Analysis of Electroporated 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(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)
```

1. Data preparation

#Loading data matrix

Convert data to "SingleCellExperiment" Class.

```
data2 <-  readRDS("EUE14ALL.RData")
sc2 <- SingleCellExperiment(list(counts=as.matrix(data2)))
sc2
```

```
## class: SingleCellExperiment
## dim: 34205 743
## metadata(0):
## assays(1): counts
## rownames(34205): 0610007P14Rik 0610009B22Rik ... Zzef1 Zzz3
## rowData names(0):
## colnames(743): EUEE14DMSORP5_1 EUEE14DMSORP5_2 ... EUEE14EPZRP9_95
##   EUEE14EPZRP9_96
## colData names(0):
## reducedDimNames(0):
## spikeNames(0):
```

```
EUE14DMS_1 <- sc2[, grep('EUEE14DMS', colnames(sc2))]
EUE14DMS <- colnames(EUE14DMS_1)

EUE14EPZ_1 <- sc2[, grep('EUEE14EPZ', colnames(sc2))]
EUE14EPZ <- colnames(EUE14EPZ_1)
```

#Define DMSO and EPZ conditions

```
sc2$condition <- "NA"
sc2$condition[which(colnames(sc2) %in% EUE14DMS)] <- "DMSO"
sc2$condition[which(colnames(sc2) %in% EUE14EPZ)] <- "EPZ"
table(sc2$condition)
```

```
##
## DMSO  EPZ
##  368  375
```

#Add mitochondrial genes and ERCC spike-ins to sce object
#There were no ERCC spike-ins used in the experiment.

```
isSpike(sc2, "MT") <- rownames(sc2)[grep("^(mt)",rownames(sc2),invert=FALSE)]
isSpike(sc2, "ERCC") <- rownames(sc2)[grep("^(ERCC)",rownames(sc2),invert=FALSE)]
sc2
```

```
## class: SingleCellExperiment
## dim: 34205 743
## metadata(0):
## assays(1): counts
## rownames(34205): 0610007P14Rik 0610009B22Rik ... Zzef1 Zzz3
## rowData names(0):
## colnames(743): EUEE14DMSORP5_1 EUEE14DMSORP5_2 ... EUEE14EPZRP9_95
##   EUEE14EPZRP9_96
## colData names(1): condition
## reducedDimNames(0):
## spikeNames(2): MT ERCC
```

2. Quality control

Remove genes that are not expressed

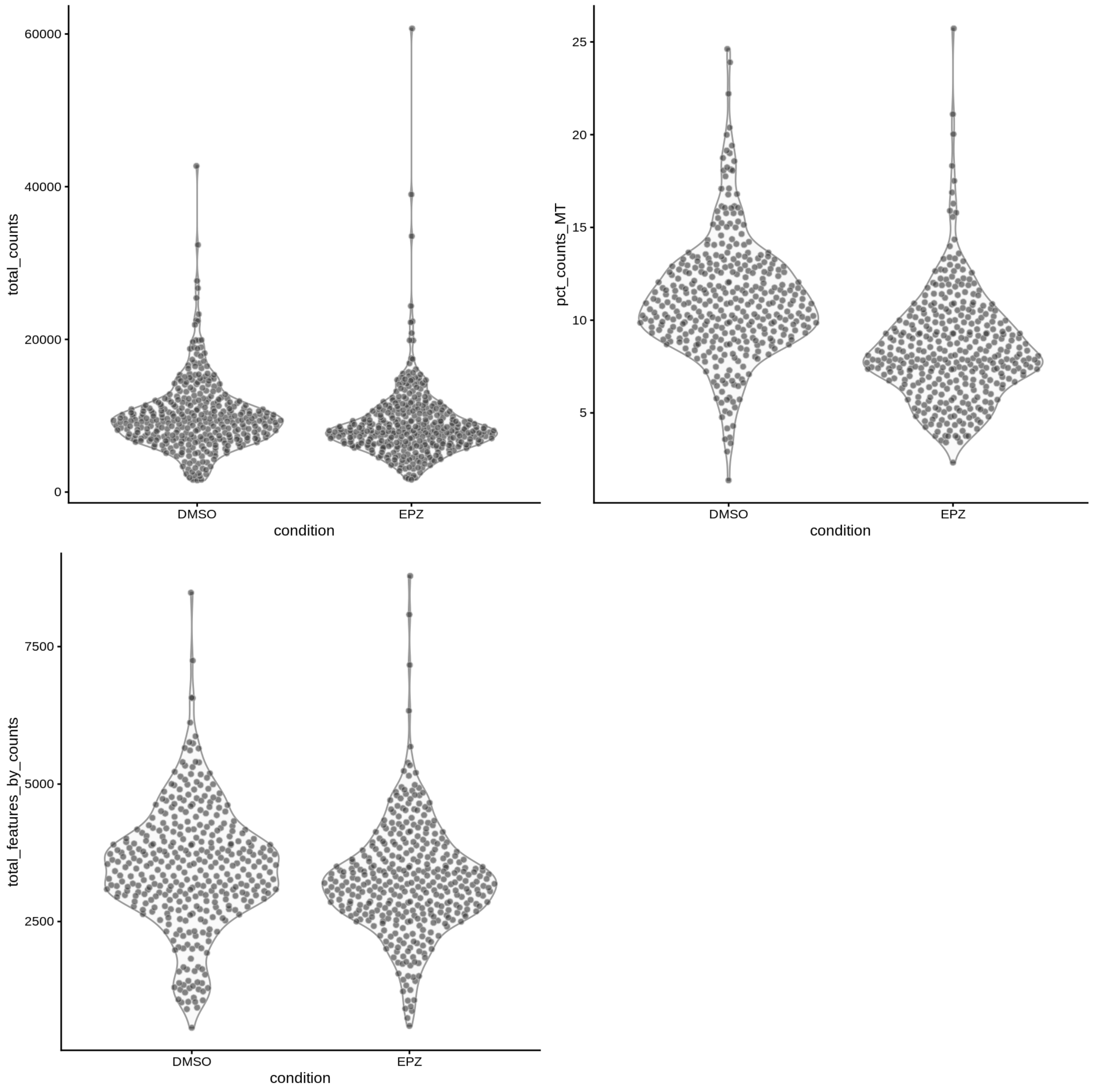
```
keep_feature <- rowSums(counts(sc2) > 0) > 0
sc2 <- sc2[keep_feature,]
```

Calculate quality metrics

It seems like there were no ERCC spike-ins used.

```
sc2 <- calculateQCMetrics(sc2)
```

```
multiplot(cols=2,
    plotColData(sc2, x="condition", y="total_counts"),
    plotColData(sc2, x="condition", y="total_features_by_counts"),
    plotColData(sc2, x="condition", y="pct_counts_MT")
)
```

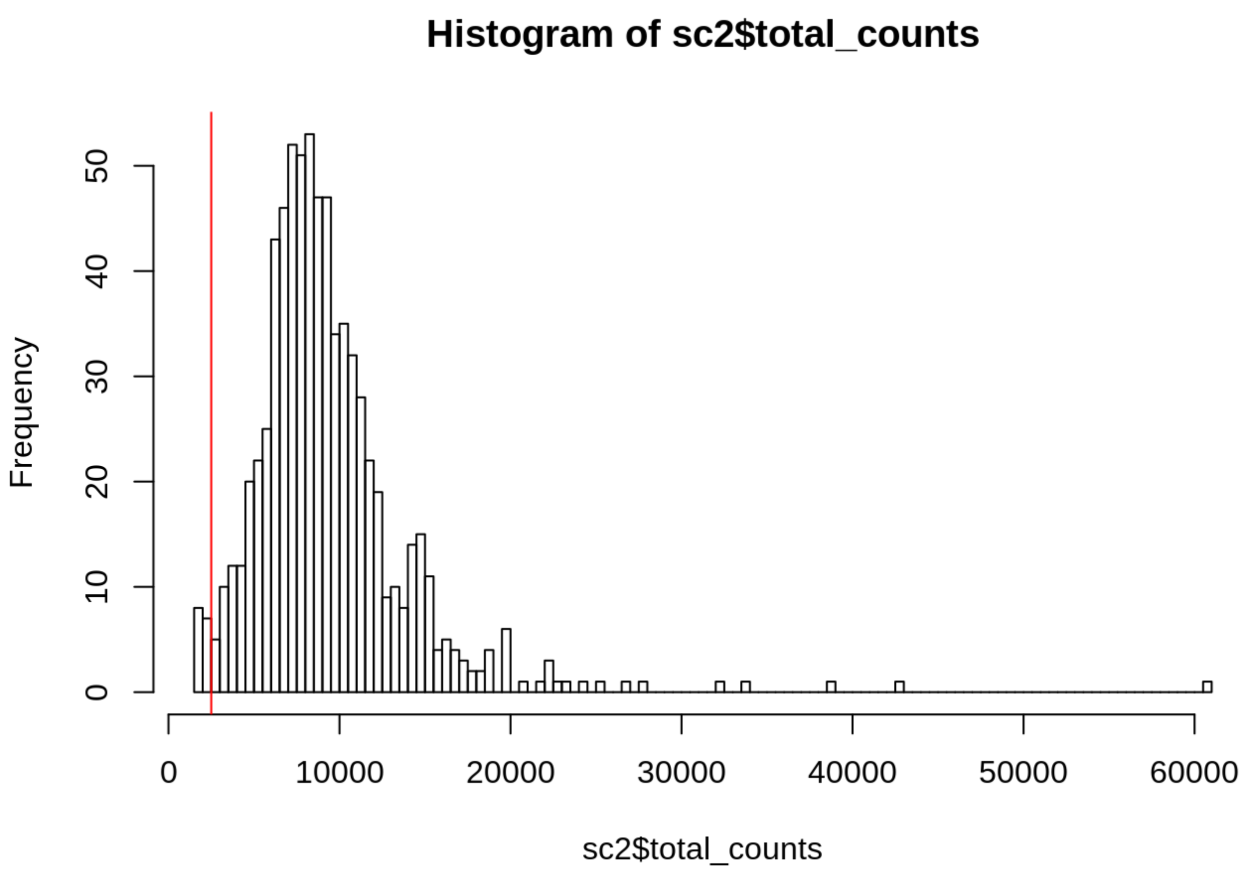


Total counts and distribution of counts for cells in both conditions looks quite good We can proceed with further analysis

```
sc2 <- sc2[grep("^(ERCC|Gm|Rik)",row.names(sc2),invert=TRUE),]
```

Visualize library sizes

