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

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

10 April 2022

Contents

1 Introduction

CONCLUS is a tool for robust clustering and positive marker features selection of single-cell RNA-seq (sc-RNA-seq) datasets. It is designed to identify rare cells populations by consensus clustering. 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 to the Bioconductor forum using the keyword 'conclus' in the post title. To contact us directly write to christophe.lancrin@embl.it or rachedi.ilyess@gmail.com. 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 tSNE, everything was generated with a random seed of 42 (default parameter of `generateTSNECoordinates`) to ensure reproducibility of the results. Because of the evolution of biomaRt, you might get slightly different figures. However, the marker genes for each cluster should be the same.

The package is currently limited to mouse and human. Other organisms can be added on demand. Write to christophe.lancrin@embl.it or rachedi.ilyess@gmail.com.Priority will be given to model organisms.

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 <- tempdir()
experimentName <- "Test"
species <- "mouse"

countmatrixPath <- system.file("extdata/countMatrix.tsv", package="conclus")
countMatrix <- loadDataOrMatrix(file=countmatrixPath, type="countMatrix",
                                ignoreCellNumber=TRUE)

coldataPath <- system.file("extdata/colData.tsv", package="conclus")
columnsMetaData <- loadDataOrMatrix(file=coldataPath, type="coldata",
                        columnID="cell_ID")

sceObjectCONCLUS <- runCONCLUS(outputDirectory, experimentName, countMatrix,
                            species, columnsMetaData=columnsMetaData,
                            perplexities=c(2,3), tSNENb=1,
                            PCs=c(4,5,6,7,8,9,10), epsilon=c(380, 390, 400),
                            minPoints=c(2,3), clusterNumber=4)
```

In your "outputDirectory", the sub-folder `pictures` contains all tSNE with dbscan coloration (sub-folder `tSNE_pictures`), the cell similarity matrix (`Test_cells_correlation_X_clusters.pdf`), the cell heatmap (`Test_clustersX_meanCenteredTRUE_orderClustersFALSE_orderGenesFALSE markersPerCluster.pdf`), and the cluster similarity matrix (`Test_clusters_similarity_10_clusters.pdf`). You will also find in the sub-folder `Results`:

- `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 these 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 Shvartsman *et al*. The Yolk Sac (YS) and Aorta-Gonad-Mesonephros (AGM) are two major haematopoietic regions during embryonic development. Interestingly, AGM is the only one generating haematopoietic stem cells (HSCs). To identify the difference between AGM and YS, Shvartsman *et al* compared them using single cell RNA sequencing between 9.5 and 11.5 days of mouse embryonic development. This vignette aims at reproducing the identification

of a rare population corresponding to liver like cells in the YS at day 11.5. The number of clusters used in this vignette is lower than in the original article for sake of simplification. This vignette neither provides a description of the differences between the AGM and YS. Please refer to Shvartsman *et al* for a complete description.

Since endothelial cells represent a small population in these tissues, cells expressing the endothelial marker VE-Cadherin (VE-Cad, also called Cdh5) were enriched and sorted, VE-Cad Negative cells constituting the microenvironment. Therefore, four cell states are defined by the cell barcodes: YS VE-Cad minus, AGM VE-Cad minus, YS VE-Cad minus, and AGM VE-Cad minus. The E11.5 data are constituted of 1482 cells. After filtering and normalization, 1303 cells were retained.

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.

Shvartsman *et al* deleted highly abundant haemoglobins having names starting with 'Hba' or 'Hbb' because they seemed to be the primary source of contamination. Additionally, they excluded poorly annotated genes that did not have gene symbols to improve the clusters annotation process. Finally, the human cell controls were removed.

4.2.1 Retrieving data from GEO

The code below format the count matrix and the columns meta-data. The source data are downloaded from the GEO page GSE161933. The URL of the count matrix and the cells meta-data were retrieved by right-click and 'copy link adress' on the http hyperlink of the supplementary file at the bottom of the page. If the columns metadata are not available in the supplementary file section, the name of the series matrix, containing columns meta-data can be retrieved by clicking the 'Series Matrix File(s)' link just above the count matrix. The function `retrieveFromGEO` used with the parameter seriesMatrixName will download all series matrices present in the GEO record however only the one of interest will be kept.

```
library(conclus)
##
##
## Setting options('download.file.method.GEOquery'='auto')
## Setting options('GEOquery.inmemory.gpl'=FALSE)
##
## Registered S3 method overwritten by 'ggtree':
##   method      from
##   identify.gg ggfun

outputDirectory <- tempdir()
dir.create(outputDirectory, showWarnings=FALSE)
species <- "mouse"

countMatrixPath <- file.path(outputDirectory, "countmatrix.txt")
metaDataPath <- file.path(outputDirectory, "metaData.txt")
matrixURL <- paste0("https://www.ncbi.nlm.nih.gov/geo/download/?acc=GSM492",
"3493&format=file&file=GSM4923493%5FE11%5F5%5Frawcounts%5Fmatrix%2Etsv%2Egz")
metaDataURL <- paste0("https://www.ncbi.nlm.nih.gov/geo/download/?acc=",
"GSM4923493&format=file&file=GSM4923493%5FMetadata%5FE11%5F5%2Etxt%2Egz")
```

```
result <- retrieveFromGEO(matrixURL, countMatrixPath, species,
metaDataPath=metaDataPath, colMetaDataURL=metaDataURL)
## Warning in .retrieveColMetaDataFromURL(bfc, colMetaDataURL, metaDataPath): The
## columns of the cells meta-data should be: cellName, state, and cellBarcode.
## Please correct the dataframe.
## Formating data.
## Warning in .filteringAndOrdering(countMatrix, columnsMetaData): The cell
## barcodes were not found in the matrix. The columns of the count matrix and the
## rows of the meta-data will not be re-ordered. Are you sure that the count matrix
## and the meta-data correspond?
## Converting ENSEMBL IDs to symbols.
## Loading required package: org.Mm.eg.db
## Loading required package: AnnotationDbi
## Loading required package: stats4
## Loading required package: BiocGenerics
##
## Attaching package: 'BiocGenerics'
## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs
## The following objects are masked from 'package:base':
##
##     Filter, Find, Map, Position, Reduce, anyDuplicated, append,
##     as.data.frame, basename, cbind, colnames, dirname, do.call,
##     duplicated, eval, evalq, get, grep, grepl, intersect, is.unsorted,
##     lapply, mapply, match, mget, order, paste, pmax, pmax.int, pmin,
##     pmin.int, rank, rbind, rownames, sapply, setdiff, sort, table,
##     tapply, union, unique, unsplit, which.max, which.min
## Loading required package: Biobase
## Welcome to Bioconductor
##
##     Vignettes contain introductory material; view with
##     'browseVignettes()'. To cite Bioconductor, see
##     'citation("Biobase")', and for packages 'citation("pkgname")'.
## Loading required package: IRanges
## Loading required package: S4Vectors
##
## Attaching package: 'S4Vectors'
## The following objects are masked from 'package:base':
##
##     I, expand.grid, unname
##
## Attaching package: 'IRanges'
## The following object is masked from 'package:grDevices':
##
##     windows
## 'select()' returned 1:many mapping between keys and columns
## Warning in clusterProfiler::bitr(matrixSym, fromType = annoType, toType =
## c("SYMBOL"), : 95.54% of input gene IDs are fail to map...
## Warning in retrieveFromGEO(matrixURL, countMatrixPath, species, metaDataPath =
```

```
## metaDataPath, : Nb of lines removed due to duplication of row names: 1
countMatrix <- result[[1]]
columnsMetaData <- result[[2]]
## Correct the columns names to fit conclus input requirement
## The columns should be: cellName, state, and cellBarcode
columnsMetaData <- columnsMetaData[,c(1,3,2)]
colnames(columnsMetaData) <- c("cellName", "state", "cellBarcode")

## 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),]

## Removing control human cells
idxHumanPos <- which(columnsMetaData$state == "Pos Ctrl")
idxHumanNeg <- which(columnsMetaData$state == "Neg Ctrl")
columnsMetaData <- columnsMetaData[-c(idxHumanPos, idxHumanNeg),]
countMatrix <- countMatrix[,-c(idxHumanPos, idxHumanNeg)]

## Removing genes not having an official symbol
idxENS <- grep("ENSMUSG", rownames(countMatrix))
countMatrix <- countMatrix[-idxENS,]
```

4.2.2 Using local data

If you executed the code above, a matrix path is stored in `countMatrixPath` and a file path to the meta-data is stored in `metaDataPath`. Just indicate the paths to your files in these two variables. Please also note that to load the metadata, you need to fill in the `columnID` argument with the name of the column containing the names of the cells. These names must be the same as those in the count matrix.

