ScRNA-seq workflow CONCLUS: from CONsensus CLUSters to a meaningful CONCLUSion

Ilyess Rachedi, Polina Pavlovich, Nicolas Descostes, and Christophe Lancrin

29 ottobre 2020

Contents

1 Introduction

CONCLUS is a tool for robust clustering and positive marker features selection of single-cell RNA-seq (sc-RNA-seq) datasets. Of note, CONCLUS does not cover the preprocessing steps of sequencing files obtained following next-generation sequencing. You can find a good resource to start with here.

CONCLUS is organized into the following steps:

- Generation of multiple t-SNE plots with a range of parameters including different selection of genes extracted from PCA.
- Use the Density-based spatial clustering of applications with noise (DBSCAN) algorithm for idenfication of clusters in each generated t-SNE plot.
- All DBSCAN results are combined into a cell similarity matrix.
- The cell similarity matrix is used to define "CONSENSUS" clusters conserved accross the previously defined clustering solutions.
- Identify marker genes for each concensus cluster.

2 Getting help

Issues can be submitted directly on the Bioconductor forum using the keyword 'conclus' in the post title. To contact us directly write to christophe.lancrin@embl.it or ilyessr@hotmail.fr. The principles of this package were originally developed by Polina Pavlovich who is now doing her Ph.D at the Max Planck Institute of Immunobiology and Epigenetics.

3 Important note

Due to the stochastic aspect of the tSNE, images of the plot are directly included (and not generated by the code) to make the descriptions consistent. You might therefore get slightly different plots. However, you should obtain the same marker genes at the end of the process.

4 Standard workflow

4.1 Quick start

CONCLUS requires to start with a raw-count matrix with reads or unique molecular identifiers (UMIs). The columns of the count matrix must contain cells and the rows – genes. CONCLUS needs a large number of cells to collect statistics, we recommend using CONCLUS if you have at least 100 cells.

In the example below, a small toy example is used to illustrate the runCONCLUS method. Real data are used later in this vignette.

```
library(conclus)
```

```
outputDirectory <- "./testDirectory"
experimentName <- "Test"
species <- "mouse"

countMatrix <- as.matrix(read.delim(file.path(system.file("extdata",
                                package = "conclus"), "test_countMatrix.tsv"),
                        stringsAsFactors = FALSE))

columnsMetaData <- read.delim(file.path(system.file("extdata",
                        package = "conclus"), "test_colData_filtered.tsv"))

sceObjectCONCLUS <- runCONCLUS(outputDirectory, experimentName, countMatrix,
        species, columnsMetaData = columnsMetaData)
## ## Building the single-cell RNA-Seq object (step 1/13) ##
## ## Performing the normalization (step 2/13) ##
##   Note: The connection to biomaRt can take a while sometimes.
## # Attempt 1/5 # Connection to Ensembl ...
## Connected with success.
## Annotating 1 genes containing ENSMUSG pattern.
## Annotating 499 genes considering them as SYMBOLs.
## 'select()' returned 1:many mapping between keys and columns
## # Attempt 1/5 # Retrieving information about genes from biomaRt ...
## Information retrieved with success.
## Adding cell info for cells filtering.
## Running filterCells.
## Running filterGenes.
## Running normalization. It can take a while depending on the number of cells.
## summary(sizeFactors(sceObject)):
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.2524  0.6160  0.8598  1.0000  1.1987  3.3547
## ## Calculating all tSNEs (step 3/13) ##
## Running TSNEs using 2 cores.
## Calculated 14 2D-tSNE plots.
## Building TSNEs objects.
## ## Clustering with DbScan (step 4/13) ##
## ## Computing the cells similarity matrix (step 5/13) ##
## Calculating cells similarity matrix.
## Assigning cells to 10 clusters.
## Cells distribution by clusters:
##  1  2  3  4  5  6  7  8  9 10
## 54 47  6  6  5 34 10 11  7  9
## ## Computing the clusters similarity matrix (step 6/13) ##
## ## Ranking genes (step 7/13) ##
## Ranking marker genes for each cluster.
## Working on cluster 1
## Working on cluster 2
## Working on cluster 3
## Working on cluster 4
## Working on cluster 5
## Working on cluster 6
## Working on cluster 7
```

```
## Working on cluster 8
## Working on cluster 9
## Working on cluster 10
## ## Getting marker genes (step 8/13) ##
## ## Getting genes info (step 9/13) ##
##   Note: The connection to biomaRt can take a while sometimes.
## # Attempt 1/5 # Connection to Ensembl ...
## Connected with success.
## # Attempt 1/5 # Retrieving information about genes from biomaRt ...
## Information retrieved with success.
## # Attempt 1/5 # Retrieving information about genes from biomaRt ...
## Information retrieved with success.
## ## Plot the cell similarity matrix (step 10/13) ##
## ## Plot the cell heatmap (step 11/13) ##
## ## Plot the clusters similarity heatmap (step 12/13) ##
## Warning in dir.create(outputDirectory): './testDirectory' already exists
## ## Plot clustered tSNE (step 13/13) ##
## Exporting all results to ./testDirectory
## Saving all results.
## Normalized expression matrix saved.
## RowData saved.
## ColData saved.
## Tsne coordinates saved.
## dbScan clustering saved.
## cellsSimilarityMatrix saved.
## clustersSimilarityMatrix saved.
## Clusters table saved.
## Full marker lists saved.
## Top markers saved.
## Genes infos saved.
## Done.
```

In your "outputDirectory", in the folder `Resuts`, you will find a `heatmaps_results.pdf` with the cells similarity, the clusters similarity, and the cells heatmap. The sub-folder `pictures` contains all tSNE with dbscan coloration. You will also find sub-folders containing:

- `1_MatrixInfo`: The normalized count matrix and its meta-data for both rows and columns.
- `2_TSNECoordinates`: The tSNE coordinates for each parameter of principal components (PCs) and perplexities.
- `3_dbScan`: The different clusters given by DBscan according to different parameters. Each file gives a cluster number for each cell.
- `4_CellSimilarityMatrix`: The matrix underlying the cells similarity heatmap.
- `5_ClusterSimilarityMatrix`: The matrix underlying the clusters similarity heatmap.
- `6_ConclusResult`: A table containing the result of the consensus clustering. This table contains two columns: clusters-cells.
- `7_fullMarkers`: Files containing markers for each cluster, defined by the consensus clustering.
- `8_TopMarkers`: Files containing the top 10 markers for each cluster.
- `9_genesInfos`: Files containing gene information for the top markers defined in the previous folder.

Further details about how all results are generated can be found below.

4.2 Data

In this vignette, we demonstrate how to use CONCLUS on a sc-RNA-seq dataset from *Bergiers et al. eLife 2018*. The design for this experiment is described in (*Figure 4—figure supplement 2*). Bergiers et al. goal was to analyze the effect of the simultaneous expression of eight transcription factors (8TFs): *Runx1* - and its partner - *Cbfb*, *Gata2*, *Tal1*, *Fli1*, *Lyl1*, *Erg* and *Lmo2* in *in vitro* differentiated embryonic stem cells (ESCs) in comparison to control. They knocked-in a polycistronic transgenic construct allowing to over-express eigth transcription factors (i8TFs) simultaneously after adding doxycycline (dox). The Empty ESC line did not have any transgene. There were **four conditions**: E_minus (Empty no dox), E_plus (Empty with dox), i8TFs_minus (i8TFs no dox) and i8TFs_plus (i8TFs with dox).

This sc-RNA-seq experiment was performed using the SMARTer ICELL8 Single-Cell System (Click here for more info). The protocol was based on 3' end RNA sequencing where each mRNA molecule is labeled with a unique molecular identifier (UMI) during reverse transcription in every single cell. The analysis performed by *Bergiers et al.* was based on the dimensionality reduction algorithm called Principal Component Analysis (PCA), and they found that there was a major gene expression difference between i8TFs_plus and the other three conditions (*Figure 4—figure supplement 2*). **However, it was not clear if other sub-clusters could be identified consistently in this dataset besides the two major clusters. In the current tutorial, we show how CONCLUS can help to answer this question.**

Labels of the four conditions are in the *state* column of *columnsMetaData*. To avoid a bias in the clustering analysis due to the high expression of the eight transcription factors construct, we deleted genes *Cbfb*, *Gata2*, *Tal1*, *Fli1*, *Lyl1*, *Erg*, and *Lmo2*. Highly abundant embryonic hemoglobins with names starting with "Hba" or "Hbb" were also excluded because they seemed to be a primary source of contamination.

The code below format the count matrix and the columns meta-data. The source data are downloaded from the GEO page GSE96982. The URL of the count matrix was retrieved by right-click and 'copy link adress' on the ftp hyperlink of the supplementary file GSE96982_countMatrix.txt.gz at the bottom of the page. The name of the series matrix, containing columns meta-data was retrieved by clicking the 'Series Matrix File(s)' link just above the count matrix. Another complete tutorial is given for retrieving and processing data of tabulamuris on the landing page of the package on the Bioconductor website. The function `retrieveFromGEO` will download all series matrices present in the GEO record however only the one of interest will be kept.

```
outputDirectory <- "./YourOutputDirectory"
dir.create(outputDirectory, showWarnings=FALSE)
species <- "mouse"

countMatrixPath <- file.path(outputDirectory, "countmatrix.txt")
matrixURL <- paste0("https://www.ncbi.nlm.nih.gov/geo/download/?acc=GSE96982",
"&format=file&file=GSE96982%5FcountMatrix%2Etxt%2Egz")
seriesMatrix <- "GSE96982-GPL19057_series_matrix.txt.gz"

result <- retrieveFromGEO(matrixURL, countMatrixPath, seriesMatrix, species)
## Downloading the count matrix.
```

```
## Downloading the columns meta-data.
## Found 3 file(s)
## GSE96982-GPL18573_series_matrix.txt.gz
## Using locally cached version: /tmp/Rtmppm8JOe/GSE96982-GPL18573_series_matrix.txt.gz
##
## -- Column specification --------------------------------------------------------------------------------
## cols(
##   ID_REF = col_character(),
##   GSM2548569 = col_character(),
##   GSM2548570 = col_character(),
##   GSM2548571 = col_character(),
##   GSM2548572 = col_character(),
##   GSM2548573 = col_character(),
##   GSM2548574 = col_character(),
##   GSM2548575 = col_character(),
##   GSM2548576 = col_character(),
##   GSM2548577 = col_character(),
##   GSM2548578 = col_character(),
##   GSM2548579 = col_character(),
##   GSM2548580 = col_character(),
##   GSM2548581 = col_character(),
##   GSM2548582 = col_character(),
##   GSM2548583 = col_character()
## )
## Using locally cached version of GPL18573 found here:
## /tmp/Rtmppm8JOe/GPL18573.soft
## GSE96982-GPL19057_series_matrix.txt.gz
## Using locally cached version: /tmp/Rtmppm8JOe/GSE96982-GPL19057_series_matrix.txt.gz
##
## -- Column specification --------------------------------------------------------------------------------
## cols(
##   .default = col_character()
## )
## i Use `spec()` for the full column specifications.
## Using locally cached version of GPL19057 found here:
## /tmp/Rtmppm8JOe/GPL19057.soft
## GSE96982-GPL24755_series_matrix.txt.gz
## Using locally cached version: /tmp/Rtmppm8JOe/GSE96982-GPL24755_series_matrix.txt.gz
##
## -- Column specification --------------------------------------------------------------------------------
## cols(
##   ID_REF = col_character(),
##   GSM2548584 = col_character(),
##   GSM2548585 = col_character(),
##   GSM2548586 = col_character(),
##   GSM2548587 = col_character()
## )
## Using locally cached version of GPL24755 found here:
## /tmp/Rtmppm8JOe/GPL24755.soft
## Formating data.
## Converting ENSEMBL IDs to symbols.
```

```
## 'select()' returned 1:many mapping between keys and columns
## Warning in clusterProfiler::bitr(matrixSym, fromType = annoType, toType =
## c("SYMBOL"), : 13.54% of input gene IDs are fail to map...
## Warning in retrieveFromGEO(matrixURL, countMatrixPath, seriesMatrix, species):
## Nb of lines removed due to duplication of row names: 8
countMatrix <- result[[1]]
columnsMetaData <- result[[2]]

## Removing the 8 TFs:
TFtoRemove <- c("Runx1", "Cbfb", "Gata2", "Tal1", "Fli1", "Lyl1", "Erg",
"Lmo2")
idxTF <- match(TFtoRemove, rownames(countMatrix))
countMatrix <- countMatrix[-idxTF,]

## Removing embryonic hemoglobins with names starting with "Hba" or "Hbb"
idxHba <- grep("Hba", rownames(countMatrix))
idxHbb <- grep("Hbb", rownames(countMatrix))
countMatrix <- countMatrix[-c(idxHba, idxHbb),]
```

4.3 Test clustering

The *TestClustering* function runs one clustering round out of the 84 (default) rounds that CONCLUS normally performs. This step can be useful to determine if the default DBSCAN parameters are suitable for your dataset. By default, they are *dbscanEpsilon = c(1.3, 1.4, 1.5)* and *minPts = c(3,4)*. If the dashed horizontal line in the k-NN distance plot lays on the "knee" of the curve (as shown below), it means that optimal epsilon is equal to the intersection of the line to the y-axis. In our example, optimal epsilon is 1.4 for 5-NN distance where 5 corresponds to MinPts.

