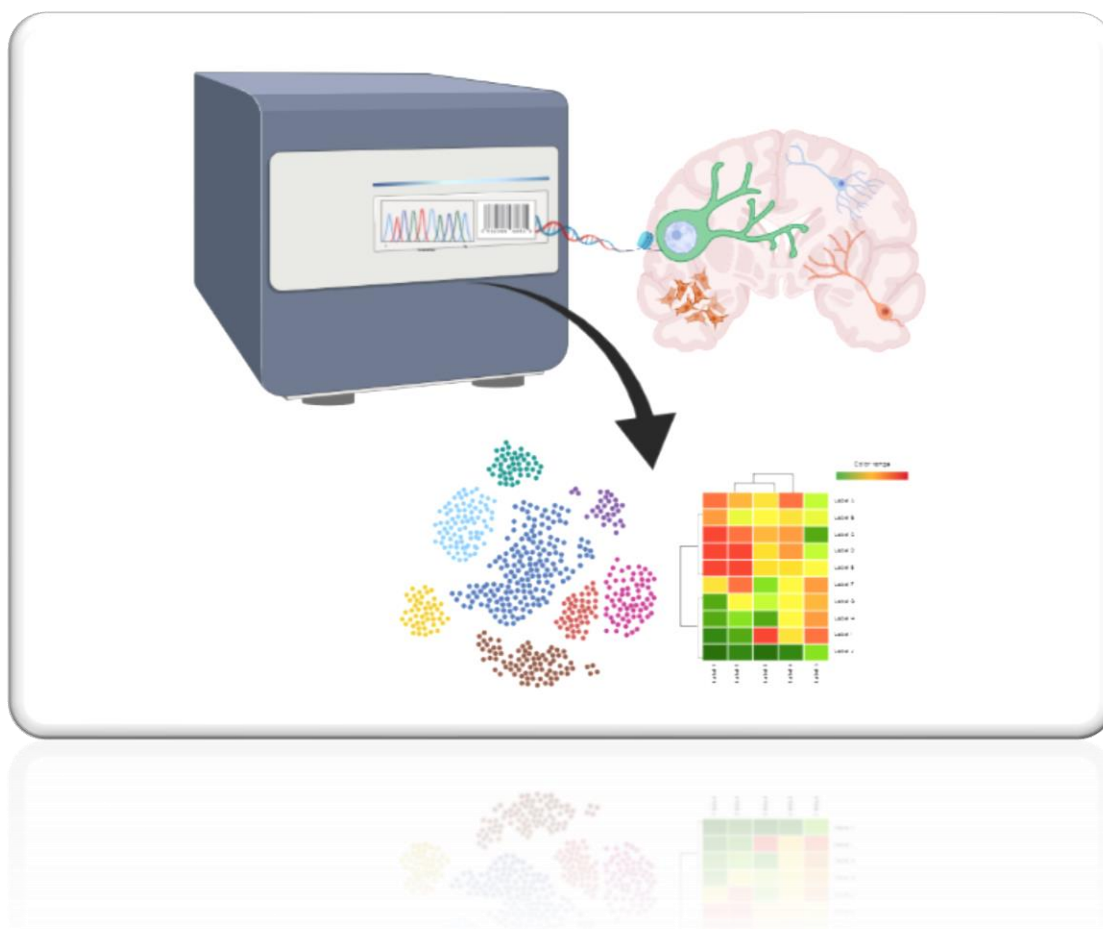


JSEQ[®] manual

Authors:
Jakub Kubiś
&
Maciej Figiel

JSEQ[®] - single cell sequencing analysis tools



Institute of Bioorganic Chemistry,
Polish Academy of Sciences
Department of Molecular Neurobiology

Table of Contents

1.	First use and installation	4
1.1.	Working directory	4
1.2.	Installation	4
1.3.	First use	5
2.	Analysis – raw data (fastq files)	7
2.1.	Genome download	7
2.2.	Create project	8
2.3.	Run analysis	9
3.	Analysis – gene count / normalized expression	11
3.1.	Create project	11
3.2.	Run analysis	12
4.	Manual analysis	14
4.1.	Run manual analysis	14
5.	Report & Analysis	16
5.1.	Quality control of the reads	16
6.	Single-cell analysis & statistics	16
6.1.	Cells content information	16
6.2.	Outliers and thresholds	18
6.3.	Data normalization	20
6.4.	Most variable genes	21
6.5.	Principle components (PCs)	21
6.6.	Selecting the right number of PCs	22
6.7.	Data clustering	23
6.8.	UMAP - Uniform Manifold Approximation and Projection for Dimension Reduction	23
6.9.	Cluster markers	24
6.10.	Cells naming - markers	24
6.11.	Dividing the cells subclasses (clusters) into cells subtypes	27
6.12.	Outliers dropping	29
6.13.	Cells subtypes visualization	30
7.	References	32

Introduction

Single-cell sequencing is a modern technique of sequencing at the single-cell level. Such an approach gives many possibilities to get knowledge about gene expression in different cell populations during development, carcinogenesis, or after treatment with novel gene therapy. However, how it works? So-called single-cell sequencing does not focus on 'single cell' but on whole populations of cells we sequence. So whence do we obtain information about a single cell? Currently, we have some approaches for single-cell sequencing, but the final effect is similar. Results of single-cell sequencing is a significant number of cell transcriptome sequences with unique sequence signs as 'barcode' and 'UMI' - Unique Molecular Identifiers. The barcode sequence is different for each cell during sequencing. Furthermore, 'UMI' is independent of the barcodes sequence, which allows distinguishing each copy of mRNA (count) in the cell (for each barcode) and removing duplicated values. So there are two essential sequences in bioinformatics single-cell analysis, barcode (about 12 bp) specific to the single cell, and UMI (about 8 bp) specific to the single copy of mRNA (count) in every single cell. Due to this, we can talk about 'single cell' sequencing, although we have thousands of cells' transcriptomes during sequencing and analysis. After sequencing, obtained results are conducted by multi-stage quality control, statistical analysis (e.g., clustering algorithm PCA, kNN, sNN, and visualization support algorithm by dimensionality reduction tSNE and UMAP). The JSEQ® fully automatic pipeline provides an easy and fast way to obtain high-resolution results for single cells using various input types.

Contact:



jkubis@ibch.poznan.pl



mfigiel@ibch.poznan.pl

1. First use and installation

The JSEQ® pipeline was prepared for UNIX/LINUX systems, so if you are a WINDOWS user and you have Windows 10, do not worry. Currently, Microsoft® offers a good LINUX subsystem without change the basic WINDOWS system. So if you still have not the LINUX subsystem on your PC, go to Microsoft Store and install Ubuntu and enjoy your first steps in the UNIX environment, and if your UNIX environment is ready, go my GitHub.

1.1. Working directory

In the JSEQ® 2.1.0 release were added option to choose where the pipeline required tools should be installed. If you choose 'local' then the whole pipeline will be installed at localhost, or when you choose 'docker' the requirements will be installed in the docker container (before using this option, install docker on your computer). In both cases, project results and requirement files you can view and modify on localhost, and only analysis and tools installation will be conducted in the container in case of the 'docker' working directory.

1.1.1. Open a correct directory in bash console and clone git repository:

git clone https://github.com/Qubix96/JSEQ_scRNAseq.git

1.1.2. Go to the JSEQ_scRNAseq repository and run the program by writing './JSEQ' in the console

1.1.3. Choose a working directory: localhost write 'local'; docker write 'docker' in the console

```
Welcome in the JSEQ® scRNAseq pipeline which was prepared at Institute of Bioorganic Chemistry, Polish Academy of Sciences in Poznań
All information and references you can check in the file JSEQ manual on my GitHub
Contact: jkubis@ibch.poznan.pl or jakub.kubis1996@gmail.com

Choose JSEQ working directory location:
Local [local]
Docker container [docker]
```

1.2. Installation

1.2.1. Write the command 'install' in the console and push enter

Docker container:

```
Choose function:
-docker container installation [install]
-start container [start]
```

Localhost:

```
Choose JSEQ function:
-local installation [install]
-download RefGenome (GENCODE) [genome]
-creat new project [project]
-start analysis [analysis]

Choose function:
```

*There, you can see all options that give JSEQ pipeline.

1.2.2. In the case of localhost, write 'sudo' to permit installing all the required files

```
Write [sudo] to give installation permission  
sudo
```

1.2.3. Wait for installation finish. You will see the installation progress in the console, but if you want to check that there weren't any errors, go to JSEQ_scRNAseq/setup/install.log.out and explore the results

1.3. First use

If the JSEQ were installed, you could know about the rest options other than 'install'. There were described each of them shortly in the following chapters, where you can find how they work and how to use them.

I want to point out that JSEQ can be used for analysis at different steps. The first option gives possibilities to start the analysis from FastQ files. When you want to run it, firstly, you should choose the option 'download RefGenome'. In this option, a reference genome and gene annotations will download from the <https://www.gencodegenes.org/> website. In this version of JSEQ, you can choose the Human or Mouse genome. If you need other genomes, you can write an email to me (email in contact).

Another option is to run an analysis when we have results after mapping raw reads to the genome. It can be data in the matrix format (or sparse matrix) for raw counts or normalized expression values. However, before running the analysis, you have to create a project. To do this, you have to run the option 'create new project'. There are two options to choose from, and they will depend on which analysis you choose: raw data (fastq files) or after mapping. The same will regard to 'start analysis' option, where you will have to choose analysis options.

After analysis in both cases, you will obtain reports: one report with single-cell results in option without mapping and two reports for raw data analysis (report for quality control of sequences and report with single-cell analysis results).

1.4. Working in docker:

1.4.1. Run docker container: write 'start' in console

```
Choose function:  
-docker container installation [install]  
-start container [start]
```

Since then, all the rest functions will be the same as those written below for the local working directory. What is more, results are saved in the local repositories. Requirements file like markers and information about UMIs and barcodes length you can also modify in the local repository.

```
Docker container  
  
Choose JSEQ function:  
-download RefGenome (GENCODE) [genome]  
-creat new project [project]  
-start analysis [analysis]  
  
Choose function:
```

2. Analysis – raw data (fastq files)

2.1. Genome download

Before start analysis, you have to download reference genome and gene annotations proper to your data sets. The downloaded genome will be deposited in a separate directory from your analysis directory. What is more, you will have to annotate genes to your genome, but this step will run during project creation because gene annotation depends on your sequenced reads length (e.g., for Nextera550 is 75 or 150 bp). This step can last a while, so to prevent lost time each time when you create a new project, the annotated genome is also saved in a different directory than your project directory. The gene annotations step will be omitted when you run an analysis in the future with the same length of the reads.

Genome downloading:

2.1.1. Write 'genome' in the console and push enter

```
Choose JSEQ function:
-first installation [install]
-download RefGenome (GENCODE) [genome]
-creat new project [project]
-start analysis [analysis]

Choose function:
genome
```

2.1.2. Choose the species for which you will run the analysis. Write the specie name in the console. There is also an option for mixed mouse/human analysis prepared for mixed data experiments. When you write 'mix' in the console, will download both genomes

```
Enter the species [human/mice/mix]:
mix_
```

2.2. Create project

Now you have to create a new project for raw reads fastq analysis. How I wrote above during this step will be run gene annotation, which depends on your reads length.

Project creating:

2.2.1. Write 'project' in the console and push enter

```
Choose JSEQ function:
-first installation [install]
-download RefGenome (GENCODE) [genome]
-creat new project [project]
-start analysis [analysis]

Choose function:
project
```

2.2.2. To choose 'fastq data' write '1' in the console

```
Choose function:
project
Create project for fastq data [1] or counts / expression data [2]:
1
```

2.2.3. Now you have to fill in some data as project name, reads length, species, the estimated number of cells

Project name – write your project name; when your name includes more than one word, then use '_' instead of 'space'.

