Running Seurat

This is a ReadMe file for the processing of single-cell RNA-seq data derived from embryonic mouse cortical interneurons Mi et al. The input of this analysis is a count matrix describing the total reads mapped to each gene feature, with slight modification. Counts derived from the same gene family have been combined using custom R script. In addition, a dataframe describing the ID of each gene is provided (feature_info.txt).

The following packages are to be loaded into the environment for this workflow:

Step 1: Import dataset into R as matrix

Firstly, locate the directory containing the count matrix and feature_info file in your computer.
Then, Import GSE109796_Oscar.GEO.singleCell.gene.count.modified.txt.gz file as a matrix.

```
setwd("~/Desktop/Seurat")
input.matrix <- read.table("GSE109796_Oscar.GEO.singleCell.gene.count.modified.txt.gz", header = TRUE, s

# preview matrix
input.matrix[1:5,1:5]
```

```
##       C1.101.A10_CGAGGCTG.GCGTAAGA_L008_R1_all
## GNAI3                                        0
## PBSN                                         0
## CDC45                                        0
## H19                                        189
## SCML2                                        0
##       C1.101.A1_TAAGGCGA.GCGTAAGA_L008_R1_all
## GNAI3                                       0
## PBSN                                        0
## CDC45                                       0
## H19                                         1
## SCML2                                       0
##       C1.101.A4_TCCTGAGC.GCGTAAGA_L008_R1_all
## GNAI3                                       0
## PBSN                                        0
## CDC45                                    1439
## H19                                        84
## SCML2                                       0
##       C1.101.A5_GGACTCCT.GCGTAAGA_L008_R1_all
## GNAI3                                       1
## PBSN                                        0
## CDC45                                      31
## H19                                       151
## SCML2                                       0
##       C1.101.A6_TAGGCATG.GCGTAAGA_L008_R1_all
## GNAI3                                       0
## PBSN                                        0
## CDC45                                       0
## H19                                      2231
## SCML2                                          0
```

Step 2: Clean-up dataset

Dataset will have to be filtered to select for high quality features and cells.
3 types of filters will be applied:

Filter 1: Select coding genes We need to download and import a recent mouse GRCm38 annotation to parse data for gene_ids of coding genes

```
setwd("~/Desktop/Seurat")
mouse.gtf <- rtracklayer::import("Mus_musculus.GRCm38.101.gtf.gz")

#F: first things first, you can preview the type of genes in the annotation
unique(mouse.gtf$gene_biotype)
```

```
##  [1] "TEC"                                "snRNA"
##  [3] "protein_coding"                     "processed_pseudogene"
##  [5] "antisense"                          "sense_intronic"
##  [7] "lincRNA"                            "processed_transcript"
##  [9] "miRNA"                              "snoRNA"
## [11] "misc_RNA"                           "transcribed_unprocessed_pseudogene"
## [13] "unprocessed_pseudogene"             "sense_overlapping"
## [15] "rRNA"                               "transcribed_processed_pseudogene"
## [17] "ribozyme"                           "unitary_pseudogene"
## [19] "scaRNA"                             "pseudogene"
## [21] "polymorphic_pseudogene"             "bidirectional_promoter_lncRNA"
## [23] "transcribed_unitary_pseudogene"     "macro_lncRNA"
## [25] "3prime_overlapping_ncRNA"           "translated_unprocessed_pseudogene"
## [27] "TR_V_gene"                          "TR_V_pseudogene"
## [29] "TR_D_gene"                          "TR_J_gene"
## [31] "TR_C_gene"                          "TR_J_pseudogene"
## [33] "IG_LV_gene"                         "IG_V_gene"
## [35] "IG_V_pseudogene"                    "IG_J_gene"
## [37] "IG_C_gene"                          "sRNA"
## [39] "scRNA"                              "IG_C_pseudogene"
## [41] "IG_D_gene"                          "IG_D_pseudogene"
## [43] "IG_pseudogene"                      "Mt_tRNA"
## [45] "Mt_rRNA"
```

```
#F: As you can see, there are alot of non-coding genes in the annotation.
# it sort of make more sense to select only protein_coding genes since it makes up the bulk of the anno

#F: Here, I am going to generate a table on the number of genes in each gene_biotype category
# You don't need to understand it, but maybe you could try to find out what does "%>%" does
mouse.gtf %>%
  as.data.frame() %>%
  filter(type == "gene") %>%
  group_by(gene_biotype) %>%
  tally() %>%
  arrange(desc(n))
```

```
## # A tibble: 45 x 2
##    gene_biotype                n
##    <chr>                   <int>
##  1 protein_coding          21936
##  2 processed_pseudogene    10003
##  3 lincRNA                  5629
##  4 TEC                      3238
##  5 antisense                2991
##  6 unprocessed_pseudogene   2723
##  7 miRNA                    2207
##  8 snoRNA                   1507
##  9 snRNA                    1385
## 10 processed_transcript      779
## # ... with 35 more rows
```

```
#F: Make a dataframe containing gene_name and gene_id of protein coding genes
coding.genes <- mouse.gtf %>%
  as.data.frame() %>%
  dplyr::select(gene_name, gene_id, gene_biotype) %>%
  filter(gene_biotype == "protein_coding") %>%
    distinct()
```

Next, we will make use of the gene_id from the feature_info.txt file to select for protein coding genes

```
setwd("~/Desktop/Seurat")
features.df <- read.table("feature_info.txt", header = TRUE, sep = "\t", stringsAsFactors = F)

# 3+2) add a new column to annotate if the gene_id corresponds to a coding gene
features.df <- features.df %>%
  mutate(coding = ifelse(gene_id %in% coding.genes$gene_id,T,F))
head(features.df)
```

```
##              gene_id gene_name coding
## 1 ENSMUSG00000000001     GNAI3   TRUE
## 2 ENSMUSG00000000003      PBSN   TRUE
## 3 ENSMUSG00000000028     CDC45   TRUE
## 4 ENSMUSG00000000031       H19  FALSE
## 5 ENSMUSG00000000037     SCML2   TRUE
## 6 ENSMUSG00000000049      APOH   TRUE
```

```
#F: you can calculate how many features are retained
sum(features.df$coding)
```

```
## [1] 20025
```

```
#4) subset input.matrix to keep only coding genes
input.matrix <- input.matrix[features.df$coding,]
nrow(input.matrix)  # to check if the number of features is reduced
```

```
## [1] 20025
```

```


ncol(input.matrix)  #before filtering
```

Filter 2: filter for cells that contain at least 50000 reads mapping to coding features(which we have already subsetted)

```
## [1] 2669
```

```
input.matrix <- input.matrix[,colSums2(as.matrix(input.matrix)) > 50000]
ncol(input.matrix)  #after filtering
```

```
## [1] 2658
```

