Analysis of Microinjected Cells (E14.5) with Seurat (version 3.1.0) Workflow for scRNA-seq data

Bismark Appiah

10 April 2020

```
library(BiocStyle)
library(readr)
library(data.table)
library(dplyr)
library(patchwork)
library(tools)
library(SingleCellExperiment)
library(HDF5Array)
library(mvoutlier)
library(scran)
library(scater)
library(SC3)
library(slingshot)
library(stringr)
library(org.Mm.eg.db)
library(mclust)
library(RColorBrewer)
library(Seurat)
library(umap)
library(latexpdf)
library(cowplot)
```

1. Data preparation

#Loading data matrix

Convert data to “SingleCellExperiment” Class. ## Define DMSO and EPZ conditions

```
AMI_ALL_NEW <- readRDS("AMI_ALL_NEW.RData")

sce <- SingleCellExperiment(list(counts=as.matrix(AMI_ALL_NEW)))
rm(AMI_ALL_NEW)
sce
```

```
## class: SingleCellExperiment
## dim: 34205 728
## metadata(0):
## assays(1): counts
## rownames(34205): 0610007P14Rik 0610009B22Rik ... Zzef1 Zzz3
## rowData names(0):
## colnames(728): aMIDMSOE14RP13_1 aMIDMSOE14RP13_3 ...
##   aMIEPZ_Bismark_6_179 aMIEPZ_Bismark_6_185
```

```
## colData names(0):
## reducedDimNames(0):
## spikeNames(0):
```

```
DM24 <- readRDS("DM24.RData")
EP24 <- readRDS("EP24.RData")

sce$condition <- "NA"
sce$condition[which(colnames(sce) %in% DM24)] <- "DMS24"
sce$condition[which(colnames(sce) %in% EP24)] <- "EPZ24"

table(sce$condition)
```

```
##
## DMS24 EPZ24
##   295   433
```

Add mitochondrial genes and ERCC spike-ins to sce object

There were no ERCC spike-ins used in the experiment.

```
isSpike(sce, "MT") <- rownames(sce)[grep("^(mt)",rownames(sce),invert=FALSE)]
isSpike(sce, "ERCC") <- rownames(sce)[grep("^(ERCC)",rownames(sce),invert=FALSE)]
sce
```

```
## class: SingleCellExperiment
## dim: 34205 728
## metadata(0):
## assays(1): counts
## rownames(34205): 0610007P14Rik 0610009B22Rik ... Zzef1 Zzz3
## rowData names(0):
## colnames(728): aMIDMSOE14RP13_1 aMIDMSOE14RP13_3 ...
##   aMIEPZ_Bismark_6_179 aMIEPZ_Bismark_6_185
## colData names(1): condition
## reducedDimNames(0):
## spikeNames(2): MT ERCC
```

2. Quality control

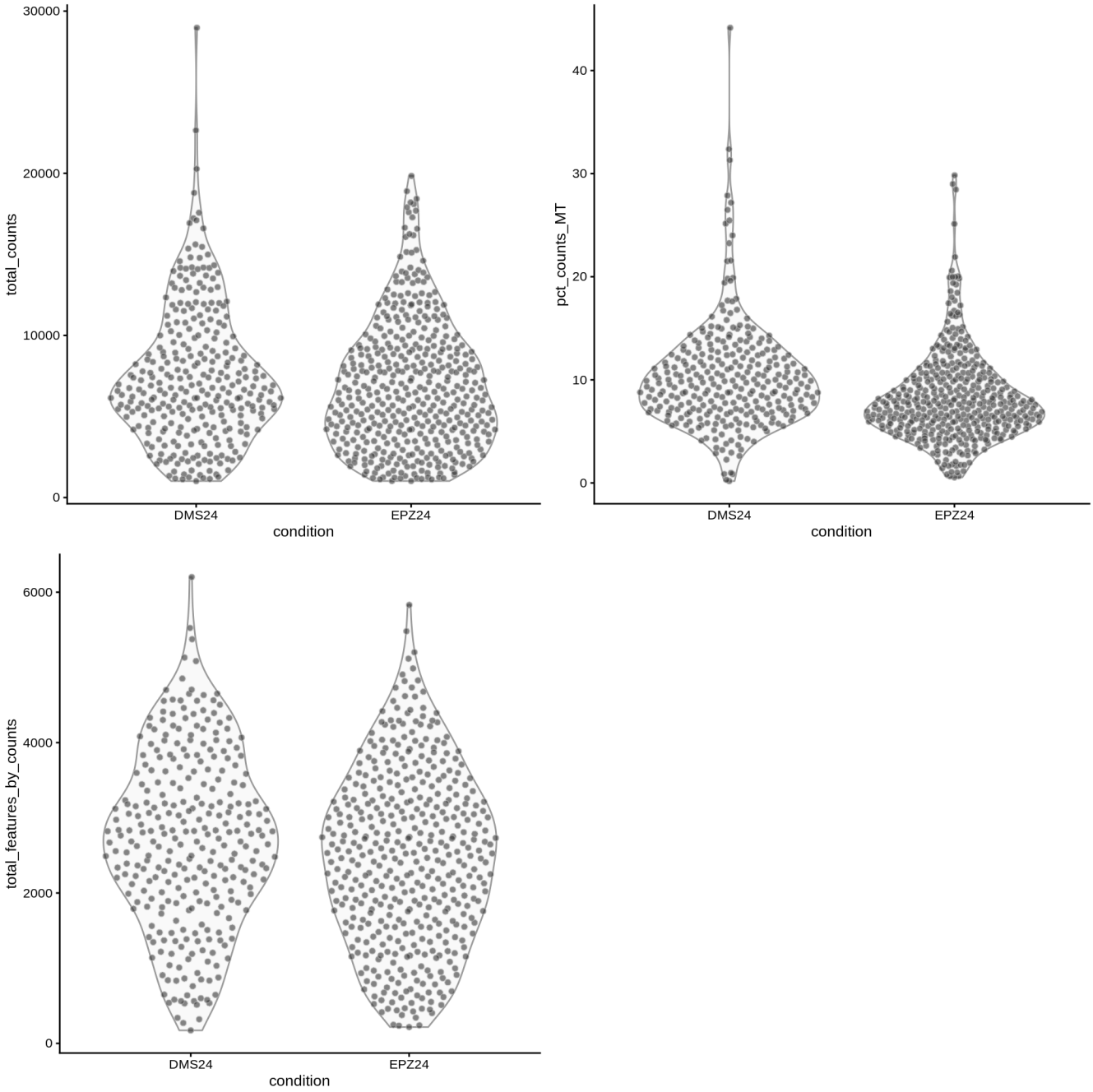
#Remove genes that are not expressed

```
keep_feature <- rowSums(counts(sce) > 0) > 0
sce <- sce[keep_feature,]
```

#Calculate quality metrics
It seems like there were no ERCC spike-ins used.

```
sce <- calculateQCMetrics(sce)
```

```
multiplot(cols=2,
    plotColData(sce, x="condition", y="total_counts"),
    plotColData(sce, x="condition", y="total_features_by_counts"),
    plotColData(sce, x="condition", y="pct_counts_MT")
)
```



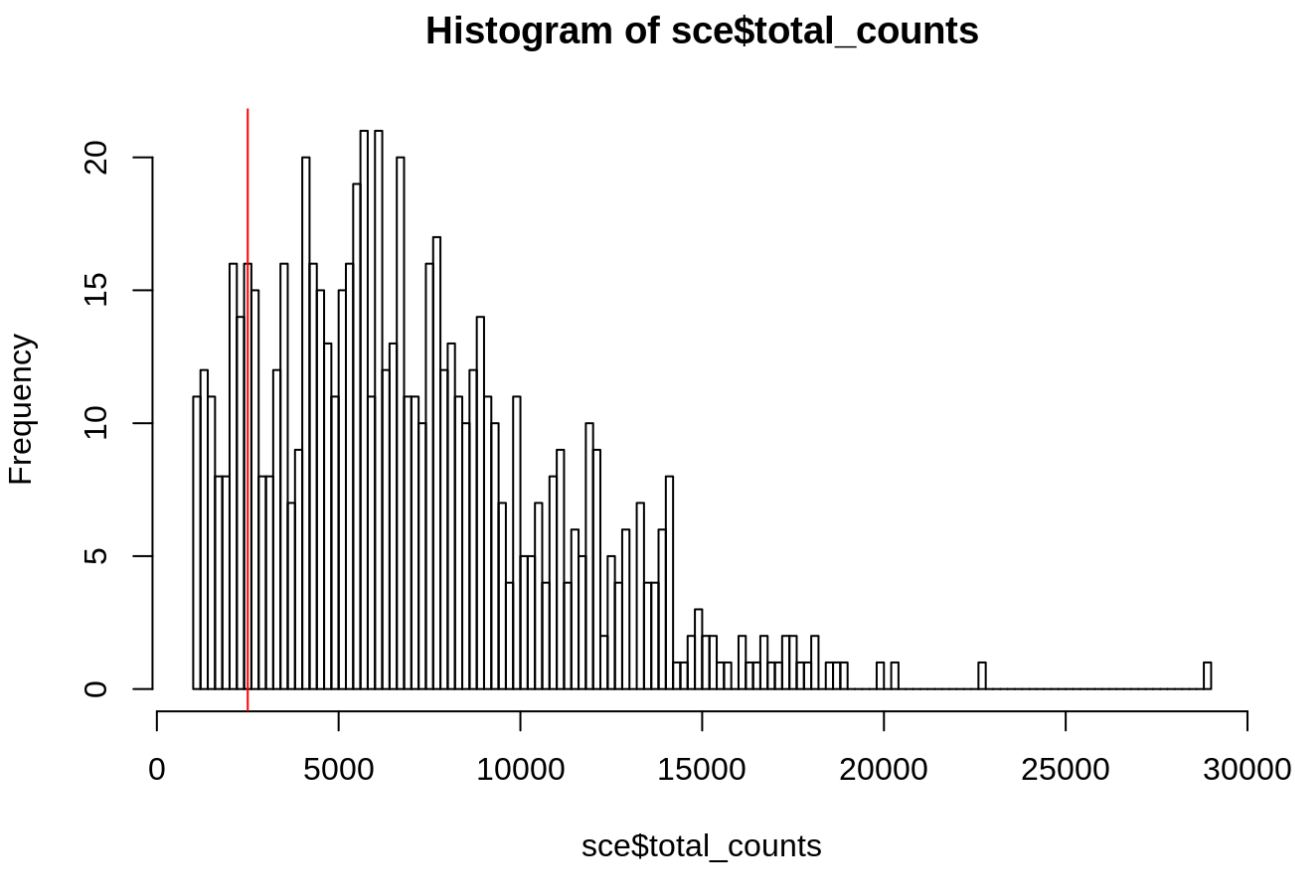
Total counts and distribution of counts for cells in both conditions looks good We can proceed with further analysis