Reads length – write reads length depends on your sequencing technology.

Estimated number of cells – write the number of cells how you expect with your single-cell technology (e.g., dropSEQ).

Marker set – choose the right markers set (default set of markers is for the brain cell types), which will be used in the current analysis.

```
Project name:
test1
Enter operation paramets:
ReadsLenght:
75
Species [human/mice/mix]:
mice
Estimated number of cells
5000
Choose marker set
1) /mnt/c/GIT_Projekty/JSEQ_scrNAseq/requirements_file/markers.xlsx
#?
```

When all parameters are correct, you will see the message and source where you should put your FastQ data. It can be multiple data, which will be combined (e.g., 5x Read1.fq and 5x Read2.fq).


```
Complete
Next step =>

Put fastQ data into => projects/23-02-2021_manual_fq/fast_data
If you done it, push ENTER
```

2.2.4. Put data into the given source directory

Remember that all Read1 files should contain in filename '_R1' and Read2 '_R2'. Without that, you will not go to the next step!

2.3. Run analysis

When your project and data are ready, you can run the first analysis.

Ensure that used markers, adapters, smart primer sequence, barcode, and UMI length are proper for your analysis. Go to JSEQ_scRNAseq/requirements_file/* and check this.

Adapters.fa	11.01.2021 09:15	Plik FA	1 KB
barcodes	09.03.2021 12:15	Plik	1 KB
markers	08.03.2021 08:52	Arkusz programu ...	19 KB
smart_primer	25.02.2021 11:29	Plik	1 KB

Adapters:

Open adapters file and check adapters sequence.

```
1 >CUSTOM_P5_INDEX_SUFFIX
2 GCCTGTCCGCGG
3 >P5_INDEX_SEQUENCE
4 AATGATACGGCGACCACCGAGATCTACACGCTGTCCGCGG
5 >P5_INDEX_SEQUENCE_PREFIX
6 AATGATACGGCGACCACCGAGATCTACAC
7 >ILLUMINA_UNIVERSAL_READ_SEQUENCE
8 AGATCGGAAGAG
9 >CUSTOM_READ_SEQ_REVERSE_COMPLEMENT
10 GTACTCTGCGTTGATACCACTGCTTCCGCGGACAGGC
11 >INDEX_NXX7_PREFIX
12 GTCTCGTGGGCTCGG
```

*if your you need more or other adapters, so change or add them

Barcodes:

Open barcodes file and check length for UMI and barcodes.

```
1 barcode_start=1
2 barcode_end=12
3 umi_start=13
4 umi_end=20
5
6 barcode='(?P<cell_1>.{'$barcode_end-$barcode_start+1'}) (?P<umi_1>.{'$umi_end-$umi_start+1'})'
```

Barcode length

UMI length

*if your single-cell technique has other lengths, so change them

Smart primer:

Go to smart_primer file and check primer sequence

```
1 smart=AAGCAGTGGTATCAACGCAGAGTAC
```

*if your single-cell technique has other smart primers, so change it

Markers:

Go to file markers.xlsx and check markers.

How works cell naming with markers, you can read in part **5.10 Cells naming – markers**

Fastq analysis includes:

- Trimming adapters and quality control of reads
- Reads mapping process
- Creating required files in analysis (whitelists, genome dictionary file)
- UMI and barcodes extraction
- Barcodes repairing
- Creating gene count matrix
- Statistical analysis
- Reports generate

Analysis:

2.3.1. Write 'analysis' in the console and push enter

```
Choose JSEQ function:
-first installation [install]
-download RefGenome (GENCODE) [genome]
-creat new project [project]
-start analysis [analysis]

Choose function:
analysis
```

2.3.2. To choose 'fastq data' write '1' in the console

```
analysis
Analysis of data for fastq analysis project [1] counts / expression analysis project [2]:
```

2.3.3. Choose a project which you want to analyze. Write the project number and push enter

```
analysis
Analysis of data for fastq analysis project [1] counts / expression analysis project [2]:
2
1) /mnt/c/GIT_Projekty/JSEQ_scrNAseq/projects/28-06-2021_test1_exp
2) /mnt/c/GIT_Projekty/JSEQ_scrNAseq/projects/28-06-2021_test2_exp
3) /mnt/c/GIT_Projekty/JSEQ_scrNAseq/projects/28-06-2021_test3_exp
#?
```

2.3.4. It can last a while, some information you can see in the console, but if you want to check that all works correctly, go to projects directory 'project/*project_name*/results/process.log.out'. You can check and refresh this log file in real-time during analysis. In the same directory, you can see reads quality control report and statistical single-cell analysis report

3. Analysis – gene count / normalized expression

During this analysis, you need not download any genome and run gene annotation because your data should be in the form of the gene count / normalized expression matrix or sparse matrix.

3.1. Create project

Project creating:

3.1.1. Write 'project' in the console and push enter

```
Choose JSEQ function:
-first installation [install]
-download RefGenome (GENCODE) [genome]
-creat new project [project]
-start analysis [analysis]

Choose function:
project
```

3.1.2. To choose 'counts / expression data' write '2' in the console

```
analysis
Analysis of data for fastq analysis project [1] counts / expression analysis project [2]:
```

3.1.3. Now you have to fill in some data as project name, species, the estimated number of cells

Project name – write your project name when your name includes more than one word, then use '_' instead of 'space'.

Estimated number of cells – write the number of cells how you expect in your data.

Marker set – choose the right markers set (default set of markers is for the brain cell types), which will be used in the current analysis.

```
Project name:
test1
Enter operation params:
Species [human/mice]:
human
Estimated number of cells
1000
Choose marker set
1) /mnt/c/GIT_Projekty/JSEQ_scRNAseq/requirements_file/markers.xlsx
#?
```

3.1.4. Choose data format: 'count matrix' write '1'; 'normalized expression matrix' write '2'; separate file genes.tsv (list of genes name), names.tsv (list of barcodes or cell names), and sparse matrix (matrix.mtx) write '3' in the console





Important: When you use the sparse matrix as input, file names must be for genes - genes.tsv, cell names or barcodes - barcodes.tsv and sparse matrix - matrix.mtx. If not, you will not go to the next step.

If you use count/normalized expression matrix, you can give any names, but the matrix must be in .tsv, .csv, or .txt format, and then data will be converted to a proper format to build the Seurat object.

3.2. Run analysis

When your project and data are ready, you can run the first analysis.

Make sure that used markers are proper for your analysis. Go to JSEQ_scRNAseq/requirements_file/* and check this.

 Adapters.fa	11.01.2021 09:15	Plik FA	1 KB
 barcodes	09.03.2021 12:15	Plik	1 KB
 markers	08.03.2021 08:52	Arkusz programu ...	19 KB
 smart_primer	25.02.2021 11:29	Plik	1 KB

Markers:

Go to file markers.xlsx and check markers. How works cell naming with markers, you can read in part

5.10 Cells naming – markers

Counts /expression analysis include:

- Statistical analysis
- Cells naming
- Outliers dropping
- Report generate

Analysis:

3.2.1. Write 'analysis' in the console and push enter

```
Choose JSEQ function:
-first installation [install]
-download RefGenome (GENCODE) [genome]
-creat new project [project]
-start analysis [analysis]

Choose function:
analysis
```

3.2.2. To choose 'counts / expression data' write '2' in the console

```
analysis
Analysis of data for fastq analysis project [1] counts / expression analysis project [2]:
```

3.2.3. To choose a project which you want to analyze. Write the project number

```
analysis
Analysis of data for fastq analysis project [1] counts / expression analysis project [2]:
2
1) /mnt/c/GIT_Projekty/JSEQ_scrNAseq/projects/28-06-2021_test1_exp
2) /mnt/c/GIT_Projekty/JSEQ_scrNAseq/projects/28-06-2021_test2_exp
3) /mnt/c/GIT_Projekty/JSEQ_scrNAseq/projects/28-06-2021_test3_exp
#?
```

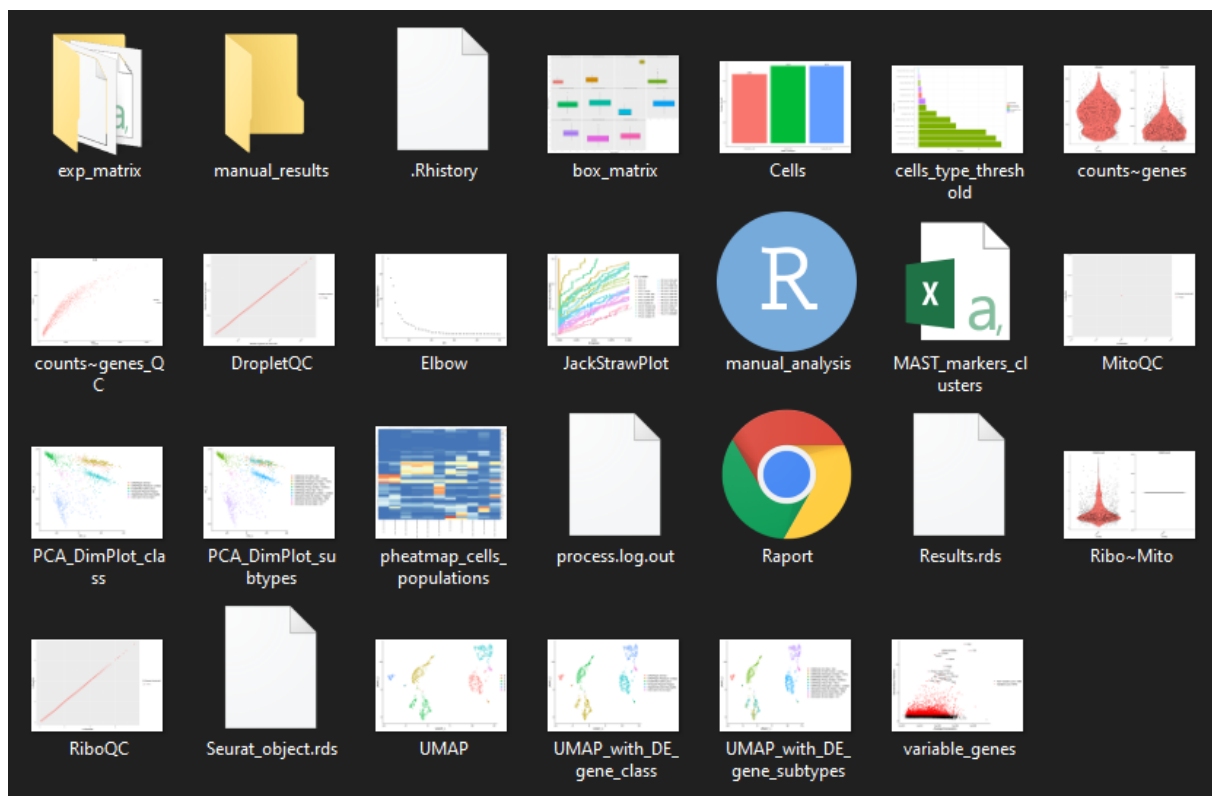
It can last a while, some information you can see in the console, but if you want to check that all works correctly, go to projects directory 'project/*project_name*/results/process.log.out'. You can check and refresh this log file in real-time during analysis. In the same directory, you can see reads quality control report and a statistical single-cell analysis report.

4. Manual analysis

In some cases, automatically obtained results could be inefficient (e.g., wrong cell names, wrong number of principal components (PC), etc.). In such a case, you can run a manual analysis. If the pipeline finished work, go to the project directory and run manual analysis in RStudio (recommended) or other R language IDE.

