3.4 for detailed cell type names. The diagonal bordered boxes, indicating the correct annotations, are mostly filled with red, yellow or blue colors that indicate the top 3 retrieved cell types, which demonstrates the effective annotation ability of CellMeSH. To see this more clearly, in Figure 3.3(b) we focus on the annotation heatmap for only the immune cell types. For that subset, the correct result is contained within top 3 candidates for all queries.

The bordered boxes that form a vertical column in Figure 3.3(a) are the result several true cell types being mapped to the same MeSH cell term due to the limited resolution of the MeSH cell types. For example, both Luminal Epithelial Cell of Mammary Gland and Kidney Collecting Duct Epithelial Cell etc are mapped to "Epithelial Cells".

Bordered boxes without color in Figure 3.3(a) imply that the correct candidate may not exist due to a limit in the coverage of the MeSH cell types, or, more likely, that it is not within the top 3 retrieved results due to the noise in CellMeSH database. Still, even then, the CellMeSH query results provide useful insights into the true cell types, as illustrated in Figure 3.3 (c, prior not used). For instance, the query Promonocyte does not have an exact same MeSH term; the closest term we could manually match is "Monocyte-Macrophage Precursor Cells". For the query Granulocyte, the correct MeSH term "Granulocytes" is rank-5 (not shown) in the retrieved results. However the top 2 results "Neutrophils" and "Myeloid Cells" are respectively the subcategory and supercategory of "Granulocytes" in the MeSH tree. Similarly, for the query Alveolar Macrophage, the correct MeSH candidate "Macrophages, Alveolar" actually is rank-5 (not shown). However, the top rank result "Macrophages" is also close as it is a supercategory of "Macrophages, Alveolar".

Finally, we have conducted a quantitative analysis for the whole dataset showing that many top results are in fact closely related to the correct cell type in the MeSH tree (Table 3.2).

By leveraging prior knowledge, in particular restricting the search to cell types contained in the Tabula Muris dataset, the CellMeSH annotation is further improved, as illustrated in Figure 3.3 (c, prior used). Specifically, the 3rd result for the query Promonocyte now is "Bone Marrow Cells". Promonocytes are cells arising from a Monoblast (in Bone Marrow (Wikipedia, 2020a)) and developing into a Monocyte (Wikipedia, 2020b) and thus seem relevant to the query.

Table 3.2: **Comparison between original evaluation and relaxed evaluation.** For the relaxed evaluation, we consider hierarchically related (grand parent, parent, child, grand child) cell types of the correct cell type as also correct.

| Dataset | num queries | top-1 (original) | top-3 (original) | top-1 (relaxed) | top-3 (relaxed) |
|---|---|---|---|---|---|
| PBMC | 10 | 0.6 | 0.9 | 0.8 | 1.0 |
| Droplet | 51 | 0.588 | 0.882 | 0.686 | 0.922 |
| FACS | 76 | 0.526 | 0.724 | 0.645 | 0.882 |
| MCA | 193 | 0.472 | 0.751 | 0.579 | 0.772 |

Moreover, for the queries Granulocyte and Alveolar Macrophage the correct MeSH cell types are now contained within top 3 results.

#### MCA and PBMC datasets

These datasets show similar annotation performance to the Tabula Muris datasets, as shown in Figures 3.6 and 3.7. Again, bordered boxes on the diagonal are mostly filled with red, yellow or blue colors, indicating the top 3 retrieved cell types match the reference labels.

### 3.3.2 Comparison with cluster-level annotation methods

We compared CellMeSH to several existing cluster-level methods (Zhang et al., 2018b; Franzén et al., 2019; Chen et al., 2013; Fisher, 1935; Hänzelmann et al., 2013; Alavi et al., 2018) (Table 3.1) using top-k ($k = 1, 3$) accuracy for the three mouse datasets (TM-Droplet, TM-FACS and MCA). We use the mouse datasets instead of the human PBMC dataset because they contain many more queries (51, 76 and 193 queries respectively) than the PBMC dataset (only 10 queries), and therefore can show more reliable quantitative trends. Some queries (e.g. Cell in Cell Cycle) are excluded as there are no matching candidates in any databases (e.g. CellMeSH, CellMarker, PanglaoDB etc).

We queried CellMeSH with the previously extracted marker genes for each cluster, and calculated the top-k accuracy (percentage of queries for which the correct cell type is in the top k results) for each dataset. We similarly queried existing methods and obtained top-k results

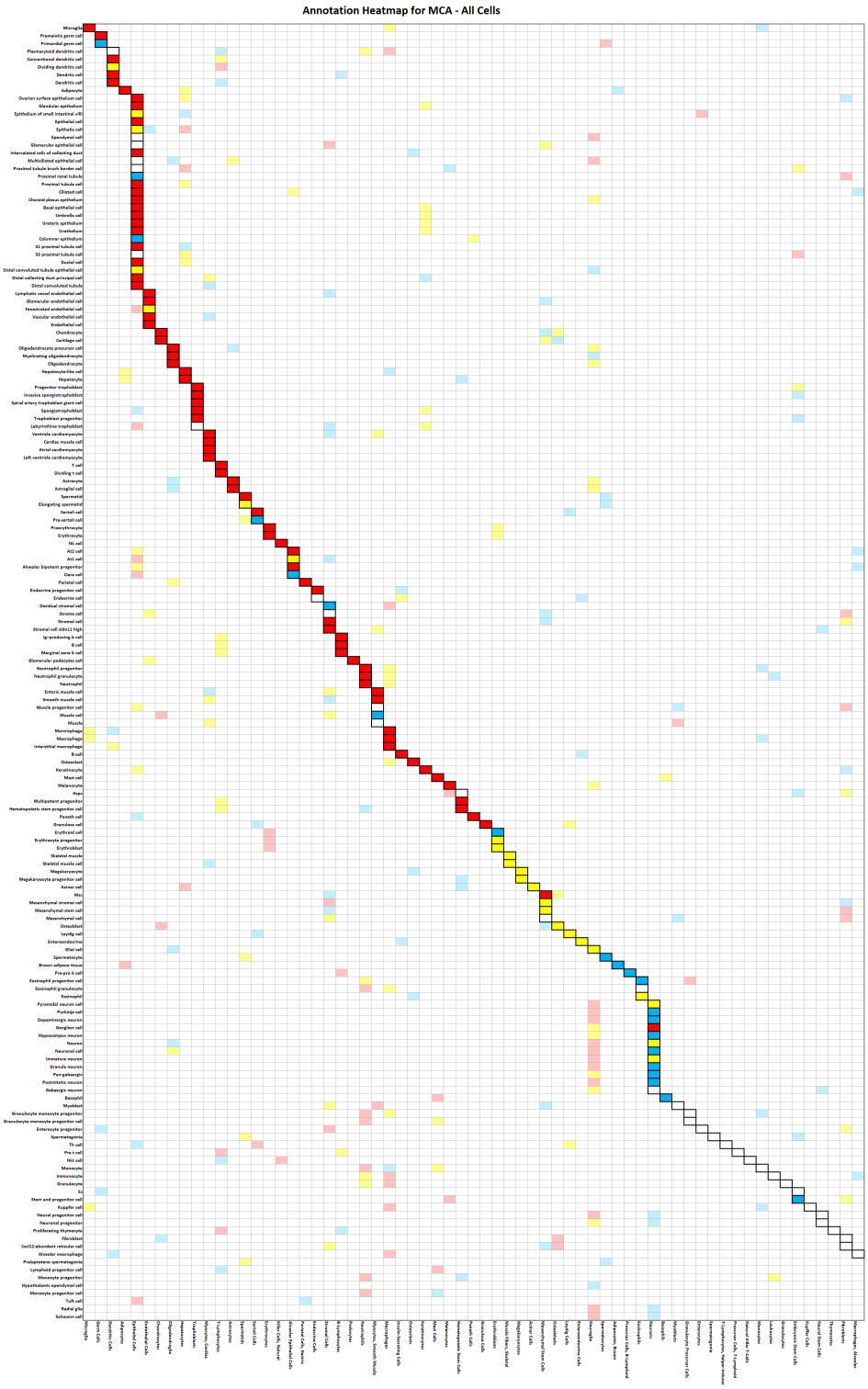



Figure 3.6: **Annotation Heatmap on MCA Dataset.** The annotation heatmap has its format similar to Fig 3.4.

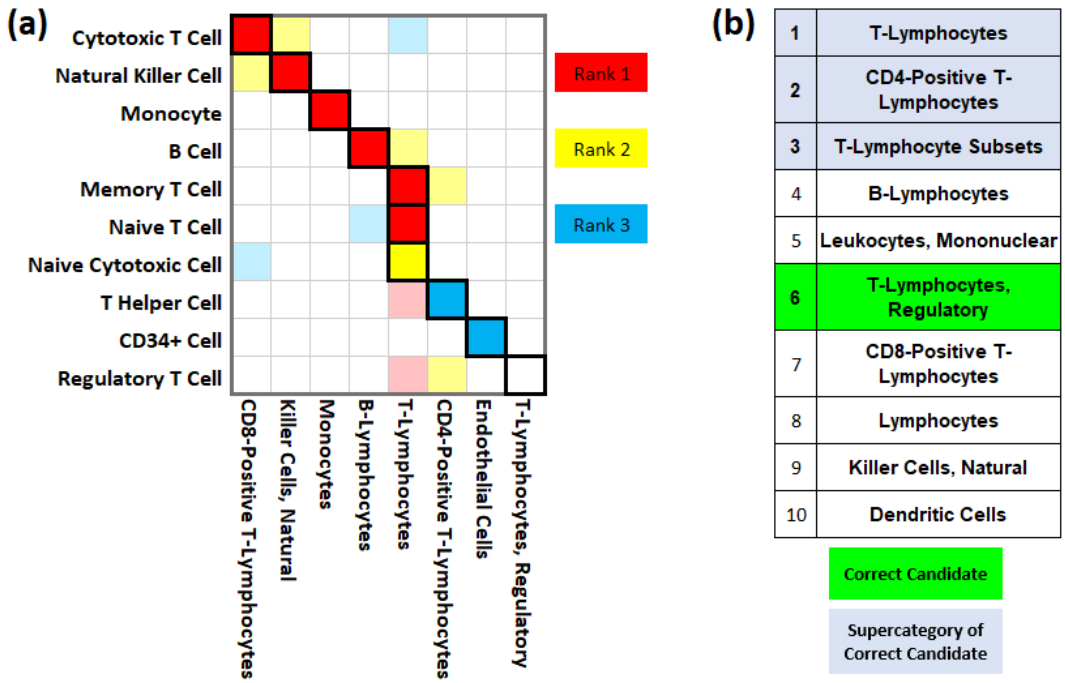



Figure 3.7: **Annotation Results for Human PBMC Dataset.** **(a)** is the annotation heatmap where y-axis represents the query cells $\{r\}$ and x-axis represents the candidate cells $\{c\}$. A border-box for entry $(x = c, y = r)$ indicates $c$ is the correct candidate for query $r$. The red, yellow or blue color indicates $c$ has rank 1, rank 2, or rank 3 among retrieved results for $r$; the colors are shown lighter if $c$ is not the correct candidate. For example, for the query cell B Cell, the red border-box at "B-Lymphocytes" indicates "B-Lymphocytes" is rank 1 in the retrieved results, and it is also the correct result. **(b)** shows the top 10 results for the query Regulatory T Cell, which has an uncolored border-box in the heatmap. This is because its correct candidate cell "T-Lymphocytes, Regulatory" is not among the top 3 (instead it is rank 6) in the retrieved results. In fact the top 3 results are closely related to the correct candidate as they are supercategories of the correct candidate in the MeSH tree hierarchy.This shows that even in cases where our retrieved cell type is not exactly matching the correct cell type, CellMeSH returns reasonable results.

for each dataset. The query results of existing methods could derive from a different ontology and therefore contain different cell type names from the MeSH terms. In order to calculate the accuracy of these methods, we thus manually created mappings between the given query cell types and the candidate cell types from other ontologies.

### Overall top-k accuracy gain

In Figure 3.8 (a)(b), we first compare CellMeSH with two other web servers: Enrichr (Chen et al., 2013) and scQuery (Alavi et al., 2018) as their input and output formats are most similar

to CellMeSH (Table 3.1).


Figure 3.8: **Comparison of CellMeSH to other methods.** Each bar plot has y-axis as the top-k (k=1 or 3) accuracy (%) and is grouped by different mouse datasets, and for each group, we show the top-k accuracy of different methods. Top-k accuracy refers to the percentage of queries where one of the candidate cells among the top k retrieved cells is accurate. **(a)(b)** Comparison of CellMeSH to other systems. The CellMarker database is queried using the hypergeometric test. **(c)(d)** Comparison of the probabilistic query method to other query methods. We use the CellMeSH database with all query methods. **(e)(f)** Comparison of the CellMeSH database to other databases. CellMeSH and CellMarker are both non-binary gene-cell matrices and therefore we use probabilistic method to query, whereas PanglaoDB is a binary gene-cell matrix and we use hypergeometric test to query. **(g)(h)** Comparison of the default version of CellMeSH that uses a search space of all 570 MeSH cell types to CellMeSH with prior knowledge. Correct MeSH cell types and MeSH tree descendants from Tabula Muris dataset are used as prior information for TM-Droplet and TM-FACS queries, while correct MeSH cell types and MeSH tree descendants from MCA dataset are used as prior for MCA queries.

We were unable to compare to the web servers of PanglaoDB (Franzén et al., 2019), CellMarker (Zhang et al., 2018b) and CellFinder (Stachelscheid et al., 2013) in an automatic way. Instead, we downloaded the existing CellMarker (Zhang et al., 2018b) database and queried it using the hypergeometric (or Fisher's exact) test (Fisher, 1935).

CellMeSH provides the most accurate results for all three mouse datasets. Specifically, in the TM-Droplet dataset, CellMeSH achieved top-1 accuracy of 58.8%, meaning that in 58.8% of queries, the first retrieved candidate cell type is correct. The top-1 accuracy is 15.7% higher than that of the second best method, Enrichr. This is to be expected because the Enrichr cell types come from the Mouse Gene Atlas (MGA) database (Su et al., 2004), which contains only 96 cell types. Besides, some of the MGA cell types (such as "Heart", "Kidney", and "Stomach" etc) actually refer to organs.

We find that CellMeSH has higher coverage and resolution than the other methods including Enrichr. For example, for query Classical Monocyte, while CellMeSH returns "Monocytes" as the first candidate, there is no monocyte term covered in MGA and Enrichr returns its first result as "Macrophage Bone Marrow 6hr LPS". For query Duct Epithelial Cell, while CellMeSH returns "Epithelial Cells" as the first result, Enrichr returns the organ terms "Bladder", "Liver" and "Stomach" as its top 3 results The top-3 accuracy of CellMeSH further increases to 88.2% (this implies that 88.2% of queries get at least one of the top 3 results correct), which is 31.3% higher than that of Enrichr.

CellMeSH also consistently outperforms other methods on the other two datasets. Its top-1 (or top-3) accuracy is 3.9% (or 11.9%) higher in the TM-FACS dataset, and 6.4% (or 22.7%) higher in the MCA dataset, than the second best method, Enrichr.

**Top-k accuracy gains from probabilistic method**

Both the CellMeSH database and the probabilistic query method contribute to the overall top-k accuracy gains of CellMeSH. To isolate the contribution of the probabilistic query method to the overall CellMeSH performance, we compared it to the more established hypergeometric test (Fisher, 1935) and GSVA (Hänzelmann et al., 2013) that are suggested in (Diaz-Mejia et al., 2019), by querying the same CellMeSH database for the three mouse datasets.

As shown in Figure 3.8 (c)(d), the probabilistic method performs uniformly better than other methods. Compared to the best performance of GSVA and the hypergeometric test, the probabilistic query method has a top-1 accuracy gain of 13.7%, 6.6% and 7.3% in the TM-Droplet, TM-FACS and MCA datasets respectively. The numbers for top-3 accuracy gain are 19.6%, 3.9% and 8.8%.

**Top-k accuracy gains from CellMeSH database**

To isolate the contribution of the CellMeSH database (as opposed to the query method), we compare the performance of alternative databases using the same query method here.

We prepared gene-cell co-occurrence matrices by aggregating the cell-type marker-genes files from PanglaoDB (Franzén et al., 2019) and CellMarker (Zhang et al., 2018b), both of which are manually compiled from the literature. The resulting mouse gene-cell co-occurrence matrix of CellMarker has 7208 genes and 313 cell terms, where the matrix value for a particular gene-cell pair represents the number of records (i.e. publications) where they co-occur. The resulting human gene-cell co-occurrence matrix of CellMarker has 8973 genes and 364 cell terms. These matrice are mostly sparse, with $< 1\%$ positive counts, and maximum counts below 10. The resulting PanglaoDB database, a binary gene-cell matrix, has 4679 genes (for both mouse and human) and 178 cell types. There are 8230 (1%) non-zero entries.

We then compared the CellMeSH database to these two databases. We queried the CellMeSH and CellMarker databases using the probabilistic query method, since these databases are count-valued matrices, which can be handled effectively by the probabilistic query method, as illustrated in Figure 3.8(c)(d) and in Figure 3.9. For PanglaoDB, the query method is the hypergeometric test, since the database is essentially a binary matrix.

As Figure 3.8 (e)(f) illustrates, using the CellMeSH database achieves a higher accuracy than using the PanglaoDB and CellMarker databases. Compared to the best performance out of PanglaoDB and CellMarker databases, the CellMeSH database has a gain in top-1 accuracy of 21.6%, 3.9% and 0.6% in the TM-Droplet, TM-FACS and MCA datasets respectively. The numbers for the top-3 accuracy gain are to 21.6%, 10.5% and 5.7%.

**Impact of prior knowledge**

Our evaluations in Figure 3.8 (a-f) for CellMeSH do not use prior knowledge and the search space covers all of the 570 MeSH cell types. In contrast, many existing methods, especially the ones that predict cell types for single-cells, leverage prior knowledge so that the reference gene expression profiles (Butler et al., 2018; Alavi et al., 2018; Kiselev et al., 2018; Aran et al., 2019; Hou et al., 2019; Ma and Pellegrini, 2019; Tan and Cahan, 2019) or manually-curated cell-type marker-gene files (Pliner et al., 2019; Zhang et al., 2019b) contain a more constrained and related set of candidate cell types. CellMeSH can similarly use such prior knowledge to reduce the search space for better annotation accuracy. For example, we selected the correct MeSH cell types for the Tabula Muris datasets and their descendant cell types in MeSH ontology to be the prior knowledge for the TM-Droplet and TM-FACS queries. Similarly we selected the correct MeSH cell types for the MCA dataset and their descendant cell types as the prior knowledge for the MCA queries. As shown in Figure 3.8 (g)(h), utilizing prior knowledge improves Top-1 accuracies by 11.8%, 11.9% and 4.1% for TM-Droplet, TM-FACS and MCA respectively. The gain is smaller for MCA because it has the largest number of queries, so that the search space is not reduced significantly.

