Eidgenössische Technische Hochschule Zürich

Boeva Lab

Praktische Arbeit - Practical Work

263-0650-00L

Towards intra tumor heterogeneity deconvolution from bulk RNA-seq data

Author
Auguste Lefevre

Supervisors
Prof. Dr. Valentina Boeva
Agnieszka Kraft

November 23, 2021

Contents

Abstract

In this report, we discuss the following topic from cancer genetics and epigenetics: intra tumor heterogeneity deconvolution from bulk RNA-seq data. A diverse collection of cells with specific expression profiles are present in a tumor. Being able to evaluate the different cell states, their proportions and expression profiles from a bulk RNA-seq raw counts would help in the creation of tailored treatments at reduced cost and in a short amount of time. This work is part of a semester project. We concentrate here on the core of the project and leave aside the additional tasks performed during the same time. We laid the groundwork for future study on intra tumor heterogeneity deconvolution from bulk RNA-seq data by working on both the simulation of bulk RNA-seq data from single-cell RNA-seq data and the classification of melanoma cells using a semi-supervised method. In this project, we use TuPro dataset and therefore focus our work on melanoma cancers.

1 Introduction

1.1 Overview

Cancer is a disease caused by the transformation of cells that become abnormal and proliferate excessively [8]. These disordered cells sometimes end up forming a mass called a *tumor*. In the 21st century, cancer is responsible for millions of deaths per year around the globe [18]. There are many different types of cancer that should be interpreted as different diseases as they originate from different types of cells. In our work we focus our analysis on cutaneous malignant melanoma, a type of skin cancer. This cancer type is known to spread to other tissues in the body - metastasis -. According to some studies, specific states of cancer cells in melanoma are responsible for its invasive and proliferative properties. This diversity of cell types within the tumor could result in higher cancer aggressiveness and higher drugs tolerance. Therefore the study of heterogeneity is a central point of research to better understand the tumor composition and the impact of this composition on the clinical prognosis of the patient to develop targeted therapies [11].

1.2 Motivation: Intra-tumor Heterogeneity from bulk RNA-seq

Cancer is a dynamic disease that evolves over time. Tumors might include a diverse collection of cells in different proportions that can show distinct morphological and phenotypic profiles [13]. This heterogeneity is intimately linked to clinical prognosis as it considerably impacts the level of sensitivity to treatments [12]. Therefore, it is very useful to be able to assess with precision the heterogeneity of a tumor in order to develop effective and tailored treatments. Recent methods as single-cell sequencing allowed us to measure expression profiles of a large number of cells simultaneously. Using this sequencing method, it is possible to quantify intra-tumor heterogeneity, with a lot of work. Unfortunately, single-cell sequencing is – technologically – an expensive and laborious task, which is not yet accessible routinely and thus cannot be used in practice. Conversely, bulk data are way simpler to obtain and much cheaper than single-cell sequencing. Therefore, quantifying tumor heterogeneity from bulk data is a very important challenge, especially in a clinical situation.

1.3 Problem Statement

In opposition to single-cell RNA sequencing which results in an expression counts matrix that shows the gene counts for each cell in the tumor, bulk RNA sequencing measures tissue gene expression levels that are average expression profiles of individual cells.

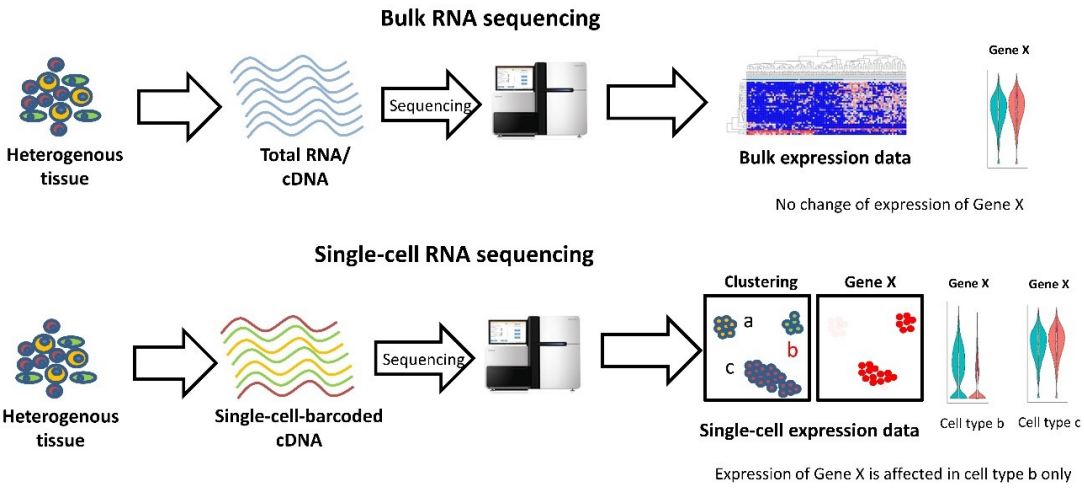


Figure 2: Bulk RNA sequencing vs Single-cell RNA sequencing. Image Credit: Dimitry Velmeshev

From a bulk RNA-seq raw counts we want to infer the different cell types present in the sample and their proportion as well as their expression profile. To accomplish this task, we have at our disposal the TuPro Dataset. From TuPro we have around 100 samples for which we have bulk RNA-seq data. Among these samples, for 67 of them we also have single-cell RNA-seq data which counts for around 20,000 expressed genes (per sample). While single-cell RNA-seq could be used as additional data for a different task, the final input of the project should be a bulk RNA-seq. The ideal output of this work would be a model able to quantify the heterogeneity of any given tumor sample (from melanoma cancer). This task is similar to a signal deconvolution problem as here we want to deconvolute multivariate signals from different cell types.

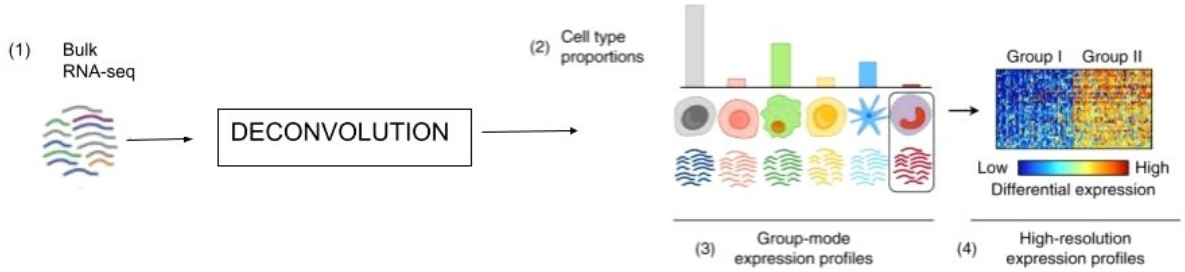


Figure 3: Schematic of the problem

Quantifying heterogeneity from bulk data would help understand its impact on clinical prognosis. Moreover, accessibility and ease of use of bulk data make this project really interesting in terms of application as such methods could allow more people to get tailored and efficient treatments based on the unique expression profile of their tumor.

1.4 Initial Strategy

In order to quantify intra tumor heterogeneity from bulk RNA-seq, there are different approaches available. First of all, deconvolution methods are divided into 2 categories: reference-free (without any prior reference knowledge) and reference-based methods (with prior reference knowledge). In our case, prior knowledge could typically be a signature matrix or a set of marker genes specific to different transcriptional states. Therefore, we can either use well known deconvolution method as Non-Negative Matrix Factorization (NNMF) [9] or Independent Component Analysis (ICA). Or we can develop our own deconvolution strategy, fine-tuned for our specific needs and based on state-of-the-art machine learning/deep learning methods. When we stared this project, NNMF and ICA method had already been tested but did not give satisfactory results. Moreover, we thought it would be more interesting to try novel methods to solve this task. But using state of the art method requires more upstream work. In our case, we need

more samples to train and validate any supervised model but also to improve matrix factorization methods as NNMF and ICA. Thus we work on the simulation of bulk RNA-seq based on single-cell RNA-seq. But to create different samples, we also need to be able to determine which cell types should be used for the simulation and in which proportion. Therefore we need to annotate each single-cell RNA-seq correctly, using semi-supervised method which take benefits of the prior knowledge on marker genes.

1.4.1 Bulk RNA-seq Simulation from Single-Cell RNA-seq

As mentioned above, whether to develop our deconvolution method or to use existing methods as NNMF and ICA: we need way more representative samples. In order to have more samples, we decide to simulate bulk RNA-seq from single-cell RNA-seq. As we already have 67 samples for which we have both single-cell RNA-seq and bulk RNA-seq data, we can use these samples to test our simulation process. We first want to be able to 'reconstruct' a bulk RNA-seq using all cells/genes of the single-cell RNA-seq from the same patient to assess the quality of our simulation. Then, by choosing the number of cells used, which cell types, and in which proportion: we could construct coherent and unique bulk RNA-seq raw count that could be used as new samples.

1.4.2 Classification of Melanoma Cells

In order to simulate samples with specific cell types in different proportion, we need to annotate our samples using the different states of melanoma cancer cells. We first use the previous annotations obtained by Philip Toma using a semi-supervised classifier: cellassign [21]. This method uses a given set of marker genes as prior knowledge to infer transcriptional states of cells. However, we are not completely satisfied with the results and as we obtained new sets of marker genes during the project we try to find other methods as UCell [3] to classify melanoma cells. Moreover, as we don't have any gold-standard reference we need to find a way to evaluate the quality of states assignments over different methods and set of marker genes.