In the "test_clustering" folder under outputDirectory, the three plots below will be saved where one corresponds to the "distance_graph.pdf", another one to "test_tSNE.pdf" (*p[[1]]*), and the last one will be saved as "test_clustering.pdf" (*p[[2]]*).

```
## Creation of the single-cell RNA-Seq object
scr <- singlecellRNAseq(experimentName = "Bergiers",
        countMatrix     = countMatrix,
        species         = "mouse",
        outputDirectory = outputDirectory)

## Normalization of the count matrix
scr <- normaliseCountMatrix(scr, coldata=columnsMetaData)
## # Attempt 1/5 # Connection to Ensembl ...
## Connected with success.
## Annotating 2200 genes containing ENSMUSG pattern.
## Annotating 14019 genes considering them as SYMBOLs.
## 'select()' returned 1:many mapping between keys and columns
## # Attempt 1/5 # Retrieving information about genes from biomaRt ...
## Information retrieved with success.
## Adding cell info for cells filtering.
```

```
## Running filterCells.
## Running filterGenes.
## Running normalization. It can take a while depending on the number of cells.
## summary(sizeFactors(sceObject)):
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.2505  0.4783  0.7945  1.0000  1.3178  4.2054
```

```
p <- testClustering(scr, writeOutput=TRUE, silent=TRUE)
## Generating TSNE.
## Calculated 1 2D-tSNE plots.
## Saving results tSNE.
## Saving results distance graph.
## Saving dbscan results.
```

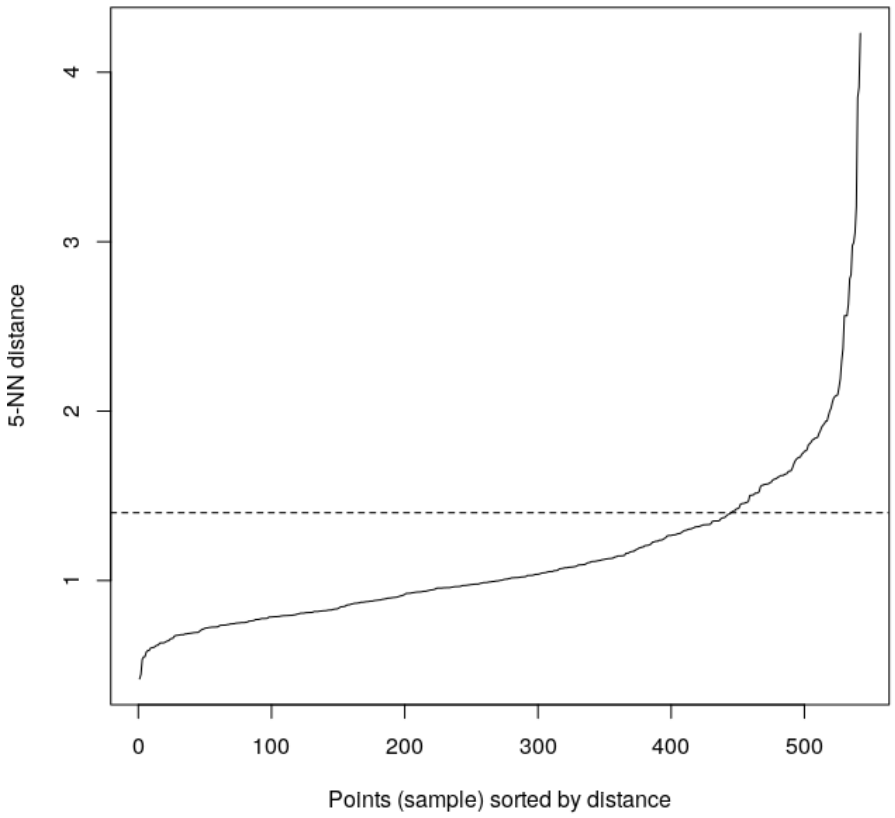

Figure 1: Knn plot
If the dashed horizontal line in the k-NN distance plot lays on the "knee" of the curve, it means that optimal epsilon is equal to the intersection of the line to the y-axis.

```
# saved as "outputDirectory/test_clustering/test_tSNE.pdf"
p[[1]]
```

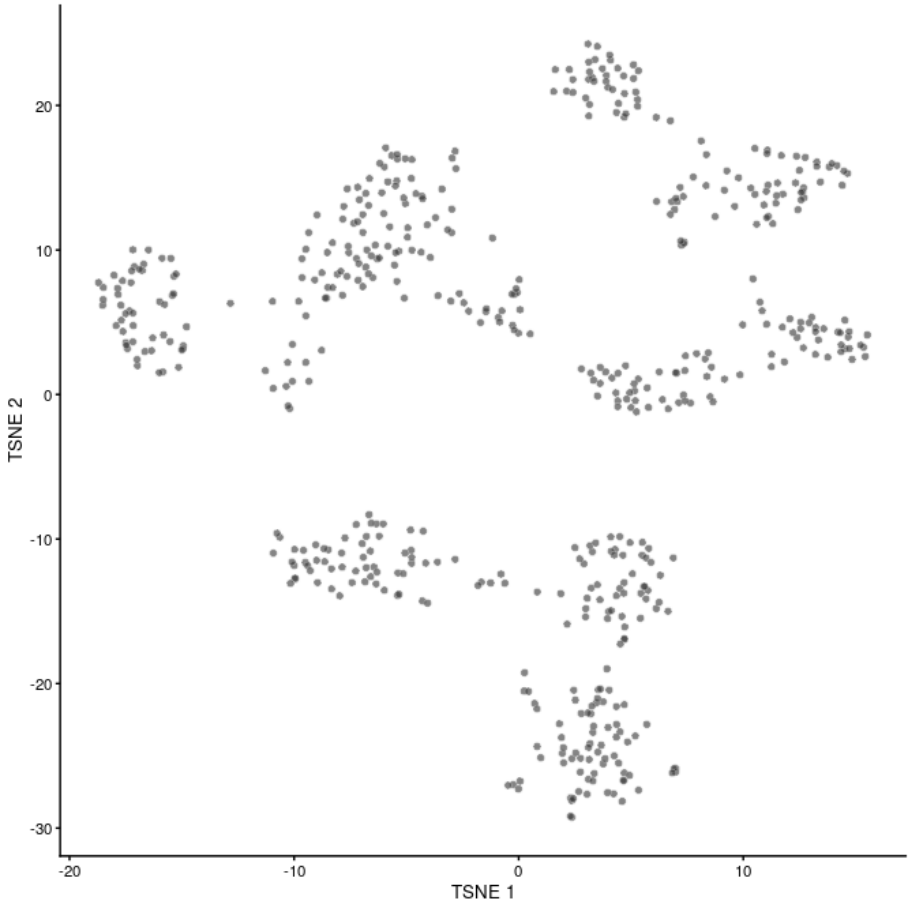

Figure 2: tsne One of the 14 tSNE (by default) generated by conclus

```
# saved as "outputDirectory/test_clustering/test_clustering.pdf"
p[[2]]
```

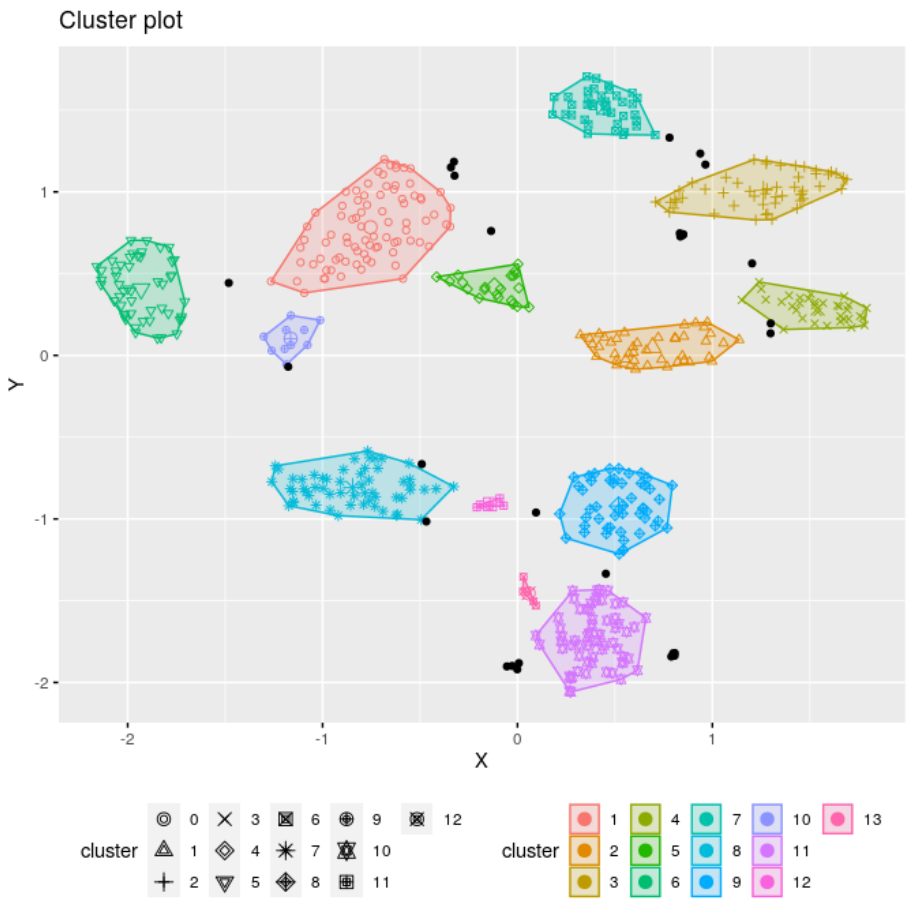

Figure 3: clusterplot One of the 84 dbscan clustering solutions generated by conclus

5 CONCLUS step by step

The wrapper function runCONCLUS is organized into 7 steps:

- Normalization of the counts matrix
- Generation of t-SNE coordinates
- Clustering with DB-SCAN
- Cell and cluster similarity matrix calculation
- Plotting
- Marker genes identification
- Results export

5.1 Normalization of the counts matrix

sc-RNA-seq datasets are quite challenging notably because of sparsity (many genes are not detected consistently yielding expression matrices with many zeroes) and also because of technical noise. To facilitate analysis, one needs to perform a step of normalization which allows for the correction of unwanted technical and biological noises (click here for a complete review on normalization techniques).

CONCLUS uses Scran and Scater packages for normalization. Beforehand, the function will annotate genes creating *rowData* and add statistics about cells into *columnsMetaData*. If you already have *columnsMetaData* and *rowData*, you can give it to the function (see manual). It will keep your columns and add new ones at the end. If you do not want to lose any cell after quality metrics check, select *alreadyCellFiltered* = *TRUE*, by default it is *FALSE*. Before *scran* and *scater* normalization, the function will call *scran::quickCluster* (see manual for details). If you want to skip this step, set *runQuickCluster* = *FALSE*, by default it is *TRUE*. We recommend to use *runQuickCluster* = *TRUE* for medium-size datasets with 500-10000 cells. However, it can take a significant amount of time for a larger amount of cells and will not be useful for small sets of 200-300 samples.

```
scr <- normaliseCountMatrix(scr, coldata=columnsMetaData)
```

The method *normaliseCountMatrix* returns a scRNASeq object with its sceNorm slot updated. This slot contains a SingleCellExperiment object having the normalized count matrix, the colData (table with cells informations), and the rowData (table with the genes informations). See ?SingleCellExperiment for more details.

The rowdata can help to study cross-talk between cell types or find surface protein-coding marker genes suitable for flow cytometry. The columns with the GO terms are *go_id* and *name_1006* (see manual).