Filter 3: Remove features that are expressed in less than 10 cells with less than 5CPM 5CPM means that instead of counts, each feature in each gene have been normalized. So we will use Seurat NormalizeData function to get this normalized value, but we will later re-normalize after filtering is done.

```
testdata <- CreateSeuratObject(counts = input.matrix, project = "filter")
testdata <- NormalizeData(testdata, normalization.method = "LogNormalize", scale.factor = 1000000)

# we will select features that have 10 cells with at least than 5CPM expression
features.to.keep <-  apply(testdata[["RNA"]]@counts, 1, function(x){sum(x>=5) >=10})

# and then filter the input.matrix
input.matrix <- input.matrix[features.to.keep,]
input.matrix[1:5,1:5]
```

```
##       C1.101.A10_CGAGGCTG.GCGTAAGA_L008_R1_all
## GNAI3                                         0
## CDC45                                         0
## SCML2                                         0
## NARF                                        193
## KLF6                                         13
##       C1.101.A1_TAAGGCGA.GCGTAAGA_L008_R1_all
## GNAI3                                        0
## CDC45                                        0
## SCML2                                        0
## NARF                                         0
## KLF6                                         0
##       C1.101.A4_TCCTGAGC.GCGTAAGA_L008_R1_all
## GNAI3                                        0
## CDC45                                     1439
## SCML2                                        0
## NARF                                         0
## KLF6                                         0
##       C1.101.A5_GGACTCCT.GCGTAAGA_L008_R1_all
## GNAI3                                        1
## CDC45                                       31
## SCML2                                        0
## NARF                                         1
## KLF6                                         0
##       C1.101.A6_TAGGCATG.GCGTAAGA_L008_R1_all
## GNAI3                                        0
## CDC45                                        0
## SCML2                                        0
## NARF                                         0
## KLF6                                         0
```

Step 3: Create Seurat Object

The filtered matrix can now be converted into a Seurat object for downstream analysis

```
mydata <- CreateSeuratObject(counts = input.matrix, min.cells = 3, min.genes = 200, project = "interneu
mydata
```

```
## An object of class Seurat
## 12324 features across 2658 samples within 1 assay
## Active assay: RNA (12324 features, 0 variable features)
```

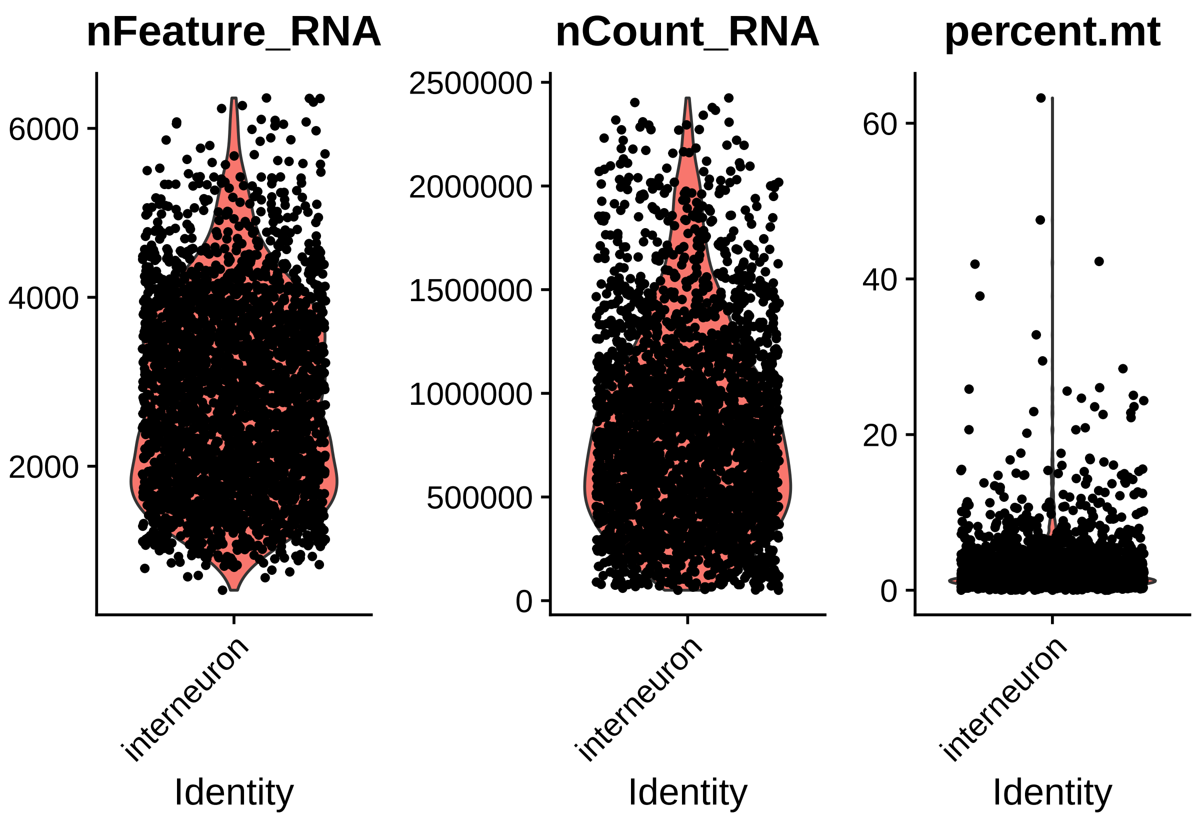
Step 4: Perform QC on dataset and subset data further

Here, we want to check the quality of the remaining cells and quantify its proportion of mitochondrial genes. This metric will provide a good indication as to whether the cell/GEM is indeed a cell.

