Single Cell ATAC-seq of Mouse Brain Cells

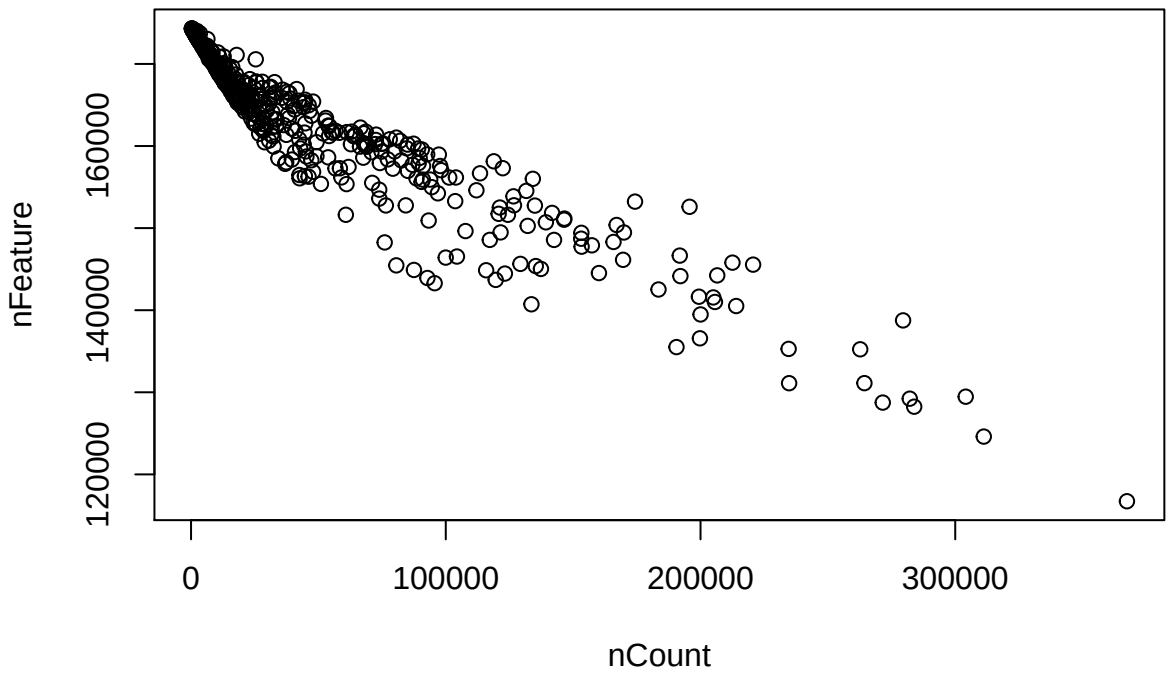
Christian Chua

Mon Mar 08 13:17:57 2021

scATAC-seq Analysis

Data Exploration

```
# read in the data
mmus_brain_data <- read.csv(
  file = "atac.mm10.counts.csv",
  header = TRUE,
  row.names = 1
)

# number of peak calls per sample
nCount <- as.data.frame(colSums(mmus_brain_data))
colnames(nCount) <- "nCount"

# number of genes per sample
nFeature <- NULL
for (cell in colnames(mmus_brain_data)){
  nFeature <- rbind(nFeature, nrow(mmus_brain_data) - sum(mmus_brain_data[[cell]] > 0))
}
nFeature <- as.data.frame(nFeature)
colnames(nFeature) <- "nFeature"
rownames(nFeature) <- colnames(mmus_brain_data)

# view features versus counts
options(scipen=5)
plot(nCount$nCount, nFeature$nFeature, xlab="nCount", ylab="nFeature")
```




```
# view count and feature distributions
# library(vioplot)
# vioplot(nCount$nCount)
# vioplot(nFeature$nFeature)
```

