workshop_Rscripts.R

yasinkaymaz

Wed Aug 8 21:40:44 2018

```
###### - Input data and analysis tool requirements (setup)

library(Matrix)
library(Seurat)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: cowplot
```

```
##
## Attaching package: 'cowplot'
```

```
## The following object is masked from 'package:ggplot2':
##
##     ggsave
```

```
library(dplyr)
```

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

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

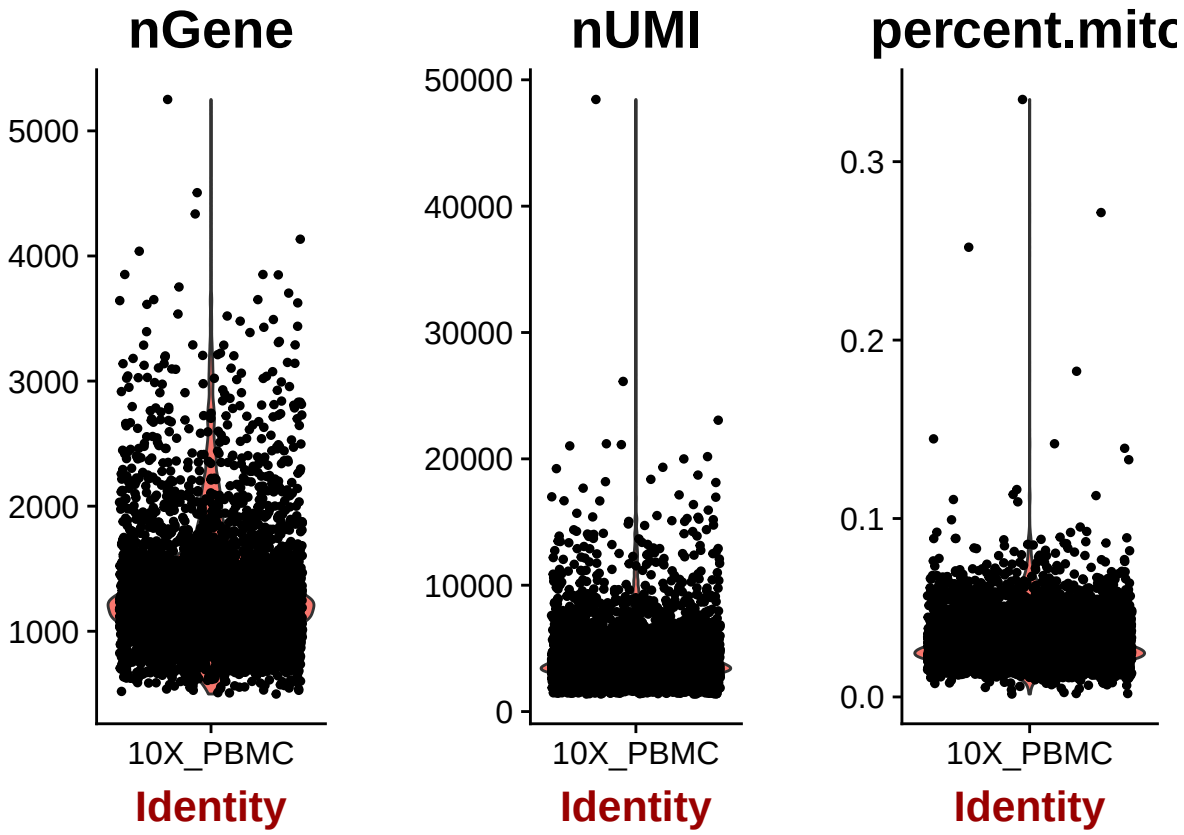
```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(SIMLR)


pbmc.data <- Read10X(
  data.dir = "~/Downloads/scRNAseqWS/10Xdata/filtered_gene_bc_matrices/GRCh38/")

#min.cells = 3 : keep all genes expressed in >= 3 cells.
#min.genes = 200 : Keep all cells with at least 200 detected genes.
pbmc <- CreateSeuratObject(raw.data = pbmc.data,
                           min.cells = 3,
                           min.genes = 200,
                           project = "10X_PBMC")

#head(pbmc@raw.data)
#head(as.matrix(pbmc@raw.data))
#head(as.matrix(pbmc@raw.data)[1:10,1:10])

#head(pbmc@meta.data)


mito.genes <- grep(pattern = "^MT-",
                   x = rownames(x = pbmc@data),
```

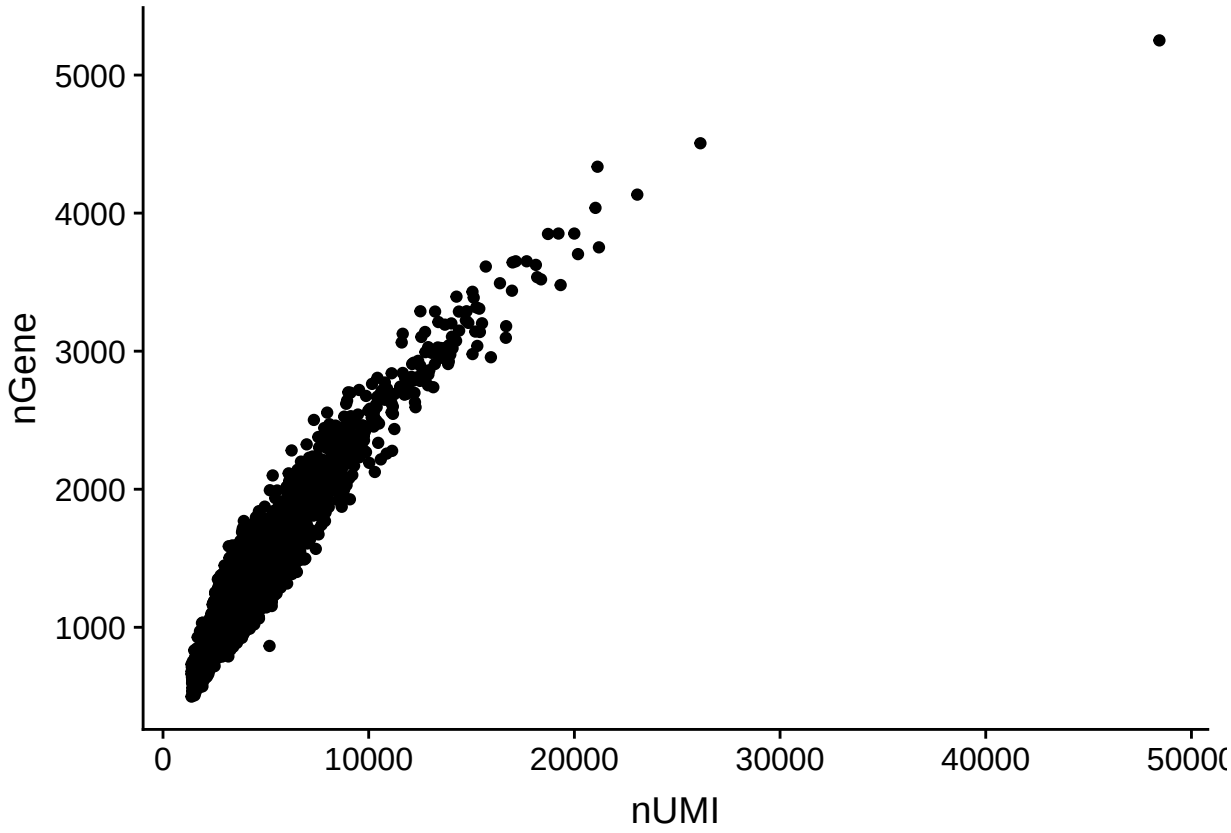
```
                    value = TRUE)

percent.mito <- Matrix::colSums(pbmc@raw.data[mito.genes, ])/Matrix::colSums(pbmc@raw.data)

pbmc <- AddMetaData(object = pbmc,
                    metadata = percent.mito,
                    col.name = "percent.mito")

head(pbmc@meta.data)
```

```
##                  nGene nUMI orig.ident percent.mito
## AAACCTGAGAAGGCCT   748 1738   10X_PBMC   0.06386651
## AAACCTGAGACAGACC  1052 3240   10X_PBMC   0.05462963
## AAACCTGAGATAGTCA   739 1683   10X_PBMC   0.07367796
## AAACCTGAGCGCCTCA   875 2319   10X_PBMC   0.03839517
## AAACCTGAGGCATGGT   951 2983   10X_PBMC   0.02246061
## AAACCTGCAAGGTTCT  1248 4181   10X_PBMC   0.02224348
```