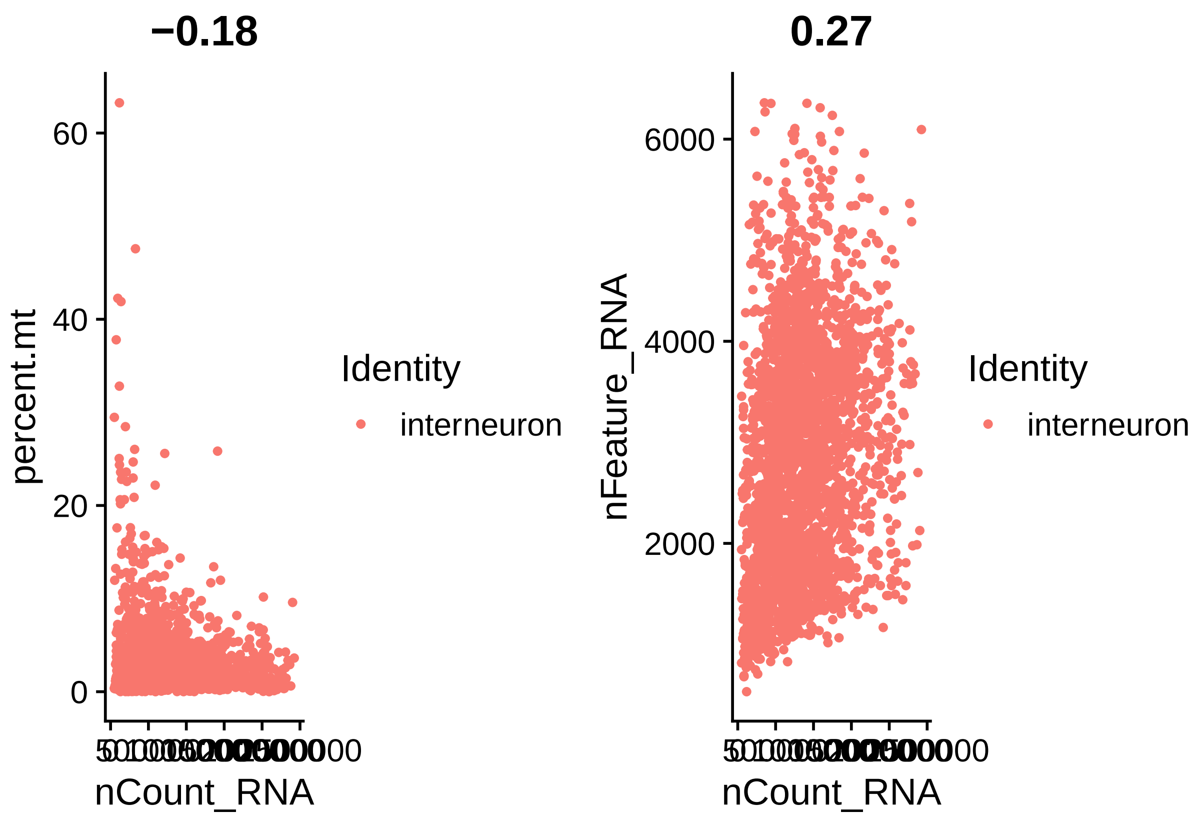
```
mydata[["percent.mt"]] <- PercentageFeatureSet(mydata, pattern = "^MT-") #F: I changed the pattern, sin
head(mydata@meta.data, 5)
```

```
##                                          orig.ident nCount_RNA nFeature_RNA
## C1.101.A10_CGAGGCTG.GCGTAAGA_L008_R1_all interneuron     627853         2841
## C1.101.A1_TAAGGCGA.GCGTAAGA_L008_R1_all  interneuron     997851         2523
## C1.101.A4_TCCTGAGC.GCGTAAGA_L008_R1_all  interneuron    1432057         2955
## C1.101.A5_GGACTCCT.GCGTAAGA_L008_R1_all  interneuron    1185941         3033
## C1.101.A6_TAGGCATG.GCGTAAGA_L008_R1_all  interneuron     537005         3013
##                                          percent.mt
## C1.101.A10_CGAGGCTG.GCGTAAGA_L008_R1_all  0.4473977
## C1.101.A1_TAAGGCGA.GCGTAAGA_L008_R1_all   1.8498754
## C1.101.A4_TCCTGAGC.GCGTAAGA_L008_R1_all   2.1534059
## C1.101.A5_GGACTCCT.GCGTAAGA_L008_R1_all   2.6126089
## C1.101.A6_TAGGCATG.GCGTAAGA_L008_R1_all   0.9106060
```

```
VlnPlot(mydata, features = c("nFeature_RNA", "nCount_RNA", "percent.mt"), ncol = 3)
```



```r
plot1 <- FeatureScatter(mydata, feature1 = "nCount_RNA", feature2 = "percent.mt")
plot2 <- FeatureScatter(mydata, feature1 = "nCount_RNA", feature2 = "nFeature_RNA")
plot1 + plot2
```



Lastly, we will subset the data further by choosing cells with less than 5% mitochondrial genes

```
mydata <- subset(mydata, subset = nFeature_RNA > 300 & nFeature_RNA < 6000 & percent.mt < 5)
```

Step 5: Normalize and scale matrix

As the name implies, we want to normalize the data so that features can be compared between cells.

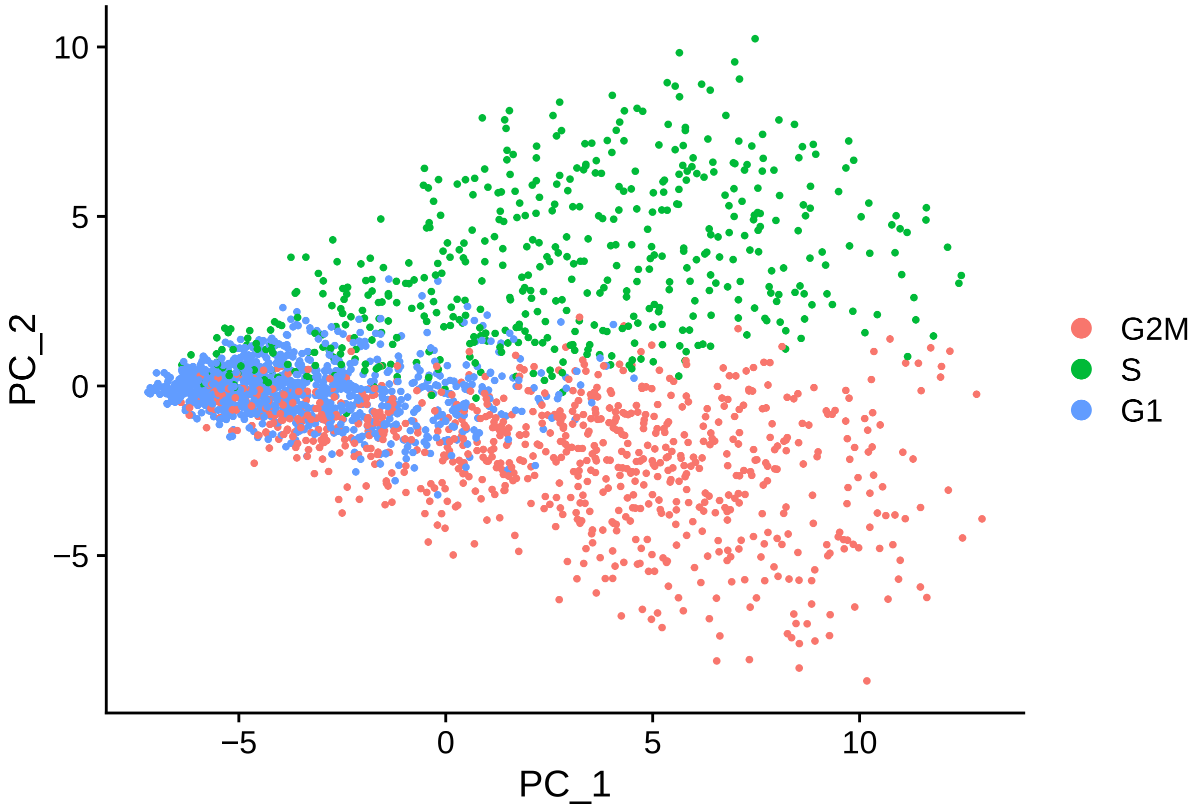
```
mydata <- NormalizeData(mydata, normalization.method = "LogNormalize", scale.factor = 1000000)
mydata <- ScaleData(mydata)
```