Remove ERCC spike-ins since they were not used.

```
sce <- sce[grep("^(ERCC|Gm|Rik)",row.names(sce),invert=TRUE),]
```

Visualize library sizes

```
hist(
    sce$total_counts,
    breaks = 100
)
abline(v = 2500, col = "red")
```



3. Filter cells by library size > 2500

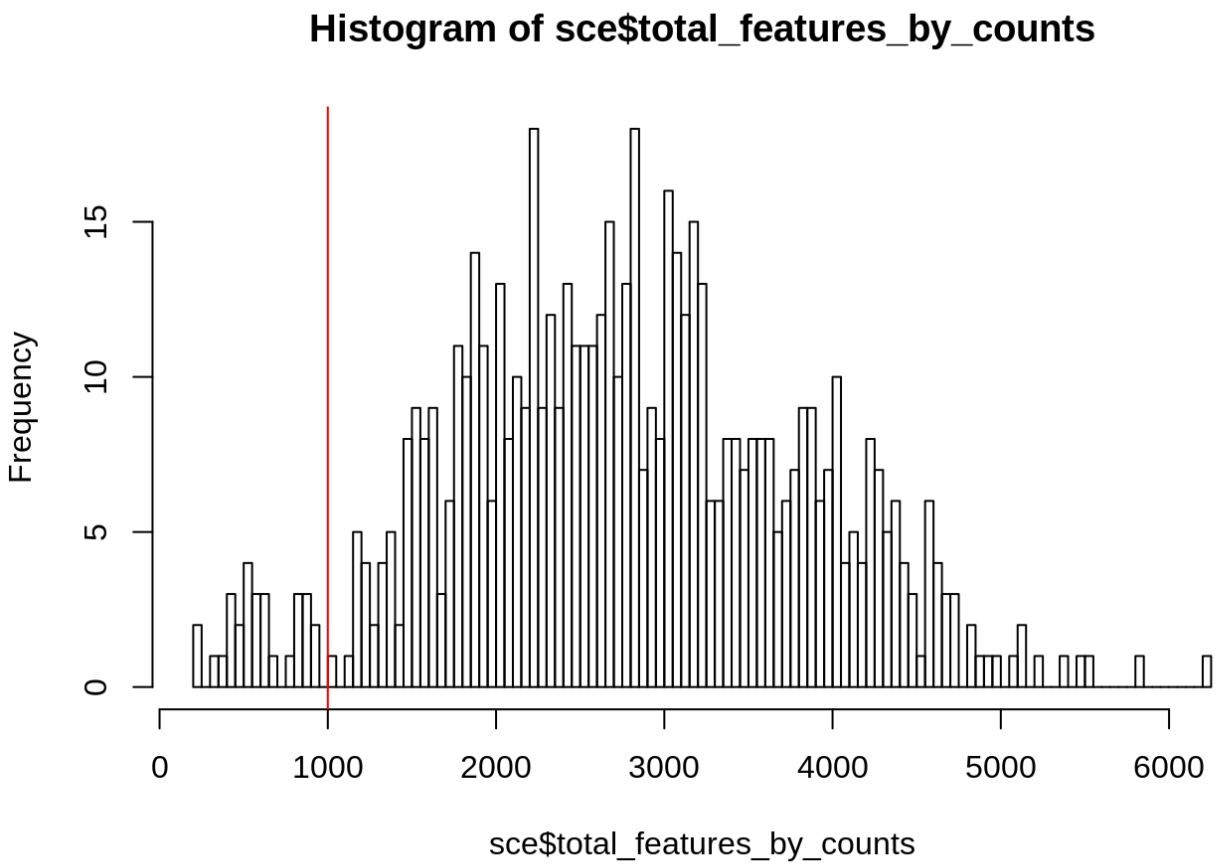
```
filter_by_total_counts <- (sce$total_counts > 2500)
table(filter_by_total_counts)
```

```
## filter_by_total_counts
## FALSE  TRUE
##    89   639
```

```
sce <- sce[,filter_by_total_counts]
```

4. Visualize the total number of unique genes detected in each sample

```
hist(
    sce$total_features_by_counts,
    breaks = 100
)
abline(v = 1000, col = "red")
```



5. Filter cells by expressed features > 1000

```
filter_by_expr_features <- (sce$total_features_by_counts > 1000)
table(filter_by_expr_features)
```

```
## filter_by_expr_features
## FALSE  TRUE
##    29   610
```

```
sce <- sce[,filter_by_expr_features]
```

Cells with outlier values for mitochondrial genes are identified based on some number of MADs from the median value.

```
high.mt <- isOutlier(sce$pct_counts_MT, nmads=3, type="higher", batch=sce$conditio
n)
data.frame(HighMito=sum(high.mt))
```

```
##   HighMito
## 1       21
```

We only retain cells that pass all of the specified criteria. Of course, this involves some assumptions about independence from biology. (For example, don't use the mitochondrial proportions if the number/activity of mitochondria changes between cell types.)

```
discard <- high.mt
data.frame(TotalLost=sum(discard), TotalLeft=sum(!discard))
```

```
##   TotalLost TotalLeft
## 1        21       589
```

We toss out the cells that we consider to be low-quality, and keep the rest.

```
sce <- sce[,!discard]
ncol(sce)
```

```
## [1] 589
```

https://scrnaseq-course.cog.sanger.ac.uk/website/cleaning-the-expression-matrix.html#exprs-qc

```
#plotColData(sce, x = "total_features_by_counts", y = "pct_counts_MT")
```

6. Examining the genes

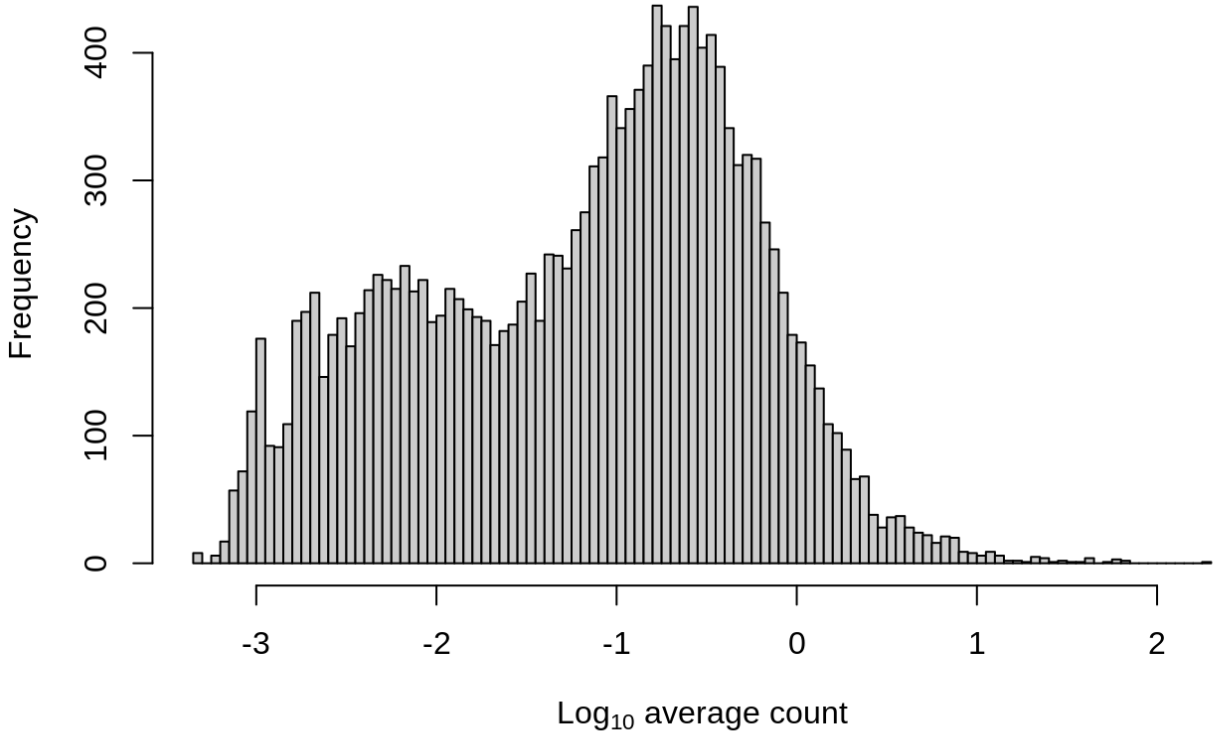
We inspect the distribution of log-mean counts across all genes. The peak represents the bulk of moderately expressed genes while the rectangular component corresponds to lowly expressed genes.

```
ave.counts <- calcAverage(sce)
```

```
## Warning in .get_all_sf_sets(object): spike-in set 'MT' should have its own
## size factors
```

```
## Warning in .get_all_sf_sets(object): spike-in set 'ERCC' should have its
## own size factors
```