1.5 Data & Code Availability

All the code used for the analysis and also additional resources can be found on the Boeva Lab shared drive at *./data/projects/Auguste_tumor_heterogeneity*. At this location you can also find the different results generated during this practical work. As mentioned previously we used raw data from TuPro, especially the single-cell RNA-seq and bulk RNA-seq raw counts. This dataset is accessible through the Leonhard Med cluster to which one needs to request access from the Tumor Profiler Board or the Lab managers.

2 Simulation of Bulk RNA-seq from single-cell RNA-seq

As mentioned earlier, to obtain good results when using matrix factorization methods as ICA or NNMF we need a large number of bulk RNA-seq raw counts. Moreover, we also need a large volume of bulk RNA-seq raw counts with their corresponding single-cell RNA-seq raw counts in order to train and validate a (semi-)supervised model which deconvolve multivariate signals. In this section, we will go through the different steps of the simulation.

2.1 Data Preprocessing and Simulation Methods

We first gather all the samples into one folder as each one is stored separately on Leonhard Med cluster. Moreover, each sample has different passes (pass1,2,3,4 or 5). Each of them stored the results of an updated analysis pipeline. Therefore, we decided to always use the latest pass available for a given sample as it contains the most reliable data. Single-cell RNA-seq raw counts are stored under .h5 files. In order to read them and to perform some data preprocessing, we used **Scanpy** python package or **hdf5r** R package. The simulation is done using both R and python scripts and is based on 3 files: *simulation_pipeline_1.py*, *simulation_pipeline_2.py* and *gc_normalization_1.r*. To simplify the process, we also made available a bash script, *simulation_pipeline.sh*, allowing the user to simulate a bulk RNA-seq in the simplest way.

2.1.1 single-cell RNA-seq De-noising: DCA

When one performs single-cell RNA-sequencing, noise due to amplification and dropout may obstruct analysis. The most known de-noising tool for high-dimensional scRNA-seq data is 'MAGIC' [5]. However, thanks to a presentation during the semester, we decide to use another tool: DCA [7]. DCA is also a de-noising tool for high-dimensinal scRNA-seq data. It is a deep count autoencoder network that takes the count distribution, overdispersion, and sparsity of the data into account using a negative binomial noise model with or without zero-inflation. According to the authors, DCA is supposed to give better results than MAGIC and also run faster for bigger dataset as the method scales linearly with the number of cells. We apply DCA on single-cell RNA-seq samples before running any other preprocessing methods.

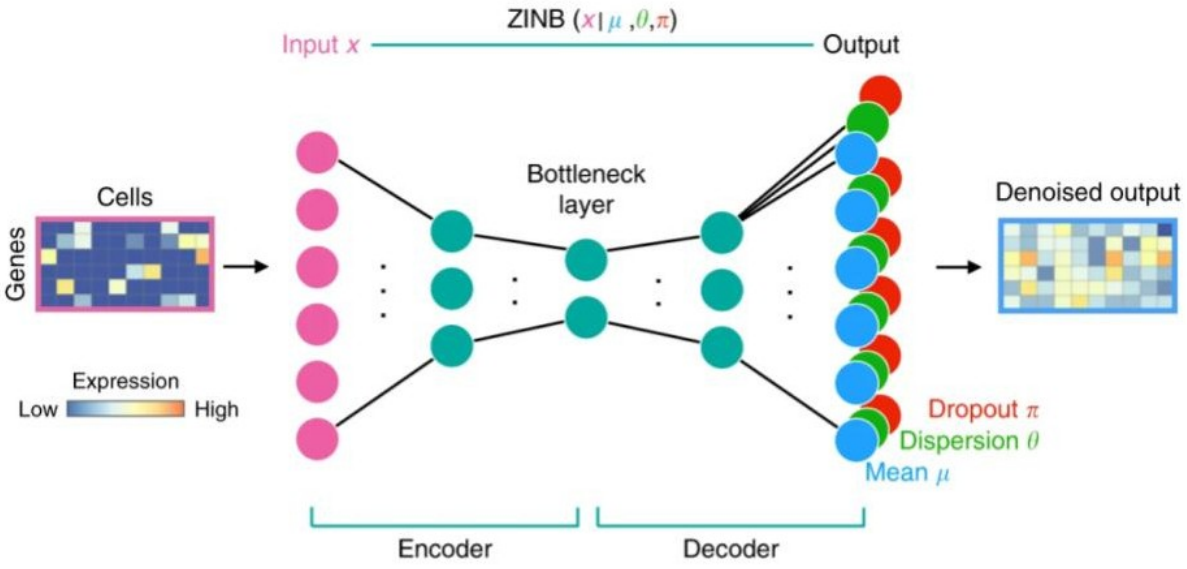


Figure 4: Schematic of DCA workflow, from [7]

2.1.2 Filtering Cells and Genes

In order to keep only relevant cells and genes for our simuluation, we used the methods *filter_cells* and *filter_genes* from the Scanpy package.

- Cells: we decided to only keep cells which have more than 500 expressend genes and a number of read counts between 1,500 and 40,000.
- Genes: we decided to only keep genes which are expressed in at least 5 cells.

2.1.3 Including Annotations

To simulate bulk RNA-seq raw counts with only certain cell types in specific proportion, we include cell annotations to our single-cell RNA-seq counts matrix. We use the annotations available on the shared drive under Philip Toma folder [1]. Cell types (Pigmented, Invasive, NCSC, and SMC) were assigned using a set of marker genes proposed by Rambow et al [15] and a probabilistic method named cellassign [21]. In the future, it should be easy to adapt the code in order to include other annotations instead.

2.1.4 Filtering Healthy Cells

From one single-cell RNA-seq raw counts we want to be able to build different –but representative– bulk RNA-seq raw counts. Therefore it is interesting to choose whether or not we want to use healthy cells in our simulation. We used the results from InferCNV package obtained by Philip Toma (available on the shared drive at *./data/projects/Philip_Tumor_heterogeneity*) in order to decide whether or not a cell is healthy. For each sample, we have two files: *inferred_healthy.txt and inferred_mel.txt* which respectively contain the list of healthy and melanoma cells for a given patient.

[1] */Philip_Tumor_heterogeneity/samplename_output_dir/results/assignment4_cell_types_new.txt*

2.1.5 Filtering Cancers Cells

As mentioned earlier, we include annotations to our samples. Based on these annotations, you can choose which types you want to use in the simulatiom of your bulk RNA-seq raw counts. At the moment you can choose between the following 4 types: Pigmented, Invasive, NCSC, and SMC. If new annotations are made available it should be easy to modify the code to integrate others transcriptional states.

2.1.6 Number of cells

In order to simulate more bulk RNA-seq samples from a single scRNA-seq sample we also add the possibility to choose the % of cells used for the simulation. You can specify a number between 0 and 1 –which defines the percentage of cell use– in the *simulation_pipeline_1.py* file; 1 being 100% and 0 being 0%.

2.1.7 Aggregation

Once all previously mentioned methods have been applied we can finally construct our simulated bulk RNA-seq sample. The selected single-cell expression profiles for every cell type are aggregated by summing their expression values, to yield the artificial bulk expression profile for this sample. This is the most intuitive method as bulk RNA-seq measures tissue gene expression levels that are average expression profiles of individual cells.

2.1.8 FPKM Normalization

Even if no gene length bias has been dectected for the single-cell RNA-seq dataset we decide to apply a Fragments Per Kilobase Million (**FPKM**) normalization [19]. We also apply this normalization to the bulk RNA-seq data as they are most likely exposed to gene length bias. We use FPKM as the raw count of two genes cannot be facing off if gene A is twice longer than gene B. Due to its length, the longest gene will have more chance to be sequenced than the short one. And in the end, for the same expression level, the longest gene will get more read than the shortest one [14]. The idea of FPKM is to take into account the gene lengths[2] when performing the normalization in order to remove any possible bias. Using this method we can face of the raw counts of two genes even if they have different length. FPKM normalization is computed as follow:

$$FPKM = \frac{\text{number of reads mapped to } gene_i \times 10^3 \times 10^6}{\text{Total number of mapped genes} \times gene_i \text{ length}}$$

2.1.9 GC Content Normalization

GC content bias describes the dependence between fragment count (read coverage) and GC content found in sequencing data.

2.1.9.1 Observing GC-content bias with loess regression

In order to observe the GC-content bias on both the single-cell RNA-seq and the bulk RNA-seq data we use the **loess** model which focused on the relationship between fragment count and GC composition[3] for particular bin sizes [20]. In order to both visualize the GC content bias and to correct it we used the R package **EDASeq**[4] which stand for 'Exploratory Data Analysis and Normalization for RNA-Seq'. We were able to visualize the GC content bias using the method *biasPlot*.