Step 6: Regress out cell cycle determinants

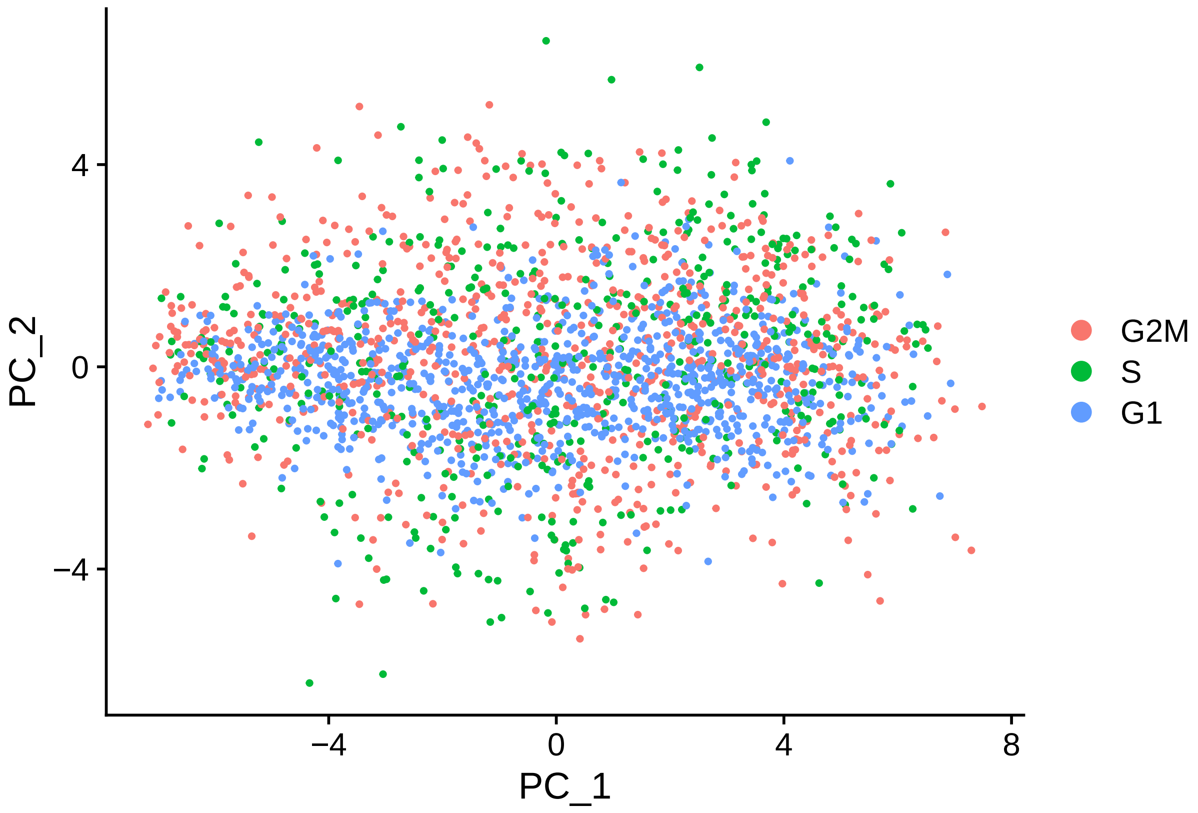
It is very common for cells to cluster based on its mitotic stage. This variable have to be removed/regressed to allow accurate identification of Highly Variable Genes

```
## Collect s and g2m genes
s.genes <- cc.genes$s.genes
g2m.genes <- cc.genes$g2m.genes

# Score Cell cycle genes
mydata <- CellCycleScoring(mydata, s.features = s.genes, g2m.features = g2m.genes, set.ident = TRUE)

# Run a PCA plot to see if cells cluster based on mitotic phase
```

```
mydata <- RunPCA(mydata, features = c(s.genes, g2m.genes))
DimPlot(mydata, reduction = "pca")
```



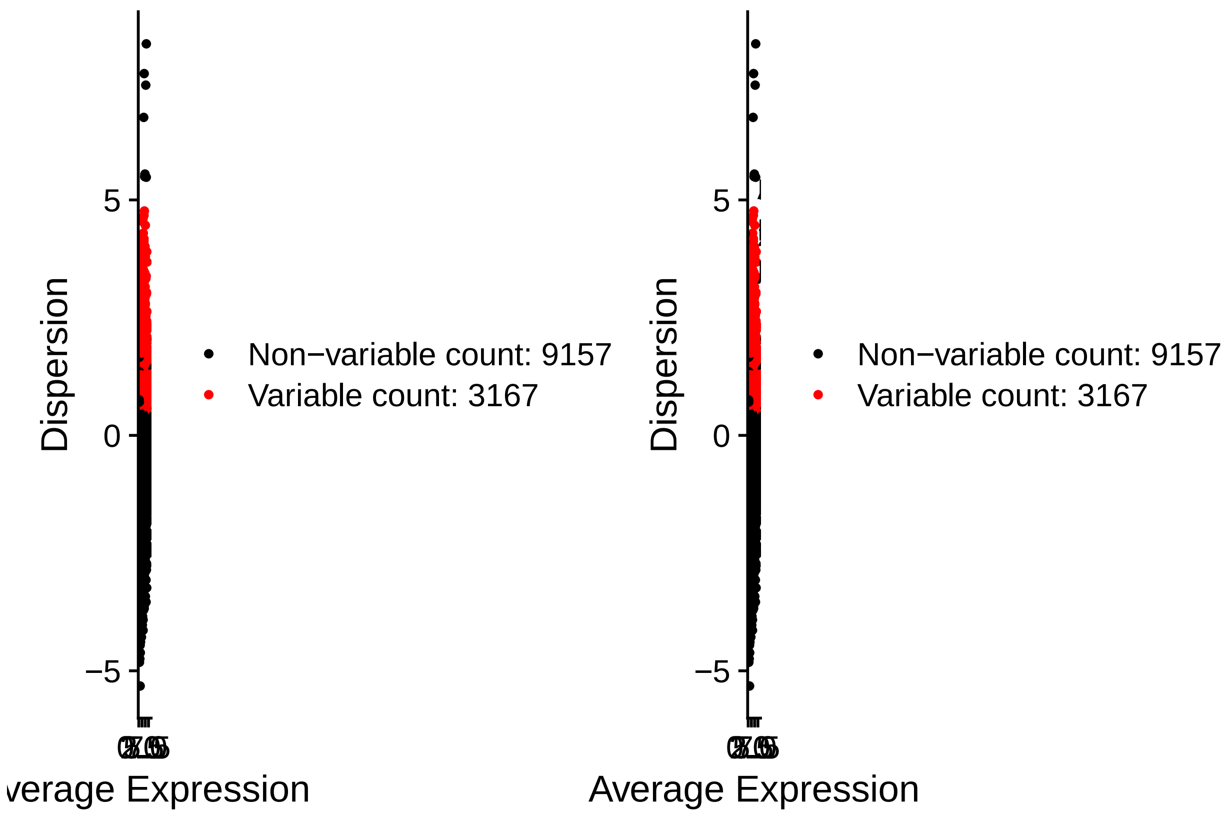
```
# Regress out cell cycle factors
mydata <- ScaleData(mydata, vars.to.regress = c("S.Score", "G2M.Score"), features = rownames(mydata))

# Re-check PCA plot
mydata <- RunPCA(mydata, features = c(s.genes, g2m.genes))
DimPlot(mydata, reduction = "pca")
```