```
hist(
    sc2$total_counts,
    breaks = 100
)
abline(v = 2500, col = "red")
```



3. Filter cells by library size > 2500

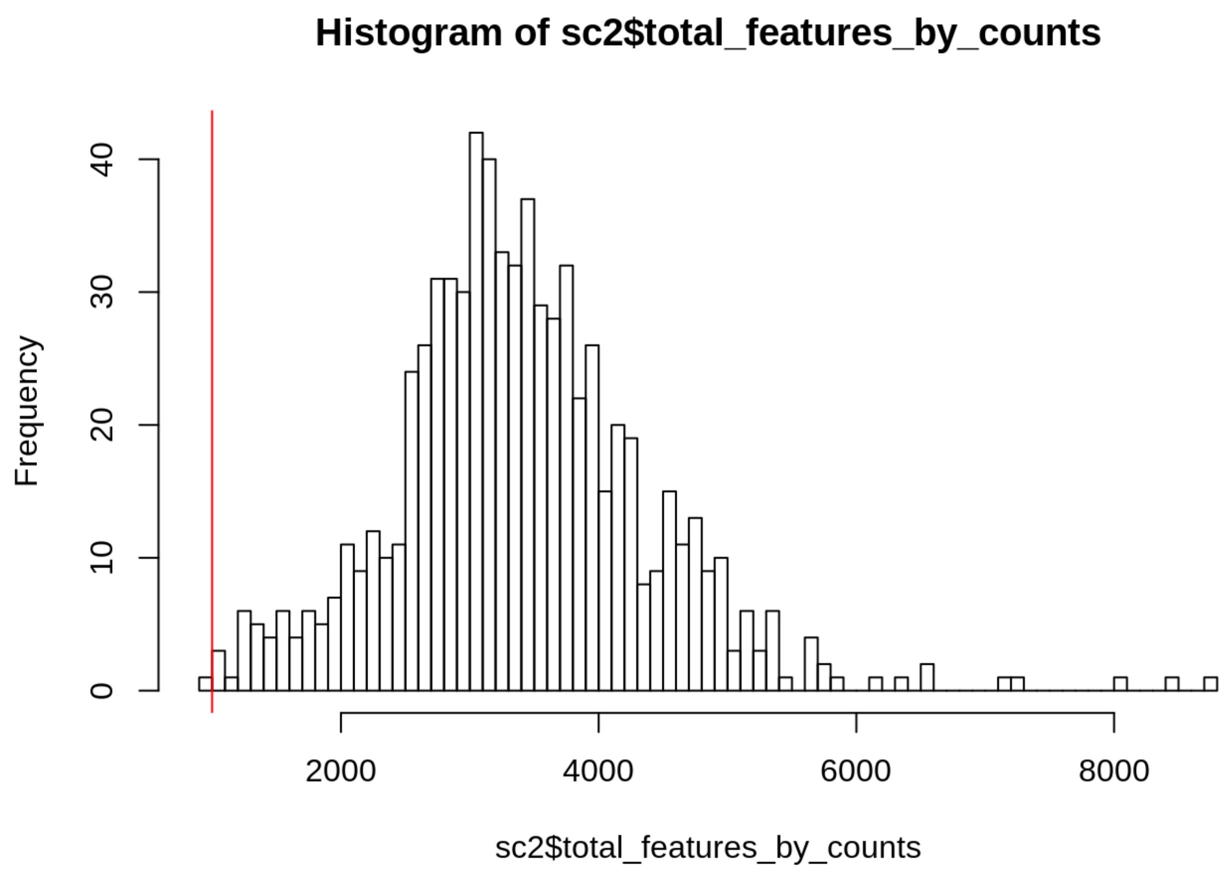
```
filter_by_total_counts <- (sc2$total_counts > 2500)
table(filter_by_total_counts)
```

```
## filter_by_total_counts
## FALSE  TRUE
##    15   728
```

```
sc2 <- sc2[,filter_by_total_counts]
```

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

```
hist(
    sc2$total_features_by_counts,
    breaks = 100
)
abline(v = 1000, col = "red")
```



5. Filter cells by expressed features > 1000

```
filter_by_expr_features <- (sc2$total_features_by_counts > 1000)
table(filter_by_expr_features)
```

```
## filter_by_expr_features
## FALSE  TRUE
##     1   727
```

```
sc2 <- sc2[,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(sc2$pct_counts_MT, nmads=3, type="higher", batch=sc2$conditio
n)
data.frame(HighMito=sum(high.mt))
```

```
##   HighMito
## 1       16
```

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        16       711
```

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

```
sc2 <- sc2[,!discard]
ncol(sc2)
```

```
## [1] 711
```

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

```
#plotColData(sc2, x = "total_features_by_counts", y = "pct_counts_MT")
```

6. Examining the genes

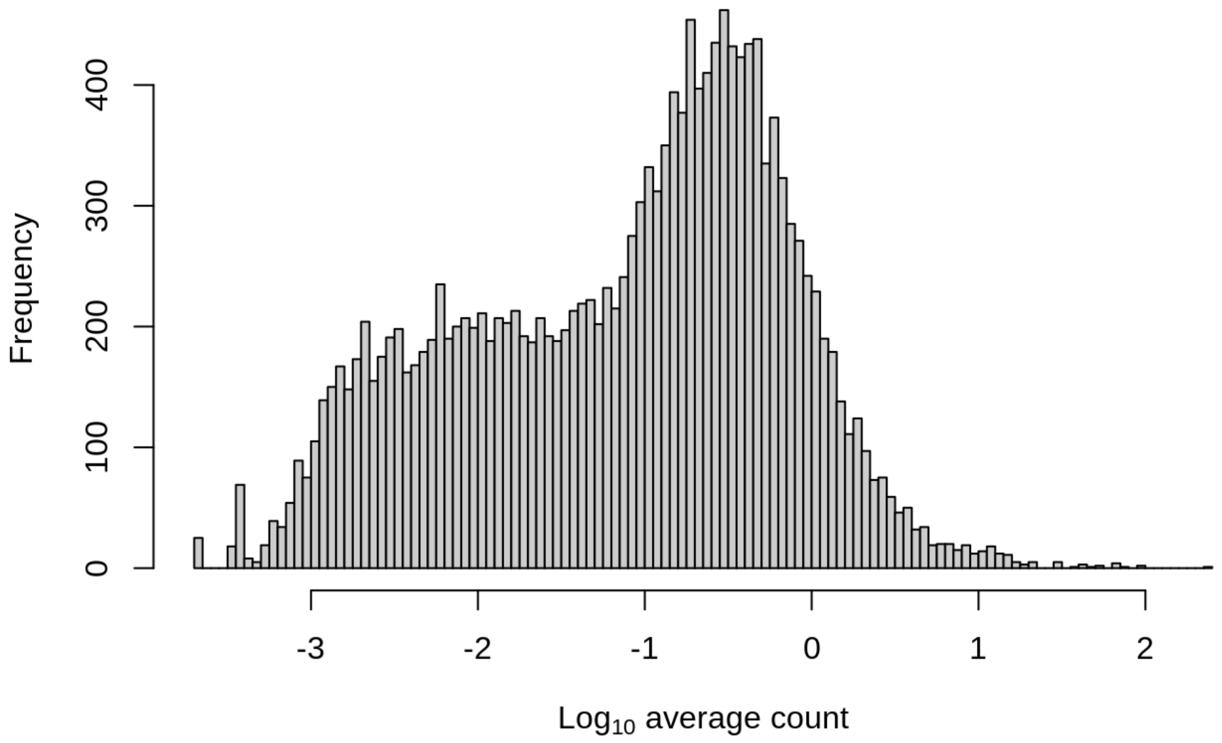
We inspect the distribution of log-mean counts across all genes. The peak represents the bulk of moderately expressed genes while the rectangular component corresponds to lowly expressed genes.

```
ave.counts <- calcAverage(sc2)
```

```
## 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
```

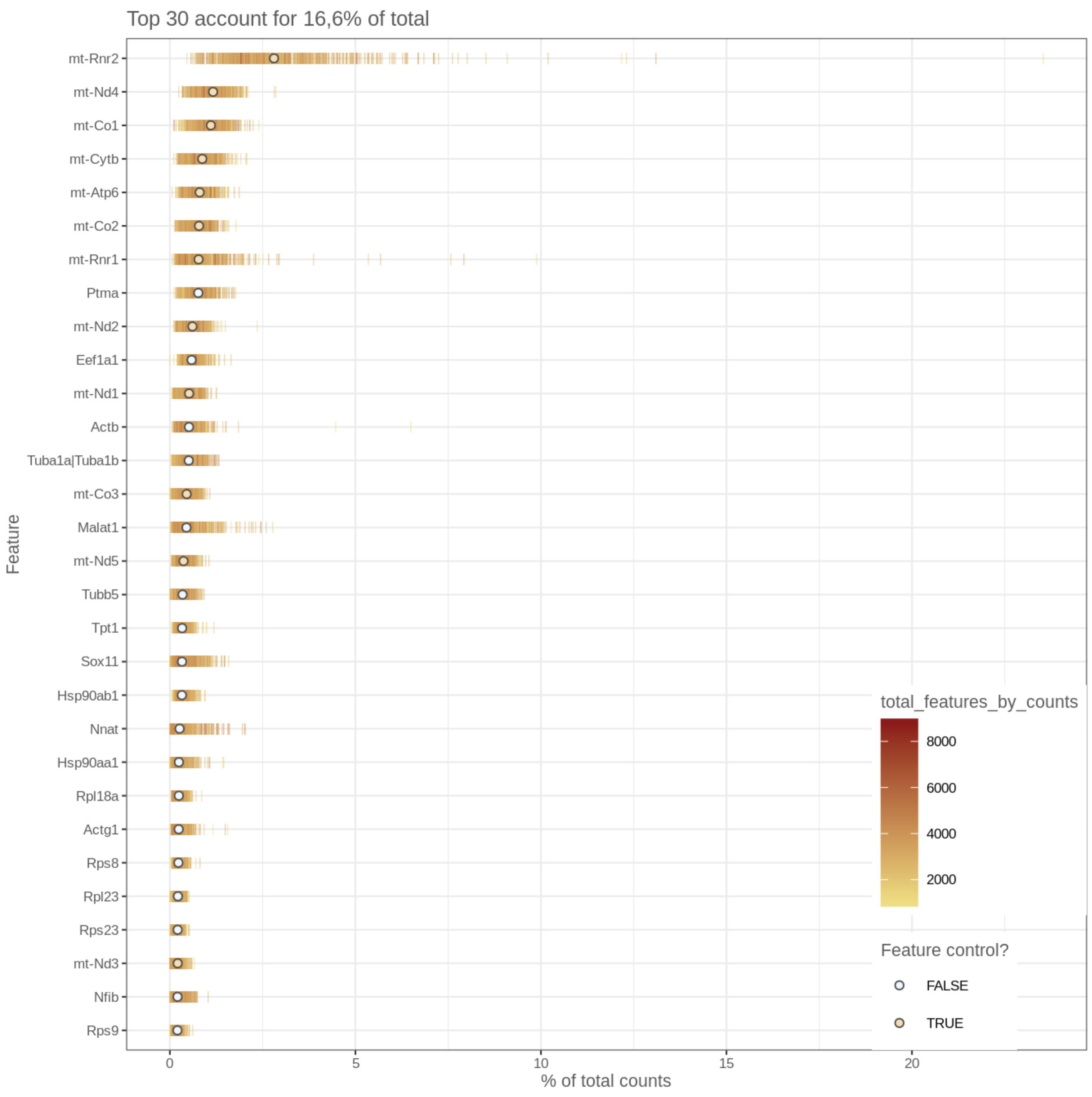
```
hist(log10(ave.counts), breaks=100, main="", col="grey80",
    xlab=expression(Log[10]~"average count"))
```