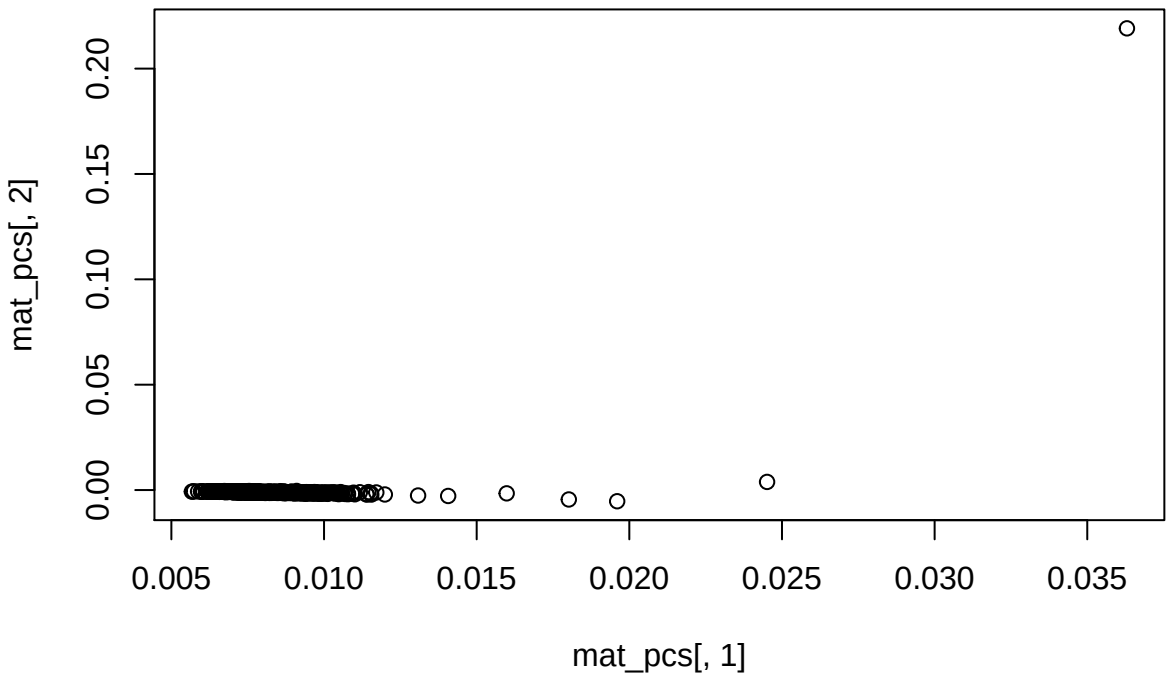
Data Normalization and Log-Transformation

```
library(Matrix)
library(readr)
library(dplyr)

# convert to dataframe
mat <- as.data.frame(mmus_brain_data)

# binarize the data
mat[mat >0] <- 1

# log transformation data
transform.idf <- function(data) {
  data <- as.matrix(data)

  # normalize the data
  npeaks <- Matrix::colSums(data)
  tf <- t(t(data) / npeaks)
```

```
  # log transformation
  idf <- log(1+ ncol(data) / Matrix::rowSums(data))
  normed.data <- Diagonal(length(idf), idf) %*% tf
  normed.data[which(is.na(normed.data))] <- 0
  return(normed.data)
}

mat <- transform.idf(mat)
# range(mat)
# dim(mat)
```

Dimensional Reduction using Latent Semantic Indexing

```
library(irlba)
set.seed(8)

# latent semantic indexing (LSI)
mat.lsi<- irlba(mat, 50)

# singular value decomposition
d_diag <- matrix(0, 50, 50)
diag(d_diag) <- mat.lsi$d # rank (number of genes) approximation of peak matrix
mat_pcs <- t(d_diag %*% t(mat.lsi$v)) # projections in latent semantic space
rownames(mat_pcs) <- colnames(mat)

# principal components
# dim(mat_pcs)
plot(mat_pcs[,1], mat_pcs[,2])
```



Clustering

```
library(RANN)

# KNN
knn.info<- RANN::nn2(mat_pcs, k = 30)

# convert to adjacency matrix
knn <- knn.info$nn.idx

adj <- matrix(0, nrow(mat_pcs), nrow(mat_pcs))
rownames(adj) <- colnames(adj) <- rownames(mat_pcs)

for(i in seq_len(nrow(mat_pcs))) {
    adj[i,rownames(mat_pcs)[knn[i,]]] <- 1
}
```

```
# convert to graph
library(igraph)
g <- simplify(igraph::graph.adjacency(adj, mode="undirected"))

# Louvain clustering
km <- igraph::cluster_louvain(g)
```

```r
com <- km$membership
names(com) <- km$names

# distribution of cells per cluster
# head(com)
table(com)
```

