Institute of Statisitcs, Biostatistics and Actuarial Sciences

LDATA2010: Information Visualisation

Visualization Dashboard for single-cell RNA transcriptomics data

Authors:
LALIEU Justin (DATI2M- 74961800)
WARNAUTS Aymeric (DATS2M- 87031800)

Professor:
LEE John
Assistants:
LAMBERT Pierre
JOOS DE TER BEERST Victor

September - December 2022

1 Introduction

The human check of single-cell RNA transcriptomics data stays primordial as researchers should use modern data analysis techniques that incorporate visual feedback to verify the appropriateness of their models. And so, visualization should be an integral component of differential expression analysis. That can be done in order to detect normalization issues, differential expression designation problems, and common analysis errors, adding statistical analysis through interactive graphics. Moreover, researchers often conduct RNA-seq studies to identify differentially expressed genes between treatment groups through clustering or linear models. In fact, the most effective approach to data analysis is to iterate between models and visuals, and enhance the appropriateness of applied models based on feedback from visuals.
A parallel coordinate plot draws each row (gene) as a line. The cells in the same ogranisms have the same DNA , but still can be very different. Single cell data allows to understand that differences on very delicate level. In our case, data - results of single cell RNA sequencing, i.e. rows - correspond to cells, columns to genes.
As it's said in the project description, the cells belong to three general families: **non-neurons, excitatory neurons, and inhibitory neurons**. Further distinctions between cells appear within each of these families for a total of **133** cell types.
As we were asked to give as short response to the questions such as **"Is the dataset complete?"**, **"What should you do with missing data?"** or **"How should you deal with categorical values ?"**, we will advance that apparently the dataset doesn't present any missing values or **NA** but it's clear that some genes are flat in every cell with values near **0**, so we will make an option available for the user of our interface to remove these genes.
As the categorical values appears in the **cell type** and **color** features, further distinctions between cells appear within each of these families for a total of 133 cell types. The color corresponds to the cell type with **133** unique values. This hexadecimal code for the colour given by biologists to each cell type will be used in our further analysis to visualize groups.

2 Software user guide

2.1 Launch of the program

To get the opportunity to run our specialized visualization dashboard for single-cell RNA transcriptomics data's, be sure to download an updated version of a python software launcher and make sure that all the required modules are installed on your computer by checking the **requirements.txt** file. In order to execute the program, make sure that the **transcriptomics_data.csv** file is in the same folder than the **main.py** file, then run the file named **main.py** by entering **python main.py** in a terminal and in our project folder. Everything is ready, you just have to copy the link to access the development server running which is **http://127.0.0.1:8050/**. This link may change as it's under development and thus take care to paste the **URL** displayed in your console in the address bar of your web browser.

2.2 Cells Overview

The first analysis that has to be done for this project is to point out cells that present special shapes. In fact, looking at the values for each genes in a single cell, it's quite simple to plot a **traceplot** showing the pics as point values in the genes. Thus, keeping an eye on the traceplot we can navigate through the different cells, and get an overview of the the expression levels of the **300** genes in the **23 822** cells. Let's recall to the main goal of this analysis is to allow the biologists to analyze the expression of specific genes in cells at a given time, which can reveal intra-tissular heterogeneity's that would remain hidden with other methods. As we want to make it easy for the scientist to display a particular cell, we implemented an input layer that allow to choose a specific cell to display or the navigate through the cells with the small vertical arrows. As it can be tedious to distinguish the precise

value of a particular gene in the displayed cell, we make it interactive to allow the user to access cell values for a each gene just by moving the cursor over the graph.

Figure 1: Cells overview

As the sorting is a commonly used technology to isolate cell for downstream applications, ensuring that the sorting process itself has little effect on the cells, we make this sort implementation available to the user in the check boxes overlay. In fact it has been demonstrated in previous studies that the sorting of such genes in a particular cell doesn't imply heavy effects on the subsequent analysis of gene expression and on downstream cellular applications. Moreover, as we want to make our presentation clearer, the user was allowed to remove flat genes. The mean traceplot has been build by taking the average value of each cells for every genes. Even if it seems flat, we invite the user to remove the variable traceplot to assess for the smaller mean variations through genes.

2.3 Cell Types Overview

The second part of the data we wanted to focus on is the different cell types. The first graph shows the value range of each gene for a corresponding cell type. The goal here is to be able to see which gene are close to each other in a cell types and which aren't. In addition, it gives the opportunity to visualize and compare the outliers and the values concentration for each genes. Once again it introduces the possibility to remove what we call flat cell type, i.e. cell type of genes with only one or almost only one value **or** cell types that only owns gene values near **0**. In our implementation, a gene is considered flat if it's first quantile is equal to the third one.