**Impact of gene number**

Our evaluations so far have been using the top $n = 50$ differentially expressed genes as the marker genes for each query cell type, as the performance tends to peak around $n = 50$ for most of the methods (see Figures 3.10); a smaller number of genes may not provide sufficient information, and a larger number of genes may increase noise, both of which could reduce annotation performance. If we select the optimal number of genes for each method (e.g. different settings of the database and the query method), the CellMeSH database together with the probabilistic query method still consistently outperforms all other methods for all datasets.

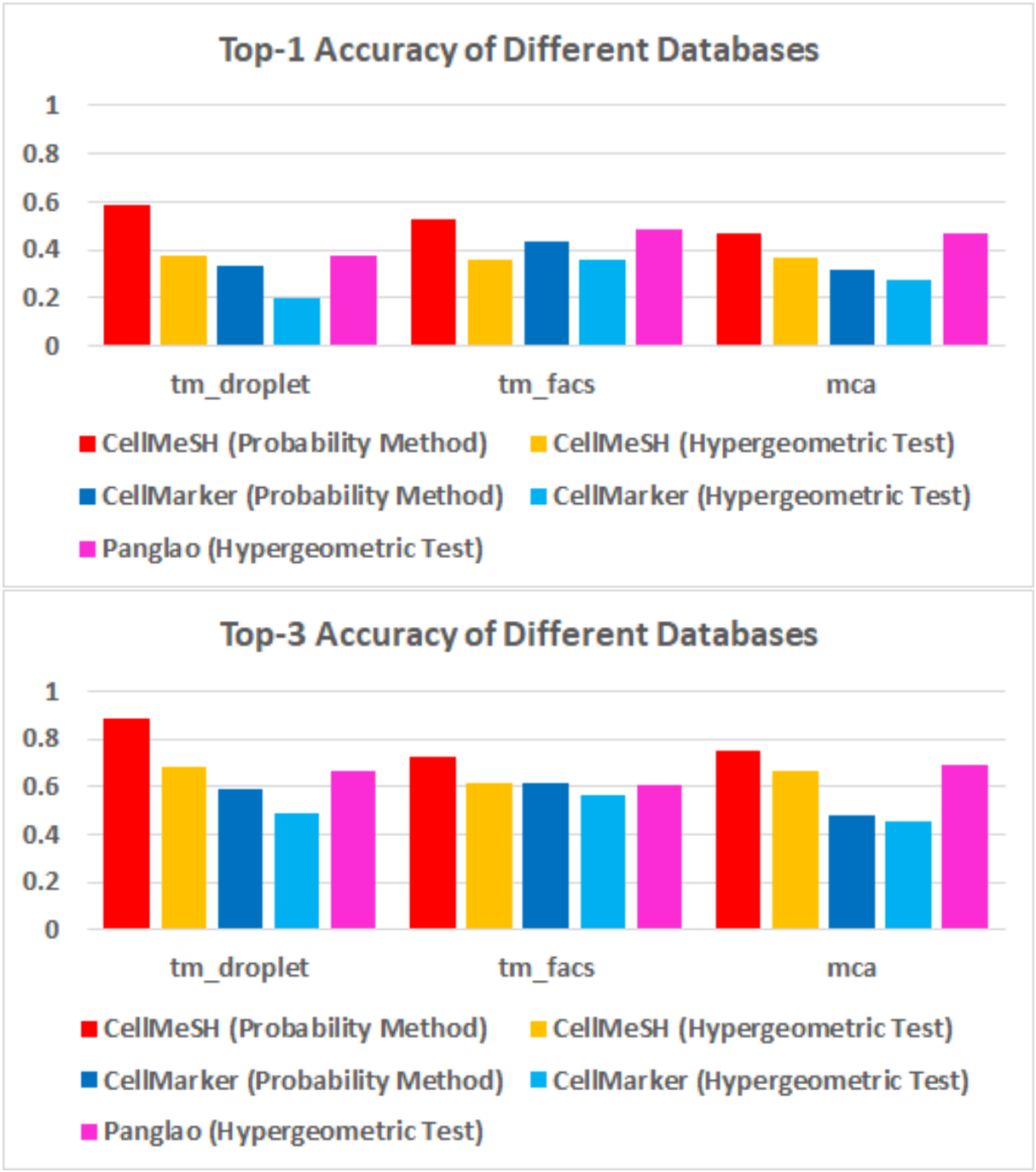



Figure 3.9: **Top-k Accuracy on Different Databases.** Each subplot has its y-axis as top-k ($k = 1, 3$) accuracy and x-axis as different datasets. Bars of different colors represent different approaches to identify cell-types. When we fix query method as hypergeometric test, PanglaoDB results (pink) actually achieves better performance than CellMeSH (orange) and CellMarker (light blue). As CellMeSH and CellMarker databases are essentially non-binary matrice and contains more noise, using probabilistic query method will better extract the information. CellMeSH (red) eventually performs the best by utilizing the probabilistic query method.


Figure 3.10: **Top-k Accuracy versus Number of Genes on TM-Droplet Dataset.** Each subplot has its y-axis as the top-k ($k = 1, 3$) accuracy and x-axis as the number of marker genes (e.g. $n$) of each query cell. The curves of different colors correspond to different approaches to identify cell-types. Most approaches peak around $n = 50$.


Figure 3.11: **Top-k Accuracy versus Number of Genes on TM-FACS Dataset.** Each subplot has its y-axis as the top-k ($k = 1, 3$) accuracy and x-axis as the number of marker genes (e.g. $n$) of each query cell. The curves of different colors correspond to different approaches to identify cell-types. Most approaches peak around $n = 50$.

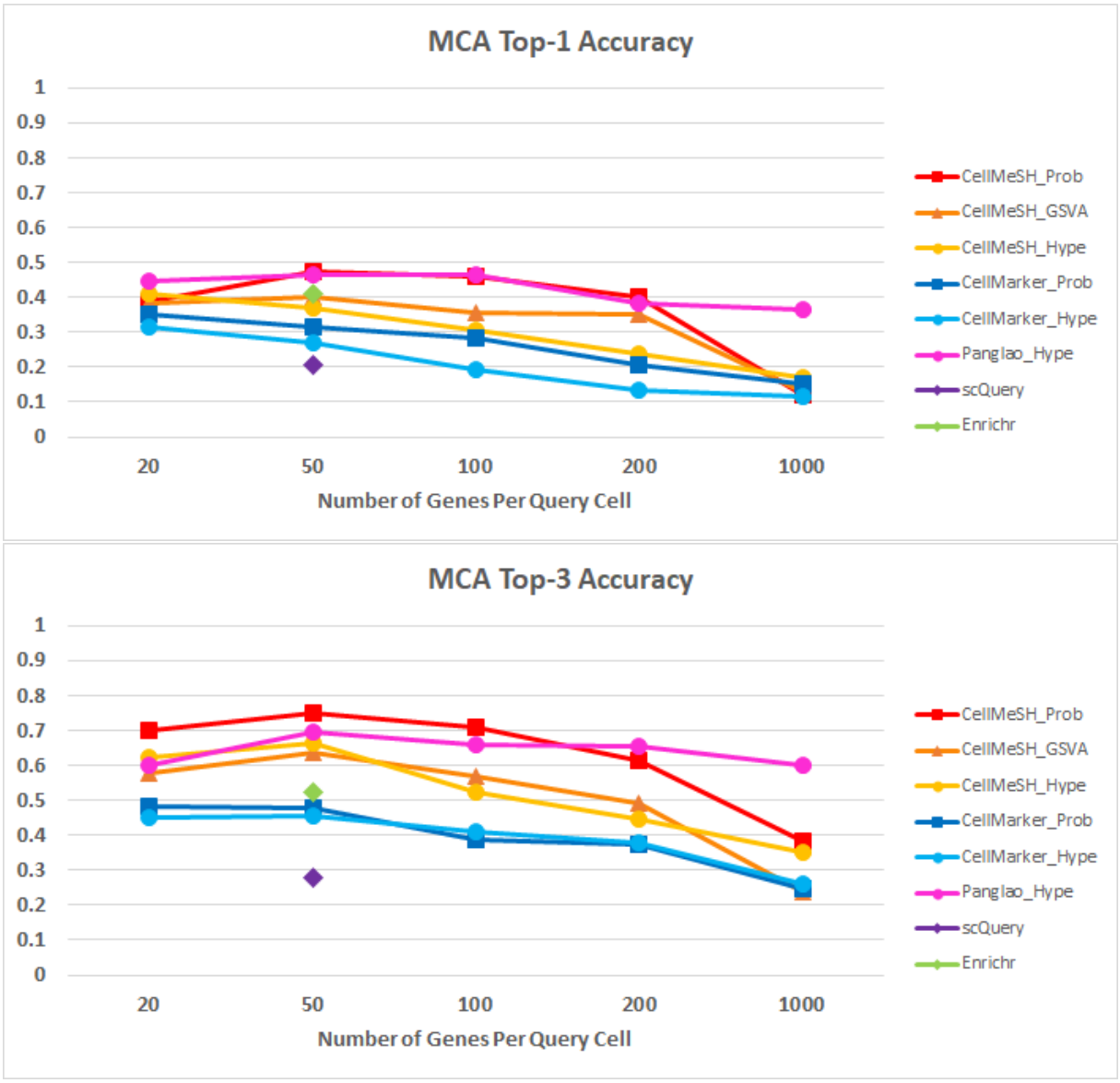



Figure 3.12: **Top-k Accuracy versus Number of Genes on MCA Dataset.** Each subplot has its y-axis as the top-k ($k = 1, 3$) accuracy and x-axis as the number of marker genes (e.g. $n$) of each query cell. The curves of different colors correspond to different approaches to identify cell-types. Most approaches peak around $n = 50$.

### Impact of overall cell population

CellMeSH suggests cell type annotations for a cluster based on its differentially expressed genes (DEGs). DEGs are obtained by comparing gene expression for cells in the cluster of interest to gene expression of cells in the remaining clusters. In principle, DEGs associated with a cluster can thus change depending on what other cells are present in a dataset. To test whether such context changes result in different cell types assignments, we compared CellMeSH predictions for the TM-Droplet immune cells in the context of all other cells with the predictions when only the immune cells are present. The predictions are mostly consistent between the different contexts. We hypothesize that if the DEGs are true markers, they should show up in most cases since our analysis looks at a large number (e.g. 50) of genes.

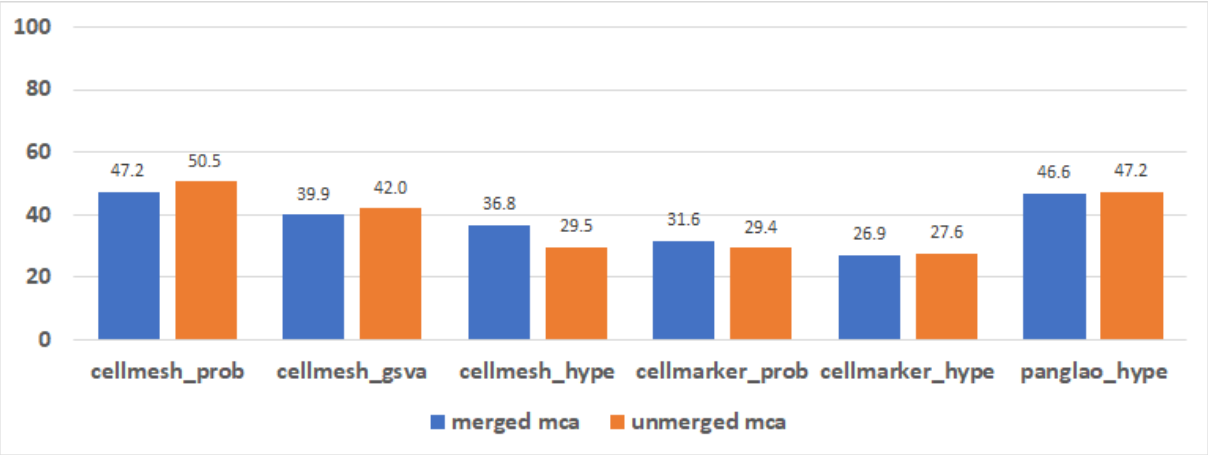
### Impact of clustering resolution

The CellMeSH annotation can be affected by clustering resolution (i.e. whether a set of cells are considered as one cluster or multiple clusters).

Using the MCA dataset, we tested how our classification accuracies are affected by clustering resolution. The MCA dataset has very fine-grained clusters; there are many smaller clusters (defined by specific gene markers) that belong to a single named cell type. In Figure 3.13, we compare the performance of various cluster-level cell type identification methods on both this fine-grained clustering, and a coarser clustering where all the clusters for a given named cell type were merged. The results were similar for both resolutions; in both settings, CellMeSH provided the best accuracy compared to other methods.

We have also explored this issue in our UNCURL-App web server which enables end-to-end cluster-level cell type annotation that integrates CellMeSH to automatically assign cell types to clusters. We found that a human-in-the-loop reclustering (either to split a large cluster or to merge several small clusters) is necessary for further improved annotation (see the following chapter for more details).

(a) Top-1 Accuracy




(b) Top-3 Accuracy

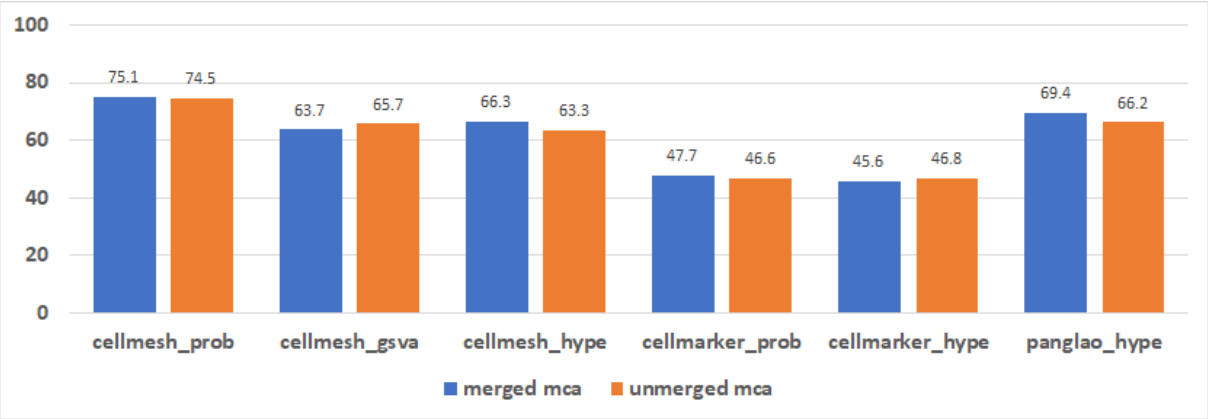



Figure 3.13: **Comparing top-k accuracy using merged and unmerged MCA queries.** We collapsed the original 840 MCA cell-types (unmerged queries; e.g. $r1$ =Alveolar Macrophage_Pclaf high (Lung), $r2$ =Alveolar Macrophage_Ear2 high (Lung)) into the 204 cell-types (merged queries; e.g. $r$ =Alveolar Macrophage). We removed some cell-types which have no equivalent candidate cell-types across any database, resulting 671 unmerged queries and 191 merged queries. (a) compares the merged queries (blue) and the unmerged queries (orange) in terms of top-1 accuracy. The y-axis represents the top-1 accuracy in percentage, and the x-axis represents different combinations of databases (CellMeSH, CellMarker, PanglaoDB) and query methods (Probablistic, GSVA, Hypergeometric test). Using merged queries, CellMeSH (the blue cellmesh_prob bar) achieves 47.2% top-1 accuracy, indicating 47.2% of the queries have their first retrieved result as correct. Overall, the merged queries and unmerged queries have very similar performance. Note that the gain of the best CellMeSH (cellmesh_prob) over the second best PanglaoDB (panglao_hype) is 0.6% (= 47.2% − 46.6%) by using merged queries, which is actually less than 3.3% (= 50.5% − 47.2%) by using unmerged queries. (b) is similar to (a) and compares the merged queries (blue) and the unmerged queries (orange) in terms of top-3 accuracy.

### 3.3.3 Comparison with cell-level annotation methods

Next, we compared CellMeSH to Seurat (label transfer) (Butler et al., 2018) and SingleR (Aran et al., 2019), two annotation methods that use existing annotated scRNA-seq (or bulk RNA-seq)

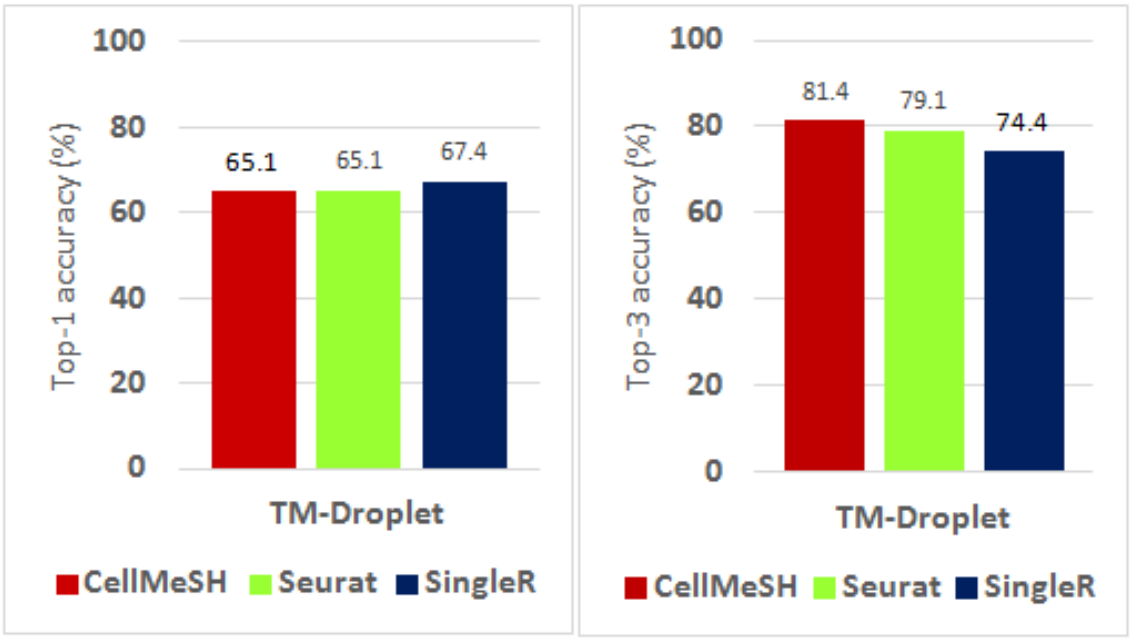



Figure 3.14: Comparison of CellMeSH to cell-level annotation methods. To compare CellMeSH with Seurat and SingleR we use the TM-Droplet dataset as the query. CellMeSH uses the TM-FACS labels as prior knowledge, and Seurat and SingleR use the TM-FACS annotated gene-cell expressions as the reference dataset. When comparing annotation results at cluster-level, all methods show comparable performance.

datasets as a reference, and assign the cell types from existing datasets to query cells that have similar gene expressions (see Table 3.1). For this comparison, we used the TM-Droplet dataset as the query dataset, and TM-FACS as the reference, since they share most cell types but have very different sequencing methodologies, a common situation in practical applications.