The slots can be accessed as indicated below:

```
## Accessing slots
originalMat <- getCountMatrix(scr)
SCEobject <- getSceNorm(scr)
normMat <- SingleCellExperiment::logcounts(SCEobject)

# checking what changed after the normalisation
dim(originalMat)
## [1] 16219   654
dim(normMat)
## [1] 9871  536

# show first columns and rows of the count matrix
originalMat[1:5,1:5]
##       c1 c2 c3 c4 c5
## Gnai3  0  0  0  0  1
## Cdc45  0  0  1  1  0
## H19    0  0  0  0  0
## Scml2  0  0  0  0  0
## Narf   0  0  0  0  0

# show first columns and rows of the normalized count matrix
normMat[1:5,1:5]
##       c1 c2        c3       c4      c5
## Gnai3  0  0 0.0000000 0.000000 1.70267
## Cdc45  0  0 0.7997804 1.810582 0.00000
## H19    0  0 0.0000000 0.000000 0.00000
## Narf   0  0 0.0000000 0.000000 0.00000
## Cav2   0  0 0.0000000 0.000000 0.00000
```

```
# visualize first rows of metadata (coldata)
coldataSCE <- as.data.frame(SummarizedExperiment::colData(SCEobject))
head(coldataSCE)
##    cellName   state cellBarcode genesNum genesSum oneUMI oneUMIper mtGenes
## c1       c1 E_minus AACCAATCGTC      952     1937    692  72.68908       6
## c2       c2 E_minus AACCAGATTCC     1065     2195    771  72.39437       7
## c3       c3 E_minus AACCAGCAACT     2225     6748   1386  62.29213       7
## c4       c4 E_minus AACCAGTCAGG      920     1751    644  70.00000       5
## c5       c5 E_minus AACCATCTATT     1060     2074    761  71.79245       7
## c6       c6 E_minus AACCATTGGCT     1849     5486   1188  64.25095       8
##    mtSum codGenes codSum     mtPer   codPer sumMtPer sumCodPer filterPassed
## c1    66      914   1661 0.6302521 96.00840 3.407331  85.75116            1
## c2   137     1019   1903 0.6572770 95.68075 6.241458  86.69704            1
## c3   250     2134   6031 0.3146067 95.91011 3.704801  89.37463            1
## c4    26      885   1572 0.5434783 96.19565 1.484866  89.77727            1
## c5    65     1014   1839 0.6603774 95.66038 3.134041  88.66924            1
## c6   156     1743   4966 0.4326663 94.26717 2.843602  90.52133            1
##    sizeFactor
## c1  0.3911648
## c2  0.4727105
## c3  1.3498262
## c4  0.3987499
## c5  0.4434534
## c6  1.0297050

# visualize beginning of the rowdata containing gene information
rowdataSCE <- as.data.frame(SummarizedExperiment:::rowData(SCEobject))
head(rowdataSCE)
##       nameInCountMatrix            ENSEMBL SYMBOL
## Gnai3             Gnai3 ENSMUSG00000000001  Gnai3
## Cdc45             Cdc45 ENSMUSG00000000028  Cdc45
## H19                 H19 ENSMUSG00000000031    H19
## Narf               Narf ENSMUSG00000000056   Narf
## Cav2               Cav2 ENSMUSG00000000058   Cav2
## Klf6               Klf6 ENSMUSG00000000078   Klf6
##                                                                GENENAME
## Gnai3 guanine nucleotide binding protein (G protein), alpha inhibiting 3
## Cdc45                                                 cell division cycle 45
## H19                       H19, imprinted maternally expressed transcript
## Narf                               nuclear prelamin A recognition factor
## Cav2                                                          caveolin 2
## Klf6                                                Kruppel-like factor 6
##       chromosome_name   gene_biotype      go_id    name_1006
## Gnai3               3 protein_coding       <NA>         <NA>
## Cdc45              16 protein_coding       <NA>         <NA>
## H19                 7        lincRNA       <NA>         <NA>
## Narf               11 protein_coding       <NA>         <NA>
## Cav2                6 protein_coding GO:0009986 cell surface
## Klf6               13 protein_coding       <NA>         <NA>
```

5.2 Generation of t-SNE coordinates

runCONCLUS creates needed output folders (if you did not run *testClustering* beforehand). Then it generates an object of fourteen (by default) tables with tSNE coordinates. Fourteen because it will vary seven values of principal components *PCs=c(4, 6, 8, 10, 20, 40, 50)* and two values of perplexity *perplexities=c(30, 40)* in all possible combinations.

The chosen values of PCs and perplexities can be changed if necessary. We found that this combination works well for sc-RNA-seq datasets with 400-2000 cells. If you have 4000-9000 cells and expect more than 15 clusters, we recommend to use more first PCs and higher perplexity, for example, *PCs=c(8, 10, 20, 40, 50, 80, 100)* and *perplexities=c(200, 240)*. For details about perplexities parameter see '?Rtsne'.

```
scr <- generateTSNECoordinates(scr)
## Running TSNEs using 2 cores.
## Calculated 14 2D-tSNE plots.
## Building TSNEs objects.
```

Results can be explored as follows:

```
tsneList <- getTSNEList(scr)
head(getCoordinates(tsneList[[1]]))
##            X        Y
## c1 11.252215 6.335510
## c2 11.176064 1.368092
## c3 -7.470344 6.605219
## c4 12.913055 5.277560
## c5 10.147703 3.187482
## c6 -8.325678 7.652778
```

5.3 Clustering with DB-SCAN

Following the calculation of t-SNE coordinates, DBSCAN is run with a range of epsilon and MinPoints values which will yield a total of 84 clustering solutions (PCs x perplexities x MinPoints x epsilon). *minPoints* is the minimum cluster size which you assume to be meaningful for your experiment and *epsilon* is the radius around the cell where the algorithm will try to find *minPoints* dots. Optimal *epsilon* must lay on the knee of the k-NN function as shown in the "test_clustering/distance_graph.pdf" (See Test clustering section above).

```
scr <- runDBSCAN(scr)
```

Results can be explored as follows:

```
dbscanList <- getDbscanList(scr)
clusteringList <- lapply(dbscanList, getClustering)
clusteringList[[1]][,1:10]
##  c1  c2  c3  c4  c5  c6  c9 c10 c11 c12
##   1   1   2   1   1   2   3   1   1   2
```

5.4 Cell and cluster similarity matrix calculation

The above calculated results are combined together in a matrix called "cell similarity matrix". *runDBSCAN* function returns an object of class scRNASeq with its dbscanList slot updated. The list represents 84 clustering solutions (which is equal to number of PCs x perplexities x MinPoints x epsilon). Since the range of cluster varies from result to result, there is no exact match between numbers in different elements of the list. Cells having the same number within an element are guaranteed to be in one cluster. We can calculate how many times out of 84 clustering solutions, every two cells were in one cluster and that is how we come to the similarity matrix of cells. We want to underline that a zero in the dbscan results means that a cell was not assigned to any cluster. Hence, cluster numbers start from one. *clusterCellsInternal* is a general method that returns an object of class scRNASeq with its cellsSimilarityMatrix slot updated.

```
scr <- clusterCellsInternal(scr, clusterNumber = 10)
## Calculating cells similarity matrix.
## Assigning cells to 10 clusters.
## Cells distribution by clusters:
##  1  2  3  4  5  6  7  8  9 10
## 92 81 52 18 51 14 36 68 53 71
```

```
cci <- getCellsSimilarityMatrix(scr)
cci[1:10,1:10]
##            c1        c2        c3        c4        c5        c6        c9
## c1  1.0000000 0.9761905 0.1904762 0.9047619 0.9761905 0.1666667 0.1666667
## c2  0.9761905 1.0000000 0.2142857 0.9285714 1.0000000 0.1785714 0.1785714
## c3  0.1904762 0.2142857 1.0000000 0.1428571 0.2142857 0.9642857 0.7976190
## c4  0.9047619 0.9285714 0.1428571 0.9642857 0.9285714 0.1190476 0.1190476
## c5  0.9761905 1.0000000 0.2142857 0.9285714 1.0000000 0.1785714 0.1785714
## c6  0.1666667 0.1785714 0.9642857 0.1190476 0.1785714 1.0000000 0.8095238
## c9  0.1666667 0.1785714 0.7976190 0.1190476 0.1785714 0.8095238 1.0000000
## c10 0.9047619 0.9285714 0.1428571 0.9642857 0.9285714 0.1190476 0.1190476
## c11 0.9761905 1.0000000 0.2142857 0.9285714 1.0000000 0.1785714 0.1785714
## c12 0.1904762 0.2142857 1.0000000 0.1428571 0.2142857 0.9642857 0.7976190
##           c10       c11       c12
## c1  0.9047619 0.9761905 0.1904762
## c2  0.9285714 1.0000000 0.2142857
## c3  0.1428571 0.2142857 1.0000000
## c4  0.9642857 0.9285714 0.1428571
## c5  0.9285714 1.0000000 0.2142857
## c6  0.1190476 0.1785714 0.9642857
## c9  0.1190476 0.1785714 0.7976190
## c10 0.9642857 0.9285714 0.1428571
## c11 0.9285714 1.0000000 0.2142857
## c12 0.1428571 0.2142857 1.0000000
```

After looking at the similarity between elements on the single-cell level, which is useful if we want to understand if there is any substructure which we did not highlight with our clustering, a "bulk" level where we pool all cells from a cluster into a representative "pseudo cell" can also be generated. This gives a *clusterSimilarityMatrix*:

```
scr <- calculateClustersSimilarity(scr)
csm <- getClustersSimilarityMatrix(scr)
csm[1:10,1:10]
##            1         2         3         4         5         6         7
## 1  0.9910714 0.1785714 0.3690476 0.2619048 0.3452381 0.5476190 0.4285714
## 2  0.1785714 0.9404762 0.1785714 0.1547619 0.1785714 0.3928571 0.1785714
## 3  0.3690476 0.1785714 1.0000000 0.2500000 0.3333333 0.2023810 0.3273810
## 4  0.2619048 0.1547619 0.2500000 0.3244048 0.3333333 0.1666667 0.2440476
## 5  0.3452381 0.1785714 0.3333333 0.3333333 1.0000000 0.2142857 0.2976190
## 6  0.5476190 0.3928571 0.2023810 0.1666667 0.2142857 1.0000000 0.2678571
## 7  0.4285714 0.1785714 0.3273810 0.2440476 0.2976190 0.2678571 0.8898810
## 8  0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
## 9  0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
## 10 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
##            8         9        10
## 1  0.0000000 0.0000000 0.0000000
## 2  0.0000000 0.0000000 0.0000000
## 3  0.0000000 0.0000000 0.0000000
## 4  0.0000000 0.0000000 0.0000000
## 5  0.0000000 0.0000000 0.0000000
## 6  0.0000000 0.0000000 0.0000000
## 7  0.0000000 0.0000000 0.0000000
## 8  1.0000000 0.5833333 0.2976190
## 9  0.5833333 1.0000000 0.3095238
## 10 0.2976190 0.3095238 1.0000000
```

5.5 Plotting

5.5.1 t-SNE colored by clusters or conditions

CONCLUS generated 14 tSNE combining different values of PCs and perplexities. Each tSNE can be visualized either using coloring reflecting the results of DBScan clustering, the conditions or without colors. Here *plotClusteredTSNE* is used to generate all these possibilities of visualization.

```
tSNEclusters <- plotClusteredTSNE(scr, columnName="clusters",
                returnPlot=TRUE, silentPlot=TRUE)

tSNEnoColor <- plotClusteredTSNE(scr, columnName="noColor",
                returnPlot=TRUE, silentPlot=TRUE)

tSNEstate <- plotClusteredTSNE(scr, columnName="state",
                returnPlot=TRUE, silentPlot=TRUE)
```

For visualizing the 5th (out of 14) tSNE cluster:

```
tSNEclusters[[5]]
```

For visualizing the 5th (out of 14) tSNE cluster without colors:

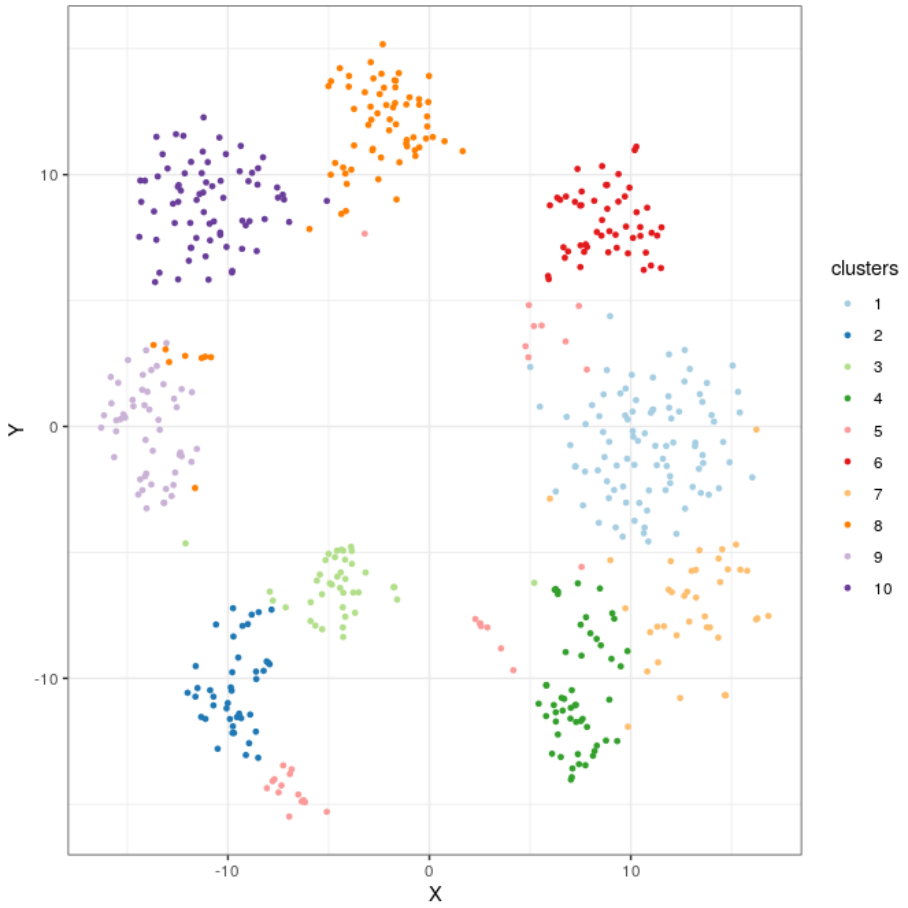

Figure 4: tSNEclusters DBscan results on the 5th tSNE

```
tSNEnoColor[[5]]
```

For visualizing the 5th (out of 14) tSNE cluster colored by state:

```
tSNEstate[[5]]
```