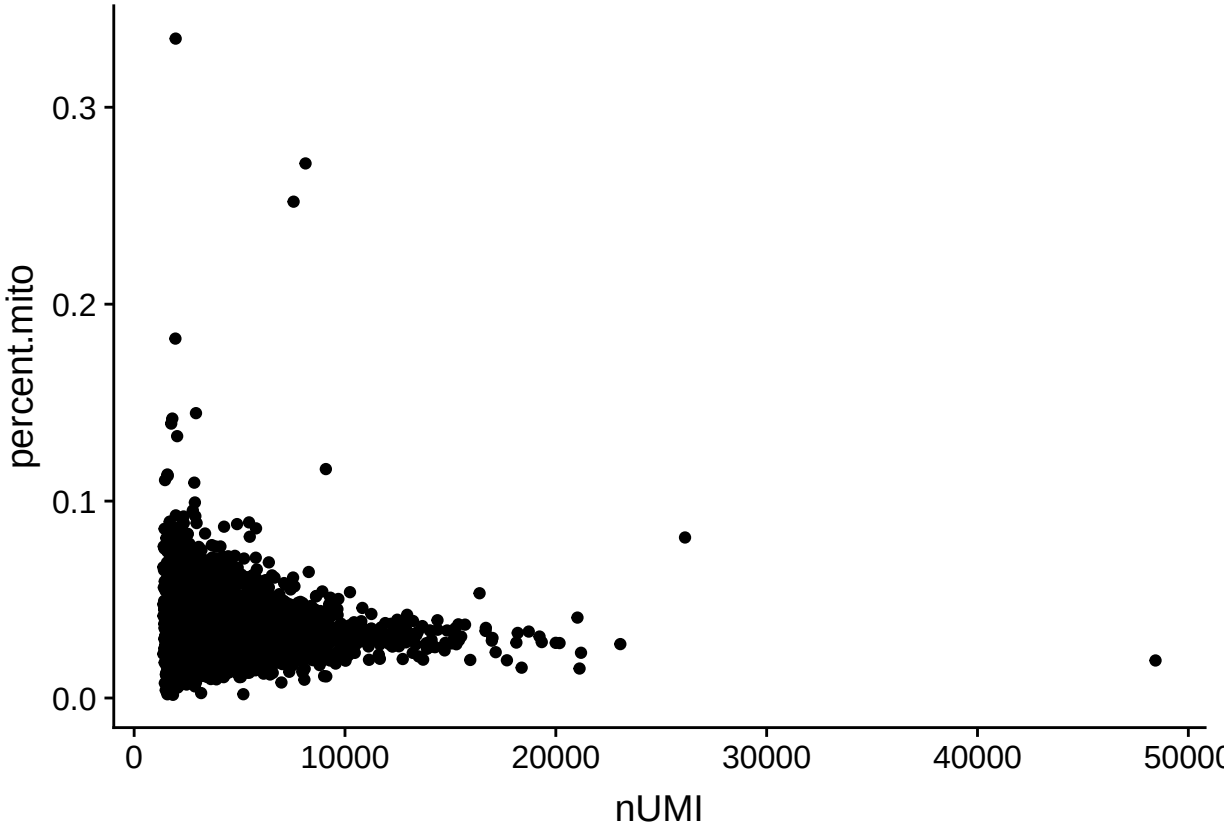
```
VlnPlot(pbmc, features.plot = c("nGene", "nUMI", "percent.mito"))
```



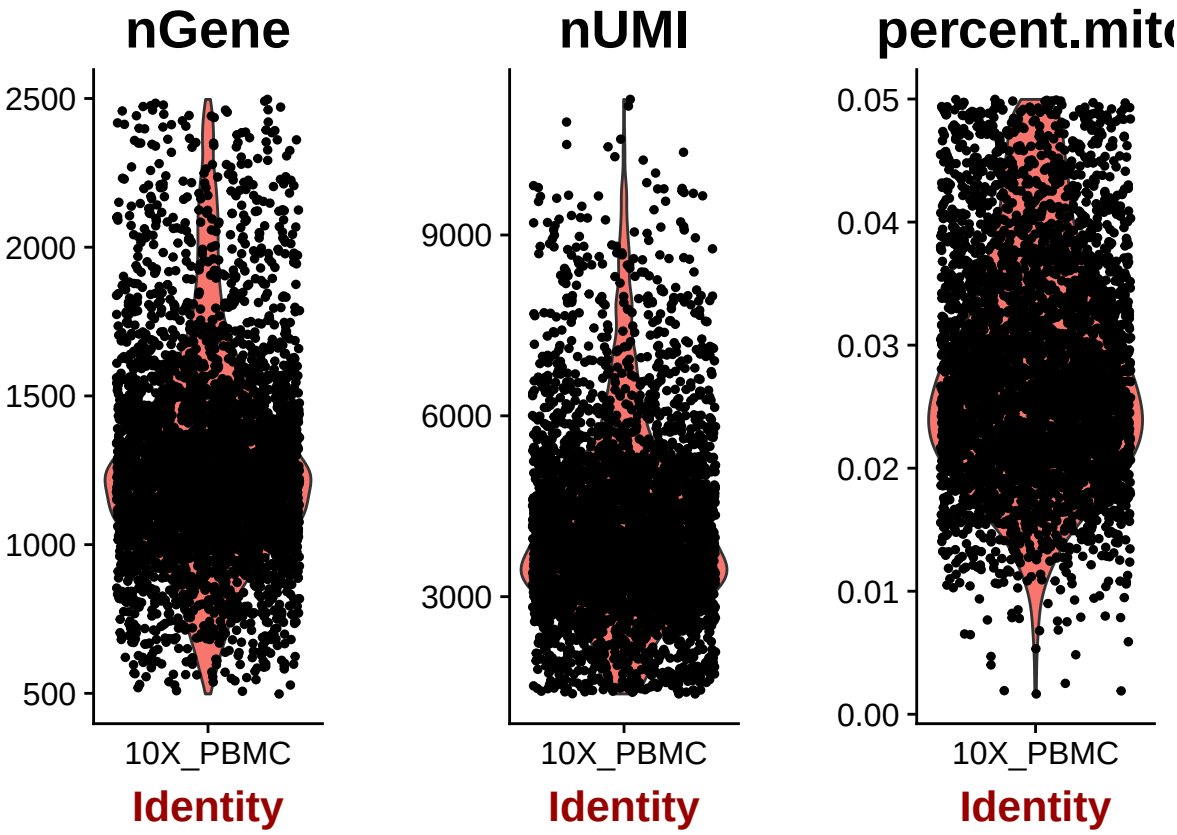

```
ggplot(pbmc@meta.data, aes(nUMI, nGene)) + geom_point()
```




```
ggplot(pbmc@meta.data, aes(nUMI, percent.mito)) + geom_point()
```