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(sc2, n=30)
```



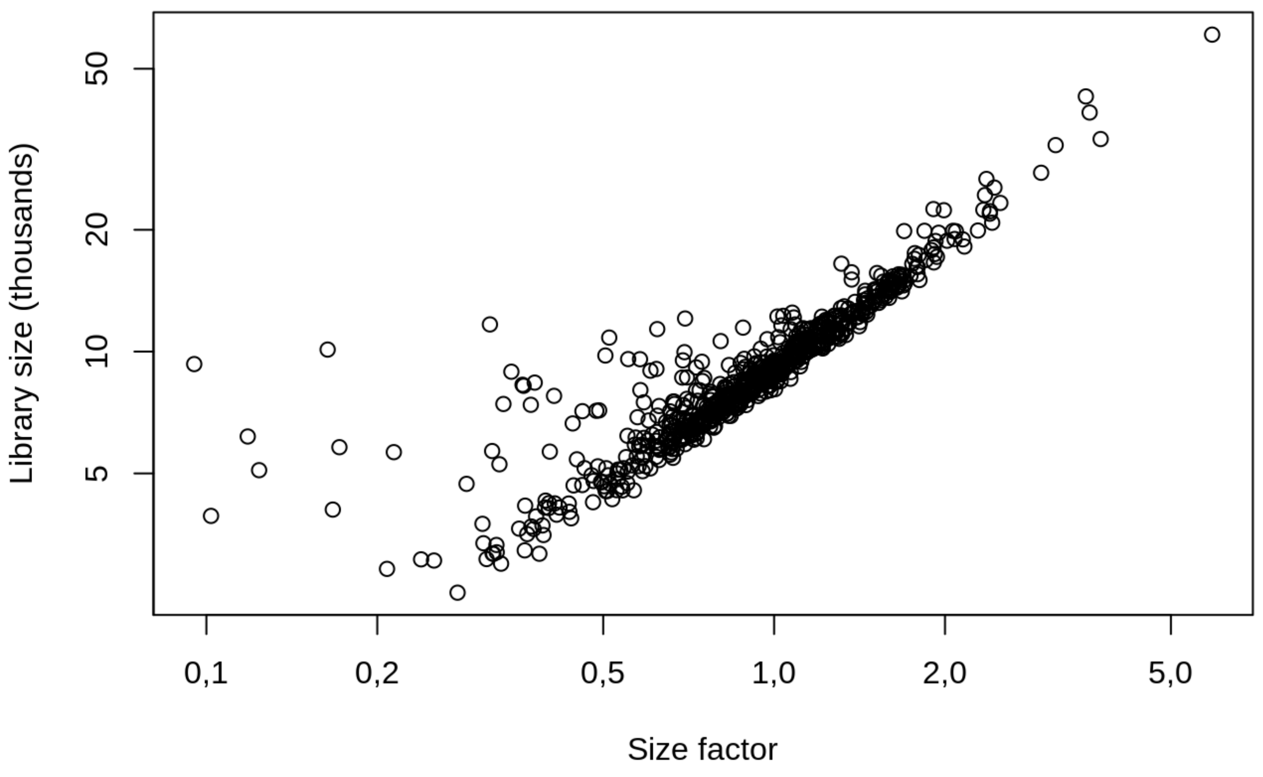
```
clusters <- quickCluster(sc2,use.ranks=FALSE)
sc2 <- computeSumFactors(sc2, cluster=clusters)
```

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(sc2), sc2$total_counts/1e3, log="xy",
ylab="Library size (thousands)", xlab="Size factor")
```



```
sc2 <- scater::normalize(sc2)
```

```
## 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
```

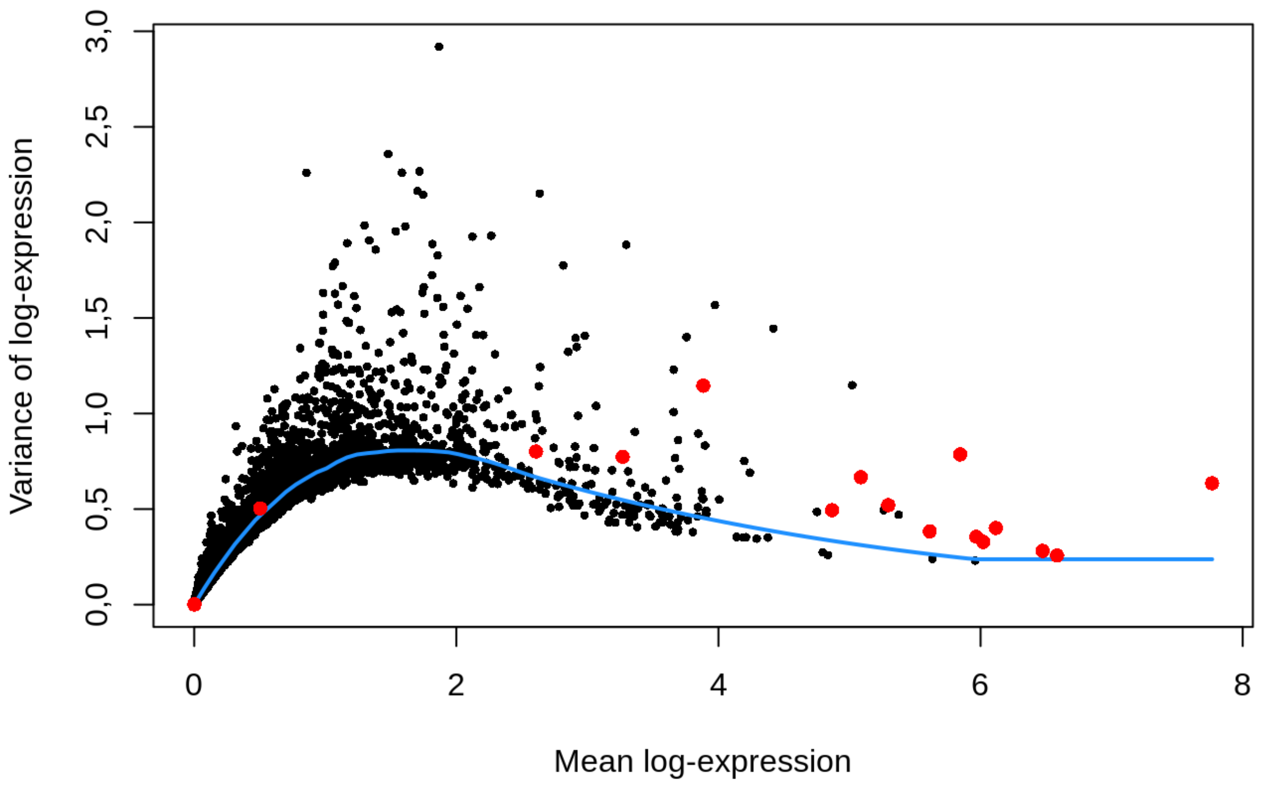
```
sc2
```

```
## class: SingleCellExperiment
## dim: 17456 711
## metadata(1): log.exprs.offset
## assays(2): counts logcounts
## rownames(17456): 0610007P14Rik 0610009B22Rik ... Zzef1 Zzz3
## rowData names(9): is_feature_control is_feature_control_MT ...
##   total_counts log10_total_counts
## colnames(711): EUEE14DMSORP5_1 EUEE14DMSORP5_2 ... EUEE14EPZRP9_95
##   EUEE14EPZRP9_96
## 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
```

```
var.fit <- trendVar(sc2, method="loess", loess.args=list(span=0.05), use.spikes=FA
LSE)
var.out <- decomposeVar(sc2, 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$trend(x), add=TRUE, col="dodgerblue", lwd=2)
cur.spike <- isSpike(sc2)
points(var.out$mean[cur.spike], var.out$total[cur.spike], col="red", pch=16)
```



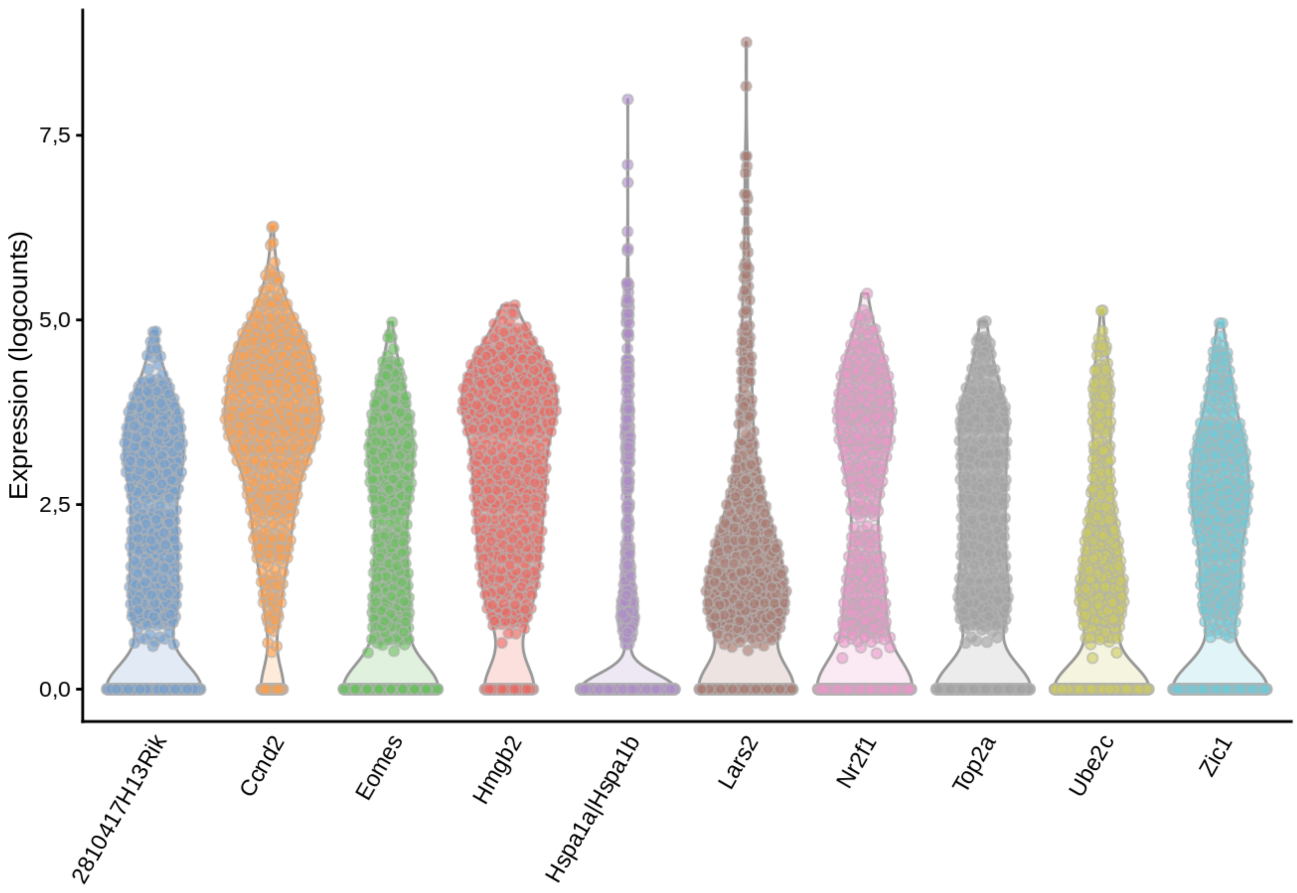
```
hvg.out <- var.out[with(var.out, order(bio,FDR, decreasing = TRUE)),][1:800,]
hvg.out <- hvg.out[order(hvg.out$bio, decreasing = TRUE),]
nrow(hvg.out)
```