4.1. Run manual analysis

4.1.1. Go to 'project/*project_name*/results/'



4.1.2. Run 'manual_analysis.R'

4.1.3. Read information inside of the script and fill required data

```
#README
#If you are here it means you want to improve analysis parameters or adjust obtained results
#Below is presented analytic part of the pipeline
#If you want to start the analysis from the beginning you choose UMI <- readRDS('Seurat_object.rds')
#If you want to adjust obtained results (e.g. cell names, plots) you choose UMI <- readRDS('Results.rds')
```

4.1.4. Start manual analysis

You can choose different analysis steps from the beginning or only adjust obtained results such as cell names – all information in script '#' tips.

```
#####

    UMI <- readRDS('Seurat_object.rds')

    #Data ^ - for new manual analysis
    #If you choose this option start from PART A of pipeline (code line 52-986)

    UMI <- readRDS('Results.rds')

    #Data ^ - for adjusting

#In some causes in your results you can see cell subtypes with wrong names obtained based on top marker
#If you want to change cell names run fallow code

    #Cells renameing

Idents(UMI) <- gsub(pattern = 'old_name', replacement = 'New_name', x = Idents((UMI)))
    #Write old name ^           Write new name ^
```

5. Report & Analysis

5.1. Quality control of the reads

For quality control of reads, adapters trimming and repairing were used 'fastp' program, which almost created report. Quality control report you can find in 'project/*project_name*/results/QC_REPORT' Report contains:

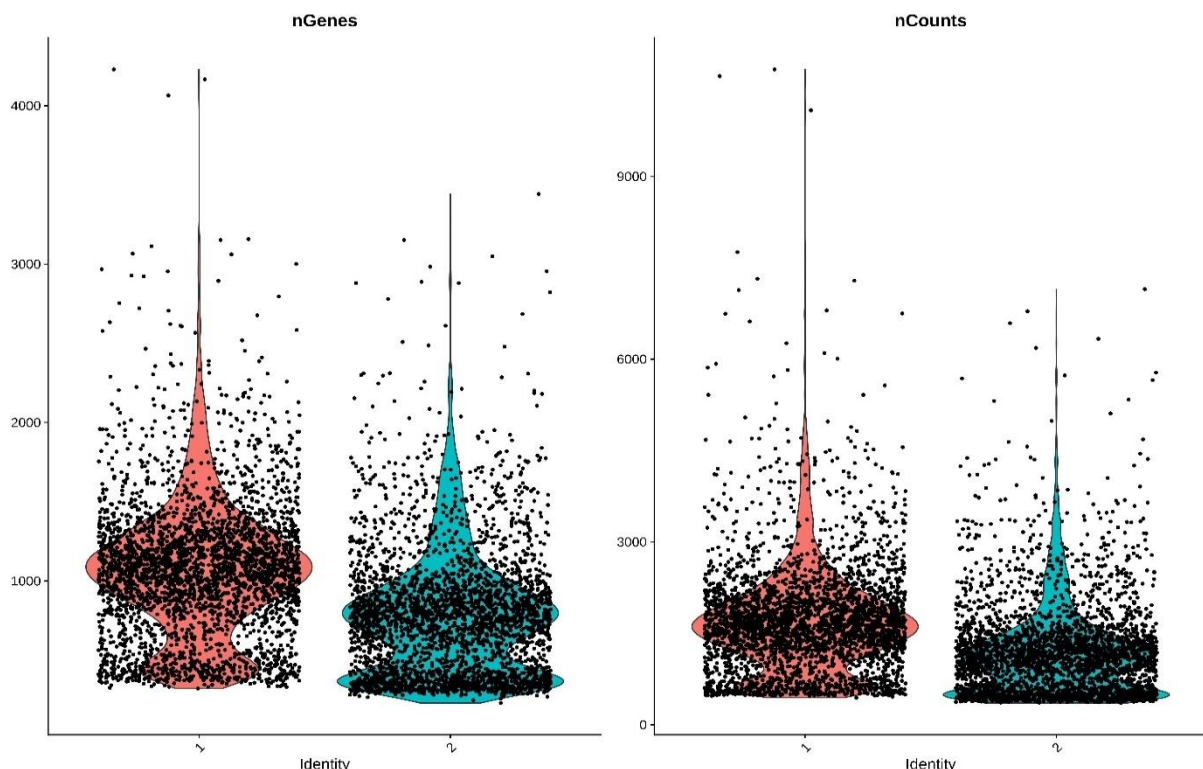
- Adapters content
- Adapters sequence
- Bad ligation adapters
- Nucleotide content graphs

6. Single-cell analysis & statistics

At this stage, many analyses are based on the Seurat library - StjaliLab library with elements that I have programmed in R language, such as cell naming algorithms based on markers, selection of the appropriate number of principal components (PCs), visualization of the content of readings, dropping outliers and creating cell subtypes based on markers selected for individual PCA clusters. More information will be provided later.

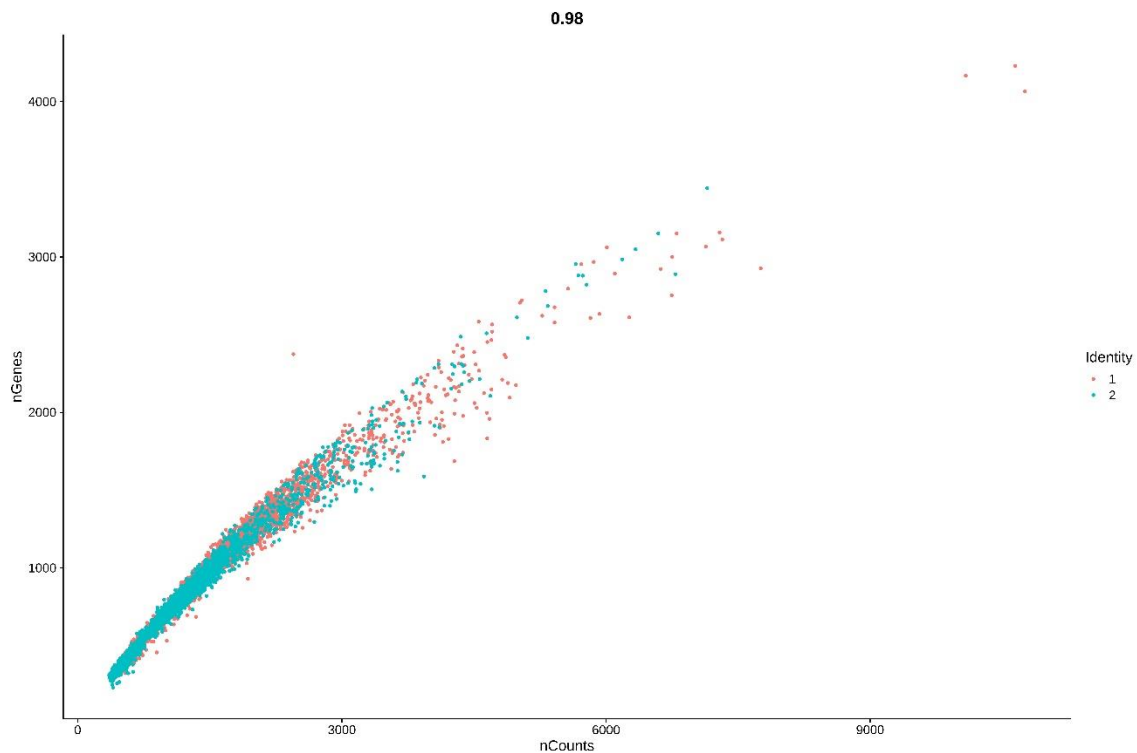
6.1. Cells content information

Counts ~ genes - density plot



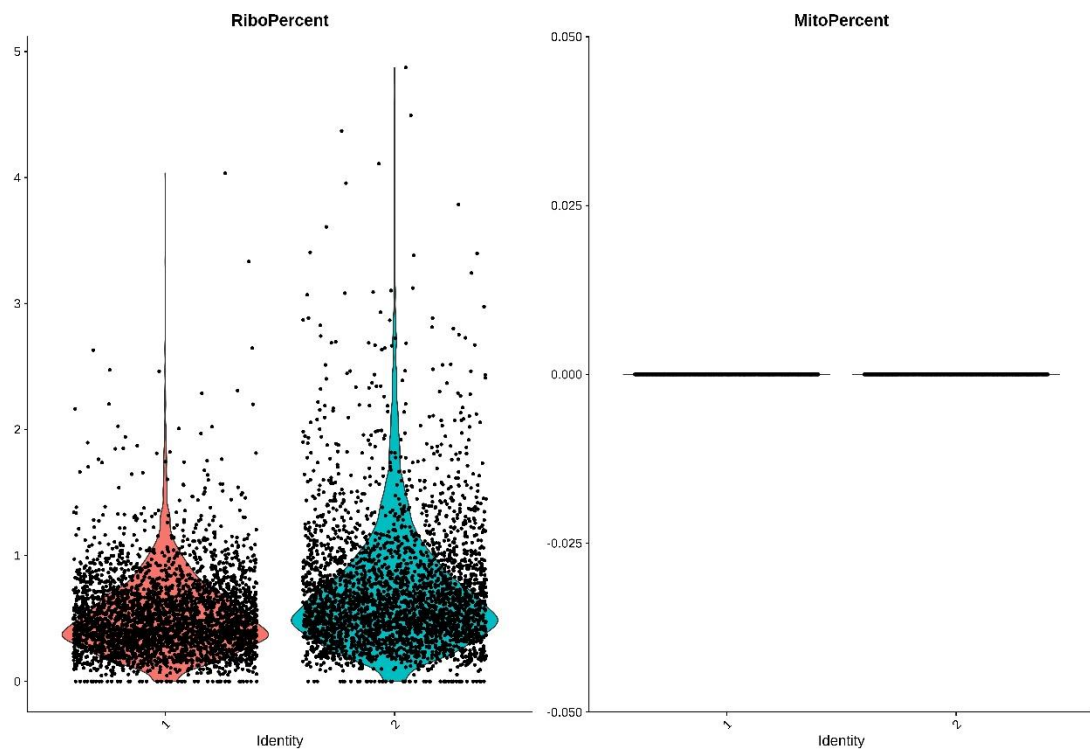
These plots show the density for the number of counts and identified genes for each cell in the analysis.

Counts ~ genes - correlation plot



This plot shows the correlation between the number of counts and identified genes for each cell in the analysis.