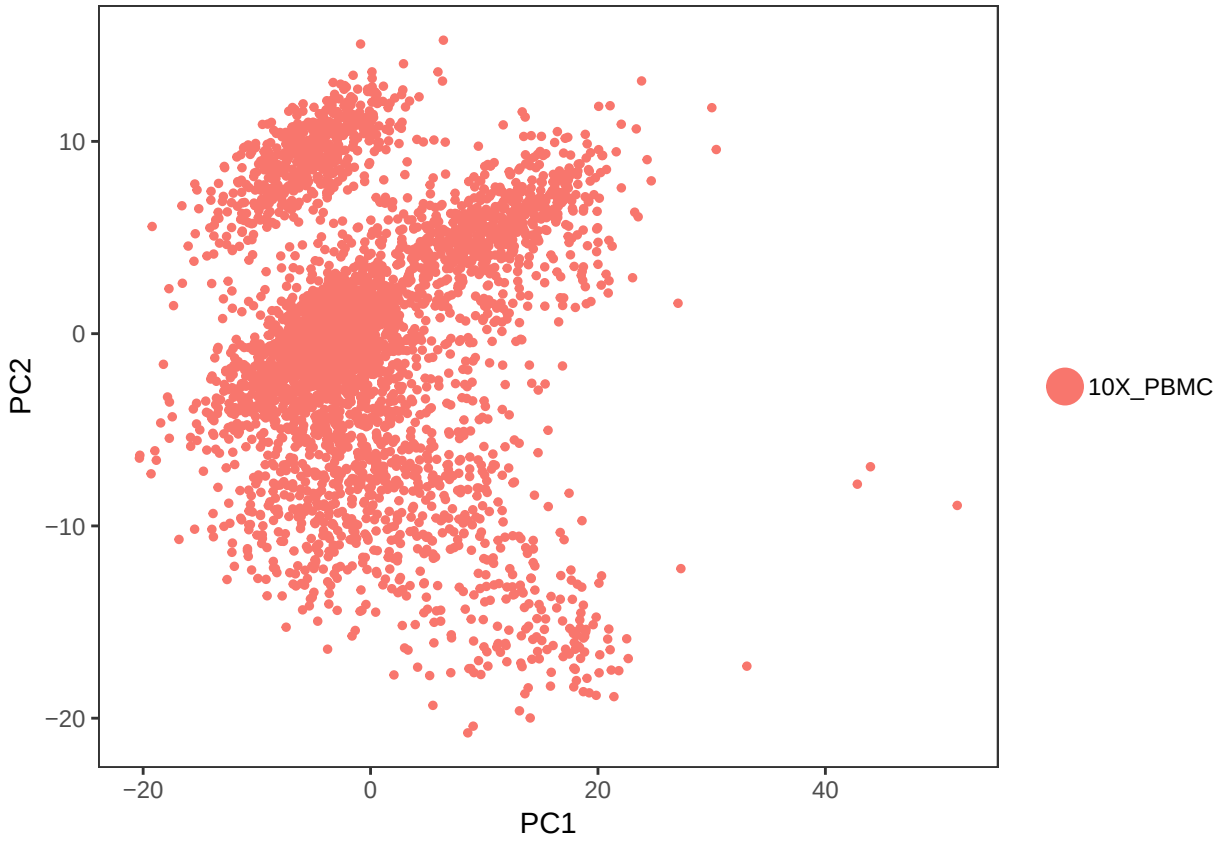


```
pbmc <- FilterCells(pbmc,
                    subset.names = c("nGene", "percent.mito"),
                    low.thresholds = c(200, -Inf),
                    high.thresholds = c(2500, 0.05))

#Check to see if filtration works as we expected.
VlnPlot(pbmc, features.plot = c("nGene", "nUMI", "percent.mito"))
```

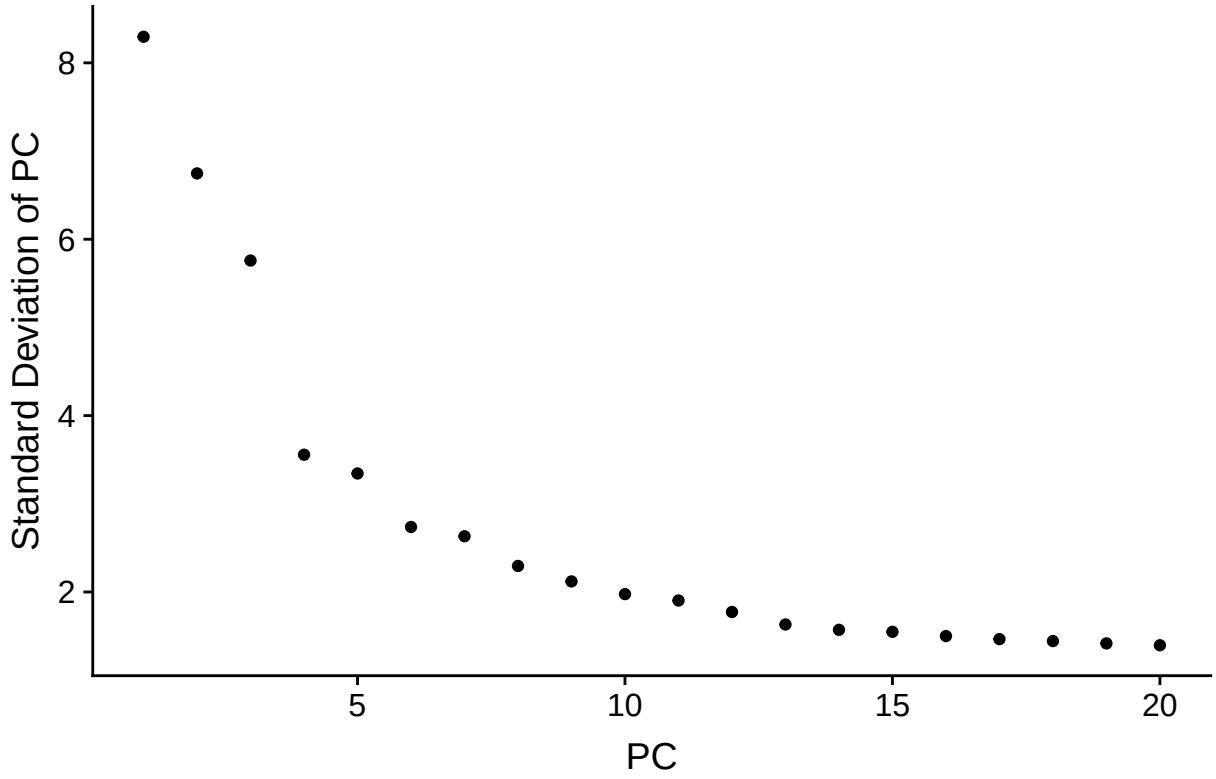



```
pbmc <- NormalizeData(pbmc,
                      normalization.method = "LogNormalize",
                      scale.factor = 10000)

pbmc <- FindVariableGenes(pbmc,
                          mean.function = ExpMean,
                          dispersion.function = LogVMR,
                          do.plot = FALSE,
                          display.progress = FALSE)

hv.genes <- head(rownames(pbmc@hvg.info), 1000)

pbmc <- ScaleData(pbmc,
                  genes.use = hv.genes,
                  vars.to.regress = c("nUMI", "percent.mito"),
                  display.progress = FALSE)

pbmc <- RunPCA(pbmc,
               pc.genes = hv.genes,
```

```
                do.print = FALSE)

PCAPlot(pbmc, dim.1 = 1, dim.2 = 2)
```