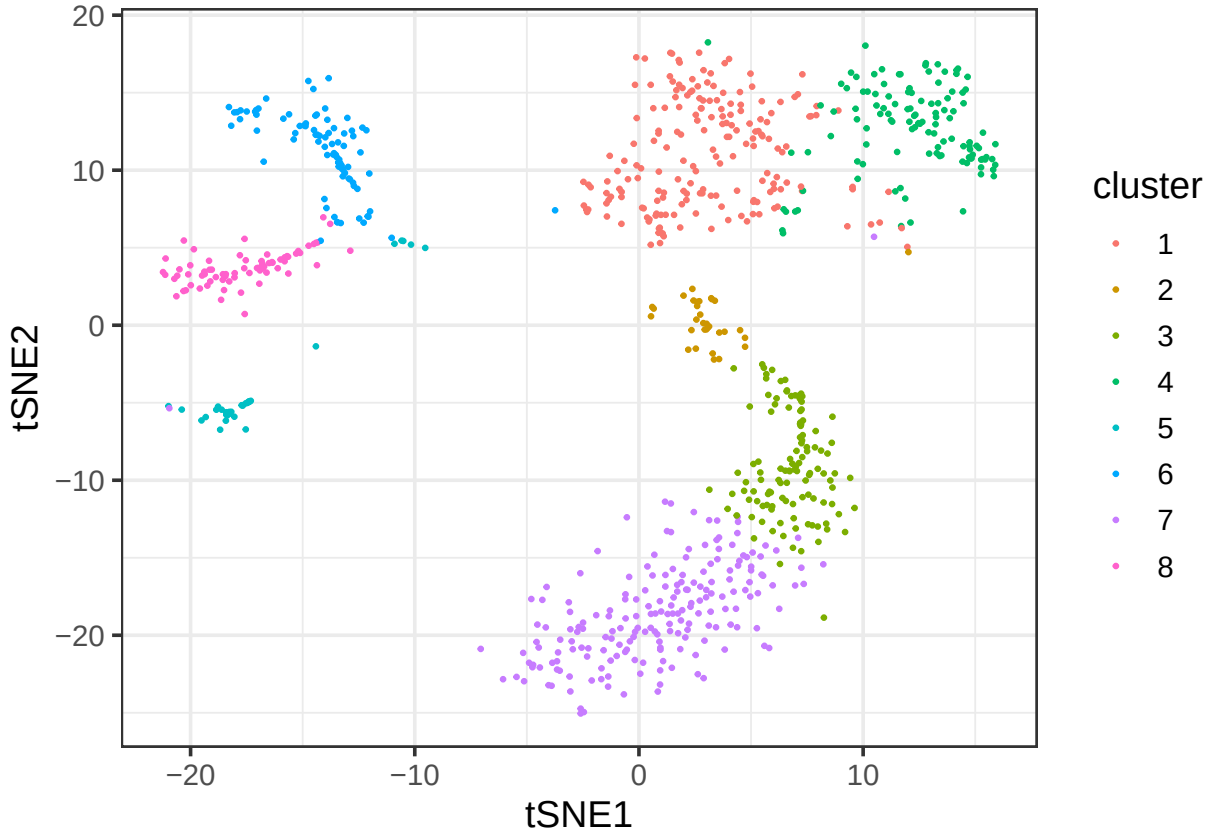
```
## com
##   1   2   3   4   5   6   7   8
## 186  32 116 118  35  84 194  71
```

Non-linear Dimensional Reduction (tSNE)

```r
library(Rtsne)
library(ggplot2)
library(tibble)
set.seed(8)

# t-distributed stochastic neighbor embedding
mat_tsne <- Rtsne(mat_pcs,  dims = 2, perplexity = 30, verbose = TRUE,
                  max_iter = 1000, check_duplicates = FALSE, is_distance = FALSE,
                  theta = 0.5, pca = FALSE, exaggeration_factor = 12)
```

```
## Read the 836 x 50 data matrix successfully!
## OpenMP is working. 1 threads.
## Using no_dims = 2, perplexity = 30.000000, and theta = 0.500000
## Computing input similarities...
## Building tree...
## Done in 0.20 seconds (sparsity = 0.158939)!
## Learning embedding...
## Iteration 50: error is 59.634355 (50 iterations in 0.16 seconds)
## Iteration 100: error is 55.675836 (50 iterations in 0.10 seconds)
## Iteration 150: error is 55.432495 (50 iterations in 0.10 seconds)
## Iteration 200: error is 55.408922 (50 iterations in 0.10 seconds)
## Iteration 250: error is 55.378263 (50 iterations in 0.10 seconds)
## Iteration 300: error is 0.809656 (50 iterations in 0.09 seconds)
## Iteration 350: error is 0.721920 (50 iterations in 0.09 seconds)
## Iteration 400: error is 0.704304 (50 iterations in 0.09 seconds)
## Iteration 450: error is 0.694573 (50 iterations in 0.09 seconds)
## Iteration 500: error is 0.687528 (50 iterations in 0.09 seconds)
## Iteration 550: error is 0.683101 (50 iterations in 0.08 seconds)
## Iteration 600: error is 0.680314 (50 iterations in 0.09 seconds)
## Iteration 650: error is 0.677074 (50 iterations in 0.09 seconds)
## Iteration 700: error is 0.675594 (50 iterations in 0.09 seconds)
## Iteration 750: error is 0.674554 (50 iterations in 0.09 seconds)
## Iteration 800: error is 0.673997 (50 iterations in 0.09 seconds)
## Iteration 850: error is 0.672254 (50 iterations in 0.09 seconds)
## Iteration 900: error is 0.671003 (50 iterations in 0.09 seconds)
## Iteration 950: error is 0.669636 (50 iterations in 0.09 seconds)
## Iteration 1000: error is 0.669066 (50 iterations in 0.09 seconds)
## Fitting performed in 1.88 seconds.
```