Figure 2: Range of values of each gene for a corresponding cell type

By default, the graph displays each gene (as *"select all"* is chosen in the gene drop-down) for the cell type **0** (as **0** is chosen in the cell type drop-down). At first sight, the graph is not really readable, that's why we added the option to remove flat genes as mentioned above, but also another important feature, the ability to sort genes in increasing order and based on their median value. This allows a better view of which gene is close to another gene in that particular cell type.

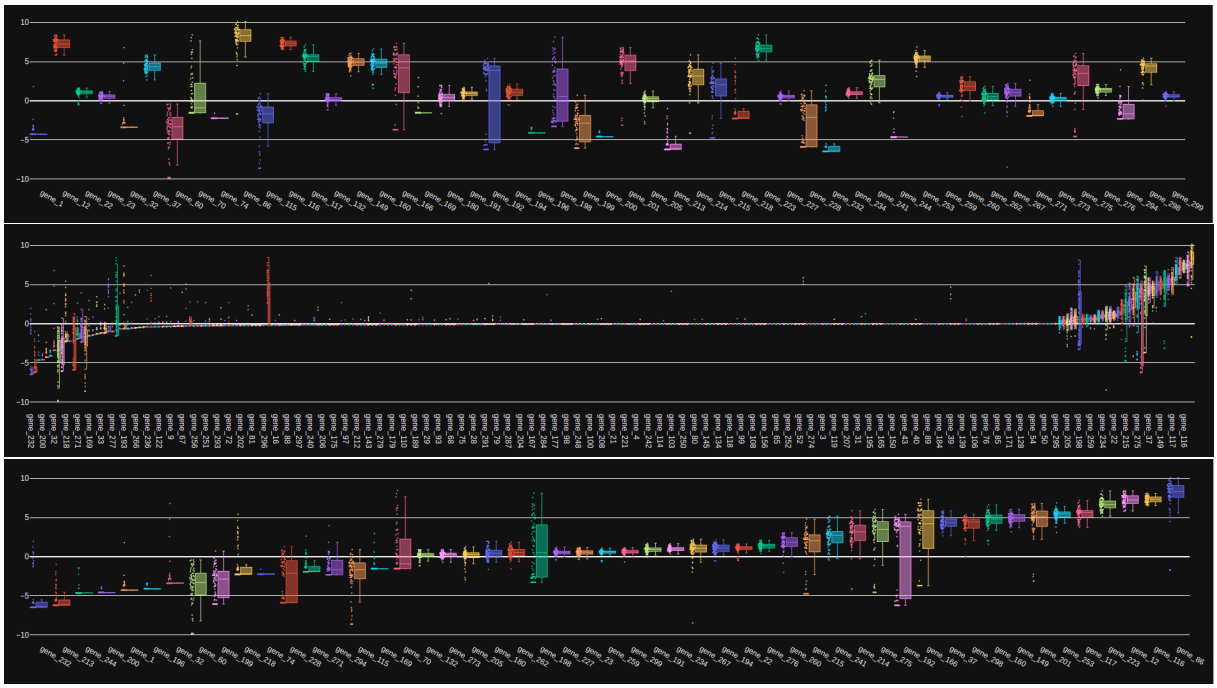


Figure 3: From top to bottom: Remove flat genes, Sort genes, Remove flat genes and Sort genes

This representation is usefull as it offers the user to compare the tails of the sorted range plot into the cell types. Once again, we make it interactive to allow the user to access group cell statistics, such as the **median**, **min** and **max** values and the **quantiles** for each cell type just by moving the cursor over the graph.

The second graph in the Cell Type Overview part goes with the third one on which we will come back later. It shows the **correlation** between two selected cell types, each point being a gene whose x-axis value is the average value of the gene value of each cell corresponding to the first cell type and the y-axis value is the same but for the second cell type. This allows to compare closely each cell type's correlation with another and led us to go further in the cell type analysis by applying a PCA and clustering algorithms on that cell type part of the data which we'll talk about in the next subsection.