For CellMeSH, we grouped the TM-Droplet cells into clusters based on the reference annotation. For each cluster, CellMeSH assigned cell types by querying the database where only the TM-FACS MeSH labels were considered as priors. For Seurat and SingleR, we first individually label all of the TM-Droplet cells using the annotated gene-cell expression profile from TM-FACS. We then grouped the TM-Droplet cells into clusters based on their original annotation. The labels for a cluster are the labels for the cells in the cluster, sorted by number of assigned cells. For example, the top assigned label for a given cluster would be the label that occurred most frequently for the cells in that cluster. There are 43 queries that were assigned cell types by all methods (Seurat, SingleR and CellMeSH).

The results are summarized in Figure 3.14. All three methods have very similar top-1 accuracy. We note that although top-1 accuracy for Seurat and CellMeSH are the same, the queries with correct results are different. More importantly, even without prior knowledge of TM-FACS labels, default CellMeSH still reaches top-1 accuracy of 58.8%. An initial CellMeSH-based annotation without prior knowledge could thus in principle be used to select a dataset that is similar to the query dataset and that can then be used for improving the annotation.

### 3.3.4 CellMeSH web server and API

CellMeSH has a stand-alone web server (Zhang and Mao, 2020), which is able to take in a list of marker genes and an optional choice of known MeSH cells as prior knowledge, and returns a ranked list of predicted MeSH cell types, together with the supporting genes and PubMed articles for further reference (Figure 3.1). The web server also provides options to use other databases (e.g. CellMarker) and query methods (e.g. GSVA, hypergeometric test).

We have open-sourced the CellMeSH database and the probabilistic query method (Mao and Zhang, 2020) as a Python library.

We have also utilized the CellMeSH API to annotate the clusters generated by Seurat (Satija et al., 2015) for the PBMC dataset.Specifically here Seurat first takes input of a gene-cell expression matrix and outputs cell clusters. CellMeSH then annotates each cluster based on its differentially expressed genes produced by Seurat. This demonstrates CellMeSH's potential to complement existing tools and avoid manual cell assignment.

We have also integrated CellMeSH API into our developed web server UNCURL-App (Zhang et al., 2020) for interactive scRNA-seq data analysis, which combines data preprocessing, dimensionality reduction, clustering, differential expression analysis, cell-type annotation and interactive re-clustering into an online graphical user interface. This will be described in the following chapter.

## 3.4 Discussion

We have developed CellMeSH, a method with accompanying web server and API to identify cell types directly from the literature, in order to make the scRNA-seq analysis more convenient.

CellMeSH is similar to gene set-enrichment methods (e.g. Enrichr, GSVA, Hypergeometric test) in terms of input and output formats, but several aspects make CellMeSH different from them. First, existing enrichment tools either focus on only query method (GSVA, Hypergeometric test) or prepare database in a limited way (Enrichr) by collecting the gene-ontology (e.g. diseases, tissues, cell types etc) relations manually from a small set of publications, which is difficult to scale. Second, existing enrichment tools assume the gene-ontology relations have no noise (e.g. no spurious gene-cell relation), whereas the CellMeSH query method is designed to handle spurious gene-cell associations, which is necessary as a database scales automatically and introduces noise (Fig 3.8 (c,d)). Lastly, existing enrichment tools are not optimized for the scRNA-seq annotation problem to the same extent as CellMeSH. For instance, CellMeSH adopts a parameter of top 50 genes for its prediction, and it optionally takes prior knowledge (Fig. 3.1, Section 3.2.3) as input and explores the MeSH ontology hierarchy to reduce the search space in order for better prediction. CellMeSH has also been experimented on various scRNA-Seq datasets to show it is able to provide better annotations than using existing enrichment methods directly (Fig. 3.8 (a-d)).

Experiments on both human and mouse scRNA-seq datasets demonstrate CellMeSH's superior cell-type identification performance. Nevertheless, there are still several limitations with CellMeSH.

In particular, the cell-type annotations provided by MeSH terms are somewhat coarse, and might not be enough to represent a comprehensive listing of all fine-grained cell types and subtypes present in certain tissues. Specifically, it is possible that the ground truth cell type (e.g. a subtype of CD4+ T cells) for a cluster does not exist in the MeSH ontology, and consequently CellMeSH will not be able to predict that exact cell type but will instead return a similar cell type assignment.

This is an issue that is common among database-driven cell type prediction methods. To resolve this issue, it should be possible to extract fine-grained gene cell-type relationships from the literature or supplemental materials.

Previously the work of CellMarker approached this direction but it required manually efforts to survey the literature and pick up cell types and their marker genes. We believe CellMeSH is

the first attempt to automate this, and have shown it actually contains richer information than CellMarker (Fig. 3.8 (e,f)).

Even when that CellMeSH does not provide an exact fine-grained cell-type, its predictions might still be a useful guide for researchers to pick related scRNA-Seq experiment data, which may not be indexed with MeSH terms, for further analysis using cell-level annotation methods such as label transfer Seurat.

One additiona limitation of the CelLMeSH database is that it does not contain information regarding *how* exactly a particular gene-cell type pair are associated for any given publication. They could be associated at the genomic, epigenomic, or transcriptomic level, or perhaps the gene is up-regulated or down-regulated for the cell type. Such information is not available from the Gene2pubmed and MeSH indices where CellMeSH database is built upon.

The association in CellMeSH only indicates whether a gene and a cell type are both referenced in one publication. While such association contains spurious relations, such noisy signals should be small when we consider large number of publications. We also designed query algorithm to model the noises, and experiments show good annotation performance (Fig. 3.3 and 3.8).

Moreover, for other species, even other model organisms, gene-cell information is limited due to a lack of indexed publications. However, the CellMeSH approach could in principle be generalized given a list of putative cell types.

To address these limitations, it is helpful to build a fully automated solution that picks up fine-grained cell types as well as gene cell-type relationships directly from raw literature. This is an interesting direction and a challenging task, as such information are expressed in a rich way (e.g. T cell is also written as T-Lymphocytes), requires relationship classifications (e.g. genomic, epigenomic or transcriptomic level, up or down regulation), and scattered in different places (e.g. main text, or supplementary tables), and the harvested data will be noisier than CellMeSH.

More advanced techniques using natural language processing might be a promising approach. Ideally such an approach would enable the identification of new cell types and gene cell type relations in papers using unsupervised or semi-supervised named entity recognition and relation extraction techniques (Nadeau and Sekine, 2007; Yadav and Bethard, 2019).

There are also terms within the MeSH ontology that may be useful but are not under the "Cell" heading, such as tissues, organs, and diseases. Designing the query methods utilizing these information is another interesting future direction. For instance, we can refine our search scope if we know the tissue information of the query; or if such information is missing we could provide them from an extended Cell/Tissue-MeSH database.

# Chapter 4

# UNCURL-App

## 4.1 Introduction

UNCURL-App combines data preprocessing, dimensionality reduction, clustering, differential expression, and interactive data analysis within an online graphical user interface. UNCURL-App introduces two key innovations: First, cell type databases are integrated directly with the rest of the analysis process, accelerating mapping of clusters to cell types. Second, UNCURL-App includes tools for interactive re-analysis that allow the user to create, merge, or delete clusters, thus making it possible to iteratively refine clusters using knowledge about gene expression, putative cell type annotations, and other information accessible through UNCURL-App. Because the entire workflow can be performed in a browser and because external knowledge is made available during the analysis process, we expect UNCURL-app not only to accelerate scRNA-seq analysis but also to further extend the user base for this technology.

## 4.2 Methods

From the user's perspective, the first step in the UNCURL-App pipeline is to upload the data as a gene-cell read count matrix. Next, the app automatically performs preprocessing and clustering using UNCURL. The result of UNCURL is then used for dimensionality reduction and clustering. Dimensionality reduction is done using standard methods, such as tSNE (Maaten and Hinton, 2008) or UMAP (McInnes and Healy, 2018). This produces a two-dimensional

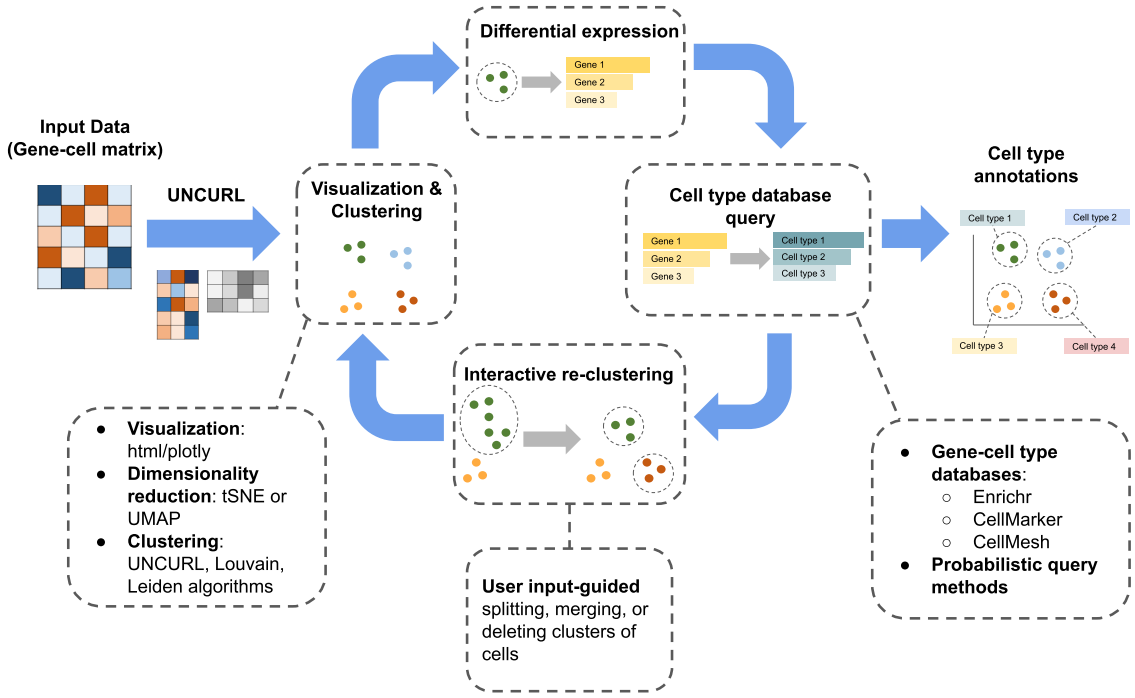



Figure 4.1: This shows the major components and workflow of UNCURL-App. Given the user-uploaded gene-cell matrix: 1) UNCURL is run on the data, producing a clustering; 2) dimensionality reduction is used for visualization, and 3) differentially expressed genes are identified. Then, the user may do cell type annotation using database queries on the top genes, and perform interactive re-clustering.

scatterplot of cells. By default, clustering is done using argmax on the $W$ matrix returned by UNCURL (as described in Mukherjee et al. (2018)). Each column in $W$ represents the weights for each archetype in one cell, so the archetype with the maximum weight is the most likely cluster assignment for that cell. Clustering can also be done using the Louvain (Blondel et al., 2008) or Leiden (Korsunsky et al., 2018) community detection algorithms, which also use the $W$ matrix as input. However, only clustering using UNCURL is compatible with iterative cluster refinement as detailed below.

In order to identify the most differentially expressed genes in each cluster, UNCURL-App uses one of two methods: the t-test, or the ratio of means. These metrics can either be calculated for one cluster against all other clusters, or against a single cluster. The t-test has been shown to be one of the best performing methods for identifying DE genes in scRNA-seq datasets, and is also much faster than more complex methods (Soneson and Robinson, 2018).

There are three main visualization components: the dimensionality-reduced scatterplot of

cells, the barplot showing the most differentially expressed genes, and the cell type query results. A labeled screenshot of the main UNCURL-App view is shown in Figure 4.2. The scatterplot, on the top left of the screen, shows a dimensionality-reduced view of the cells in the dataset, where each point represents a cell. This view can be colored by cluster, gene expression for selected gene(s), or custom label sets based on uploaded files or user-defined criteria. For example, a user may select all cells belonging to a given cluster that also have positive expression of a certain gene, or select all cells that both belong to a certain cluster and have a certain label in an uploaded file (Figure 4.6). The user may also select cells by drawing a box or shape on the plot itself.

On the top right of the screen, the barplot shows the top differentially expressed genes for the selected cluster or label. The barplot is automatically updated whenever the user clicks on a cell on the scatterplot, showing the top genes for the cluster that the cell belongs to.

The bottom left of the screen shows the database query view. From this view, the user may query cell type databases using the top differentially expressed genes. Databases include Enrichr (Kuleshov et al., 2016), CellMarker (Zhang et al., 2018b), and Gene Ontology (Ashburner et al., 2000; noa, 2019a), as well as our new cell type database, CellMeSH (Mao et al., 2022). Submitting a query will return a list of cell types with a confidence score, overlapping genes, and references for each gene-cell type pair.

**Interactive data analysis**

UNCURL-App has the capacity to merge, split, or delete clusters of cells in an interactive fashion. After the initial analysis process is completed, there are often refinements to the clustering that users would like to make, no matter the quality of the initial clustering. For example, the user may want to split a large cluster, merge multiple similar clusters, or delete a group of poor quality cells or potential doublets. This cluster refinement might be based on the shape of the scatterplot, differential expression results, cell type queries, or some other metrics.

The user-driven changes in clustering are incorporated into UNCURL by using them to generate new initializations and then re-running the optimization process, as shown in Figure 4.3. This process fundamentally relies on the UNCURL algorithm (Mukherjee et al., 2018), and

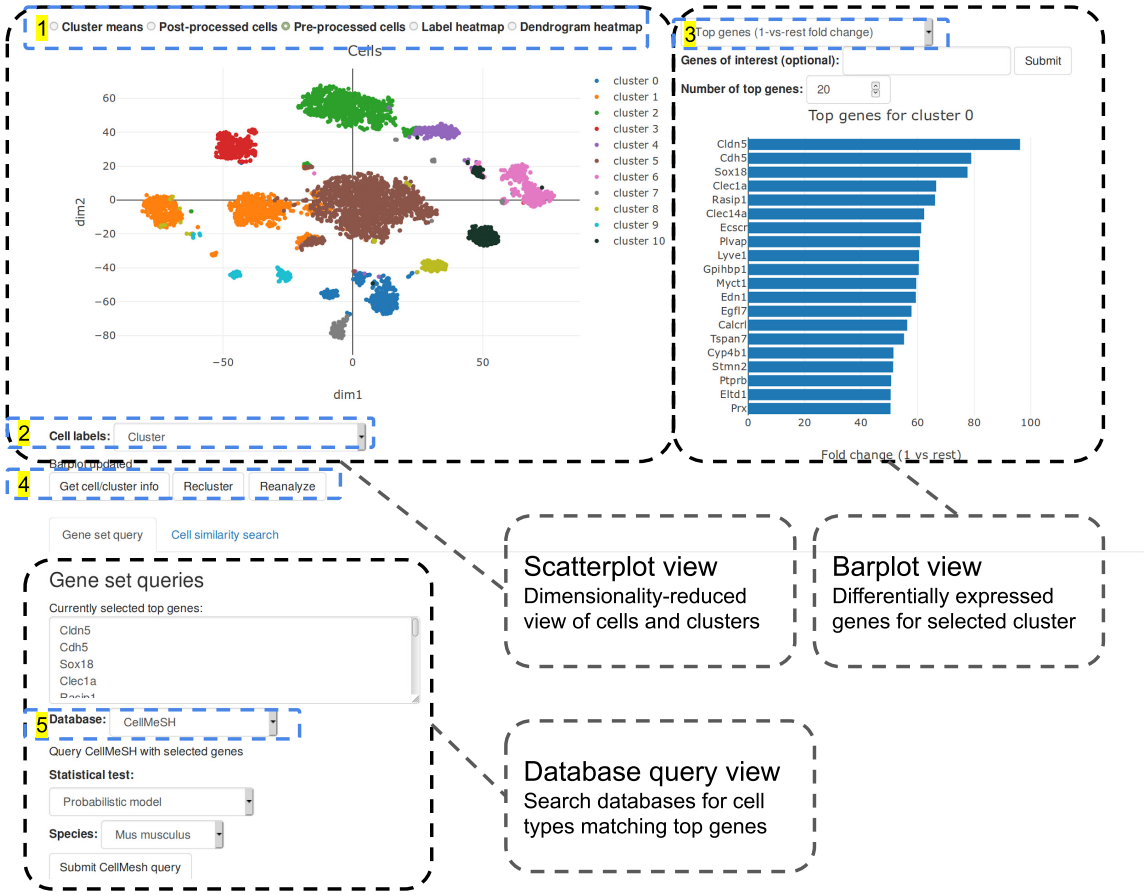



Figure 4.2: This figure shows the main user interface of UNCURL-App. The top left contains the scatterplot, showing a views of the cells and clusters. The top right is a barplot containing the top genes per cluster. Clicking on a cluster in the scatterplot updates the top genes. On the bottom is the database query view, which is used to identify cell types present in the dataset. **Options: (1)** These are options for what type of visualization to show in the scatterplot."Cluster means" shows one point for each cluster. "Post-processed cells" shows a dimensionality-reduced view of cells that takes the $W$ matrix returned by UNCURL as input. "Pre-processed cells" (default) shows a similar view, but using the original data as input for dimensionality reduction. "Label heatmap" shows a comparison of two cell label sets. "Dendrogram heatmap" shows a clusters vs genes dendrogram. **(2)** Options for how to color the cells/points in the scatterplot. Can be based on cluster, gene expression, or a custom label set. **(3)** Options for how to identify the top genes. Can be 1-vs-rest or pairwise, p-value or ratio. **(4)** "Get cell/cluster info": get read/gene count for the selected cell and cluster. "Recluster": options to merge, split, delete, or create a new cluster. "Reanalyze": options to re-run the whole analysis. **(5)** There are a number of databases that can be used for cell type identification, including CellMarker, CellMeSH, and Gene Ontology.

was inspired by the UTOPIAN software for interactive non-negative matrix factorization (Choo et al., 2013), but in UNCURL-App, cells take the place of documents. Say that we have matrices $M$ and $W$, with shapes $g \times K$ and $K \times c$. In order to split a selected cluster, we first run k-means

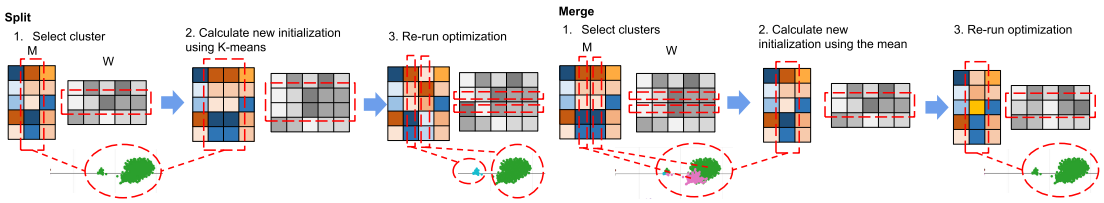



Figure 4.3: **Splitting/Merging clusters** This shows how the process of splitting and merging clusters works in UNCURL-App. Both processes begin with the $M$ and $W$ matrices returned by UNCURL, and the cluster(s) to merge or split. In order to split a cluster, a new initialization for $M$ and $W$ is created by running k-means on the cluster of cells to be split. Then, the UNCURL optimization process is re-run to create new matrices. The process for merging is analogous: a new initialization is created using the mean of the selected cells, and then the UNCURL optimization process is re-run to create a new $M$ and $W$.

with $k = 2$ on the cells assigned to the selected cluster. This generates new matrices $M_{cluster}$ and $W_{cluster}$, of shape $g \times 2$ and $2 \times c$ representing the means and cell cluster assignments. The column and row corresponding to the selected cluster are deleted from $M$ and $W$, and $M_{cluster}$ and $W_{cluster}$ are appended to $M$ and $W$, creating $M_{new}$ and $W_{new}$, with shapes $g \times (K+1)$ and $(K+1) \times c$. Then, UNCURL is re-run with $M_{new}$ and $W_{new}$ as the initializations, which affects other clusters as well. The process is analogous for merging clusters and assigning cells to new clusters: we create new initializations for $M$ and $W$ using the selected clusters or cells, and then re-run the optimization process.