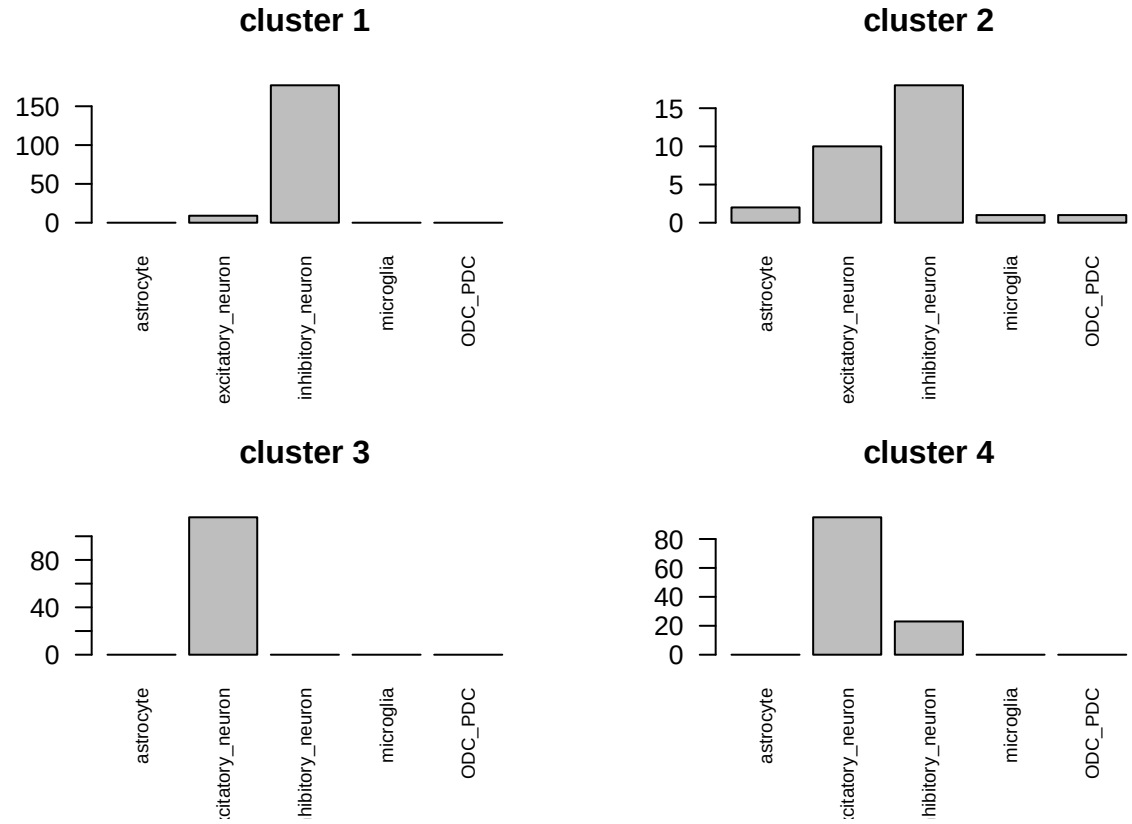
```
# convert to dataframe
df_tsne <- as.data.frame(mat_tsne$Y)
colnames(df_tsne) <- c("tSNE1", "tSNE2")
df_tsne$barcode <- rownames(mat_pcs)

# add cluster ids
df_tsne <- left_join(df_tsne, enframe(com), by = c("barcode" = "name")) %>%
        dplyr::rename(cluster = value) %>%
        mutate(cluster = as.factor(cluster))

# tsne plot
ggplot(df_tsne, aes(tSNE1, tSNE2)) +
        geom_point(aes(col = cluster), size = 0.5) +
        theme_bw(base_size = 14)
```