[2] Gene lengths can be found in the following file: *geneLengths_V32.txt*
[3] GC content for each gene can be found in the following file: *mart_export.txt*
[4] https://bioconductor.org/packages/release/bioc/html/EDASeq.html

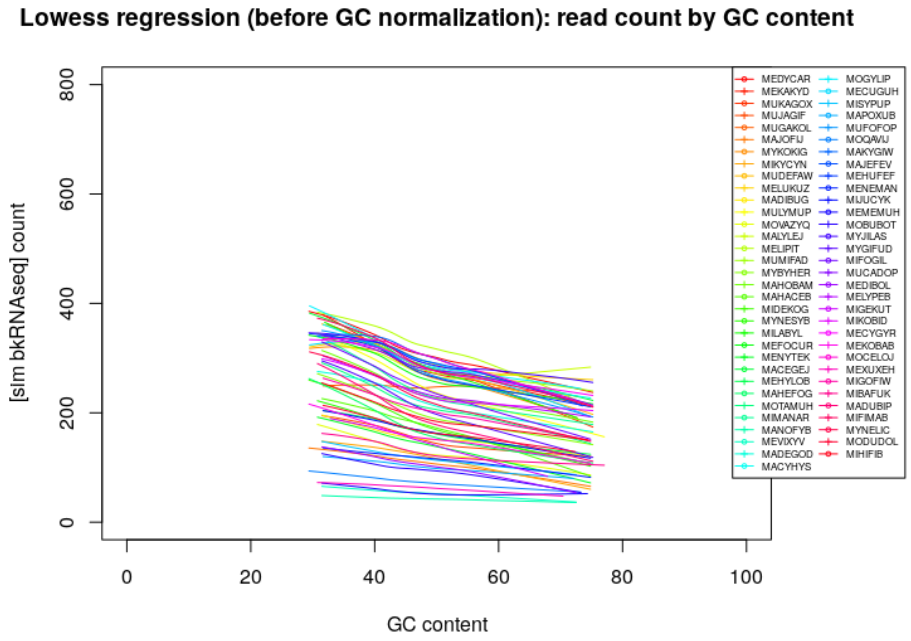


Figure 5: Visualization of the loess regression on GC-content for simulated bkRNA-seq samples before full-quantile normalization [obtained with the *biasPlot* method from EDAseq package]

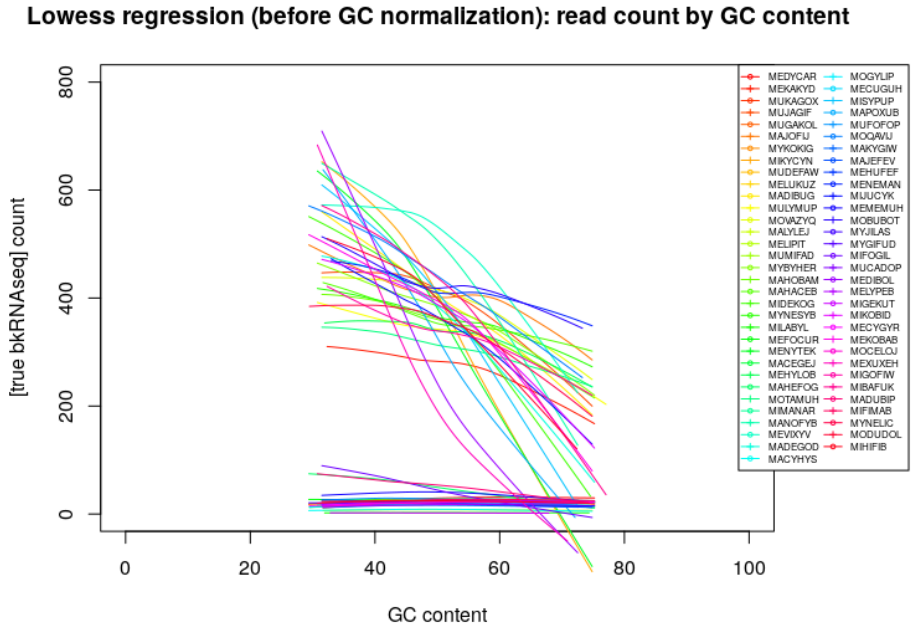


Figure 6: Visualization of the loess regression on GC-content for TuPro bkRNA-seq samples before full-quantile normalization [obtained with the *biasPlot* method from EDAseq package]

As one may observe on figures 7 and 8, both the simulated bulk RNA-seq samples from single-cell RNA-seq and the 'true' bulk RNA-sew samples from TuPro have a GC-content bias. We will now discuss how did we remove this bias using full-quantile normalization.

2.1.9.2 Correcting GC-content: Full-Quantile Normalization

To remove the GC-content bias from our dataset, we followed one of the methods described in [16]: within-lane GC-content normalization with **full-quantile normalization**. To perform full-quantile normalization genes are stratified according to GC-content. The quantiles of the read count distributions are then matched between GC-bins, by sorting counts within bins and then taking the median of quantiles across bins. Once again we used the EDAseq R package, and especially the *withinLaneNormalization* method.

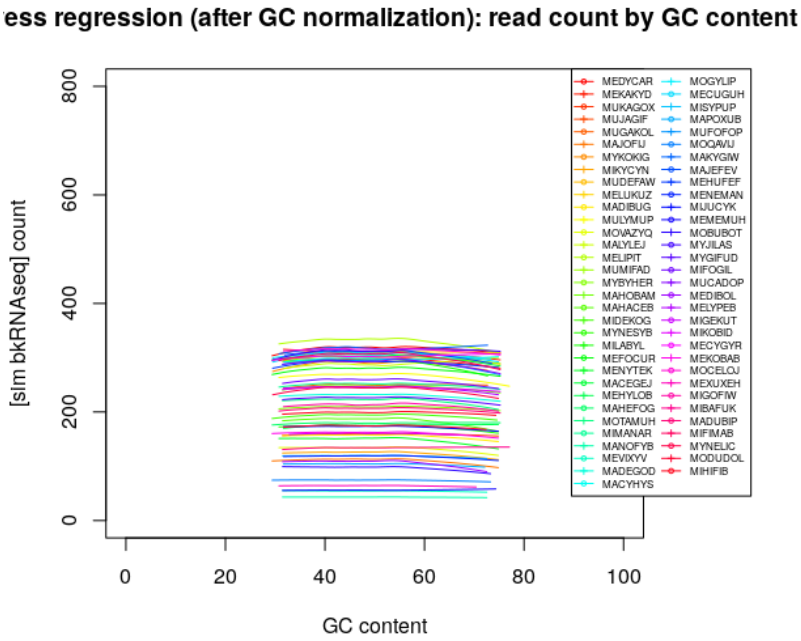


Figure 7: Visualization of the loess regression on GC-content for simulated bkRNA-seq samples after full-quantile normalization [obtained with the *biasPlot* method from EDAseq package]

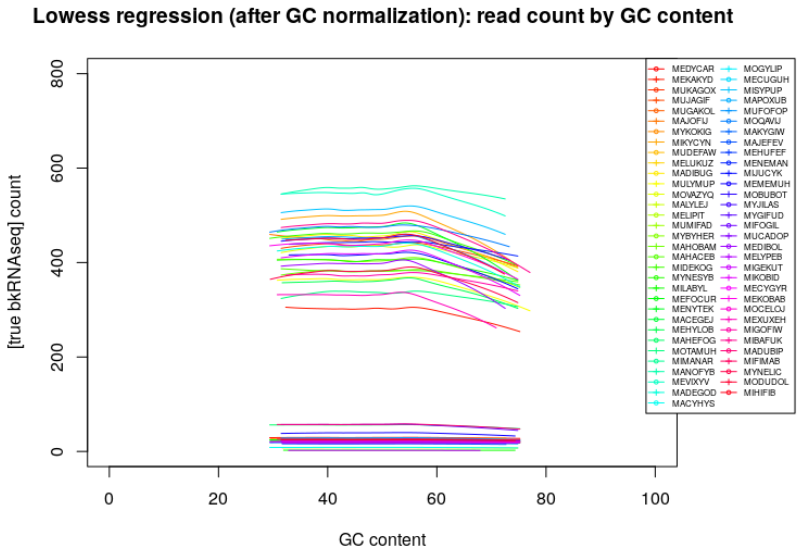


Figure 8: Visualization of the loess regression on GC-content for TuPro bkRNA-seq samples after full-quantile normalization [obtained with the *biasPlot* method from EDAseq package]

As one may observe on figures 7 and 8 that we were able to remove the GC-content bias for both the simulated bulk RNA-seq samples from single-cell RNA-seq and the 'true' bulk RNA-sew samples from TuPro. Therefore, we decide to add this normalization process to our simulation pipeline.

2.1.10 Logarithmic Transformation

Finally, in order to assess the quality of our simulated samples we need to compare them with their reference bulk RNA-seq sample. The expression values are transformed into logarithmic space by adding a pseudo count of 1 and then taking the Logarithm (natural) [4]. For each gene expressions value we apply the following transformation:

$$x' = \log(x + 1)$$

Note that we add a pseudo count of 1 in order to avoid negative values, especially for genes with 0 expression value.

2.2 Simulation Results

In section 2.1 we described the simulation pipeline used to simulate bulk RNA-seq data from single-cell RNA-seq data. To assess the quality of the simulations we now compare simulated bulk RNA-seq data using all cells (healthy and all types of cancer cells) with their associated original bulk RNA-seq sample. As mentioned earlier we have 67 samples for which we have both bulk RNA-seq data and single-cell RNA-seq data. We will then base our analysis on these samples. In order to compare two samples (simulated vs. original one), we use a scatter plot for the visualization and Spearman's correlation coefficient between the two raw counts as a measurement of the quality of the simulation. The higher the correlation, the better the simulation. Note that to be able to compute the Spearman's correlation coefficient we sometimes have to remove some genes that are not present in both the simulated and the original sample. We first perform the simulation without DCA denoising method and without GC-content normalization.