After generating $M_{new}$ and $W_{new}$, a new visualization, clustering, and differential expression results are calculated using $W_{new}$. Running re-clustering also automatically updates the differential expression results.

### 4.2.1 Implementation

UNCURL-App and the associated backend tools and databases are written in Python. The primary package is the uncurl-app package, which uses the Flask library as the server backend. Visualization is done in javascript using the plotly library. The backend, which interfaces with the dimensionality reduction and differential expression methods as well as UNCURL, is provided by the uncurl-analysis package, and the databases are provided by the cellmarker and cellmesh packages.

#### Deployment

UNCURL-App has been tested to run on Ubuntu 16.04 and above, using Python 3.6 and above, and can be deployed on a local or cloud server using Docker. We have created an example UNCURL-App deployment at https://uncurl.cs.washington.edu/. This deployment limits its upload size to 150MB.

### 4.2.2 Pre-processing details

UNCURl-App has a number of options for pre-processing the data prior to running UNCURL, which are described below.

#### Batch effect correction

When running UNCURL-App with multiple datasets, there is the option of using batch effect correction to integrate the data. This uses the mutual nearest neighbors approach Haghverdi et al. (2018), as implemented in mnnpy (`https://github.com/chriscainx/mnnpy`).

#### Data Filtering

By default, UNCURL-App removes the top and bottom 5% of cells by total gene expression level. This threshold can be adjusted - the user can set the maximum and minimum read counts for cells to be included in the analysis. Cells can also be filtered by the fraction of reads that come from mitochondrial genes, by setting a maximum threshold.

Genes are sub-selected based on variance: first, the genes are binned into $n$ bins based on mean gene expression level. Next, for each bin, the top $k$-fraction of genes are selected based on variance. The default values for $n$ and $k$ are 5 and 0.2. This is the same process that is used in Mukherjee et al. (2018).

#### Data Normalization

By default, UNCURL-App normalizes the data by dividing the value of every gene in each cell by the sum of gene expression levels in that cell, so that the total gene expression level sums to 1, and then multiplying the expression levels by the median gene expression levels across all

cells. This is the same process that is used in Mukherjee et al. (2018). This is done before gene selection. The normalized and filtered data is then used for running UNCURL, as well as for differential expression analysis.

### 4.2.3 User Interface

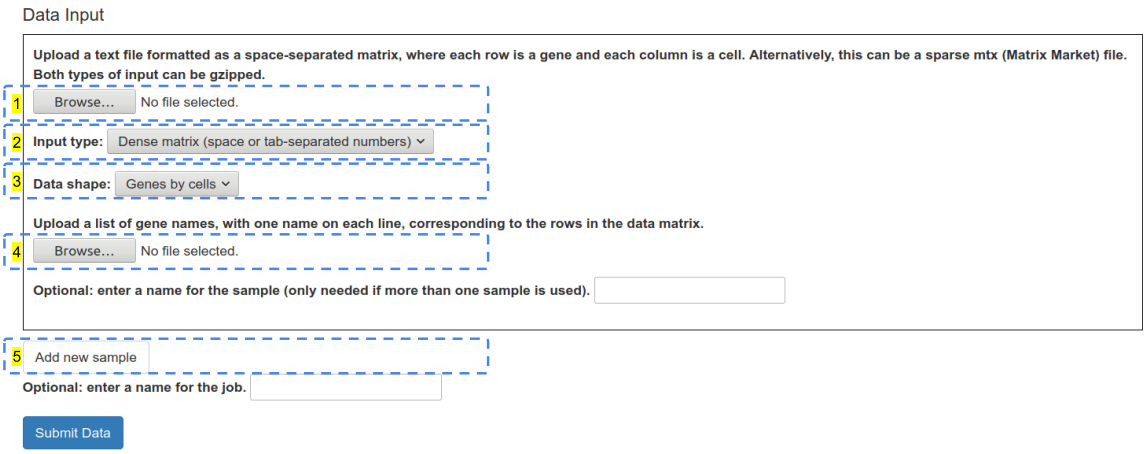



Figure 4.4: Data input interface of UNCURL-App. **(1, 2)** The scRNA-seq data uploaded can be a tab-separated file, or a sparse matrix in the .mtx format. **(3)** The input matrix shape can be genes by cells or cells by genes. **(4)** A separate text file containing gene names should be uploaded, with one gene name per line. **(5)** UNCURL-App can be used with multiple samples; this allows the user to upload additional data matrices.

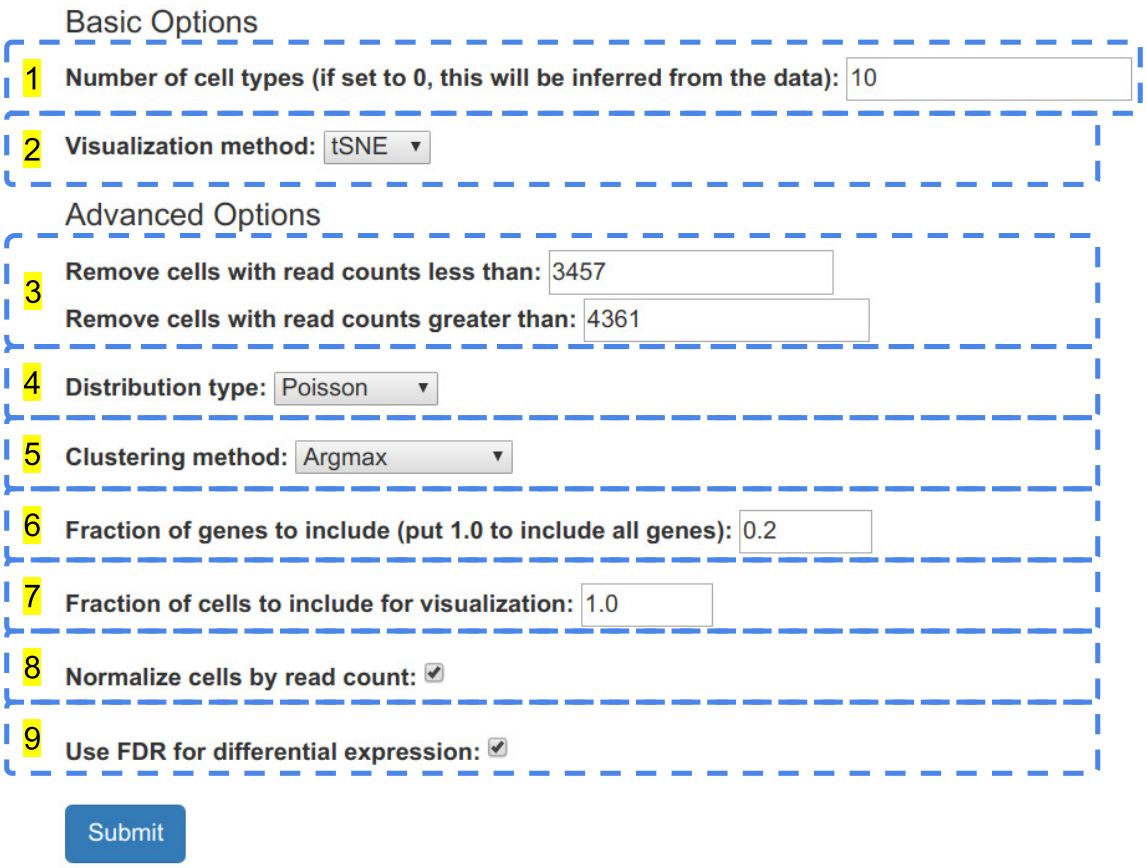



Figure 4.5: View after uploading a dataset and options that can be used for analysis. **(1)** The number of initial cell types (default: 10); if set to 0, this is automatically inferred using the gap score metric. **(2)** The dimensionality reduction method used for visualization can be tSNE, PCA, or UMAP. **(3)** Option to filter cells by min/max read counts. By default, this is set to the bottom and top 5%. **(4)** The distribution type is an option used by UNCURL, representing the sampling distribution of the dataset. This is Poisson by default, but could also be Log-Normal or Gaussian. **(5)** The default clustering method is Argmax with Louvain or Leiden as alternative options. **(6)** By default, UNCURL selects genes by first binning the genes into 5 bins by mean, and selecting the top 20% of genes by variance within each bin. The fraction can be set to any number between 0.01 and 1. **(7)** If this parameter is less than 1, then a random sample of cells is selected for visualization. **(8)** If this option is checked, then before running any other analysis step, the input matrix is normalized so that the total read count per cell is approximately the same. **(9)** If this is checked, the p-values shown for the differential expression will be adjusted using the Benjamini-Hochberg procedure for false discovery rate.

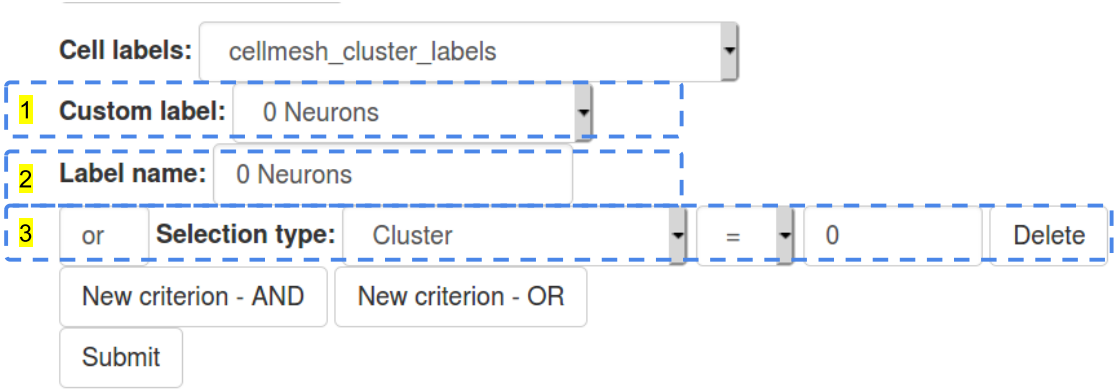



Figure 4.6: User interface for creating custom cell selections. Custom cell selections provide a way to create labels based on user-defined criteria. This view is shown after selecting "Custom cell selection" from the "Cell labels" dropdown menu. After creating a name for the cell selection (in this case, "cellmesh_cluster_labels"), the user can create custom groups of cells using a variety of criteria. **(1)** Each "Custom label" is essentially a cluster of cells defined using custom criteria. **(2)** Input for naming the custom label. **(3)** For each custom label, there can be any number of selection criteria. The "Selection type" may be UNCURL-App clusters, user-uploaded labels, gene expression values, or selections drawn on the scatterplot using Plotly's box or lasso select tools. New selection criteria can be added using the "New criterion - AND" and "New criterion - OR" buttons below.

## 4.3 Results

In order to validate the UNCURL-App workflow, we used the app to analyze three different scRNA-seq datasets, as described below. For these datasets, we performed clustering and cell type annotation using UNCURL-App with default settings. Cell type labels were generated by querying the top 50 genes by 1-vs-rest ratio with the CellMeSH database (see Methods).

The running times of the non-interactive steps are shown in Table 4.1. The running time for UNCURL scales linearly with the number of cells, while the running time for tSNE scales with order $n \log n$, where $n$ is the number of cells. With larger numbers of cells, running tSNE is the most time-consuming step. This can be obviated by using UMAP as the dimensionality reduction method.

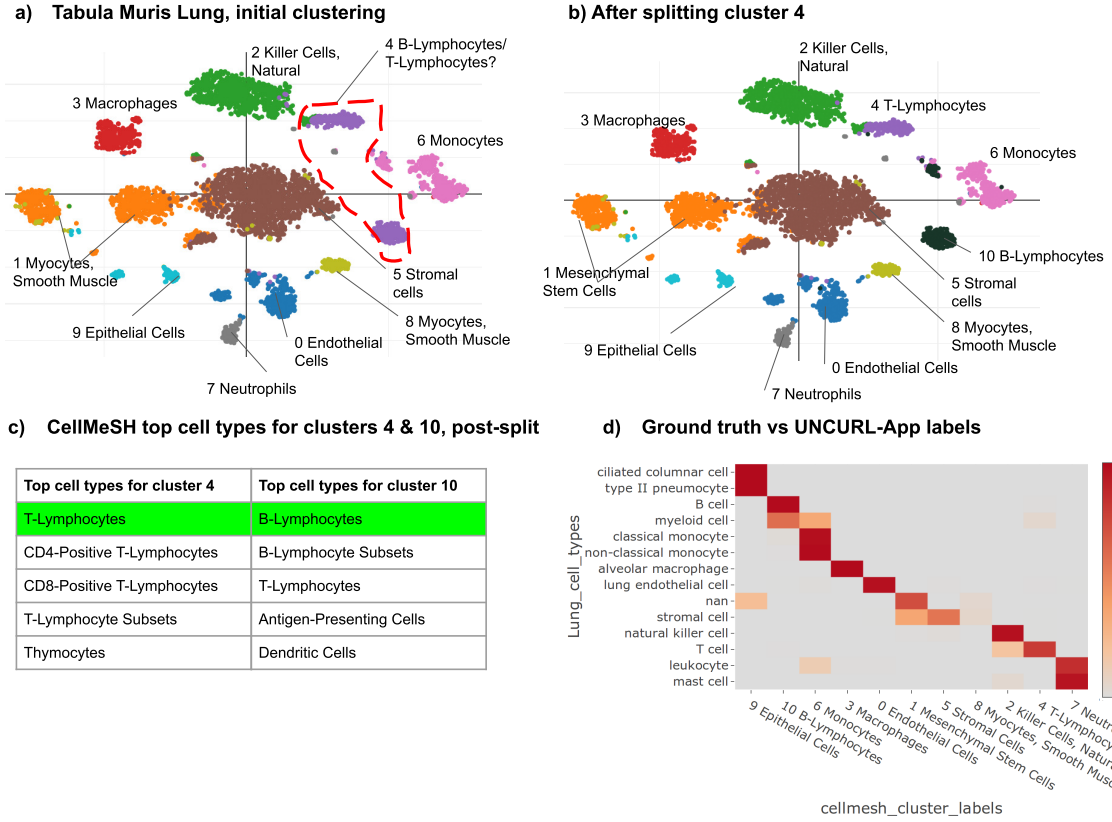
| Dataset | # of Cells | UNCURL | tSNE | DiffExp | Total |
|---|---|---|---|---|---|
| Tabula Muris Lung | 5449 | 127 | 96 | 11 | 234 |
| 10X PBMC | 8000 | 104 | 129 | 9 | 242 |
| Split-seq Spinal Cord | 22614 | 268 | 550 | 16 | 834 |

Table 4.1: Runtime of UNCURL-App. These times were based on an Amazon Web Services (AWS) t2.medium instance, with two processors and 4GB of memory. All times are in seconds.

### 4.3.1 Example: Tabula Muris lung cells

As a first example, we consider a subset of the Tabula Muris dataset from Consortium (2018) containing only cell types found in the lung. This dataset contains 5449 cells and 14 annotated cell types. The labels in the original study were generated by first running graph-based clustering and then manually examining the marker genes for each cluster.

After uploading the dataset and processing it with default settings, we see the clustering and initial cell type assignments in Figure 4.7a. The clustering was based on UNCURL, and the scatterplot visualization was generated using tSNE. Based on the scatterplot, cluster 4 appeared to consist of at least two distinct groups of cells. In addition, the top cell types from a CellMeSH query on the top genes in this cluster included both B and T cells (Figure 4.8a), suggesting that this cluster might be a mixture of at least these two cell types. Based on these observations, we decided to split this cluster using our interactive data analysis tools, resulting in the clusters




c) CellMeSH top cell types for clusters 4 & 10, post-split

| Top cell types for cluster 4 | Top cell types for cluster 10 |
|---|---|
| T-Lymphocytes | B-Lymphocytes |
| CD4-Positive T-Lymphocytes | B-Lymphocyte Subsets |
| CD8-Positive T-Lymphocytes | T-Lymphocytes |
| T-Lymphocyte Subsets | Antigen-Presenting Cells |
| Thymocytes | Dendritic Cells |




Figure 4.7: **Tabula Muris Lung**. Examples of running UNCURL-App on the Tabula Muris lung dataset from Consortium (2018) **(a)** **(b)** These show the same tSNE scatterplot, with clusters from the initial clustering and labels from CellMeSH and the clusters and labels after splitting cluster 4 **(c)** These show the most relevant cell types returned by CellMeSH for the given clusters. Cell types matching ground truth labels are highlighted in green. **(d)** This is a heatmap showing the relationship between the ground truth clusters and the clusters generated by UNCURL-App, after the split. The colors indicate the proportion of the ground truth cluster (left) that overlap with the UNCURL-App cluster (bottom).

given in Figure 4.7b. The post-split cell type assignments (Figure 4.7c) appeared to be more consistent with known biology than the original assignments.