Step 7: Identify Highly Variable Genes and Scale data

A list of Highly Variable Genes (HVG) will be generated and to be used for downstream analysis including dimensional reduction and clustering.
Also, data will be scaled so that values for each feature is comparable

```
#F: THe authors used a different method for selection.method.
mydata <- FindVariableFeatures(mydata, selection.method = "mvp", mean.cutoff = c(0.5,8), dispersion.cut
top10 <- head(VariableFeatures(mydata), 10)
plot1 <- VariableFeaturePlot(mydata)
plot2 <- LabelPoints(plot = plot1, points = top10, repel = TRUE)
plot1 + plot2
```



```r
#F: I scaled the data based on highly variable features
mydata <- ScaleData(mydata)
```

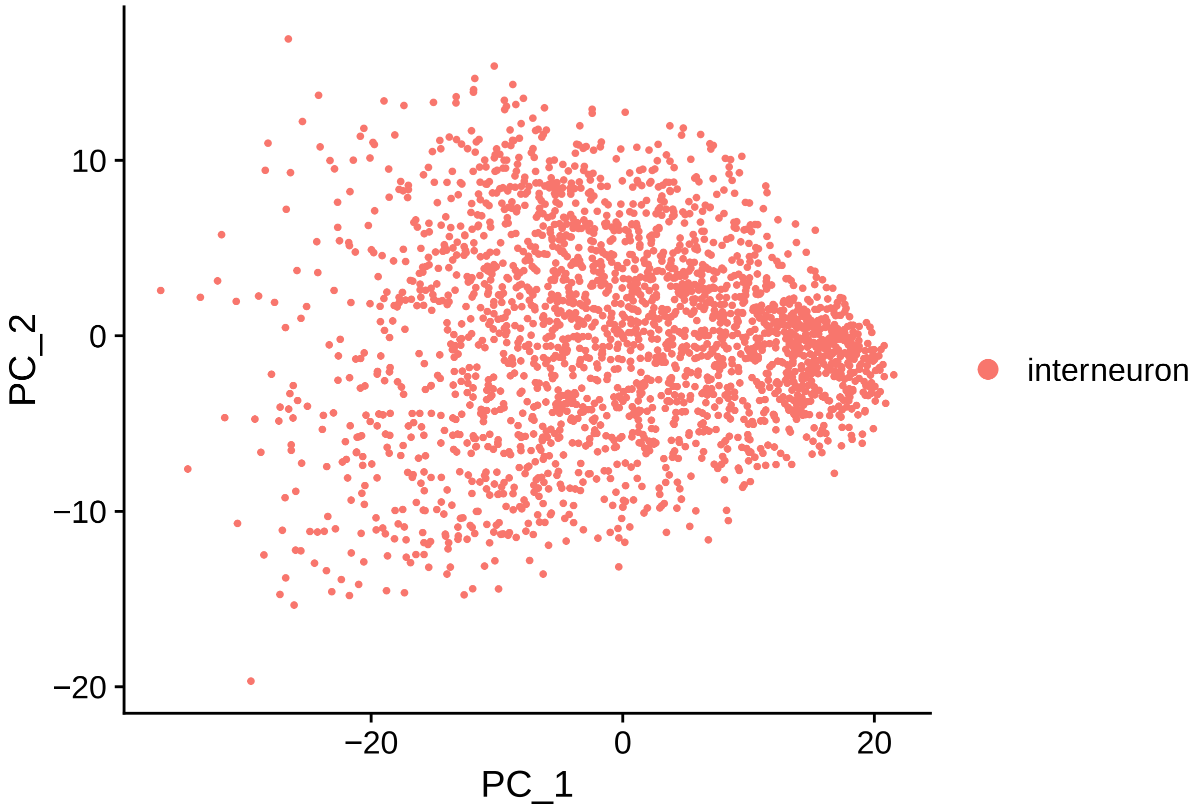
Step 8: Carry out linear dimensional reduction

This serves as a means determine the relationship between cells from the dataset, using expression patterns from the HVG

