

Computational single-cell biology course

Marc Jan Bonder (m.bonder@dkfz.de) and Hakime Öztürk (h.oeztuerk@dkfz-heidelberg.de)

05 May, 2022

Contents

1	Dimensionality reduction and clustering on scRNA-seq data (hands-on)	2
1.1	Getting familiar with the pre-processed data	2
2	TASK 1: Pre-processing	3
3	TASK 2: Cluster cells.	7
4	Homework (Hierarchical Clustering)	10

1 Dimensionality reduction and clustering on scRNA-seq data (hands-on)

We will work on scRNA-seq data of mouse gastrulation and early organogenesis from [Pijuan-Sala et al., 2019](#). [This Shiny application](#) provides an interactive interface that allows users to validate their own analysis on this data. You can reach the original data and related scripts from the [Github page](#).

Gastrulation is a phase early in the embryonic development of most animals, during which the single-layered blastula is reorganized into a multilayered structure known as the gastrula ([Wikipedia, 2020-06-02](#)).

1.1 Getting familiar with the pre-processed data

Let's start with including required libraries.

```
suppressPackageStartupMessages({  
  library(Seurat)  
  library(ggplot2)  
  library(dplyr)  
  library(data.table)  
  library(cowplot)  
  set.seed(1)  
})
```

In the previous practical, we have learned the essential preprocessing steps for working with scRNA-seq. Here we will start working with the preprocessed version of the mouse gastrulation scRNA-seq data. We will load the pre-saved `Seurat` object of this data.

Warning: The pre-processed mouse gastrulation scRNA-seq `Seurat` object is large (~2GBs). In today's practical, we will subset the data for fast computation. You can, however, work on the [original data](#) for additional practices. The [meta-data](#) is provided separately.

```
mjsc <- readRDS('data/gastrulation/mjsc.rds')  
mjsc  
## An object of class Seurat  
## 29452 features across 116312 samples within 1 assay  
## Active assay: RNA (29452 features, 0 variable features)
```

Reminder:

- number of rows = genes = features,
- number of columns = cells = samples

```
# let's see the available slots  
slotNames(mjsc)  
## [1] "assays" "meta.data" "active.assay" "active.ident" "graphs"  
## [6] "neighbors" "reductions" "project.name" "misc" "version"  
## [11] "commands" "tools"
```

Now let's look at to the metadata table of the dataset that contains an overview of the samples.

```
# see it is empty for now
head(mgsc@meta.data, 5)
## data frame with 0 columns and 5 rows

# now let's load the metadata
metadata <- fread('data/gastrulation/sample_metadata.txt.gz') %>% .[stripped==FALSE & doublet==FALSE]

# and add the metadata to our seurat object
mgsc <- AddMetaData(mgsc, metadata = data.frame(metadata, row.names = metadata$cell))

# now let's see once more
head(mgsc@meta.data, 5)
##           cell          barcode sample stage sequencing.batch doublet stripped
## cell_1 cell_1 AAAGGCCTCCACAA      1 E6.5                1 FALSE FALSE
## cell_2 cell_2 AACAACTCGCCTT      1 E6.5                1 FALSE FALSE
## cell_5 cell_5 AACAGAGAATCAGC      1 E6.5                1 FALSE FALSE
## cell_6 cell_6 AACATATGAATCGC      1 E6.5                1 FALSE FALSE
## cell_8 cell_8 AACCGATGGCTTCC      1 E6.5                1 FALSE FALSE
##           celltype      umapX      umapY      celltype2      celltype3
## cell_1      Epiblast -10.227546 -2.8816875      Epiblast Epiblast-PS
## cell_2 Primitive_Streak -6.625458 0.1089605 Primitive_Streak Epiblast-PS
## cell_5 ExE_ectoderm 10.061009 -0.0293132 ExE_ectoderm ExE_ectoderm
## cell_6      Epiblast -10.454418 -0.2694517      Epiblast Epiblast-PS
## cell_8      Epiblast -11.047206 -2.2052687      Epiblast Epiblast-PS
```