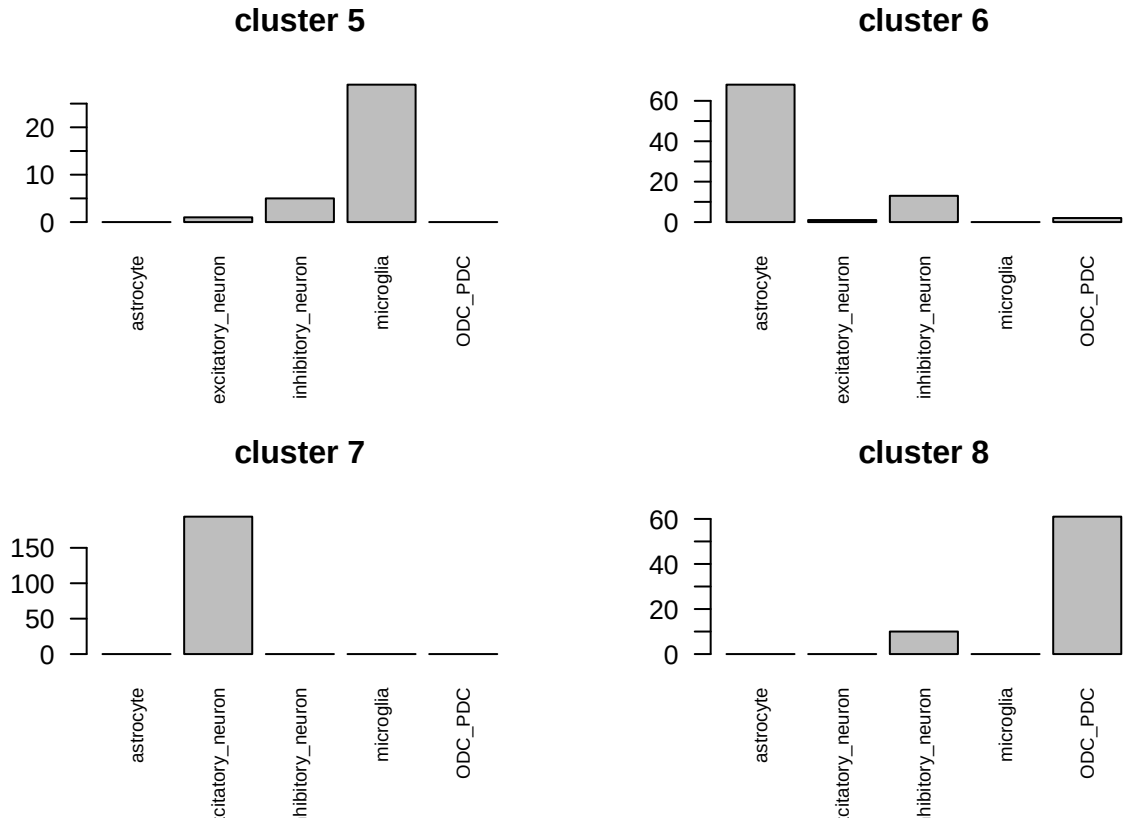


Cluster Identification

```
# read in cell labels file
cell_labels <- read.delim(
  file = "ATAC-seq.cell.labels.txt",
  header = FALSE
)
```

```r
# reformat dataframe
df_tsne_plot <- df_tsne
df_tsne_plot["type"] <- cell_labels
df_tsne_plot$type <- as.factor(df_tsne_plot$type)
# levels(df_tsne_plot$cluster)

# replace oligodendrocytes_polydendrocytes string
levels(df_tsne_plot$type)[levels(df_tsne_plot$type)=="oligodendrocytes_polydendrocytes"] <- "ODC_PDC"
categories <- levels(df_tsne_plot$type)

# plot distrbution of cell types in each cluster
par(mfrow=c(2,2))

# cluster 1 is astrocytes
for (i in levels(df_tsne_plot$cluster)) {
  plot(subset(df_tsne_plot, cluster == i)$type, main = paste0("cluster ", i), cex.names=0.70, las = 2)
}
```





Discussion

I normalized and log-transformed the peak matrix followed by latent semantic indexing to reduce the dimensions. Clustering was then performed using the k-nearest neighbor method. Visualization of the 836 single cells was done using a tSNE plot. There is a clear separation between cell types (see scatter plot matrix in the Machine Learning section). However, there is a group of excitatory neurons that is clustering together with the inhibitory neurons. These cells may have been mis-labeled. However, the abundance and clear clustering suggest another possibility. This group of cells is of a different type than either inhibitory neurons or excitatory neurons. Furthermore, there are several cells identified as inhibitory neurons that clustering together with the other cell types. These cells may have been mis-identified.

Machine Learning

```
# pull cell type information
df_tsne_sub <- df_tsne_plot["type"]
row.names(df_tsne_sub) <- df_tsne_plot$barcode

# convert dataframe. sample barcode are rownames
df_tsne_ml <- df_tsne
rownames(df_tsne_ml) <- df_tsne_ml$barcode
df_tsne_ml <- merge(df_tsne_ml, df_tsne_sub, by=0)
rownames(df_tsne_ml) <- df_tsne_ml$barcode
```

```
df_tsne_ml <- df_tsne_ml[c("tSNE1", "tSNE2", "type")]
ML <- df_tsne_ml
```

Split Data Into Training and Validation Sets

```
# separate data into validation and training sets
library(caret)
validation_index <- createDataPartition(ML$type, p = 0.80, list = FALSE)

validation <- ML[-validation_index,]
ML <- ML[validation_index,]
```

