UMAP on murine fetal brain 10k cells

Nur-Taz Rahman, PhD

Feb 12, 2020

####The clustering and annotation of clusters shown here are only to illustrate how to use the Seurat package as a tool to analyze and visualize single-cell RNA-seq datasets. The analyses presented here are not meant to be used/considered as accurate representations of clustering and cluster annotations of any kind neurological/biological data. The results shown in these analyses are purely for the purpose of exercise and may nor may not have biological relevance.

Loading necessary packages

```
library (dplyr, quietly = TRUE)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library (Seurat)
```

Direct R to the directory you want to work in

```
setwd ("/Users/nr267/Desktop/All Work/Classes/Spring 2020/Seurat_class_resources/")
```

Load data and carry out QC

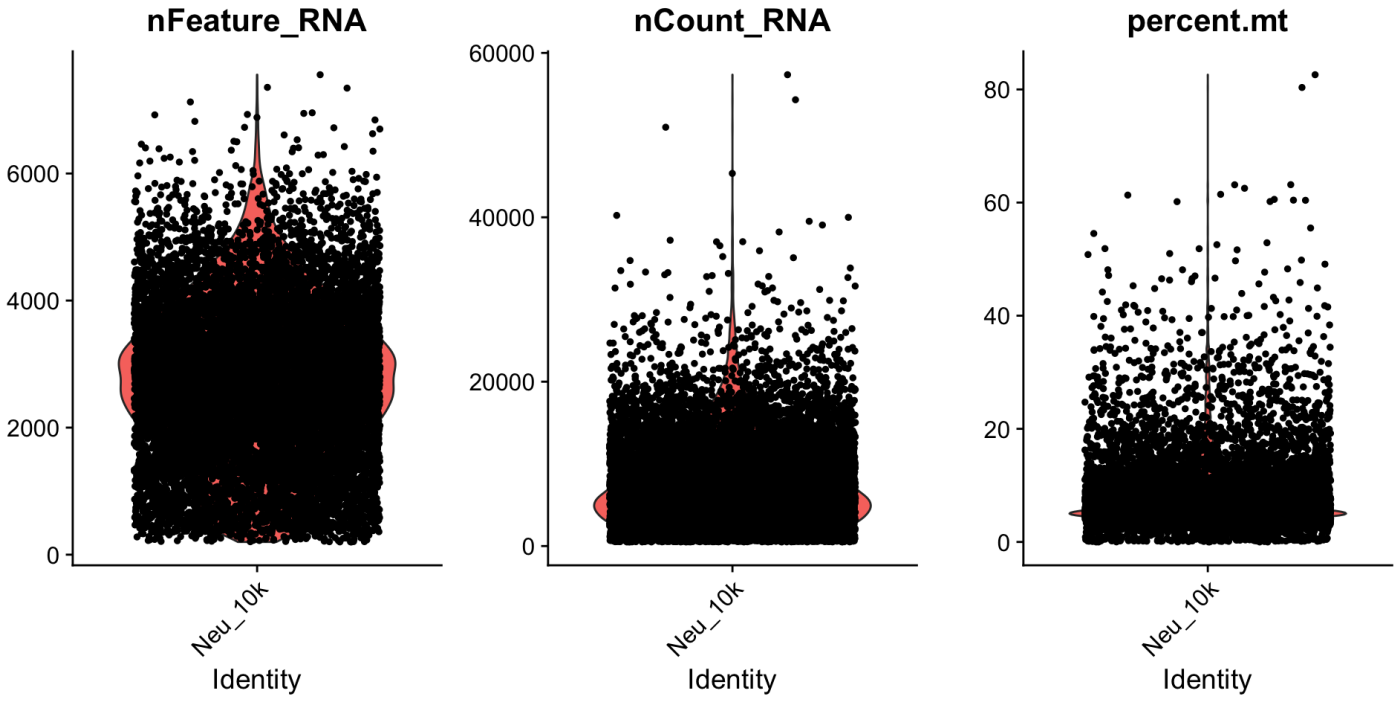
```
Neu_10k <- Read10X(data.dir = "../Seurat_class_resources/filtered_feature_bc_matrix_10k_
neuron/10k_neuron/")
Neu_10k <- CreateSeuratObject(counts = Neu_10k, project = "Neu_10k", min.cells = 3, min.
features = 200)

Neu_10k
```

```
## An object of class Seurat
## 18564 features across 11648 samples within 1 assay
## Active assay: RNA (18564 features)
```