```
## countMatrixPath <- ""
## metaDataPath <- ""

countMatrix <- loadDataOrMatrix(countMatrixPath, type="countMatrix")
columnsMetaData <- loadDataOrMatrix(file=metaDataPath, type="coldata",
                                    columnID="")

## Filtering steps to add here as performed above
```

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 = "E11_5",
        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 17484 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.
## Number of cells removed after filtering: 169
## Number of cells remaining after filtering: 1313
## Running filterGenes.
## Number of genes removed after filtering: 5790
## Number of genes remaining after filtering: 11694
## Running normalization. It can take a while depending on the number of cells.
## summary(sizeFactors(sceObject)):
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.2488  0.5213  0.7882  1.0000  1.2617 10.1622
```

```
p <- testClustering(scr, writeOutput=TRUE, silent=TRUE)
```

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

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

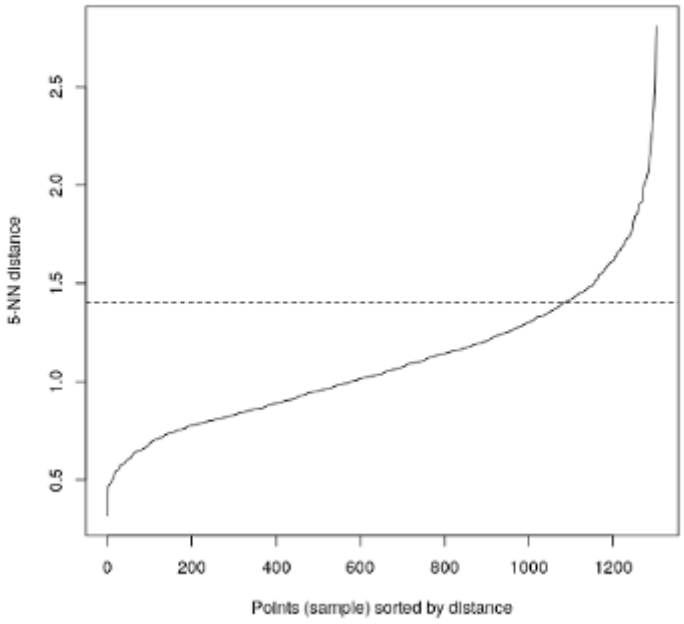

Figure 1: If the dashed horizontal line in the k-NN distance plot lays on the “knee” of the curve, it means that optimal epsilon is equal to the intersection of the line to the y-axis

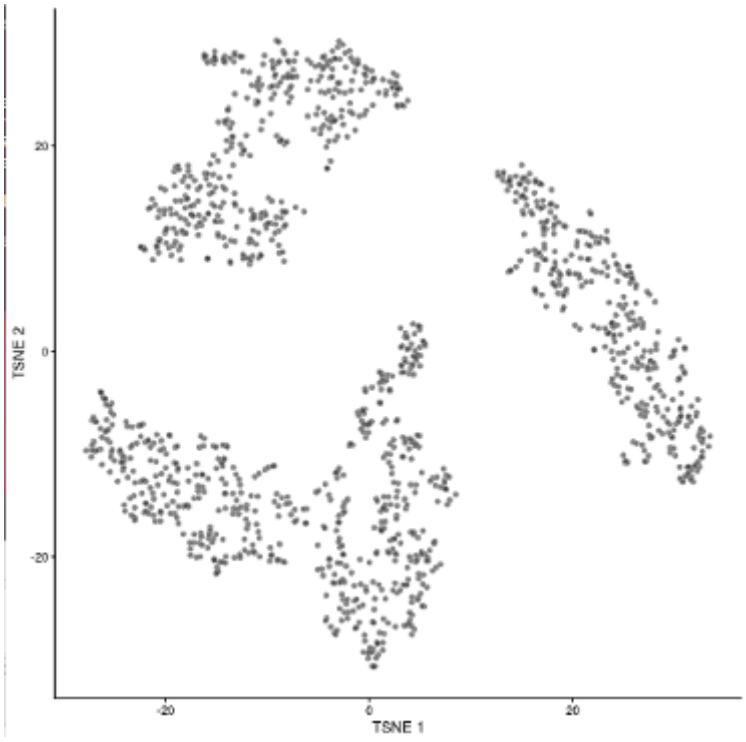

Figure 2: One of the 14 tSNE (by default) generated by conclus

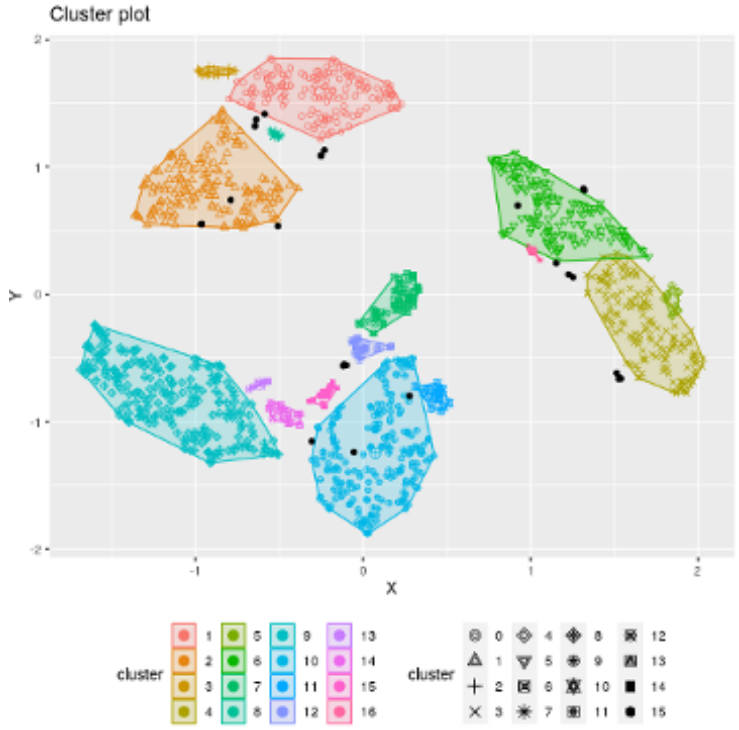

Figure 3: One of the 84 dbscan clustering solutions generated by conclus

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 advice to first try the analysis with this option and to set it to FALSE if no rare populations are found.

```
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] 17484  1482
dim(normMat)
## [1] 11694  1313

# show first columns and rows of the count matrix
originalMat[1:5,1:5]
##       c1 c2 c3 c4 c5
## Gnai3  0  0  0  0  0
## Cdc45  0  0  0  0  0
## H19   13 10 17  9  7
## Scml2  0  0  0  0  0
## Apoh   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 0.000000 0.000000 0.000000 0.000000 0.000000
## Cdc45 0.000000 0.000000 0.000000 0.000000 0.000000
## H19   3.223584 3.540887 4.705018 4.315128 3.505963
## Scml2 0.000000 0.000000 0.000000 0.000000 0.000000
## Narf  0.000000 0.000000 1.307690 0.000000 0.000000

# 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 YS_VECad_min AACCAACCTGC     1621     2795   1168  72.05429       6
## c2       c2 YS_VECad_min AACCAAGACGA      984     1552    738  75.00000       6
## c3       c3 YS_VECad_min AACCAAGCCTG      912     1373    696  76.31579       7
```

```
## c4       c4 YS_VECad_min AACCAAGCTAA      653      904    527  80.70444       1
## c5       c5 YS_VECad_min AACCAAGGCCT      846     1333    656  77.54137       6
## c6       c6 YS_VECad_min AACCAAGTTGG      474      626    391  82.48945       3
##    mtSum codGenes codSum     mtPer   codPer  sumMtPer sumCodPer filterPassed
## c1    42     1525   2472 0.3701419 94.07773 1.5026834  88.44365            1
## c2    39      932   1393 0.6097561 94.71545 2.5128866  89.75515            1
## c3    21      867   1235 0.7675439 95.06579 1.5294975  89.94902            1
## c4     1      616    816 0.1531394 94.33384 0.1106195  90.26549            1
## c5    41      807   1224 0.7092199 95.39007 3.0757689  91.82296            1
## c6    10      447    590 0.6329114 94.30380 1.5974441  94.24920            1
##    sizeFactor
## c1  1.5585578
## c2  0.9399435
## c3  0.6777597
## c4  0.4760406
## c5  0.6756387
## c6  0.2952552

# 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
## Scml2             Scml2 ENSMUSG00000000037  Scml2
## Narf               Narf ENSMUSG00000000056   Narf
## Cav2               Cav2 ENSMUSG00000000058   Cav2
##                                                                  GENENAME
## Gnai3 guanine nucleotide binding protein (G protein), alpha inhibiting 3
## Cdc45                                                 cell division cycle 45
## H19                        H19, imprinted maternally expressed transcript
## Scml2                                   Scm polycomb group protein like 2
## Narf                                 nuclear prelamin A recognition factor
## Cav2                                                          caveolin 2
##       chromosome_name   gene_biotype      go_id    name_1006
## Gnai3               3 protein_coding       <NA>         <NA>
## Cdc45              16 protein_coding       <NA>         <NA>
## H19                 7         lncRNA       <NA>         <NA>
## Scml2               X protein_coding       <NA>         <NA>
## Narf               11 protein_coding       <NA>         <NA>
## Cav2                6 protein_coding GO:0009986 cell surface
```

5.2 Generation of t-SNE coordinates

runCONCLUS 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, cores=2)
## 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 -15.20261 -18.671392
## c2 -14.50912 -19.335823
## c3 -17.04793  -1.500220
## c4 -19.38192  -1.180351
## c5 -16.69633   1.251342
## c6 -17.27813   1.144071
```

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, cores=2)
## The following matrix shows how many times a number of clusters 'k' has been found among the dbscan solutio
##                         1  2  3 4  5  6  7  8 9 10 11 12 13 14
## Number of clusters k :  6  7  8 9 10 11 13 14 5 12  4 15 16 17
## Count :                16 13 13 9  9  7  4  4 3  2  1  1  1  1
##
## Statistics about number of clusters 'k' among dbscan solutions:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   4.000   7.000   8.000   8.786  10.000  17.000
##
## Suggested clusters number to use in clusterCellsInternal() : clusterNumber=8.
```

Results can be explored as follows:

```
dbscanList <- getDbscanList(scr)
clusteringList <- lapply(dbscanList, getClustering)
clusteringList[[1]][, 1:10]
##  c1  c2  c3  c4  c5  c6  c7  c8  c9 c10
##   1   1   2   2   2   2   1   2   2   3
```

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.