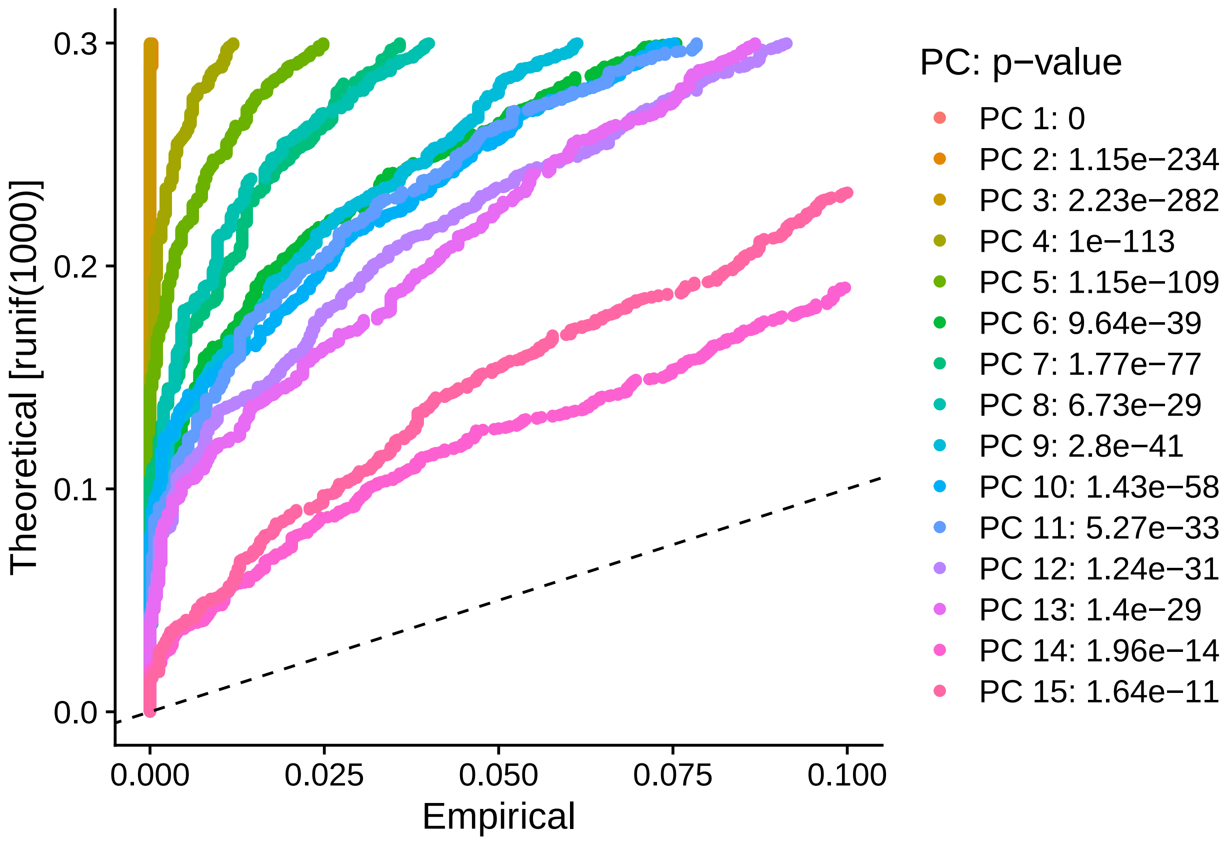
```r
mydata <- RunPCA(mydata, features = VariableFeatures(object = mydata))

# check top features in each principal component
print(mydata[["pca"]], dims = 1:5, nfeatures = 5)
```

```
## PC_ 1
## Positive:  GM10801, SLC17A9, VWA1, OPRM1, RAB37
## Negative:  SRSF6, PUF60, CCT8, CAPRIN1, SF3B1
## PC_ 2
## Positive:  DCX, RUNX1T1, NRXN3, GAD1, GAD2
## Negative:  HAT1, RFC4, CENPK, RANBP1, SYCE2
## PC_ 3
## Positive:  DPYSL2, GM10801, PTPRS, PTP4A2, PFAS
## Negative:  1500032L24RIK, SPCS2, DYNLL2, BCAS2, DDA1
## PC_ 4
## Positive:  ELAVL4, LHX6, CENPE, MKI67, PFN2
## Negative:  FLT1, IGFBP7, DLC1, COL4A1, MICAL2
## PC_ 5
## Positive:  CENPF, BPIFB5, SPC25, MIS18BP1, RAB37
## Negative:  SEPT2, CSDE1, MFGE8, CDK6, TRP53
```

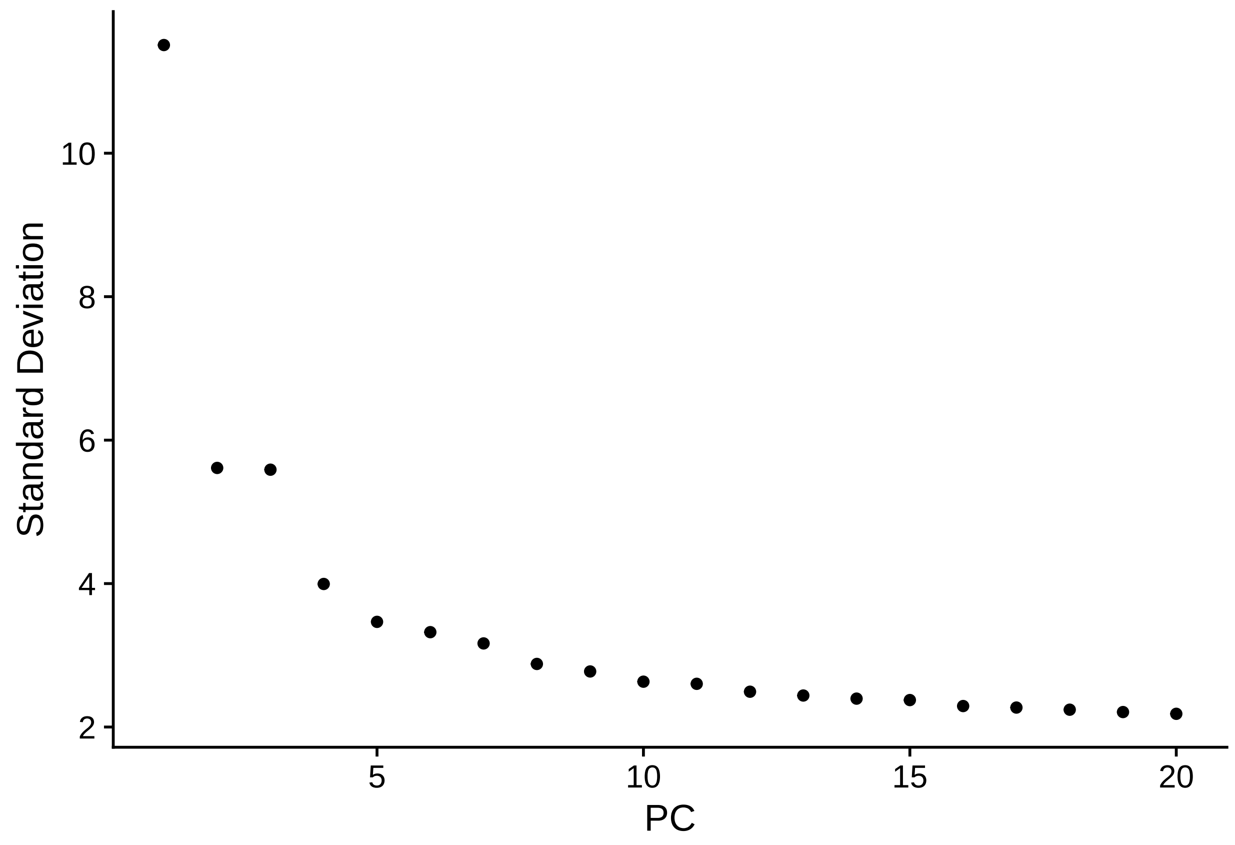
```
# plot PCA
DimPlot(mydata, reduction = "pca", group.by = "old.ident")
```



```
# Inspect statistics of principal compenents
mydata <- JackStraw(mydata, num.replicate = 100)
mydata <- ScoreJackStraw(mydata, dims = 1:20)
JackStrawPlot(mydata, dims = 1:15)
```



```
ElbowPlot(mydata)
```