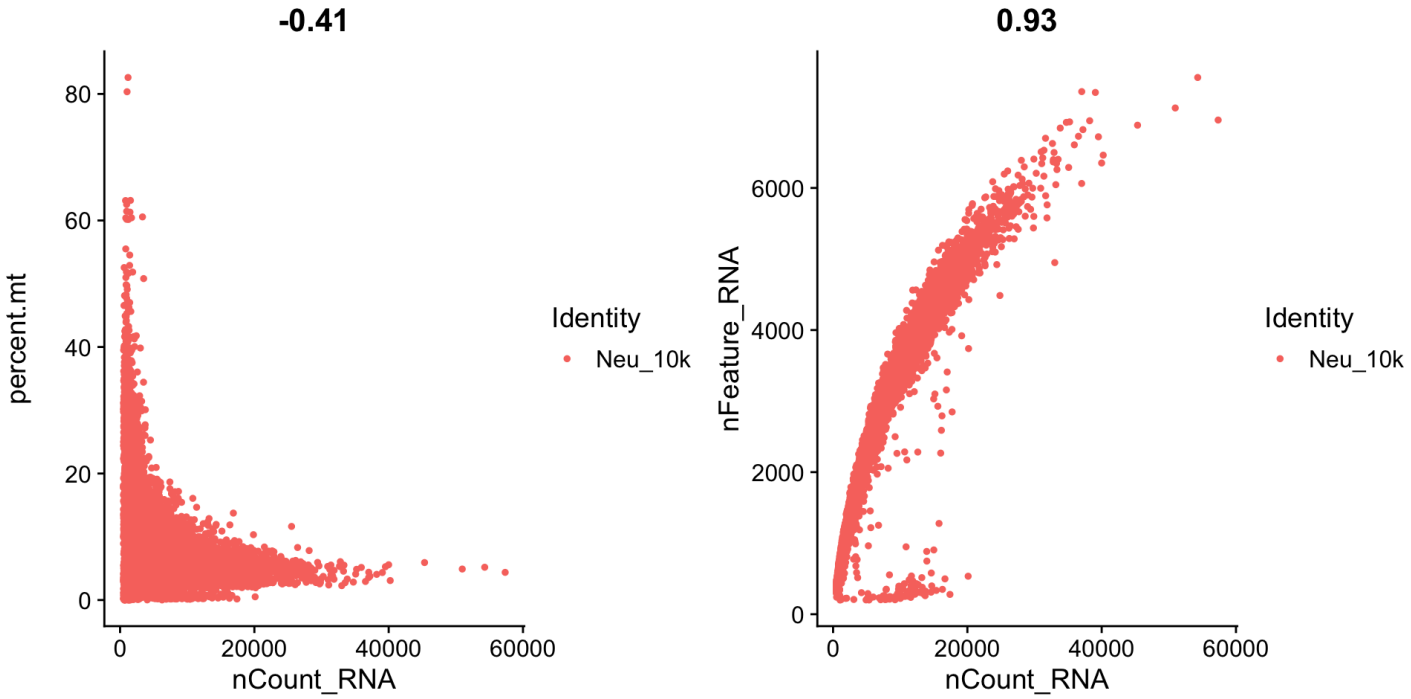
```
Neu_10k[["percent.mt"]] <- PercentageFeatureSet(Neu_10k, pattern = "^mt-")

head (Neu_10k@meta.data, 10)
```

```
##                  orig.ident nCount_RNA nFeature_RNA percent.mt
## AAACCCAAGCAACTCT    Neu_10k       1421          784  0.2111189
## AAACCCACACGCGGTT    Neu_10k       5832         2399 11.9684499
## AAACCCACAGCATACT    Neu_10k       2977         1775 12.1598925
## AAACCCACATACCATG    Neu_10k       3599         1934  6.6129480
## AAACCCAGTCGCACAC    Neu_10k       9152         3473  4.9388112
## AAACCCAGTGCACATT    Neu_10k       3889         1733 12.7796349
## AAACCCAGTGGTAATA    Neu_10k       5206         2314  3.5920092
## AAACCCATCAGGTAAA    Neu_10k       3844         1952  6.1914672
## AAACGAAAGCAATAAC    Neu_10k       5934         2604  3.7074486
## AAACGAAAGGACAAGA    Neu_10k       2536         1410  9.9763407
```

```
VlnPlot (Neu_10k, features = c("nFeature_RNA", "nCount_RNA", "percent.mt"), ncol = 3)
```




```
plot1 <- FeatureScatter(Neu_10k, feature1 = "nCount_RNA", feature2 = "percent.mt")
plot2 <- FeatureScatter(Neu_10k, feature1 = "nCount_RNA", feature2 = "nFeature_RNA")

CombinePlots(plots = list(plot1, plot2))
```