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

8. 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(sc2, rownames(hvg.out)[1:10])
```



9. Extract highly variable genes (HVG)

```
sce_hvg <- sc2[row.names(hvg.out),]
```

10. 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),]
```

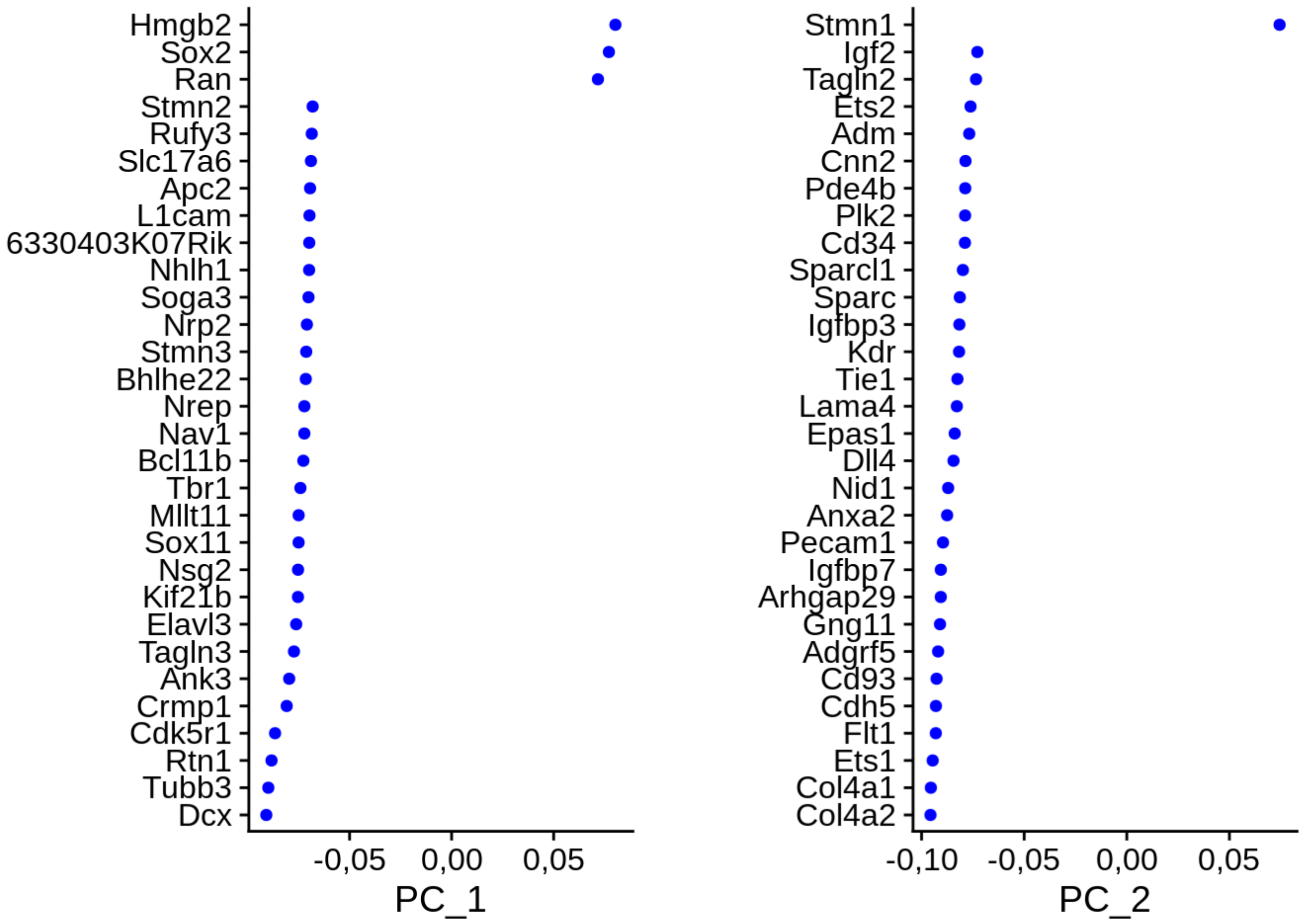
11. Run PCA on Seurat object.

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

```
## PC_ 1
## Positive:  Hmgb2, Sox2, Ran, Npm1, H2afz, Top2a, Dek, Psat1, Phgdh, Smc4
##     Plpp3, Ccnd2, Ccna2, Anp32b, Qk, Nap1l1, 2810417H13Rik, Rrm2, Dut, Zfp36l1
##     Cdca7, Grb10, Mki67, Rpsa, Lig1, Ybx1, Sox9, Tmpo, Pcna, Sfrp1
## Negative:  Dcx, Tubb3, Rtn1, Cdk5r1, Crmp1, Ank3, Tagln3, Elavl3, Kif21b, Nsg2
##     Sox11, Mllt11, Tbr1, Bcl11b, Nav1, Nrep, Bhlhe22, Stmn3, Nrp2, Soga3
##     Nhlh1, 6330403K07Rik, L1cam, Apc2, Slc17a6, Rufy3, Stmn2, Basp1, Rnd2, Islr
2
## PC_ 2
## Positive:  Stmn1, Hnrnpab, Nfib, Hmgn2, Cdk2ap1, Lhx2, Tuba1aTuba1b, Sox11, H2a
fv, Hmgb3
##     Erh, H2afz, Insm1, Nfia, Ezh2, Calm2, Nfix, Elavl4, Mki67, Hn1
##     Epha4, Foxg1, Elavl2, Tcf4, Sfpq, Zbtb18, Bcl11a, Igfbpl1, Birc5, Zfp462
## Negative:  Col4a2, Col4a1, Ets1, Flt1, Cdh5, Cd93, Adgrf5, Gng11, Arhgap29, Igf
bp7
##     Pecam1, Anxa2, Nid1, Dll4, Epas1, Lama4, Tie1, Kdr, Igfbp3, Sparc
##     Sparcl1, Cd34, Plk2, Pde4b, Cnn2, Adm, Ets2, Tagln2, Igf2, Apold1
## PC_ 3
## Positive:  Ckb, Aldoc, Ttyh1, Dbi, Mfge8, Id4, Sox9, Slc1a3, Hes5, Fabp7
##     Acot1, Ndrg2, Clu, Ptprz1, Ddah1, Plpp3, Calr, Hspd1, Dkk3, Nes
##     Atxn7l3b, Igfbp5, Fgfbp3, Draxin, Ptn, Arx, Mt1, Rplp1, Mdk, Serpinh1
## Negative:  Ube2c, Cenpf, Cenpe, Kif23, Plk1, Aspm, Cks2, Ckap2l, Ccnb1, Tpx2
##     Mki67, Aurka, Hmmr, Arhgef39, Top2a, Kpna2, Cdc20, Ccna2, Sgol2a, Cdk1
##     Tacc3, Gas2l3, Cdca3, Aurkb, Arl6ip1, Mis18bp1, Cdca2, Kif22, Racgap1, Nusa
p1
## PC_ 4
## Positive:  Zic1, Zbtb20, Zic4, Zic3, Zic5, Wnt8b, Draxin, Mpped2, Ttyh1, Ildr2
##     Nnat, Vcan, Prox1, Rmst, Tmem47, Mmp14, Sox9, Fgfbp3, Hes5, Ptms
##     Hopx, C130071C03Rik, Hes1, Socs2, Syt11, Msi2, Mir99ahg, Ptprz1, Marcksl1,
Ndrg2
## Negative:  Nr2f1, Igsf8, Ddit4, Ldha, Bnip3, Vegfa, Gapdh, Foxo6, Mn1, Hspa1aHs
pa1b
##     Meis2, Aldoa, Pgk1, Pkm, Tpi1, Hk2, Gpi1, P4ha1, Rplp0, Pgam1
##     2410006H16Rik, Rpsa, Slc2a1, Nxph4, Gas5, Higd1a, Fgfr3, Mif, Abracl, Neuro
d2
## PC_ 5
## Positive:  Eomes, Trp53i11, Chd7, Mcm6, Neurog1, Neurog2, Btg2, Cdk2ap1, Mdk, I
gfbpl1
##     Pcna-ps2, Pcna, Mcm2, Tmem2, Miat, Hes6, Elavl2, Dhfr, Lhx2, Ier5
##     Ranbp1, Wnt5a, Gadd45g, Mcm3, Cdk6, Cdt1, Rprm, Rcor2, Nfix, BC051077
## Negative:  Arl6ip1, Tubb2aTubb2b, Nr2f1, Ptn, Tuba1c, Arhgef39, Thbs1, Pea15a,
Fcer1g, Kpna2
##     Ccnb1, Igfbp5, Ubb, Hmmr, Cdc20, Ube2c, Ckap2, Ednrb, Actg1, Cenpa
##     Plek, Aspm, Fermt3, Gas2l3, Gpm6a, Malat1, Nr2f2, Npas3, Itgb3, Pf4
```

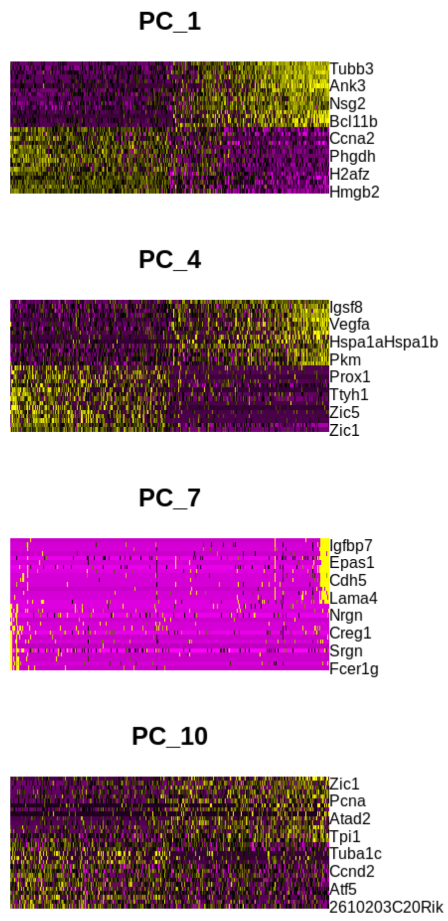
12. Inspecting first two principal components

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



13. 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.