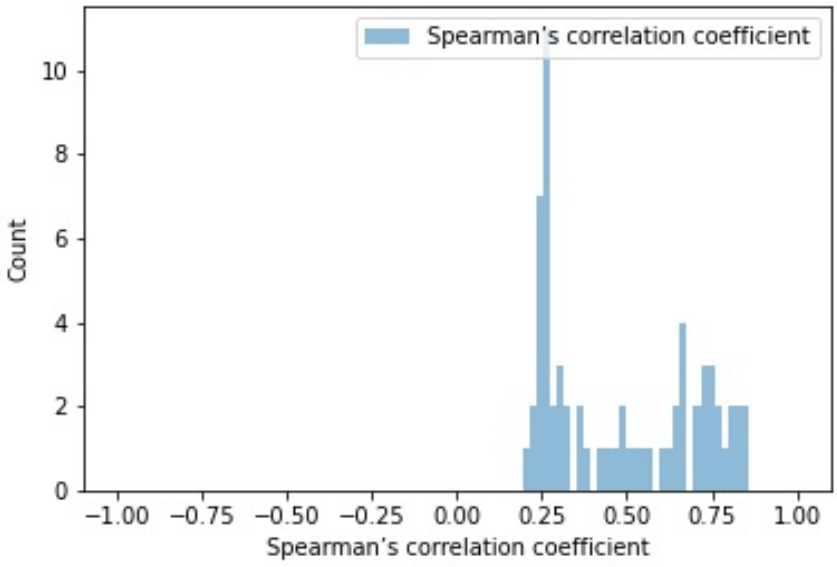
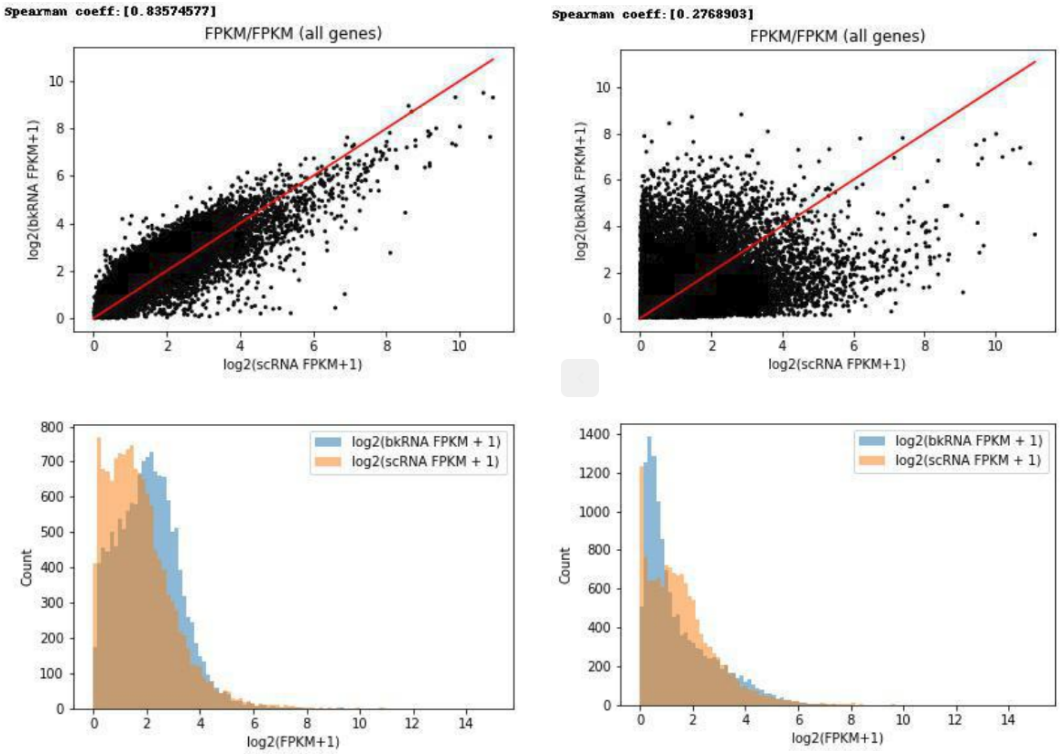


Figure 9: Distribution of the Spearman's correlation coefficient between simulated and original bulk RNA-seq data without DCA and GC-content normalization

On figure 9 one can observe that we obtain a good correlation for some samples ($\tilde{=}$ 0.75) but we also have samples for which the simulated and original data have low correlation ($\tilde{=}$ 0.25). For example, we can visualize the scatter plot and the comparison of the distribution of genes expression value between the simulated and the original data for both a 'good' and a 'bad' sample in figure 10.



(a) 'good' sample (corr=0.83)

(b) 'bad' sample (corr=0.27)

Figure 10: Scatter plot and comparison of the distribution of the gene expressions values between simulated and original bulk RNA-seq data

After obtaining these results we have two concerns. First, we want to understand why do samples have a way better correlation than others, and secondly we wonder how can we improve theses results. As we have the feeling that something is wrong with the bad samples we decide to divide our samples into 2 categories in order to analyze the impact of the different methods used to improve the results of the simulation (DCA and GC-content normalization). The first category includes samples for which the correlation between the simulated and the original bulk RNA data is greater or equal to 0.5. The second category then includes samples with a correlation lower than 0.5.

2.2.1 DCA

In order to improve the results of the simulation, we try to remove the dropout effect from the single-cell RNA-seq data using the method described in section 2.1.1: DCA. You can observe the results in figure 11.

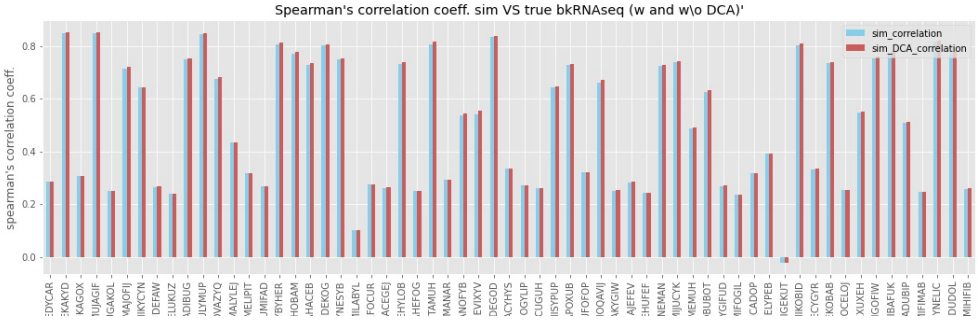


Figure 11: Spearman's correlation coefficient between simulated and original bulk RNA-seq data with and without DCA for denoising single-cell RNA-seq

As one might see, it is not evident to conclude on the impact of DCA with figure 11. Then we decide to compute the difference between correlations obtained without DCA implemented in the simulation

pipeline and correlations obtained with DCA implemented in the simulation pipeline. Moreover we divide our sample into two categories as mentioned before in order to see if the impact is the same for each group. We expect to see an improvement on the results as DCA should remove most of the false zero expression that can be seen as 'corrupted' data.

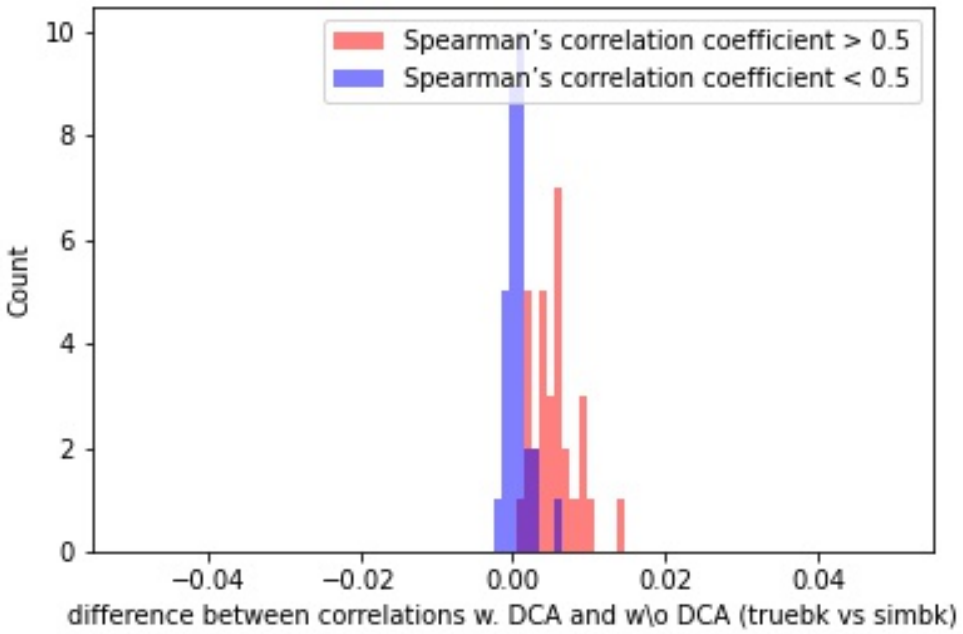


Figure 12: Difference of the Spearman's correlation coefficient between simulated and original bulk RNA-seq data with and without DCA for denoising single-cell RNA-seq

From figure 12 we can first deduce that using DCA as a tool for de-noising single-cell RNA-seq improves a bit the results. The improvement is low –between 0 and 0.02– but as DCA is not time consuming it could still be interesting to apply this method in order to obtain more relevant results. But the most interesting things is that we can observe a clear separation between the 'good' samples and the 'bad' ones. By looking closely at the results we observe that DCA slightly improves the correlation for all 'good' samples but it is not the case for the 'bad' ones. For most of them the difference between the correlation with and without DCA are the lowest overall and sometimes the correlation even decrease when using DCA.

2.2.2 GC-content normalization

In order to improve the results of the simulation we try to remove the GC-content bias. We apply the method described in section 2.1.9.2 for both our simulated bulk RNA data and the original one in order to correct the GC content bias. Once again we decided to compute the difference between correlations that were obtained without GC-content normalization and correlations obtained with GC-content normalization. The results can be found in figure 13. We expect to see an improvement as we demonstrated in section 2.1.9 that there is a GC-content bias and that we are able to correct it.