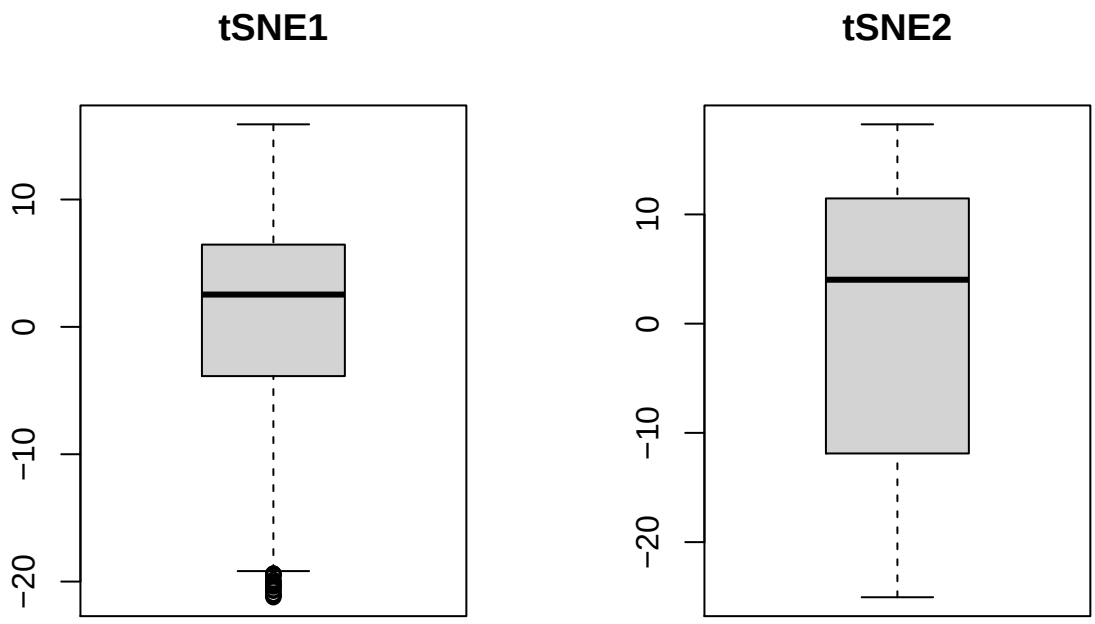
Training Set Data Exploration

```
# view data
# dim(ML)
# sapply(ML, class)
# levels(ML$type)

# proportion of cell types
percentage <- prop.table(table(ML$type)) * 100
cbind(freq=table(ML$type), percentage=percentage)
```

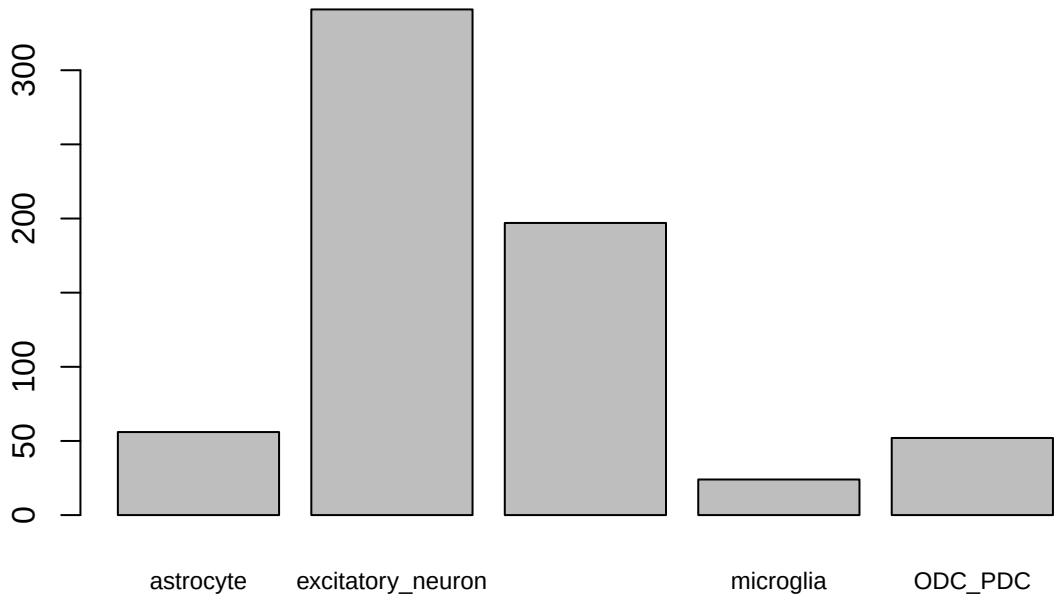
```
##                   freq percentage
## astrocyte           56   8.358209
## excitatory_neuron  341  50.895522
## inhibitory_neuron  197  29.402985
## microglia           24   3.582090
## ODC_PDC             52   7.761194
```

```
# summary(ML)
```