Now we are able to see the annotations (e.g. cell type) for each cell. There is also a column named `stage` which shows the embryonic day the cells were sequenced. To speed up our experiments, we will work on a subset of cells that belong to stage E6.75.

```
mgsc_subset <- mgsc[ , mgsc@meta.data$stage=='E6.75']
mgsc_subset
## An object of class Seurat
## 29452 features across 2075 samples within 1 assay
## Active assay: RNA (29452 features, 0 variable features)
# now lets save this subset
saveRDS(mgsc_subset, file = "data/gastrulation/mgsc_e675.rds")

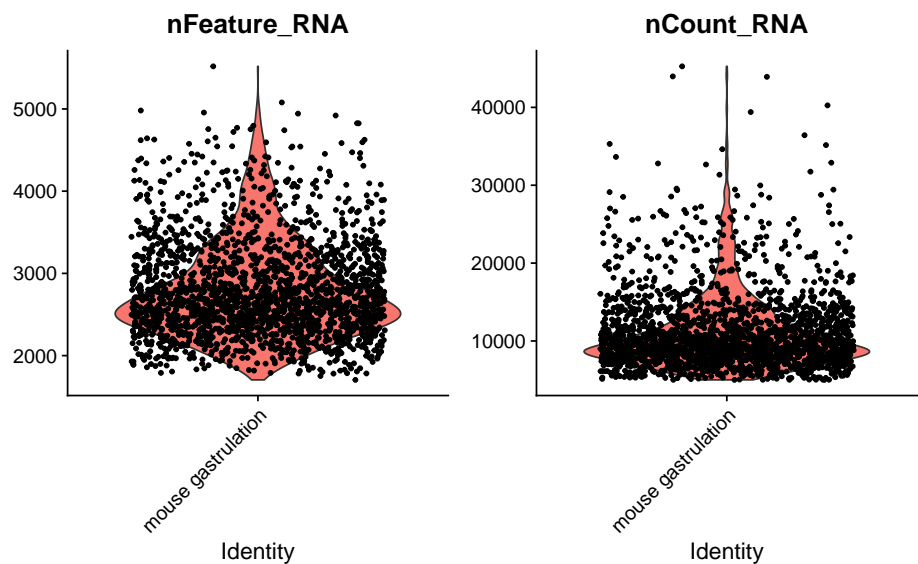
mgsc <- mgsc_subset
```

This is where you are starting from! :) Make sure to download `mgsc_e675.rds` if you haven't already.

2 TASK 1: Pre-processing

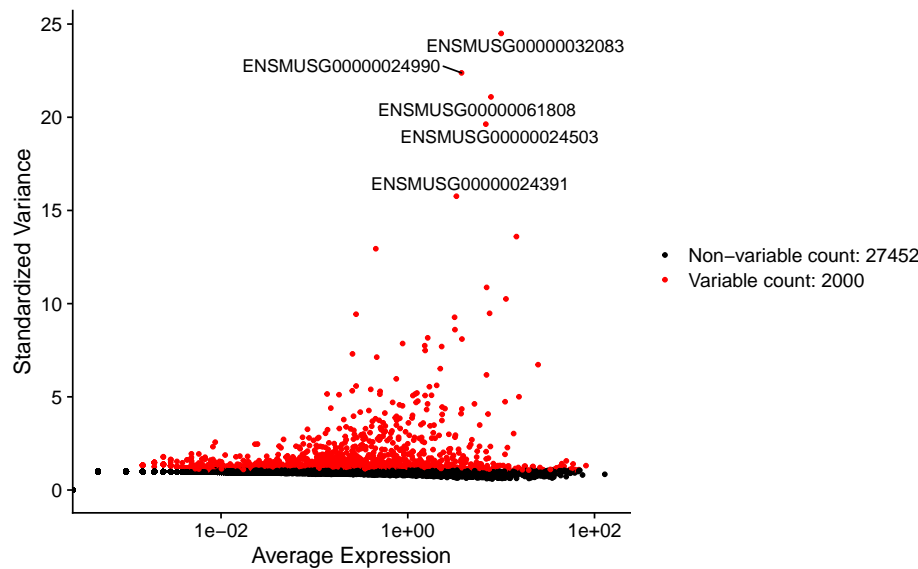
1.1: Let's load the `mgsc_e675` data and take look at the distribution of the features to observe whether there are remaining outliers.

```
## An object of class Seurat
## 29452 features across 2075 samples within 1 assay
## Active assay: RNA (29452 features, 0 variable features)
```



1.2: Now let's plot the most variable features and annotate top5 most variable genes. Scale the data afterwards.

```
## Warning: Using `as.character()` on a quosure is deprecated as of rlang 0.3.0.
## Please use `as_label()` or `as_name()` instead.
## This warning is displayed once per session.
## Warning: Transformation introduced infinite values in continuous x-axis
```