Pre-process data

```
Neu_10k <- subset(Neu_10k, subset = nFeature_RNA > 200 & nFeature_RNA < 6000 & percent.m
t < 10)
```

Check data dimensions after subsetting

```
Neu_10k
```

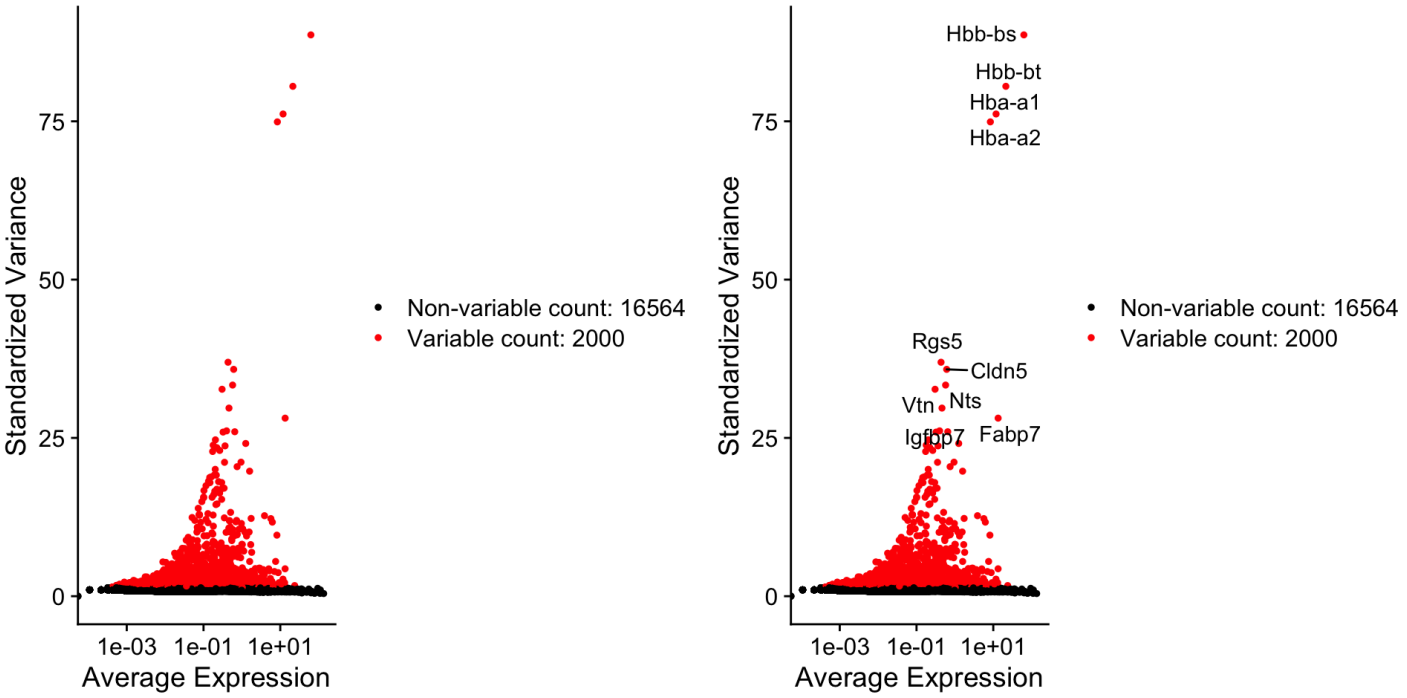
```
## An object of class Seurat
## 18564 features across 9276 samples within 1 assay
## Active assay: RNA (18564 features)
```

LogNormalize Data (normalize count data per cell and transform to log scale)

```
Neu_10k <- NormalizeData(Neu_10k, verbose = TRUE)
```