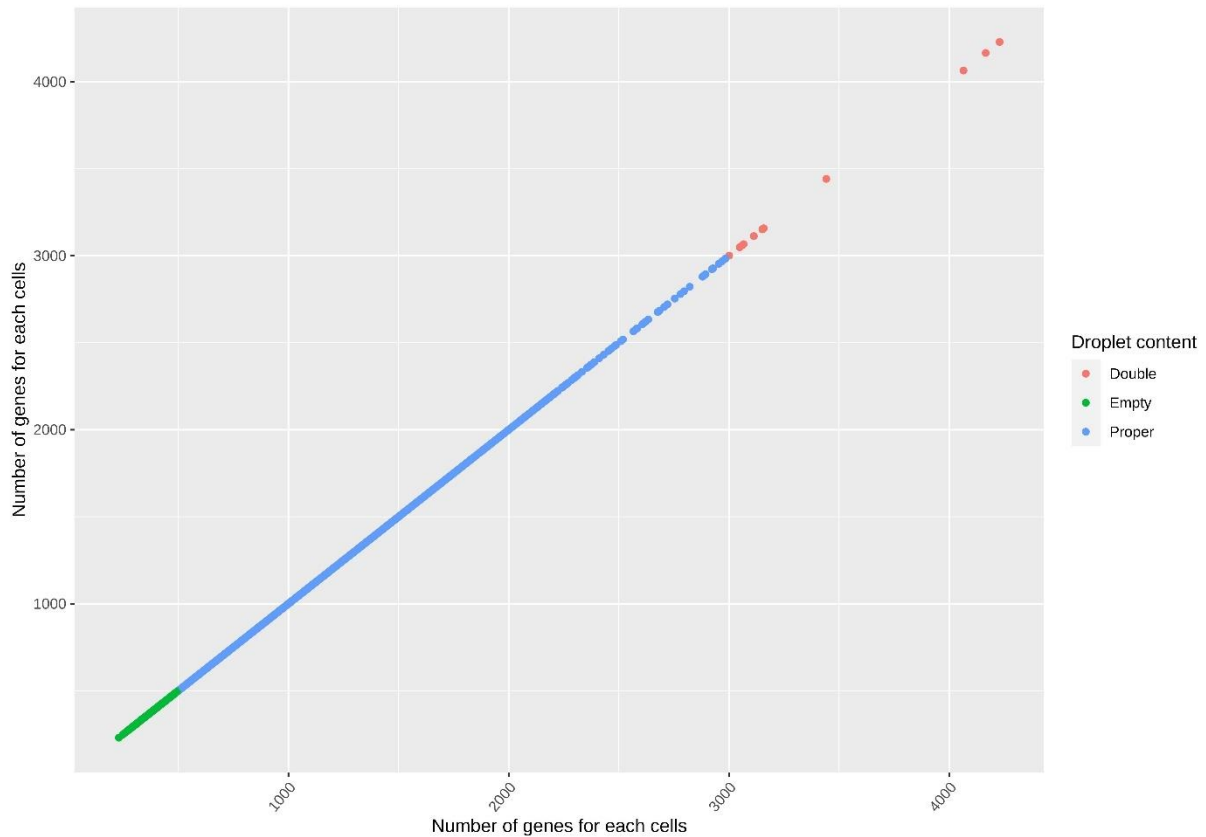
Percent of mitochondrial and ribosomal genes – density plot



These plots show the density of mitochondrial and ribosomal genes on a percent scale [%] for each cell

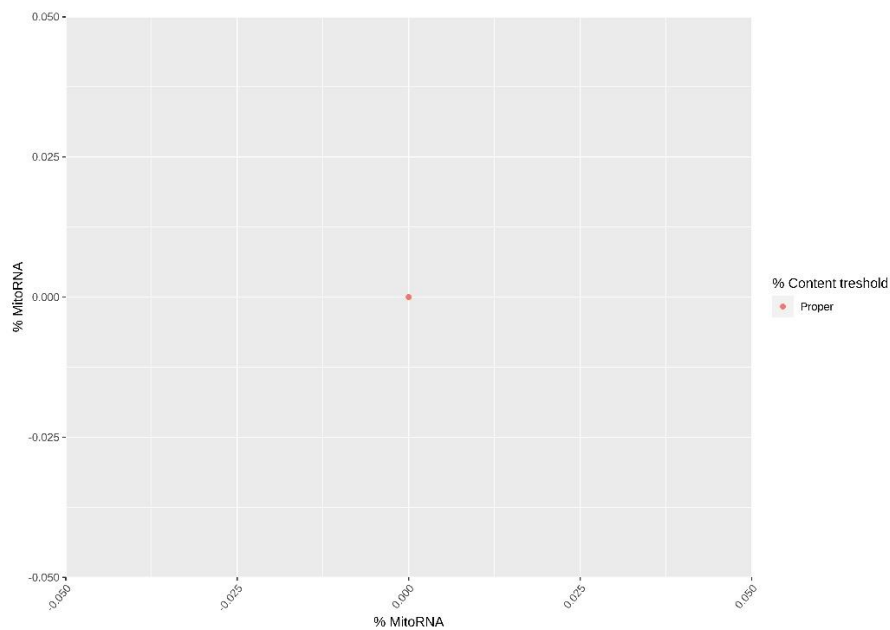
6.2. Outliers and thresholds

Genes number plot



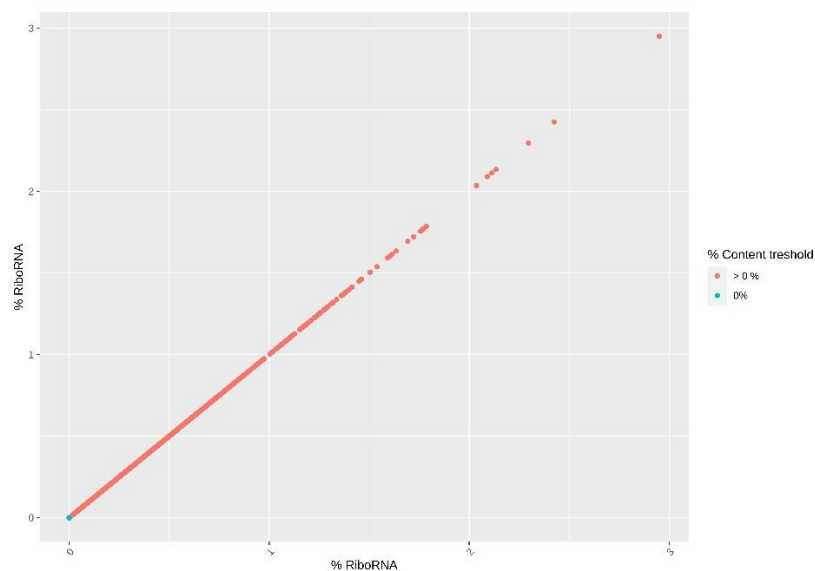
This plot shows compare the number of genes to the number of genes for each cell. Thanks to this, we can see outliers' values. The threshold is set for bottom values on a minimum of 500 genes per cell, and for maximal values is calculated based on two times mean value plus one and half times the interquartile range ($2 * \text{MEAN} + 1,5 * \text{IQR}$) as the maximum value. Cells below bottom values are recognized as empty cells, and upper top values are doublets, triplets, etc.

Mitochondrial genes content - plot



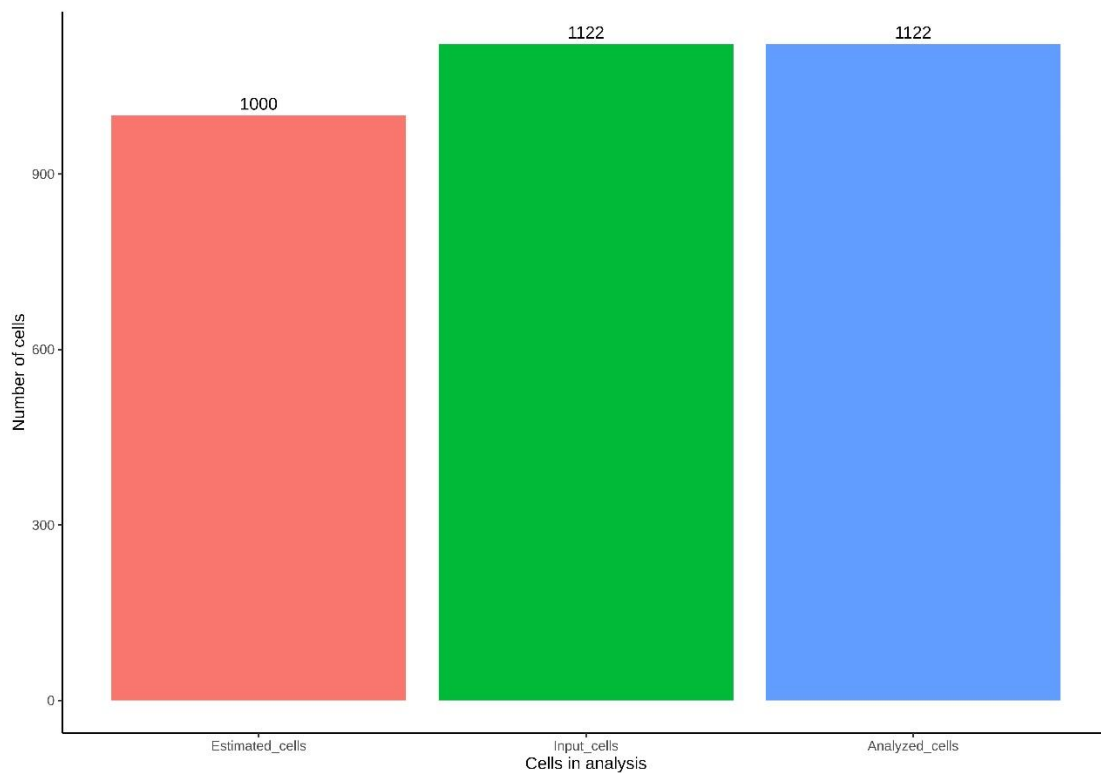
The mitochondrial gene content threshold was set at 5% of all genes displayed in a single cell. All cells that display mitochondrial genes greater than 5% are removed. In this chart, we can see that all cells have 0% mitochondrial genes.

Ribosomal genes content - plot



For ribosomal genes, there are not set thresholds. This plot shows only ribosomal genes content equal to 0% and above 0% in each cell. This knowledge can be used in some experiments to see outliers' values.

Amount of the cells in different stages of analysis – plot



This plot shows us the estimated amount of cells and cells amount at the beginning of analysis and after quality control with set thresholds.

6.3. Data normalization

The Seurat Normalize data function with "Log Normalize" normalization, and the scale factor "1e6" (CPM) was used to normalize the data.

Formula:

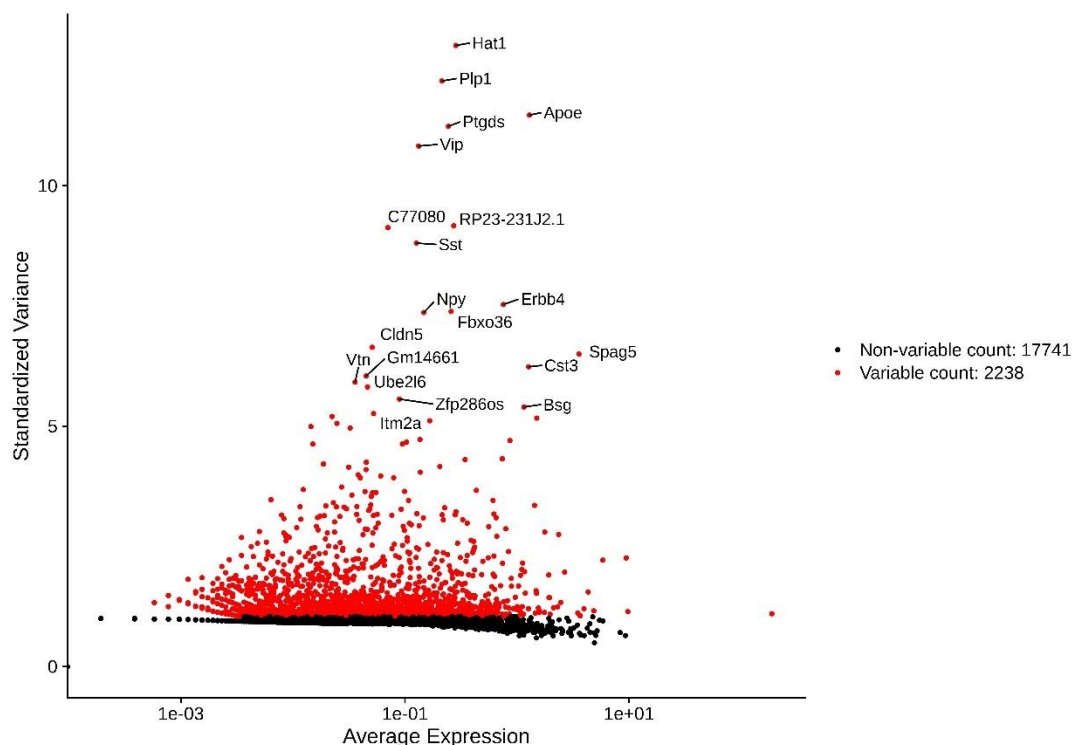
$$CPM = \frac{\frac{\text{counts of gene}}{\text{sum of counts}}}{1000000}$$

$$NormalizedData = \log (CPM + 1)$$

6.4. Most variable genes

To calculate the most variable genes, the 'vst' (Variance Stabilizing Transformation) selection method was used with the 'equal_frequency' method based on the Seurat function FindVariableFeatures. The number of genes used by function was estimated as 75% of the top expressed cell genes.