##		mean	variance	variance.standardized
##	ENSMUSG00000032083	10.008193	971.0756	24.49617
##	ENSMUSG00000024990	3.765301	210.4092	22.38176
##	ENSMUSG00000061808	7.784578	546.4217	21.09222
##	ENSMUSG00000024503	6.853976	419.2819	19.63029
##	ENSMUSG00000024391	3.319036	112.9888	15.76248

```
## Centering and scaling data matrix
```

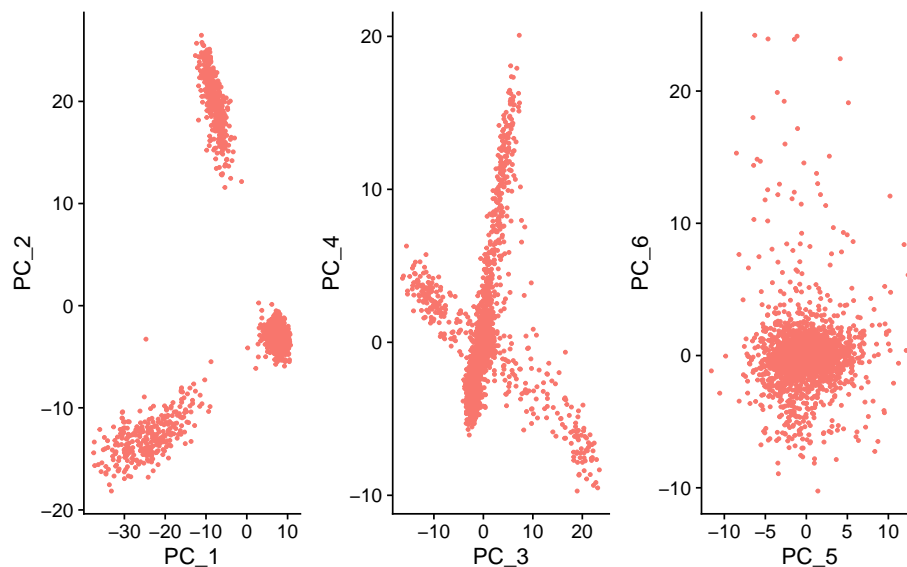
It might be possible that instead of gene symbols, we get our genes with Ensemble GeneIDs. In that case we will need to map to gene symbols first.

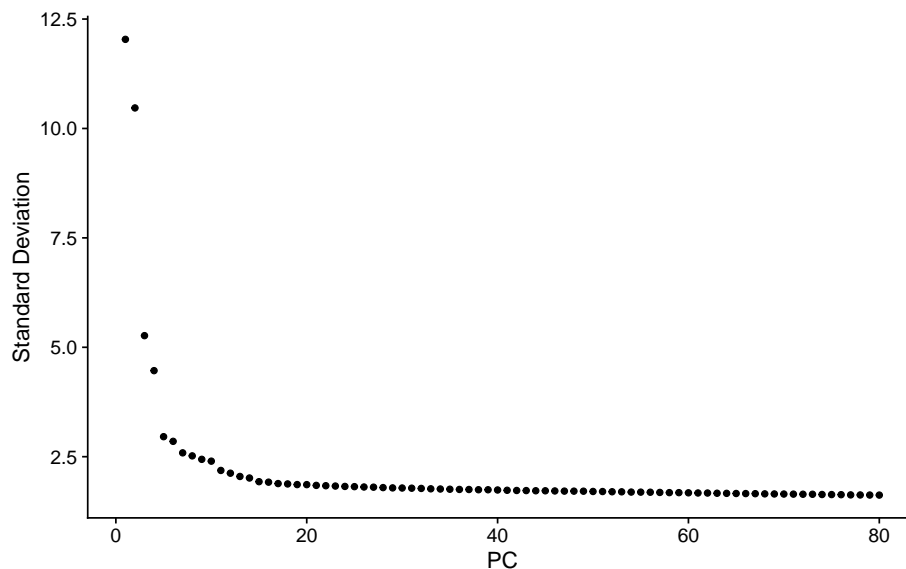
```
library(biomaRt)
ensembl <- useMart("ensembl", dataset="mmusculus_gene_ensembl")
annot<-getBM(c("ensembl_gene_id", "mgi_symbol", "chromosome_name", "strand", "start_position", "end_position")
## Warning: `select_()` is deprecated as of dplyr 0.7.0.
## Please use `select()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
## Warning: `filter_()` is deprecated as of dplyr 0.7.0.
## Please use `filter()` instead.
## See vignette('programming') for more help
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
## Cache found
```