Note that the number of clusters is set to 10 (it was set to 11 in the original study, see data section). Considering that clusters can be merged afterwards (see Supervised clustering section), we advise to keep this number. One might want to increase it if the cell similarity matrix or the tSNE strongly suggest more clusters but this number is suitable for most experiments in our hands.

```
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
## 190 186 341  55 137  28 111   8 126 131
```

```
cci <- getCellsSimilarityMatrix(scr)
cci[1:10, 1:10]
##            c1        c2        c3        c4        c5        c6        c7
## c1  1.0000000 1.0000000 0.6309524 0.6309524 0.6309524 0.6309524 0.9523810
## c2  1.0000000 1.0000000 0.6309524 0.6309524 0.6309524 0.6309524 0.9523810
## c3  0.6309524 0.6309524 1.0000000 1.0000000 1.0000000 1.0000000 0.6071429
## c4  0.6309524 0.6309524 1.0000000 1.0000000 1.0000000 1.0000000 0.6071429
## c5  0.6309524 0.6309524 1.0000000 1.0000000 1.0000000 1.0000000 0.6071429
## c6  0.6309524 0.6309524 1.0000000 1.0000000 1.0000000 1.0000000 0.6071429
## c7  0.9523810 0.9523810 0.6071429 0.6071429 0.6071429 0.6071429 0.9642857
## c8  0.6309524 0.6309524 1.0000000 1.0000000 1.0000000 1.0000000 0.6071429
## c9  0.4761905 0.4761905 0.7619048 0.7619048 0.7619048 0.7619048 0.4523810
## c10 0.5833333 0.5833333 0.7142857 0.7142857 0.7142857 0.7142857 0.5595238
##            c8        c9       c10
## c1  0.6309524 0.4761905 0.5833333
## c2  0.6309524 0.4761905 0.5833333
## c3  1.0000000 0.7619048 0.7142857
## c4  1.0000000 0.7619048 0.7142857
## c5  1.0000000 0.7619048 0.7142857
## c6  1.0000000 0.7619048 0.7142857
## c7  0.6071429 0.4523810 0.5595238
## c8  1.0000000 0.7619048 0.7142857
## c9  0.7619048 0.8095238 0.4761905
## c10 0.7142857 0.4761905 0.7857143
```

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  1.0000000 0.6309524 0.0000000 0.00000000 0.00000000 0.00000000 0.0000000
## 2  0.6309524 1.0000000 0.0000000 0.00000000 0.00000000 0.00000000 0.0000000
## 3  0.0000000 0.0000000 0.9761905 0.00000000 0.00000000 0.00000000 0.0000000
## 4  0.0000000 0.0000000 0.0000000 1.00000000 0.02380952 0.47619048 0.2380952
## 5  0.0000000 0.0000000 0.0000000 0.02380952 1.00000000 0.04761905 0.1547619
## 6  0.0000000 0.0000000 0.0000000 0.47619048 0.04761905 1.00000000 0.4404762
## 7  0.0000000 0.0000000 0.0000000 0.23809524 0.15476190 0.44047619 1.0000000
## 8  0.0000000 0.0000000 0.0000000 0.14285714 0.02380952 0.14285714 0.1428571
## 9  0.0000000 0.0000000 0.0000000 0.02380952 0.50000000 0.04761905 0.1190476
## 10 0.0000000 0.0000000 0.0000000 0.14285714 0.33333333 0.16666667 0.2380952
##             8          9        10
## 1  0.00000000 0.00000000 0.0000000
## 2  0.00000000 0.00000000 0.0000000
## 3  0.00000000 0.00000000 0.0000000
## 4  0.14285714 0.02380952 0.1428571
## 5  0.02380952 0.50000000 0.3333333
## 6  0.14285714 0.04761905 0.1666667
## 7  0.14285714 0.11904762 0.2380952
## 8  1.00000000 0.02380952 0.1428571
## 9  0.02380952 1.00000000 0.6904762
## 10 0.14285714 0.69047619 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 (if the metadata contain a 'state' column) 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]]
```

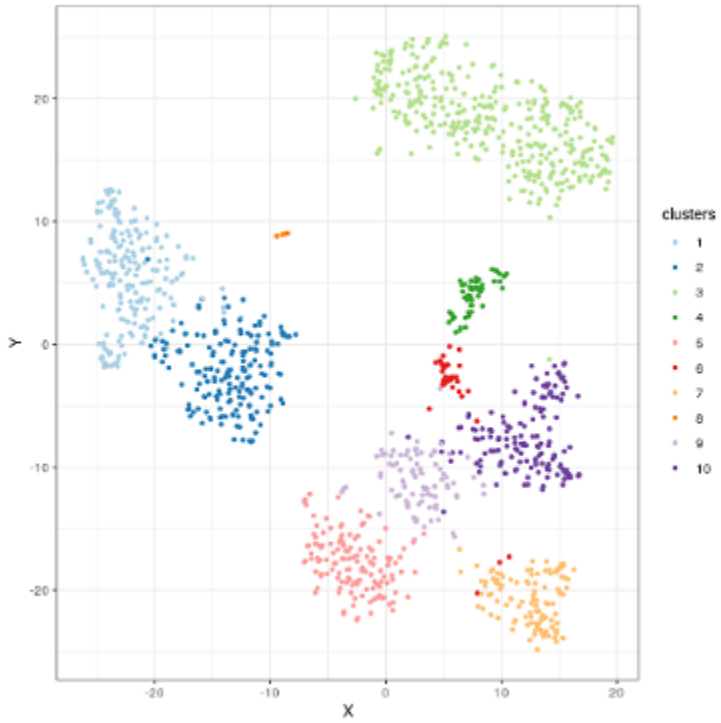

Figure 4: DBscan results on the 5th tSNE

This tSNE suggests that the cluster 8 corresponds to a rare population of cells. We can also appreciate that clusters 1 and 2, as clusters 9 and 10 could be considered as single clusters respectively.

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

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

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

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

One can see that the cluster 8 contains only cells of the YS.

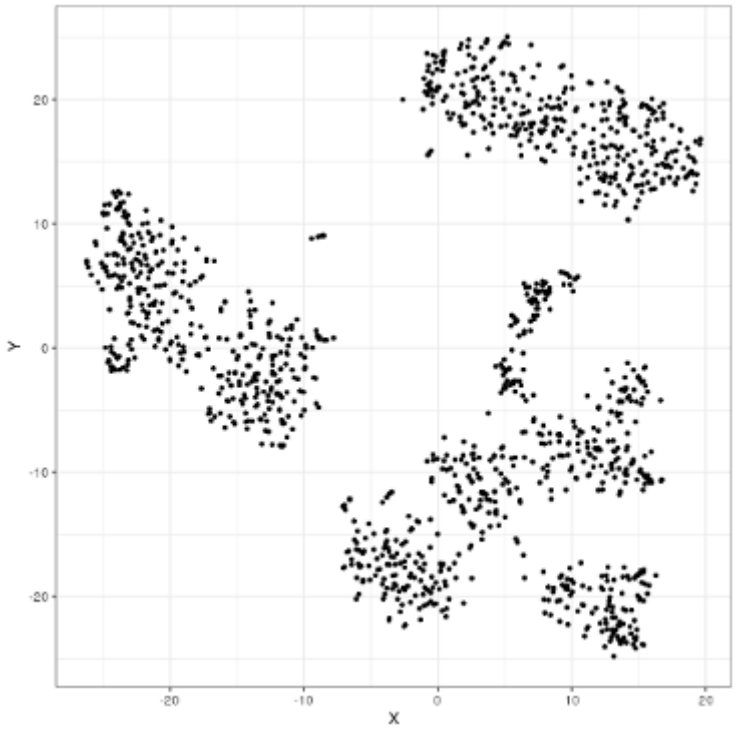

Figure 5: The 5th tSNE solution without coloring

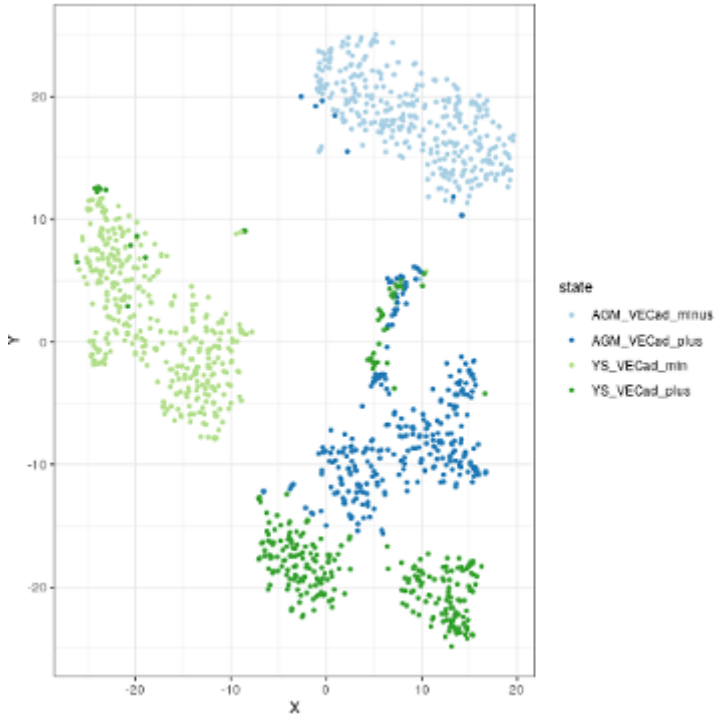

Figure 6: The 5th tSNE solution colored by cell condition

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)
```