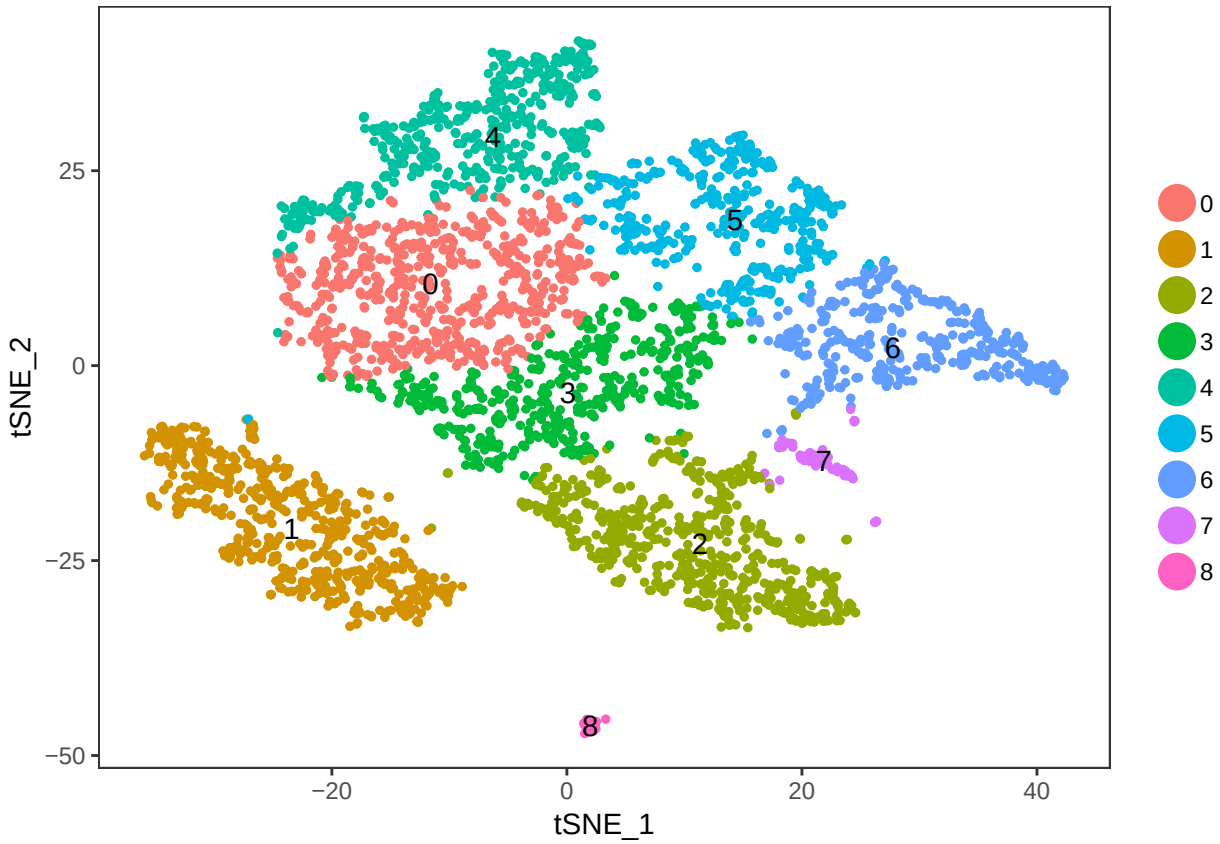

```
PCElbowPlot(pbmc)
```



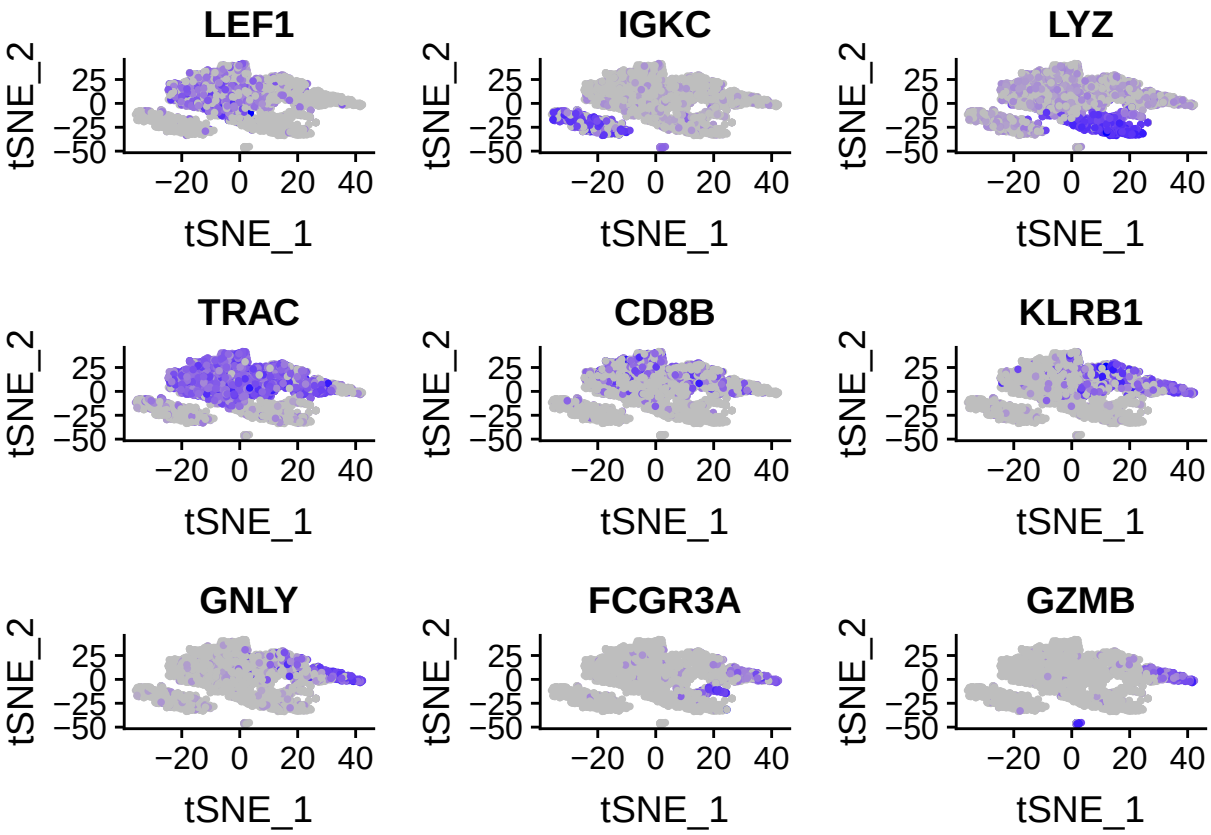

```
pbmc <- FindClusters(pbmc,
                     reduction.type = "pca",
                     dims.use = 1:5,
                     resolution = 0.6,
                     print.output = TRUE,
                     save.SNN = TRUE)

#Check cluster id assignments to cells
head(pbmc@ident,20)
```

```
## AAACCTGAGCGCCTCA AAACCTGAGGCATGGT AAACCTGCAAGGTTCT AAACCTGCATCCCATC
##                5                0                0                1
## AAACCTGCATGAAGTA AAACCTGGTACATCCA AAACCTGGTGCGGTAA AAACCTGTCGTGGTCG
##                2                1                0                0
## AAACCTGTCTCTGCTG AAACGGGAGGCTAGCA AAACGGGAGTGTCCAT AAACGGGGTCTTCTCG
##                4                0                4                2
## AAACGGGGTGGACGAT AAACGGGGTTTGCATG AAACGGGTCTGGGCCA AAACGGGTCTGGTATG
##                6                2                4                5
## AAAGATGAGACATAAC AAAGATGGTCCCTACT AAAGATGTCCGAATGT AAAGATGTCTGGTTCC
##                3                1                4                2
## Levels: 0 1 2 3 4 5 6 7 8
```

```
pbmc <- RunTSNE(pbmc,
                dims.use = 1:5,
                do.fast = TRUE)

TSNEPlot(pbmc, do.label = TRUE)
```




```
# !!! pbmc.markers !!!
pbmc.markers <- FindAllMarkers(pbmc,
                               only.pos = TRUE,
                               min.pct = 0.25,
                               logfc.threshold = 0.25)

top5.markers <- pbmc.markers %>% group_by(cluster) %>% top_n(5, avg_logFC)

best.markers <- pbmc.markers %>% group_by(cluster) %>% top_n(1, avg_logFC)

FeaturePlot(pbmc,
            features.plot = best.markers$gene,
            cols.use = c("grey", "blue"),
            reduction.use = "tsne")
```