Most variable genes - plot



This plot shows the highly variable genes among all genes from all analyzed cells.

6.5. Principle components (PCs)

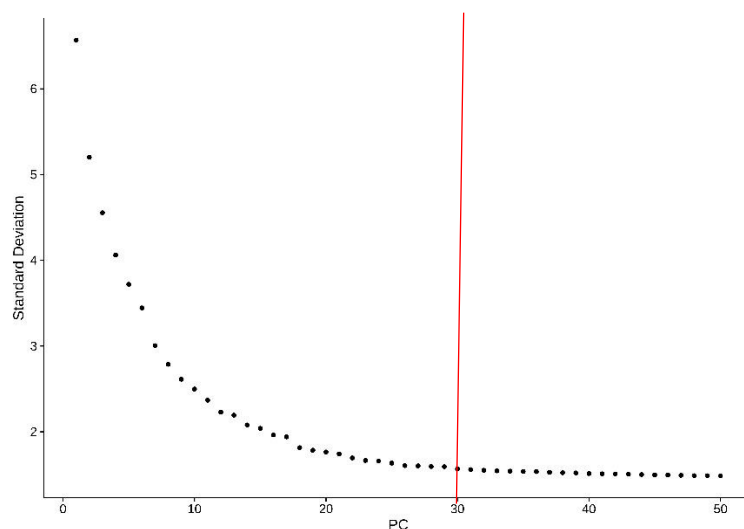
Principle components are a method that is responsible for reducing the dimensionality of 'p' numerical variables for each 'n' element, increasing interpretability without losing any information (Jolliffe & Cadima, 2016). Thanks to this method, we can deal with gene expression matrix for a great number of cells. It is very difficult to compare all genes (p) (for example, for humans and mice, it is about 30 000 genes) in all cells (n), so when we use the PCs method, we can obtain only important statistical information in the form of PCs which explain a maximal amount of variance in data set.

Principal components were calculated on scaled data (ScaleData function) with most variable features (genes) using the Seurat function 'RunPCA'.

6.6. Selecting the right number of PCs

To select the right number of PCs in the analysis, the selection was based on the Elbow plot, JackStraw plot, and a special algorithm that checked changes in standard deviation between following PCs. The Elbow plot was created using Seurat functions' ElbowPlot' with the maximal number of dimension 50.

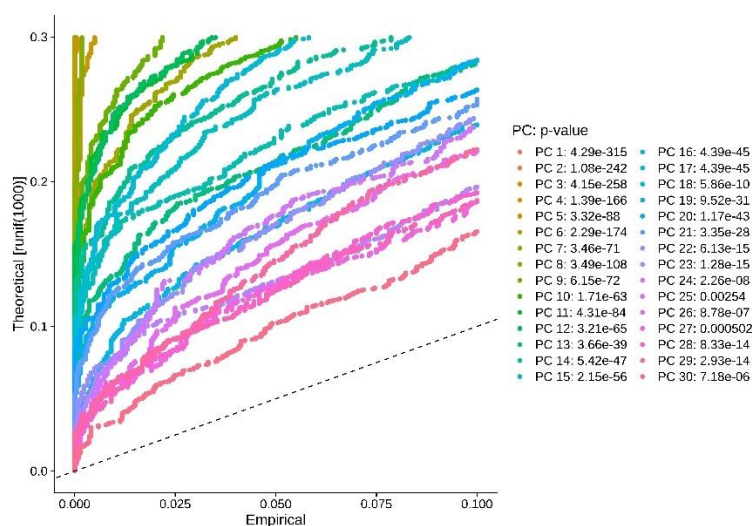
ElbowPlot



In this case, the algorithm chose 30 as the right number of PCs for this data set.

In order to check the correctness of the selected number of PCs, the JackStraw function was used to indicate the statistical significance of individual PCs.

JackStraw plot



Here we can see a p-value for each PCs and if it is statistically significant.

6.7. Data clustering

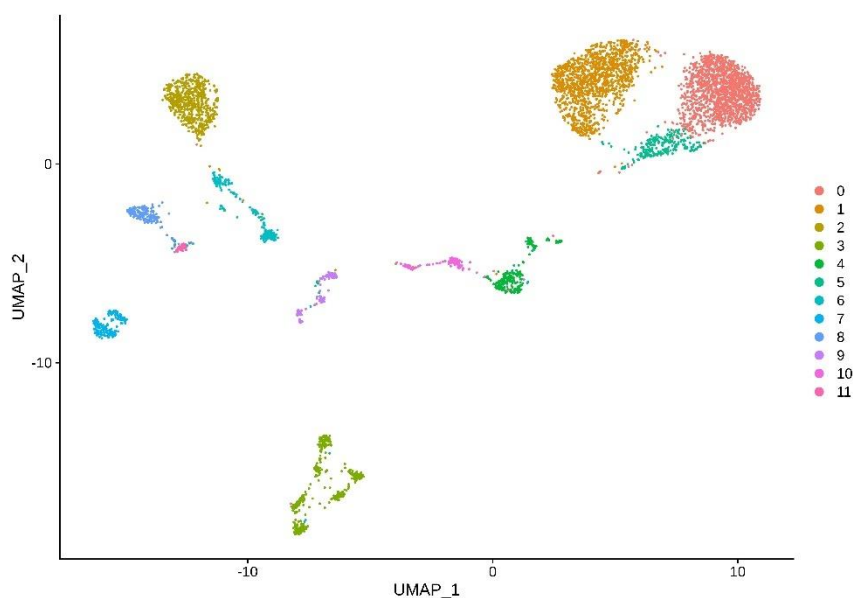
Data clustering based on previously selected PCs used two Seurat functions: 'FindNeighbors' and 'FindClusters'. FindNeighbors function is based on kNN (k-nearest neighbor) algorithm. This function return kNN information for whole datasets based on PCs. Default parameter k in this pipeline for the KNN algorithm is set on 49. FindCluster function based on kNN results identify cells clusters by a shared nearest neighbor (sNN) algorithm. FindCluster function is set on resolution 0.5, the number of starts 10, and the number of iterations 1000.

Both algorithms are common for single-cell analysis and provide clusters connected with different cell populations (Zhu et al., 2020). In the following steps based on these clusters, we obtain marker genes for each cluster, name cells populations using known gene markers, and divide cell populations into cell subtypes.

6.8. UMAP - Uniform Manifold Approximation and Projection for Dimension Reduction

In contrast to traditional linear dimensionality reduction methods like PCA is non-linear UMAP. The UMAP method is another method based on dimensionality reduction and is similar to tSNE method also belongs to the non-linear visualization method. The main role of UMAP algorithms was single-cell data visualization (Narayan et al., 2020). What is more, UMAP seems to be more convenient than tSNE method, which can be problematic with large data sets. UMAP optimizes the embedding coordinates of individual data points using iterative algorithms and constructs a high dimensional graph representation of the data, then optimizes a low-dimensional graph to be as structurally similar as possible.

UMAP plot



This step obtained the UMAP plot with signed clusters (numeric factors) obtained via kNN / sNN algorithms from previous steps.

6.9. Cluster markers

To obtain specific markers for clusters were used Seurat function 'FindAllMarkers'. The selecting method used algorithm 'MAST' with only positive markers and with the minimum frequency of markers in cluster 0.25. MAST algorithm for single-cell gene expression is based on two-part generalized linear models (GLM). First part models the discrete expression rate of each gene across cells, and the second part models the conditional continuous expression level (Finak et al., 2015).

MAST markers

##	p_val	avg_logFC	pct.1	pct.2	p_val_adj	cluster	gene
## 1	0.000000e+00	1.7252321	0.678	0.240	0.000000e+00	0	Gm12027
## 2	0.000000e+00	1.5150667	0.857	0.422	0.000000e+00	0	Akt3
## 3	0.000000e+00	1.4701830	0.992	0.594	0.000000e+00	0	Pde1c
## 4	2.487357e-160	1.6234647	0.414	0.126	4.969490e-156	0	Gm8257
## 5	8.480706e-134	2.1502626	0.267	0.049	1.694360e-129	0	Gm31925
## 6	3.431790e-130	0.9527619	0.657	0.280	6.856374e-126	1	C1ql3
## 7	2.919557e-120	1.0356293	0.588	0.230	5.832983e-116	1	Dgkh
## 8	4.785292e-101	1.1191092	0.489	0.189	9.560535e-97	1	Lman2l
## 9	1.614001e-97	0.9957944	0.507	0.196	3.224613e-93	1	Tanc1
## 10	8.523628e-69	1.5826682	0.271	0.084	1.702936e-64	1	Kri1
## 11	0.000000e+00	3.0471364	0.937	0.070	0.000000e+00	2	Gm28928
## 12	3.418833e-240	2.7580509	0.607	0.052	6.830486e-236	2	Pdzrn3
## 13	4.567684e-168	2.4996415	0.453	0.035	9.125776e-164	2	Vwc2l
## 14	3.283246e-141	3.0890368	0.374	0.033	6.559597e-137	2	2810459M11Rik
## 15	1.263893e-93	2.5857361	0.261	0.018	2.525133e-89	2	Cux2
## 16	0.000000e+00	3.9607792	0.900	0.103	0.000000e+00	3	ErbB4
## 17	3.074380e-191	4.1234410	0.488	0.011	6.142304e-187	3	Gad2
## 18	8.603883e-174	4.6134760	0.420	0.006	1.718970e-169	3	Dlx6os1
## 19	2.778522e-108	3.9512850	0.292	0.006	5.551210e-104	3	Dlx1as
## 20	8.807278e-108	4.0559266	0.332	0.019	1.759606e-103	3	Npy

Obtained markers are helpful to determine to which a cell population belongs cluster.

Moreover, these markers will use for dividing subclass clusters into cell subtypes based on the Cell Subtypes Selection by Genes (CSSG) algorithm.

6.10. Cells naming - markers

For cell naming, a particular Cell Clusters Naming (CCN) algorithm was written, which checked the most expressed marker for each cluster. There are two approaches for cluster naming. The first approach is based on known, defined by user markers.

User markers are in the excel file in JSEQ_scRNAseq/requirements_file/markers.xlsx. We have two types of markers: the first type is in the first sheet (cell class), and the second is in the second sheet (cell subclass).

Cell classes markers

Astrocytes	Oligodendrocytes	Microglial	Endothelial	OPC	Ependymal	Macrophage	Pericytes	Purkinji	Tanocytes	Fibroblast	Granule	Intermediate progenitors	Neuroepithelial
+AQP4	+OPALIN	+TYROBP	+NOSTRIN	+PDGFRA	+FOXJ1	+MRC1	+VTN	+PPP1R17	+RAX	+DCN	+GABRA6	+PAX6	+PAX2
+GFAP	+MOG	+C1QC	+CLDN5	+CSPG4		+CX3CR1							
+SLC1A3	+PTGDS												
	+OLIG2												
	+OLIG1												

You can change your markers depending on yours data and experiments, but remember if you write your own markers for cell classes, you have to add '+' before the marker gene name. Gene markers without '+' will not be readable. It is an excellent manner to save markers without using them in analysis.