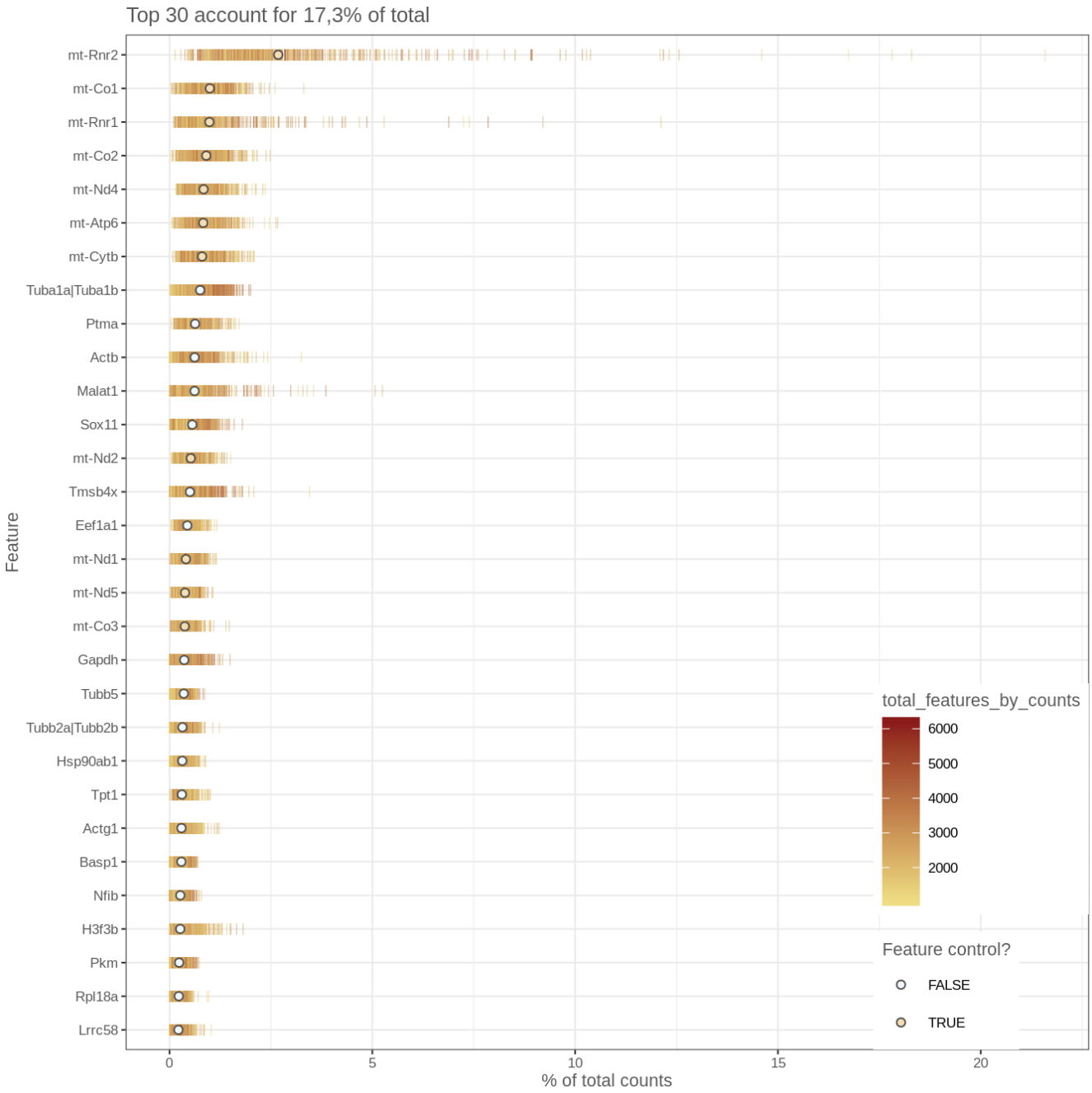
```
hist(log10(ave.counts), breaks=100, main="", col="grey80",
    xlab=expression(Log[10]~"average count"))
```



7. Show top 30 of most highly expressed genes

We also look at the identities of the most highly expressed genes. This should generally be dominated by constitutively expressed transcripts, such as those for ribosomal or mitochondrial proteins. The presence of other classes of features may be cause for concern if they are not consistent with expected biology.

```
plotHighestExprs(sce, n=30)
```



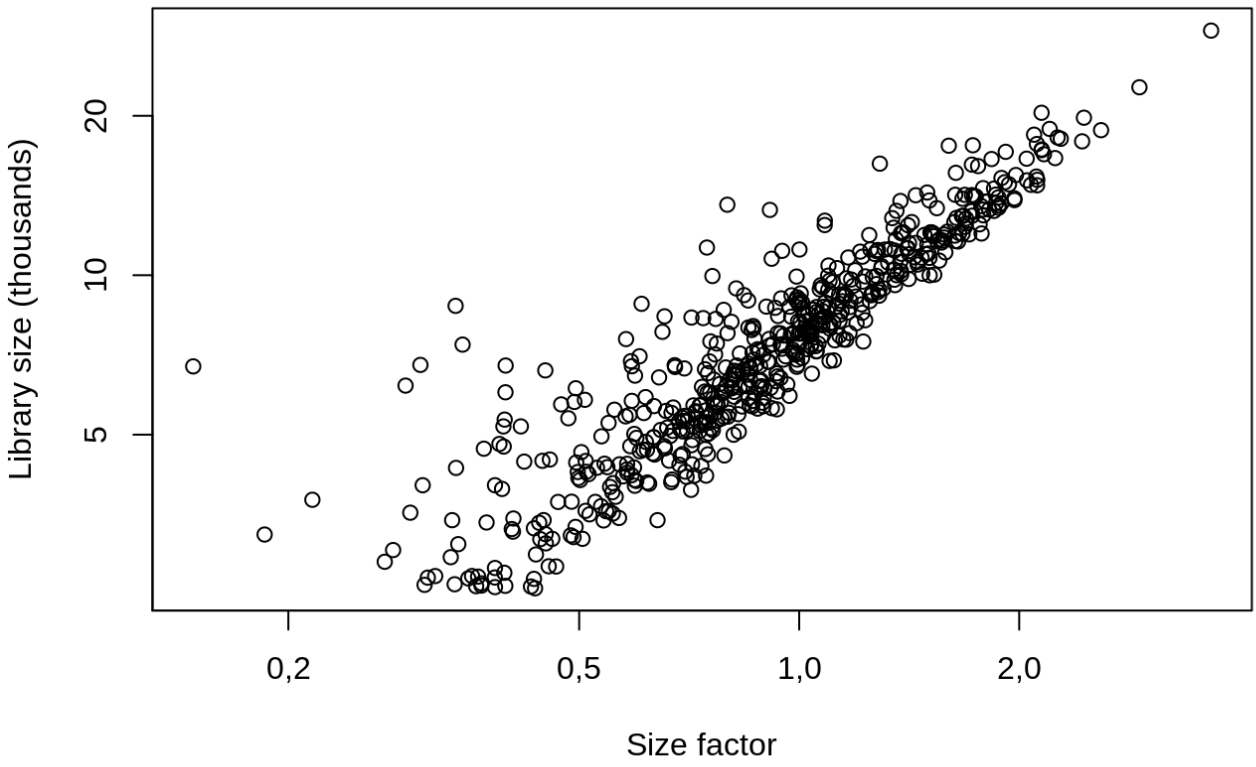

```
clusters <- quickCluster(sce,use.ranks=FALSE)
sce <- computeSumFactors(sce, cluster=clusters)
```

From: Lun et al. (2016) A step-by-step workflow for low-level analysis of single-cell RNA-seq data with Bioconductor.

“In this case, the size factors are tightly correlated with the library sizes for all cells. This suggests that the systematic differences between cells are primarily driven by differences in capture efficiency or sequencing depth. Any DE between cells would yield a non-linear trend between the total count and size factor, and/or increased scatter around the trend. This does not occur here as strong DE is unlikely to exist within a homogeneous population of cells.”

In our case there is also a linear trend, but with some scatter around it.

```
plot(sizeFactors(sce), sce$total_counts/1e3, log="xy",
ylab="Library size (thousands)", xlab="Size factor")
```




```
sce <- scater::normalize(sce)
```

```
## Warning in .get_all_sf_sets(object): spike-in set 'MT' should have its own
## size factors
```

```
## Warning in .get_all_sf_sets(object): spike-in set 'ERCC' should have its
## own size factors
```

```
sce
```

```
## class: SingleCellExperiment
## dim: 17192 589
## metadata(1): log.exprs.offset
## assays(2): counts logcounts
## rownames(17192): 0610007P14Rik 0610009B22Rik ... Zzef1 Zzz3
## rowData names(9): is_feature_control is_feature_control_MT ...
##   total_counts log10_total_counts
## colnames(589): aMIDMSOE14RP13_3 aMIDMSOE14RP13_4 ...
##   aMIEPZ_Bismark_6_155 aMIEPZ_Bismark_6_185
## colData names(46): condition is_cell_control ...
##   pct_counts_in_top_200_features_ERCC
##   pct_counts_in_top_500_features_ERCC
## reducedDimNames(0):
## spikeNames(2): MT ERCC
```

8. Modelling the technical component of variation

We identify HVGs to focus on the genes that are driving heterogeneity across the population of cells. This requires estimation of the variance in expression for each gene, followed by decomposition of the variance into biological and technical components.

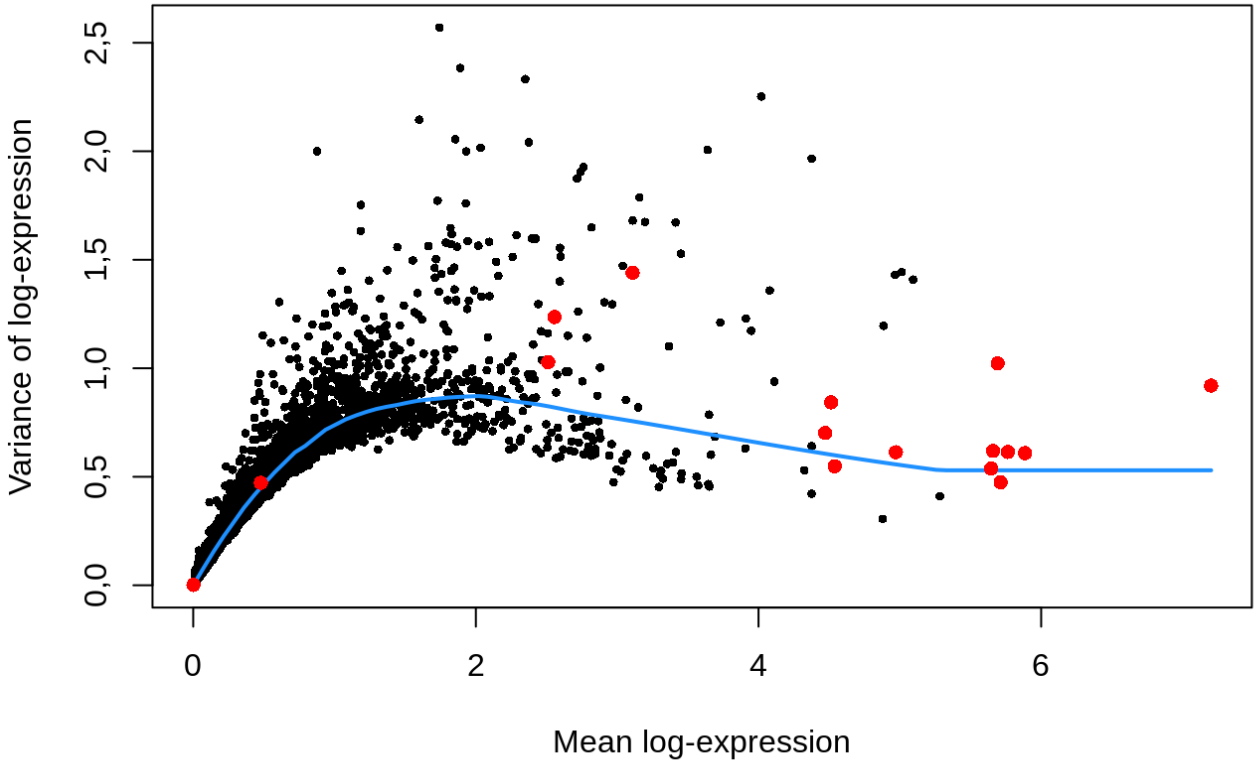
From: Lun et al. (2016) A step-by-step workflow for low-level analysis of single-cell RNA-seq data with Bioconductor.