Pre-process data for PCA

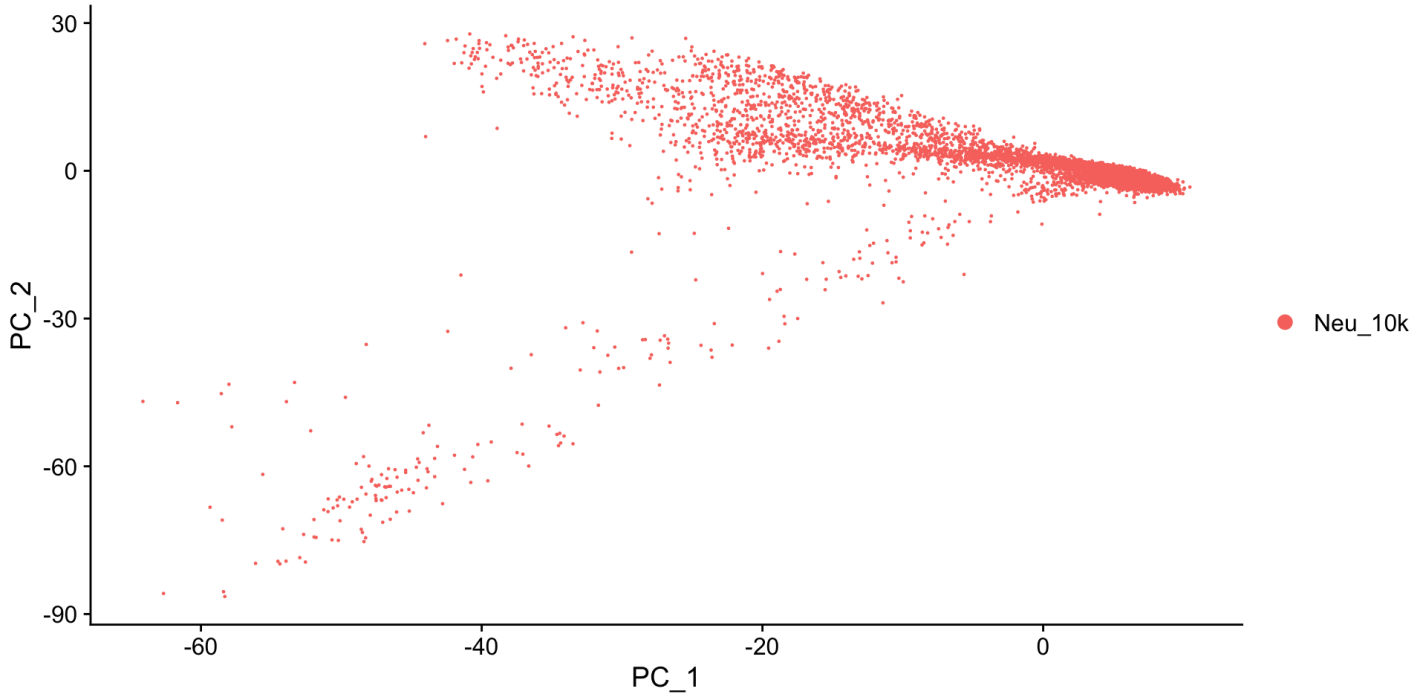
```
Neu_10k <- FindVariableFeatures(Neu_10k, selection.method = "vst", nfeatures = 2000)

top10 <- head(VariableFeatures(Neu_10k), 10)

plot3 <- VariableFeaturePlot(Neu_10k)

plot4 <- LabelPoints(plot = plot3, points = top10, repel = TRUE)

CombinePlots(plots = list(plot3, plot4))
```