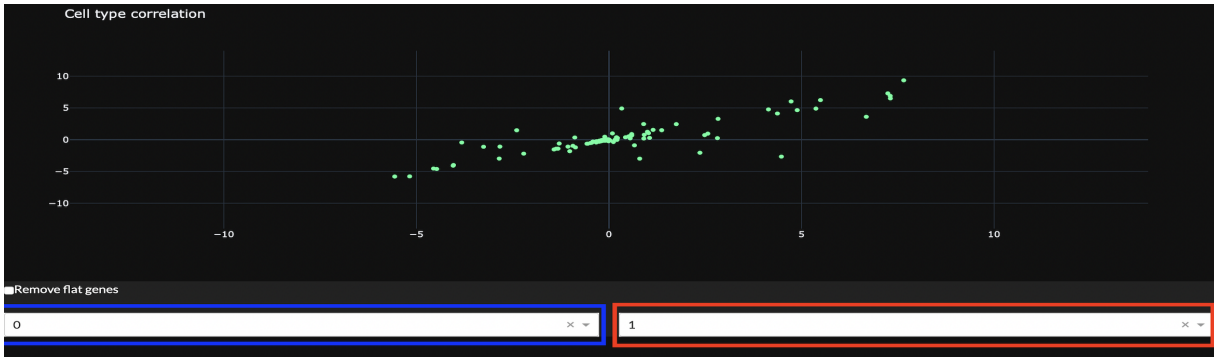


Figure 4: Correlation between cell types 0 and 1

As can be seen in Figure 4, we again propose to remove the flat genes as they can be a source of noise even though this is not annoying on this particular graph.

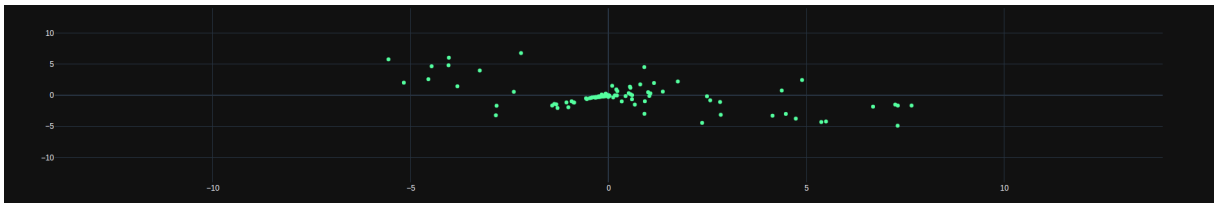


Figure 5: Correlation between cell types 0 and 100

Here, we can clearly see the difference of correlation between Figure 4 and 5. Indeed, cell types 0 and 1 shows a clear positive correlation while on the other side, cell types 0 and 100 are negatively correlated. In general, we observed that cell types numerically close to each others are more correlated than cell types numerically far from each others. A clue of that difference might rely in the difference of cell family as discussed in the introduction of this report.

Finally, the last graph of this Cell Type Overview part is a **correlation heatmap** between each cell types. It gives the user a great overview of which cell types are correlated and already gives some insights of what can be expected for the dimensionality reduction and clustering part of this project.

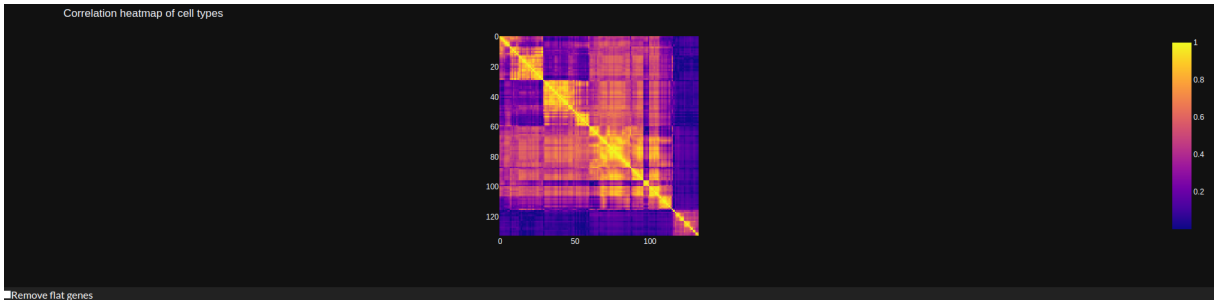
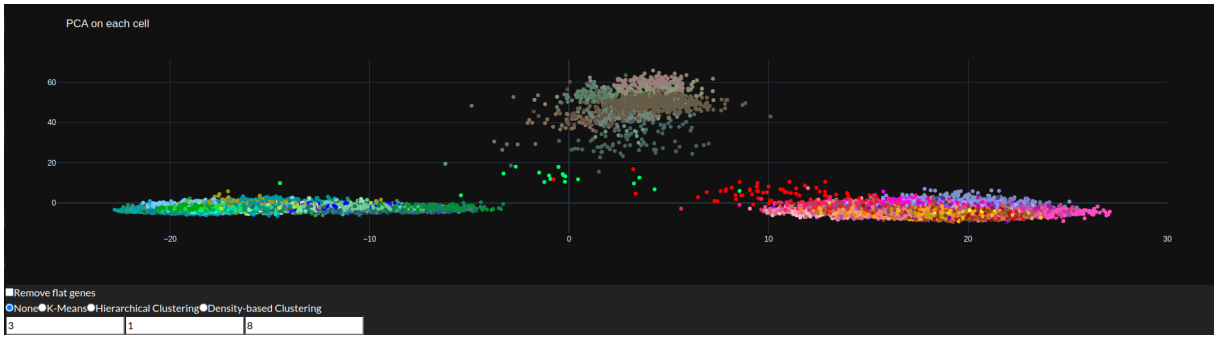