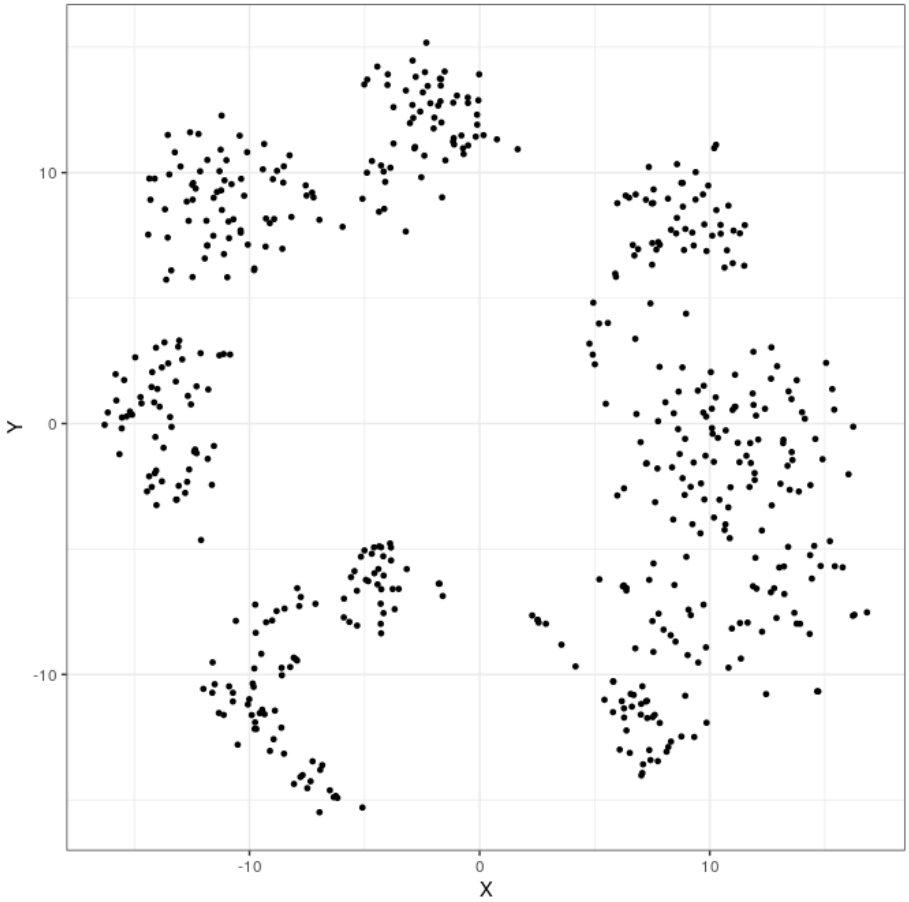

Figure 5: tSNEnoColor The 5th tSNE solution without coloring

5.5.2 Cell similarity heatmap

The *cellsSimilarityMatrix* is then used to generate a heatmap summarizing the results of the clustering and to show how stable the cell clusters are across the 84 solutions.

```
plotCellSimilarity(scr)
```

CellsSimilarityMatrix is symmetrical and its size proportional of to the "number of cells x number of cells". Each vertical or horizontal tiny strip is a cell. Intersection shows the proportion of clustering iterations in which a pair of cells was in one cluster (score between 0 and 1, between blue and red). We will call this combination "consensus clusters" and use them everywhere later. We can appreciate that cellsSimilarityMatrix is the first evidence showing that CONCLUS managed not only to distinguish i8TFs_plus cells from the three other groups (as in the original publication) but also find sub-populations within these groups which were impossible using PCA alone.

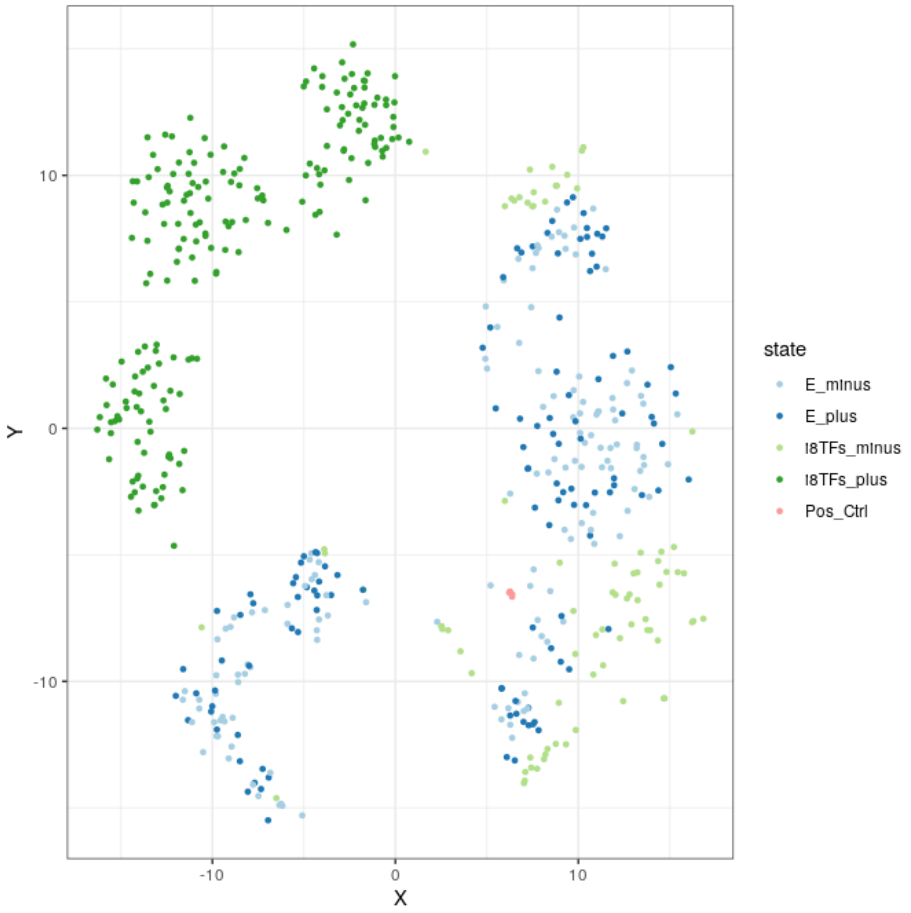

Figure 6: tSNEstate The 5th tSNE solution colored by cell condition

5.5.3 Cluster similarity heatmap

```
plotClustersSimilarity(scr)
```

In the *clusterSimilarityMatrix*, we can still see two major families of clusters: clusters with 8, 9, and 10 on one side and 1, 2, 3, 4, 5, 6 and 7, on the other. Almost all clusters have a high value of similarity across all clustering solutions. Only cluster 5 has a quite low similarity value. Red color on the diagonal means that the group is homogenous, and usually, it is what we want to get. The yellow on the diagonal indicates that either that group consists of two or more equal sized subgroups. Bluish color points to a cluster of dbscan "outliers" that usually surrounds dense clouds of cells in t-SNE plots.

5.6 Marker genes identification

To understand the nature of the consensus clusters identified by CONCLUS, it is essential to identify genes which could be classified as marker genes for each cluster. To this aim, each gene should be "associated" to a particular cluster. This association is performed by looking

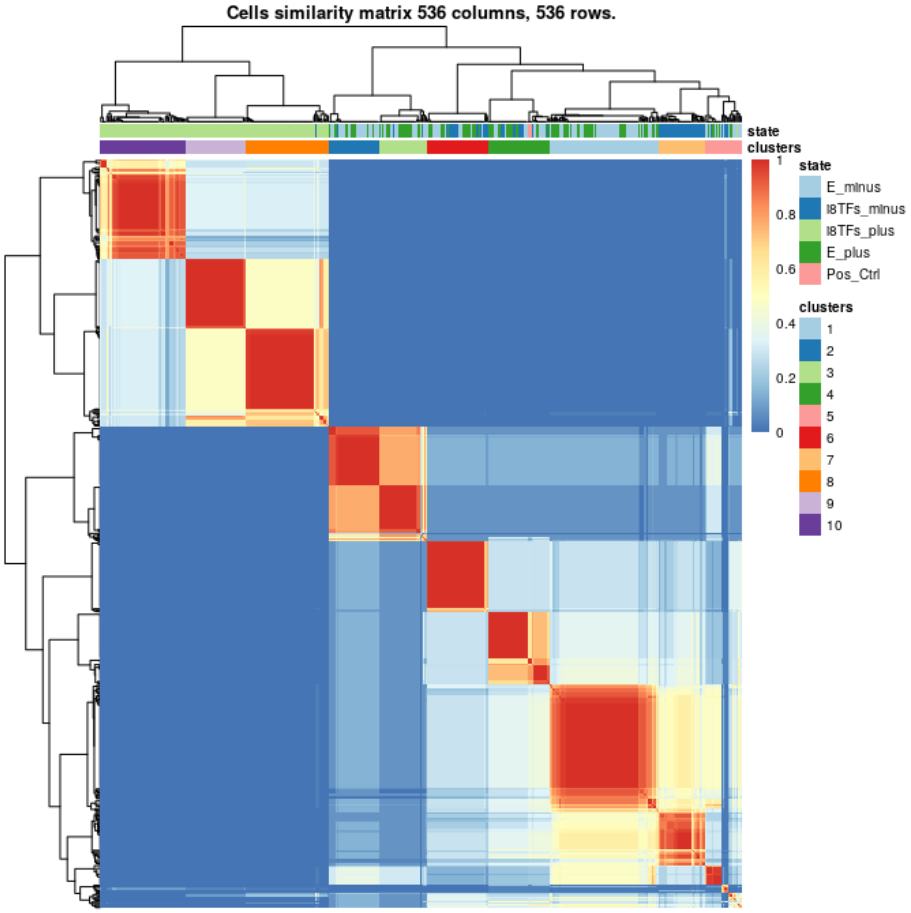

Figure 7: **cellSim Cell similarity matrix showing the conservation of clustering across the 84 solutions**

at up-regulated genes in a particular cluster compared to the others (multiple comparisons). The method *rankGenes* performs multiple comparisons of all genes from the object and rank them according to a score reflecting a FDR power.

In summary, the method *conclus::rankGenes()* gives a list of marker genes for each cluster, ordered by their significance. See ?rankGenes for more details.

```
scr <- rankGenes(scr)
## Ranking marker genes for each cluster.
## Working on cluster 1
## Working on cluster 2
## Working on cluster 3
## Working on cluster 4
## Working on cluster 5
## Working on cluster 6
## Working on cluster 7
## Working on cluster 8
## Working on cluster 9
## Working on cluster 10
```