“HVGs are defined as genes with biological components that are significantly greater than zero at a false discovery rate (FDR) of 5%. These genes are interesting as they drive differences in the expression profiles between cells, and should be prioritized for further investigation. In addition, we only consider a gene to be a HVG if it has a biological component greater than or equal to 0.5. For transformed expression values on the log2 scale, this means that the average difference in true expression between any two cells will be at least 2-fold. (This reasoning assumes that the true log-expression values are Normally distributed with variance of 0.5. The root-mean-square of the difference between two values is treated as the average log2-fold change between cells and is equal to unity.) We rank the results by the biological component to focus on genes with larger biological variability.”

```
var.fit <- trendVar(sce, method="loess", loess.args=list(span=0.05), use.spikes=FA
LSE)
var.out <- decomposeVar(sce, var.fit)
#head(var.out)
```

“We can have a look at the fitted trend. Some tinkering may be required to get a good fit, usually by modifying `span=`.”

```
plot(var.out$mean, var.out$total, pch=16, cex=0.6, xlab="Mean log-expression",
    ylab="Variance of log-expression")
curve(var.fit$trend(x), add=TRUE, col="dodgerblue", lwd=2)
cur.spike <- isSpike(sce)
points(var.out$mean[cur.spike], var.out$total[cur.spike], col="red", pch=16)
```



Extracting highly variable genes (HVG)

I selected 800 of the HVGs extracted for clustering. This helps to focus on the genes that are driving heterogeneity across the population of cells.

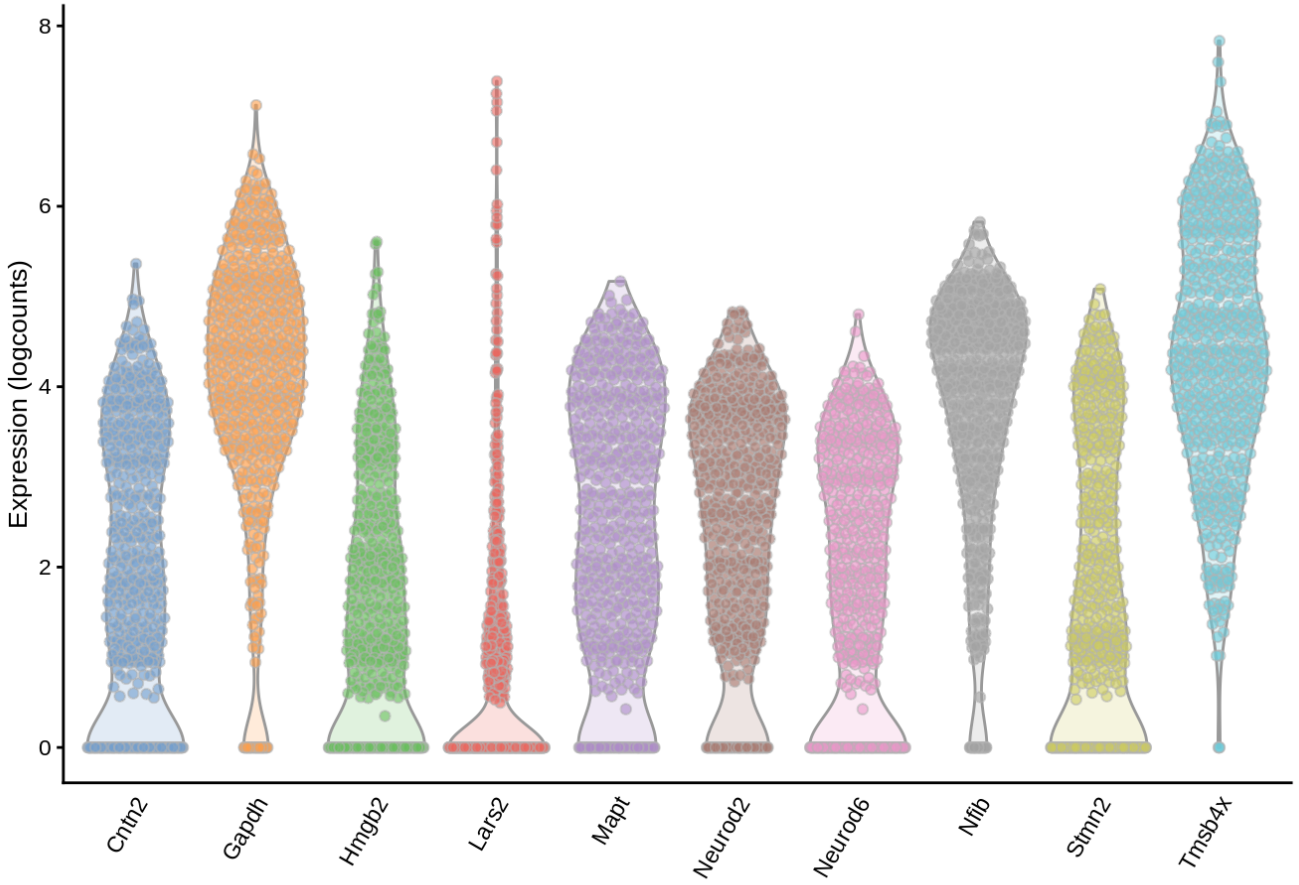
```
hvg.out <- var.out[with(var.out, order(bio,FDR, decreasing = TRUE)),][1:800,]
hvg.out <- hvg.out[order(hvg.out$bio, decreasing = TRUE),]
nrow(hvg.out)
```

```
## [1] 800
```

9. Checking the distribution of expression values for the top HVGs

This ensures that the variance estimate is not being dominated by one or two outlier cells.

```
plotExpression(sce, rownames(hvg.out)[1:10])
```



10. Extract highly variable genes (HVG)

```
sce_hvg <- sce[row.names(hvg.out),]
```

11. Seurat clustering analysis

Convert SingleCellExperiment object into Seurat object.

```
rownames(sce_hvg) <- str_remove(rownames(sce_hvg), "[|]")
```

```
sce_hvg.seurat <- as.Seurat(x = sce_hvg)
```

Scaling data before running PCA.

```
all.genes <- rownames(sce_hvg.seurat)
sce_hvg.seurat <- ScaleData(sce_hvg.seurat, features = all.genes)
```

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

```
sce_hvg.seurat <- sce_hvg.seurat[grep("^(mt)",row.names(sce_hvg.seurat),invert=TRU
E),]
```