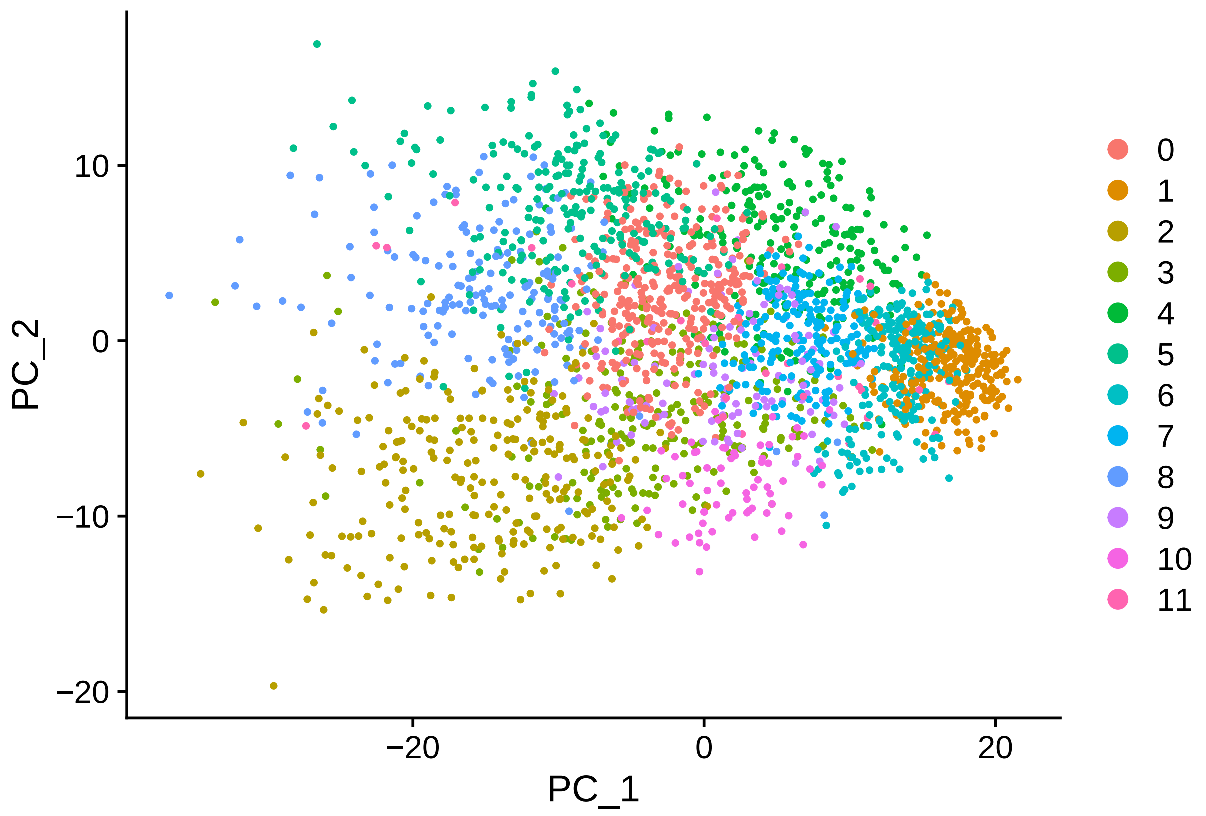


Step 9: Cluster cells

```
mydata <- FindNeighbors(mydata, dims = 1:20)
mydata <- FindClusters(mydata, resolution = 1)
```

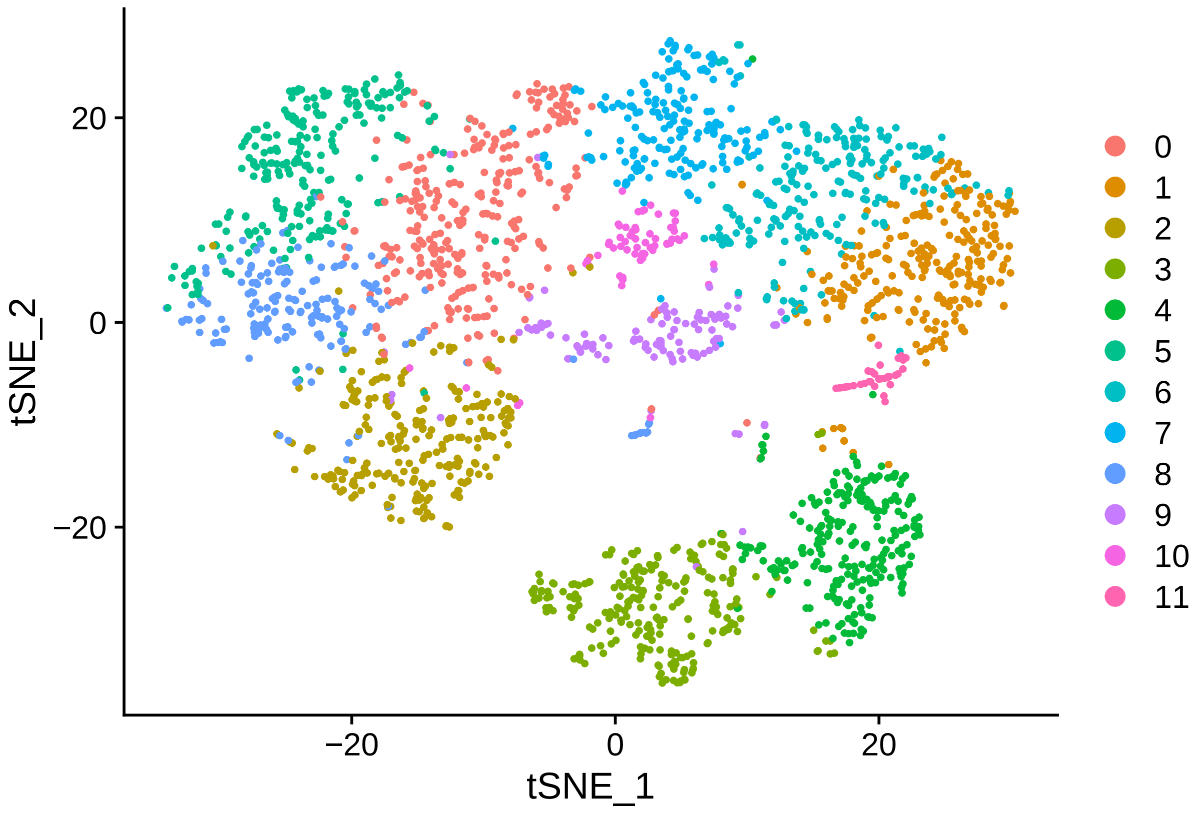
```
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 2347
## Number of edges: 84153
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.7604
## Number of communities: 12
## Elapsed time: 0 seconds
```

```
# colour code cells based on cluster
DimPlot(mydata, reduction = "pca")
```