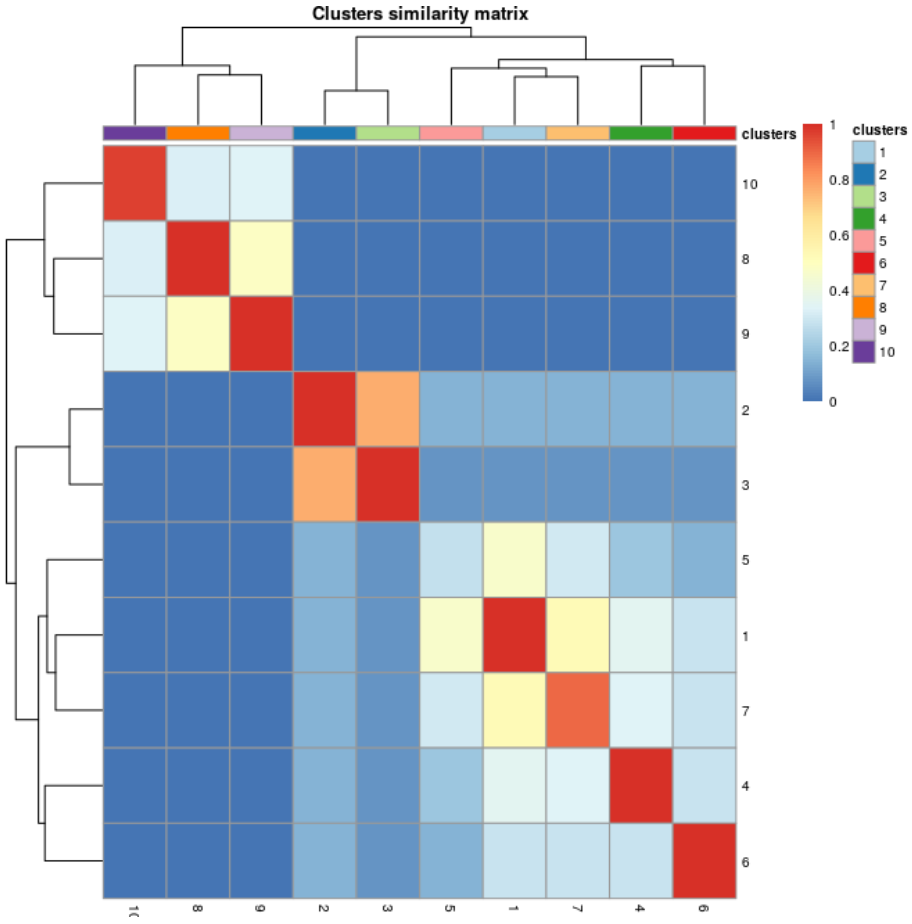

Figure 8: **clustersim Cluster similarity matrix**
Each cell belonging to a particular cluster were merged into a pseudo-cell.

```
markers <- getMarkerGenesList(scr)
head(markers[[1]])
##     Gene         vs_2         vs_3 vs_4     vs_5      vs_6         vs_7
## 1 Tuba1a 1.882589e-11 2.301748e-03    1 1.000000 1.0000000 1.901868e-04
## 2 Tmsb4x 1.000000e+00 4.608840e-09    1 0.611453 1.0000000 1.514166e-06
## 3  Acta2 7.078111e-10 3.111231e-03    1 1.000000 1.0000000 2.242615e-01
## 4  Tagln 2.897984e-12 1.118500e-03    1 1.000000 0.1255327 2.185667e-01
## 5   Actb 1.000000e+00 3.664535e-10    1 1.000000 1.0000000 3.394225e-04
## 6 S100a6 7.175322e-06 2.301748e-03    1 1.000000 1.0000000 1.598867e-03
##          vs_8         vs_9        vs_10 mean_log10_fdr n_05     score
## 1 2.685021e-02 5.514483e-16 3.674073e-13      -5.149823    6 0.7556451
## 2 3.454843e-01 1.000000e+00 4.170926e-01      -1.690135    2 0.7116732
## 3 1.501333e-12 2.367297e-24 3.687863e-23      -7.798766    5 0.7052070
## 4 1.097417e-01 8.552204e-12 1.717314e-06      -3.760403    4 0.6242543
## 5 1.000000e+00 1.000000e+00 1.000000e+00      -1.433916    2 0.6238238
## 6 7.608642e-11 4.201815e-13 2.847982e-15      -5.291001    6 0.6079381
```

The top 10 markers by cluster (default) can be selected with:

```
scr <- retrieveTopClustersMarkers(scr, removeDuplicates=FALSE)
topMarkers <- getClustersMarkers(scr)
topMarkers
##                geneName clusters
## 1                Tuba1a        1
## 2                Tmsb4x        1
## 3                 Acta2        1
## 4                 Tagln        1
## 5                  Actb        1
## 6                S100a6        1
## 7                 Cryab        1
## 8                Lgals1        1
## 9                  Igf2        1
## 10                 Tpm1        1
## 11                 Rps9        2
## 12               Rpl37a        2
## 13                Rps4x        2
## 14               Rps27a        2
## 15                Rps23        2
## 16                Rplp1        2
## 17                Rpl14        2
## 18                Rpl41        2
## 19                  Emb        2
## 20               H3f3a        2
## 21                Gypa        3
## 22               Alas2        3
## 23               Smim1        3
## 24                Car2        3
## 25               Blvrb        3
## 26              Slc4a1        3
## 27  ENSMUSG00000085700        3
## 28               Mgst3        3
## 29               Spta1        3
## 30             Tmem14c        3
## 31                 Gp9        4
## 32               Icam2        4
## 33               Gp1bb        4
## 34              Ctla2a        4
## 35              Tmem40        4
## 36              Adgrl4        4
## 37               Isg20        4
## 38                  F8        4
## 39               Ramp2        4
## 40              Gimap4        4
## 41              Col1a1        5
## 42                 Nog        5
## 43               Postn        5
## 44                 Fn1        5
## 45  ENSMUSG00000097451        5
## 46              Col1a2        5
## 47              Col5a1        5
```

```
## 48               Mfge8        5
## 49                Nid2        5
## 50               Lama1        5
## 51               Cd180        6
## 52               C3ar1        6
## 53                Rin2        6
## 54               Sirpa        6
## 55               Csf1r        6
## 56               Sorl1        6
## 57               Mpeg1        6
## 58             Clec4a2        6
## 59                Ccl4        6
## 60               Itgam        6
## 61                 H19        7
## 62            Slc25a21        7
## 63                 Kel        7
## 64                Snca        7
## 65  ENSMUSG00000098178        7
## 66               Arl4c        7
## 67  ENSMUSG00000076258        7
## 68              Jarid2        7
## 69              Malat1        7
## 70  ENSMUSG00000097625        7
## 71                 Ptn        8
## 72                Mest        8
## 73            Serpinf1        8
## 74              Lgals1        8
## 75  ENSMUSG00000086567        8
## 76                Tpm1        8
## 77                 Ak1        8
## 78              Tmsb10        8
## 79                Ppic        8
## 80               Gngt2        8
## 81             Alox5ap        9
## 82              Ifitm1        9
## 83                Ccl9        9
## 84                Cd52        9
## 85              Fcer1g        9
## 86               Tuba8        9
## 87              Ifitm2        9
## 88             Pstpip1        9
## 89               Fxyd5        9
## 90               Alox5        9
## 91              Cstdc5       10
## 92              S100a8       10
## 93              Malat1       10
## 94              Ifitm1       10
## 95                Apoe       10
## 96                Cd52       10
## 97              Mamdc2       10
## 98                CYTB       10
```

```
## 99               Gng11       10
## 100               Mest       10
```

6 Plot a heatmap with positive marker genes

Following the execution of the `retrieveTopClustersMarkers` method, CONCLUS offers the option to visualize the marker genes on a heatmap. Below we chose to show the selected 10 marker genes per cluster which should generate a heatmap with 100 genes (10 marker genes x 10 clusters). This is convenient for visualization. In practice, the number of genes in this heatmap will be less than 100 because some genes were classified as markers for more than one cluster. This can happen when several clusters correspond to similar cellular types.

After selecting the top markers with the method `retrieveTopClustersMarkers`, the method *plotCellHeatmap* is used to order clusters and genes by similarity (the same order as in the *clusterSimilarityMatrix*) and show mean-centered normalized data. Mean-centering allows seeing the relative expression of a gene compared to the mean.

```
plotCellHeatmap(scr, orderClusters=TRUE, orderGenes=TRUE)
```

The second heatmap below also shows the order of genes and clusters by similarity but for normalized expression data. As you can see, genes expressed at a level of seven and nine look very similar. It is hard to highlight all the differences in expression of both lowly and highly detected genes in one heatmap using normalized data. For this reason, mean-centering helps to solve this issue.

```
plotCellHeatmap(scr, orderClusters=TRUE, orderGenes=TRUE, meanCentered=FALSE)
```

Alternative order of clusters is by name or by hierarchical clustering as in the default pheatmap function.

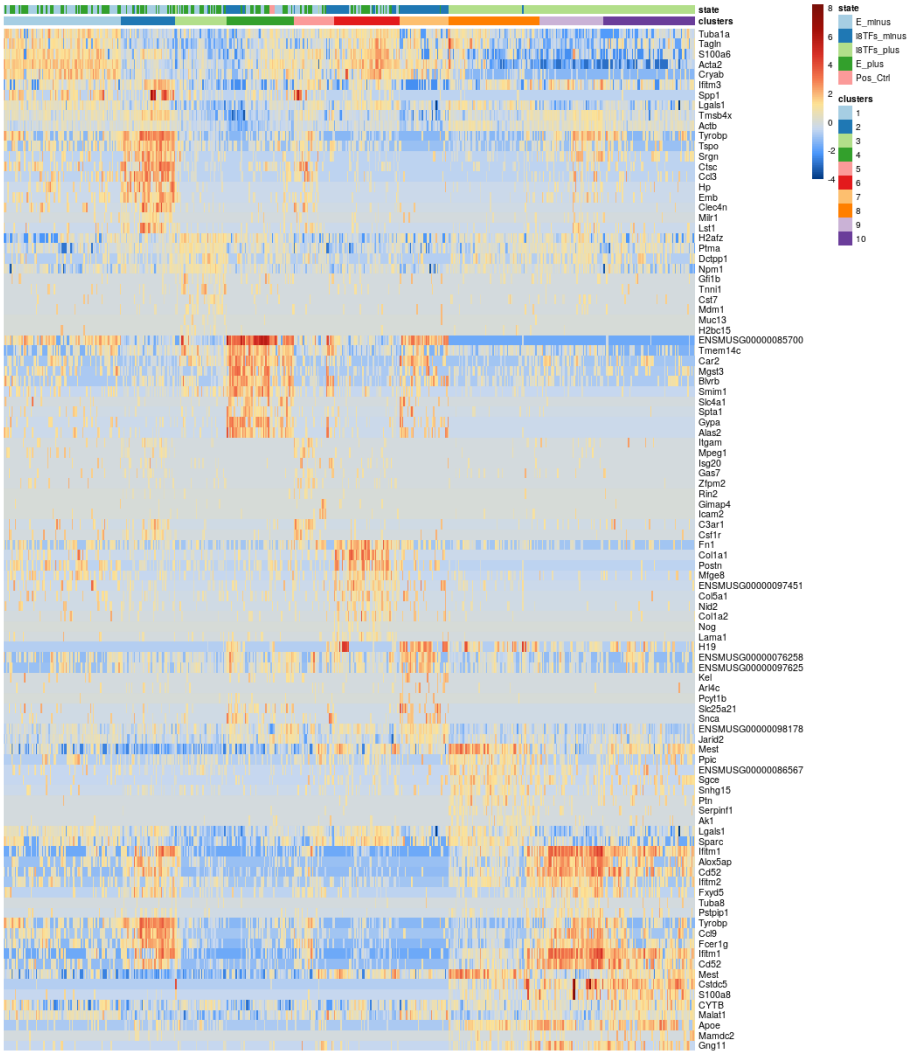

Figure 9: cellHeatmap Heatmap showing the expression of the top 10 markers for each cluster
The values are normalized according to the mean.

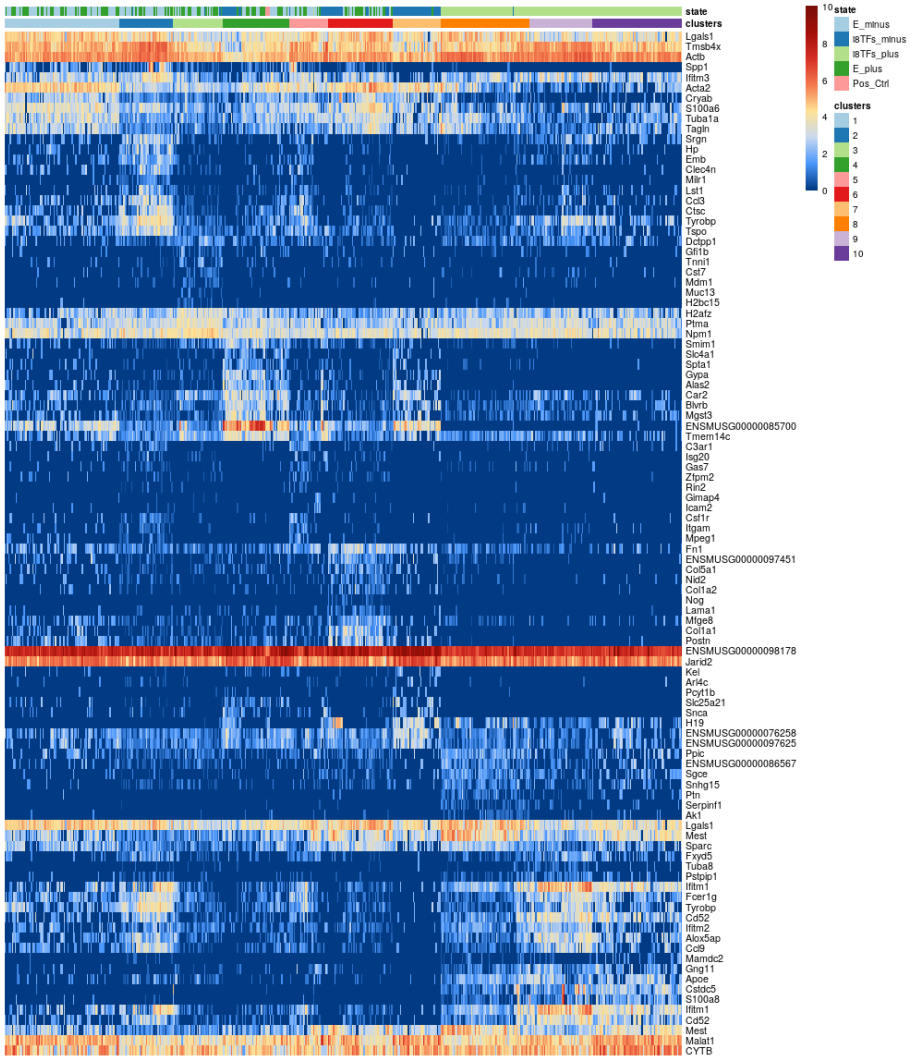

Figure 10: cellHeatmapNoMean Same heatmap as before without normalizing by the mean