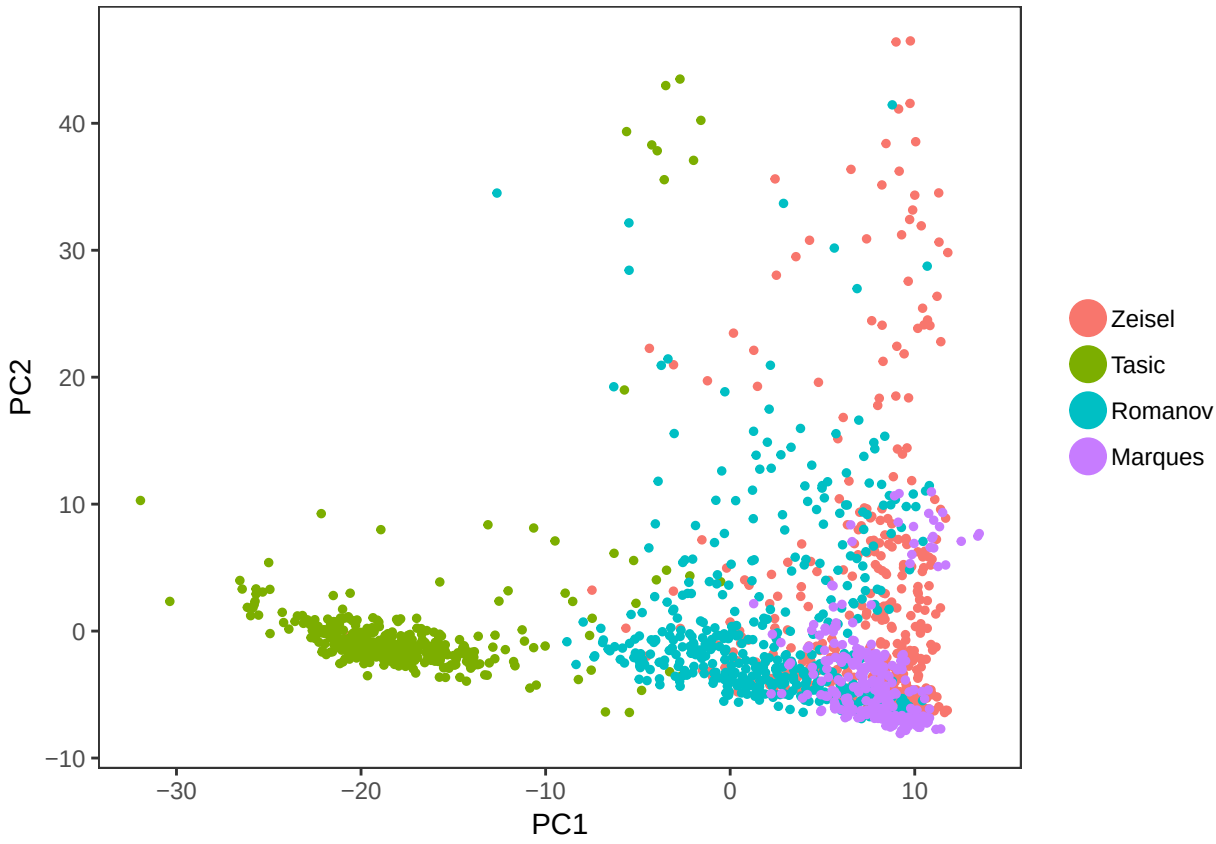



```
########## Second Part #########

ob.list <- list("zeisel", "romanov", "tasic", "marques")

#Load the expression and meta data of each study.

for (i in 1:length(ob.list)){
  obj.data <- paste(ob.list[[i]],".data",sep="");
  #Read the expression matrix from a text file for each dataset.
  assign(obj.data, read.delim(paste("~/Downloads/scRNAseqWS/",ob.list[[i]],".expression.txt",sep=""),
                              header = TRUE, row.names = 1))
}

#Since the expression matrices of these datasets are in TPM (already normalized), we are going to skip
zeisel.data <- log1p(zeisel.data)
romanov.data <- log1p(romanov.data)
tasic.data <- log1p(tasic.data)
marques.data <- log1p(marques.data)

for (i in 1:length(ob.list)){
  obj.meta <- paste(ob.list[[i]],".meta",sep="");
  #Reading the Run information meta data from a text file for each dataset.
  assign(obj.meta, read.delim(paste("~/Downloads/scRNAseqWS/",ob.list[[i]],".RunTable.txt",sep=""),
                              header = TRUE))
}

rownames(zeisel.meta) <- zeisel.meta$Run_s
```