Cell subclasses markers

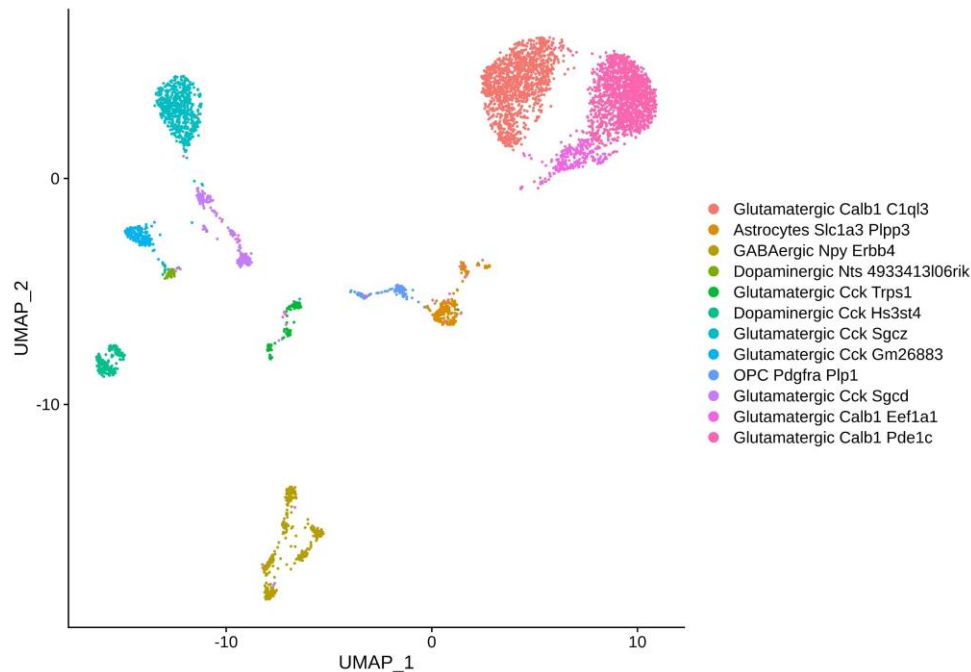
VIP	SST	PVALB	CCK	GAL	CARTPT	NMU	TAC1	NTS	AVP	OXT	GNRH1	PENK	HCRT	TRH	POMC	AGRP	NPY	NMU
-----	-----	-------	-----	-----	--------	-----	------	-----	-----	-----	-------	------	------	-----	------	------	-----	-----

In this case, you need not use additional '+' for markers.

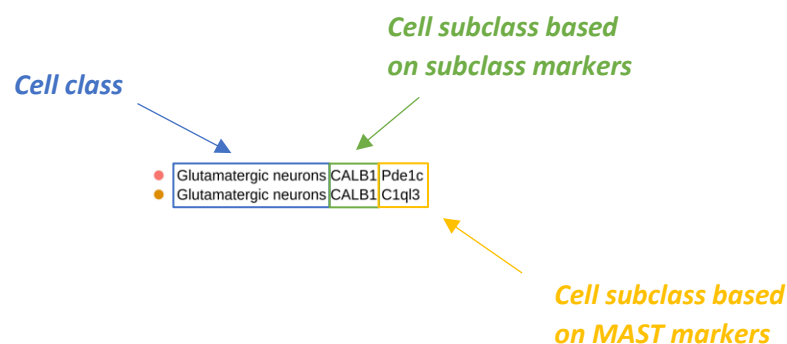
The presented set of markers include markers that determined non-neuronal and neuronal cells class and additionally markers for neurotransmitters, neuropeptides, their receptors, and calcium-binding proteins as genes involved in interneuronal signaling. Both sets of markers are based on a significant number of publications (Anderson et al., 2020; Artegiani et al., 2017; Carter et al., 2018; Chen et al., 2017; Ehman et al., 2017; Eze et al., 2020; Fan et al., 2020; Gokce et al., 2016; Guo & Li, 2019, 2019; Habib et al., 2017; Harris et al., 2017; Kalish et al., 2018; Keo et al., 2017; Kishimoto et al., 2018; Koirala & Corfas, 2010; Kozareva et al., 2020; Langlet, 2019; Martínez-Cerdeño & Noctor, 2018; McKenzie et al., 2018; Mickelsen et al., 2020; Muñoz-Manchado et al., 2018; Nelson et al., 2020; Peng et al., 2019; Smith et al., 2019; Takeuchi et al., 2020, 2020; Zeisel et al., 2018).

The first naming manner is based on three types of markers: class markers, subclass markers, and cluster-specific markers selected by MAST. In the first step, the algorithm checks the cell class based on the most expressed class marker ($\log(\text{CPM}+1)$) and gives the name to the class. In the following steps, the algorithm gives names for cell subclass and subtypes in the same way.

Cells naming scheme – first manner

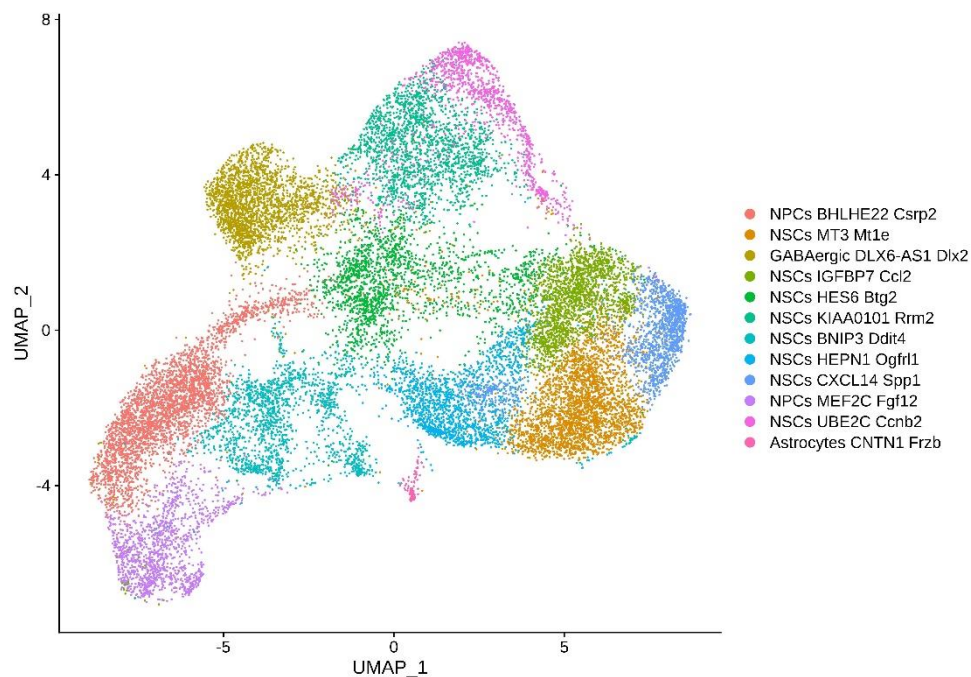


In this plot, are presented cell classes and subclasses after naming. Below is showed scheme of the cell name.

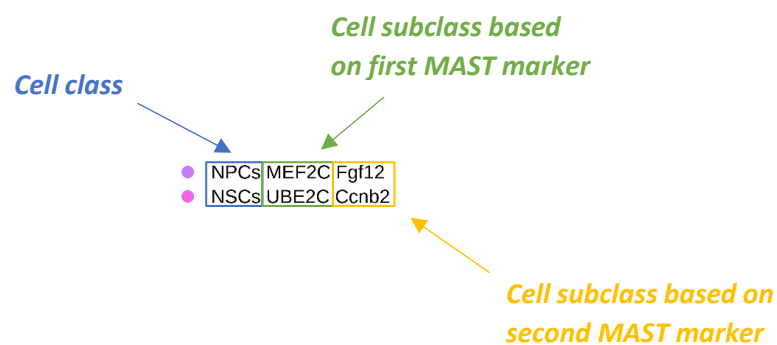


The second naming manner is based on two types of markers: class markers and selected by MAST markers. In the first step, the algorithm checks the cell class based on the most expressed class marker ($\log(\text{CPM}+1)$) and giving the class name. In the following steps, the algorithm checks the cell subclass based on the first and second most expressed MAST typed markers and giving the subclasses names.

Cells naming scheme – second manner



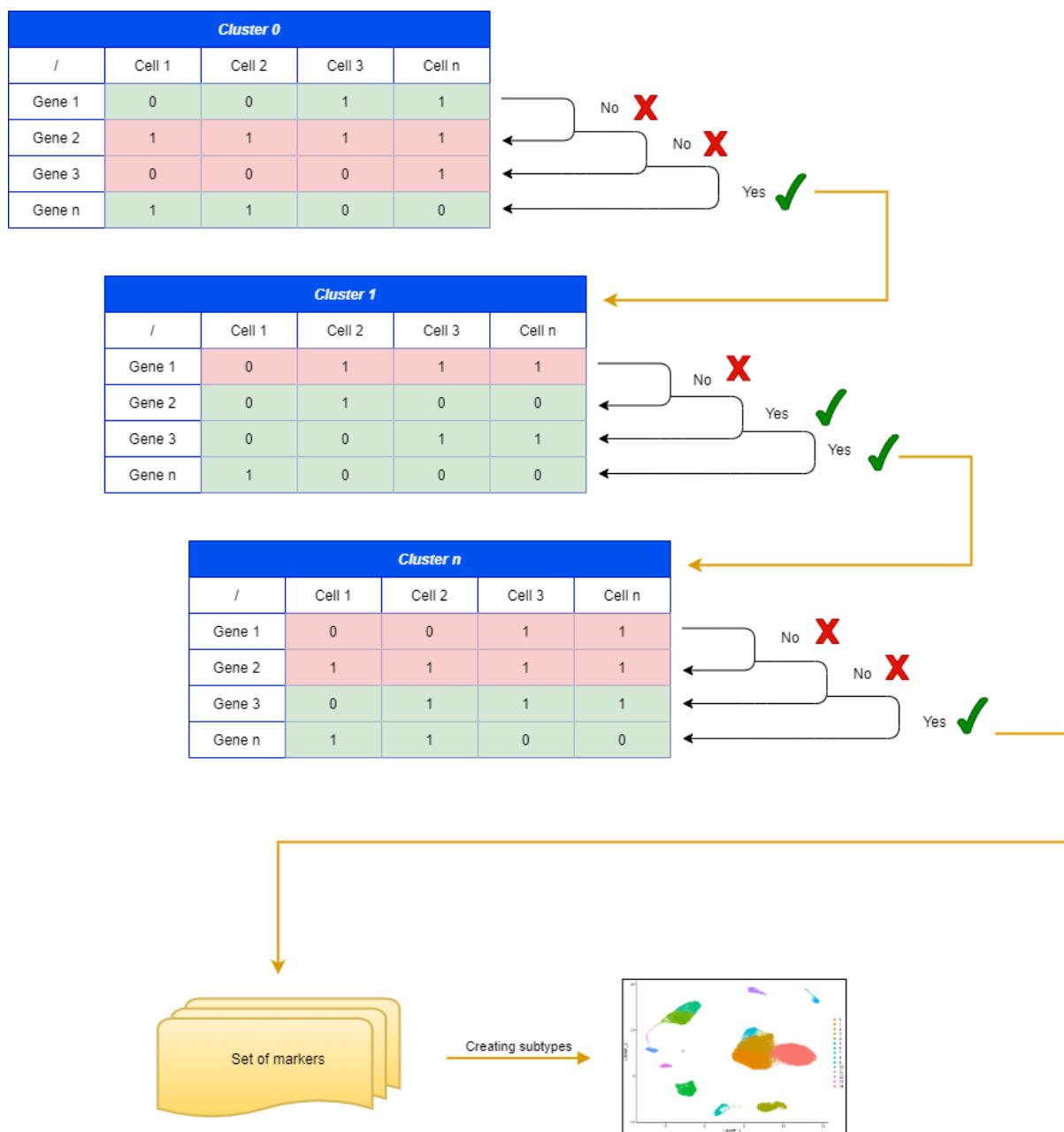
In this plot, are presented cell classes and subclasses after naming. Below is showed scheme of the cell name.