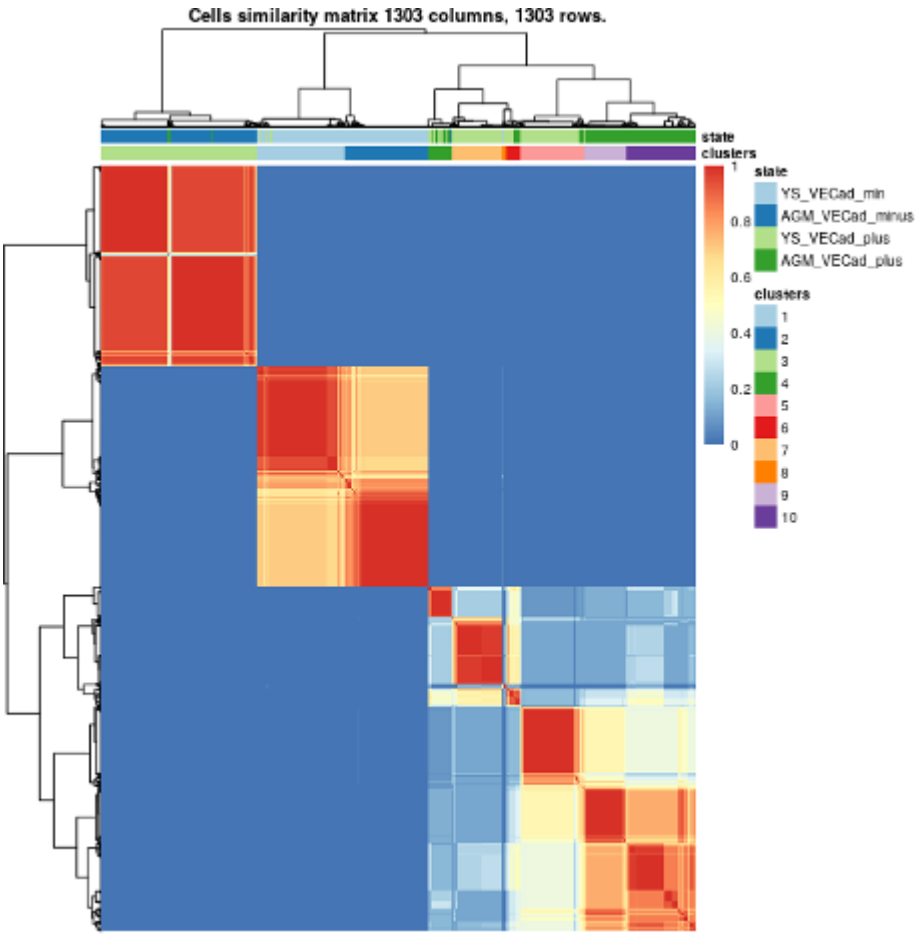

Figure 7: **Cell similarity matrix showing the conservation of clustering across the 84 solutions**

CellsSimilarityMatrix is symmetrical and its size proportional 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 VE-Cad plus cells from the VE-Cad minus but also find sub-populations within these groups.

5.5.3 Cluster similarity heatmap

```
plotClustersSimilarity(scr)
```

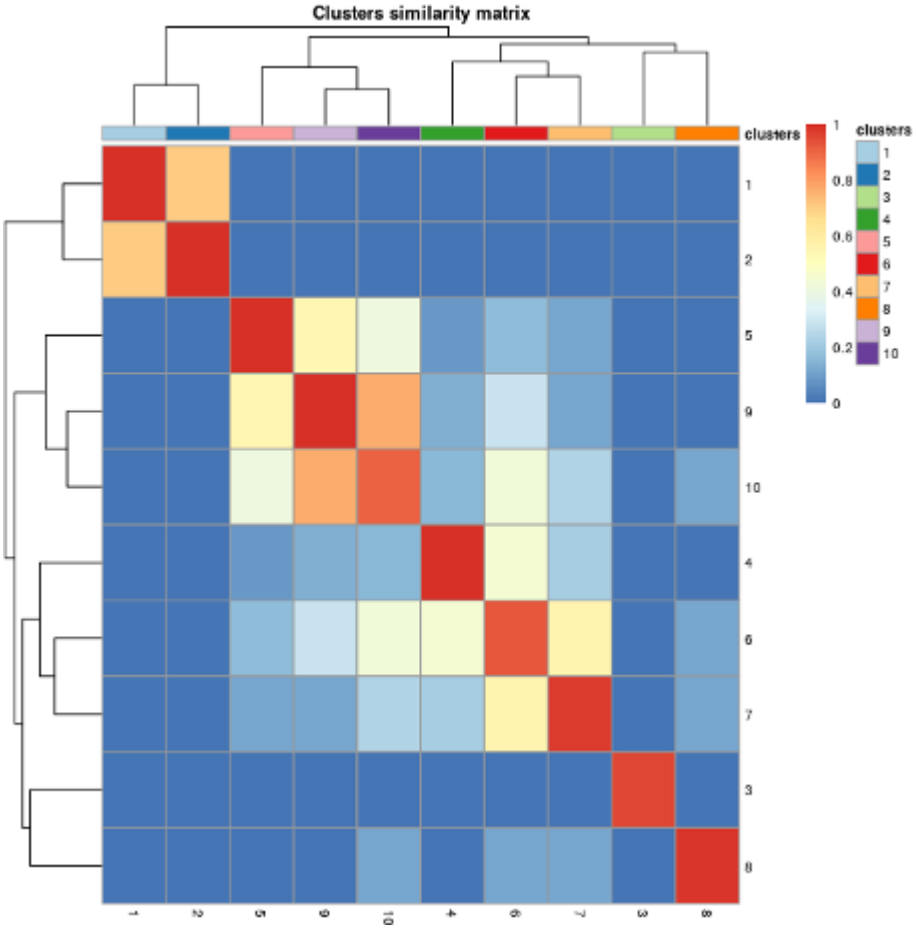

Figure 8: **Cluster similarity matrix**
Each cell belonging to a particular cluster were merged into a pseudo-cell.

In the *clusterSimilarityMatrix*, we can see that all clusters have a high value of similarity across all clustering solutions. 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. The orange on the diagonal suggests that groups can be merged. Bluish color points to a cluster of dbscan “outliers” that usually surrounds dense clouds of cells in t-SNE plots.

As observed previously on the tSNE, clusters 1 and 2 as 9 and 10 are very similar. Clusters 3 and 8 are very different.

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 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
```

```
markers <- getMarkerGenesList(scr, cluster=1)
```

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

```
scr <- retrieveTopClustersMarkers(scr, removeDuplicates=TRUE)
topMarkers <- getTopMarkers(scr)
topMarkers
##       geneName clusters
## 1         Meg3        1
## 2          Lum        1
## 3       Col1a2        1
## 4      Smarca1        1
## 5      Colec10        1
## 6       Col1a1        1
## 7       Col3a1        1
## 8       Col5a1        1
## 9       Pdzrn3        1
## 10      Klhl29        1
## 11       Prdx2        2
## 13        Gpx1        2
## 14       Rpl41        2
## 15        Fth1        2
## 16        Car2        2
## 17       Acta2        2
## 18       Rplp0        2
## 19       Alas2        2
## 20      Slc4a1        2
## 21        Gpc6        3
## 22         Ptn        3
## 23       Tenm3        3
## 24         Mdk        3
## 25        Aff3        3
## 26        Peg3        3
## 27      Igfbp5        3
## 28       Foxp2        3
## 29       Zfhx4        3
```