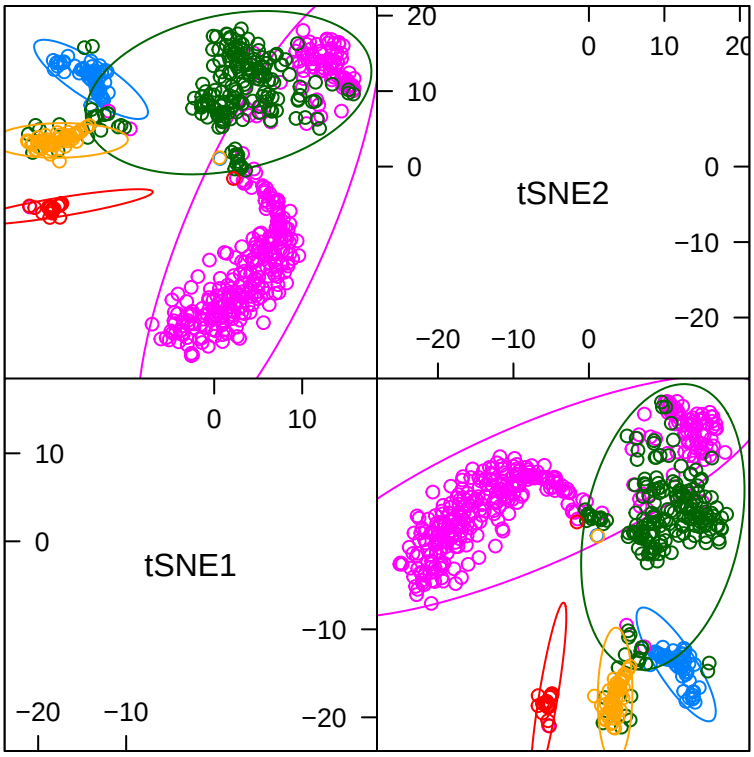
```
# univariate plots
x <- ML[,1:2]
y <- ML[,3]

# variation over tsne1 and tsne2
par(mfrow=c(1,2))
for (i in 1:2) {
  boxplot(x[,i], main=names(ML)[i])
}
```




```
# distribution of cell types
par(mfrow=c(1,1))
plot(y, cex.names=0.75)
```

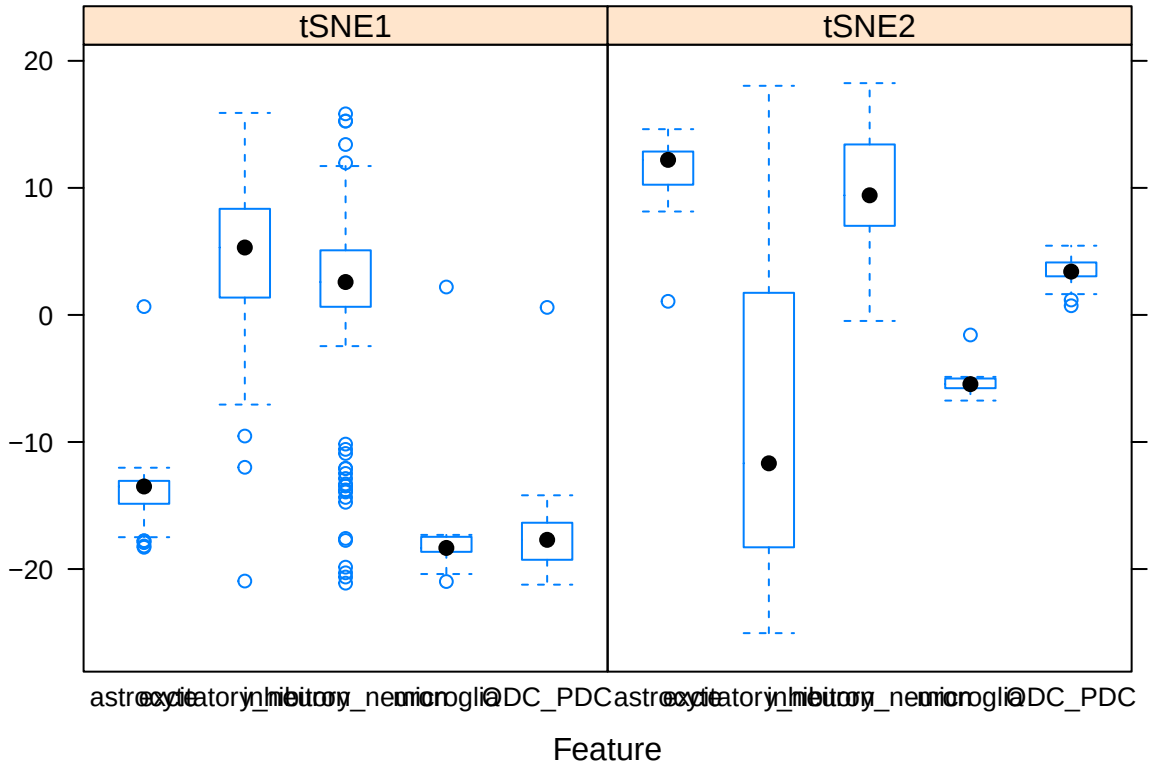



```
library("ellipse")
# multivariate plots
featurePlot(x=x, y=y, plot="ellipse")
```