7 Plot t-SNE colored by expression of a selected gene

PlotGeneExpression allows visualizing the normalized expression of one gene in a t-SNE plot. It can be useful to inspect the specificity of top markers. Below are examples of marker genes that define a particular cluster.

```
# Plot gene expression in a selected tSNE plot
plotGeneExpression(scr, "Hp", tSNEpicture=5)
```

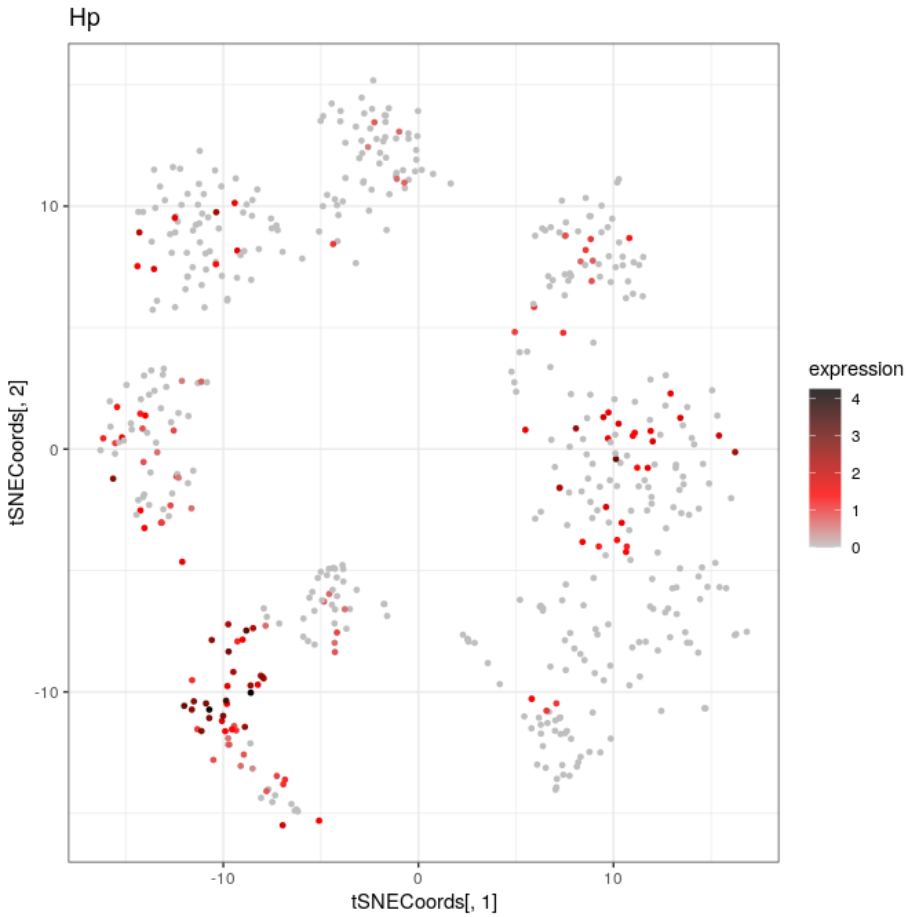

Figure 11: Hp Marker gene of cluster 2

```
plotGeneExpression(scr, "Alox5ap", tSNEpicture=5)
```

```
plotGeneExpression(scr, "Cd52", tSNEpicture=5)
```

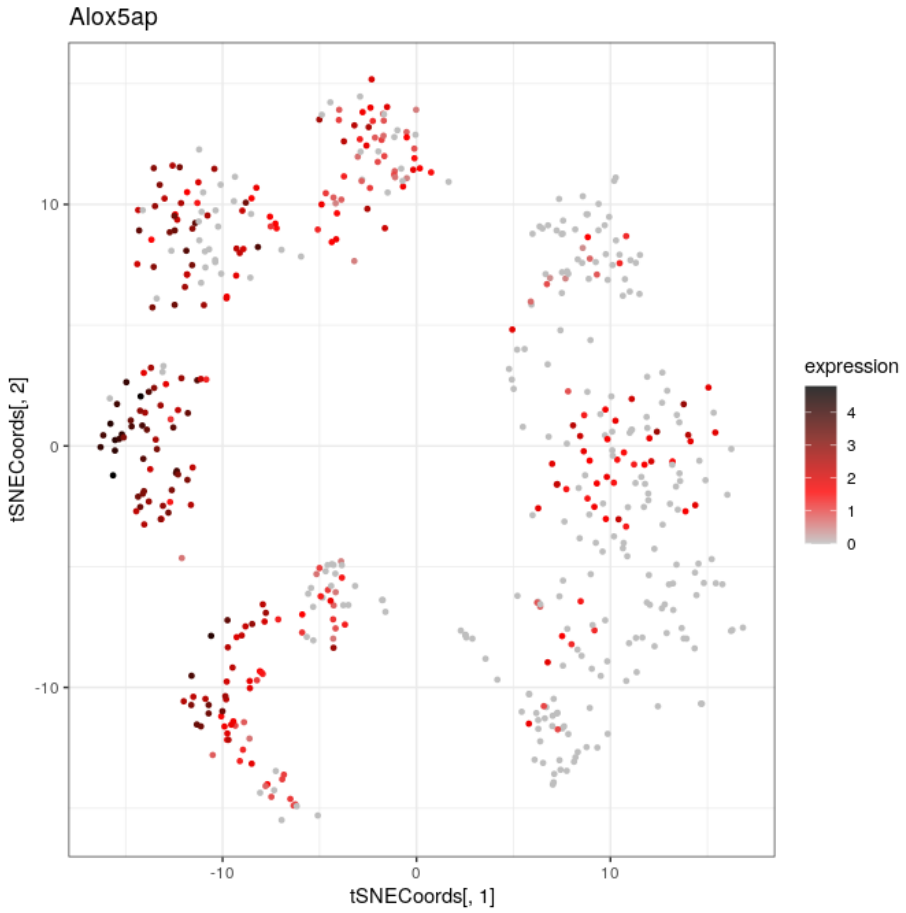

Figure 12: Alox5ap Marker gene of cluster 9

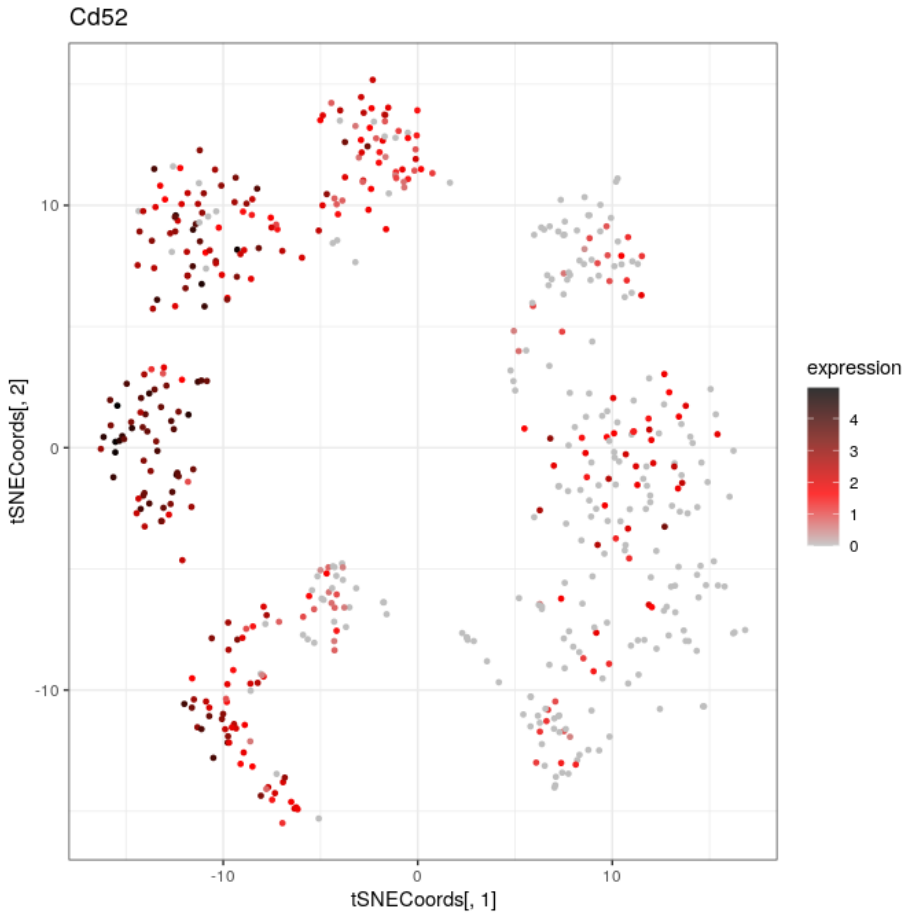

Figure 13: CD52 Marker gene of cluster 9

8 Collect publicly available info about marker genes

8.1 Collect information for the top 10 markers for each cluster

`retrieveGenesInfo` retrieves gene information from NCBI, MGI, and UniProt. It requires the `retrieveTopMarkers` method to have been run on the object.

```
scr <- retrieveGenesInfo(scr)
## # Attempt 1/5 # Connection to Ensembl ...
## Connected with success.
## # Attempt 1/5 # Retrieving information about genes from biomaRt ...
## Information retrieved with success.
## # Attempt 1/5 # Retrieving information about genes from biomaRt ...
## Information retrieved with success.
result <- getGenesInfos(scr)
head(result)
##   uniprot_gn_symbol clusters external_gene_name
## 1            Tuba1a        1             Tuba1a
## 2            Tmsb4x        1             Tmsb4x
## 3             Acta2        1              Acta2
## 4             Tagln        1              Tagln
## 5              Actb        1               Actb
## 6            S100a6        1             S100a6
##
## 1
## 2
## 3
## 4
## 5 GO:0005737, GO:0005634, GO:0032991, GO:0005829, GO:0005515, GO:0000166, GO:0005524, GO:0042802, GO:0016
## 6
##                               mgi_description
## 1                            tubulin, alpha 1A
## 2              thymosin, beta 4, X chromosome
## 3        actin, alpha 2, smooth muscle, aorta
## 4                                  transgelin
## 5                                actin, beta
## 6 S100 calcium binding protein A6 (calcyclin)
##                        entrezgene_description   gene_biotype chromosome_name
## 1                           tubulin, alpha 1A protein_coding              15
## 2              thymosin, beta 4, X chromosome protein_coding               X
## 3        actin, alpha 2, smooth muscle, aorta protein_coding              19
## 4                                  transgelin protein_coding               9
## 5                                actin, beta protein_coding               5
## 6 S100 calcium binding protein A6 (calcyclin) protein_coding               3
##   Symbol    ensembl_gene_id      mgi_id entrezgene_id
## 1 Tuba1a ENSMUSG00000072235   MGI:98869         22142
## 2 Tmsb4x ENSMUSG00000049775   MGI:99510         19241
## 3  Acta2 ENSMUSG00000035783   MGI:87909         11475
## 4  Tagln ENSMUSG00000032085  MGI:106012         21345
## 5   Actb ENSMUSG00000029580   MGI:87904         11461
```

```
## 6 S100a6 ENSMUSG00000001025 MGI:1339467        20200
##                                    uniprot_gn_id
## 1                                          P68369
## 2                                  P20065, Q6ZWX2
## 3                               P62737, A0A494B9T3
## 4                               P37804, A0A1L1STN8
## 5 P60710, B2RRX1, E9Q1F2, E9Q606, E9Q5F4, E9Q2D1
## 6                                  P14069, Q545I9
```

`result` contains the following columns:

- uniprot_gn_symbol: Uniprot gene symbol.
- clusters: The cluster to which the gene is associated.
- external_gene_name: The complete gene name.
- go_id: Gene Ontology (GO) identification number.
- mgi_description: If the species is mouse, description of the gene on MGI.
- entrezgene_description: Description of the gene by the Entrez database.
- gene_biotype: protein coding gene, lincRNA gene, miRNA gene, unclassified non-coding RNA gene, or pseudogene.
- chromosome_name: The chromosome on which the gene is located.
- Symbol: Official gene symbol.
- ensembl_gene_id: ID of the gene in the ensembl database.
- mgi_id: If the species is mouse, ID of the gene on the MGI database.
- entrezgene_id: ID of the gene on the entrez database.
- uniprot_gn_id: ID of the gene on the uniprot database.

9 Supervised clustering

Until now, we have been using CONCLUS in an unsupervised fashion. This is a good way to start the analysis of a sc-RNA-seq dataset. However, the knowledge of the biologist remains a crucial asset to get the maximum of the data. This is why we have included in CONCLUS, additional options to do supervised analysis (or "manual" clustering) to allow the researcher to use her/his biological knowledge in the CONCLUS workflow. Going back to the example of the Bergiers et al. dataset above (cluster similarity heatmap), one can see that some clusters clearly belong to the same family of cells after examining the clusters_similarity matrix generated by CONCLUS.