```
## 30       Auts2        3
## 31      Fcer1g        4
## 32        Ctss        4
## 33      Laptm5        4
## 34        Aif1        4
## 35      Tyrobp        4
## 36        C1qc        4
## 37        Lst1        4
## 38       Fcgr3        4
## 39        C1qb        4
## 40       Hcls1        4
## 41         Afp        5
## 42       Apoa2        5
## 43         Ttr        5
## 44       Apoa1        5
## 45  St6galnac3        5
## 46         Mt1        5
## 47        Apob        5
## 48       Nudt4        5
## 49       S100g        5
## 50      Hsd3b6        5
## 51       Mctp1        6
## 52       Cd180        6
## 53      P2ry12        6
## 54        Mitf        6
## 55       Ptprj        6
## 56      Tbxas1        6
## 57       Hdac9        6
## 58      Slc9a9        6
## 59       F13a1        6
## 60   Tnfrsf11a        6
## 61       Stab2        7
## 62       Lyve1        7
## 63      Calcrl        7
## 64        Flt1        7
## 65    Ralgapa2        7
## 66      Gpr182        7
## 68     Col13a1        7
## 69        Ldb2        7
## 70         Tek        7
## 71       Cldn1        8
## 72         A2m        8
## 73        Maob        8
## 74    Serpind1        8
## 75       Krt20        8
## 76         Klb        8
## 77       Pcbd1        8
## 78          F2        8
## 79    Serpinf2        8
## 80        Lrp2        8
## 81        Cdk8        9
```

```
## 82      Camk1d        9
## 84       Epb41        9
## 85     Mir6236        9
## 87         Kel        9
## 89        Ank1        9
## 90    Slc25a21        9
## 91       Ptprg       10
## 92       Pde8a       10
## 93       Ptprm       10
## 94     Col18a1       10
## 95     Tspan18       10
## 96        Cdh5       10
## 97       Mecom       10
## 98       Mast4       10
## 99      Pecam1       10
## 100     Col4a1       10
```

Cluster 8 contains the marker genes A2m, Cldn1, Maob, Pcbd1, Pdzk1, Krt20, Klb, Serpind1, and F2. Alpha-2-Macroglobulin (A2m) is a large (720 KDa) plasma protein found in the blood and is mainly produced by the liver; Claudins (CLDNs) are a family of integral membrane proteins, Cldn1 being expressed by epithelial liver cells; The monoamine oxidase (Maob) is found in abundance in the liver. All these markers play a role in the liver. This sub-population is hence called liver-like hereafter.

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)
```

One can also visualize the heatmap without mean centering and appreciate the importance of the normalization of colors by row. Indeed, the different markers are much harder to identify.

```
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.

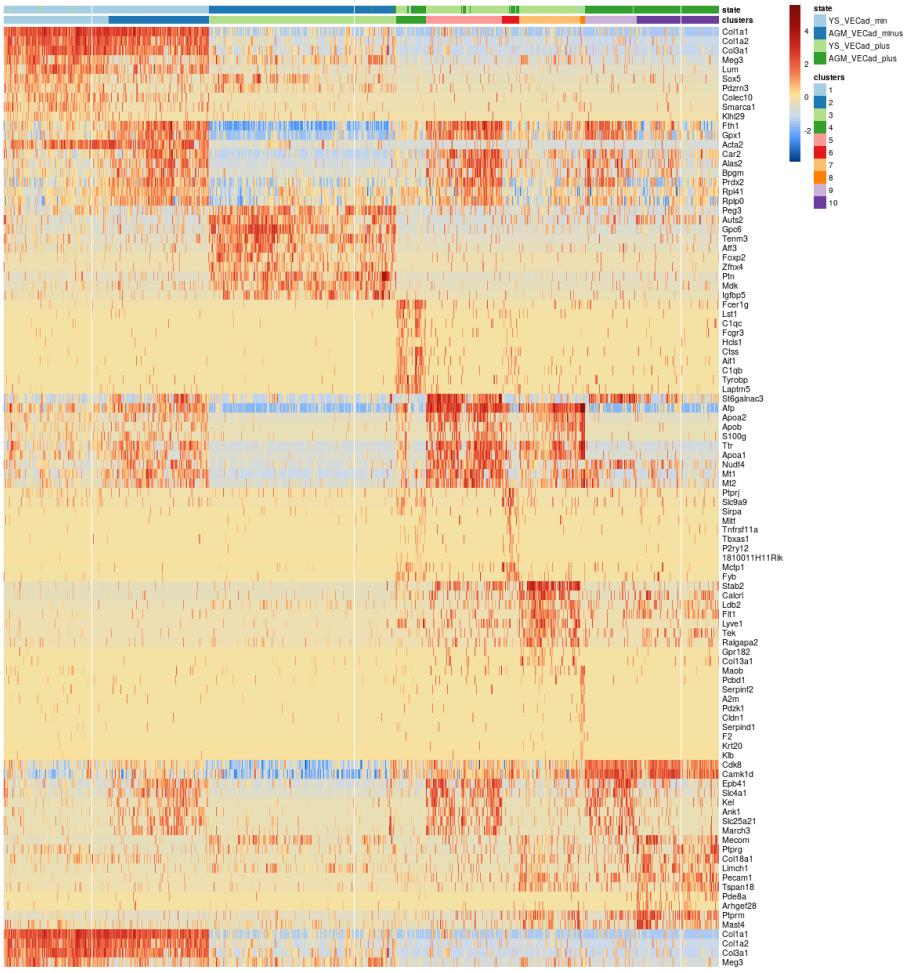

Figure 9: Heatmap showing the expression of the top 10 markers for each cluster
The values are normalized according to the mean.

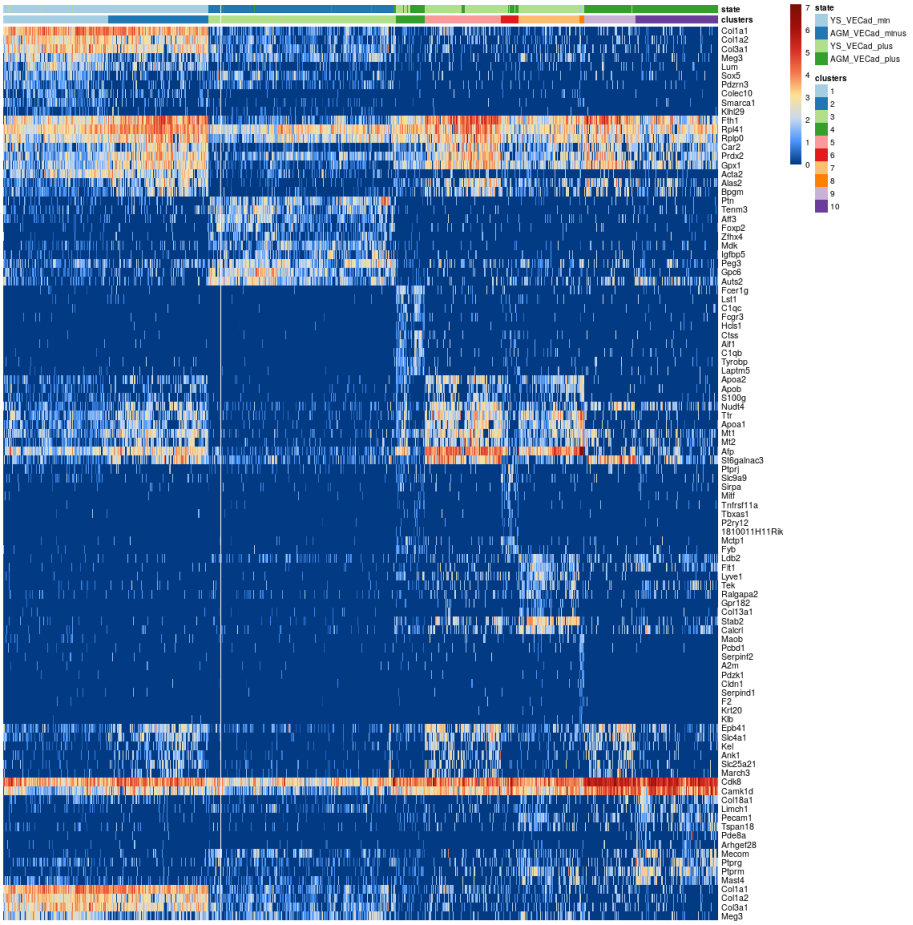

Figure 10: Same heatmap as before without normalizing by the mean

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.

Cluster 8 was identified as being a liver-like rare population. Below are examples of the expression of its marker genes:

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

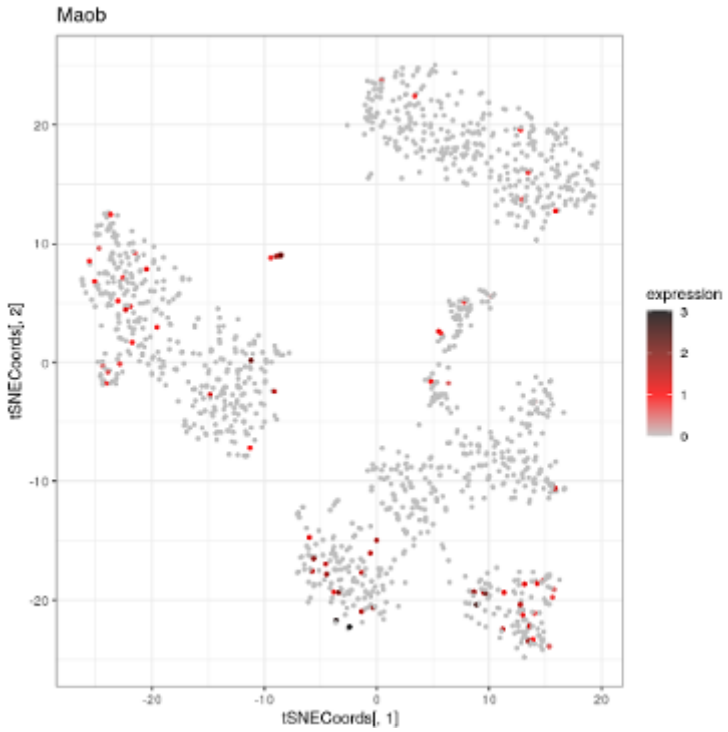

Figure 11: Maob Marker gene of cluster 8