Based on Figure 4.7d, there is generally good concordance between the generated clusters and original clusters, as well as between assigned labels and the original labels. Of the labels that were different, in most cases UNCURL-App assigned cell types that were closely related to the original ground truth label (for example, pneumocytes and columnar cells are subsets of epithelial cells, and neutrophils are a subset of leukocytes). The stromal cells are split into multiple clusters in UNCURL-App, which could represent heterogeneity in the original sample that was not captured by the original labels. No prior information about the cell types present

a) CellMeSH top cell types for cluster 4

| Top cell types for cluster 4 |
|---|
| B-Lymphocytes |
| T-Lymphocytes |
| B-Lymphocyte Subsets |
| T-Lymphocyte Subsets |
| T-Lymphocytes, Helper-Inducer |

b) Ground truth labels

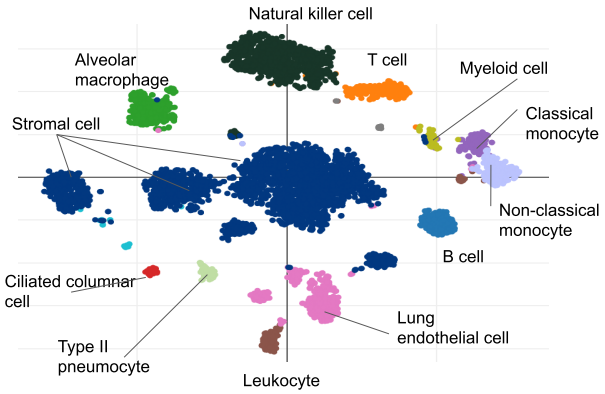



Figure 4.8: **a)** Top cell types for cluster 4 (in Figure 4a) **b)** tSNE plot with ground truth labels

in the dataset was used at any point in this process.

### 4.3.2 Example: 10X PBMCs

Next, we turned to a dataset comprised of 8000 human peripheral blood mononuclear cells (PBMCs) from Zheng et al. (2017). This dataset was created by randomly sampling 1000 cells from each of 8 scRNA-seq datasets comprised of cells that were flow-sorted based on known cell-type markers. Thus, the ground truth cell type labels represent pure samples, as opposed to the computational assignments used as ground truth in the other example datasets.

UNCURL-App was run with default settings to generate 10 initial clusters (Figure 4.9a). Looking at the resulting clusters and putative cell type assignments (4.9b), it appeared that clusters 2 and 6, labeled Neutrophils and Monocytes, were very similar, and could just represent a single group of cells. A pairwise differential expression analysis (Figure 4.9c) further illustrates that only related genes, S100A8 and S100A9, appear to be significantly differentially expressed between these clusters. Plotting the expression levels of these genes (Figure 4.9d), it seems that the small group of cells to the left of the main cluster has much higher expression of these genes, suggesting that this group might constitute a separate cluster. Thus, we first merged clusters 2 and 6, and then split off that small group of cells. These operations resulted in the clustering shown in Figure 4.9e.

As with the previous dataset, there now is good correspondence with the ground truth

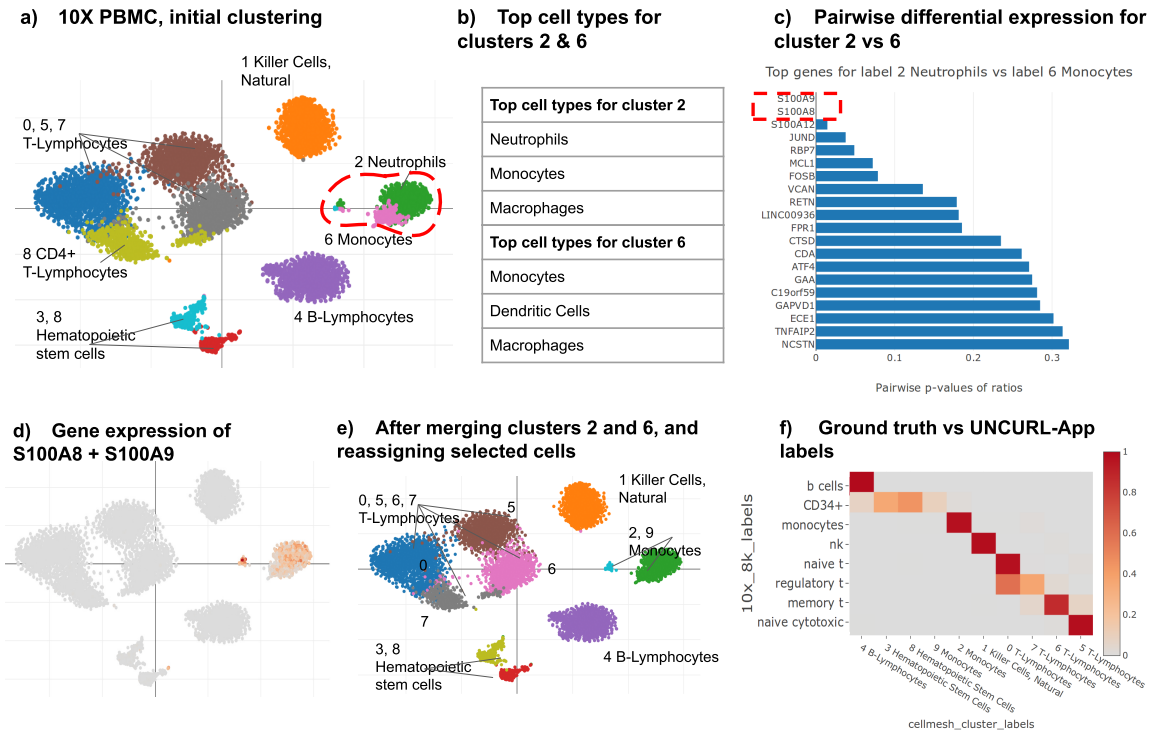
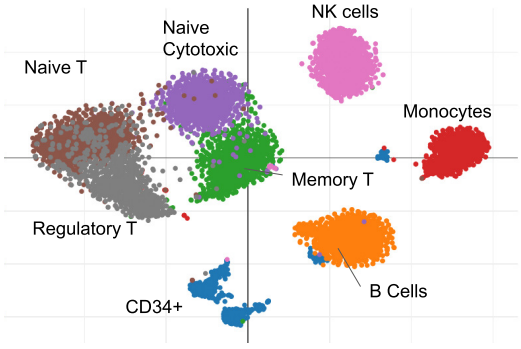



Figure 4.9: **10X PBMC**. Examples of running UNCURL-App on the 10X PBMC dataset from Zheng et al. (2017) **(a)** **(e)** These show the same tSNE scatterplot, with clusters from the initial clustering and labels from CellMeSH, and the clusters and labels after merging clusters 2 and 6 and creating cluster 9,. **(b)** **(g)** This shows the most relevant cell types returned by CellMeSH for the given clusters. **(c)** This shows the top differentially expressed genes for Cluster 2 vs Cluster 6. **(d)** This shows the sum of the gene expressions of the two genes S100A8 and S100A9. **(i)** This is a heatmap showing the relationship between the ground truth clusters and the clusters generated by UNCURL-App, after the split.

**a) Ground truth labels**




**b) Top cell types for T-Lymphocyte clusters**

| Top cell types for cluster 0 | Top cell types for cluster 5 |
|---|---|
| T-Lymphocytes | T-Lymphocytes |
| Lymphocytes | Lymphocytes |
| CD4-Positive T-Lymphocytes | CD8-Positive T-Lymphocytes |
| **Top cell types for cluster 7** | **Top cell types for cluster 6** |
| T-Lymphocytes | T-Lymphocytes |
| CD4-Positive T-Lymphocytes | CD8-Positive T-Lymphocytes |
| T-Lymphocytes, Regulatory | CD4-Positive T-Lymphocytes |

Figure 4.10: **a)** tSNE scatterplot showing ground truth labels. **b)** Top 3 most likely cell types as identified by CellMeSH for clusters from Figure 5e.

clusters and labels (Figure 4.9f). Cells of the same ground truth type are generally assigned to the same cluster, and the cluster labels returned by CellMeSH generally correspond to the ground truth labels. CD34+ cells are generally recognized as hematopoietic stem cells Siena et al. (1989), so the CellMeSH label here seems to be accurate. One major difference is that CellMeSH labeled all four T cell subtypes as "T-Lymphocytes", even though they were clustered into distinct clusters. To investigate further, we looked at the full list of CellMeSH labels for these clusters, not just the top one. These results are shown in Figure 4.10b, with the cell types most similar to the ground truth highlighted in green. For example, Cluster 0 corresponds to naive T-cells, which are selected as CD4+. Cluster 5 corresponds to naive cytotoxic T-cells, which are CD8+, and the "CD8+ T-Lymphocytes" label is the third highest label, below "T-Lymphocytes" and "Lymphocytes". Cluster 6 corresponds to memory T-cells, which can be either CD8+ or CD4+; the second and third labels are "CD8+ T-Lymphocytes" and "CD4-Positive T-Lymphocytes". Cluster 7 corresponds largely to regulatory T-cells, which are CD4+, and the second and third highest CellMeSH labels are "CD4-Positive T-Lymphocytes" and "T-Lymphocytes, Regulatory". This shows a good correspondence between the true and assigned labels at a more fine-grained level.

### 4.3.3 Example: SPLiT-seq spinal cord

For a final test we turned to a larger dataset comprised of 22,614 mouse spinal cord nuclei from 2 and 11-day old mice sequenced using SPLiT-seq (Rosenberg et al., 2018). This dataset has 44 annotated cell types, which is substantially more than the previous two datasets. However, many of these annotated cell types are closely related (for example, there are 15 types of excitatory neurons), so for the "ground truth" comparisons in this section, we combine many of the annotated cell types into larger clusters of similar cells. Even after this process, many of the cell types are similar, with many subtypes of neurons.

We first ran UNCURL-App with default settings to generate an initial clustering with 10 clusters, several of which exhibit substantial heterogeneity (Figure 4.11a). For example, cluster 8 (labeled as "Endothelial Cells") represents at least four different groups of non-neuronal cells. Thus, we split them into four different clusters (Figure 4.11b). It is clear that splitting the

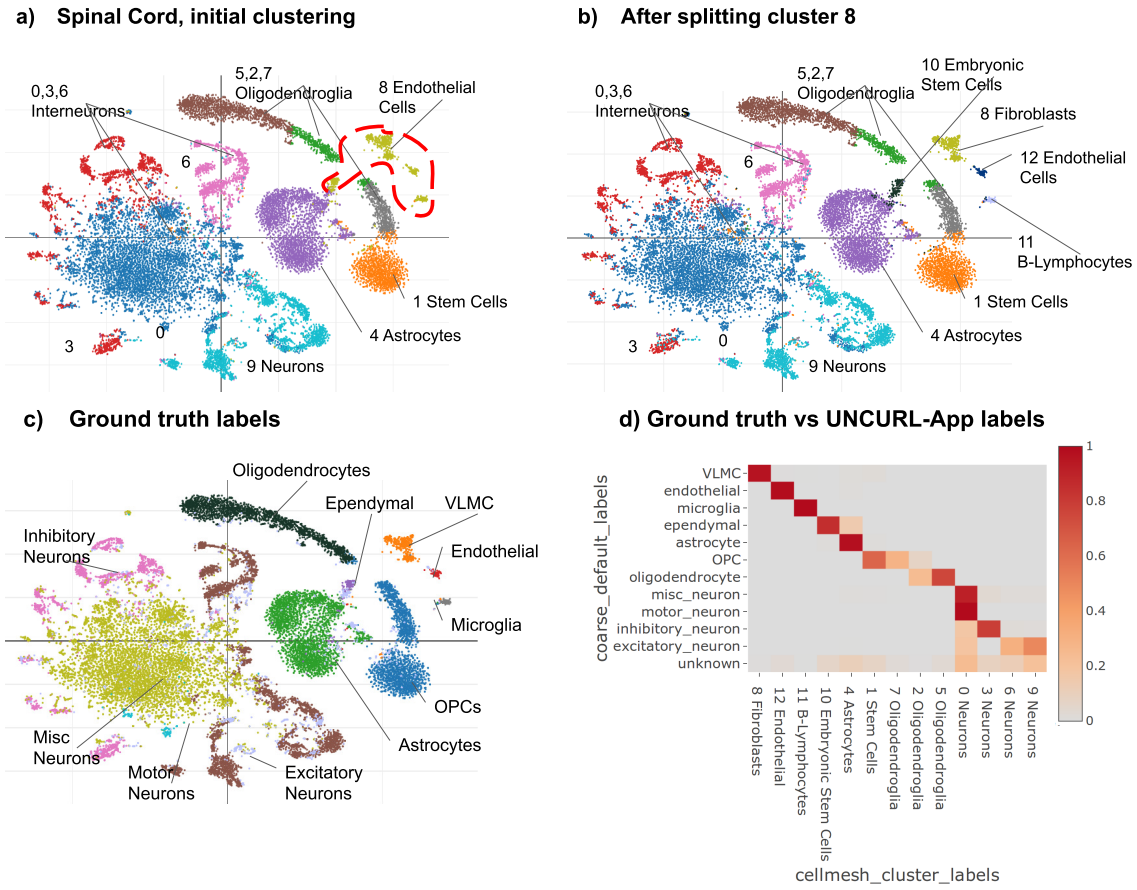



Figure 4.11: **Spinal Cord** Examples of running UNCURL-App on the split-seq spinal cord dataset from Rosenberg et al. (2018) **(a)** tSNE scatterplot, with 10 clusters from the initial clustering and labels from CellMeSH. Cluster 8 which consists of multiple disconnected groups of cells is highlighted. **(b)**Clusters and labels from CellMeSH after splitting cluster 8. **(c)** Clusters and labels after refinement with ground truth labels from the original dataset. **(d)** This is a heatmap showing the relationship between the ground truth clusters and the clusters generated by UNCURL-App, after the split.

clusters worked to separate what appeared to be distinct cell types. In addition, the clusters that CellMeSH labels as "Neurons" or "Interneurons" (3, 6, 9, 0) all appear to be rather heterogeneous. Results after splitting some of the neuronal clusters are shown in Figures 4.12-4.15.

As with the previous datasets, there is generally good concordance between the cell types from the original paper and the clusters generated by UNCURL-App, as shown in Figure 4.11d. Also similarly to previous datasets, the CellMeSH annotations were generally coarser grained than the original hand-annotated labels, with all of the neuron clusters being labeled as "Interneurons" or just "Neurons". For the non-neuronal results, interpreting the labels identified

by CellMeSH is more challenging (Figure 4.13). Oligodendrocytes, astrocytes, and endothelial cells were correctly identified. For cluster 8, the ground-truth label was "VLMC", or "vascular and leptomeningeal cells". This is a highly specific category that does not appear in the CellMeSH ontology but was used as a cell label in Marques et al. (2016). Still, while coarse, the first three labels suggested by CellMeSH (Stromal Cells, Fibroblasts, Mesenchymal Stem Cells) seem consistent with cells derived from the meninges, the membrane enveloping the brain and spinal cord. In cluster 10, the ground-truth label "Ependymal" was not correctly identified by CellMeSH, and the returned results did not seem to relate to ependymal cells. This points to a paucity of annotated publications with gene markers for this cell type. For cluster 11, all of the top CellMeSH results were immune cells, a group which the published label, "microglia", belongs to. "Microglia" was one of the top 10 cell types returned.

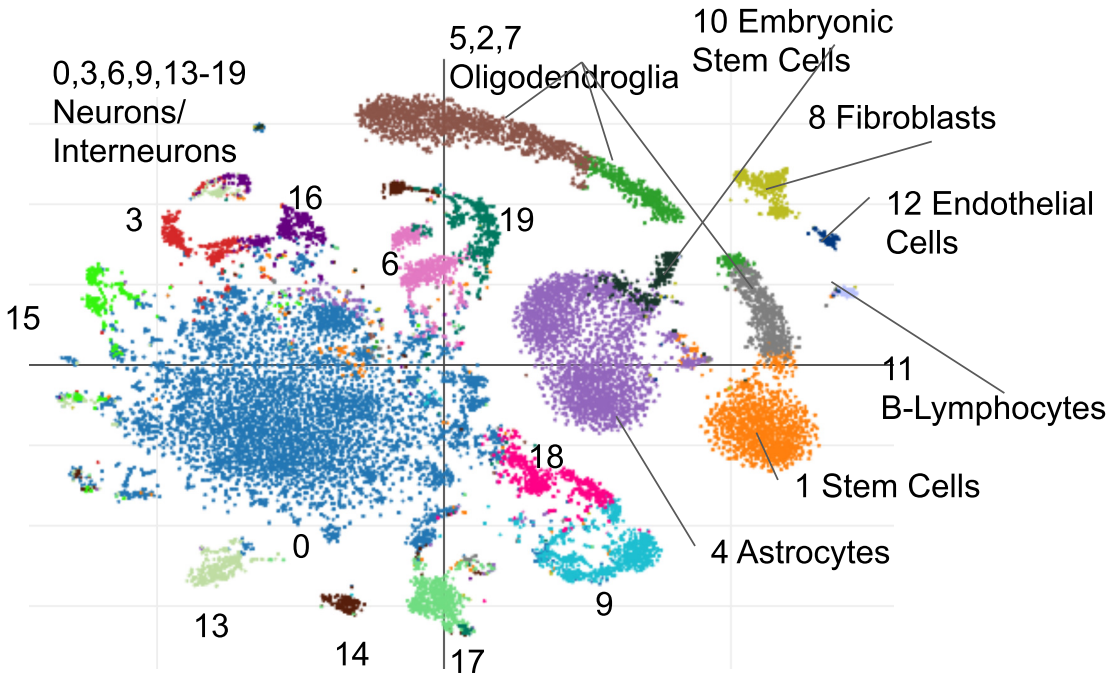



Figure 4.12: Scatterplot view of the split-seq dataset after further splitting out the neuronal clusters. The non-neuronal clusters are the same as in Figure 4.11c.

| Cluster | CellMeSH Cell type #1 | Cell type #2 | Cell type #3 | Most common published label |
|---|---|---|---|---|
| 0 | Neurons | Interneurons | Amacrine Cells | Misc Neuron |
| 1 | Stem Cells | Neural Stem Cells | Neuroglia | Oligodendrocyte precursor cell |
| 2 | Oligodendroglia | Schwann Cells | Neurons | Oligodendrocyte |
| 3 | Interneurons | Motor Neurons | Neurons | Inhibitory neuron |
| 4 | Astrocytes | Neuroglia | Neural Stem Cells | Astrocyte |
| 5 | Oligodendroglia | Nerve Fibers, Myelinated | Schwann Cells | Oligodendrocyte |
| 6 | Interneurons | Neurons | Posterior Horn Cells | Excitatory neuron |
| 7 | Oligodendroglia | Stem Cells | Neurons | Oligodendrocyte precursor cell |
| 8 | Stromal Cells | Fibroblasts | Mesenchymal Stem Cells | VLMC (Vascular and leptomeningeal cell) |
| 9 | Neurons | Interneurons | Dopaminergic Neurons | Excitatory neuron |
| 10 | Embryonic Stem Cells | Olfactory Receptor Neurons | Neurons, Afferent | Ependymal cell |
| 11 | B-Lymphocytes | T-Lymphocytes | Bone Marrow Cells | Microglia |
| 12 | Endothelial Cells | Human Umbilical Vein Endothelial Cells | Pericytes | Endothelial cells |
| 13 | Interneurons | Neurons | Purkinje Cells | Inhibitory neuron |
| 14 | Neurons | Posterior Horn Cells | Interneurons | Excitatory neuron |
| 15 | Interneurons | Posterior Horn Cells | Neurons | Inhibitory neuron |
| 16 | Interneurons | Neurons | Stem Cells | Inhibitory neuron |
| 17 | Posterior Horn Cells | Neurons | Interneurons | Excitatory neuron |
| 18 | Neurons | Dopaminergic Neurons | Neurons, Afferent | Excitatory neuron |
| 19 | Posterior Horn Cells | Interneurons | Neurons | Excitatory neuron |

Figure 4.13: This table shows the top CellMeSH cluster labels for the clusters shown in Figure 4.12, and the label from the original publication on the right. For most cell types, the top CellMeSH labels belong to the same general category as the published label. Most cells assigned as "Interneurons" were originally labeled as "Inhibitory neurons", while most cells assigned as just "Neurons" or "Posterior Horn Cells" were originally labeled as "Excitatory neurons". The CellMeSH label for cluster 10 is the only one that appears to be clearly incorrect.