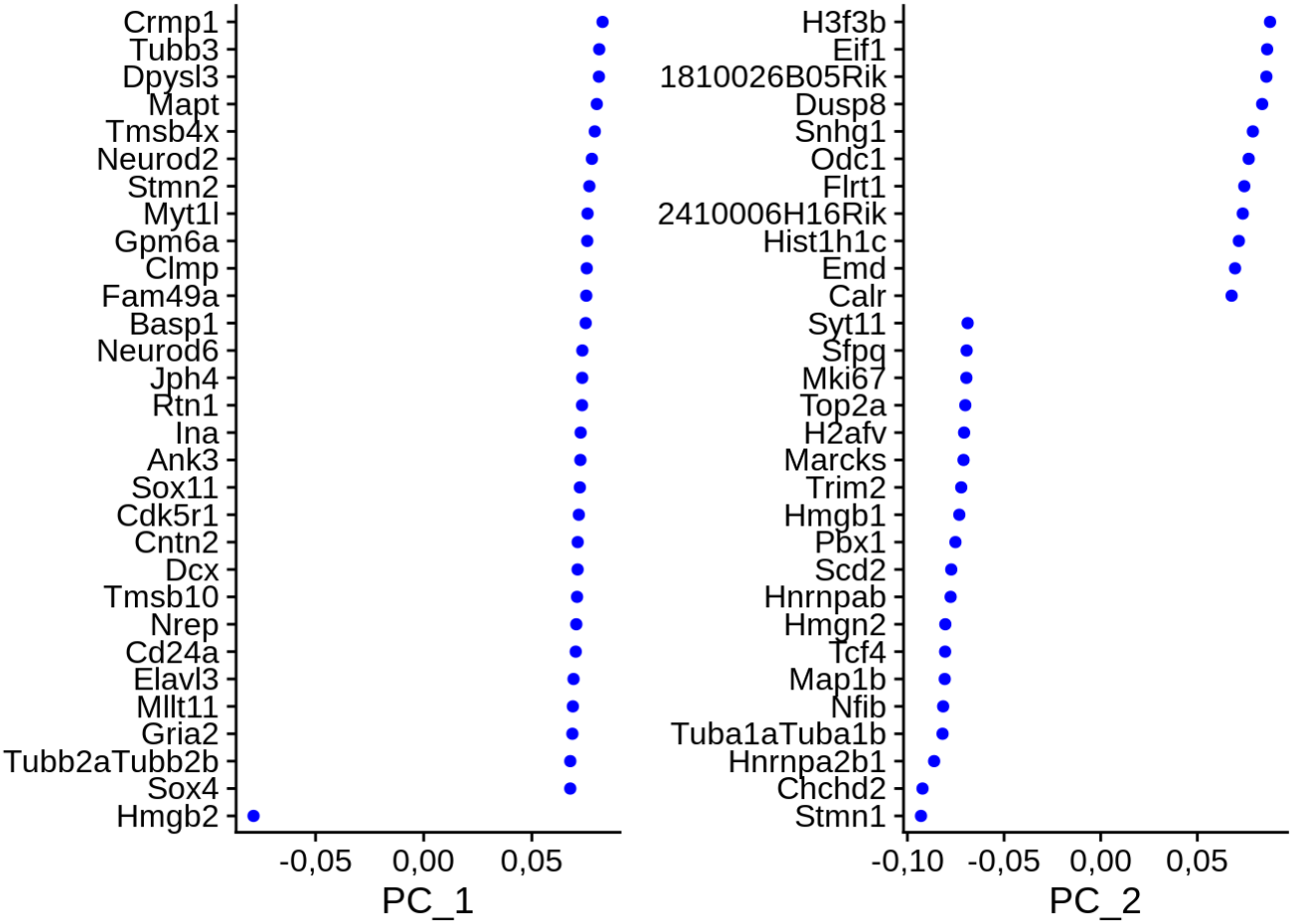
12. Run PCA on Seurat object

```
sce_hvg.seurat <- RunPCA(sce_hvg.seurat, npcs = 60, features = rownames(sce_hvg.se
urat))
```

```
## PC_ 1
## Positive:  Crmp1, Tubb3, Dpysl3, Mapt, Tmsb4x, Neurod2, Stmn2, Myt1l, Gpm6a, Cl
mp
##     Fam49a, Basp1, Neurod6, Jph4, Rtn1, Ina, Ank3, Sox11, Cdk5r1, Cntn2
##     Dcx, Tmsb10, Nrep, Cd24a, Elavl3, Mllt11, Gria2, Tubb2aTubb2b, Sox4, Mpped1
## Negative:  Hmgb2, Smc4, Top2a, Sox2, Cdk1, Mki67, Ccna2, Tpx2, Zfp36l1, Sox9
##     Dek, 2810417H13Rik, Grb10, Tacc3, Smc2, Pax6, Vim, Cdca3, Ube2c, Cks2
##     Birc5, Cenpf, Spc25, Pbk, Nusap1, Ccnb1, Gli3, Aspm, Mdk, H2afz
## PC_ 2
## Positive:  H3f3b, Eif1, 1810026B05Rik, Dusp8, Snhg1, Odc1, Flrt1, 2410006H16Ri
k, Hist1h1c, Emd
##     Calr, Atp1a3, Atxn2l, Cbx8, Gabarap, Pim3, Slc7a5, Crebrf, Foxj3, Mn1
##     Foxo3, Sqstm1, Per1, Sh2d3c, Gnl1, Atf4, Eef2, Ptp4a1, Rybp, Eif2s2
## Negative:  Stmn1, Chchd2, Hnrnpa2b1, Tuba1aTuba1b, Nfib, Map1b, Tcf4, Hmgn2, Hn
rnpab, Scd2
##     Pbx1, Hmgb1, Trim2, Marcks, H2afv, Top2a, Mki67, Sfpq, Syt11, Ddx5
##     Zbtb20, 2610203C20Rik, Ccna2, Dbi, Tpx2, Cks2, Tubb4b, H2afx, Cdca3, Mir99a
hg
## PC_ 3
## Positive:  RP23-271M19.2, RP23-341H2.4, Kcnq1ot1, Hsp90aa1, AI838599, Zic1, Mm
e, Mapkbp1, Stap1, A230046K03Rik
##     Xist, Klhl32, A430073D23Rik, Syt11, Tmem220, Cmah, Cadm1, Ildr2, Grk4, 9930
111J21Rik19930111J21Rik2
##     RP24-439I22.3, Rbm28, Trp53inp1, Rps21, Hsph1, Reln, Vcan, Ccdc36, Kctd12,
Malat1
## Negative:  Gapdh, Tpi1, Eno1, Pkm, Mif, Rps19, Ftl1, Rps3, Pgk1, Eif5a
##     Rps5, Rpl18, Bsg, Eef2, Rpsa, Aldoa, Rpl28, Rpl10-ps3, Eno1b, Ptov1
##     Pgk1-rs7, Gpi1, Banf1, Ddit4, Gabarap, Rpl12, Shmt2, Lrrc58, Cdk4, Tecr
## PC_ 4
## Positive:  Actg1, Actb, Lrrc58, Pou3f1, Nnat, Mn1, Dynll1, Hmgb2, Sfpq, Eif1b
##     Hnrnpa2b1, 9130024F11Rik, Tagln3, Igfbpl1, Pim3, Marcksl1, 1810026B05Rik, C
bx8, Ngfrap1, Tubb2aTubb2b
##     Gng3, Sh3bgrl, Abracl, Sbk1, Smarcc2, Dcx, Slc25a5, Ube2c, Hmgb1, Cdc20
## Negative:  Fam162a, Hk2, Aldoa, Bnip3, Ddit4, Ldha, Higd1a, Pfkp, Pgam1, Vegfa
##     Pgk1, Slc2a3, Gpi1, P4hb, Nxph4, Malat1, Stc2, Pfkfb3, Nfia, Hspa5
##     Igf1r, Slc2a1, Bsg, Ppp3ca, Gys1, Ero1l, Ier3, App, Bnip3l, Plod2
## PC_ 5
## Positive:  Lhx2, Rnd2, Sema3c, Insm1, Nrp1, Sstr2, Elavl4, Igfbpl1, Eomes, Trim
2
##     Unc5d, Nhlh1, Fzd1, Epha3, Ctnnb1, Pou3f3, Neurog2, Sorbs2, Rasgef1b, Pou3f
2
##     Sox11, Elavl2, Sema6d, Neurod1, 2610203C20Rik, Tcf4, Meis2, Rcor2, Pantr1,
Robo2
## Negative:  Cdh13, Camk2b, Grin2b, Kif26a, Runx1t1, Ptprz1, Mef2c, Thra, Tubb4b,
Mapt
##     Ncam1, Hivep2, Ldb2, Dab1, Fezf2, Kpna2, Ina, Birc5, Bach2, Cadm2
##     Stmn4, Cdca3, Cenpf, Dusp8, Adcy1, Sla, Ckap2l, Map1lc3a, Aspm, Ntm
```

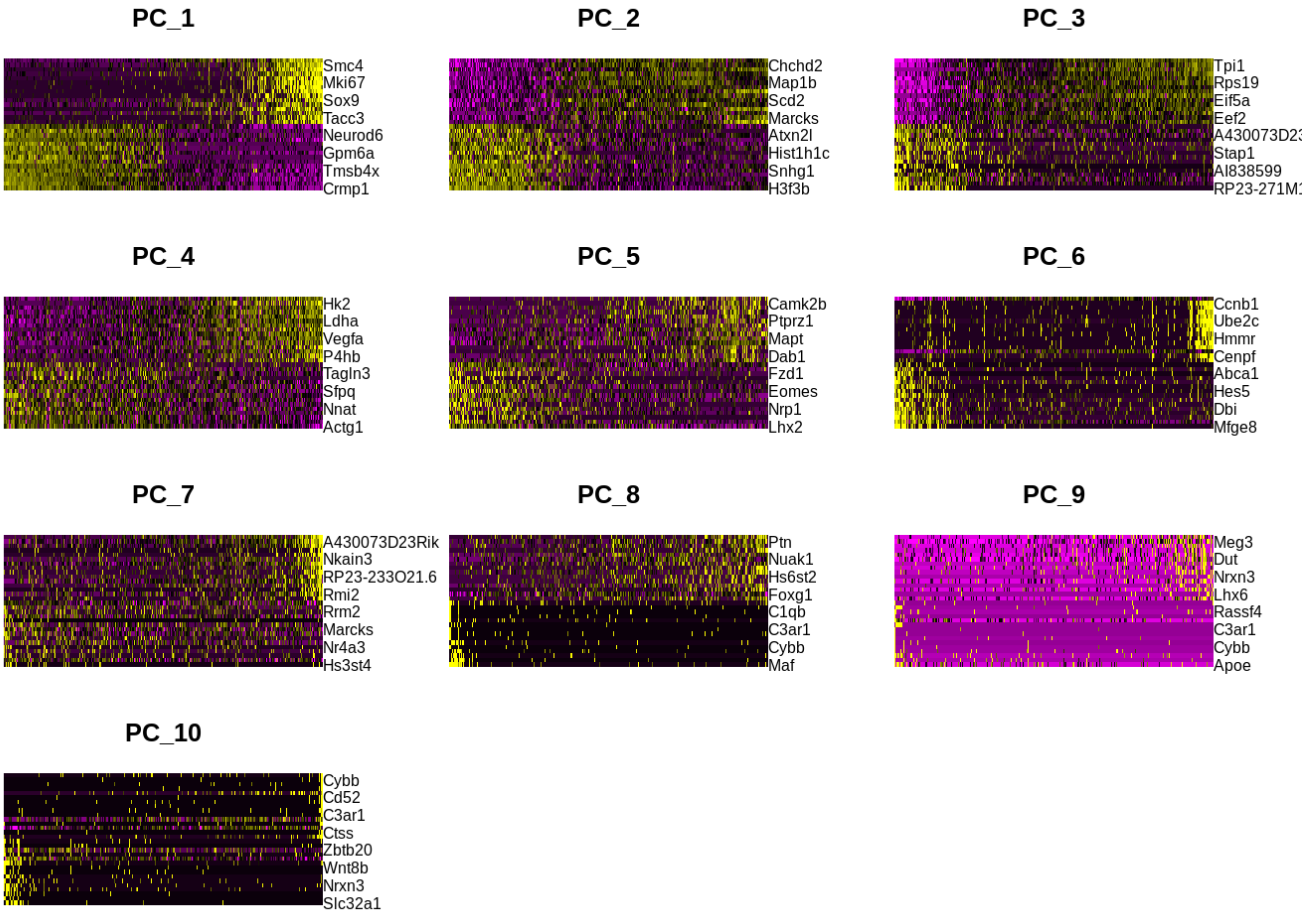
13. Inspecting first two principal components

```
VizDimLoadings(sce_hvg.seurat, dims = 1:2, reduction = "pca")
```



14. Showing heatmap of first 10 principal components

```
DimHeatmap(sce_hvg.seurat, dims = 1:10, cells = 500, balanced = TRUE)
```



Heatmap allows to see some structure in the data. It suggest that the first 8 PCs include some structure. These can then be used for the cluster analysis.

15. Determine the dimensionality of the dataset

This part is based on: https://satijalab.org/seurat/v3.0/pbmc3k_tutorial.html

"To overcome the extensive technical noise in any single feature for scRNA-seq data, Seurat clusters cells based on their PCA scores, with each PC essentially representing a ‘metafeature’ that combines information across a correlated feature set. The top principal components therefore represent a robust compression of the dataset. However, how many components should we choose to include? 10? 20? 100?