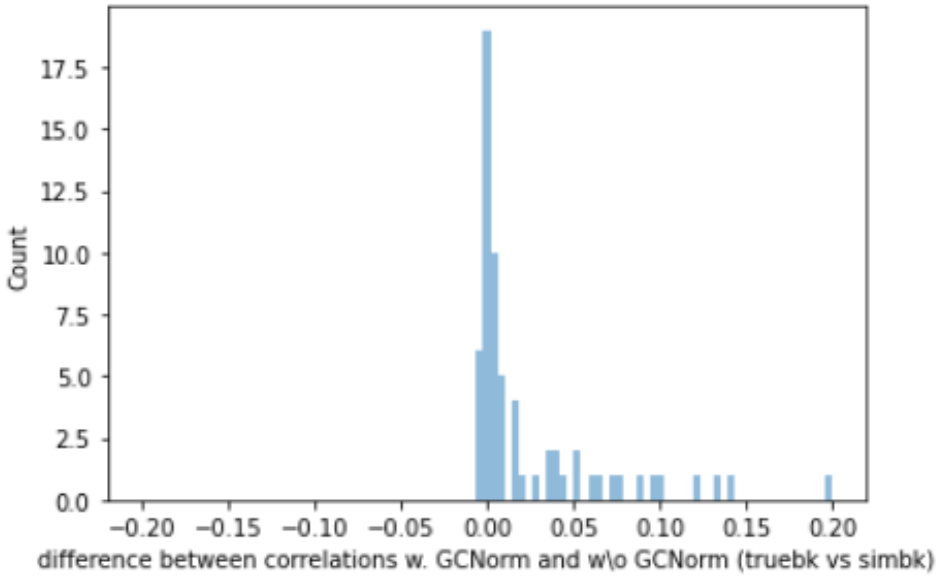


Figure 13: Difference of the Spearman's correlation coefficient between simulated and original bulk RNA-seq data with and without GC-content normalization

From figure 13 we can first observe that removing the GC content bias improves the results. This improvement –between 0 and 0.20– is 10 times higher than the one obtain with DCA. Moreover we have once more a big cluster of samples for which there is almost no improvement for some of them the correlation is even worst with the GC-content normalization. Once again we divide our sample into two categories as mentioned before in order to see if the impact is the same for each group. The results can be observe in figure 14.

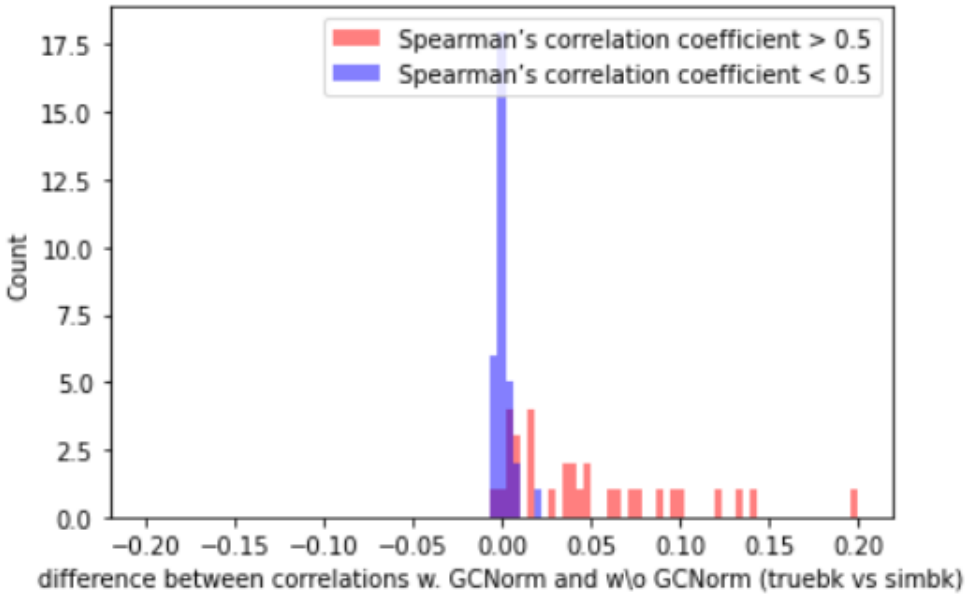


Figure 14: Difference of the Spearman's correlation coefficient between simulated and original bulk RNA-seq data with and without GC-content normalization

This is even clearer in figure 14 that we have a separation between the 'good' and the 'bad' samples. Samples with low correlation are the ones that don't improve with GC-content normalization, once again. This supported the idea that the poor results come from the quality of the samples more than the simulation process.

2.2.3 Discussions

During this first part of the project, we developed a simulation pipeline able to construct bulk RNA-seq raw counts from single-cell RNA-seq raw counts. In order to build different bulk RNA-seq samples from one single-cell RNA-seq sample, we have the possibility to use only specific cell types and also to choose the % of cells to use for the simulation. We assess the quality of this pipeline by reconstructing the original bulk RNA-seq from the single-cell RNA-seq (for the 67 samples for which we have both data) using all cells available. We then used Spearman's correlation coefficient between the simulated data and the original one in order to have a measure of quality. We were able to improve our results using specific pre-processing methods as DCA or GC content normalization. However, for some samples, we obtained bad correlations and were not able to improve them no matter what. Later in the project, we have been able to remove some of them as they did not pass the quality control. Unfortunately, there were always so-called valid samples for which we were not able to obtain good results. Despite our efforts, we were not able to find the reason why these samples did not work well.

3 Classification of Melanoma Cancer Cells and Evaluation

As mentioned earlier we have a dataset of 67 single-cell RNA-seq data without transcriptional state labels. In order to simulate bulk RNA-seq data using only some combination of cell types we first need to annotate the transcriptional state of melanoma cancer cells. Moreover, these annotations are useful in many other projects. When we started this project we had at our disposal the annotations obtained by Philip Toma using semi-supervised method cellassign[21] using the set of marker genes proposed by Rambow & al in [15]. As we were not completely satisfied with the results we tried to annotate the transcriptional state of melanoma cancer cells using other methods and other sets of marker genes.

In this section we will discuss the different methods available to determine the transcriptional state of melanoma cancer cells then we will see how can we evaluate theses annotations without gold-standard reference, and finally we will compare the results between the different methods based on these evaluations. The method used to determine the transcriptional state of melanoma cancer cells can be devised into 3 categories: supervised, semi-supervised and unsupervised methods. Supervised method as *CIPR* or *ClustifyR* rely on a labeled reference set for cell type assignments. However, for our data set we do not have any labeled reference set and therefore using these methods is not an option. Unsupervised methods as k-mean clustering are usually not efficient and do not take advantage of prior knowledge. Fortunately, Rambow & al observed that it was possible to annotate transcriptional state of melanoma cancer cells using sets of marker genes that correspond to different cell types. Therefore, semi-supervised methods that use a specified set of marker genes as prior knowledge were developed to assign the transcriptional state of melanoma cancer cells. As mentioned above, we already have the annotation of transcriptional state of melanoma cancer cells obtained with cellassign using the set of marker genes proposed by Rambow & al (which correspond to the following four transcriptional states: *invasive, pigmented, neural-crest stem-cell like (NCSC) and starved-like melanoma (SMC)*). We also have annotations of transcriptional state of melanoma cancer cells obtained with SCINA [22], another semi-supervised method. In this project we try to use a method named *UCELL* [3] in order to obtain these annotations. To compare UCell's results and the ones obtained previously with cellassign and SCINA we will use the same set of marker genes. But one should know that we were able to obtain different sets of marker genes specific to our dataset using method as debCAM. We will briefly observe the results obtained with this new sets of marker genes, especially the *'allexpression'* one.

3.1 Semi-supervised Method

In this section we will describe the two methods for which we already have the results: SCINA and cellassign. As we did not work directly with this method, we will simply explain how they work. We will then describe the new method we used: UCELL. For UCell we will also discuss the different combinations of hyper-parameter used in order to achieve the best possible results.

3.1.1 Cellassign

cellassign [21] is based on Bayesian statistical model optimized with the expectation maximization (EM) algorithm. The model use four inputs:

- ρ : a marker/cell type matrxix (binary signature matrix)
- Y : single-cell count matrix
- s : cell size factor used for normalization purposed
- (optional) X : covariate matrix used to account for batch effect

With the model inputs it is then possible to infer the probabilities that a cellz_n is of a given cell type c. This probability can be described as follow: $p\left(z_n = c|Y, \hat{\Theta}\right)$ where $\hat{\Theta}$ are the maximum a posteriori probability (MAP) estimates of the model parameters. They then used the expected genes expression $\mathbb{E}\left[y_{ng}|z_n = c\right] = \mu_{ngc}$ in order to predict the probability descrbibed above. This expressions is a combination of the four inputs and can be described as follow:

$$\overbrace{\log \mu_{ngc}}^{\text{Log mean expression}} = \underbrace{\log s_n}_{\text{Cell size factor}} + \overbrace{\delta_{gc}\rho_{gc}}^{\text{Cell type specific}} + \underbrace{\beta_{g0}}_{\text{Base expression}} + \overbrace{\sum_{p=1}^{P} \beta_{gp}x_{pn}}^{\text{Other covariates (incl.batch)}}$$

You can find an illustrated summary of the cellassing model given by their authors on figure 15.