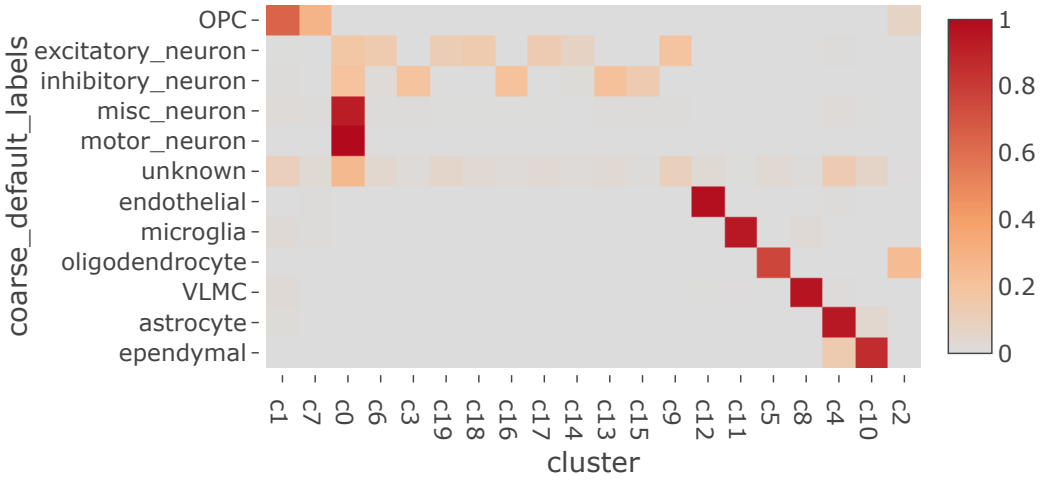



Figure 4.14: Heatmap showing the (coarse-grained) ground-truth cell types on the y-axis and the UNCURL-App clusters on the x-axis, for the clustering in Figure 4.12.

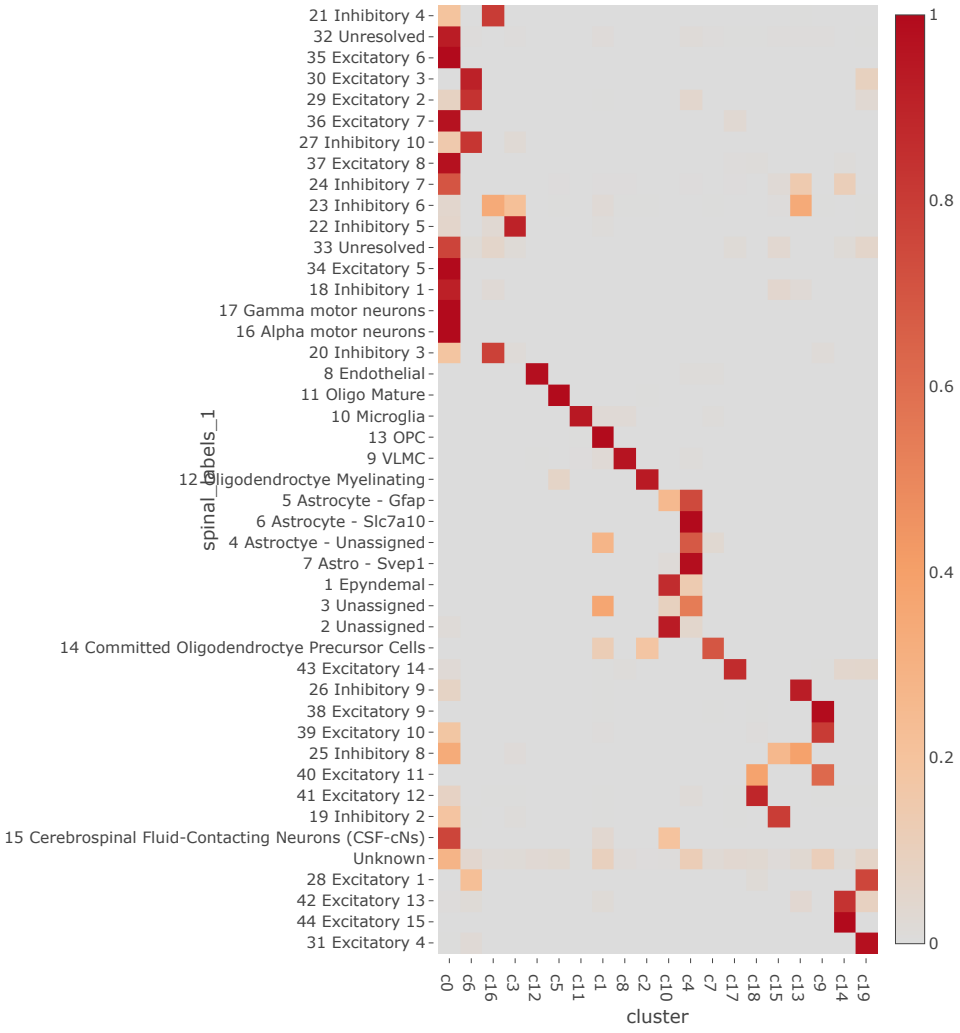



Figure 4.15: Heatmap showing the (fine-grained) ground-truth cell types on the y-axis and the UNCURL-App clusters on the x-axis, for the clustering in Figure 4.12. The fine-grained ground-truth labels were the labels provided in the original paper, without combining labels representing similar cell types.

## 4.4 Discussion

### 4.4.1 Dealing with unknown cell types

In many circumstances, scRNA-seq experiments will reveal cell populations that are distinct from any known cell types. This is a notable advantage of scRNA-seq, but it makes the problem of cell type annotation somewhat ill-defined. With UNCURL-App, if a user attempts to annotate a previously unknown cell type with CellMeSH, then CellMeSH will return the closest cell type present in the database. Even if the exact cell type is not present, this still provides useful information, as the returned cell type is usually a supertype of the more specific cell type of interest. In addition, UNCURL-App allows for gene set queries to other databases, such as the Gene Ontology (Ashburner et al., 2000; noa, 2019a) and KEGG Pathway databases Kanehisa and Goto (2000), that can be used to aid in cell type identification. This might be a supertype of the more specific cell type of interest.

### 4.4.2 Comparison with existing tools

nlike other general-purpose toolkits for scRNA-seq analysis such as scanpy Wolf et al. (2018), Seurat Satija et al. (2015), and Monocle 2 Qiu et al. (2017), UNCURL-App is a web-based GUI tool that does not require command line usage. This allows a much wider range of potential users, such as biologists who are not programmers.

There are a number of web-based tools for scRNA-seq analysis, such as scQuery Alavi et al. (2018), Granatum Zhu et al. (2017), Cumulus Li et al. (2020), and Galaxy Tekman et al. (2020). All of these tools, along with UNCURl-App, can perform dimensionality reduction, clustering, and differential expression on uploaded single-cell datasets. Both scQuery and Cumulus (along with UNCURL-App) but not Granatum or Galaxy also do cell type annotation. Cumulus requires users to provide cell type gene markers in order to annotate cell types, while UNCURL-App does not require any user input for cell type annotation, as it uses the CellMeSH database. ScQuery identifies cell types using cell type annotations derived from GEO. Compared to scQuery, the annotation accuracy of CellMeSH (used by UNCURL-App) is significantly higher Mao et al. (2022). Unlike all of the aforementioned tools, UNCURL-App is also capable

of interactively merging, splitting, and deleting clusters of cells within the GUI.

There are a number of tools that classify cells given gene markers for known cell types, such as Pliner et al. (2019); Zhang et al. (2019a). We view these tools as complementary to UNCURL-App and CellMeSH. These tools require some knowledge of the cell types present in the dataset, as well as a way to manually find gene markers for these cell types, whereas such prior knowledge is unnecessary in the UNCURL-App/CellMeSH pipeline. In addition, CellMeSH can be used to improve the workflow for these tools by automatically selecting gene markers, obviating the need for manually finding them.

There also exist tools that perform single cell similarity search on reference datasets, such as CellAtlasSearch and scMatch (Srivastava et al., 2018; Hou et al., 2019). Rather than using marker genes, these methods compare the entire gene expression profile of every single cell to a reference database, using locality-sensitive hashing in the case of CellAtasSearch (Srivastava et al., 2018) or Pearson or Spearman correlation in the case of scMatch (Hou et al., 2019). These tools do not include functionality for clustering or low-dimensional visualization. The advantage of UNCURL-App comes with its integration of clustering, differential expression, interactive re-analysis, and cell type querying into one easy-to-use platform.

### 4.4.3 Conclusions

UNCURL-App provides a useful way to perform interactive scRNA-seq data analysis, including cell type annotation. In the future, we hope to augment UNCURL-App with new analysis capabilities, such as cell lineage and gene network analysis. We also hope to connect UNCURL-App to additional sources of information for cell type and functional annotation. This could come in the form of connections to new databases, or expansions to the CellMeSH database. Our ultimate goal is to increase UNCURL-App's utility as a general tool for scRNA-seq analysis.

# Chapter 5

# Conclusions

The three tools that we have described, UNCURL, CellMeSH, and UNCURL-App, comprise a novel and useful pipeline for the analysis of scRNA-seq data. Our pipeline has been shown to be useful for cell type identification, outperforming existing methods on a number of datasets, both as individual tools and as a unified pipeline.

In the future, we hope to augment UNCURL-App with new analysis capabilities, such as improved batch effect correction, cell lineages, and gene network and pathway analysis. We also hope to connect UNCURL-App to additional sources of information for cell type and functional annotation. This could come in the form of connections to new databases, or expansions to the CellMeSH database.

In order to expand the CellMeSH database, one approach would be to use natural language processing on the scientific literature. This would involve using NLP techniques such as named entity recognition (NER) to recognize cell types and genes in publications, to expand the set of cell-gene relationships present in CellMeSH. We can use methods such as SciSpacy (Neumann et al., 2019) to aid in entity recognition in scientific publications. One further area of development would be to identify new, unannotated cell types from the literature using unsupervised NER.

Our ultimate goal is to increase UNCURL-App's utility as a general tool for scRNA-seq analysis.

# Chapter 6

# Acknowledgements

Thanks to my advisors, Georg Seelig and Sreeram Kannan, for all of their support. Thanks to all of my collaborators in the Seelig Lab, and especially to Sumit Mukherjee and Shunfu Mao for collaboration on UNCURL and CellMeSH. Thanks to Li Liu, Tao Peng, and Matthew Hirano for testing UNCURL-App and suggesting new features. Finally, thanks to my parents, Na and Wenbo, and my sister Jessica.

# Bibliography

MEDLINE Indexing Online Training Course, 2015. URL https://www.nlm.nih.gov/bsd/indexing/training/USE_010.html.

The Gene Ontology Resource: 20 years and still GOing strong. *Nucleic Acids Research*, 47 (D1):D330–D338, Jan. 2019a. ISSN 0305-1048. doi: 10.1093/nar/gky1055. URL https://academic.oup.com/nar/article/47/D1/D330/5160994.

Medical Subject Headings, 2019b. URL https://www.nlm.nih.gov/mesh/meshhome.html.

MEDLINE®: Description of the Database, 2019c. URL https://www.nlm.nih.gov/bsd/medline.html.

A. Alavi, M. Ruffalo, A. Parvangada, Z. Huang, and Z. Bar-Joseph. A web server for comparative analysis of single-cell RNA-seq data. *Nature Communications*, 9(1):4768, Nov. 2018. ISSN 2041-1723. doi: 10.1038/s41467-018-07165-2. URL https://www.nature.com/articles/s41467-018-07165-2.

S. Anders and W. Huber. Differential expression analysis for sequence count data. *Genome Biology*, 11(10):R106, Oct. 2010. ISSN 1465-6906. doi: 10.1186/gb-2010-11-10-r106. URL http://genomebiology.com/2010/11/10/R106/abstract.

T. S. Andrews, V. Y. Kiselev, D. McCarthy, and M. Hemberg. Tutorial: guidelines for the computational analysis of single-cell RNA sequencing data. *Nature Protocols*, 16(1):1–9, Jan. 2021. ISSN 1750-2799. doi: 10.1038/s41596-020-00409-w. URL https://www.nature.com/articles/s41596-020-00409-w. Number: 1 Publisher: Nature Publishing Group.

D. Aran, A. P. Looney, L. Liu, E. Wu, V. Fong, A. Hsu, S. Chak, R. P. Naikawadi, P. J. Wolters, A. R. Abate, A. J. Butte, and M. Bhattacharya. Reference-based analysis of lung single-cell sequencing reveals a transitional profibrotic macrophage. *Nature Immunology*, 20 (2):163–172, Jan. 2019. doi: 10.1038/s41590-018-0276-y. URL https://doi.org/10.1038/s41590-018-0276-y.

D. Arthur and S. Vassilvitskii. K-means++: The Advantages of Careful Seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, pages 1027–1035, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics. ISBN 978-0-89871-624-5. URL http://dl.acm.org/citation.cfm?id=1283383.1283494.

M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, M. A. Harris, D. P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J. C. Matese, J. E. Richardson, M. Ringwald, G. M. Rubin, and G. Sherlock. Gene Ontology: tool for the unification of biology. *Nature genetics*, 25(1):25–29, May 2000. ISSN 1061-4036. doi: 10.1038/75556. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3037419/.

J. Barretina, G. Caponigro, N. Stransky, K. Venkatesan, A. A. Margolin, S. Kim, C. J. Wilson, J. Lehár, G. V. Kryukov, D. Sonkin, A. Reddy, M. Liu, L. Murray, M. F. Berger, J. E. Monahan, P. Morais, J. Meltzer, A. Korejwa, J. Jané-Valbuena, F. A. Mapa, J. Thibault, E. Bric-Furlong, P. Raman, A. Shipway, I. H. Engels, J. Cheng, G. K. Yu, J. Yu, P. Aspesi, M. de Silva, K. Jagtap, M. D. Jones, L. Wang, C. Hatton, E. Palescandolo, S. Gupta, S. Mahan, C. Sougnez, R. C. Onofrio, T. Liefeld, L. MacConaill, W. Winckler, M. Reich, N. Li, J. P. Mesirov, S. B. Gabriel, G. Getz, K. Ardlie, V. Chan, V. E. Myer, B. L. Weber, J. Porter, M. Warmuth, P. Finan, J. L. Harris, M. Meyerson, T. R. Golub, M. P. Morrissey, W. R. Sellers, R. Schlegel, and L. A. Garraway. The cancer cell line encyclopedia enables predictive modelling of anticancer drug sensitivity. *Nature*, 483(7391):603–607, Mar. 2012. doi: 10.1038/nature11003. URL https://doi.org/10.1038/nature11003.

H. H. Bauschke, J. Bolte, and M. Teboulle. A Descent Lemma Beyond Lipschitz Gradient Continuity: First-Order Methods Revisited and Applications. *Mathematics of Operations*

*Research*, 42(2):330–348, Nov. 2016. ISSN 0364-765X. doi: 10.1287/moor.2016.0817. URL http://pubsonline.informs.org/doi/10.1287/moor.2016.0817.

S. Bendall, K. Davis, E.-a. Amir, M. Tadmor, E. Simonds, T. Chen, D. Shenfeld, G. Nolan, and D. Pe'er. Single-Cell Trajectory Detection Uncovers Progression and Regulatory Coordination in Human B Cell Development. *Cell*, 157(3):714–725, Apr. 2014. ISSN 0092-8674. doi: 10.1016/j.cell.2014.04.005. URL http://www.sciencedirect.com/science/article/pii/S0092867414004711.

V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008. ISSN 1742-5468. doi: 10.1088/1742-5468/2008/10/P10008. URL http://stacks.iop.org/1742-5468/2008/i=10/a=P10008.

C. Boutsidis and E. Gallopoulos. SVD based initialization: A head start for nonnegative matrix factorization. *Pattern Recognition*, 41(4):1350–1362, Apr. 2008. ISSN 0031-3203. doi: 10.1016/j.patcog.2007.09.010. URL http://www.sciencedirect.com/science/article/pii/S0031320307004359.

A. Butler, P. Hoffman, P. Smibert, E. Papalexi, and R. Satija. Integrating single-cell transcriptomic data across different conditions, technologies, and species. *Nature Biotechnology*, 36(5):411–420, Apr. 2018. doi: 10.1038/nbt.4096. URL https://doi.org/10.1038/nbt.4096.

J. Cao, J. S. Packer, V. Ramani, D. A. Cusanovich, C. Huynh, R. Daza, X. Qiu, C. Lee, S. N. Furlan, F. J. Steemers, A. Adey, R. H. Waterston, C. Trapnell, and J. Shendure. Comprehensive single-cell transcriptional profiling of a multicellular organism. *Science*, 357(6352):661–667, Aug. 2017. ISSN 0036-8075, 1095-9203. doi: 10.1126/science.aam8940. URL https://science.sciencemag.org/content/357/6352/661. Publisher: American Association for the Advancement of Science Section: Research Article.

J. Cao, M. Spielmann, X. Qiu, X. Huang, D. M. Ibrahim, A. J. Hill, F. Zhang, S. Mundlos, L. Christiansen, F. J. Steemers, C. Trapnell, and J. Shendure. The single-cell transcriptional landscape of mammalian organogenesis. *Nature*, 566(7745):496–502, Feb. 2019. ISSN

1476-4687. doi: 10.1038/s41586-019-0969-x. URL https://www.nature.com/articles/s41586-019-0969-x. Number: 7745 Publisher: Nature Publishing Group.