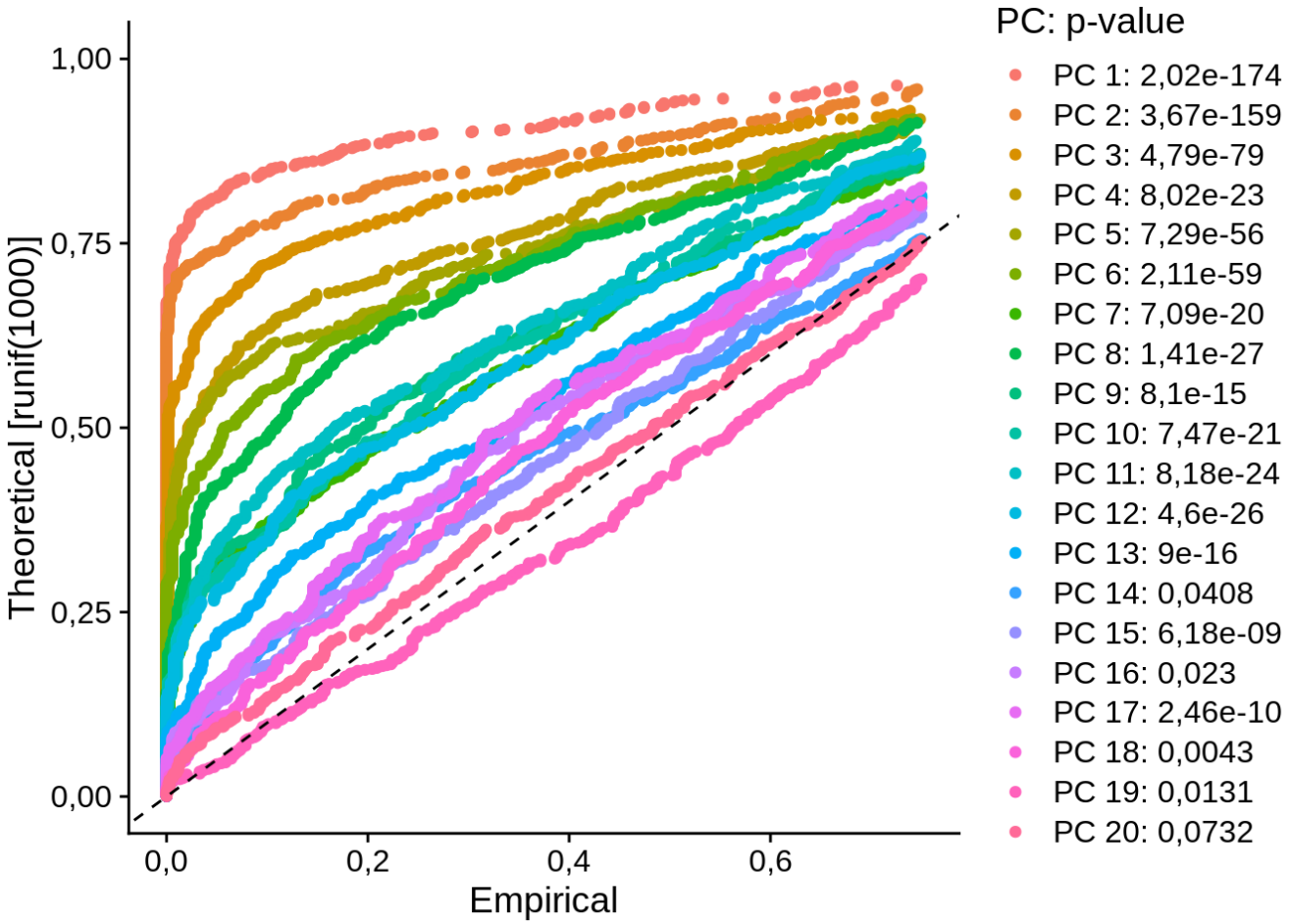
In Macosko et al, we implemented a resampling test inspired by the JackStraw procedure. We randomly permute a subset of the data (1% by default) and rerun PCA, constructing a ‘null distribution’ of feature scores, and repeat this procedure. We identify ‘significant’ PCs as those who have a strong enrichment of low p-value features."

```
sce_hvg.seurat <- JackStraw(sce_hvg.seurat, num.replicate = 100, dims = 20)
sce_hvg.seurat <- ScoreJackStraw(sce_hvg.seurat, dims = 1:20)
```

Plot Supplemntary Figure 6B, top

```
JackStrawPlot(sce_hvg.seurat, dims = 1:20, xmax = 0.75, ymax = 1.0)
```

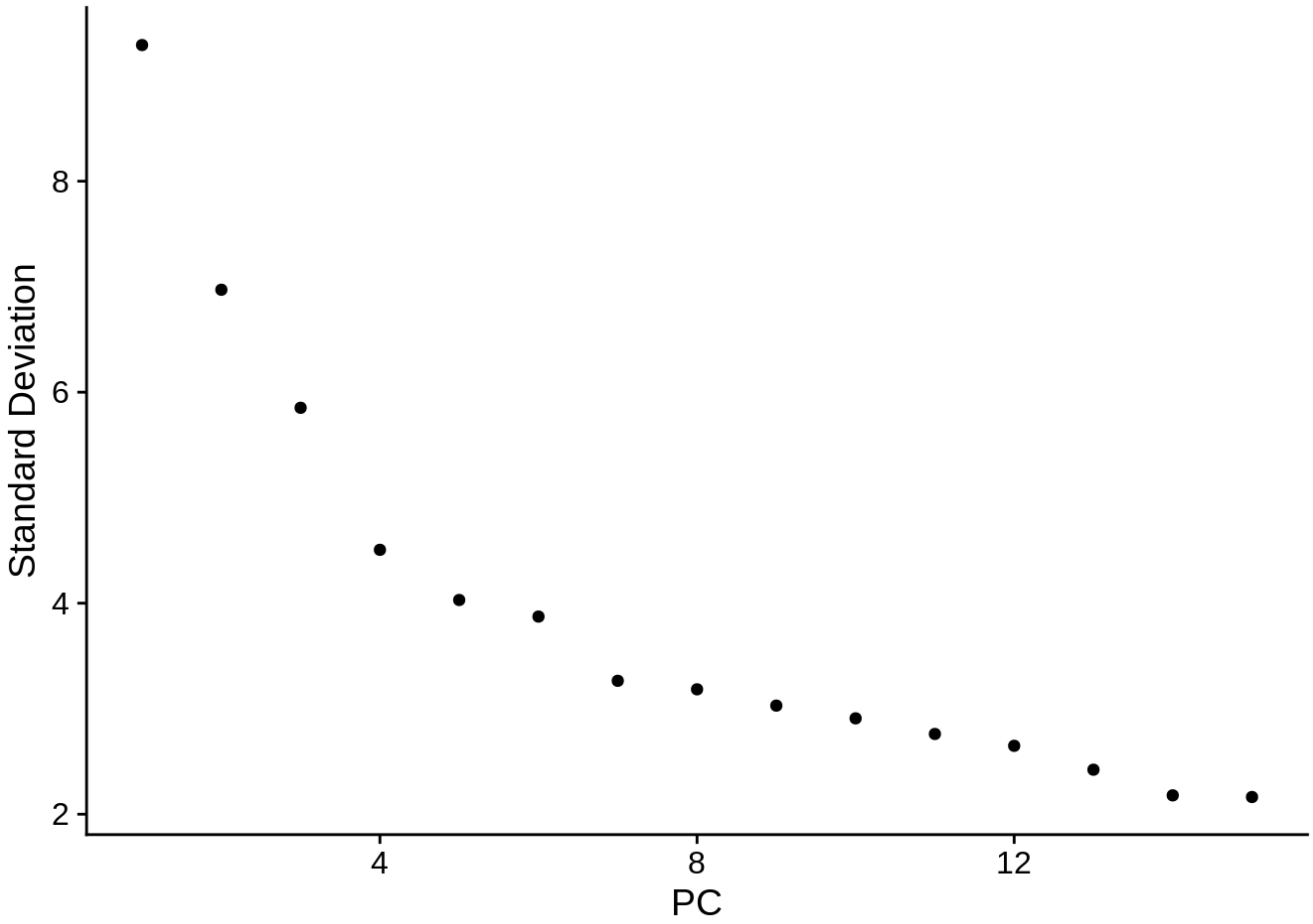
```
## Warning: Removed 2278 rows containing missing values (geom_point).
```



The drop of the p-value at PC8 suggest to use the first 8 dimensions for clustering.

Plot Supplemntary Figure 6B, bottom

```
ElbowPlot(sce_hvg.seurat, ndims = 15, reduction = "pca")
```



The more approximate elbow plot also suggests to use 8 PCs.

Cluster evaluation

Run Seurat clustering with varying resolution parameters to assess cluster stability. Parameter is varied between 0.4 to 1.2 in steps of 0.1. The resulting clusterings are then compared to each other using the adjusted rand index (ARI).

Compute average silhouette width for resolution parameter.

16. Cluster the cells

Using a resolution of 0.4 seems to be the most stable version according to the average silhouette and ARI.

```
sce_hvg.seurat <- FindNeighbors(sce_hvg.seurat, dims = 1:8)
```

```
## Computing nearest neighbor graph
```

```
## Computing SNN
```

```
sce_hvg.seurat <- FindClusters(sce_hvg.seurat, resolution = 0.4, algorithm = 1)
```

```
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 589
## Number of edges: 15645
##
```

```
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0,8643
## Number of communities: 5
## Elapsed time: 0 seconds
```

```
#change resolution to 1.2 for Sup. Figure 6D
#sce_hvg.seurat <- FindClusters(sce_hvg.seurat, resolution = 1.2, algorithm = 1)
#################

# Run this instead of the code above for Supplementary Figure 6 plots
#sce_hvg.seurat <- FindNeighbors(sce_hvg.seurat, dims = 1:10)
#sce_hvg.seurat <- FindClusters(sce_hvg.seurat, resolution = 0.4, algorithm = 1)
```

```
head(Idents(sce_hvg.seurat), 10)
```

```
##  aMIDMSOE14RP13_3  aMIDMSOE14RP13_4  aMIDMSOE14RP13_5  aMIDMSOE14RP13_7
##                 3                 0                 1                 1
##  aMIDMSOE14RP13_8 aMIDMSOE14RP13_10 aMIDMSOE14RP13_11 aMIDMSOE14RP13_12
##                 1                 0                 3                 3
## aMIDMSOE14RP13_14 aMIDMSOE14RP13_15
##                 1                 0
## Levels: 0 1 2 3 4
```