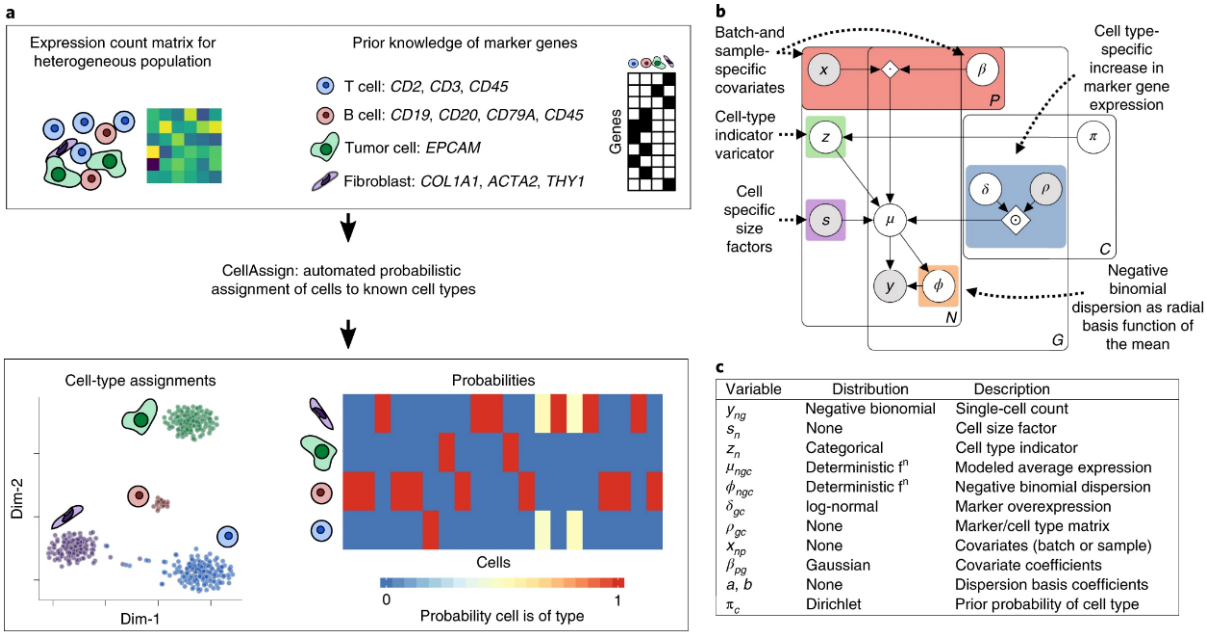


Figure 15: **a,** Overview of the functionalities offer by cellassign. **b,** relationship between the different random variable of the model. **c,** prior distribution of all the model's random variable.

3.1.2 SCINA

Unlike cellassign, SCINA [22] use a bi-modal distribution assumption of the expression of the signature genes in order to compute the probability that a cell is of type c. The SCINA model assumes that there is a bi-modal distribution for each signature gene, with the higher mode corresponding to the cell types, in which this gene is designated as a signature, and the lower mode corresponding to all the other cell types. Once again it used EM algorithm in order to optimize the parameters of the bi-modal distribution. It estimates and optimized the following parameters:

- probabilities that a cell is of type c
- the 2 distribution center $\mu_{1,r}$ and $\mu_{2,r}$ of the bi-modal distribution for each marker genes r
- the covariance matrices $\Sigma_{1,r}$ and $\Sigma_{1,r}$ of the bi-modal distribution for each marker genes r

You can find an illustrated summary of the SCINA model given by their authors on figure 16. One should know that SCINA method runs way faster than cellassign (10 to 100 times faster).

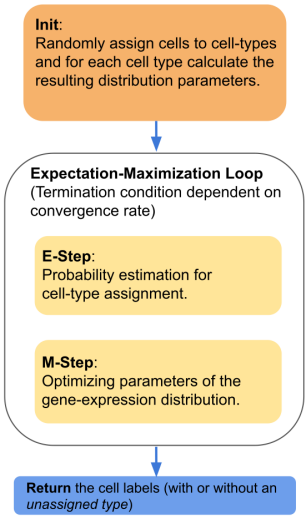


Figure 16: Illustrated overview of the SCINA model

3.1.3 UCell

3.1.3.1 Method

In this section we will discuss the UCell [3] method. Unlike cellassign and SCINA, UCell does not give the probability that a cell is of type c. Instead it gives a score in the range *[0,1]* for each cell types. UCell calculates gene signature scores for single-cell RNA-seq data based on the Mann-Whitney U statistic [10]. The U statistic is very similar to area-under-curve (AUC) statistic for ROC [17] curves which is the method used by another scoring model: AUCell [2]. Therefore we expect UCell to obtain similar results to AUCell. But the main advantage of UCell is that it used 100 times less memory and run 3 times faster than AUCell. Therefore UCell is suitable for every machine even the ones with low computing power.

UCell method requires 3 inputs:

- $\boldsymbol{X}$: a $g \times c$ matrix representing the gene expressions for g genes in c cells
- $\boldsymbol{s}$: signatures/marker gene sets for each transcriptional states
- r_{max} : the max rank value (1500 by default)

It then computes the relative ranks $\boldsymbol{R}$ by sorting each column in $\boldsymbol{X}$ (eq. a ranked list of genes –for each cell in the sample– by expression value in descending order). As transcript counts matrices contain many zeros, which results in a long tail of bottom-ranking genes, it sets every the rank of every genes at max rank + 1 in order to mitigate the uninformative tail. In other words, it set $r_{g,c} = r_{max}$ for all $r_{g,c} > r_{max}$. Finally, to evaluate the Uscore $\boldsymbol{U'}$ for a gene signature composed of n genes (s_1, ..., s_n for each cells j in $\boldsymbol{X}$ it used the following formula:

$$U'_j = \frac{1-U_j}{n \times r_{max}}$$

where U_j is the Mann-Whitney U statistic calculated by:

$$U_j = \sum_{i=1}^{n} r'_{i,j} - \frac{n(n+1)}{2}$$

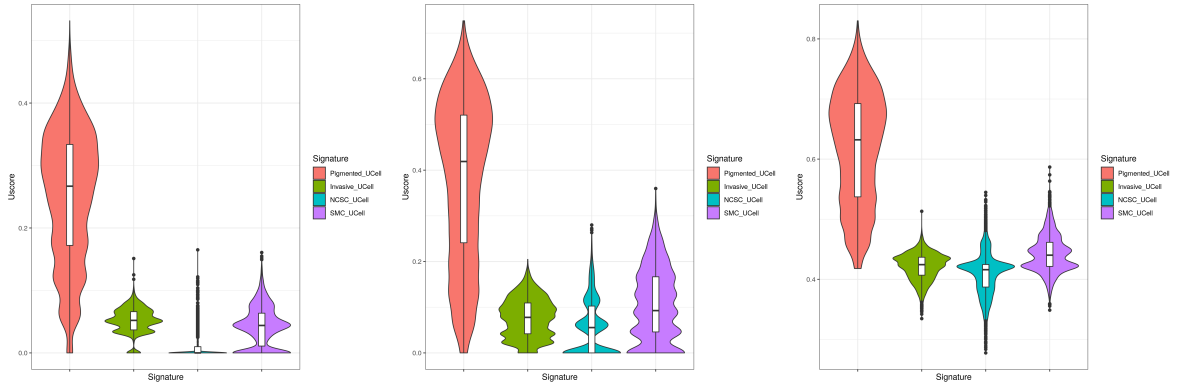
and $\boldsymbol{R'}$ is obtained by sub-setting $\boldsymbol{R}$ on the genes in signature $\boldsymbol{s}$.

3.1.3.2 Experiments

One might observe that the only parameter we can optimize is the max rank value: r_{max}. Remembering that the max rank value is intended to avoid a long tail of bottom-ranking gene, we try three different values:

- ***Basic***: the default value: 1,500.
- ***10kRank***: as we have –in average– way more than 1500 genes expressed by cell in our dataset, we try a way higher value: 10,000.
- ***Adaptive***: finally, in order to have a specific max rank for each sample we compute the average number of genes expressed per cell in a sample. We then set the max rank to this value for this specific sample. We repeat this operation for each sample.

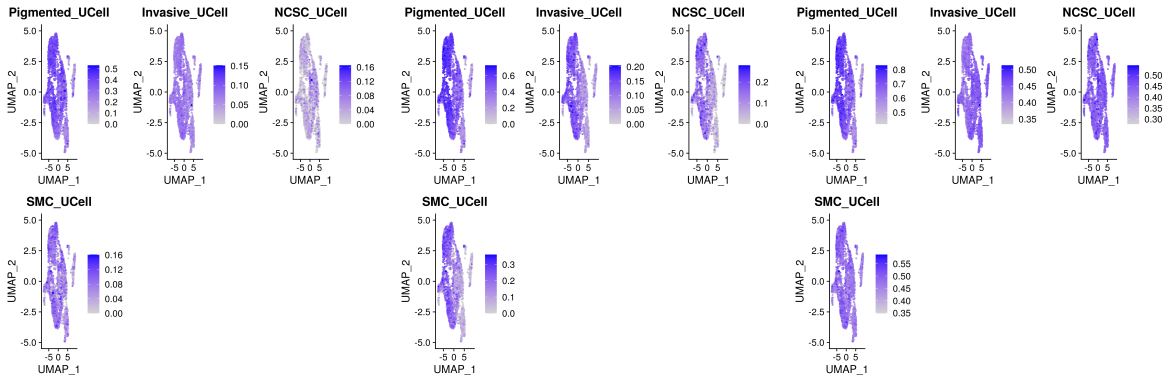
As mentioned earlier, UCell assigns a Uscore for each transcriptional states to each cell in a sample. Therefore on way to visualize the results for a specific sample is to plot the distribution of predicted scores. In figure 17 you can observe the distribution of the predicted scores for one sample.