E. Y. Chen, C. M. Tan, Y. Kou, Q. Duan, Z. Wang, G. Meirelles, N. R. Clark, and A. Ma'ayan. Enrichr: interactive and collaborative HTML5 gene list enrichment analysis tool. *BMC Bioinformatics*, 14(1):128, 2013. doi: 10.1186/1471-2105-14-128. URL https://doi.org/10.1186/1471-2105-14-128.

G. Chen, B. Ning, and T. Shi. Single-Cell RNA-Seq Technologies and Related Computational Data Analysis. *Frontiers in Genetics*, 10, 2019. ISSN 1664-8021. URL https://www.frontiersin.org/article/10.3389/fgene.2019.00317.

X. Chen, S. A. Teichmann, and K. B. Meyer. From Tissues to Cell Types and Back: Single-Cell Gene Expression Analysis of Tissue Architecture. *Annual Review of Biomedical Data Science*, 1(1):29–51, 2018. doi: 10.1146/annurev-biodatasci-080917-013452. URL https://doi.org/10.1146/annurev-biodatasci-080917-013452.

J. Choo, C. Lee, C. K. Reddy, and H. Park. UTOPIAN: User-Driven Topic Modeling Based on Interactive Nonnegative Matrix Factorization. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):1992–2001, Dec. 2013. ISSN 1077-2626. doi: 10.1109/TVCG.2013.212.

T. T. M. Consortium. Single-cell transcriptomics of 20 mouse organs creates a Tabula Muris. *Nature*, 562(7727):367, Oct. 2018. ISSN 1476-4687. doi: 10.1038/s41586-018-0590-4. URL https://www.nature.com/articles/s41586-018-0590-4.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B*, 39 (1):1–38, 1977.

J. J. Diaz-Mejia, E. C. Meng, A. R. Pico, S. A. MacParland, T. Ketela, T. J. Pugh, G. D. Bader, and J. H. Morris. Evaluation of methods to assign cell type labels to cell clusters from single-cell RNA-sequencing data. *bioRxiv*, page 562082, Feb. 2019. doi: 10.1101/562082. URL https://www.biorxiv.org/content/10.1101/562082v1.

D. v. Dijk, J. Nainys, R. Sharma, P. Kathail, A. J. Carr, K. R. Moon, L. Mazutis, G. Wolf, S. Krishnaswamy, and D. Pe'er. MAGIC: A diffusion-based imputation method reveals gene-gene interactions in single-cell RNA-sequencing data. *bioRxiv*, page 111591, Feb. 2017. doi: 10.1101/111591. URL http://biorxiv.org/content/early/2017/02/25/111591.

D. v. Dijk, R. Sharma, J. Nainys, K. Yim, P. Kathail, A. J. Carr, C. Burdziak, K. R. Moon, C. L. Chaffer, D. Pattabiraman, B. Bierie, L. Mazutis, G. Wolf, S. Krishnaswamy, and D. Pe'er. Recovering Gene Interactions from Single-Cell Data Using Data Diffusion. *Cell*, 174(3):716–729.e27, July 2018. ISSN 0092-8674, 1097-4172. doi: 10.1016/j.cell.2018.05.061. URL http://www.cell.com/cell/abstract/S0092-8674(18)30724-4.

A. Dobin, C. A. Davis, F. Schlesinger, J. Drenkow, C. Zaleski, S. Jha, P. Batut, M. Chaisson, and T. R. Gingeras. Star: ultrafast universal rna-seq aligner. *Bioinformatics*, 2012. doi: 10.1093/bioinformatics/bts635. URL http://bioinformatics.oxfordjournals.org/content/early/2012/10/25/bioinformatics.bts635.abstract.

G. Finak, A. McDavid, M. Yajima, J. Deng, V. Gersuk, A. K. Shalek, C. K. Slichter, H. W. Miller, M. J. McElrath, M. Prlic, P. S. Linsley, and R. Gottardo. MAST: a flexible statistical framework for assessing transcriptional changes and characterizing heterogeneity in single-cell RNA sequencing data. *Genome Biology*, 16:278, Dec. 2015. ISSN 1474-760X. doi: 10.1186/s13059-015-0844-5. URL https://doi.org/10.1186/s13059-015-0844-5.

R. A. Fisher. The logic of inductive inference. *Journal of the Royal Statistical Society*, 98(1):39, 1935. doi: 10.2307/2342435. URL https://doi.org/10.2307/2342435.

O. Franzén, L.-M. Gan, and J. L. M. Björkegren. PanglaoDB: a web server for exploration of mouse and human single-cell RNA sequencing data. *Database*, 2019, Jan. 2019. doi: 10.1093/database/baz046. URL https://doi.org/10.1093/database/baz046.

W. Gong, I.-Y. Kwak, P. Pota, N. Koyano-Nakagawa, and D. J. Garry. DrImpute: imputing dropout events in single cell RNA sequencing data. *BMC Bioinformatics*, 19(1):220, June 2018. ISSN 1471-2105. doi: 10.1186/s12859-018-2226-y. URL https://doi.org/10.1186/s12859-018-2226-y.

K. Gu, H. K. T. Ng, M. L. Tang, and W. R. Schucany. Testing the ratio of two poisson rates. *Biometrical Journal*, 50(2):283–298, 2008.

L. Haghverdi, A. T. L. Lun, M. D. Morgan, and J. C. Marioni. Batch effects in single-cell RNA-sequencing data are corrected by matching mutual nearest neighbors. *Nature Biotechnology*, 36(5):421–427, May 2018. ISSN 1546-1696. doi: 10.1038/nbt.4091. URL `https://www.nature.com/articles/nbt.4091`.

X. Han, R. Wang, Y. Zhou, L. Fei, H. Sun, S. Lai, A. Saadatpour, Z. Zhou, H. Chen, F. Ye, D. Huang, Y. Xu, W. Huang, M. Jiang, X. Jiang, J. Mao, Y. Chen, C. Lu, J. Xie, Q. Fang, Y. Wang, R. Yue, T. Li, H. Huang, S. H. Orkin, G.-C. Yuan, M. Chen, and G. Guo. Mapping the Mouse Cell Atlas by Microwell-Seq. *Cell*, 172(5):1091–1107.e17, Feb. 2018. ISSN 0092-8674, 1097-4172. doi: 10.1016/j.cell.2018.02.001. URL `http://www.cell.com/cell/abstract/S0092-8674(18)30116-8`.

N. K. Hanchate, K. Kondoh, Z. Lu, D. Kuang, X. Ye, X. Qiu, L. Pachter, C. Trapnell, and L. B. Buck. Single-cell transcriptomics reveals receptor transformations during olfactory neurogenesis. *Science*, page aad2456, Nov. 2015. ISSN 0036-8075, 1095-9203. doi: 10.1126/science.aad2456. URL `http://science.sciencemag.org/content/early/2015/11/04/science.aad2456`.

S. Hänzelmann, R. Castelo, and J. Guinney. GSVA: gene set variation analysis for microarray and RNA-seq data. *BMC Bioinformatics*, 14(1):7, 2013. doi: 10.1186/1471-2105-14-7. URL `https://doi.org/10.1186/1471-2105-14-7`.

T. K. Ho. Random decision forests. In *Proceedings of the Third International Conference on Document Analysis and Recognition (Volume 1) - Volume 1*, ICDAR '95, pages 278–282, M, 1995. IEEE Computer Society. ISBN 0-8186-7128-9.

R. Hou, E. Denisenko, and A. R. R. Forrest. scMatch: a single-cell gene expression profile annotation tool using reference datasets. *Bioinformatics*, Apr. 2019. doi: 10.1093/bioinformatics/btz292. URL `https://academic.oup.com/bioinformatics/advance-article/doi/10.1093/bioinformatics/btz292/5480299`.

S. Islam, U. Kjällquist, A. Moliner, P. Zajac, J.-B. Fan, P. Lönnerberg, and S. Linnarsson. Characterization of the single-cell transcriptional landscape by highly multiplex RNA-seq. *Genome Research*, 21(7):1160–1167, July 2011. ISSN 1088-9051, 1549-5469. doi: 10.1101/gr.110882.110. URL http://genome.cshlp.org/content/21/7/1160.

S. Islam, A. Zeisel, S. Joost, G. La Manno, P. Zajac, M. Kasper, P. Lönnerberg, and S. Linnarsson. Quantitative single-cell RNA-seq with unique molecular identifiers. *Nature Methods*, 11(2):163–166, Feb. 2014. ISSN 1548-7105. doi: 10.1038/nmeth.2772. URL https://www.nature.com/articles/nmeth.2772. Number: 2 Publisher: Nature Publishing Group.

M. Kanehisa and S. Goto. KEGG: Kyoto Encyclopedia of Genes and Genomes. *Nucleic Acids Research*, 28(1):27–30, Jan. 2000. ISSN 0305-1048. doi: 10.1093/nar/28.1.27. URL https://doi.org/10.1093/nar/28.1.27.

P. V. Kharchenko, L. Silberstein, and D. T. Scadden. Bayesian approach to single-cell differential expression analysis. *Nature Methods*, 11(7):740–742, July 2014. ISSN 1548-7091. doi: 10.1038/nmeth.2967. URL http://www.nature.com/nmeth/journal/v11/n7/full/nmeth.2967.html.

V. Y. Kiselev, K. Kirschner, M. T. Schaub, T. Andrews, A. Yiu, T. Chandra, K. N. Natarajan, W. Reik, M. Barahona, A. R. Green, and M. Hemberg. SC3: consensus clustering of single-cell RNA-seq data. *Nature Methods*, 14(5):483–486, May 2017. ISSN 1548-7091. doi: 10.1038/nmeth.4236. URL https://www.nature.com/nmeth/journal/v14/n5/full/nmeth.4236.html.

V. Y. Kiselev, A. Yiu, and M. Hemberg. scmap: projection of single-cell RNA-seq data across data sets. *Nature Methods*, Apr. 2018. ISSN 1548-7105. doi: 10.1038/nmeth.4644. URL https://www.nature.com/articles/nmeth.4644.

I. Korsunsky, J. Fan, K. Slowikowski, F. Zhang, K. Wei, Y. Baglaenko, M. Brenner, P.-R. Loh, and S. Raychaudhuri. Fast, sensitive, and accurate integration of single cell data with

Harmony. *bioRxiv*, page 461954, Nov. 2018. doi: 10.1101/461954. URL https://www.biorxiv.org/content/early/2018/11/05/461954.

M. V. Kuleshov, M. R. Jones, A. D. Rouillard, N. F. Fernandez, Q. Duan, Z. Wang, S. Koplev, S. L. Jenkins, K. M. Jagodnik, A. Lachmann, M. G. McDermott, C. D. Monteiro, G. W. Gundersen, and A. Ma'ayan. Enrichr: a comprehensive gene set enrichment analysis web server 2016 update. *Nucleic Acids Research*, 44(W1):W90–W97, July 2016. ISSN 0305-1048. doi: 10.1093/nar/gkw377. URL https://academic.oup.com/nar/article/44/W1/W90/2499357.

A. N. Langville, C. D. Meyer, R. Albright, J. Cox, and D. Duling. Initializations for the nonnegative matrix factorization. In *Proceedings of the twelfth ACM SIGKDD international conference on knowledge discovery and data mining*, pages 23–26. Citeseer, 2006.

B. Li and C. N. Dewey. Rsem: accurate transcript quantification from rna-seq data with or without a reference genome. *BMC Bioinformatics*, 12(1):323, 2011. ISSN 1471-2105. doi: 10.1186/1471-2105-12-323. URL http://dx.doi.org/10.1186/1471-2105-12-323.

B. Li, J. Gould, Y. Yang, S. Sarkizova, M. Tabaka, O. Ashenberg, Y. Rosen, M. Slyper, M. S. Kowalczyk, A.-C. Villani, T. Tickle, N. Hacohen, O. Rozenblatt-Rosen, and A. Regev. Cumulus provides cloud-based data analysis for large-scale single-cell and single-nucleus RNA-seq. *Nature Methods*, 17(8):793–798, Aug. 2020. ISSN 1548-7105. doi: 10.1038/s41592-020-0905-x. URL https://www.nature.com/articles/s41592-020-0905-x. Number: 8 Publisher: Nature Publishing Group.

P. Lin, M. Troup, and J. W. K. Ho. CIDR: Ultrafast and accurate clustering through imputation for single-cell RNA-seq data. *Genome Biology*, 18:59, Mar. 2017. ISSN 1474-760X. doi: 10.1186/s13059-017-1188-0. URL https://doi.org/10.1186/s13059-017-1188-0.

M. I. Love, W. Huber, and S. Anders. Moderated estimation of fold change and dispersion for rna-seq data with deseq2. *Genome biology*, 15(12):550, 2014.

D. Lähnemann, J. Köster, E. Szczurek, D. J. McCarthy, S. C. Hicks, M. D. Robinson, C. A. Vallejos, K. R. Campbell, N. Beerenwinkel, A. Mahfouz, L. Pinello, P. Skums, A. Stamatakis,

C. S.-O. Attolini, S. Aparicio, J. Baaijens, M. Balvert, B. d. Barbanson, A. Cappuccio, G. Corleone, B. E. Dutilh, M. Florescu, V. Guryev, R. Holmer, K. Jahn, T. J. Lobo, E. M. Keizer, I. Khatri, S. M. Kielbasa, J. O. Korbel, A. M. Kozlov, T.-H. Kuo, B. P. Lelieveldt, I. I. Mandoiu, J. C. Marioni, T. Marschall, F. Mölder, A. Niknejad, L. Raczkowski, M. Reinders, J. d. Ridder, A.-E. Saliba, A. Somarakis, O. Stegle, F. J. Theis, H. Yang, A. Zelikovsky, A. C. McHardy, B. J. Raphael, S. P. Shah, and A. Schönhuth. Eleven grand challenges in single-cell data science. *Genome Biology*, 21(1):31, Feb. 2020. ISSN 1474-760X. doi: 10.1186/s13059-020-1926-6. URL `https://doi.org/10.1186/s13059-020-1926-6`.

F. Ma and M. Pellegrini. ACTINN: automated identification of cell types in single cell RNA sequencing. *Bioinformatics*, July 2019. doi: 10.1093/bioinformatics/btz592. URL `https://doi.org/10.1093/bioinformatics/btz592`.

L. v. d. Maaten and G. Hinton. Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008. ISSN ISSN 1533-7928. URL `http://www.jmlr.org/papers/v9/vandermaaten08a.html`.

E. Macosko, A. Basu, R. Satija, J. Nemesh, K. Shekhar, M. Goldman, I. Tirosh, A. Bialas, N. Kamitaki, E. Martersteck, J. Trombetta, D. Weitz, J. Sanes, A. Shalek, A. Regev, and S. McCarroll. Highly Parallel Genome-wide Expression Profiling of Individual Cells Using Nanoliter Droplets. *Cell*, 161(5):1202–1214, May 2015. ISSN 0092-8674. doi: 10.1016/j.cell.2015.05.002. URL `http://www.sciencedirect.com/science/article/pii/S0092867415005498`.

D. Maglott, J. Ostell, K. D. Pruitt, and T. Tatusova. Entrez Gene: gene-centered information at NCBI. *Nucleic Acids Research*, 35(Database issue):D26–D31, Jan. 2007. ISSN 0305-1048. doi: 10.1093/nar/gkl993. URL `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1761442/`.

C. Manning. *Introduction to information retrieval*. Cambridge University Press, New York, 2008. ISBN 0521865719.

S. Mao and Y. Zhang. Cellmesh api, 2020. `https://github.com/shunfumao/cellmesh`, Last accessed on 2020-06-05.

S. Mao, Y. Zhang, G. Seelig, and S. Kannan. CellMeSH: probabilistic cell-type identification using indexed literature. *Bioinformatics*, 38(5):1393–1402, Mar. 2022. ISSN 1367-4803. doi: 10.1093/bioinformatics/btab834. URL https://doi.org/10.1093/bioinformatics/btab834.

S. Marques, A. Zeisel, S. Codeluppi, D. v. Bruggen, A. M. Falcão, L. Xiao, H. Li, M. Häring, H. Hochgerner, R. A. Romanov, D. Gyllborg, A. B. Muñoz-Manchado, G. L. Manno, P. Lönnerberg, E. M. Floriddia, F. Rezayee, P. Ernfors, E. Arenas, J. Hjerling-Leffler, T. Harkany, W. D. Richardson, S. Linnarsson, and G. Castelo-Branco. Oligodendrocyte heterogeneity in the mouse juvenile and adult central nervous system. *Science*, 352(6291): 1326–1329, June 2016. ISSN 0036-8075, 1095-9203. doi: 10.1126/science.aaf6463. URL https://science.sciencemag.org/content/352/6291/1326. Publisher: American Association for the Advancement of Science Section: Report.

L. McInnes and J. Healy. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *arXiv:1802.03426 [cs, stat]*, Feb. 2018. URL http://arxiv.org/abs/1802.03426. arXiv: 1802.03426.

S. Mukherjee, Y. Zhang, J. Fan, G. Seelig, and S. Kannan. Scalable preprocessing for sparse scRNA-seq data exploiting prior knowledge. *Bioinformatics*, 34(13):i124–i132, July 2018. ISSN 1367-4803. doi: 10.1093/bioinformatics/bty293. URL https://academic.oup.com/bioinformatics/article/34/13/i124/5045758.

D. Nadeau and S. Sekine. A survey of named entity recognition and classification. *Lingvisticæ Investigationes. International Journal of Linguistics and Language Resources*, 30(1):3–26, Aug. 2007. doi: 10.1075/li.30.1.03nad. URL https://doi.org/10.1075/li.30.1.03nad.

M. Neumann, D. King, I. Beltagy, and W. Ammar. Scispacy: Fast and robust models for biomedical natural language processing. *CoRR*, abs/1902.07669, 2019.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine*

*Learning Research*, 12:2825–2830, Oct. 2011. URL http://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html.

W. Perkins, M. Tygert, and R. Ward. Computing the confidence levels for a root-mean-square test of goodness-of-fit. *Applied Mathematics and Computation*, 217(22):9072–9084, July 2011. ISSN 0096-3003. doi: 10.1016/j.amc.2011.03.124. URL http://www.sciencedirect.com/science/article/pii/S0096300311004929.

E. Pierson and C. Yau. ZIFA: Dimensionality reduction for zero-inflated single-cell gene expression analysis. *Genome Biology*, 16:241, Nov. 2015. ISSN 1474-760X. doi: 10.1186/s13059-015-0805-z. URL https://doi.org/10.1186/s13059-015-0805-z.

M. Plass, J. Solana, F. A. Wolf, S. Ayoub, A. Misios, P. Glažar, B. Obermayer, F. J. Theis, C. Kocks, and N. Rajewsky. Cell type atlas and lineage tree of a whole complex animal by single-cell transcriptomics. *Science*, 360(6391):eaaq1723, May 2018. ISSN 0036-8075, 1095-9203. doi: 10.1126/science.aaq1723. URL http://science.sciencemag.org/content/360/6391/eaaq1723.