It is mostly obvious for clusters 2 and 3. In order to figure out what marker genes are defining these families of clusters, one can use manual clustering in CONCLUS to fuse clusters of similar nature: i.e. combine clusters 2 and 3 together.

```
## Retrieving the table indicating to which cluster each cell belongs
clustCellsDf <- retrieveTableClustersCells(scr)

## Replace "3" by "2" to merge 2/3
clustCellsDf$clusters[which(clustCellsDf$clusters == 3)] <- 2

## Modifying the object to take into account the new classification
scrUpdated <- addClustering(scr, clusToAdd=clustCellsDf)
## Computing new markers..
```

```
## Calculating cells similarity matrix.
## Assigning cells to 9 clusters.
## Cells distribution by clusters:
##  1  2  3  4  5  6  7  8  9
## 92 81 52 32 51 36 68 53 71
## Ranking marker genes for each cluster.
## Working on cluster 1
## Working on cluster 2
## Working on cluster 3
## Working on cluster 4
## Working on cluster 5
## Working on cluster 6
## Working on cluster 7
## Working on cluster 8
## Working on cluster 9
## # Attempt 1/5 # Connection to Ensembl ...
## Connected with success.
## # Attempt 1/5 # Retrieving information about genes from biomaRt ...
## Information retrieved with success.
## # Attempt 1/5 # Retrieving information about genes from biomaRt ...
## Information retrieved with success.
```

Now we can visualize the new results taking into account the new classification:

```
plotClustersSimilarity(scrUpdated)
```

```
plotCellSimilarity(scrUpdated)
```

```
plotCellHeatmap(scrUpdated, orderClusters=TRUE, orderGenes=TRUE)
```

```
tSNEclusters <- plotClusteredTSNE(scrUpdated, columnName="clusters",
                                  returnPlot=TRUE, silentPlot=TRUE)
tSNEclusters[[5]]
```

The cell heatmap above shows that Tyrobp and Ccl9 are good markers of cluster 2 (at the bottom). One can visualize them in the t-SNE plots below.

```
plotGeneExpression(scrUpdated, "Tyrobp", tSNEpicture=5)
```

```
plotGeneExpression(scrUpdated, "Ccl9", tSNEpicture=5)
```

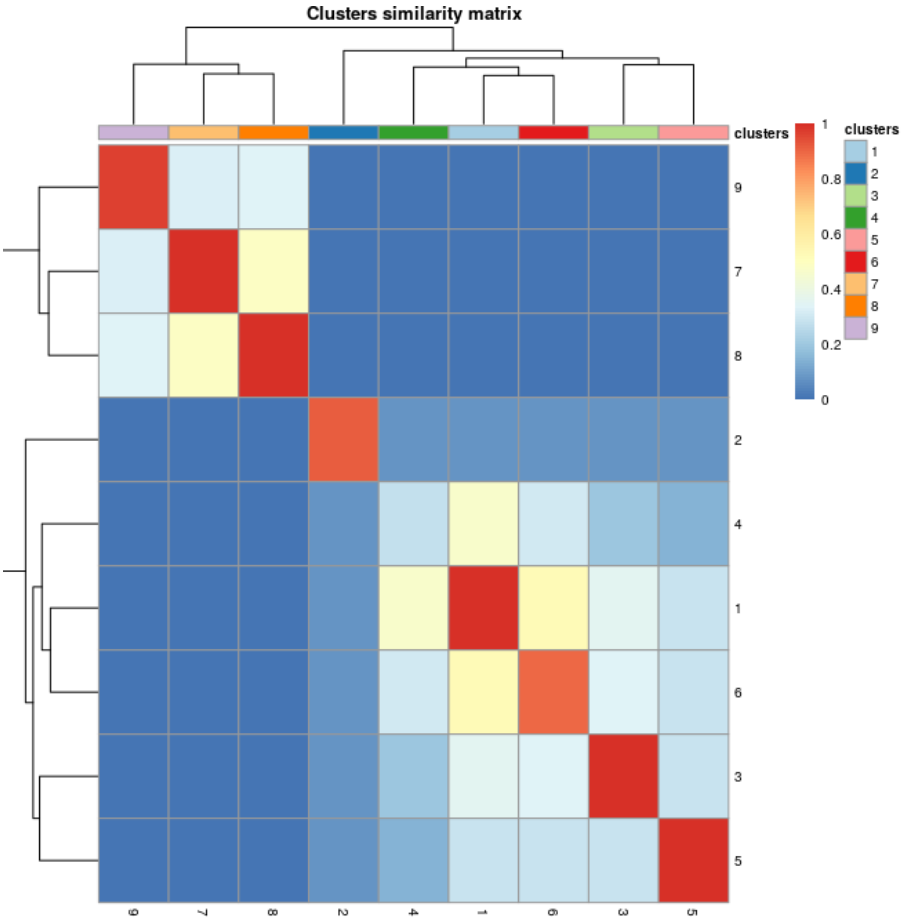

Figure 14: updatedClustSim Updated clusters similarity matrix with cluster 2 representing the merged old clusters 2 and 3

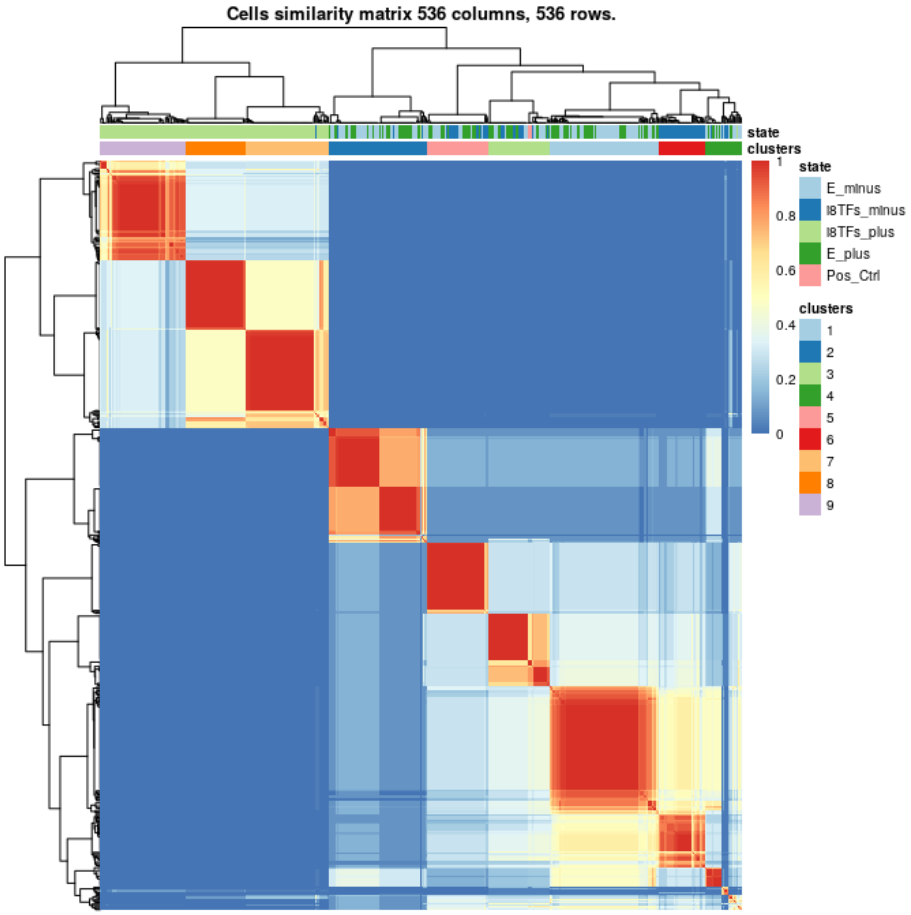

Figure 15: updatedCellSim Updated cells similarity matrix with cluster 2 representing the merged old clusters 2 and 3

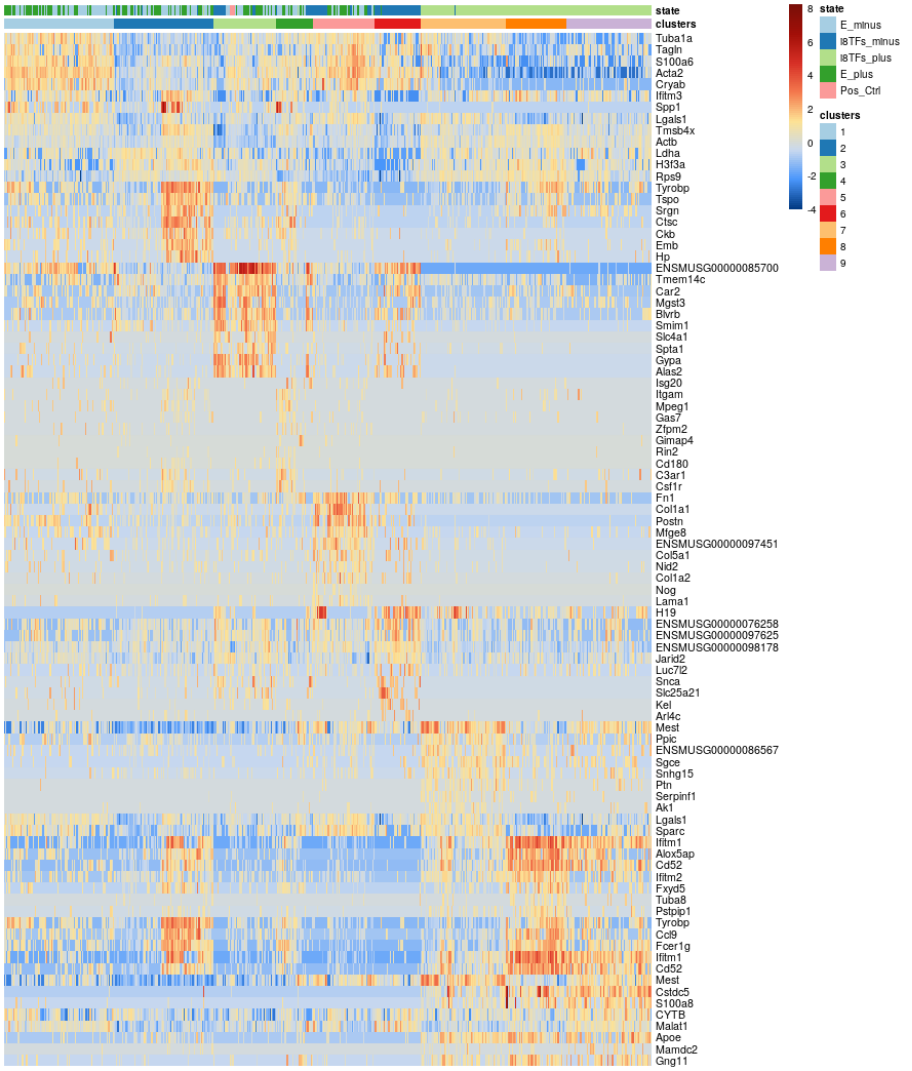

Figure 16: updatedCellHeatmap Updated cells heatmap with cluster 2 representing the merged old clusters 2 and 3

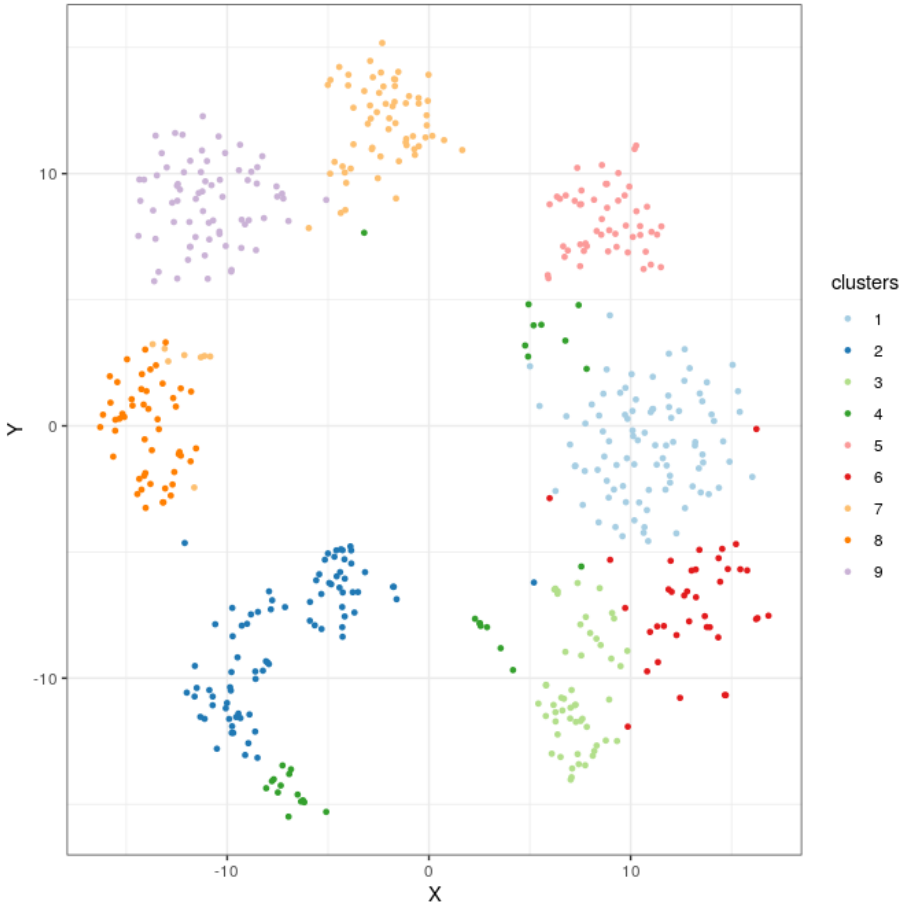

Figure 17: updatedTSNE 5th tSNE solution colored by dbscan result showing the cluster 2 as being the merge of the old clusters 2 and 3