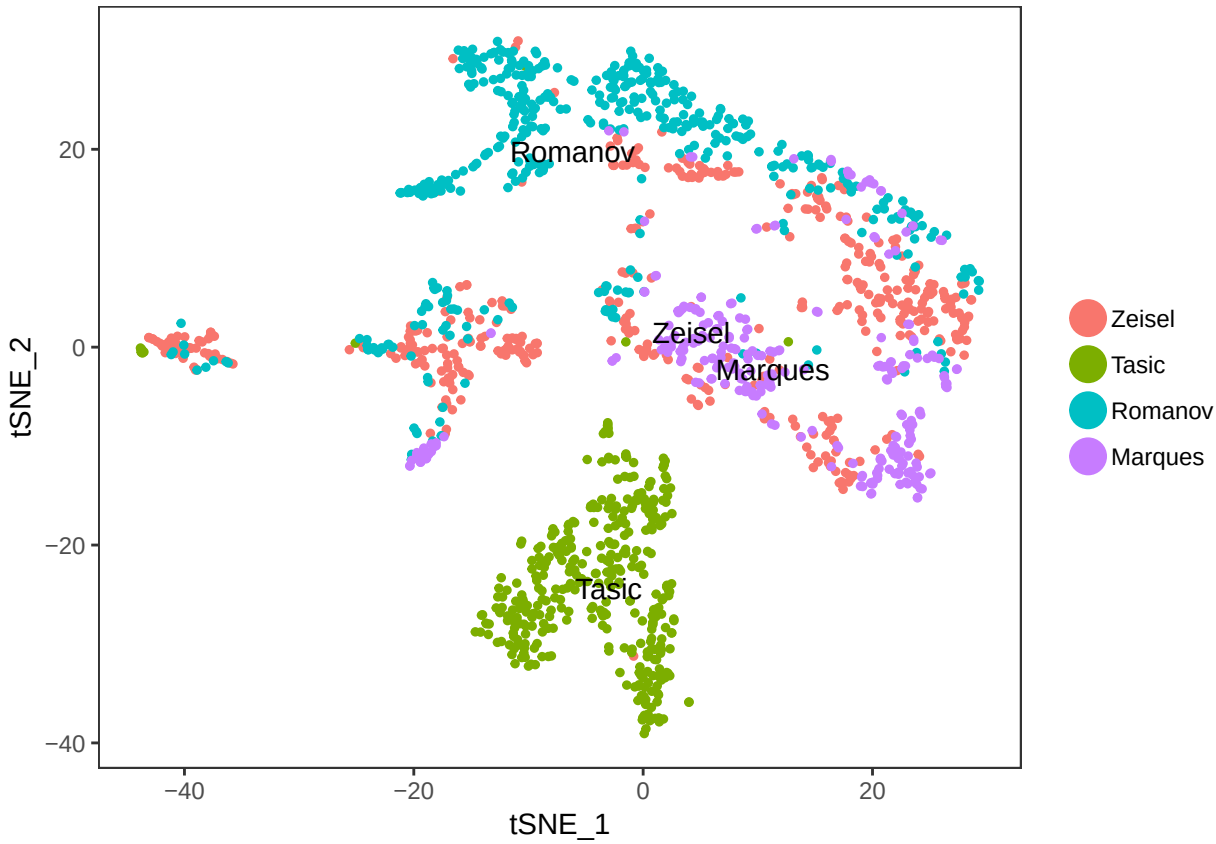
```
rownames(romanov.meta) <- romanov.meta$Run_s
rownames(tasic.meta) <- tasic.meta$Run_s
rownames(marques.meta) <- marques.meta$Run_s

batches <- rbind(zeisel.meta[,c("Run_s","Owner","SRA_Study_s")],
                 tasic.meta[,c("Run_s","Owner","SRA_Study_s")],
                 romanov.meta[,c("Run_s","Owner","SRA_Study_s")],
                 marques.meta[,c("Run_s","Owner","SRA_Study_s")])

combined.data <- cbind(zeisel.data, tasic.data, romanov.data, marques.data)

combined.data <- as(as.matrix(combined.data), "dgCMatrix")

fourDataset <- CreateSeuratObject(raw.data = combined.data, project = "4dataset.Pre")

fourDataset <- AddMetaData(fourDataset, metadata = batches)

fourDataset <- FilterCells(fourDataset, subset.names = "nGene", low.thresholds = 2500)

fourDataset <- FindVariableGenes(fourDataset, do.plot = F, display.progress = F)

fourDataset <- ScaleData(fourDataset, display.progress = F)
```

```
## NormalizeData has not been run, therefore ScaleData is running on non-normalized values. Recommended
## ScaleData is running on non-normalized values. Recommended workflow is to run NormalizeData first.
```

```
fourDataset <- RunPCA(fourDataset, pc.genes = fourDataset@var.genes, pcs.compute = 10, do.print = FALSE)

PCAPlot(fourDataset, pt.size=1, group.by ="Owner", dim.1 = 1, dim.2 = 2)
```




```
fourDataset <- RunTSNE(fourDataset, reduction.use = "pca", dims.use = 1:5)

TSNEPlot(fourDataset, do.label = T, group.by ="Owner")
```




```
#Subset the data
zeisel <- SubsetData(fourDataset, cells.use=names(zeisel.data),do.center=T, do.scale=T)
```

```
## NormalizeData has not been run, therefore ScaleData is running on non-normalized values. Recommended
## ScaleData is running on non-normalized values. Recommended workflow is to run NormalizeData first.
```

```
## Scaling data matrix
```

```
tasic <- SubsetData(fourDataset, cells.use=names(tasic.data),do.center=T, do.scale=T)
```

```
## NormalizeData has not been run, therefore ScaleData is running on non-normalized values. Recommended
## ScaleData is running on non-normalized values. Recommended workflow is to run NormalizeData first.
```

```
## Scaling data matrix
```

```
romanov <- SubsetData(fourDataset, cells.use=names(romanov.data),do.center=T, do.scale=T)
```

```
## NormalizeData has not been run, therefore ScaleData is running on non-normalized values. Recommended
## ScaleData is running on non-normalized values. Recommended workflow is to run NormalizeData first.
```

```
## Scaling data matrix
```

```
marques <- SubsetData(fourDataset, cells.use=names(marques.data),do.center=T, do.scale=T)
```

```
## NormalizeData has not been run, therefore ScaleData is running on non-normalized values. Recommended
## ScaleData is running on non-normalized values. Recommended workflow is to run NormalizeData first.
```

```
## Scaling data matrix
```

```
ob.list <- list(zeisel, romanov, tasic, marques)

genes.use <- c()
```

```
for (i in 1:length(ob.list)) {
  genes.use <- c(genes.use, head(rownames(ob.list[[i]]@hvg.info), 1000))
}

genes.use <- names(which(table(genes.use) > 1))

for (i in 1:length(ob.list)) {
  genes.use <- genes.use[genes.use %in% rownames(ob.list[[i]]@scale.data)]
}

mouseBrain.integrated <- RunMultiCCA(ob.list,
                                     genes.use = genes.use,
                                     num.ccs = 5)
```

```
## [1] "Computing CC 1"
## [1] "Computing CC 2"
## [1] "Computing CC 3"
## [1] "Computing CC 4"
## [1] "Computing CC 5"
```

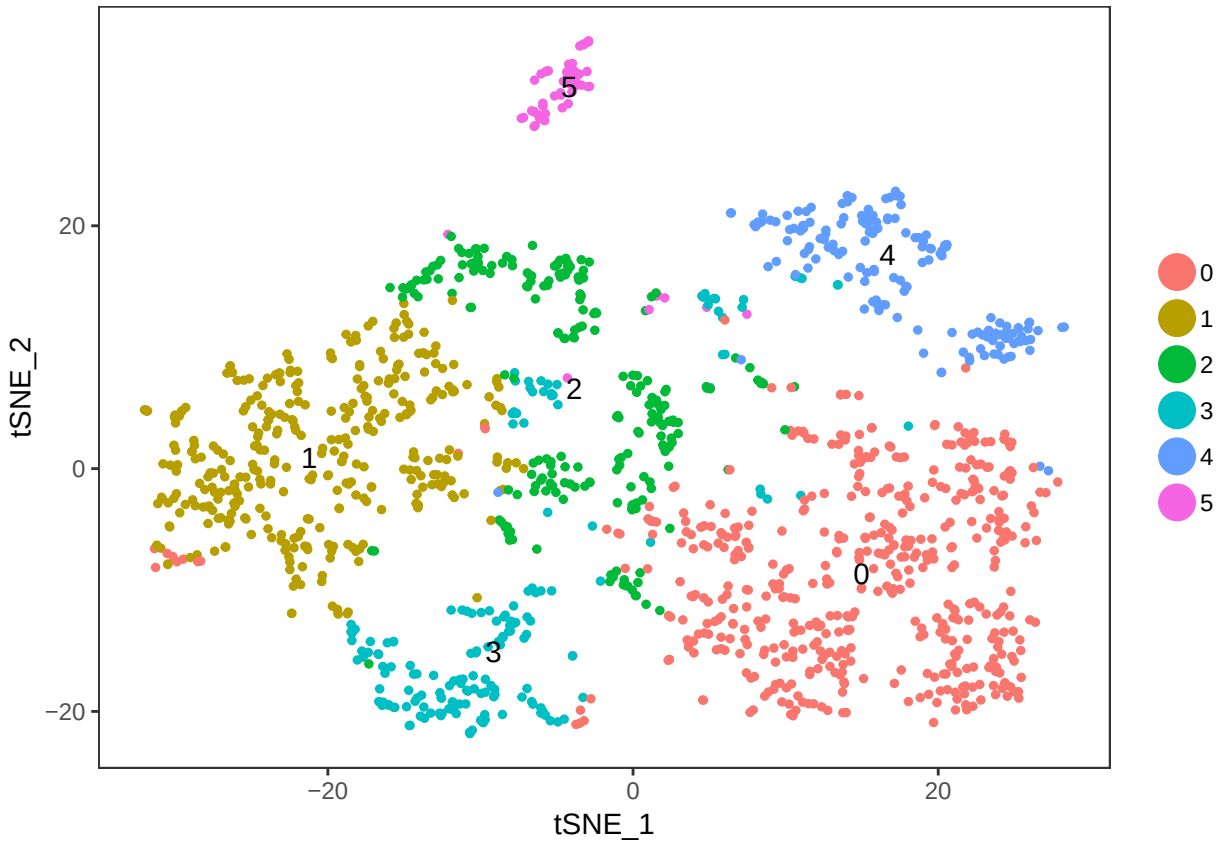
```
## Scaling data matrix
```

```
mouseBrain.integrated <- CalcVarExpRatio(mouseBrain.integrated,
                                         reduction.type = "pca",
                                         grouping.var = "Owner",
                                         dims.use = 1:5)

mouseBrain.integrated <- SubsetData(mouseBrain.integrated,
                                    subset.name = "var.ratio.pca",
                                    accept.low = 0.5)

mouseBrain.integrated <- AlignSubspace(mouseBrain.integrated,
                                       reduction.type = "cca",
                                       grouping.var = "Owner",
                                       dims.align = 1:5)
```