Figure 6: Heatmap of the correlation between each cell types

2.4 Clustering and Dimensionality Reduction

For this last main part of our visualization dashboard, we decided to let the user choose to apply the dimensionality reduction and clustering algorithms whether on **Genes**, **Cell types** or even **Cells** themselves by selecting the according data type at the top of the window. However, depending of the computing power of your computer, you might face performance issues with the **Cells** data type due to the enormous amount of data that has to be processed by those algorithms but feel free to navigate through the different views implemented.



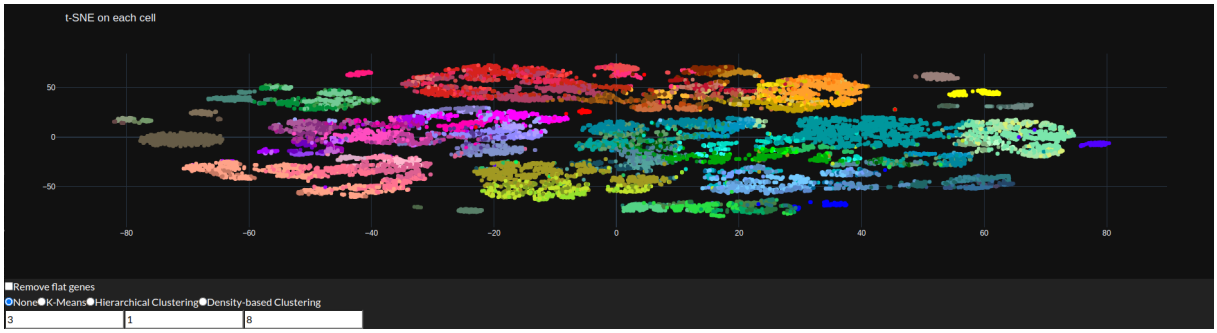


Figure 7: From top to bottom: PCA on each cell and t-SNE on each cell. Colors being the cell type's one

Whatever the data type chosen and as seen above, this page offers two graphs, the first graph is for the **PCA algorithm** and the second one for the **t-SNE algorithm**. Nevertheless, for the conciseness of this report, we will focus on the PCA applied on the **Cells** data type as it gives clear and interesting results. At first glance, we can distinguish three main clusters, then confirmed by our clustering algorithms. Those clusters corresponds to the three general cell families discussed in the Introduction, with each point being a cell. When no clustering algorithm is chosen ("None") each color corresponds to one cell type as seen on both figures above.

It is now time to apply the clustering algorithms to this dimension reduced view of the dataset. We used three different algorithms, each with different properties whose are going to be explained in the third section of this report. The first one is the well-known **K-Means** algorithm, then followed by an **agglomerative hierarchical clustering** which gives similar results and finally, the density-based DBSCAN algorithm which is quite different than the two others.