(a) $r_{max} = 1500$ (b) $r_{max} = 10000$ (c) $r_{max} =$ adaptive

Figure 17: Distribution of the predicted Uscore for three different values of max rank (sample: MEDYCAR)

Moreover, we can see how the signatures distribute on low-dimensional representations of the data by creating a Seurat object from the expression matrix and generating PCA and UMAP embeddings. This is what we obtain using the same sample as in figure 17:



(a) $r_{max} = 1500$ (b) $r_{max} = 10000$ (c) $r_{max} =$ adaptive

Figure 18: UCell score distribution in UMAP space for three different values of max rank (sample: MEDYCAR)

Using these results we annotate our cells using 2 different methods. The first method consists of comparing the Uscores of each signature for each cell and selecting the highest one. The other method is similar but for each sample we normalize the Uscores before doing the comparison. Due to the lack of time we use basic approaches for interpretation of the Uscores. One might find it interesting to dig in the interpretation of the Uscore in order to improve the annotation.

3.2 Evaluation of State Assignments

We discussed previously the absence of a gold-standard reference that could have been used to evaluate cell assignments with metrics as specificity or accuracy. Therefore we need to find another way to evaluate the cell assignments and to compare the results between the different methods presented in section 3.1. In this section we will then talk about the methods used to evaluate state assignments and then compare our different semi-supervised methods together based on these evaluations.

3.2.1 Methods

We can see the state assignments problem as a clustering/classification problem and therefore we can use metrics that evaluate the shape and the separation of the clusters to evaluate the quality of our assignments. Each semi-supervised method used prior knowledge of the marker genes in order to determine the transcriptional state of each cell. We can assume that for any of these methods, cells assigned with the same transcriptional state should be 'similar' with relation to the expression value of their respective sets of marker genes. Following the same logic, cells assigned with different transcriptional states should be less similar with relation to the expression value of their respective sets of marker genes. We decide to use *intra-cluster compactness* and *inter-cluster complexity* as evaluation metrics. Note that in the following section we only consider marker genes and remove other genes from the calculation.

3.2.1.1 Inter-Cluster Complexity

As described in [1], the complexity can be seen as the pairwise similarity between cell populations. They demonstrated that when the complexity increases the performance decreases (eq. the accuracy of state assignments decreases). Therefore we want our complexity as low as possible. The term inter-cluster complexity can be derived as follow:

$$\text{Complexity} = \text{mean}\left(\max_{\forall i, i \neq j} \text{corr}\left(\text{avg}_{C_i}, \text{avg}_{C_j}\right)\right)$$

where avg_{C_i} is the average expression of all marker genes for state C_i and the $corr(a, b)$ make reference to the Pearson correlation.
This formula give the complexity of the entire data set. We are interesting in computing the inter-cluster complexity for each transcriptional state separately. Therefore we define the term inter-cluster complexity for a transcriptional state k as:

$$\text{Complexity}_\text{k} = \max_{\forall r \in R} \frac{1}{n_k} \sum_{i=1}^{n_k} \text{corr}(c_r, Y_{k,i})$$

where n_k is the number of cells in state k, c_r denotes the mean gene expression values of cells not labelled with state k, R denotes the set of transcriptional states without state k and Y_k denote the gene expression matrix of marker genes for cells annotated with state k.

3.2.1.2 Intra-Cluster Compactness

As described in [6], the compactness can be seen as the degree of intraclass similarity for each cell type in the dataset. We want the compactness to be as high as possible. The term intra-cluster compactness can be derived as follow:

$$\text{Compactness} = \frac{1}{n} \sum_{i=1}^{n} \left(\frac{1}{m_i} \sum_{j=1}^{m_i} \text{corr}(avg_i, value_{i,j}) \right)$$

where n is the number of cell type, m_i is the cell number of the i-th cell type, $corr()$ is the calculated Pearson correlation coefficient, avg_i is the average expression of marker genes for the i-th cell type, and $value_{i,j}$ is the j-th cell expression in the i-th cell type

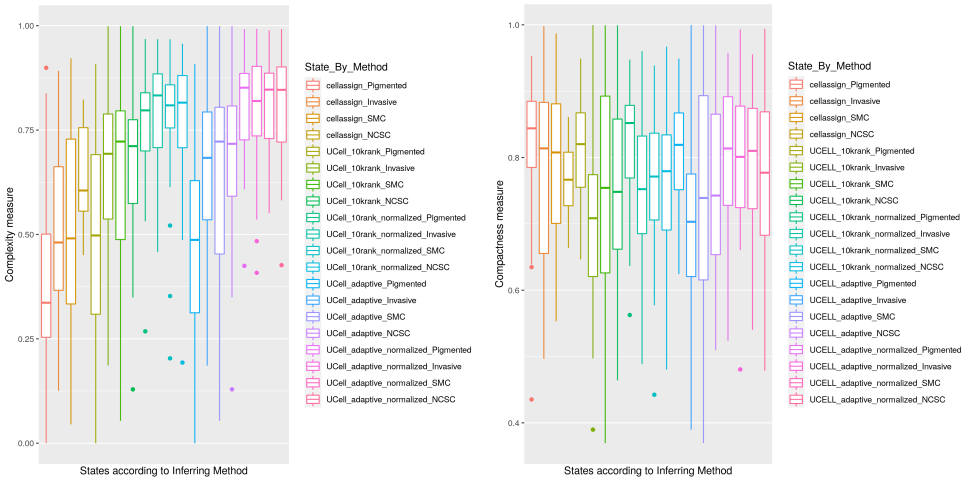
This formula give the compactness of the entire data set. We are interesting in computing the intra-cluster compactness for each transcriptional state separately. Therefore we define the term of intra-cluster compactness for a transcriptional state k as:

$$\text{Compactness}_\text{k} = \frac{1}{n_k} \sum_{i=1}^{n_k} corr(c_k, Y_{k,i})$$

where n_k is the number of cells in state k, c_k denotes the mean gene expression values of cells labelled with state k and Y_k denote the gene expression matrix of marker genes for cells annotated with state k.

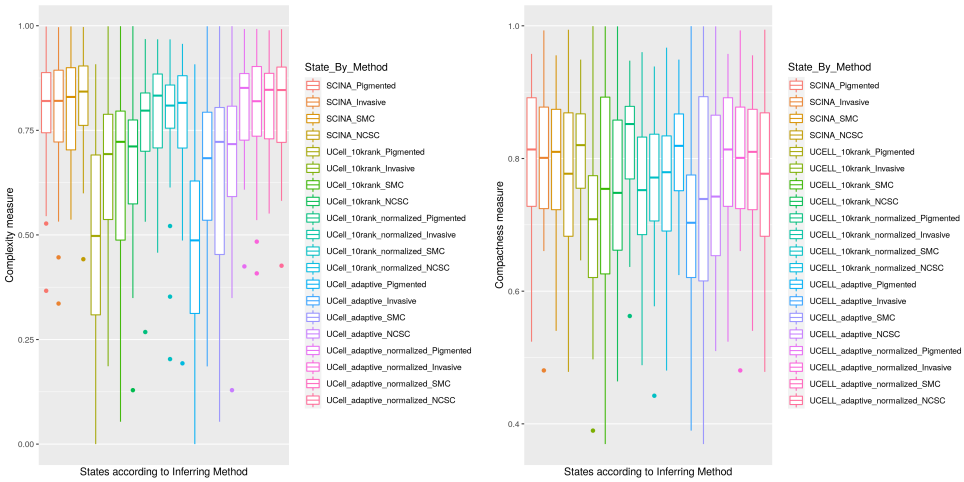
3.2.2 Results

We compute the 2 metrics – inter-cluster complexity and intra-cluster compactness – for both cellassign and SCINA annotations. We also compute the 2 metrics for the different experiments mentioned in section 3.1.3.2. In order to visualize the complexity/compactness of the dataset in its entirety, we use box plot to visualize the results.



(a) Complexity (b) Compactness

Figure 19: Boxplot comparing complexity and compactness for cellassign and 4 configurations of UCell.



(a) Complexity (b) Compactness

Figure 20: Boxplot comparing complexity and compactness for SCINA and 4 configurations of UCell.

From figure 19 and 20 we can deduce interesting information. First we observe that in terms of complexity, UCell outperforms SCINA no matter the configuration used. Note that the best results are achieved with 'non-normalized' configurations. Unfortunately, when comparing the results with cellassign, none of the configurations outperformed cellassign's results in terms of both complexity and compactness. We observe –as mentioned in Philip's thesis– that SCINA assigned to existing transcriptional states quite evenly, increasing the likelihood of cells from the same true state to be annotated with different states, and therefore decreasing cluster separation. This is also why we observe similar complexity for all existing transcriptional states when using SCINA. However, even if UCell does not outperform cellassign, we observe similar shapes in the distribution of complexity over the different transcriptional states for both methods. This means that we are able to catch inconsistent quality of annotations between the different transcriptional states. Precisely, both cellassign and UCell results demonstrate that they are able to annotate Pigmented cells with higher 'confidence' than the three others types. It is hard to determine with precision why we obtain these results but several hypotheses are offered to us. First, it can be related to the set of markers genes given as input. It might be possible that the marker genes used are not as representative as one might think for some transcriptional states. It is therefore a good idea to use specific marker genes obtained from our dataset. This could solve 3 problems: first it might give similar distributions of complexity for each transcriptional states. Secondly, it could reduce the overall complexity. Moreover, it can reduce the difference of complexity between samples. In fact we observe on figures 19 and 20 that the complexity can variate from 0 to 1 depending on the sample. In order to assess this assumption, we re-do the same experiment using another set of marker genes specifically derived from our dataset. As mentioned earlier this dataset is obtained using debCAM package.