H. A. Pliner, J. Shendure, and C. Trapnell. Supervised classification enables rapid annotation of cell atlases. *Nature Methods*, 16(10):983–986, Oct. 2019. ISSN 1548-7105. doi: 10.1038/s41592-019-0535-3. URL https://www.nature.com/articles/s41592-019-0535-3.

X. Qiu, Q. Mao, Y. Tang, L. Wang, R. Chawla, H. A. Pliner, and C. Trapnell. Reversed graph embedding resolves complex single-cell trajectories. *Nature Methods*, 14(10):979, Aug. 2017. ISSN 1548-7105. doi: 10.1038/nmeth.4402. URL https://www.nature.com/articles/nmeth.4402.

A. Rajaraman. *Mining of Massive Datasets*. Cambridge University Press, Cambridge, 2011. ISBN 978-1-139-05845-2.

D. Ramsköld, S. Luo, Y.-C. Wang, R. Li, Q. Deng, O. R. Faridani, G. A. Daniels, I. Khrebtukova, J. F. Loring, L. C. Laurent, G. P. Schroth, and R. Sandberg. Full-length mRNA-Seq from single-cell levels of RNA and individual circulating tumor cells. *Nature Biotechnology*, 30(8):

777–782, Aug. 2012. ISSN 1546-1696. doi: 10.1038/nbt.2282. URL https://www.nature.com/articles/nbt.2282. Number: 8 Publisher: Nature Publishing Group.

A. Regev, S. A. Teichmann, E. S. Lander, I. Amit, C. Benoist, E. Birney, B. Bodenmiller, P. Campbell, P. Carninci, M. Clatworthy, H. Clevers, B. Deplancke, I. Dunham, J. Eberwine, R. Eils, W. Enard, A. Farmer, L. Fugger, B. Göttgens, N. Hacohen, M. Haniffa, M. Hemberg, S. Kim, P. Klenerman, A. Kriegstein, E. Lein, S. Linnarsson, E. Lundberg, J. Lundeberg, P. Majumder, J. C. Marioni, M. Merad, M. Mhlanga, M. Nawijn, M. Netea, G. Nolan, D. Pe'er, A. Phillipakis, C. P. Ponting, S. Quake, W. Reik, O. Rozenblatt-Rosen, J. Sanes, R. Satija, T. N. Schumacher, A. Shalek, E. Shapiro, P. Sharma, J. W. Shin, O. Stegle, M. Stratton, M. J. T. Stubbington, F. J. Theis, M. Uhlen, A. van Oudenaarden, A. Wagner, F. Watt, J. Weissman, B. Wold, R. Xavier, N. Yosef, and Human Cell Atlas Meeting Participants. The Human Cell Atlas. *eLife*, 6:e27041, Dec. 2017. ISSN 2050-084X. doi: 10.7554/eLife.27041. URL https://doi.org/10.7554/eLife.27041.

D. Risso, F. Perraudeau, S. Gribkova, S. Dudoit, and J.-P. Vert. A general and flexible method for signal extraction from single-cell RNA-seq data. *Nature Communications*, 9(1):284, Jan. 2018. ISSN 2041-1723. doi: 10.1038/s41467-017-02554-5. URL https://www.nature.com/articles/s41467-017-02554-5.

M. D. Robinson, D. J. McCarthy, and G. K. Smyth. edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. *Bioinformatics*, 26(1):139–140, Jan. 2010. ISSN 1367-4803. doi: 10.1093/bioinformatics/btp616. URL http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2796818/.

A. B. Rosenberg, C. M. Roco, R. A. Muscat, A. Kuchina, P. Sample, Z. Yao, L. Gray, D. J. Peeler, S. Mukherjee, W. Chen, S. H. Pun, D. L. Sellers, B. Tasic, and G. Seelig. Single-cell profiling of the developing mouse brain and spinal cord with split-pool barcoding. *Science*, page eaam8999, Mar. 2018. ISSN 0036-8075, 1095-9203. doi: 10.1126/science.aam8999. URL http://science.sciencemag.org/content/early/2018/03/14/science.aam8999.

R. Satija, J. A. Farrell, D. Gennert, A. F. Schier, and A. Regev. Spatial reconstruction of

single-cell gene expression data. *Nature Biotechnology*, 33(5):495, May 2015. ISSN 1546-1696. doi: 10.1038/nbt.3192. URL https://www.nature.com/articles/nbt.3192.

I. Shmulevich and W. Zhang. Binary analysis and optimization-based normalization of gene expression data. *Bioinformatics*, 18(4):555–565, 2002.

S. Siena, M. Bregni, B. Brando, F. Ravagnani, G. Bonadonna, and A. M. Gianni. Circulation of CD34+ hematopoietic stem cells in the peripheral blood of high-dose cyclophosphamide-treated patients: enhancement by intravenous recombinant human granulocyte-macrophage colony-stimulating factor. *Blood*, 74(6):1905–1914, Nov. 1989. ISSN 0006-4971.

T. S. Smith, A. Heger, and I. Sudbery. UMI-tools: Modelling sequencing errors in Unique Molecular Identifiers to improve quantification accuracy. *Genome Research*, page gr.209601.116, Jan. 2017. ISSN 1088-9051, 1549-5469. doi: 10.1101/gr.209601.116. URL http://genome.cshlp.org/content/early/2017/01/18/gr.209601.116.

C. Soneson and M. D. Robinson. Bias, robustness and scalability in single-cell differential expression analysis. *Nature Methods*, 15(4):255–261, Apr. 2018. ISSN 1548-7105. doi: 10.1038/nmeth.4612. URL https://www.nature.com/articles/nmeth.4612.

D. Srivastava, A. Iyer, V. Kumar, and D. Sengupta. CellAtlasSearch: a scalable search engine for single cells. *Nucleic Acids Research*, 2018. doi: 10.1093/nar/gky421. URL https://academic.oup.com/nar/advance-article/doi/10.1093/nar/gky421/5000022.

H. Stachelscheid, S. Seltmann, F. Lekschas, J.-F. Fontaine, N. Mah, M. Neves, M. A. Andrade-Navarro, U. Leser, and A. Kurtz. CellFinder: a cell data repository. *Nucleic Acids Research*, 42(D1):D950–D958, Dec. 2013. doi: 10.1093/nar/gkt1264. URL https://doi.org/10.1093/nar/gkt1264.

A. I. Su, T. Wiltshire, S. Batalov, H. Lapp, K. A. Ching, D. Block, J. Zhang, R. Soden, M. Hayakawa, G. Kreiman, M. P. Cooke, J. R. Walker, and J. B. Hogenesch. A gene atlas of the mouse and human protein-encoding transcriptomes. *Proceedings of the National Academy of Sciences*, 101(16):6062–6067, Apr. 2004. doi: 10.1073/pnas.0400782101. URL https://doi.org/10.1073/pnas.0400782101.

Z. Sun, L. Chen, H. Xin, Y. Jiang, Q. Huang, A. R. Cillo, T. Tabib, J. K. Kolls, T. C. Bruno, R. Lafyatis, D. A. A. Vignali, K. Chen, Y. Ding, M. Hu, and W. Chen. A Bayesian mixture model for clustering droplet-based single-cell transcriptomic data from population studies. *Nature Communications*, 10(1):1649, Apr. 2019. ISSN 2041-1723. doi: 10.1038/s41467-019-09639-3. URL https://www.nature.com/articles/s41467-019-09639-3.

V. Svensson, E. d. V. Beltrame, and L. Pachter. A curated database reveals trends in single-cell transcriptomics. *bioRxiv*, page 742304, Oct. 2019. doi: 10.1101/742304. URL https://www.biorxiv.org/content/10.1101/742304v2.

Y. Tan and P. Cahan. SingleCellNet: A computational tool to classify single cell RNA-seq data across platforms and across species. *Cell Systems*, 9(2):207–213.e2, Aug. 2019. doi: 10.1016/j.cels.2019.06.004. URL https://doi.org/10.1016/j.cels.2019.06.004.

F. Tang, C. Barbacioru, Y. Wang, E. Nordman, C. Lee, N. Xu, X. Wang, J. Bodeau, B. B. Tuch, A. Siddiqui, K. Lao, and M. A. Surani. mRNA-Seq whole-transcriptome analysis of a single cell. *Nature Methods*, 6(5):377–382, May 2009. ISSN 1548-7105. doi: 10.1038/nmeth.1315. Number: 5 Publisher: Nature Publishing Group.

B. Tasic, V. Menon, T. N. Nguyen, T. K. Kim, T. Jarsky, Z. Yao, B. Levi, L. T. Gray, S. A. Sorensen, T. Dolbeare, D. Bertagnolli, J. Goldy, N. Shapovalova, S. Parry, C. Lee, K. Smith, A. Bernard, L. Madisen, S. M. Sunkin, M. Hawrylycz, C. Koch, and H. Zeng. Adult mouse cortical cell taxonomy revealed by single cell transcriptomics. *Nature Neuroscience*, 19(2):nn.4216, Jan. 2016. ISSN 1546-1726. doi: 10.1038/nn.4216. URL https://www.nature.com/articles/nn.4216.

M. Tekman, B. Batut, A. Ostrovsky, C. Antoniewski, D. Clements, F. Ramirez, G. J. Etherington, H.-R. Hotz, J. Scholtalbers, J. R. Manning, L. Bellenger, M. A. Doyle, M. Heydarian, N. Huang, N. Soranzo, P. Moreno, S. Mautner, I. Papatheodorou, A. Nekrutenko, J. Taylor, D. Blankenberg, R. Backofen, and B. Grüning. A single-cell RNA-sequencing training and analysis suite using the Galaxy framework. *GigaScience*, 9(10), Oct. 2020. ISSN 2047-217X. doi: 10.1093/gigascience/giaa102. URL https://doi.org/10.1093/gigascience/giaa102.

V. Traag, L. Waltman, and N. J. van Eck. From Louvain to Leiden: guaranteeing well-connected communities. *arXiv:1810.08473 [physics]*, Oct. 2018. URL http://arxiv.org/abs/1810.08473. arXiv: 1810.08473.

C. Trapnell, B. A. Williams, G. Pertea, A. Mortazavi, G. Kwan, M. J. van Baren, S. L. Salzberg, B. J. Wold, and L. Pachter. Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation. *Nature Biotechnology*, 28(5):511–515, May 2010. ISSN 1546-1696. doi: 10.1038/nbt.1621. URL https://www.nature.com/articles/nbt.1621. Number: 5 Publisher: Nature Publishing Group.

C. Trapnell, D. Cacchiarelli, J. Grimsby, P. Pokharel, S. Li, M. Morse, N. J. Lennon, K. J. Livak, T. S. Mikkelsen, and J. L. Rinn. The dynamics and regulators of cell fate decisions are revealed by pseudotemporal ordering of single cells. *Nature Biotechnology*, 32(4):381–386, Apr. 2014. ISSN 1087-0156. doi: 10.1038/nbt.2859. URL http://www.nature.com/nbt/journal/v32/n4/full/nbt.2859.html.

D. Usoskin, A. Furlan, S. Islam, H. Abdo, P. Lönnerberg, D. Lou, J. Hjerling-Leffler, J. Haeggström, O. Kharchenko, P. V. Kharchenko, S. Linnarsson, and P. Ernfors. Unbiased classification of sensory neuron types by large-scale single-cell RNA sequencing. *Nature Neuroscience*, 18(1):145–153, 2015. ISSN 1097-6256. doi: 10.1038/nn.3881. URL http://www.nature.com/neuro/journal/v18/n1/full/nn.3881.html.

A.-C. Villani, R. Satija, G. Reynolds, S. Sarkizova, K. Shekhar, J. Fletcher, M. Griesbeck, A. Butler, S. Zheng, S. Lazo, L. Jardine, D. Dixon, E. Stephenson, E. Nilsson, I. Grundberg, D. McDonald, A. Filby, W. Li, P. L. D. Jager, O. Rozenblatt-Rosen, A. A. Lane, M. Haniffa, A. Regev, and N. Hacohen. Single-cell RNA-seq reveals new types of human blood dendritic cells, monocytes, and progenitors. *Science*, 356(6335):eaah4573, Apr. 2017. ISSN 0036-8075, 1095-9203. doi: 10.1126/science.aah4573. URL http://science.sciencemag.org.offcampus.lib.washington.edu/content/356/6335/eaah4573.

B. Wang, J. Zhu, E. Pierson, D. Ramazzotti, and S. Batzoglou. Visualization and analysis of single-cell RNA-seq data by kernel-based similarity learning. *Nature Methods*, 14(4):414, Apr.

2017. ISSN 1548-7105. doi: 10.1038/nmeth.4207. URL https://www.nature.com/articles/nmeth.4207.

J. N. Weinstein. Spotlight on molecular profiling: "integromic" analysis of the NCI-60 cancer cell lines. *Molecular Cancer Therapeutics*, 5(11):2601–2605, Nov. 2006. doi: 10.1158/1535-7163.mct-06-0640. URL https://doi.org/10.1158/1535-7163.mct-06-0640.

J. D. Welch, A. J. Hartemink, and J. F. Prins. SLICER: inferring branched, nonlinear cellular trajectories from single cell RNA-seq data. *Genome Biology*, 17:106, 2016. ISSN 1474-760X. doi: 10.1186/s13059-016-0975-3. URL http://dx.doi.org/10.1186/s13059-016-0975-3.

Wikipedia. Monoblast - wikipedia, 2020a. https://en.wikipedia.org/wiki/Monoblast, Last accessed on 2020-01-20.

Wikipedia. Promonocyte - wikipedia, 2020b. https://en.wikipedia.org/wiki/Promonocyte, Last accessed on 2020-01-20.

Wikipedia. tf-idf - wikipedia, 2020c. https://en.wikipedia.org/wiki/Tf–idf, Last accessed on 2020-02-21.

F. A. Wolf, P. Angerer, and F. J. Theis. SCANPY: large-scale single-cell gene expression data analysis. *Genome Biology*, 19:15, Feb. 2018. ISSN 1474-760X. doi: 10.1186/s13059-017-1382-0. URL https://doi.org/10.1186/s13059-017-1382-0.

V. Yadav and S. Bethard. A survey on recent advances in named entity recognition from deep learning models, 2019.

A. Zeisel, A. B. Muñoz-Manchado, S. Codeluppi, P. Lönnerberg, G. L. Manno, A. Juréus, S. Marques, H. Munguba, L. He, C. Betsholtz, C. Rolny, G. Castelo-Branco, J. Hjerling-Leffler, and S. Linnarsson. Cell types in the mouse cortex and hippocampus revealed by single-cell RNA-seq. *Science*, 347(6226):1138–1142, Mar. 2015. ISSN 0036-8075, 1095-9203. doi: 10.1126/science.aaa1934. URL http://science.sciencemag.org/content/347/6226/1138.

A. W. Zhang, C. O'Flanagan, E. Chavez, J. L. Lim, A. McPherson, M. Wiens, P. Walters, T. Chan, B. Hewitson, D. Lai, A. Mottok, C. Sarkozy, L. Chong, T. Aoki, X. Wang, A. P.

Weng, J. N. McAlpine, S. Aparicio, C. Steidl, K. R. Campbell, and S. P. Shah. Probabilistic cell type assignment of single-cell transcriptomic data reveals spatiotemporal microenvironment dynamics in human cancers. *bioRxiv*, page 521914, Jan. 2019a. doi: 10.1101/521914. URL https://www.biorxiv.org/content/10.1101/521914v1.

A. W. Zhang, C. O'Flanagan, E. A. Chavez, J. L. P. Lim, N. Ceglia, A. McPherson, M. Wiens, P. Walters, T. Chan, B. Hewitson, D. Lai, A. Mottok, C. Sarkozy, L. Chong, T. Aoki, X. Wang, A. P. Weng, J. N. McAlpine, S. Aparicio, C. Steidl, K. R. Campbell, and S. P. Shah. Probabilistic cell-type assignment of single-cell RNA-seq for tumor microenvironment profiling. *Nature Methods*, 16(10):1007–1015, Sept. 2019b. doi: 10.1038/s41592-019-0529-1. URL https://doi.org/10.1038/s41592-019-0529-1.

J. M. Zhang, J. Fan, H. C. Fan, D. Rosenfeld, and D. N. Tse. An interpretable framework for clustering single-cell RNA-Seq datasets. *BMC Bioinformatics*, 19, Mar. 2018a. ISSN 1471-2105. doi: 10.1186/s12859-018-2092-7. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5845381/.

X. Zhang, Y. Lan, J. Xu, F. Quan, E. Zhao, C. Deng, T. Luo, L. Xu, G. Liao, M. Yan, Y. Ping, F. Li, A. Shi, J. Bai, T. Zhao, X. Li, and Y. Xiao. CellMarker: a manually curated resource of cell markers in human and mouse. *Nucleic Acids Research*, 2018b. doi: 10.1093/nar/gky900. URL https://academic.oup.com/nar/advance-article/doi/10.1093/nar/gky900/5115823.

Y. Zhang and S. Mao. Cellmesh web server, 2020. https://uncurl.cs.washington.edu/db_query, Last accessed on 2020-06-05.

Y. Zhang, S. Mao, S. Mukherjee, S. Kannan, and G. Seelig. UNCURL-app: Interactive database-driven analysis of single cell RNA sequencing data. *bioRxiv*, Apr. 2020. doi: 10.1101/2020.04.15.043737. URL https://doi.org/10.1101/2020.04.15.043737.

G. X. Y. Zheng, J. M. Terry, P. Belgrader, P. Ryvkin, Z. W. Bent, R. Wilson, S. B. Ziraldo, T. D. Wheeler, G. P. McDermott, J. Zhu, M. T. Gregory, J. Shuga, L. Montesclaros, J. G. Underwood, D. A. Masquelier, S. Y. Nishimura, M. Schnall-Levin, P. W. Wyatt, C. M. Hindson,

R. Bharadwaj, A. Wong, K. D. Ness, L. W. Beppu, H. J. Deeg, C. McFarland, K. R. Loeb, W. J. Valente, N. G. Ericson, E. A. Stevens, J. P. Radich, T. S. Mikkelsen, B. J. Hindson, and J. H. Bielas. Massively parallel digital transcriptional profiling of single cells. *Nature Communications*, 8:14049, Jan. 2017. ISSN 2041-1723. doi: 10.1038/ncomms14049. URL `http://www.nature.com/ncomms/2017/170116/ncomms14049/full/ncomms14049.html`.

X. Zhu, T. K. Wolfgruber, A. Tasato, C. Arisdakessian, D. G. Garmire, and L. X. Garmire. Granatum: a graphical single-cell RNA-Seq analysis pipeline for genomics scientists. *Genome Medicine*, 9:108, Dec. 2017. ISSN 1756-994X. doi: 10.1186/s13073-017-0492-3. URL `https://doi.org/10.1186/s13073-017-0492-3`.