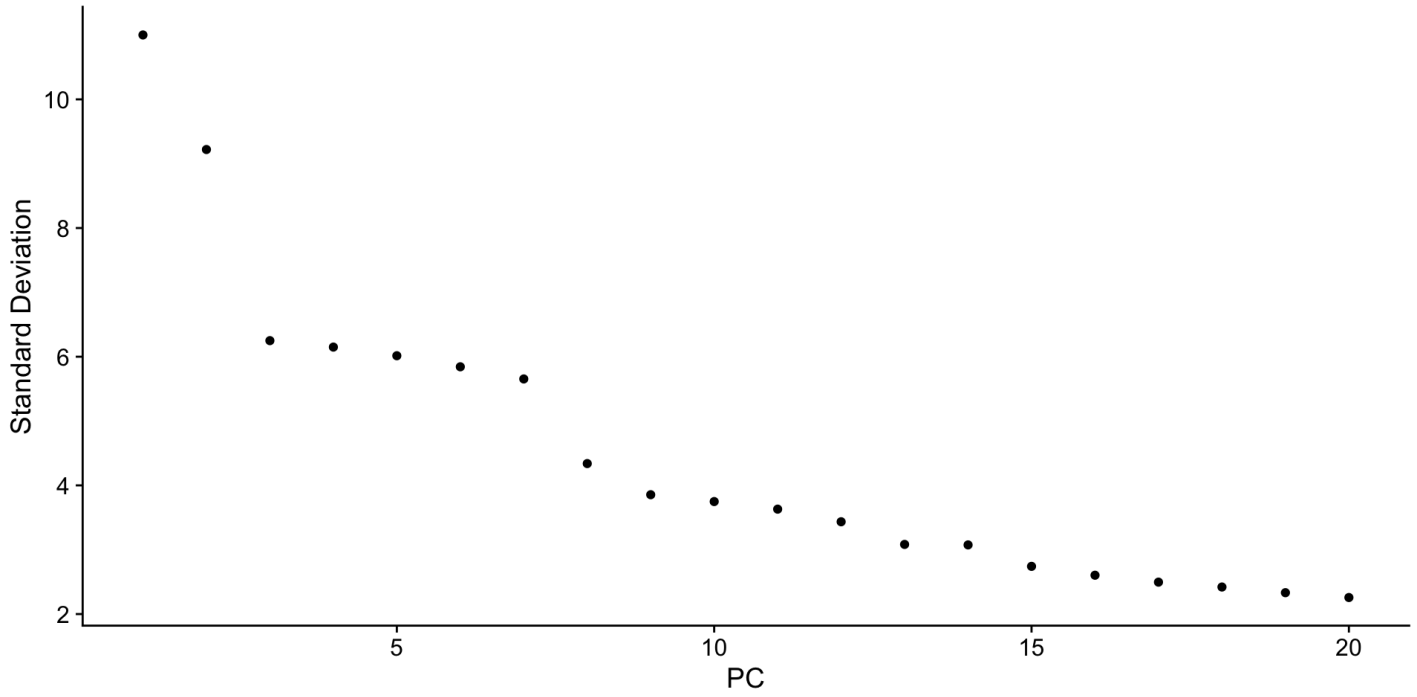

```
all.genes <- rownames(Neu_10k)

Neu_10k <- ScaleData(Neu_10k, features = all.genes)

Neu_10k
```

```
## An object of class Seurat
## 18564 features across 9276 samples within 1 assay
## Active assay: RNA (18564 features)
```