1.3: Print out the gene names of the top10 variable genes. (Hint: you can make use of `match()` function.)

```
## [1] "Apoa1" "Rbp4" "Ttr" "Spink1" "Apom" "ApoE" "Dkk1" "Ctsl"
## [9] "RhoX5" "Gsto1"
```

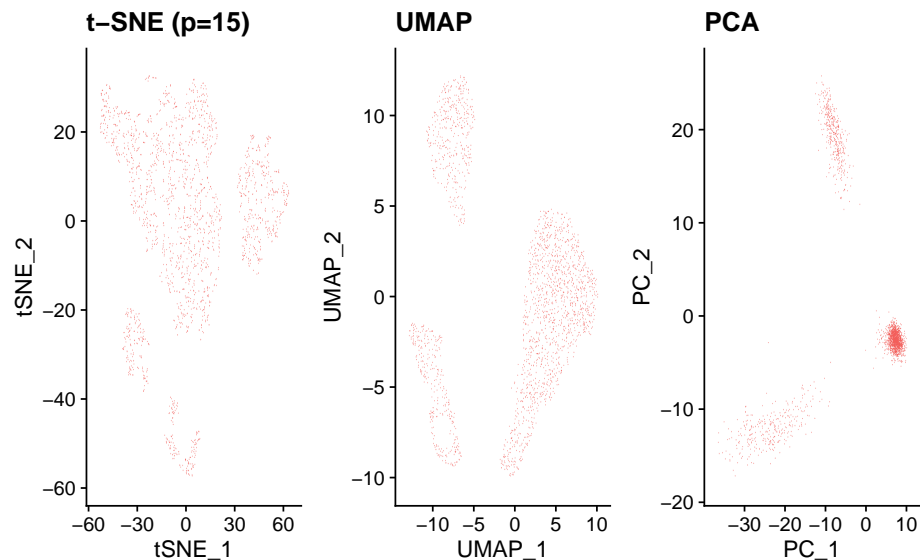
1.4: Apply PCA and determine the dimensionality. Generate the following output: three 2D plots with the first six PCs and print them side by side. i.e. PC1-PC2, PC3-PC4, PC5-PC6.





1.5. Plot the projections of the dataset with three of the dimensionality reduction techniques printed side by side. Use UMAP with `n.neighbors=20`, `min.dist=7` and tSNE with `perplexity=15`. Use first 10 PCs.

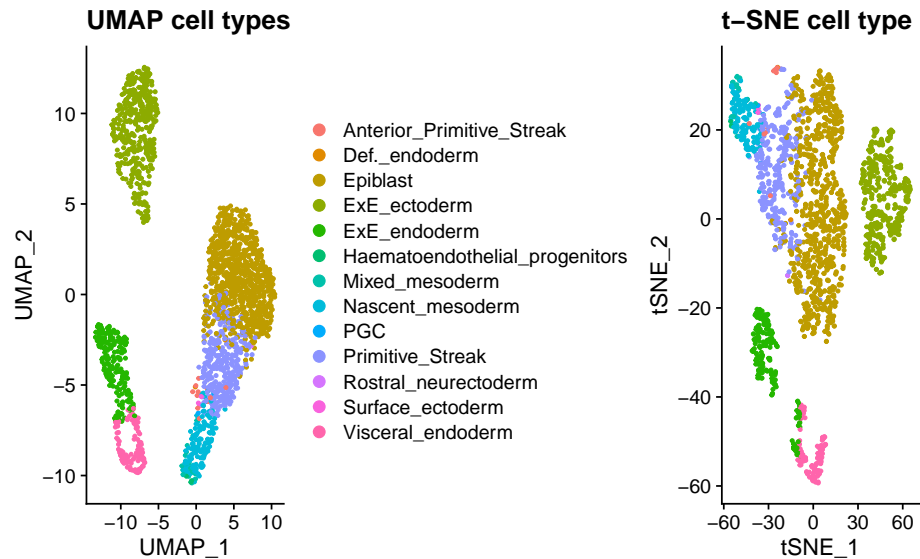
```
## Warning: The default method for RunUMAP has changed from calling Python UMAP via reticulate to the R-native method
## To use Python UMAP via reticulate, set umap.method to 'umap-learn' and metric to 'correlation'
## This message will be shown once per session
## 15:20:24 UMAP embedding parameters a = 0.3208 b = 1.563
## 15:20:24 Read 2075 rows and found 10 numeric columns
## 15:20:24 Using Annoy for neighbor search, n_neighbors = 20
## 15:20:24 Building Annoy index with metric = cosine, n_trees = 50
## 0% 10 20 30 40 50 60 70 80 90 100%
## [----|----|----|----|----|----|----|----|----|----|
## *****|
## 15:20:24 Writing NN index file to temp file /var/folders/l5/1nfd7mk14835td6hgk_71p880000gs/T//RtmpNfw7b3/
## 15:20:24 Searching Annoy index using 1 thread, search_k = 2000
## 15:20:24 Annoy recall = 100%
## 15:20:25 Commencing smooth kNN distance calibration using 1 thread
## 15:20:25 Found 2 connected components, falling back to 'spca' initialization with init_sdev = 1
## 15:20:25 Initializing from PCA
## 15:20:25 PCA: 2 components explained 74.03% variance
## 15:20:25 Commencing optimization for 500 epochs, with 53554 positive edges
## 15:20:28 Optimization finished
```