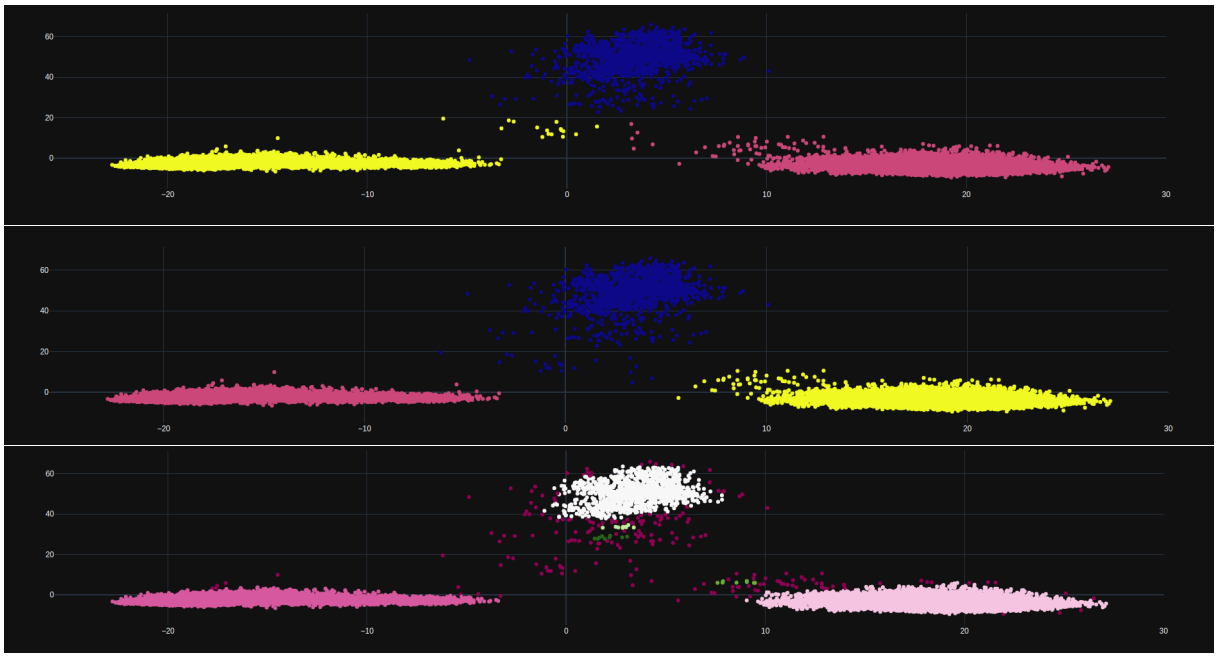


Figure 8: From top to bottom: K-Means and hierarchical clustering with K = 3 and Density based clustering with $\epsilon = 1$ and min samples = 8

Concerning the **K-Means** and **hierarchical** clustering algorithms, we offer the opportunity to **increase the number of clusters** displayed (see following sections to understand how we implement this) to the user. On the other side, for the **DBSCAN** algorithm, you can **choose the epsilon (or r) value** and the **minimum samples** (as they

are explained in the part 3.1.3 of this report) by playing respectively with the second and third boxes beyond the algorithm choice.

3 Algorithms employed

3.1 Clustering algorithms

3.1.1 K-means

The first implemented clustering algorithm is the **K-Means** that aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean so that it minimizes the following within-cluster sum of squares.
Thus we have to minimize the following equation:

$$argmin_S \sum_{i=1}^{k} \sum_{w \in S_i} ||x - \mu_i||^2$$

which is the within-cluster sum of squares and where μ_i is the mean of point in the cluster S_i. This is equivalent to minimizing the pairwise squared deviations of points in the same cluster but this is not the chosen strategy for our implementation.
The **K-means algorithm** works intwo steps:

- Assignment step: Assign each observation to the cluster with the nearest mean (least squared Euclidean distance)
- Update step: Recalculate means (centroids) for observations assigned to each cluster

Moreover, the implementation we have done gives the user the opportunity to choose the number of cluster to display. This is done by simply updating the number of initial centroids of the algorithm.
The **assignment step** is referred to as the **expectation step**, while the **update step** is a **maximization step**, making this algorithm a variant of the generalized **EM** algorithm.

3.1.2 Hierarchical clustering

The hierarchical clustering is a method of cluster analysis that seeks to build a hierarchy of clusters. Strategies for hierarchical clustering generally fall into two categories, **agglomerative** (where each observation starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy) and **Divisive** (where all observations start in one cluster, and splits are performed recursively as one moves down the hierarchy). As this kind of algorithm as a time complexity of $\mathcal{O}(n^3)$ and thus is not as efficient for large sample in comparison with the previous and following algorithm. we have chosen the **agglomerative** one for our implementation that will use the distance matrix for the linkage criterion, recursively merging pair of clusters of sample data and thus using this linkage distance. Once again we can choose the number of clusters to find, updating this value with the layer.
Finally we define the distance between two clusters to be the average distance between data points in the first cluster and data points in the second cluster keeping in mind that at each stage of the process we combine the **two clusters that have the smallest average linkage distance**:

$$d_{12} = max_{ij} d(X_i, Y_j)$$

where **X** and **Y** are the set of observations from cluster **1** and **2** and $\boldsymbol{d(x_i, y_j)}$ is the distance between the members that are farthest apart.