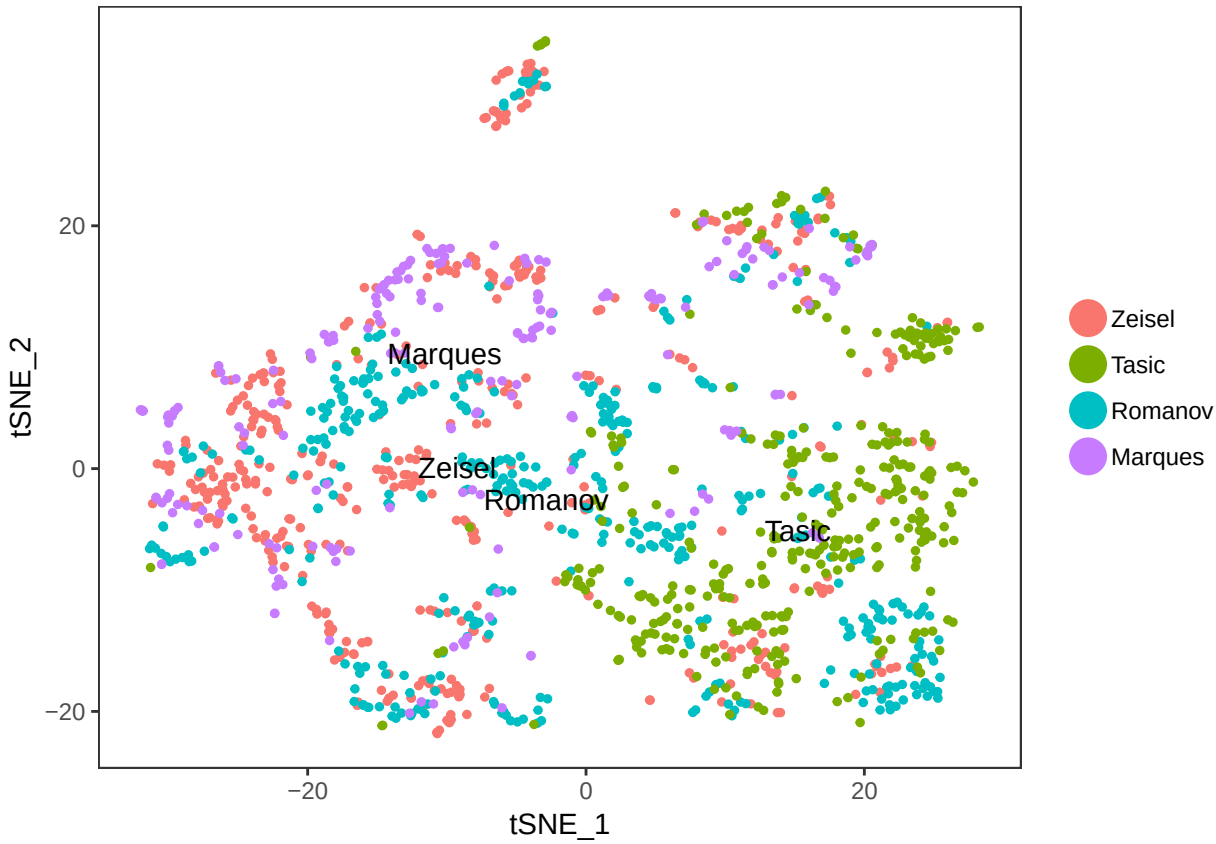
```
## Scaling data matrix
```

```
## Scaling data matrix
## Scaling data matrix
## Scaling data matrix
```

```
mouseBrain.integrated <- FindClusters(mouseBrain.integrated,
                                      reduction.type = "cca.aligned",
                                      dims.use = 1:5,
                                      save.SNN = T,
                                      resolution = 0.4)

mouseBrain.integrated <- RunTSNE(mouseBrain.integrated,
                                 reduction.use = "cca.aligned",
                                 dims.use = 1:5,
                                 check_duplicates = FALSE)

# Visualization
TSNEPlot(mouseBrain.integrated, do.label = T)
```




```
TSNEPlot(mouseBrain.integrated, do.label = T,group.by ="Owner")
```