3 TASK 2: Cluster cells

For our subset of time point E6.75, the original meta-data stores the assigned cell type annotation information.

2.1: How many clusters did the original study identify? Reproduce the following plots colored by annotated cell types. (Hint: You can make use `group.by` argument in `DimPlot` to extract stored cluster IDs.)



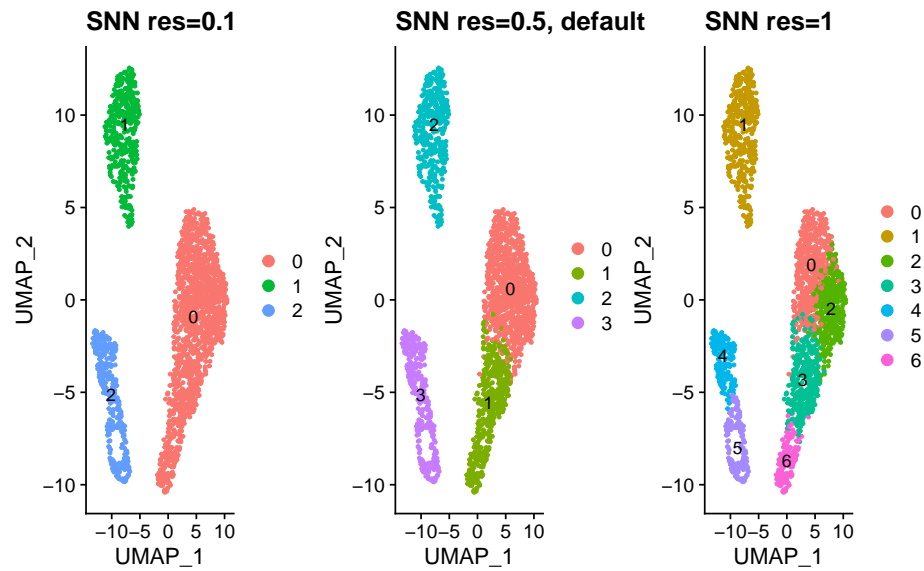
```
## [1] "Epiblast" "Primitive_Streak"
## [3] "ExE_endoderm" "Visceral_endoderm"
## [5] "ExE_ectoderm" "Nascent_mesoderm"
## [7] "Anterior_Primitive_Streak" "PGC"
## [9] "Mixed_mesoderm" "Rostral_neurectoderm"
## [11] "Surface_ectoderm" "Def._endoderm"
```

```
## [13] "Haematoendothelial_progenitors"
```

2.2: Now let's use Seurat's graph-based clustering to identify the clusters and observe whether we can reproduce the above conclusion. Use first 30 PCs and 10 nearest neighbors for resolutions $r=0.1$, 0.5 , 1 .

Hint: The output of FindClusters is saved in `mgsc meta.data$seurat_clusters`. This resets each time clustering is performed. You can use `Idents` function of Seurat to save cluster ids.