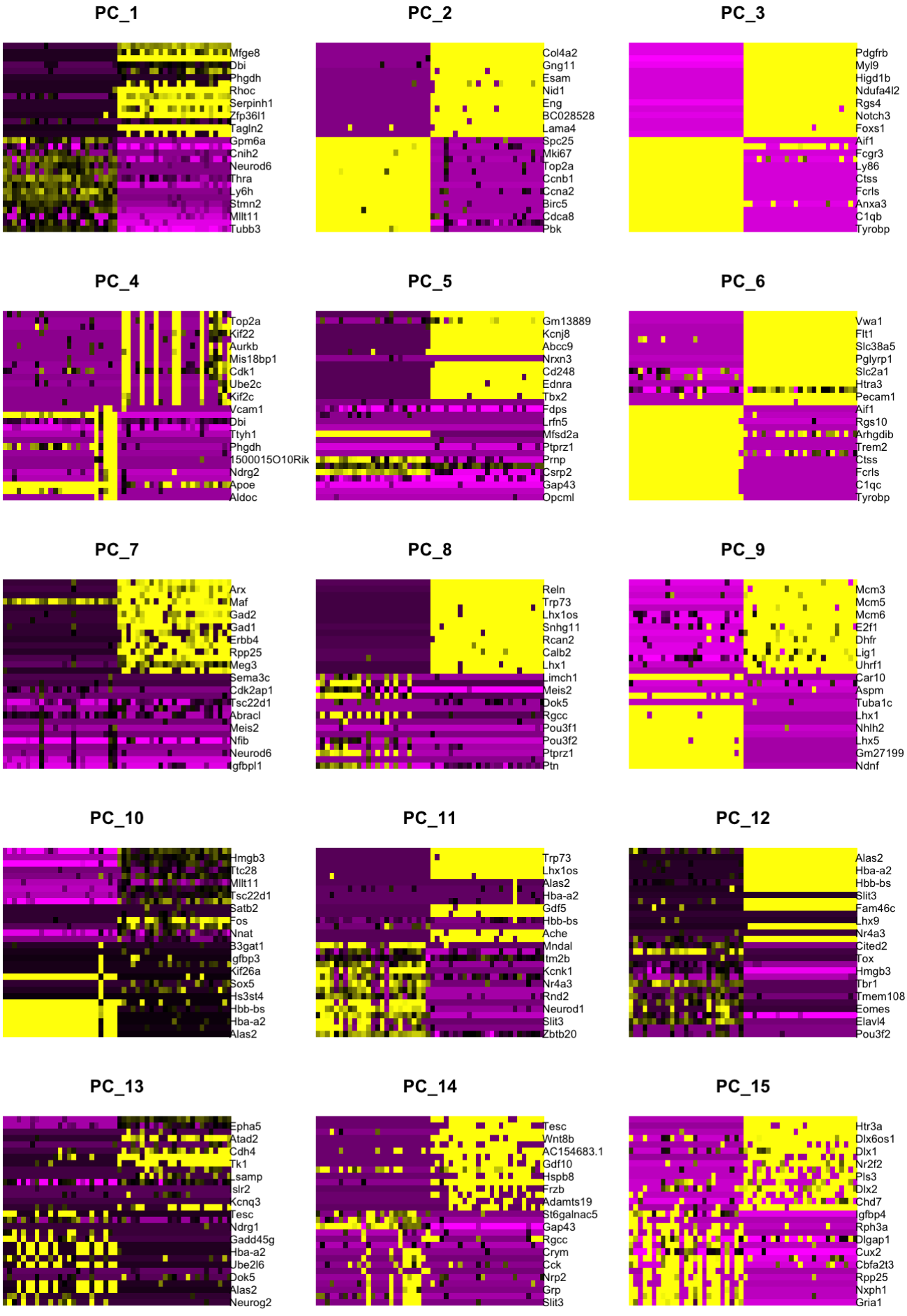
```
Neu_10k<- RunPCA(Neu_10k, features = VariableFeatures(object = Neu_10k))

DimPlot (Neu_10k, reduction = "pca")
```




```
ElbowPlot(Neu_10k)
```




```
DimHeatmap (Neu_10k, dims = 1:15, cells = 50, balanced = TRUE)
```


PC_1
Mfge8
Dbi
Phgdh
Rhoc
Serpinh1
Zfp36l1
Tagln2
Gpm6a
Cnih2
Neurod6
Thra
Ly6h
Stmn2
Mllt11
Tubb3
PC_2
Col4a2
Gng11
Esam
Nid1
Eng
BC028528
Lama4
Spc25
Mki67
Top2a
Ccnb1
Ccna2
Birc5
Cdca8
Pbk
PC_3
Pdgfrb
Myl9
Higd1b
Ndufa4l2
Rgs4
Notch3
Foxs1
Aif1
Fcgr3
Ly86
Ctss
Fcrls
Anxa3
C1qb
Tyrobp
PC_4
Top2a
Kif22
Aurkb
Mis18bp1
Cdk1
Ube2c
Kif2c
Vcam1
Dbi
Ttyh1
Phgdh
1500015O10Rik
Ndrg2
Apoe
Aldoc
PC_5
Gm13889
Kcnj8
Abcc9
Nrxn3
Cd248
Ednra
Tbx2
Fdps
Lrfn5
Mfsd2a
Ptprz1
Prnp
Csrp2
Gap43
Opcml
PC_6
Vwa1
Flt1
Slc38a5
Pglyrp1
Slc2a1
Htra3
Pecam1
Aif1
Rgs10
Arhgdib
Trem2
Ctss
Fcrls
C1qc
Tyrobp
PC_7
Arx