3.1.3 Density based clustering

For the implementation of this algorithm we used the **DBSCAN** one that constructs clusters iteratively, starting a new cluster S with a non-assigned core object x, and assigning all points to S that are directly or transitively connected to x. Therefore, a spatial index structure is used to perform range queries with radius r for each object that is newly added to a current cluster. And thus, it estimates the density for each point x_i as the number of k_i points that lie inside this radius around the observation x and points are considered directly connected if the distance between them is no greater than r. In our implementation we use $\boldsymbol{r = 1.5}$ and a number of samples (or total weight) in a neighborhood for a point to be considered as a core point equal to $\mathbf{8}$. Moreover we use the **"auto"** algorithm to be used by the NearestNeighbors module to compute pointwise distances.
In other words, our algorithm finds core samples of high density and expands clusters from them. It's a efficient implementation for data which contains clusters of similar density.

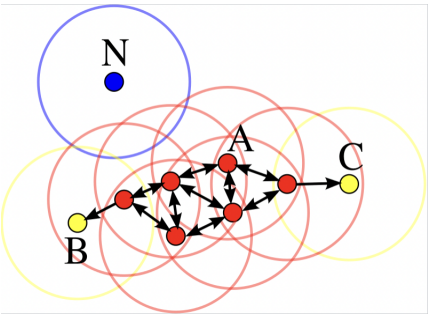


Figure 9: DBSCAN algorithm

In this diagram, the **min sample** has been set to $\mathbf{4}$. Point A and the other red points are core points. Because they are all reachable from one another point, the points B and C belong to the cluster as well but not the point N.

3.2 Dimensionality reduction

3.2.1 Principal Component Analysis

The linear dimensionality reduction algorithm we chose to implement is the well-known Principal Component Analysis (or PCA). This algorithm allows to analyze huge datasets with a high number of dimensions by linearly transforming the data to a corresponding representation of this data with less dimensions than the original one. Thus, here we will focus on the correlation or not of the different cells and cell types, allowing to then apply our clustering algorithms and the PCA's results. We invite you to check the results on the figure 7 displayed above in the report.

3.2.2 t-distributed Stochastic Neighbor Embedding

The other dimensionality reduction algorithm we use is the non linear method of t-distributed stochastic neighbor embedding (or t-SNE) which allows to represent a set of points living in a high dimensional space in smaller two or three dimensional space. Once again, feel free to check the figure 7.

4 Potential improvements

One main improvement we could bring to the project is on the user interface side. Indeed, a user discovering the dashboard without having read this report might need some time to understand each plot and variables he can play with. Finally, the other UX improvement is about interactivity between each plots. For now, although they are related to each other, plots are not interactive between each other without the intervention of the user.

References

Chambers, D., Carew, A., Lukowski, S., & Powell, J. (2018). Transcriptomics and single-cell rna-sequencing. *Respirology*, *24*, 29-36. Retrieved from https://arxiv.org/abs/2006.05626 doi: 10.48550/ARXIV.2006.05626

Colin, W. (2004). *INFORMATION VISUALIZATION, Perception for Design*. Morgan Kaufmann. Retrieved from https://moodle.uclouvain.be/pluginfile.php/183435/mod_resource/content/1/Information%2C%20Visualization%2C%20Perception%20For%20Design%2C%202Nd%20Ed%20-%20Morgan%20Kaufmann.pdf

John, L. (2022). *LDATA2010: Information visualisation.* University Lecture, https://moodle.uclouvain.be/course/view.php?id=3502.

Stegle, O., Teichmann, S., & Marioni, J. (2015). *Computational and analytical challenges in single-cell transcriptomics*. Nature Reviews Genetics. Retrieved from https://www.nature.com/articles/nrg3833

Stephenson, M., Nip, K. M., HafezQorani, S., Gagalova, K., Yang, C., & Warren, I., René L Birol. (2021). Rna-scoop: interactive visualization of transcripts in single-cell transcriptomes. *NAR Genomics and Bioinformatics*, *3*. Retrieved from https://arxiv.org/abs/2006.05626