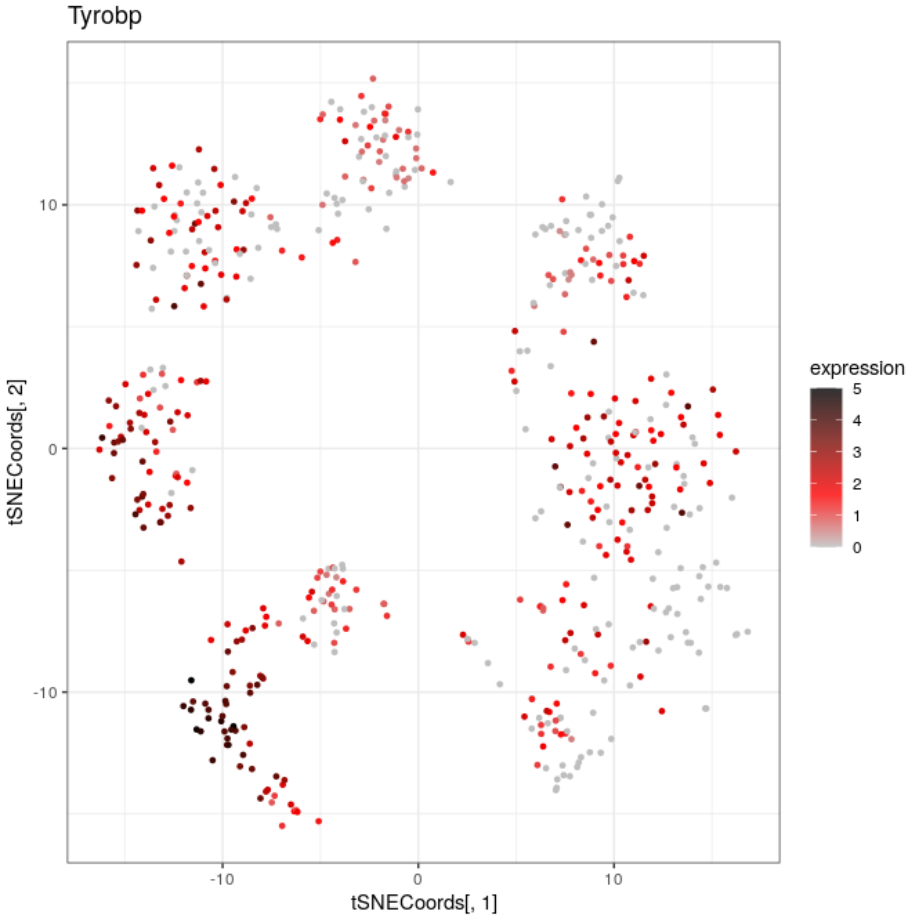

Figure 18: Tyrobp Marker gene for cluster 2 (mix of old clusters 2 and 3)

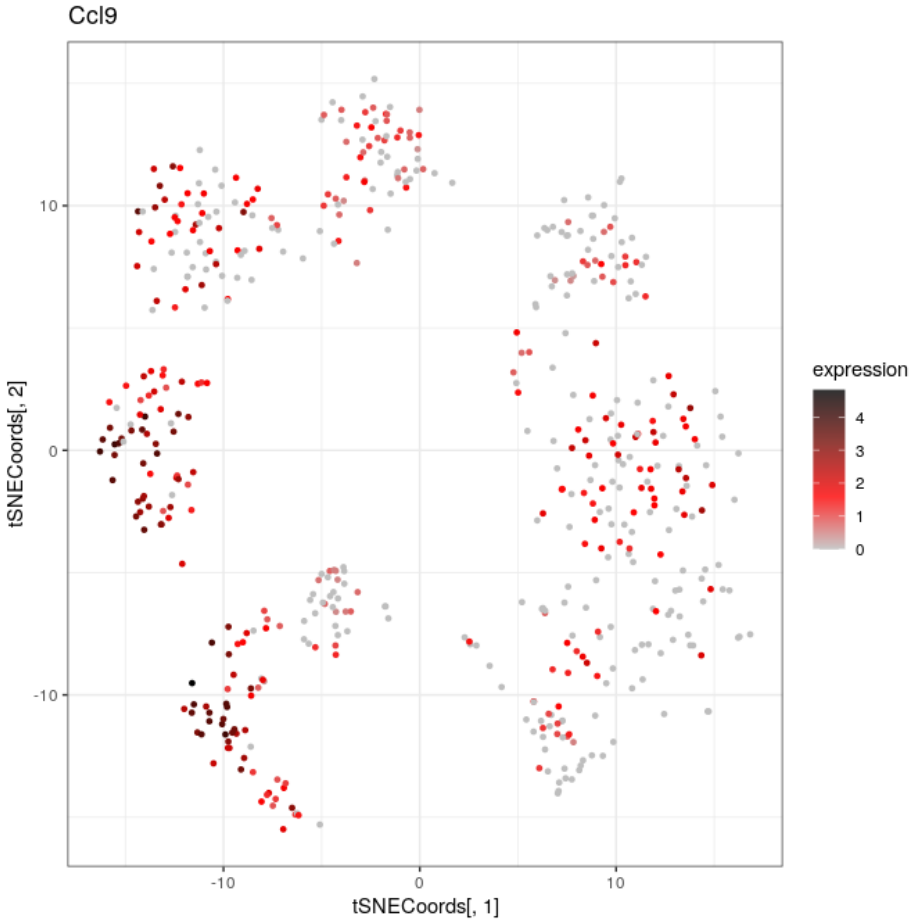

Figure 19: Ccl9 Marker gene for clusters 2 (mix of old clusters 2 and 3)

10 Conclusion

Here we demonstrated how to use CONCLUS and combine multiple parameters testing for sc-RNA-seq analysis. It allowed us to gain more information on the dataset of Bergiers et al and will help gaining deeper insights into others.

Indeed in the original analysis using PCA, two major clusters were found (one composed of i8TFs_plus cells and another comprising E_minus, E_plus, i8TFs_minus cells). Using CONCLUS, we see that there is still a big difference between the i8TFs_plus experimental group and the other three. Interestingly, CONCLUS was able to unveil heterogeneity within the i8TFs group while the previous analysis performed by Bergiers et al was not able to reveal it. This analysis offers additional information on the function of these eight transcription factors.

11 Session info

```
sessionInfo()
## R version 4.0.3 (2020-10-10)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 18.04.4 LTS
##
## Matrix products: default
## BLAS:   /usr/lib/x86_64-linux-gnu/blas/libblas.so.3.7.1
## LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.7.1
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=it_IT.UTF-8        LC_COLLATE=en_US.UTF-8
##  [5] LC_MONETARY=it_IT.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=it_IT.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=it_IT.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] parallel  stats4    stats     graphics  grDevices utils     datasets
## [8] methods   base
##
## other attached packages:
## [1] org.Mm.eg.db_3.11.4  AnnotationDbi_1.50.3 IRanges_2.22.2
## [4] S4Vectors_0.26.1     Biobase_2.48.0       BiocGenerics_0.34.0
## [7] conclus_0.99.0       BiocStyle_2.16.1     rmarkdown_2.5
##
## loaded via a namespace (and not attached):
##   [1] tidyselect_1.1.0           RSQLite_2.2.1
##   [3] grid_4.0.3                 BiocParallel_1.22.0
##   [5] Rtsne_0.15                 scatterpie_0.1.5
##   [7] munsell_0.5.0              codetools_0.2-17
##   [9] statmod_1.4.35             scran_1.16.0
```

```
##  [11] colorspace_1.4-1           GOSemSim_2.14.2
##  [13] knitr_1.30                 rstudioapi_0.11
##  [15] SingleCellExperiment_1.10.1 robustbase_0.93-6
##  [17] ggsignif_0.6.0             DOSE_3.14.0
##  [19] labeling_0.4.2             urltools_1.7.3
##  [21] GenomeInfoDbData_1.2.3     polyclip_1.10-0
##  [23] bit64_4.0.5                farver_2.0.3
##  [25] pheatmap_1.0.12            downloader_0.4
##  [27] vctrs_0.3.4                generics_0.0.2
##  [29] xfun_0.18                  BiocFileCache_1.12.1
##  [31] diptest_0.75-7             R6_2.4.1
##  [33] doParallel_1.0.16          GenomeInfoDb_1.24.2
##  [35] ggbeeswarm_0.6.0           graphlayouts_0.7.1
##  [37] rsvd_1.0.3                 locfit_1.5-9.4
##  [39] flexmix_2.3-17             bitops_1.0-6
##  [41] fgsea_1.14.0               gridGraphics_0.5-0
##  [43] DelayedArray_0.14.1        assertthat_0.2.1
##  [45] scales_1.1.1               ggraph_2.0.3
##  [47] nnet_7.3-14                enrichplot_1.8.1
##  [49] beeswarm_0.2.3             gtable_0.3.0
##  [51] tidygraph_1.2.0            rlang_0.4.8
##  [53] splines_4.0.3              rstatix_0.6.0
##  [55] GEOquery_2.56.0            broom_0.7.2
##  [57] europepmc_0.4              abind_1.4-5
##  [59] BiocManager_1.30.10        yaml_2.2.1
##  [61] reshape2_1.4.4             backports_1.1.10
##  [63] qvalue_2.20.0              clusterProfiler_3.16.1
##  [65] tools_4.0.3                bookdown_0.21
##  [67] ggplotify_0.0.5            ggplot2_3.3.2
##  [69] ellipsis_0.3.1             RColorBrewer_1.1-2
##  [71] ggridges_0.5.2             Rcpp_1.0.5
##  [73] plyr_1.8.6                 progress_1.2.2
##  [75] zlibbioc_1.34.0            purrr_0.3.4
##  [77] RCurl_1.98-1.2             ps_1.4.0
##  [79] prettyunits_1.1.1          ggpubr_0.4.0
##  [81] openssl_1.4.3              dbscan_1.1-5
##  [83] viridis_0.5.1              cowplot_1.1.0
##  [85] haven_2.3.1                SummarizedExperiment_1.18.2
##  [87] ggrepel_0.8.2              cluster_2.1.0
##  [89] factoextra_1.0.7           tinytex_0.26
##  [91] magrittr_1.5               data.table_1.13.2
##  [93] openxlsx_4.2.2             DO.db_2.9
##  [95] triebeard_0.3.0            matrixStats_0.57.0
##  [97] hms_0.5.3                  evaluate_0.14
##  [99] XML_3.99-0.5               rio_0.5.16
## [101] readxl_1.3.1               mclust_5.4.6
## [103] gridExtra_2.3              compiler_4.0.3
## [105] biomaRt_2.44.4             scater_1.16.2
## [107] tibble_3.0.4               crayon_1.3.4
## [109] htmltools_0.5.0            tidyr_1.1.2
## [111] DBI_1.1.0                  tweenr_1.0.1
```

```
## [113] dbplyr_1.4.4               MASS_7.3-53
## [115] fpc_2.2-8                  rappdirs_0.3.1
## [117] car_3.0-10                 Matrix_1.2-18
## [119] readr_1.4.0                cli_2.1.0
## [121] igraph_1.2.6               forcats_0.5.0
## [123] GenomicRanges_1.40.0       pkgconfig_2.0.3
## [125] rvcheck_0.1.8              foreign_0.8-79
## [127] xml2_1.3.2                 foreach_1.5.1
## [129] vipor_0.4.5                dqrng_0.2.1
## [131] XVector_0.28.0             stringr_1.4.0
## [133] digest_0.6.27              cellranger_1.1.0
## [135] fastmatch_1.1-0            edgeR_3.30.3
## [137] DelayedMatrixStats_1.10.1  curl_4.3
## [139] kernlab_0.9-29             modeltools_0.2-23
## [141] lifecycle_0.2.0            jsonlite_1.7.1
## [143] carData_3.0-4              BiocNeighbors_1.6.0
## [145] viridisLite_0.3.0          askpass_1.1
## [147] limma_3.44.3               fansi_0.4.1
## [149] pillar_1.4.6               lattice_0.20-41
## [151] httr_1.4.2                 DEoptimR_1.0-8
## [153] GO.db_3.11.4               glue_1.4.2
## [155] zip_2.1.1                  prabclus_2.3-2
## [157] iterators_1.0.13           bit_4.0.4
## [159] ggforce_0.3.2              class_7.3-17
## [161] stringi_1.5.3              blob_1.2.1
## [163] BiocSingular_1.4.0         memoise_1.1.0
## [165] dplyr_1.0.2                irlba_2.3.3
```