Maf
Gad2
Gad1
Erbb4
Rpp25
Meg3
Sema3c
Cdk2ap1
Tsc22d1
Abracl
Meis2
Nfib
Neurod6
Igfbpl1
PC_8
Reln
Trp73
Lhx1os
Snhg11
Rcan2
Calb2
Lhx1
Limch1
Meis2
Dok5
Rgcc
Pou3f1
Pou3f2
Ptprz1
Ptn
PC_9
Mcm3
Mcm5
Mcm6
E2f1
Dhfr
Lig1
Uhrf1
Car10
Aspm
Tuba1c
Lhx1
Nhlh2
Lhx5
Gm27199
Ndnf
PC_10
Hmgb3
Ttc28
Mllt11
Tsc22d1
Satb2
Fos
Nnat
B3gat1
Igfbp3
Kif26a
Sox5
Hs3st4
Hbb-bs
Hba-a2
Alas2
PC_11
Trp73
Lhx1os
Alas2
Hba-a2
Gdf5
Hbb-bs
Ache
Mndal
Itm2b
Kcnk1
Nr4a3
Rnd2
Neurod1
Slit3
Zbtb20
PC_12
Alas2
Hba-a2
Hbb-bs
Slit3
Fam46c
Lhx9
Nr4a3
Cited2
Tox
Hmgb3
Tbr1
Tmem108
Eomes
Elavl4
Pou3f2
PC_13
Epha5
Atad2
Cdh4
Tk1
Lsamp
Islr2
Kcnq3
Tesc
Ndrg1
Gadd45g
Hba-a2
Ube2l6
Dok5
Alas2
Neurog2
PC_14
Tesc
Wnt8b
AC154683.1
Gdf10
Hspb8
Frzb
Adamts19
St6galnac5
Gap43
Rgcc
Crym
Cck
Nrp2
Grp
Slit3
PC_15
Htr3a
Dlx6os1
Dlx1
Nr2f2
Pls3
Dlx2
Chd7
Igfbp4
Rph3a
Dlgap1
Cux2
Cbfa2t3
Rpp25
Nxph1
Gria1