14. 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?

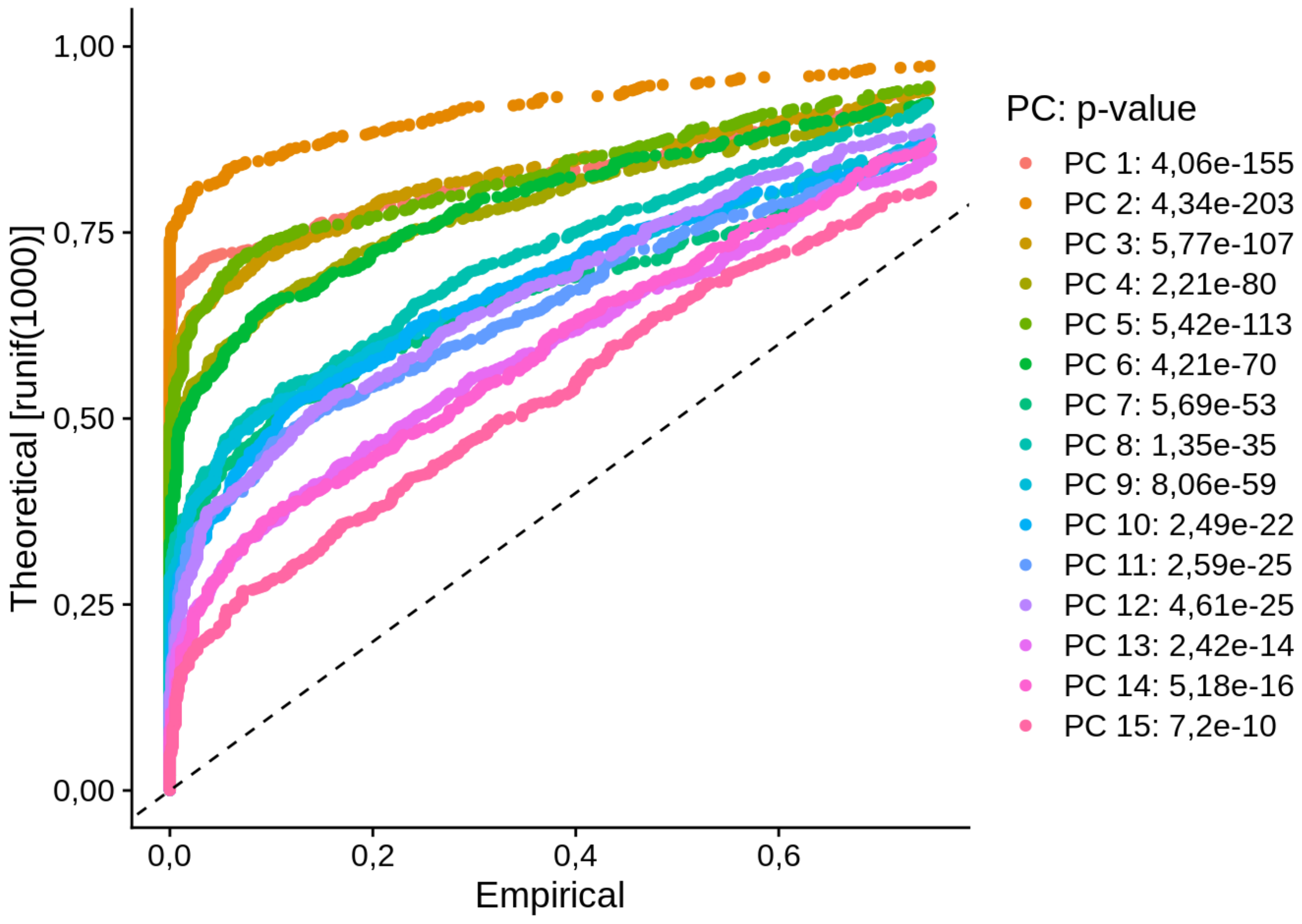
In Macosko et al, we implemented a resampling test inspired by the JackStraw procedure. We randomly permute a subset of the data (1% by default) and rerun PCA, constructing a 'null distribution' of feature scores, and repeat this procedure. We identify 'significant' PCs as those who have a strong enrichment 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 Supplementary Figure 6A, top

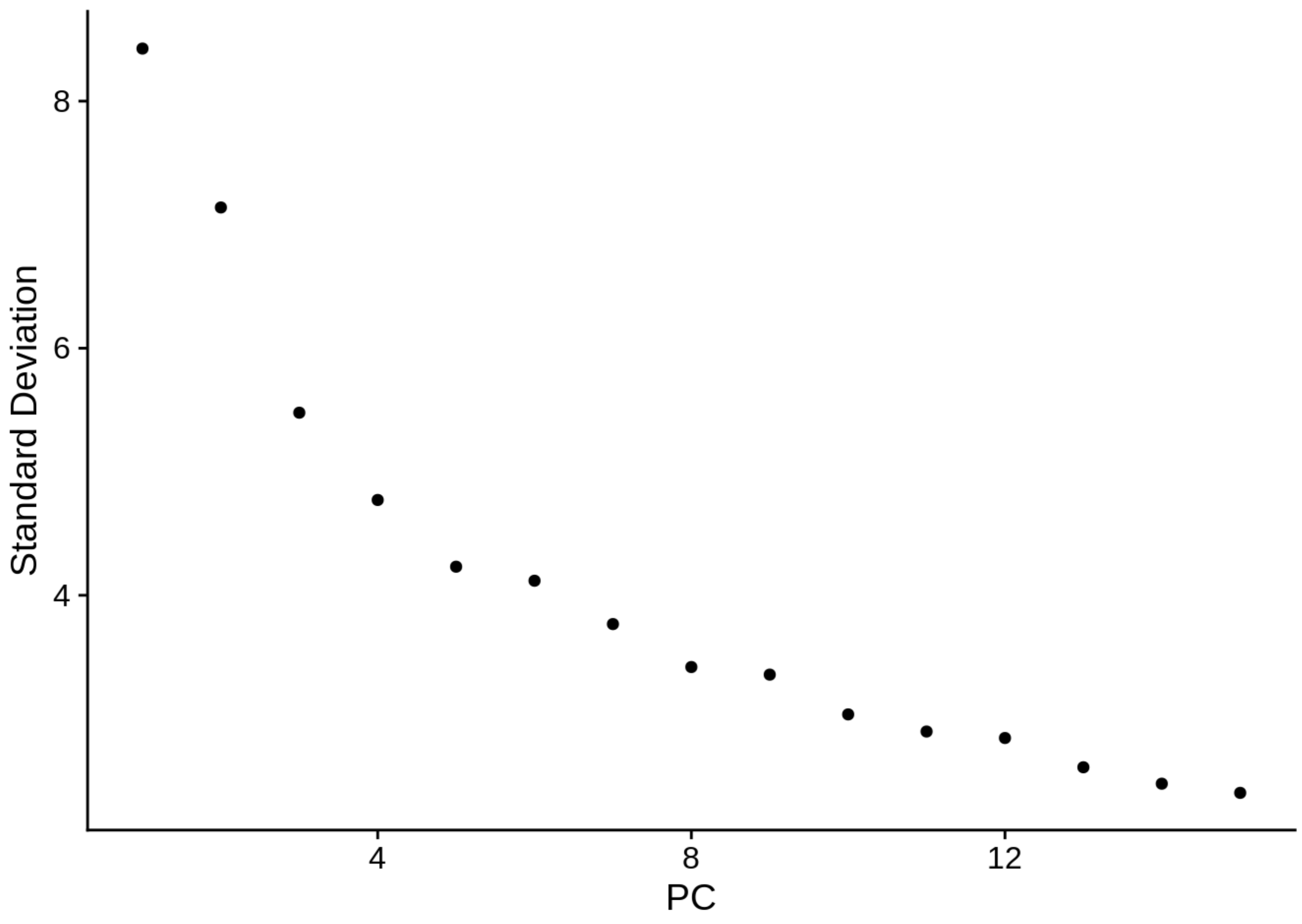
```
JackStrawPlot(sce_hvg.seurat, dims = 1:15, xmax = 0.75, ymax = 1.0)
```

```
## Warning: Removed 1195 rows containing missing values (geom_point).
```



Plot Supplementary Figure 6A, bottom

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



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

15. 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: 711
## Number of edges: 19052
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0,8312
## Number of communities: 5
## Elapsed time: 0 seconds
```

```
head(Idents(sce_hvg.seurat), 5)
```

```
## EUEE14DMSORP5_1 EUEE14DMSORP5_2 EUEE14DMSORP5_3 EUEE14DMSORP5_4
##               0               0               3               0
## EUEE14DMSORP5_6
##               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
```

```
## 19:09:11 UMAP embedding parameters a = 0,9922 b = 1,112
```

```
## 19:09:11 Read 711 rows and found 8 numeric columns
```

```
## 19:09:11 Using Annoy for neighbor search, n_neighbors = 30
```

```
## 19:09:11 Building Annoy index with metric = cosine, n_trees = 50
```

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

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

```
## **************************************************|
## 19:09:11 Writing NN index file to temp file /tmp/RtmpE4Egoj/file1b201957a59
## 19:09:11 Searching Annoy index using 1 thread, search_k = 3000
## 19:09:11 Annoy recall = 100%
## 19:09:12 Commencing smooth kNN distance calibration using 1 thread
## 19:09:13 Initializing from normalized Laplacian + noise
## 19:09:13 Commencing optimization for 500 epochs, with 25122 positive edges
## 19:09:15 Optimization finished
```

```r
t <- brewer.pal(n = 6,name = "Dark2") # Assigning color code for clusters
# Renaming clusters