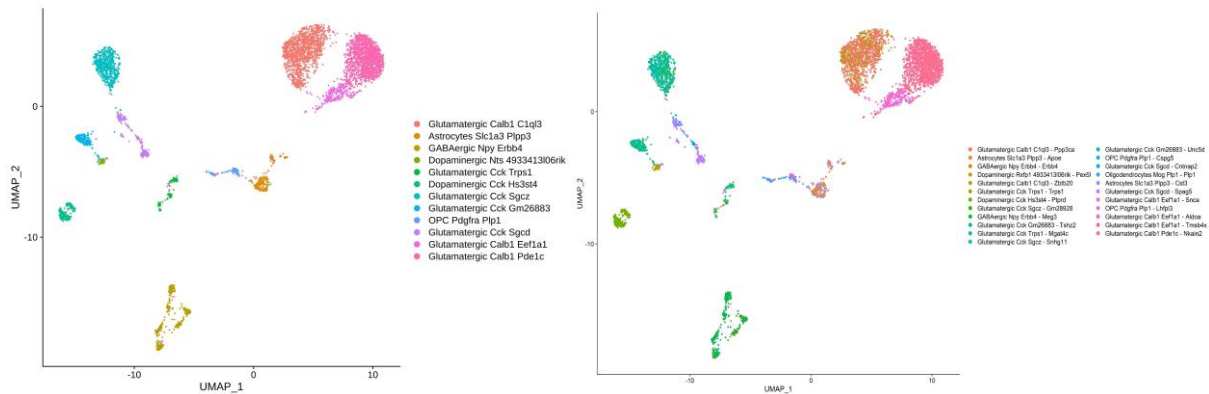
6.11. Dividing the cells subclasses (clusters) into cells subtypes

Due to the high heterogeneity inside clusters obtained in the fundamental single-cell analysis, the Cell Subtypes Selection by Genes (CSSG) algorithm was written. The CSSG algorithm is based on markers selected separately for each cluster by the MAST algorithm. The algorithm creates a binary matrix where 0 means no expressed genes and 1 means genes with an expression not equal to 0. Then, for each cluster, the best combinations of genes are selected in successive iterations, explaining the most significant number of cell subtypes using the smallest possible number of genes at the level of significance in the range $p = 0.001-0.1$.

Example of CSSG algorithm iteration for dominant genes selection in the cell subtypes



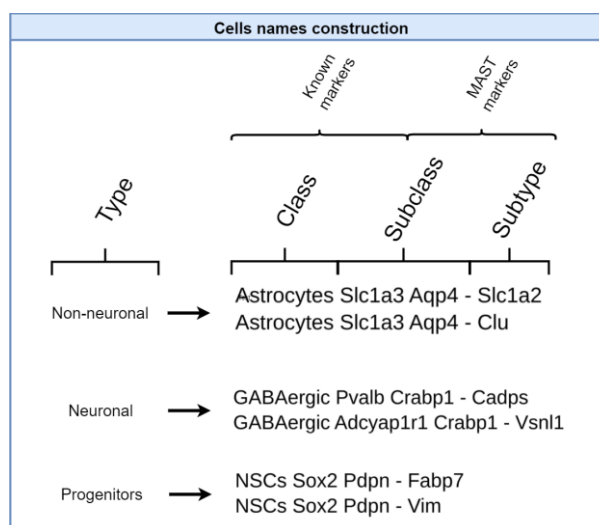
Data presentation with and without CSSG algorithm



The JSEQ® CSSG algorithm was tested on 812 945 single cells from different brain regions. Results after using the JSEQ® CSSG algorithm show that the algorithm can improve the resolution of obtained results. Based on the obtained result, we can see that algorithm allow:

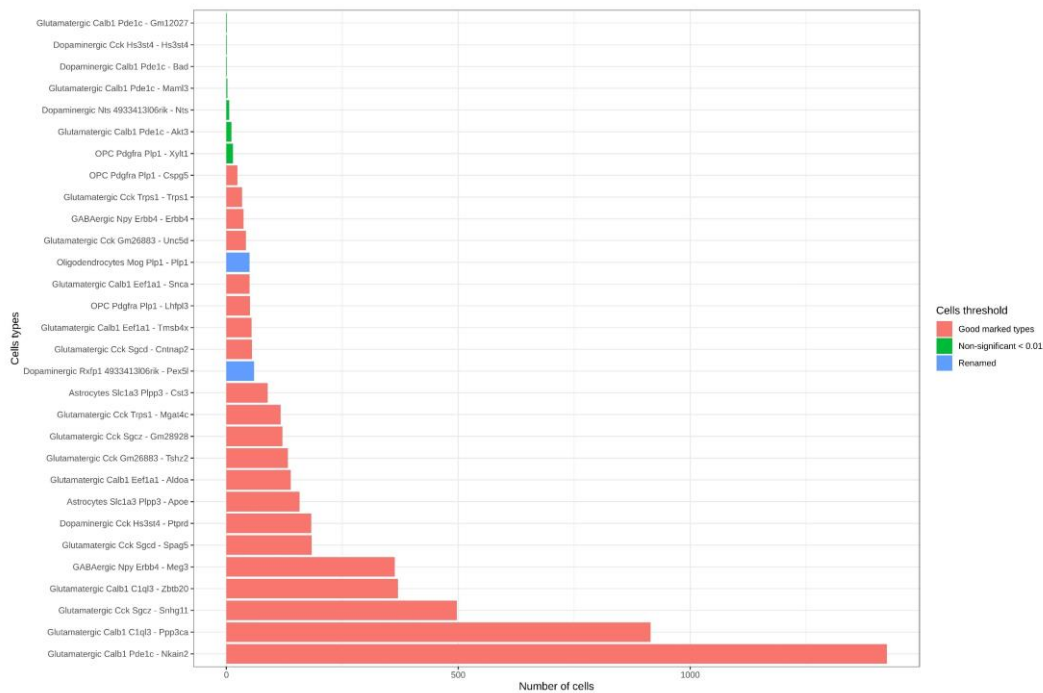
- split cells subclasses into two different subclasses (e.g., from oligodendrocytes to oligodendrocytes and OPCs) or subtypes with different development stages,
- based on dominant marker shows a difference in cell's maturation stage,
- shows the different status of obtained cell subtypes, e.g., based on signaling peptides secretion in the neuronal cells.

6.12. Construction of cell subtypes names



6.13. Outliers dropping

This pipeline contains many checkpoints that protect against lousy quality or badly clustered cells. Even though duplicates removing at the beginning and additional selection points in the pipeline were projected as another quality control step. After dividing cells populations with the CSSG algorithm, obtained cells subtypes in new clusters are checked in terms of proper names. Cell subtypes groups that were poorly marked or their names were changed are renamed to the correct form. Furthermore, when new cell subtypes do not express markers selected by CSSG, they drop out from the analysis. Also, cells subtypes whose amount is lower than 0.01 amount of the subtype with the most number of cells in the analysis are dropped.

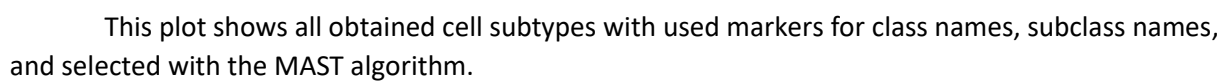


This plot shows how many cells were good marked, bad marked, renamed, and non-significant.

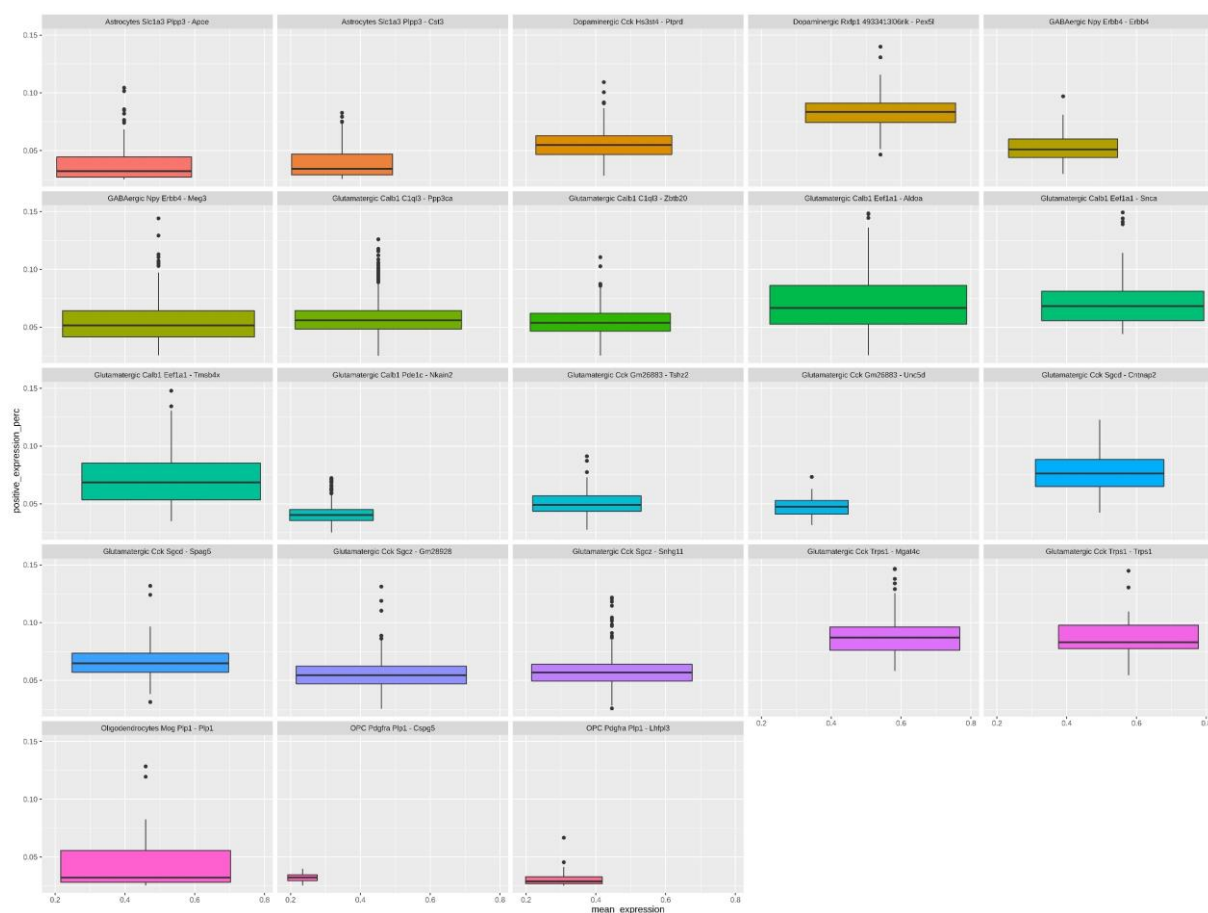
All dropped cells from the analysis are saved to a new dataset in the repository:

'JSEQ_scRNAseq/projects/*project_name*/results/exp_matrix/unknow_cells_count_matrix.xlsx'

Pheatmap



Boxplots for genes ratio and level expression



It is the last plot in the analysis, which shows the x-axis percent of expressed genes for all cells included in this cell subtype and the y-axis mean expression value for all cells in this subtype. Thanks to this plot, we can check the heterogeneity/homogeneity obtained in analysis cell subtypes.