Run UMAP on dataset

```
Neu_10k <- FindNeighbors(Neu_10k, dims = 1:10)
```

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

```
## Computing SNN
```

```
Neu_10k <- FindClusters (Neu_10k, resolution = 0.5)
```

```
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 9276
## Number of edges: 298365
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.9073
## Number of communities: 17
## Elapsed time: 0 seconds
```

```
Neu_10k <- RunUMAP(Neu_10k, dims = 1:10)
```

```
## Warning: The default method for RunUMAP has changed from calling Python UMAP via reti
culate to the R-native UWOT using the cosine metric
## To use Python UMAP via reticulate, set umap.method to 'umap-learn' and metric to 'cor
relation'
## This message will be shown once per session
```

```
## 00:31:00 UMAP embedding parameters a = 0.9922 b = 1.112
```

```
## 00:31:00 Read 9276 rows and found 10 numeric columns
```

```
## 00:31:00 Using Annoy for neighbor search, n_neighbors = 30
```

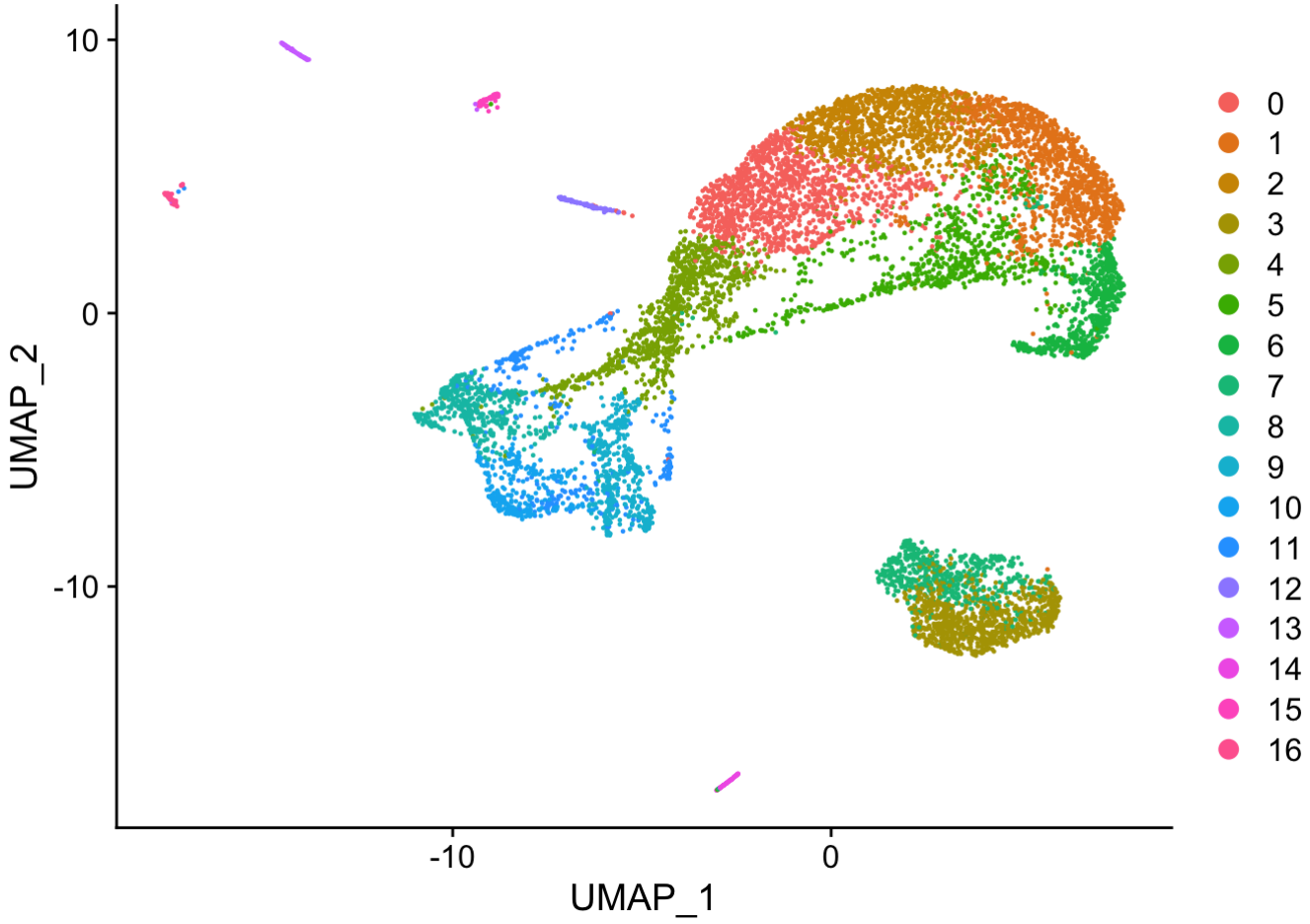
```
## 00:31:00 Building Annoy index with metric = cosine, n_trees = 50
```

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

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

```
## ************************************************|
## 00:31:01 Writing NN index file to temp file /var/folders/s9/6k4hffrx25gb0_43dcj132p15
4t8k1/T//Rtmp6NUwtY/file36f026a016b
## 00:31:01 Searching Annoy index using 1 thread, search_k = 3000
## 00:31:03 Annoy recall = 100%
## 00:31:03 Commencing smooth kNN distance calibration using 1 thread
## 00:31:04 Initializing from normalized Laplacian + noise
## 00:31:04 Commencing optimization for 500 epochs, with 374866 positive edges
## 00:31:15 Optimization finished
```

```
DimPlot (Neu_10k, reduction = "umap")
```



Find all markers of each cluster to identify them

```
cluster0.markers <- FindMarkers (Neu_10k, ident.1 = 0, min.pct = 0.5)
cluster1.markers <- FindMarkers (Neu_10k, ident.1 = 1, min.pct = 0.5)
cluster2.markers <- FindMarkers (Neu_10k, ident.1 = 2, min.pct = 0.5)
cluster3.markers <- FindMarkers (Neu_10k, ident.1 = 3, min.pct = 0.5)
cluster4.markers <- FindMarkers (Neu_10k, ident.1 = 4, min.pct = 0.5)
cluster5.markers <- FindMarkers (Neu_10k, ident.1 = 5, min.pct = 0.5)
cluster6.markers <- FindMarkers (Neu_10k, ident.1 = 6, min.pct = 0.5)
```

To view the genes lists on your RStudio console

```
View(cluster0.markers)
View(cluster1.markers)
View(cluster2.markers)
View(cluster3.markers)
View(cluster4.markers)
View(cluster5.markers)
View(cluster6.markers)
```

To “export” the genes lists to your current directory

```
write.csv(cluster0.markers, “cluster0_markers_UMAP”)
write.csv(cluster1.markers, “cluster1_markers_UMAP”)
write.csv(cluster2.markers, “cluster2_markers_UMAP”)
write.csv(cluster3.markers, “cluster3_markers_UMAP”)
write.csv(cluster4.markers, “cluster4_markers_UMAP”)
write.csv(cluster5.markers, “cluster5_markers_UMAP”)
write.csv(cluster6.markers, “cluster6_markers_UMAP”)
```