```
plotGeneExpression(scr, "Pcbd1", tSNEpicture=5)
```

```
plotGeneExpression(scr, "Serpinf2", tSNEpicture=5)
```

```
plotGeneExpression(scr, "Cldn1", tSNEpicture=5)
```

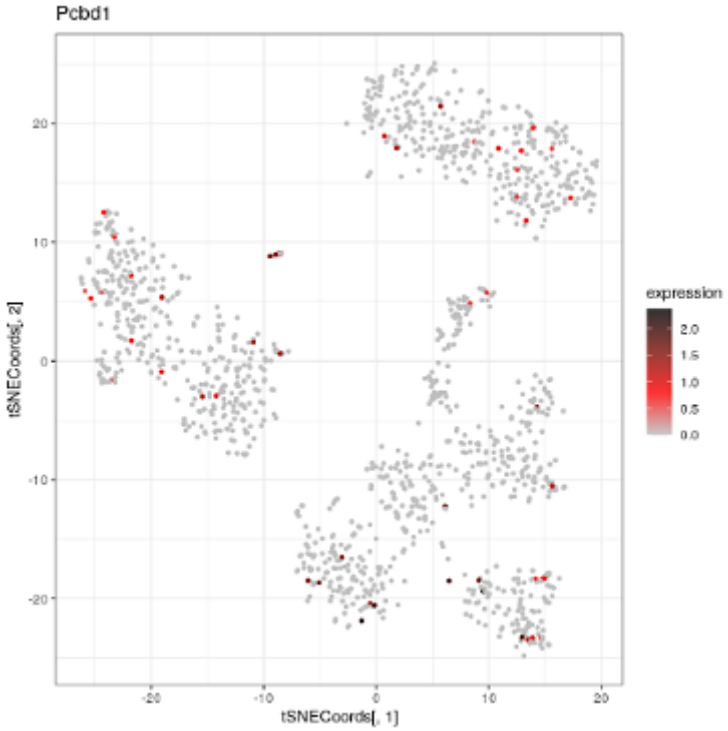

Figure 12: Pcbd1 Marker gene of cluster 8

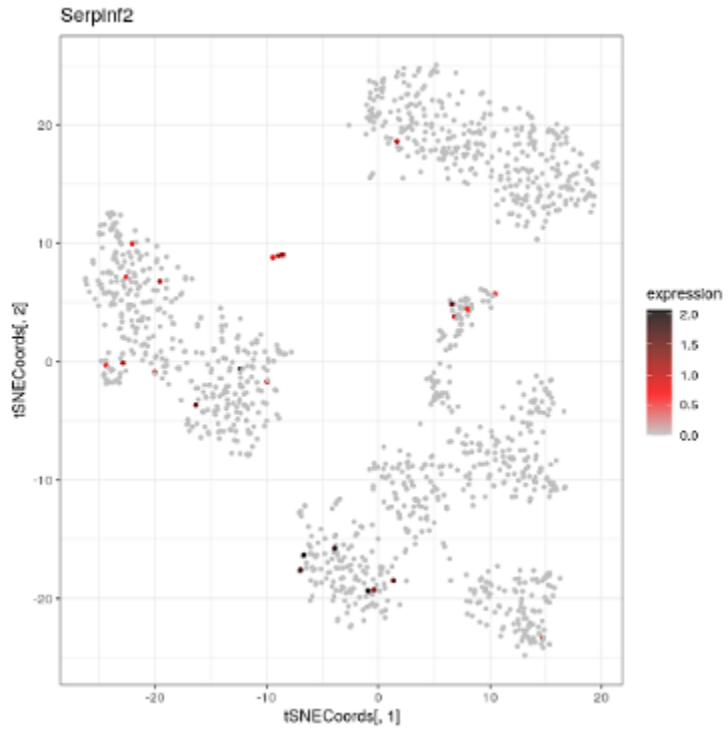

Figure 13: Serpinf2 Marker gene of cluster 8

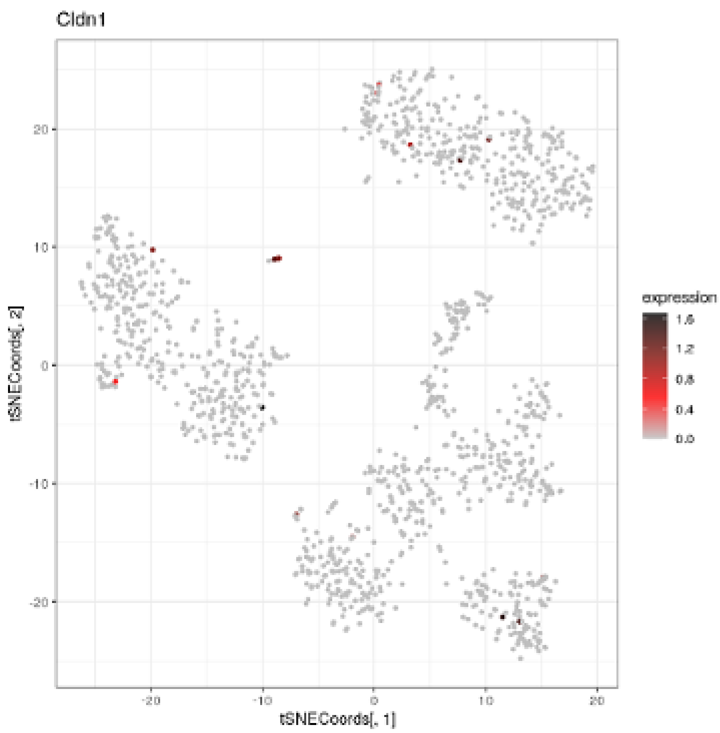

Figure 14: Cldn1 Marker gene of cluster 8

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)
##    Symbol clusters   gene_biotype
## 1    Meg3        1         lncRNA
## 2     Lum        1 protein_coding
## 3  Col1a2        1 protein_coding
## 4 Smarca1        1 protein_coding
## 5 Colec10        1 protein_coding
## 6  Col1a1        1 protein_coding
##                                                      entrezgene_description
## 1
```

```
## 2                                                                                        lumican
## 3                                                                           collagen, type I, alpha 2
## 4 SWI/SNF related, matrix associated, actin dependent regulator of chromatin, subfamily a, member 1
## 5                                                                       collectin sub-family member 10
## 6                                                                           collagen, type I, alpha 1
##                                                                                    mgi_description
## 1                                                                               maternally expressed 3
## 2                                                                                        lumican
## 3                                                                           collagen, type I, alpha 2
## 4 SWI/SNF related, matrix associated, actin dependent regulator of chromatin, subfamily a, member 1
## 5                                                                       collectin sub-family member 10
## 6                                                                           collagen, type I, alpha 1
##   chromosome_name    ensembl_gene_id      mgi_id entrezgene_id
## 1              12 ENSMUSG00000021268 MGI:1202886            NA
## 2              10 ENSMUSG00000036446  MGI:109347         17022
## 3               6 ENSMUSG00000029661   MGI:88468         12843
## 4               X ENSMUSG00000031099 MGI:1935127         93761
## 5              15 ENSMUSG00000038591 MGI:3606482        239447
## 6              11 ENSMUSG00000001506   MGI:88467         12842
##                uniprot_gn_id uniprot_gn_symbol
## 1
## 2                     P51885               Lum
## 3        Q01149, Q3TX57, E0CXI2            Col1a2
## 4 Q6PGB8, Q8BS67, F6QS43, F6Z6F4           Smarca1
## 5                     Q8CF98           Colec10
## 6                     P11087            Col1a1
##
## 1
## 2
## 3
## 4
## 5
## 6 GO:0005515, GO:0005201, GO:0007605, GO:0071230, GO:0005737, GO:0001501, GO:0043588, GO:0046872, GO:0005
```

`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 Shvartsman *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.

As previously mentioned, clusters 1 and 2 as 9 and 10 are very similar. 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 1 and 2 (9 and 10) together.

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

## Replace "2/10" by "1/9" to merge 1/2 and 9/10
clustCellsDf$clusters[which(clustCellsDf$clusters == 2)] <- 1
clustCellsDf$clusters[which(clustCellsDf$clusters == 10)] <- 9

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

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

```
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 Col1a1 is a good marker of cluster 1 and that Cdk8 is a good marker of cluster 9 (at the bottom). One can visualize them in the t-SNE plots below.