7. References:

Graphics:

Biorender

<https://biorender.com/>

Used software and scripts:

DropSeqPipe

<https://github.com/Hoohm/dropSeqPipe>

STAR

<https://github.com/alexdobin/STAR>

Drop-seq

<https://github.com/broadinstitute/Drop-seq>

Picard

<https://github.com/broadinstitute/picard>

fastp

<https://github.com/OpenGene/fastp>

Seurat

<https://github.com/satijalab/seurat>

Publications:

Anderson, A. G., Kulkarni, A., Harper, M., & Konopka, G. (2020). Single-Cell Analysis of Foxp1-Driven Mechanisms Essential for Striatal Development. *Cell Reports*, 30(9), 3051-3066.e7. <https://doi.org/10.1016/j.celrep.2020.02.030>

Artegiani, B., Lyubimova, A., Muraro, M., van Es, J. H., van Oudenaarden, A., & Clevers, H. (2017). A Single-Cell RNA Sequencing Study Reveals Cellular and Molecular Dynamics of the Hippocampal Neurogenic Niche. *Cell Reports*, 21(11), 3271–3284. <https://doi.org/10.1016/j.celrep.2017.11.050>

Carter, R. A., Bihannic, L., Rosencrance, C., Hadley, J. L., Tong, Y., Phoenix, T. N., Natarajan, S., Easton, J., Northcott, P. A., & Gawad, C. (2018). A Single-Cell Transcriptional Atlas of the Developing Murine Cerebellum. *Current Biology*, 28(18), 2910-2920.e2. <https://doi.org/10.1016/j.cub.2018.07.062>

Chen, R., Wu, X., Jiang, L., & Zhang, Y. (2017). Single-Cell RNA-Seq Reveals Hypothalamic Cell Diversity. *Cell Reports*, 18(13), 3227–3241. <https://doi.org/10.1016/j.celrep.2017.03.004>

Ehman, E. C., Johnson, G. B., Villanueva-meyer, J. E., Cha, S., Leynes, A. P., Eric, P., Larson, Z., & Hope, T. A. (2017). A Molecular Census of Arcuate Hypothalamus and Median Eminence Cell Types.

- Nature Neuroscience*, 20(3), 484–496. <https://doi.org/10.1038/nn.4495.A>
- Eze, U., Bhaduri, A., Haeussler, M., Nowakowski, T., & Kriegstein, A. (2020). *Single-Cell Atlas of Early Human Brain Development Highlights Heterogeneity of Human Neuroepithelial Cells and Early Radial Glia*. <https://doi.org/10.1101/2020.03.06.981423>
- Fan, X., Fu, Y., Zhou, X., Sun, L., Yang, M., Wang, M., Chen, R., Wu, Q., Yong, J., Dong, J., Wen, L., Qiao, J., Wang, X., & Tang, F. (2020). Single-cell transcriptome analysis reveals cell lineage specification in temporal-spatial patterns in human cortical development. *Science Advances*, 6(34), 1–16. <https://doi.org/10.1126/sciadv.aaz2978>
- Finak, G., McDavid, A., Yajima, M., Deng, J., Gersuk, V., Shalek, A. K., Slichter, C. K., Miller, H. W., McElrath, M. J., Prlic, M., Linsley, P. S., & Gottardo, R. (2015). MAST: A flexible statistical framework for assessing transcriptional changes and characterizing heterogeneity in single-cell RNA sequencing data. *Genome Biology*, 16(1), 1–13. <https://doi.org/10.1186/s13059-015-0844-5>
- Gokce, O., Stanley, G. M., Treutlein, B., Neff, N. F., Camp, J. G., Malenka, R. C., Rothwell, P. E., Fuccillo, M. V., Südhof, T. C., & Quake, S. R. (2016). Cellular Taxonomy of the Mouse Striatum as Revealed by Single-Cell RNA-Seq. *Cell Reports*, 16(4), 1126–1137. <https://doi.org/10.1016/j.celrep.2016.06.059>
- Guo, Q., & Li, J. Y. H. (2019). Defining developmental diversification of diencephalon neurons through single cell gene expression profiling. *Development (Cambridge)*, 146(12). <https://doi.org/10.1242/dev.174284>
- Habib, N., Avraham-Davidi, I., Basu, A., Burks, T., Shekhar, K., Hofree, M., Choudhury, S. R., Aguet, F., Gelfand, E., Ardlie, K., Weitz, D. A., Rozenblatt-Rosen, O., Zhang, F., & Regev, A. (2017). Massively parallel single-nucleus RNA-seq with DroNc-seq. *Nature Methods*, 14(10), 955–958. <https://doi.org/10.1038/nmeth.4407>
- Harris, K. D., Hochgerner, H., Skene, N. G., Magno, L., Katona, L., Gonzales, C. B., Somogyi, P., Kessaris, N., Linnarsson, S., & Hjerling-Leffler, J. (2017). Classes and continua of hippocampal CA1 inhibitory neurons revealed by single-cell transcriptomics. In *bioRxiv*. <https://doi.org/10.1101/143354>
- Kalish, B. T., Cheadle, L., Hrvatin, S., Nagy, M. A., Rivera, S., Crow, M., Gillis, J., Kirchner, R., & Greenberg, M. E. (2018). Single-cell transcriptomics of the developing lateral geniculate nucleus reveals insights into circuit assembly and refinement. *Proceedings of the National Academy of Sciences of the United States of America*, 115(5), E1051–E1060. <https://doi.org/10.1073/pnas.1717871115>
- Keo, A., Ahmad Aziz, N., Dzyubachyk, O., Van Der Grond, J., Van Roon-Mom, W. M. C., Lelieveldt, B. P. F., Reinders, M. J. T., & Mahfouz, A. (2017). Co-expression patterns between ATN1 and ATXN2 coincide with brain regions affected in huntington's disease. *Frontiers in Molecular Neuroscience*, 10(November), 1–13. <https://doi.org/10.3389/fnmol.2017.00399>
- Kishimoto, T. E., Uchida, K., Thongtharb, A., Shibato, T., Chambers, J. K., Nibe, K., Kagawa, Y., & Nakayama, H. (2018). Expression of Oligodendrocyte Precursor Cell Markers in Canine Oligodendrogliomas. *Veterinary Pathology*, 55(5), 634–644. <https://doi.org/10.1177/0300985818777794>
- Koirala, S., & Corfas, G. (2010). Identification of novel glial genes by single-cell transcriptional profiling of Bergmann glial cells from mouse cerebellum. *PLoS ONE*, 5(2). <https://doi.org/10.1371/journal.pone.0009198>
- Kozareva, V., Martin, C., Osorno, T., Rudolph, S., Guo, C., Vanderburg, C., Nadaf, N., Regev, A., Regehr,

- W., & Macosko, E. (2020). A transcriptomic atlas of the mouse cerebellum reveals regional specializations and novel cell types. *BioRxiv*. <https://doi.org/10.1101/2020.03.04.976407>
- Langlet, F. (2019). Tanycyte gene expression dynamics in the regulation of energy homeostasis. *Frontiers in Endocrinology*, 10(MAY). <https://doi.org/10.3389/fendo.2019.00286>
- Martínez-Cerdeño, V., & Noctor, S. C. (2018). Neural progenitor cell terminology. *Frontiers in Neuroanatomy*, 12(December), 1–8. <https://doi.org/10.3389/fnana.2018.00104>
- McKenzie, A. T., Wang, M., Hauberg, M. E., Fullard, J. F., Kozlenkov, A., Keenan, A., Hurd, Y. L., Dracheva, S., Casaccia, P., Roussos, P., & Zhang, B. (2018). Brain Cell Type Specific Gene Expression and Co-expression Network Architectures. *Scientific Reports*, 8(1), 1–19. <https://doi.org/10.1038/s41598-018-27293-5>
- Mickelsen, L. E., Bolisetty, M., Chimileski, B. R., Fujita, A., Eric, J., Costanzo, J. T., Naparstek, J. R., Robson, P., & Alexander, C. (2020). *HHS Public Access*. 22(4), 642–656. <https://doi.org/10.1038/s41593-019-0349-8>.Single-cell
- Muñoz-Manchado, A. B., Bengtsson Gonzales, C., Zeisel, A., Munguba, H., Bekkouche, B., Skene, N. G., Lönnerberg, P., Ryge, J., Harris, K. D., Linnarsson, S., & Hjerling-Leffler, J. (2018). Diversity of Interneurons in the Dorsal Striatum Revealed by Single-Cell RNA Sequencing and PatchSeq. *Cell Reports*, 24(8), 2179–2190.e7. <https://doi.org/10.1016/j.celrep.2018.07.053>
- Narayan, A., Berger, B., & Cho, H. (2020). Density-preserving data visualization unveils dynamic patterns of single-cell transcriptomic variability. *BioRxiv*, 1–50. <https://doi.org/10.1101/2020.05.12.077776>
- Nelson, B. R., Hodge, R. D., Daza, R. A. M., Tripathi, P. P., Arnold, S. J., Millen, K. J., & Hevner, R. F. (2020). Intermediate progenitors support migration of neural stem cells into dentate gyrus outer neurogenic niches. *ELife*, 9, 1–30. <https://doi.org/10.7554/eLife.53777>
- Peng, J., Sheng, A. L., Xiao, Q., Shen, L., Ju, X. C., Zhang, M., He, S. T., Wu, C., & Luo, Z. G. (2019). Single-cell transcriptomes reveal molecular specializations of neuronal cell types in the developing cerebellum. *Journal of Molecular Cell Biology*, 11(8), 636–648. <https://doi.org/10.1093/jmcb/mjy089>
- Smith, S. J., Smbül, U., Graybuck, L. T., Collman, F., Seshamani, S., Gala, R., Gliko, O., Elabbady, L., Miller, J. A., Bakken, T. E., Rossier, J., Yao, Z., Lein, E., Zeng, H., Tasic, B., & Hawrylycz, M. (2019). Single-cell transcriptomic evidence for dense intracortical neuropeptide networks. *ELife*, 8(1), 1–35. <https://doi.org/10.7554/eLife.47889>
- Takeuchi, A., Takahashi, Y., Iida, K., Hosokawa, M., Irie, K., Ito, M., Brown, J. B., Ohno, K., Nakashima, K., & Hagiwara, M. (2020). Identification of Qk as a Glial Precursor Cell Marker that Governs the Fate Specification of Neural Stem Cells to a Glial Cell Lineage. *Stem Cell Reports*, 15(4), 883–897. <https://doi.org/10.1016/j.stemcr.2020.08.010>
- Zeisel, A., Hochgerner, H., Lönnerberg, P., Johnsson, A., Memic, F., van der Zwan, J., Häring, M., Braun, E., Borm, L. E., La Manno, G., Codeluppi, S., Furlan, A., Lee, K., Skene, N., Harris, K. D., Hjerling-Leffler, J., Arenas, E., Ernfors, P., Marklund, U., & Linnarsson, S. (2018). Molecular Architecture of the Mouse Nervous System. *Cell*, 174(4), 999–1014.e22. <https://doi.org/10.1016/j.cell.2018.06.021>
- Zhu, X., Zhang, J., Xu, Y., Wang, J., Peng, X., & Li, H. D. (2020). Single-Cell Clustering Based on Shared Nearest Neighbor and Graph Partitioning. *Interdisciplinary Sciences: Computational Life Sciences*, 12(2), 117–130. <https://doi.org/10.1007/s12539-019-00357-4>