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"
```

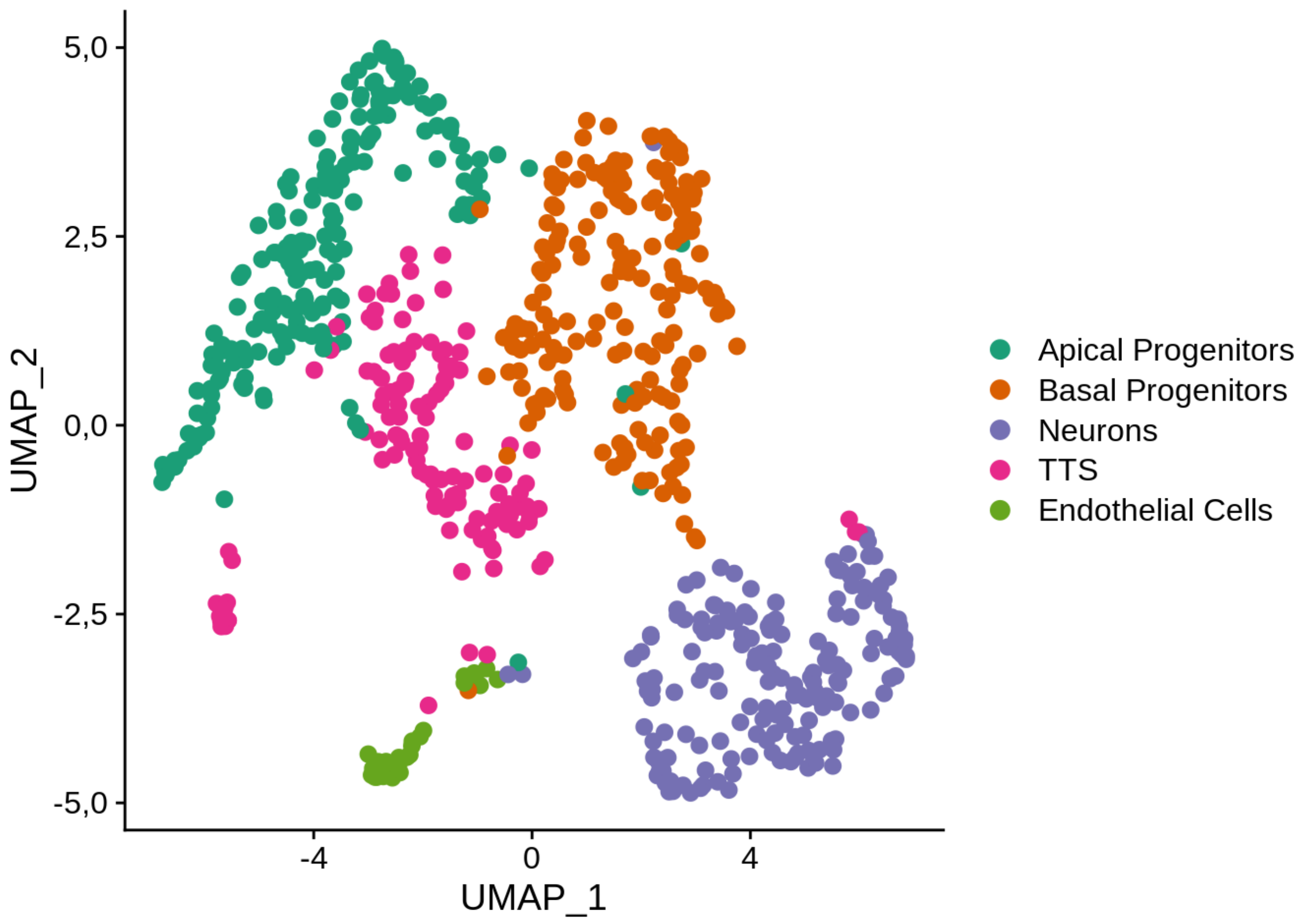
```
Idents(object = sce_hvg.seurat) <- 'seurat_clusters'
levels(sce_hvg.seurat)
```

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

```
### Assign conditions to Idents for running differential expression between cluste
rs within a defined condition (DMSO/EPZ)
#sce_hvg.seurat$condition <- paste(Idents(sce_hvg.seurat), sce_hvg.seurat$conditio
n, sep = "_")
#Idents(sce_hvg.seurat) <- "condition"
##DEG analysis between cluster "3" (TTS) and Cluster "0" (Apical progenitors) in t
he control condition (DMSO)
#TTS_vs_AP_DEG <- FindMarkers(sce_hvg.seurat, ident.1 = "3_DMSO", ident.2 = "0_DMS
O",test.use = "negbinom")
#TTS_vs_AP_DEG_UP <- TTS_vs_AP_DEG[which(TTS_vs_AP_DEG$avg_logFC > 0),]
#TTS_vs_AP_DEG_DOWN <- TTS_vs_AP_DEG[which(TTS_vs_AP_DEG$avg_logFC < 0),]
current.cluster.ids <- c(0,1,2,3,4) # c(0, 1, 2, 3, 4,5)
new.cluster.ids <- c("Apical Progenitors","Basal Progenitors","Neurons", "TTS","En
dothelial Cells")
names(new.cluster.ids) <- levels(sce_hvg.seurat)
sce_hvg.seurat <- RenameIdents(sce_hvg.seurat, new.cluster.ids)
```

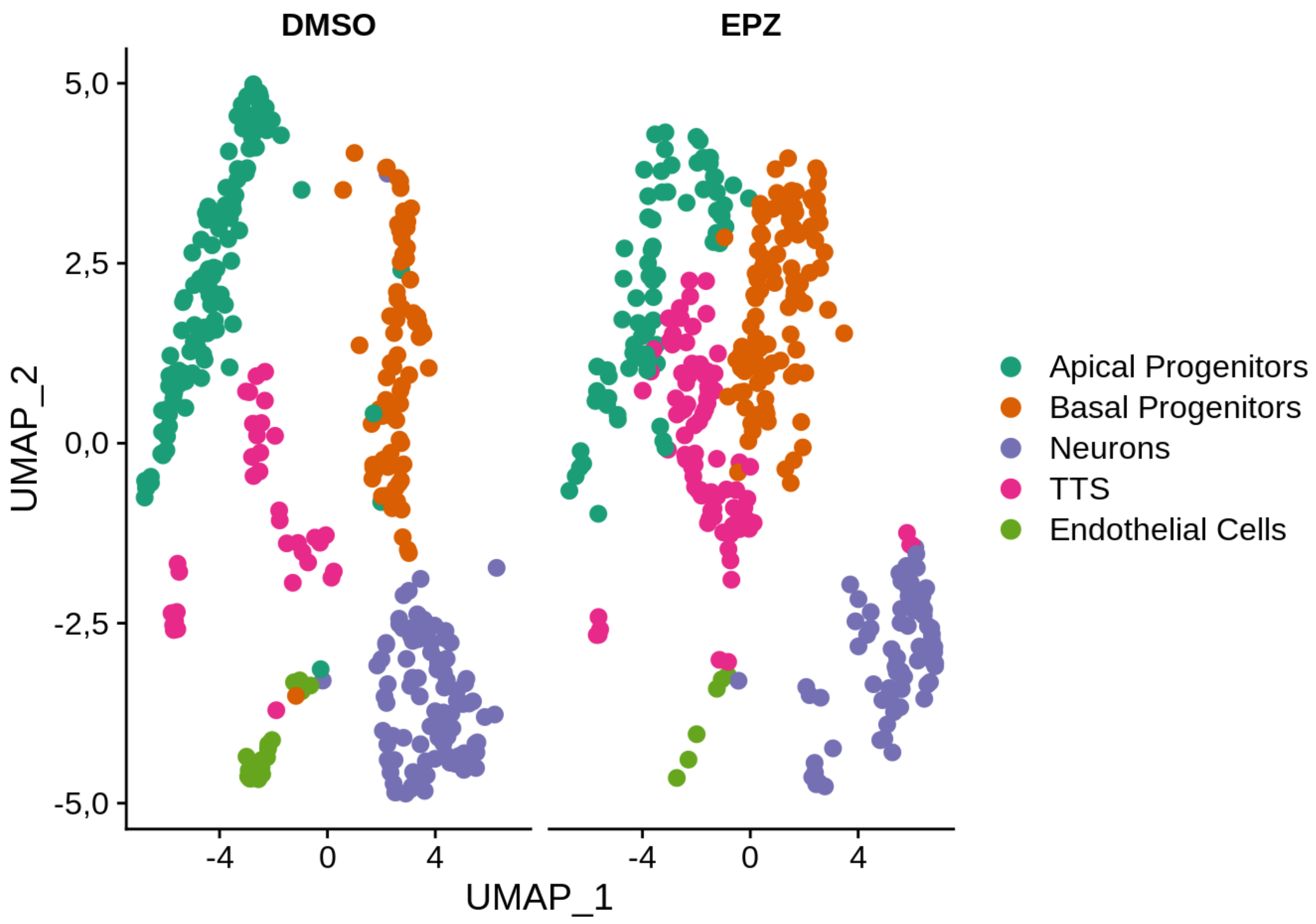
18. Plot Figure 4 F, Appiah et al. | Note: Order of labels may be different from final figure in manuscript

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



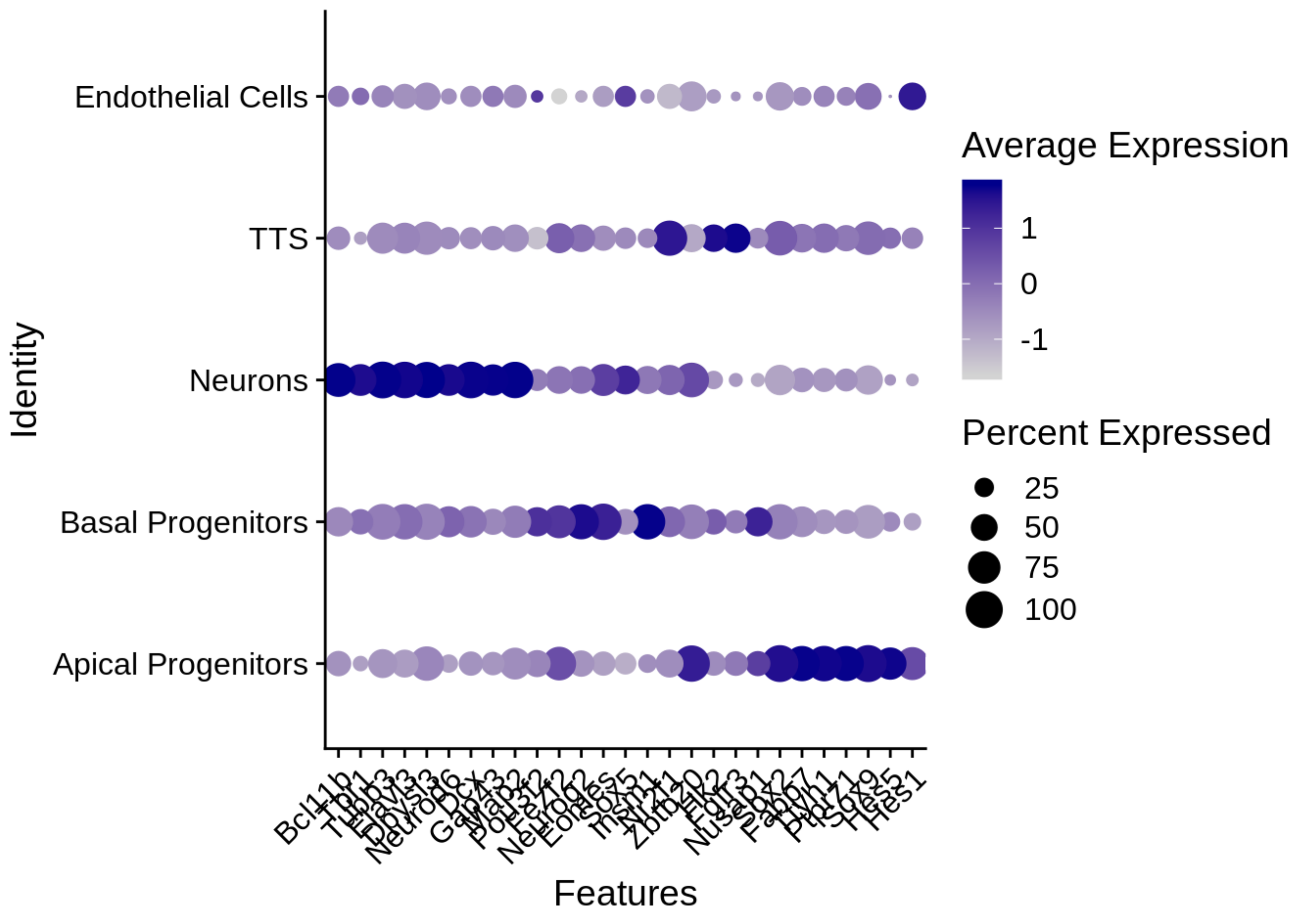
19. Plot Figure 4G, Appiah et al. | Note: Order of labels may be different from final figure in manuscript

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



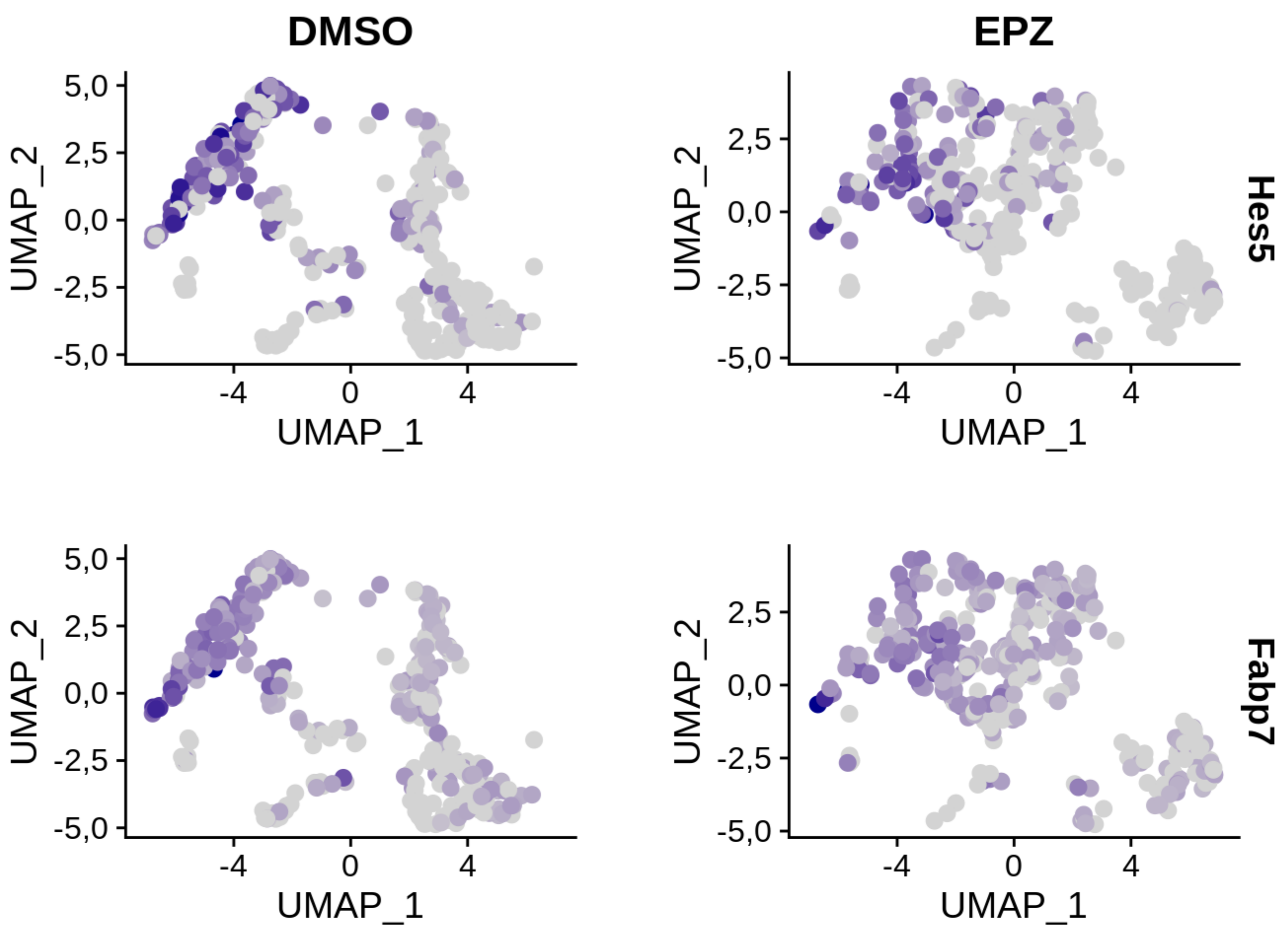
20. Plot Supplementary Figure 4A | Note: labels were rearranged for final figure in manuscript