```
plotGeneExpression(scrUpdated, "Col1a1", tSNEpicture=5)
```

```
plotGeneExpression(scrUpdated, "Cdk8", tSNEpicture=5)
```

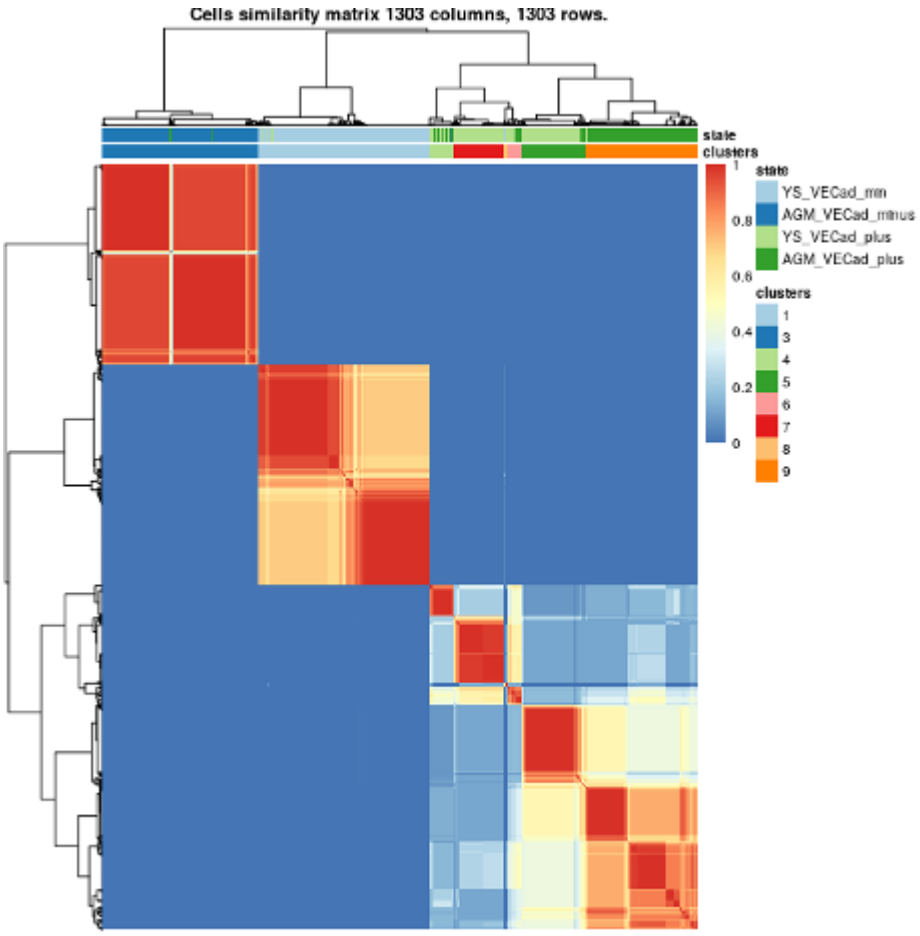

Figure 15: Updated cells similarity matrix with merged 1/2 and 9/10

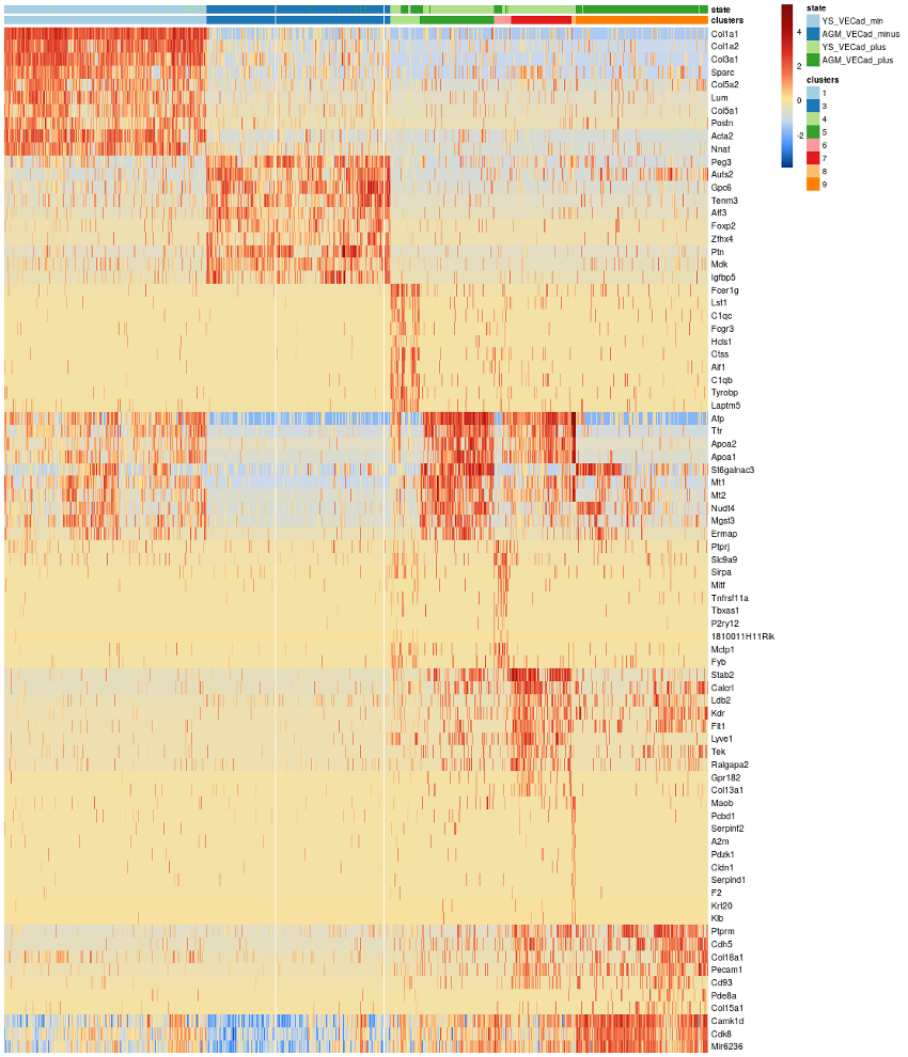

Figure 16: Updated cells heatmap with merged 1/2 and 9/10

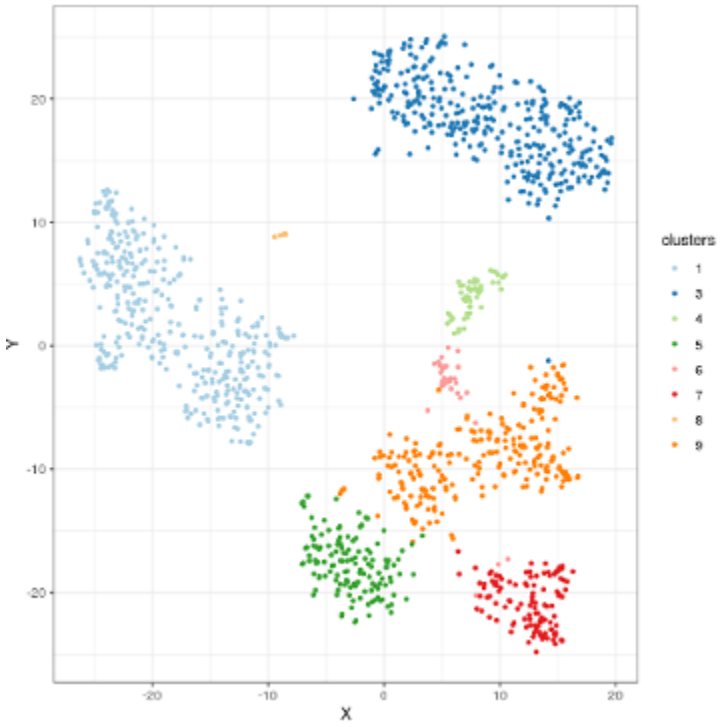

Figure 17: 5th tSNE solution colored by dbscan result showing the merged clusters

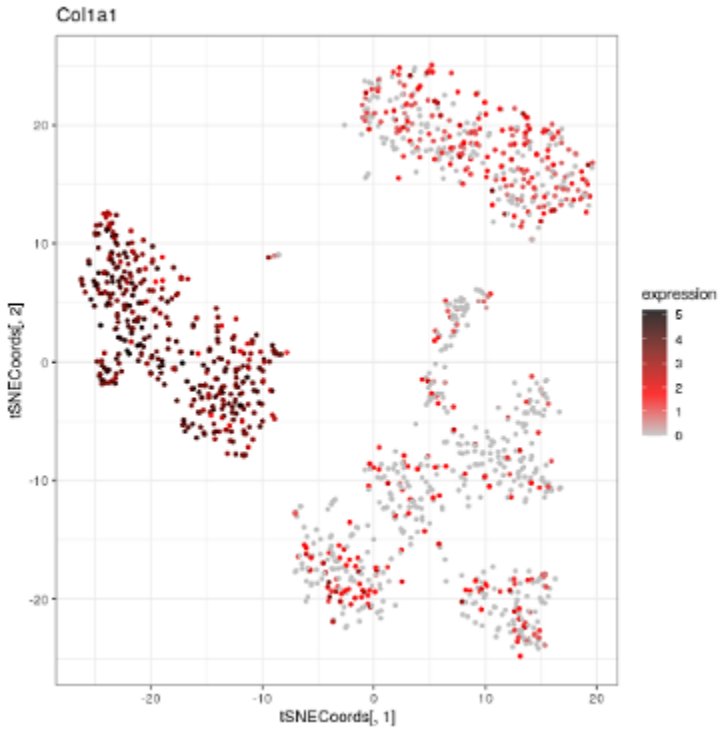

Figure 18: Col1a1 Marker gene for cluster 1 (mix of old clusters 1 and 2)

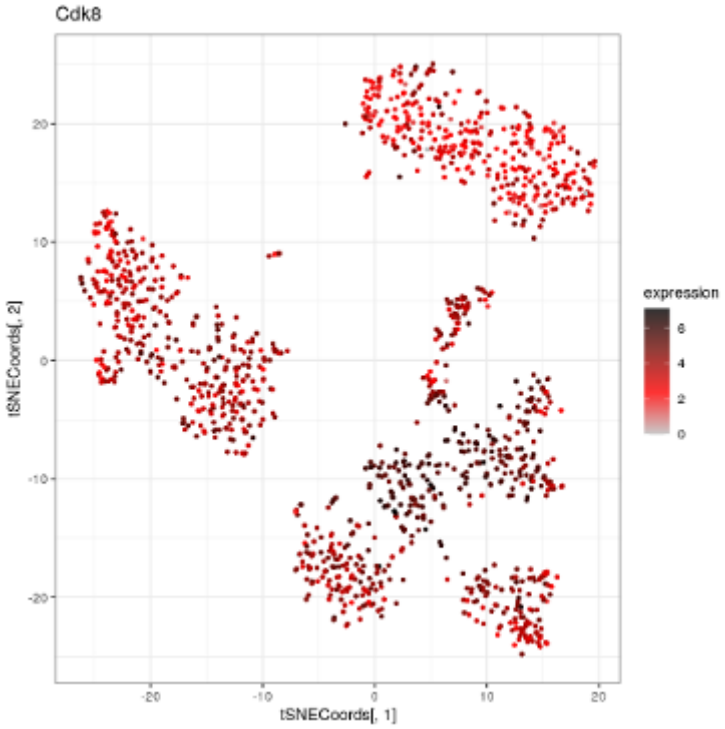

Figure 19: Cdk8 Marker gene for clusters 9 (mix of old clusters 9 and 10)

10 Conclusion

Here we demonstrated how to use CONCLUS and combine multiple parameters testing for sc-RNA-seq analysis. It allowed us to identify a rare population in the data of Shvartsman *et al* and will help gaining deeper insights into others.

11 Session info