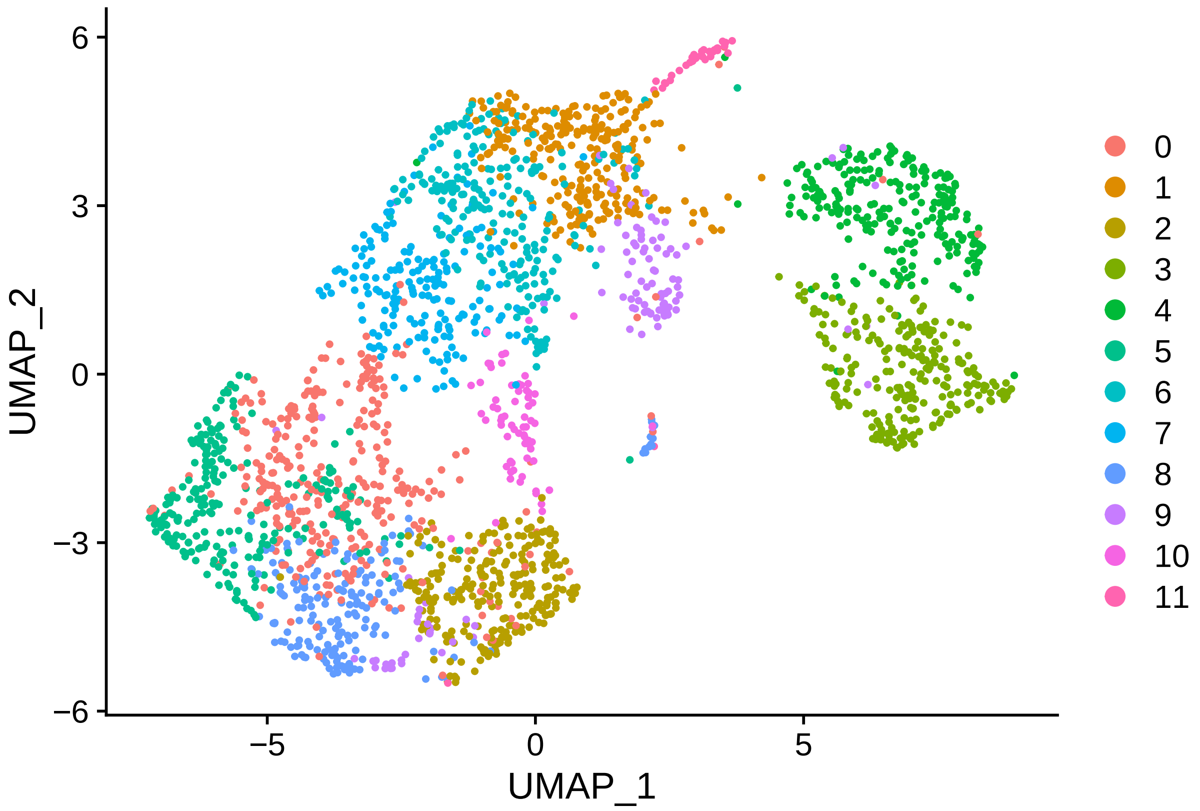


## Step 10: Carry out non-linear dimensional reduction

```
mydata <- RunTSNE(mydata, dims = 1:10)
mydata <- RunUMAP(mydata, dims = 1:10)


DimPlot(mydata, reduction = "tsne")
```

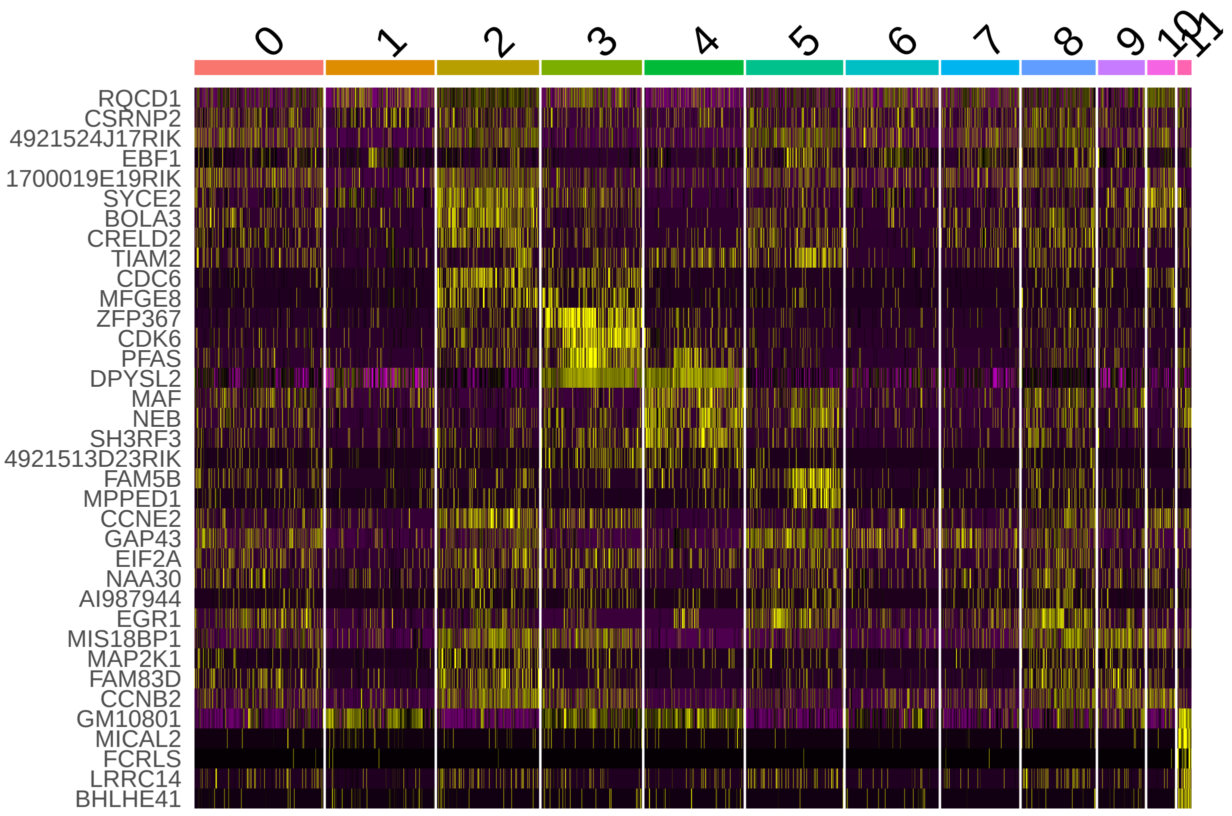


```
DimPlot(mydata, reduction = "umap")
```



Step 10: Find Biomarkers

```
mydata.markers <- FindAllMarkers(mydata, test.use = "negbinom", only.pos = TRUE, min.pct = 0.25, logfc.
top10 <- mydata.markers %>% group_by(cluster) %>% top_n(n = 10, wt = avg_logFC)
DoHeatmap(mydata, features = top10$gene) + NoLegend()
```


0
1
2
3
4
5
6
7
8
9
10
11
RQCD1
CSRNP2
4921524J17RIK
EBF1
1700019E19RIK
SYCE2
BOLA3
CRELD2
TIAM2
CDC6
MFGE8
ZFP367
CDK6
PFAS
DPYSL2
MAF
NEB
SH3RF3
4921513D23RIK
FAM5B
MPPED1
CCNE2
GAP43
EIF2A
NAA30
AI987944
EGR1
MIS18BP1
MAP2K1
FAM83D
CCNB2
GM10801
MICAL2
FCRLS
LRRC14
BHLHE41