```
DotPlot(sce_hvg.seurat, features = c("Hes1","Hes5","Sox9","Ptprz1","Ttyh1","Fabp7"
,"Sox2","Nusap1","Fgfr3","Hk2","Zbtb20","Nr2f1","Insm1","Sox5","Eomes","Neurog2",
"Fezf2","Pou3f2","Map2","Gap43","Dcx","Neurod6","Dpysl3","Elavl3","Tubb3","Tbr1",
"Bcl11b"),cols = c(rep("lightgrey"), "dark blue"),col.min = -2, col.max = 2, scal
e.by = "size", scale.max = 100) + RotatedAxis()
```

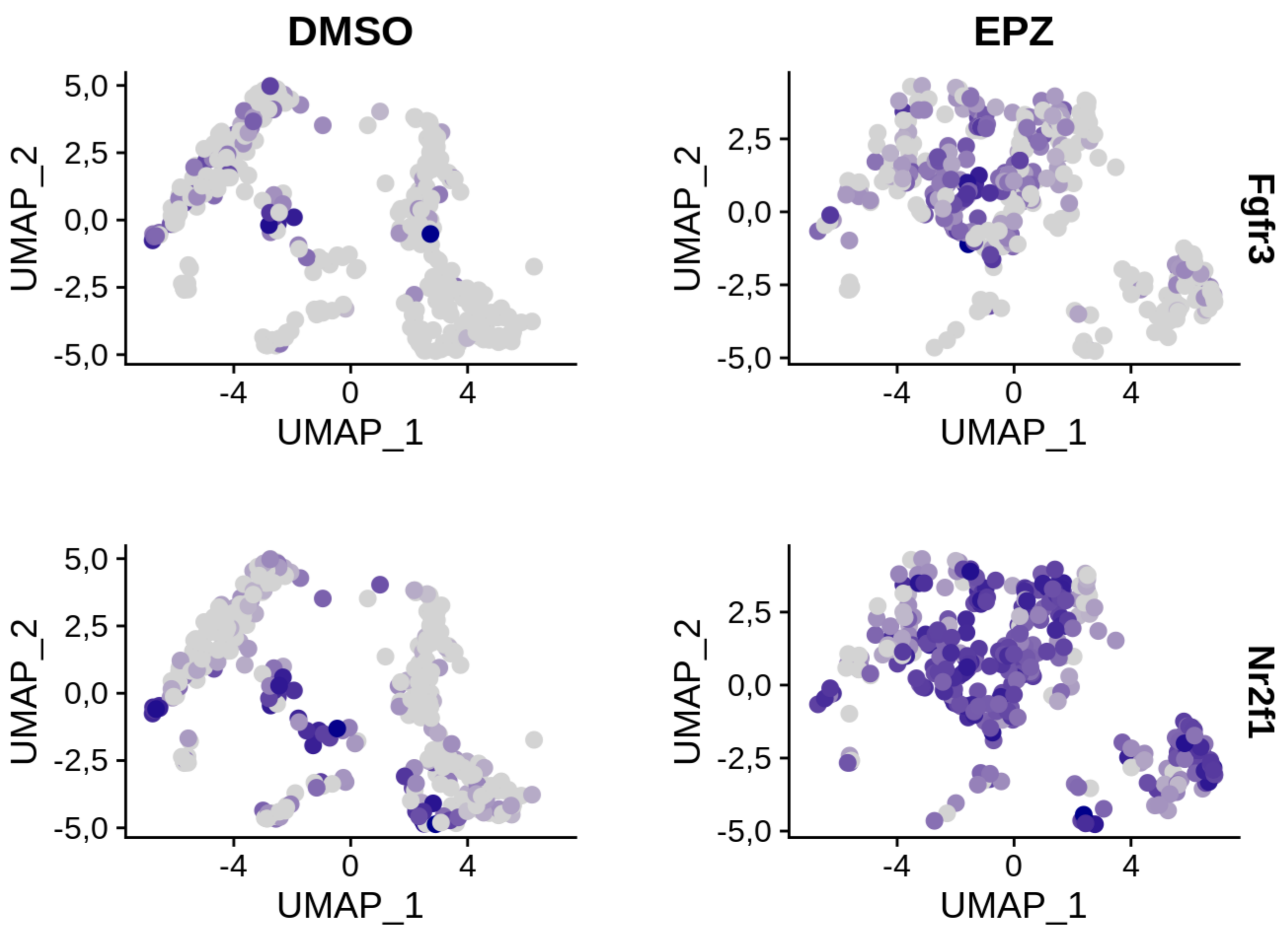


21. Plot Supplementary Figure 4B | Note: rearranged in manuscript

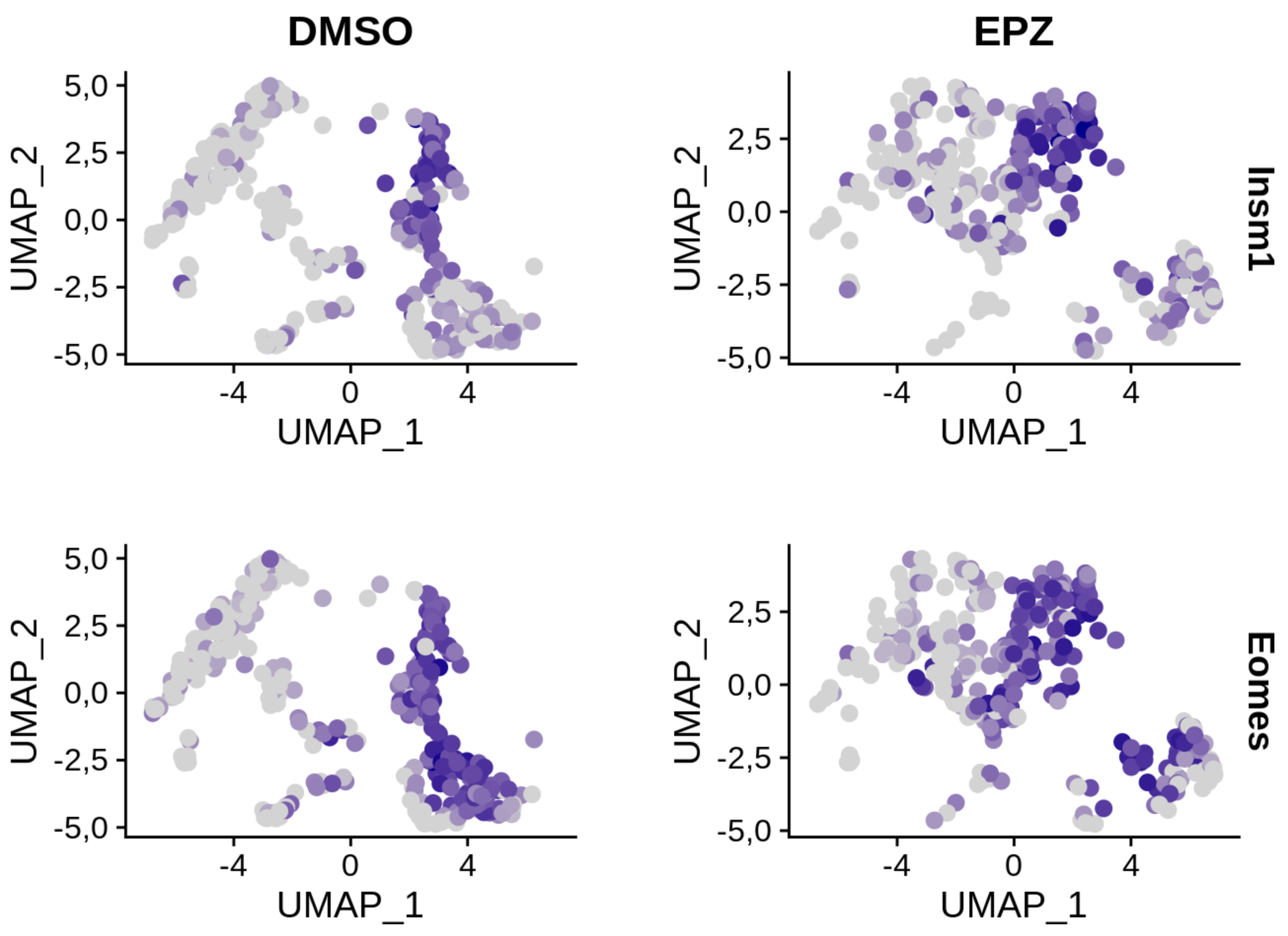
```
# Apical progenitors
FeaturePlot(sce_hvg.seurat, features = c("Hes5","Fabp7"),split.by = "condition",p
t.size = 2.5, cols = c(rep("lightgrey"), "dark blue"))
```



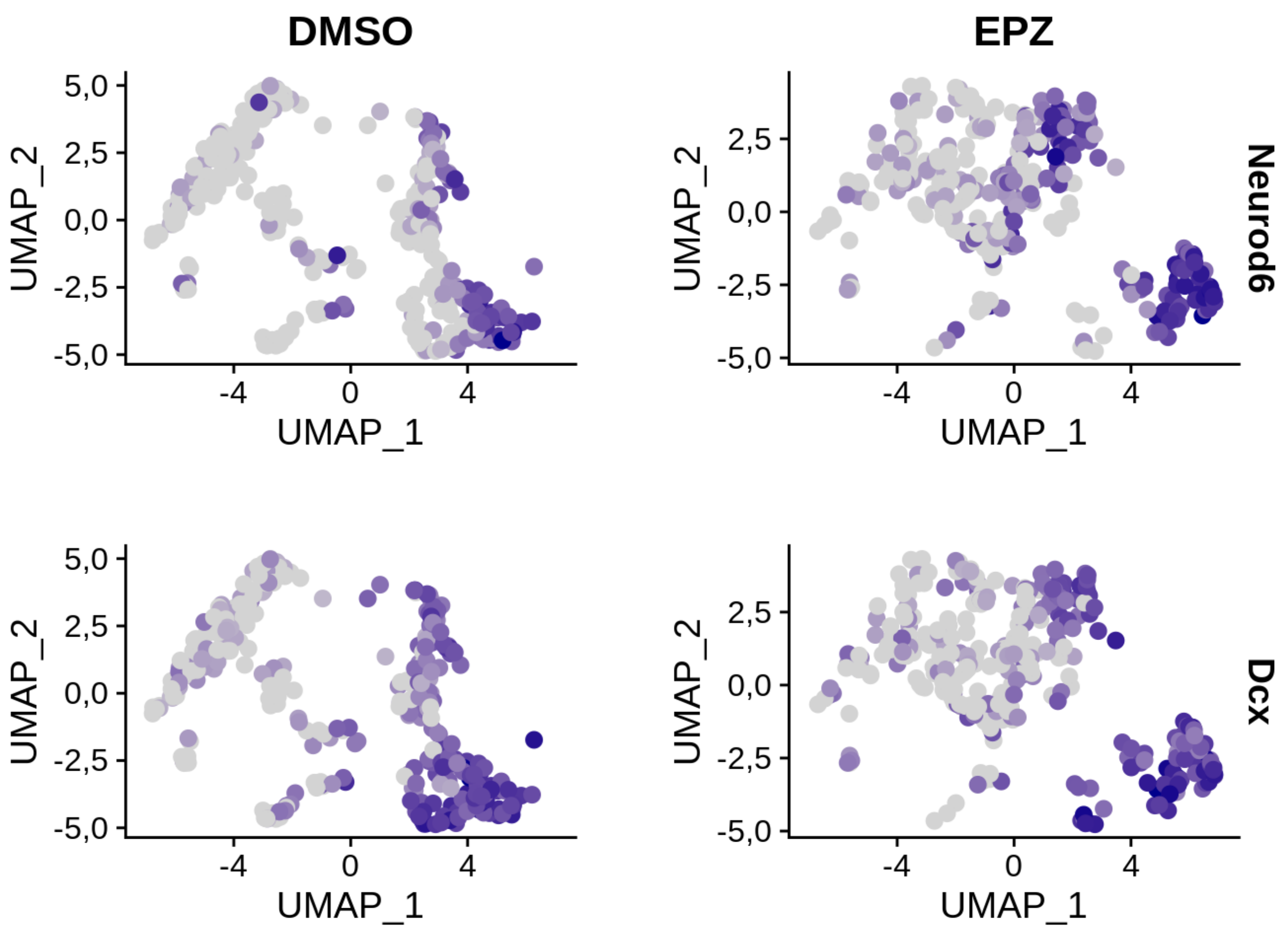
```r
# Transient transcriptional state
FeaturePlot(sce_hvg.seurat, features = c("Fgfr3","Nr2f1"),split.by = "condition",p
t.size = 2.5, cols = c(rep("lightgrey"), "dark blue"))
```