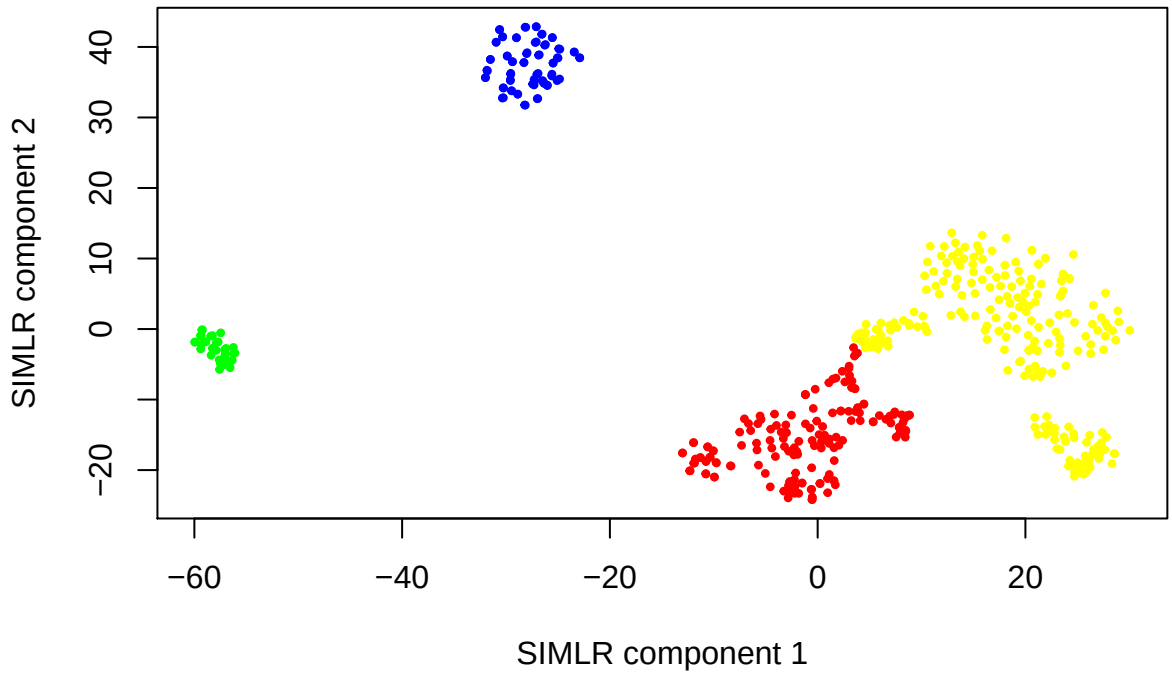



```
#Alternative way of determining cell subset clusters with SIMLR
set.seed(11111)
# Determine optimal number of clusters as described in the Nat. Methods paper
# picka cluster range and reports two metrics; the lower the value the more
# support for that number of clusters; in my limited experience these methods
# are concordant.

zclust<-SIMLR_Estimate_Number_of_Clusters(zeisel.data, NUMC=2:5)

#run SIMLR
zsimlr<-SIMLR(zeisel.data, 4)
```

```
## Computing the multiple Kernels.
## Performing network diffiusion.
## Iteration:  1
## Iteration:  2
## Iteration:  3
## Iteration:  4
## Iteration:  5
## Iteration:  6
## Iteration:  7
## Iteration:  8
## Iteration:  9
## Iteration:  10
## Iteration:  11
## Iteration:  12
## Iteration:  13
```

```
## Performing t-SNE.
## Epoch: Iteration # 100  error is:  0.1901695
## Epoch: Iteration # 200  error is:  0.1655055
## Epoch: Iteration # 300  error is:  0.1599297
## Epoch: Iteration # 400  error is:  0.1546565
## Epoch: Iteration # 500  error is:  0.1535049
## Epoch: Iteration # 600  error is:  0.1529816
## Epoch: Iteration # 700  error is:  0.1499769
## Epoch: Iteration # 800  error is:  0.1501158
## Epoch: Iteration # 900  error is:  0.1503416
## Epoch: Iteration # 1000  error is:  0.1478741
## Performing Kmeans.
## Performing t-SNE.
## Epoch: Iteration # 100  error is:  12.04924
## Epoch: Iteration # 200  error is:  0.2614296
## Epoch: Iteration # 300  error is:  0.2471728
## Epoch: Iteration # 400  error is:  0.2426525
## Epoch: Iteration # 500  error is:  0.2393708
## Epoch: Iteration # 600  error is:  0.2380072
## Epoch: Iteration # 700  error is:  0.2370422
## Epoch: Iteration # 800  error is:  0.2362994
## Epoch: Iteration # 900  error is:  0.235715
## Epoch: Iteration # 1000  error is:  0.2352412
```

```r
# Create plotting function, color-coding points by cluster membership
plotSIMLRclusters <- function(obj) {
  col <- ifelse(obj$y$cluster==1, 'red',
                ifelse(obj$y$cluster==2, 'blue',
                       ifelse(obj$y$cluster==3, 'green','yellow')))
  plot(obj$ydata,
       col=col,
       xlab = "SIMLR component 1",
       ylab = "SIMLR component 2",
       pch=20,
       cex=0.7)
}

# Call plotting function
plotSIMLRclusters(zsimlr)
```




```
sessionInfo()
```

```
## R version 3.5.0 (2018-04-23)
## Platform: x86_64-apple-darwin17.5.0 (64-bit)
## Running under: macOS High Sierra 10.13.6
##
## Matrix products: default
## BLAS: /System/Library/Frameworks/Accelerate.framework/Versions/A/Frameworks/vecLib.framework/Versions
## LAPACK: /System/Library/Frameworks/Accelerate.framework/Versions/A/Frameworks/vecLib.framework/Versi
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] bindrcpp_0.2.2 SIMLR_1.6.0    dplyr_0.7.6    Seurat_2.3.3
## [5] cowplot_0.9.3  ggplot2_3.0.0  Matrix_1.2-14
##
## loaded via a namespace (and not attached):
##   [1] diffusionMap_1.1-0   Rtsne_0.13           colorspace_1.3-2
##   [4] class_7.3-14         modeltools_0.2-22    ggridges_0.5.0
##   [7] mclust_5.4.1         rprojroot_1.3-2      htmlTable_1.12
##  [10] base64enc_0.1-3      rstudioapi_0.7       proxy_0.4-22
##  [13] flexmix_2.3-14       bit64_0.9-7          RSpectra_0.13-1
##  [16] mvtnorm_1.0-8        codetools_0.2-15     splines_3.5.0
##  [19] R.methodsS3_1.7.1    robustbase_0.93-1    knitr_1.20
##  [22] Formula_1.2-3        jsonlite_1.5         ica_1.0-2
##  [25] cluster_2.0.7-1      kernlab_0.9-26       png_0.1-7
##  [28] R.oo_1.22.0          compiler_3.5.0       backports_1.1.2
##  [31] assertthat_0.2.0     lazyeval_0.2.1       lars_1.2
##  [34] acepack_1.4.1        htmltools_0.3.6      tools_3.5.0
```

```
##  [37] igraph_1.2.1         gtable_0.2.0         glue_1.2.0
##  [40] RANN_2.6             reshape2_1.4.3       Rcpp_0.12.17
##  [43] trimcluster_0.1-2    gdata_2.18.0         ape_5.1
##  [46] nlme_3.1-137         iterators_1.0.10     fpc_2.1-11
##  [49] lmtest_0.9-36        stringr_1.3.1        irlba_2.3.2
##  [52] gtools_3.8.1         DEoptimR_1.0-8       MASS_7.3-50
##  [55] zoo_1.8-3            scales_0.5.0         doSNOW_1.0.16
##  [58] parallel_3.5.0       RColorBrewer_1.1-2   reticulate_1.9
##  [61] pbapply_1.3-4        gridExtra_2.3        rpart_4.1-13
##  [64] segmented_0.5-3.0    latticeExtra_0.6-28  stringi_1.2.3
##  [67] foreach_1.4.4        checkmate_1.8.5      caTools_1.17.1
##  [70] SDMTools_1.1-221     rlang_0.2.1          pkgconfig_2.0.1
##  [73] dtw_1.20-1           prabclus_2.2-6       bitops_1.0-6
##  [76] pracma_2.1.4         evaluate_0.10.1      lattice_0.20-35
##  [79] ROCR_1.0-7           purrr_0.2.5          bindr_0.1.1
##  [82] labeling_0.3         htmlwidgets_1.2      bit_1.1-14
##  [85] tidyselect_0.2.4     RcppAnnoy_0.0.10     plyr_1.8.4
##  [88] magrittr_1.5         R6_2.2.2             snow_0.4-2
##  [91] gplots_3.0.1         Hmisc_4.1-1          pillar_1.3.0
##  [94] foreign_0.8-70       withr_2.1.2          fitdistrplus_1.0-9
##  [97] mixtools_1.1.0       survival_2.42-6      scatterplot3d_0.3-41
## [100] nnet_7.3-12          tibble_1.4.2         tsne_0.1-3
## [103] crayon_1.3.4         hdf5r_1.0.0          KernSmooth_2.23-15
## [106] rmarkdown_1.10       grid_3.5.0           data.table_1.11.4
## [109] metap_0.9            digest_0.6.15        diptest_0.75-7
## [112] tidyr_0.8.1          R.utils_2.6.0        stats4_3.5.0
## [115] munsell_0.5.0
```