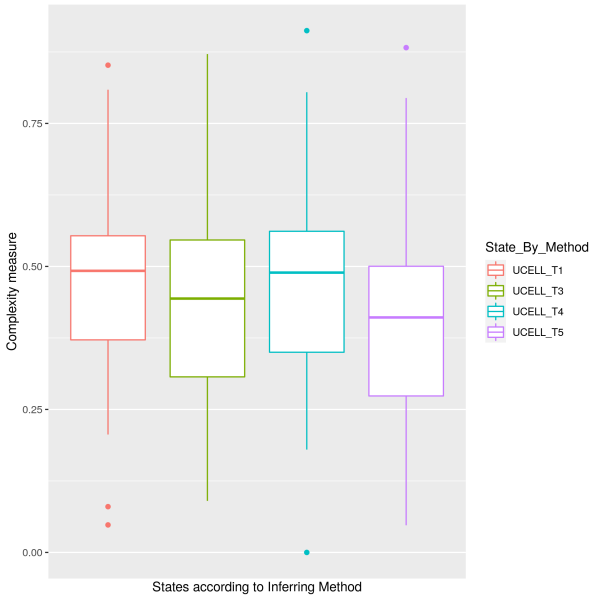


Figure 21: Complexity of our dataset using UCell with *'allexpression'* marker genes obtained with debCAM.

By looking at figure 21 we observe that using marker genes specific to our dataset allow us to obtain way better results in term of complexity and also to obtain similar complexity between the different trasncriptional states. It means that we have consistent marker genes for each transcriptional state and one should work with these new annotations from now. Unfortunately, we still have a huge difference of complexity between samples that we are not able to explain.

We conclude that one should use cellassign when annotating the transcriptional state of melanoma cells. However, UCell gives interesting results as it is able to replicate some characteristics of cellassign results where SCINA is not. Therefore it could be interesting to continue to work on the optimization of the r_{max} parameter and the interpretation of the Uscores as UCell use way less memory and run way faster than cellassign. Moreover, we demonstrate that it is very promising to use a set of marker genes that are specific to our dataset as it could results in better complexity which means better cell assignments.

4 Conclusion and Discussion

During this practical work, we laid the groundwork for the analysis of intra-tumor heterogeneity from bulk RNA-seq data. In order to conduct this work, we had at our disposal a dataset including both single-cell RNA-seq and bulk RNA-seq data. Some conventional techniques as Non-Negative Matrix Factorization or Independent Component Analysis has been used without producing sufficiently satisfactory results. A potential reason for this is the lack of bulk RNA-seq data. Moreover, in order to use state of the art machine/deep learning solution, we also need more samples with references. Therefore our first task was to create a simulation pipeline which from a single-cell RNA-seq can simulate multiple representative bulk RNA-seq. This was made possible by allowing the choice of cell types and the percentage of cells used for the simulation. To obtain results as accurate and relevant as possible, we used different pre-processing techniques as for instance the use of DCA to remove the dropout effect. We also have highlighted the presence of the GC-content bias in both single-cell RNA-seq and bulk RNA-seq data. We performed within-lane GC-content normalization with full-quantile normalization in order to correct the bias. We also demonstrated its impact on the simulation results. However, for some samples we were not able to obtain satisfying enough results no matter what methods were used. One should be interested in taking a closer look at these samples and see what alternative data could be used to improve the performance of the simulation.

On the other hand, we also worked on the classification of melanoma cancer cells. Without gold-standard reference, using supervised was not a possibility. Therefore we used UCell, a semi-supervised method, which takes advantage of prior knowledge on marker genes. We compared the results of UCell with two other methods used previously: SCINA and cellassign. If we manage to outperform SCINA, cellassign still gives the best results according to our evaluation's metrics: inter-cluster complexity and intra-cluster compactness. However, we were able to highlight some clues that suggest that by continuing to optimize UCell's hyper-parameters and by improving the interpretation of the Uscores, one could get results as good or better than cellassign. All this while using much less computational resources. Moreover, we have highlighted that using a set of marker genes specifically designed for our dataset could help to improve the classification of melanoma cells. Combining the results of the future annotations with the simulation process could help improving NNMF or ICA and be a starting point for building a new deconvolution model.

References

[1] Michielsen-L. Cats D. et al. Abdelaal, T. A comparison of automatic cell identification methods for single-cell rna sequencing data. *Genome Biol*, 2019.

[2] González-Blas C. Moerman T. et al. Aibar, S. Scenic: single-cell regulatory network inference and clustering. *Nat Methods*, 2018.

[3] Massimo Andreatta and Santiago J. Carmona. Ucell: Robust and scalable single-cell gene signature scoring. *Computational and Structural Biotechnology Journal*, 19:3796–3798, 2021.

[4] A Booeshaghi and Lior Pachter. Normalization of single-cell rna-seq counts by log(x + 1)* or log(1 + x). *Bioinformatics (Oxford, England)*, 03 2021.

[5] David van Dijk, Juozas Nainys, Roshan Sharma, Pooja Kaithail, Ambrose J. Carr, Kevin R. Moon, Linas Mazutis, Guy Wolf, Smita Krishnaswamy, and Dana Pe'er. Magic: A diffusion-based imputation method reveals gene-gene interactions in single-cell rna-sequencing data. *bioRxiv*, 2017.

[6] Bin Duan, Chenyu Zhu, Guohui Chuai, Chen Tang, Xiaohan Chen, Shaoqi Chen, Shaliu Fu, Gaoyang Li, and Qi Liu. Learning for single-cell assignment. *Science Advances*, 6(44), 2020.

[7] Simon-L.M. Mircea M. et al. Eraslan, G. Single-cell rna-seq denoising using a deep count autoencoder. *Nat Methods*, 2019.

[8] Douglas Hanahan and Robert Weinberg. Hallmarks of cancer: The next generation. *Cell*, 144:646–74, 03 2011.

[9] Daniel Lee and H. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401:788–91, 11 1999.

[10] H. B. Mann and D. R. Whitney. On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other. *The Annals of Mathematical Statistics*, 18(1):50 – 60, 1947.

[11] Oskar Marin-Bejar, Aljosja Rogiers, Michael Dewaele, Julia Femel, Panagiotis Karras, Joanna Pozniak, Greet Bervoets, Nina Van Raemdonck, Dennis Pedri, Toon Swings, Jonas Demeulemeester, Sara Vander Borght, Francesca Bosisio, Joost J. van den Oord, Isabelle Vanden Bempt, Diether Lambrechts, Thierry Voet, Oliver Bechter, Helen Rizos, Mitch Levesque, Eleonora Leucci, Amanda W. Lund, Florian Rambow, and Jean-Christophe Marine. A neural crest stem cell-like state drives nongenetic resistance to targeted therapy in melanoma. *bioRxiv*, 2020.

[12] Almendro V. Polyak K. Marusyk, A. Intra-tumour heterogeneity: a looking glass for cancer? *Nat Rev Cancer*, 2012.

[13] Swanton C. McGranahan N. Biological and therapeutic impact of intratumor heterogeneity in cancer evolution. *Cancer Cell*, 2015.

[14] Oshlack A. Phipson B, Zappia L. Gene length and detection bias in single cell rna sequencing protocols. 2017.

[15] Marin-Bejar O Aibar S Femel J Dewaele M Karras P Brown D Chang YH Debiec-Rychter M Adriaens C Radaelli E Wolter P Bechter O Dummer R Levesque M Piris A Frederick DT Boland G Flaherty KT van den Oord J Voet T Aerts S Lund AW Marine JC Rambow F, Rogiers A. Toward minimal residual disease-directed therapy in melanoma. 2018.

[16] Schwartz K. Sherlock-G. et al. Risso, D. Gc-content normalization for rna-seq data. *BMC Bioinformatics*, 2011.

[17] N. E. Graham S. J. Mason. Areas beneath the relative operating characteristics (roc) and relative operating levels (rol) curves: Statistical significance and interpretation. 2002.

[18] Rebecca L. Siegel, Kimberly D. Miller, and Ahmedin Jemal. Cancer statistics, 2020. *CA: A Cancer Journal for Clinicians*, 70(1):7–30, 2020.

[19] Cole Trapnell, Brian Williams, Geo Pertea, Ali Mortazavi, Gordon Kwan, Marijke Baren, Steven Salzberg, Barbara Wold, and Lior Pachter. Transcript assembly and quantification by rna-seq reveals unannotated transcripts and isoform switching during cell differentiation. *Nature Biotechnology*, 28, 05 2010.

[20] Terence P. Speed Yuval Benjamini. Summarizing and correcting the gc content bias in high-throughput sequencing. *Nucleic Acids Research*, 2012.

[21] O'Flanagan C. Chavez E.A. et al Zhang, A.W. Probabilistic cell-type assignment of single-cell rna-seq for tumor microenvironment profiling. *Nat Methods*, 2019.

[22] Zhong X Choi JH Ma Y Wang S Mahrt E Guo W Stawiski EW Modrusan Z-Seshagiri S Kapur P Hon GC Brugarolas J Wang T. Zhang Z, Luo D. Scina: A semi-supervised subtyping algorithm of single cells and bulk samples. 2019.