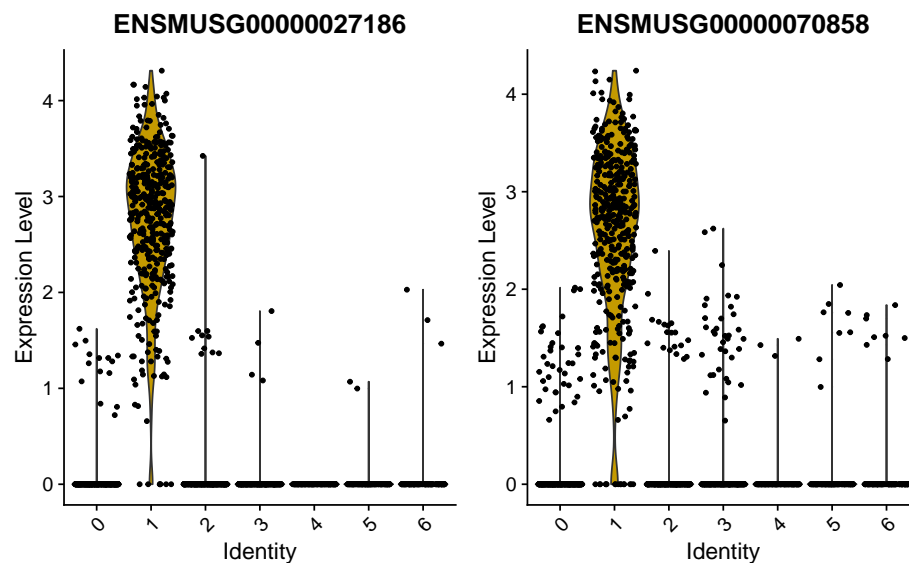
```
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 2075
## Number of edges: 143303
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.8227
## Number of communities: 4
## Elapsed time: 0 seconds
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 2075
## Number of edges: 143303
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.9544
## Number of communities: 3
## Elapsed time: 0 seconds
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 2075
## Number of edges: 143303
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.7164
## Number of communities: 7
## Elapsed time: 0 seconds
```

2.3: Let's find the markers of the cluster 1. Investigate the first two markers and find out gene names of the top 10.

```
## For a more efficient implementation of the Wilcoxon Rank Sum Test,
## (default method for FindMarkers) please install the limma package
## -----
## install.packages('BiocManager')
## BiocManager::install('limma')
## -----
## After installation of limma, Seurat will automatically use the more
## efficient implementation (no further action necessary).
## This message will be shown once per session
```

Note that the original study does not employ *Seurat* but *scan* and *Scanpy* packages, therefore it's expected that we might have slightly different results.

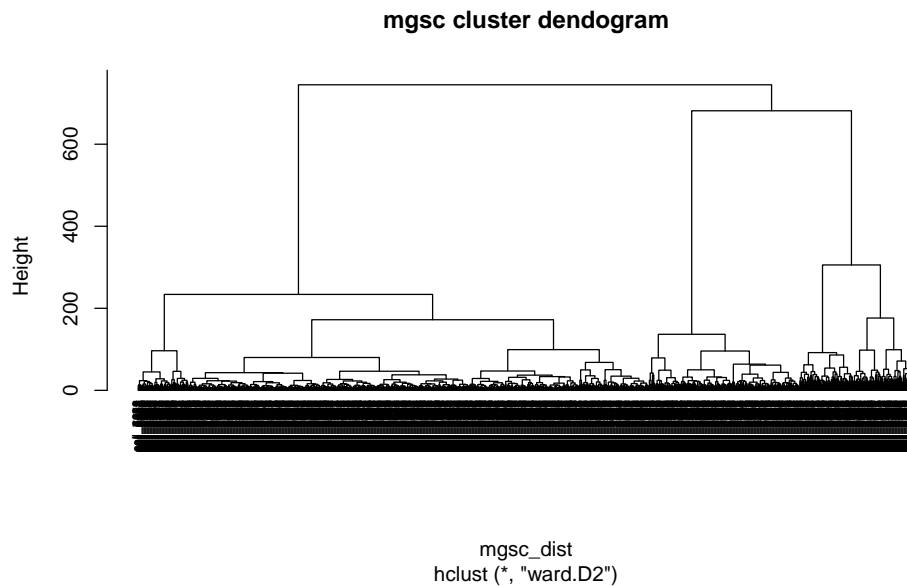


```
## [1] "Elf5" "Gm1673" "Nup62cl" "Tex19.1" "Sl00a6" "Gjb3" "Clbn3"
## [8] "Anxa5" "Dppa4" "Zfp42"
```

4 Homework (Hierarchical Clustering)

Seurat does not support hierarchical clustering too? But we can do it!!

H1: Which elements do we need to perform hierarchical clustering? Try to reproduce the following dendrogram. Use `Euclidian` as distance metric and `ward.D2` as linkage method.



Looking at the dendrogram, we can investigate different numbers of clusters using `cutree` function that cuts the hierarchical clustering tree into given number of clusters.

H2: How about we print different clustering outcomes with $k = 4, 8, 16$ with UMAP?