17. Run UMAP for visualization in two dimensions

```
sce_hvg.seurat <- RunUMAP(sce_hvg.seurat, dims = 1:8)
```

```
## Warning: The default method for RunUMAP has changed from calling Python UMAP vi
a reticulate to the R-native UWOT using the cosine metric
## To use Python UMAP via reticulate, set umap.method to 'umap-learn' and metric t
o 'correlation'
## This message will be shown once per session
```

```
## 19:30:00 UMAP embedding parameters a = 0,9922 b = 1,112
```

```
## 19:30:00 Read 589 rows and found 8 numeric columns
```

```
## 19:30:00 Using Annoy for neighbor search, n_neighbors = 30
```

```
## 19:30:00 Building Annoy index with metric = cosine, n_trees = 50
```

```
## 0%   10   20   30   40   50   60   70   80   90   100%
```

```
## [----|----|----|----|----|----|----|----|----|----|
```

```
## ************************************************|
## 19:30:00 Writing NN index file to temp file /tmp/RtmpFTvbuf/file1f143250d209
## 19:30:00 Searching Annoy index using 1 thread, search_k = 3000
## 19:30:00 Annoy recall = 100%
## 19:30:01 Commencing smooth kNN distance calibration using 1 thread
## 19:30:02 Initializing from normalized Laplacian + noise
## 19:30:02 Commencing optimization for 500 epochs, with 21130 positive edges
## 19:30:03 Optimization finished
```

```
colnames(sce_hvg.seurat[[]])
```

```
##  [1] "condition"
##  [2] "is_cell_control"
##  [3] "total_features_by_counts"
##  [4] "log10_total_features_by_counts"
##  [5] "total_counts"
##  [6] "log10_total_counts"
##  [7] "pct_counts_in_top_50_features"
##  [8] "pct_counts_in_top_100_features"
##  [9] "pct_counts_in_top_200_features"
## [10] "pct_counts_in_top_500_features"
## [11] "total_features_by_counts_endogenous"
## [12] "log10_total_features_by_counts_endogenous"
## [13] "total_counts_endogenous"
## [14] "log10_total_counts_endogenous"
## [15] "pct_counts_endogenous"
## [16] "pct_counts_in_top_50_features_endogenous"
## [17] "pct_counts_in_top_100_features_endogenous"
## [18] "pct_counts_in_top_200_features_endogenous"
## [19] "pct_counts_in_top_500_features_endogenous"
## [20] "total_features_by_counts_feature_control"
## [21] "log10_total_features_by_counts_feature_control"
## [22] "total_counts_feature_control"
## [23] "log10_total_counts_feature_control"
## [24] "pct_counts_feature_control"
## [25] "pct_counts_in_top_50_features_feature_control"
## [26] "pct_counts_in_top_100_features_feature_control"
## [27] "pct_counts_in_top_200_features_feature_control"
## [28] "pct_counts_in_top_500_features_feature_control"
## [29] "total_features_by_counts_MT"
## [30] "log10_total_features_by_counts_MT"
## [31] "total_counts_MT"
## [32] "log10_total_counts_MT"
## [33] "pct_counts_MT"
## [34] "pct_counts_in_top_50_features_MT"
## [35] "pct_counts_in_top_100_features_MT"
## [36] "pct_counts_in_top_200_features_MT"
## [37] "pct_counts_in_top_500_features_MT"
## [38] "total_features_by_counts_ERCC"
## [39] "log10_total_features_by_counts_ERCC"
## [40] "total_counts_ERCC"
## [41] "log10_total_counts_ERCC"
## [42] "pct_counts_ERCC"
## [43] "pct_counts_in_top_50_features_ERCC"
```

```
## [44] "pct_counts_in_top_100_features_ERCC"
## [45] "pct_counts_in_top_200_features_ERCC"
## [46] "pct_counts_in_top_500_features_ERCC"
## [47] "nCount_RNA"
## [48] "nFeature_RNA"
## [49] "RNA_snn_res.0,4"
## [50] "seurat_clusters"
```

```
Idents(object = sce_hvg.seurat) <- 'seurat_clusters'
levels(sce_hvg.seurat)
```

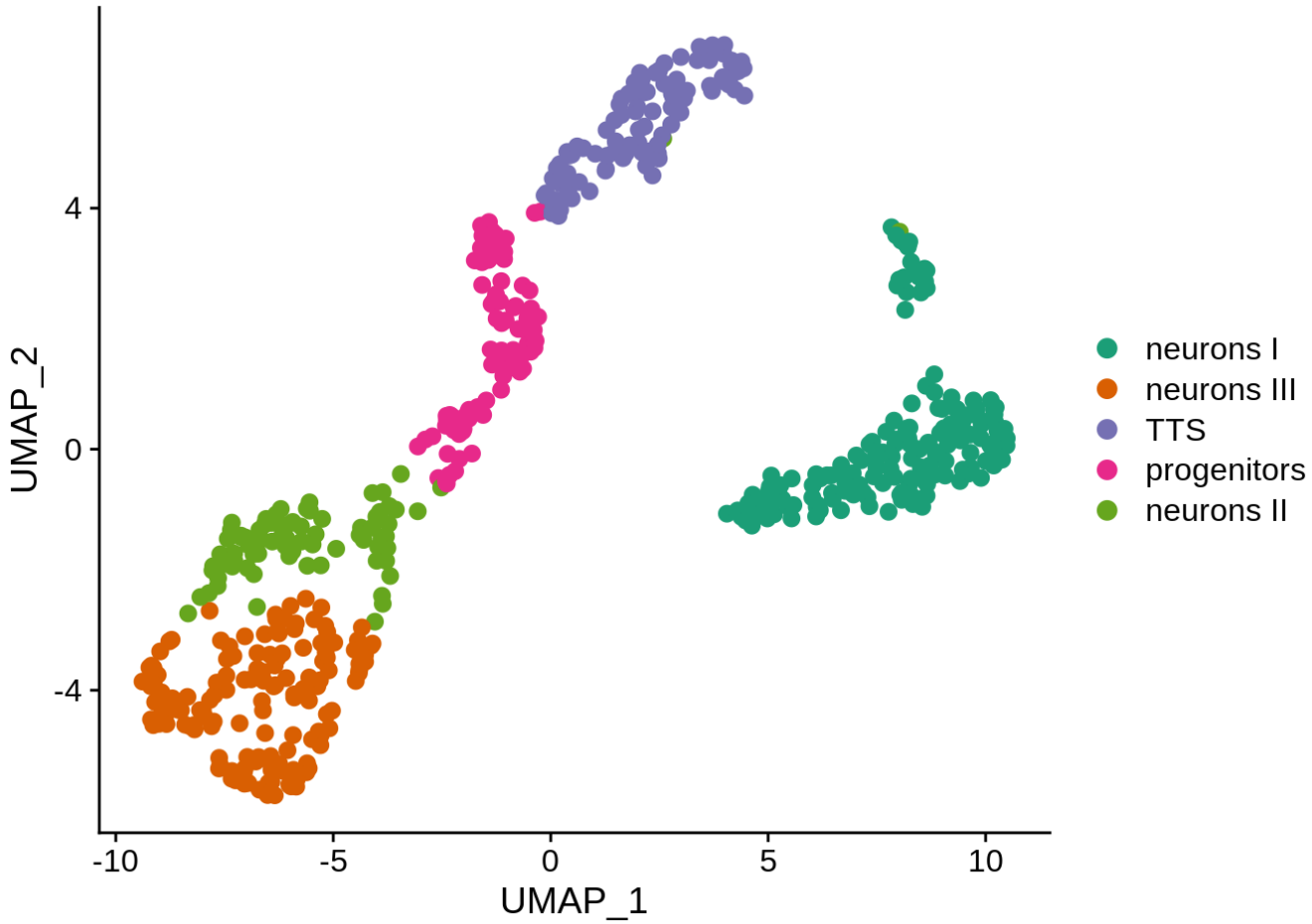
```
## [1] "0" "1" "2" "3" "4"
```

```
current.cluster.ids <- c(0, 1, 2, 3, 4)
t <- brewer.pal(n = 6,name = "Dark2") # Assigning color code for clusters
new.cluster.ids <- c("neurons I","neurons III","TTS", "progenitors","neurons II")
names(new.cluster.ids) <- levels(sce_hvg.seurat)
sce_hvg.seurat <- RenameIdents(sce_hvg.seurat, new.cluster.ids)
########################

## Run this instead of the code above for Supplementary Figure 6 plots
#colnames(sce_hvg.seurat[[]])
#Idents(object = sce_hvg.seurat) <- 'seurat_clusters'
#levels(sce_hvg.seurat)
#sce_hvg.seurat <- RunUMAP(sce_hvg.seurat, dims = 1:10)
#t <- brewer.pal(n = 6,name = "Dark2") # Assigning color code for clusters
#current.cluster.ids <- c(0, 1, 2, 3, 4,5)
#new.cluster.ids <- c("NI","NIII","TTS", "progenitors","NII", "Maf/Sst+")
#names(new.cluster.ids) <- levels(sce_hvg.seurat)
#sce_hvg.seurat <- RenameIdents(sce_hvg.seurat, new.cluster.ids)
```

18. Plot Figure 4 B, Appiah et al. | Note: Order of labels may be different from final figure in manuscript

```
DimPlot(sce_hvg.seurat, reduction = "umap", pt.size = 2.5, cols = t) # Run the sam
e script for Sup. Fig. 2C
```

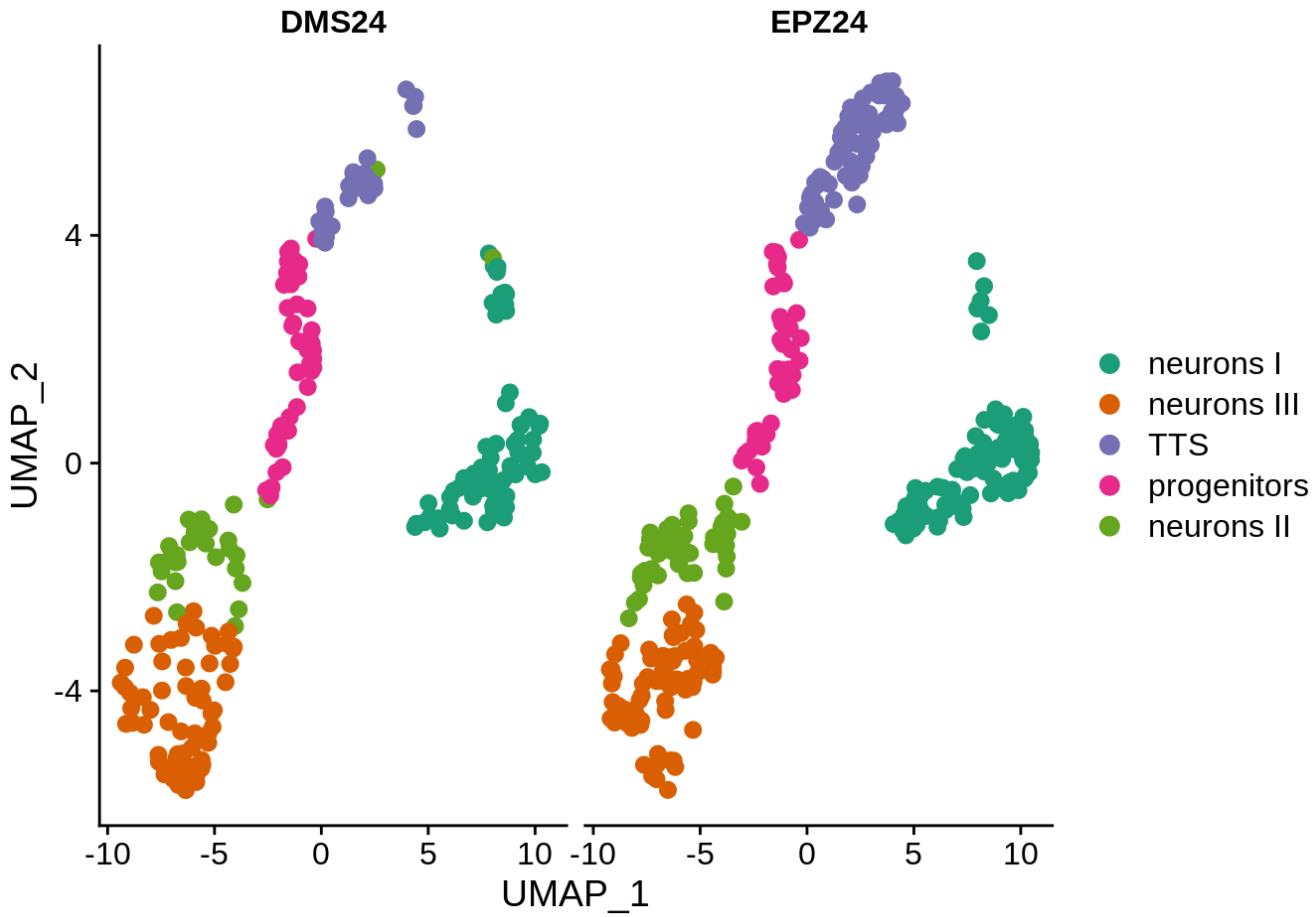



```
############

# Run this only for Supplementary Figure 6D
# Plot Supplementary Figure 6D
#DimPlot(sce_hvg.seurat, reduction = "umap", pt.size = 2.5)
```