```
sessionInfo()
## R version 4.1.3 (2022-03-10)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows Server 2012 R2 x64 (build 9600)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=C
## [2] LC_CTYPE=English_United States.1252
## [3] LC_MONETARY=English_United States.1252
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United States.1252
##
## attached base packages:
## [1] stats4    stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
## [1] org.Mm.eg.db_3.14.0  AnnotationDbi_1.56.2 IRanges_2.28.0
## [4] S4Vectors_0.32.4     Biobase_2.54.0       BiocGenerics_0.40.0
## [7] conclus_1.2.4        BiocStyle_2.22.0
##
## loaded via a namespace (and not attached):
##   [1] utf8_1.2.2                  tidyselect_1.1.2
##   [3] RSQLite_2.2.12              htmlwidgets_1.5.4
##   [5] grid_4.1.3                  BiocParallel_1.28.3
##   [7] Rtsne_0.15                  scatterpie_0.1.7
##   [9] munsell_0.5.0               ScaledMatrix_1.2.0
##  [11] codetools_0.2-18            statmod_1.4.36
##  [13] scran_1.22.1                DT_0.22
##  [15] withr_2.5.0                 colorspace_2.0-3
##  [17] GOSemSim_2.20.0             filelock_1.0.2
##  [19] knitr_1.38                  SingleCellExperiment_1.16.0
##  [21] robustbase_0.95-0           DOSE_3.20.1
##  [23] MatrixGenerics_1.6.0        GenomeInfoDbData_1.2.7
##  [25] polyclip_1.10-0             bit64_4.0.5
##  [27] farver_2.1.0                pheatmap_1.0.12
##  [29] downloader_0.4              vctrs_0.4.0
##  [31] treeio_1.18.1               generics_0.1.2
##  [33] xfun_0.30                   BiocFileCache_2.2.1
```

```
##  [35] diptest_0.76-0              R6_2.5.1
##  [37] doParallel_1.0.17           GenomeInfoDb_1.30.1
##  [39] ggbeeswarm_0.6.0            graphlayouts_0.8.0
##  [41] rsvd_1.0.5                  locfit_1.5-9.5
##  [43] flexmix_2.3-17              bitops_1.0-7
##  [45] cachem_1.0.6                fgsea_1.20.0
##  [47] gridGraphics_0.5-1          DelayedArray_0.20.0
##  [49] assertthat_0.2.1            scales_1.1.1
##  [51] ggraph_2.0.5                nnet_7.3-17
##  [53] enrichplot_1.14.2           beeswarm_0.4.0
##  [55] gtable_0.3.0                beachmat_2.10.0
##  [57] tidygraph_1.2.1             rlang_1.0.2
##  [59] splines_4.1.3               lazyeval_0.2.2
##  [61] GEOquery_2.62.2             BiocManager_1.30.16
##  [63] yaml_2.3.5                  reshape2_1.4.4
##  [65] crosstalk_1.2.0             qvalue_2.26.0
##  [67] clusterProfiler_4.2.2       tools_4.1.3
##  [69] bookdown_0.25               ggplotify_0.1.0
##  [71] ggplot2_3.3.5               ellipsis_0.3.2
##  [73] jquerylib_0.1.4             RColorBrewer_1.1-3
##  [75] Rcpp_1.0.8.3                plyr_1.8.7
##  [77] sparseMatrixStats_1.6.0     progress_1.2.2
##  [79] zlibbioc_1.40.0             purrr_0.3.4
##  [81] RCurl_1.98-1.6              prettyunits_1.1.1
##  [83] dbscan_1.1-10               viridis_0.6.2
##  [85] SummarizedExperiment_1.24.0 ggrepel_0.9.1
##  [87] cluster_2.1.3               factoextra_1.0.7
##  [89] magrittr_2.0.3              data.table_1.14.2
##  [91] DO.db_2.9                   matrixStats_0.61.0
##  [93] hms_1.1.1                   patchwork_1.1.1
##  [95] evaluate_0.15               XML_3.99-0.9
##  [97] mclust_5.4.9                gridExtra_2.3
##  [99] compiler_4.1.3              biomaRt_2.50.3
## [101] scater_1.22.0               tibble_3.1.6
## [103] crayon_1.5.1                shadowtext_0.1.1
## [105] htmltools_0.5.2             ggfun_0.0.6
## [107] tzdb_0.3.0                  tidyr_1.2.0
## [109] aplot_0.1.3                 DBI_1.1.2
## [111] tweenr_1.0.2                dbplyr_2.1.1
## [113] MASS_7.3-56                 fpc_2.2-9
## [115] rappdirs_0.3.3              Matrix_1.4-1
## [117] readr_2.1.2                 cli_3.2.0
## [119] metapod_1.2.0               parallel_4.1.3
## [121] igraph_1.3.0                GenomicRanges_1.46.1
## [123] pkgconfig_2.0.3             scuttle_1.4.0
## [125] xml2_1.3.3                  foreach_1.5.2
## [127] ggtree_3.2.1                bslib_0.3.1
## [129] vipor_0.4.5                 dqrng_0.3.0
## [131] XVector_0.34.0              yulab.utils_0.0.4
## [133] stringr_1.4.0               digest_0.6.29
## [135] Biostrings_2.62.0           rmarkdown_2.13
```

```
## [137] fastmatch_1.1-3            tidytree_0.3.9
## [139] edgeR_3.36.0               DelayedMatrixStats_1.16.0
## [141] curl_4.3.2                 kernlab_0.9-30
## [143] modeltools_0.2-23          lifecycle_1.0.1
## [145] nlme_3.1-157               jsonlite_1.8.0
## [147] BiocNeighbors_1.12.0       viridisLite_0.4.0
## [149] limma_3.50.3               fansi_1.0.3
## [151] pillar_1.7.0               lattice_0.20-45
## [153] KEGGREST_1.34.0            fastmap_1.1.0
## [155] httr_1.4.2                 DEoptimR_1.0-11
## [157] GO.db_3.14.0               glue_1.6.2
## [159] png_0.1-7                  prabclus_2.3-2
## [161] iterators_1.0.14           bluster_1.4.0
## [163] bit_4.0.4                  sass_0.4.1
## [165] ggforce_0.3.3              class_7.3-20
## [167] stringi_1.7.6              blob_1.2.3
## [169] org.Hs.eg.db_3.14.0        BiocSingular_1.10.0
## [171] memoise_2.0.1              dplyr_1.0.8
## [173] irlba_2.3.5                ape_5.6-2
```