```
# Basal progenitors
FeaturePlot(sce_hvg.seurat, features = c("Insm1","Eomes"),split.by = "condition",p
t.size = 2.5, cols = c(rep("lightgrey"), "dark blue"))
```



```r
# Neurons
FeaturePlot(sce_hvg.seurat, features = c("Neurod6","Dcx"),split.by = "condition",p
t.size = 2.5, cols = c(rep("lightgrey"), "dark blue"))
```



```
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] latexpdf_0.1.6             umap_0.2.3.1
##  [3] Seurat_3.1.0               RColorBrewer_1.1-2
##  [5] mclust_5.4.5               org.Mm.eg.db_3.10.0
##  [7] AnnotationDbi_1.48.0       stringr_1.4.0
##  [9] slingshot_1.2.0            princurve_2.1.4
## [11] SC3_1.12.0                 scater_1.12.2
## [13] ggplot2_3.2.1                scran_1.12.1
## [15] mvoutlier_2.0.9              sgeostat_1.0-27
## [17] HDF5Array_1.12.1             rhdf5_2.28.0
## [19] SingleCellExperiment_1.6.0  SummarizedExperiment_1.14.0
## [21] DelayedArray_0.10.0          BiocParallel_1.18.0
## [23] matrixStats_0.55.0           Biobase_2.44.0
## [25] GenomicRanges_1.36.0         GenomeInfoDb_1.20.0
## [27] IRanges_2.18.2               S4Vectors_0.22.0
## [29] BiocGenerics_0.30.0          dplyr_0.8.3
## [31] data.table_1.12.4            readr_1.3.1
## [33] 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] cowplot_1.0.0               statmod_1.4.32
## [113] webshot_0.5.1               Rtsne_0.15
## [115] forcats_0.4.0               copula_0.999-19.1
## [117] softImpute_1.4              uwot_0.1.8
## [119] igraph_1.2.4.1              survival_2.44-1.1
## [121] numDeriv_2016.8-1.1         yaml_2.2.0
## [123] prabclus_2.3-1              htmltools_0.4.0
## [125] memoise_1.1.0               modeltools_0.2-22
## [127] locfit_1.5-9.1              viridisLite_0.3.0
## [129] digest_0.6.21               rrcov_1.4-7
## [131] assertthat_0.2.1            mime_0.7
## [133] registry_0.5-1              npsurv_0.4-0
## [135] RSQLite_2.1.2               future.apply_1.3.0
## [137] lsei_1.2-0                  blob_1.2.0
## [139] R.oo_1.22.0                 RNeXML_2.3.0
## [141] labeling_0.3                splines_3.6.1
## [143] Rhdf5lib_1.6.0              fpc_2.2-3
## [145] RCurl_1.95-4.12             cvTools_0.3.2
## [147] hms_0.5.1                   colorspace_1.4-1
## [149] BiocManager_1.30.7          ggbeeswarm_0.6.0
## [151] SDMTools_1.1-221.1          nnet_7.3-12
## [153] Rcpp_1.0.2                  ADGofTest_0.3
## [155] RANN_2.6.1                  mvtnorm_1.0-11
## [157] pspline_1.0-18              VIM_4.8.0
## [159] truncnorm_1.0-8             R6_2.4.0
## [161] grid_3.6.1                  ggridges_0.5.1
## [163] lifecycle_0.1.0             zip_2.0.4
## [165] curl_4.2                    gdata_2.18.0
## [167] leiden_0.3.1                robustbase_0.93-5
## [169] Matrix_1.2-17               howmany_0.3-1
## [171] RcppAnnoy_0.0.13            iterators_1.0.12
## [173] htmlwidgets_1.5.1           purrr_0.3.2
## [175] crosstalk_1.0.0             globals_0.12.4
## [177] openssl_1.4.1               clusterExperiment_2.4.4
## [179] codetools_0.2-16            gtools_3.8.1
## [181] prettyunits_1.0.2           gridBase_0.4-7
## [183] RSpectra_0.15-0             R.methodsS3_1.7.1
## [185] gtable_0.3.0                tsne_0.1-3
## [187] DBI_1.0.0                   dynamicTreeCut_1.63-1
## [189] httr_1.4.1                  KernSmooth_2.23-15
## [191] stringi_1.4.6               progress_1.2.2
## [193] reshape2_1.4.3              uuid_0.1-2
## [195] diptest_0.75-7              annotate_1.62.0
## [197] viridis_0.5.1               xml2_1.2.2
## [199] boot_1.3-23                 BiocNeighbors_1.2.0
## [201] ade4_1.7-13                 sROC_0.1-2
## [203] DEoptimR_1.0-8           bit_1.1-14
## [205] pkgconfig_2.0.3          gsl_2.1-6
## [207] gbRd_0.4-11              knitr_1.25
```