19. Plot Figure 4D, Appiah et al. | Note: Order of labels may be different from final figure in manuscript

```
DimPlot(sce_hvg.seurat, reduction = "umap", split.by = "condition", pt.size = 2.5,
cols = t)
```




```
sessionInfo()
```

```
## R version 3.6.1 (2019-07-05)
## Platform: x86_64-conda_cos6-linux-gnu (64-bit)
## Running under: Ubuntu 18.10
##
## Matrix products: default
## BLAS/LAPACK: /home/bisi/anaconda3/envs/sc3/lib/R/lib/libRblas.so
##
## locale:
##  [1] LC_CTYPE=en_GB.UTF-8          LC_NUMERIC=de_DE.UTF-8
##  [3] LC_TIME=de_DE.UTF-8           LC_COLLATE=en_GB.UTF-8
##  [5] LC_MONETARY=de_DE.UTF-8       LC_MESSAGES=en_GB.UTF-8
##  [7] LC_PAPER=de_DE.UTF-8          LC_NAME=de_DE.UTF-8
##  [9] LC_ADDRESS=de_DE.UTF-8        LC_TELEPHONE=de_DE.UTF-8
## [11] LC_MEASUREMENT=de_DE.UTF-8    LC_IDENTIFICATION=de_DE.UTF-8
##
## attached base packages:
##  [1] parallel  stats4    tools     stats     graphics  grDevices utils
##  [8] datasets  methods   base
##
## other attached packages:
##  [1] cowplot_1.0.0               latexpdf_0.1.6
##  [3] umap_0.2.3.1                Seurat_3.1.0
##  [5] RColorBrewer_1.1-2          mclust_5.4.5
##  [7] org.Mm.eg.db_3.10.0         AnnotationDbi_1.48.0
##  [9] stringr_1.4.0               slingshot_1.2.0
## [11] princurve_2.1.4             SC3_1.12.0
```

```
## [13] scater_1.12.2                ggplot2_3.2.1
## [15] scran_1.12.1                 mvoutlier_2.0.9
## [17] sgeostat_1.0-27              HDF5Array_1.12.1
## [19] rhdf5_2.28.0                 SingleCellExperiment_1.6.0
## [21] SummarizedExperiment_1.14.0 DelayedArray_0.10.0
## [23] BiocParallel_1.18.0          matrixStats_0.55.0
## [25] Biobase_2.44.0               GenomicRanges_1.36.0
## [27] GenomeInfoDb_1.20.0          IRanges_2.18.2
## [29] S4Vectors_0.22.0             BiocGenerics_0.30.0
## [31] patchwork_1.0.0              dplyr_0.8.3
## [33] data.table_1.12.4            readr_1.3.1
## [35] BiocStyle_2.12.0
##
## loaded via a namespace (and not attached):
##   [1] rgl_0.100.30              rsvd_1.0.2
##   [3] vcd_1.4-4                 ica_1.0-2
##   [5] zinbwave_1.6.0            class_7.3-15
##   [7] foreach_1.4.7             lmtest_0.9-37
##   [9] glmnet_2.0-18             crayon_1.3.4
##  [11] laeken_0.5.0              MASS_7.3-51.4
##  [13] WriteXLS_5.0.0            nlme_3.1-141
##  [15] backports_1.1.5           rlang_0.4.0
##  [17] XVector_0.24.0            ROCR_1.0-7
##  [19] readxl_1.3.1              irlba_2.3.3
##  [21] limma_3.40.2              phylobase_0.8.6
##  [23] manipulateWidget_0.10.0  bit64_0.9-7
##  [25] glue_1.3.1                pheatmap_1.0.12
##  [27] rngtools_1.4              sctransform_0.2.0
##  [29] vipor_0.4.5               haven_2.1.1
##  [31] tidyselect_0.2.5          NADA_1.6-1.1
##  [33] rio_0.5.16                fitdistrplus_1.0-14
##  [35] XML_3.98-1.20             tidyr_1.0.0
##  [37] zoo_1.8-9                 xtable_1.8-4
##  [39] magrittr_1.5              evaluate_0.14
##  [41] bibtex_0.4.2              Rdpack_0.11-0
##  [43] zlibbioc_1.30.0           doRNG_1.7.1
##  [45] miniUI_0.1.1.1            sp_1.3-1
##  [47] robCompositions_2.2.1     pls_2.7-2
##  [49] zCompositions_1.3.4       locfdr_1.1-8
##  [51] shiny_1.4.0               BiocSingular_1.0.0
##  [53] xfun_0.10                 askpass_1.1
##  [55] cluster_2.1.0             caTools_1.17.1.2
##  [57] tibble_2.1.3              ggrepel_0.8.2
##  [59] ape_5.3                   listenv_0.7.0
##  [61] stabledist_0.7-1          png_0.1-7
##  [63] reshape_0.8.8             future_1.14.0
##  [65] zeallot_0.1.0             withr_2.1.2
##  [67] bitops_1.0-6              ranger_0.11.2
##  [69] plyr_1.8.4                cellranger_1.1.0
##  [71] pcaPP_1.9-73              e1071_1.7-2
##  [73] dqrng_0.2.1               pillar_1.4.2
##  [75] gplots_3.0.1.1            flexmix_2.3-15
##  [77] kernlab_0.9-27            DelayedMatrixStats_1.6.0
##  [79] vctrs_0.2.0               NMF_0.21.0
##  [81] metap_1.1                 foreign_0.8-72
##  [83] rncl_0.8.3                beeswarm_0.2.3
##  [85] munsell_0.5.0             fastmap_1.0.1
```

```
##  [87] compiler_3.6.1           abind_1.4-5
##  [89] httpuv_1.5.2             pkgmaker_0.27
##  [91] plotly_4.9.0             GenomeInfoDbData_1.2.1
##  [93] gridExtra_2.3            edgeR_3.26.5
##  [95] lattice_0.20-38          later_1.0.0
##  [97] jsonlite_1.6             GGally_1.5.0
##  [99] scales_1.0.0             pbapply_1.4-2
## [101] carData_3.0-2            genefilter_1.66.0
## [103] lazyeval_0.2.2           promises_1.1.0
## [105] car_3.0-3                doParallel_1.0.15
## [107] R.utils_2.9.0            reticulate_1.13
## [109] rmarkdown_1.16           openxlsx_4.1.0.1
## [111] statmod_1.4.32           webshot_0.5.1
## [113] Rtsne_0.15               forcats_0.4.0
## [115] copula_0.999-19.1        softImpute_1.4
## [117] uwot_0.1.8               igraph_1.2.4.1
## [119] survival_2.44-1.1        numDeriv_2016.8-1.1
## [121] yaml_2.2.0               prabclus_2.3-1
## [123] htmltools_0.4.0          memoise_1.1.0
## [125] modeltools_0.2-22        locfit_1.5-9.1
## [127] viridisLite_0.3.0        digest_0.6.21
## [129] rrcov_1.4-7              assertthat_0.2.1
## [131] mime_0.7                 registry_0.5-1
## [133] npsurv_0.4-0             RSQLite_2.1.2
## [135] future.apply_1.3.0       lsei_1.2-0
## [137] blob_1.2.0               R.oo_1.22.0
## [139] RNeXML_2.3.0             labeling_0.3
## [141] splines_3.6.1            Rhdf5lib_1.6.0
## [143] fpc_2.2-3                RCurl_1.95-4.12
## [145] cvTools_0.3.2            hms_0.5.1
## [147] colorspace_1.4-1         BiocManager_1.30.7
## [149] ggbeeswarm_0.6.0         SDMTools_1.1-221.1
## [151] nnet_7.3-12              Rcpp_1.0.2
## [153] ADGofTest_0.3            RANN_2.6.1
## [155] mvtnorm_1.0-11           pspline_1.0-18
## [157] VIM_4.8.0                truncnorm_1.0-8
## [159] R6_2.4.0                 grid_3.6.1
## [161] ggridges_0.5.1           lifecycle_0.1.0
## [163] zip_2.0.4                curl_4.2
## [165] gdata_2.18.0             leiden_0.3.1
## [167] robustbase_0.93-5        Matrix_1.2-17
## [169] howmany_0.3-1            RcppAnnoy_0.0.13
## [171] iterators_1.0.12         htmlwidgets_1.5.1
## [173] purrr_0.3.2              crosstalk_1.0.0
## [175] globals_0.12.4           openssl_1.4.1
## [177] clusterExperiment_2.4.4  codetools_0.2-16
## [179] gtools_3.8.1             prettyunits_1.0.2
## [181] gridBase_0.4-7           RSpectra_0.15-0
## [183] R.methodsS3_1.7.1        gtable_0.3.0
## [185] tsne_0.1-3               DBI_1.0.0
## [187] dynamicTreeCut_1.63-1    httr_1.4.1
## [189] KernSmooth_2.23-15       stringi_1.4.6
## [191] progress_1.2.2           reshape2_1.4.3
## [193] uuid_0.1-2               diptest_0.75-7
## [195] annotate_1.62.0          viridis_0.5.1
## [197] xml2_1.2.2               boot_1.3-23
## [199] BiocNeighbors_1.2.0      ade4_1.7-13
```

```
## [201] sROC_0.1-2                DEoptimR_1.0-8
## [203] bit_1.1-14                pkgconfig_2.0.3
## [205] gsl_2.1-6                 gbRd_0.4-11
## [207] knitr_1.25
```