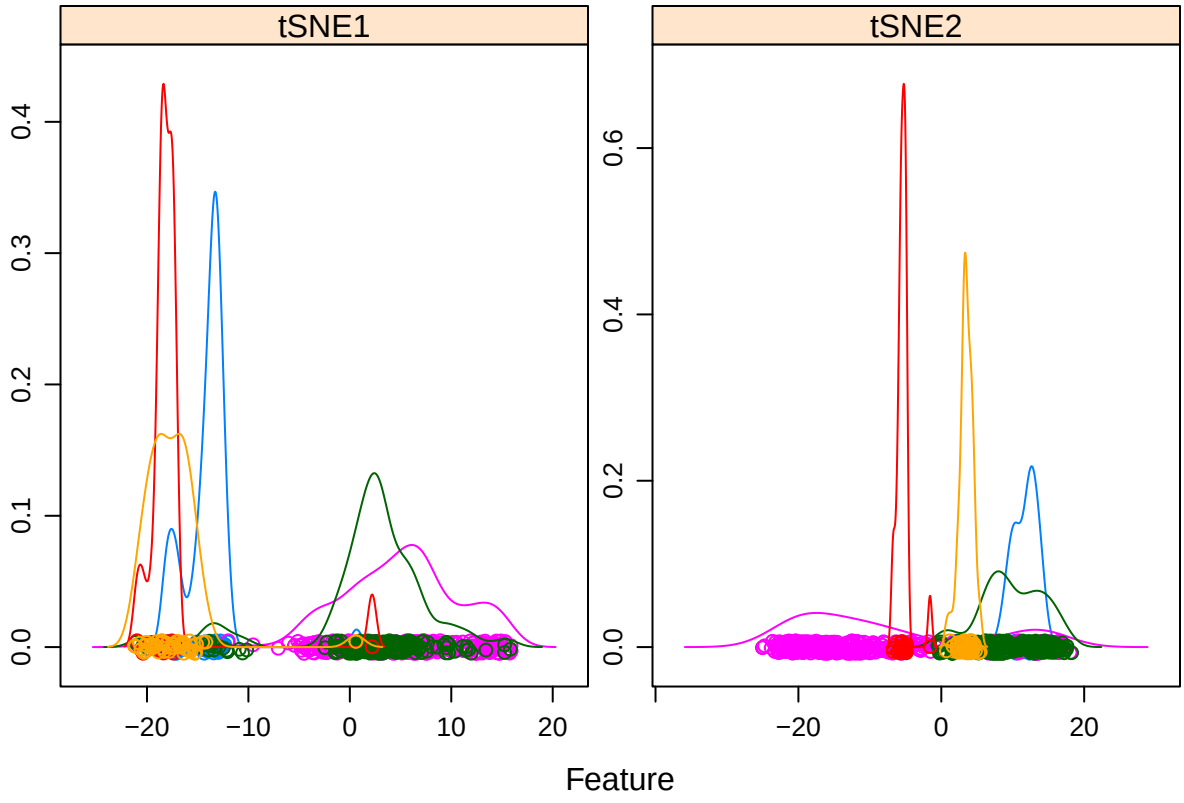


Scatter Plot Matrix

```
featurePlot(x=x, y=y, plot="box")
```




```
# density plots, distributions of cell types over tsne1 and tsne2
scales <- list(x=list(relation="free"), y=list(relation="free"))
featurePlot(x=x, y=y, plot="density", scales=scales)
```



Training Different Models

```
control <- trainControl(method="cv", number = 10)
metric <- "Accuracy"

## the models

# linear discriminant analysis
set.seed(8)
fit.lda <- train(type~., data=ML, method="lda", metric=metric, trControl=control)

# classification and regressiontrees
set.seed(8)
fit.cart <- train(type~., data=ML, method="rpart", metric=metric, trControl=control)

# k-Nearest Neighbors (kNN)
set.seed(8)
fit.knn <- train(type~., data=ML, method="knn", metric=metric, trControl=control)

# Support Vector Machines with a linear kernel
set.seed(8)
fit.svm <- train(type~., data=ML, method="svmRadial", metric=metric, trControl=control)

# Random Forest
```

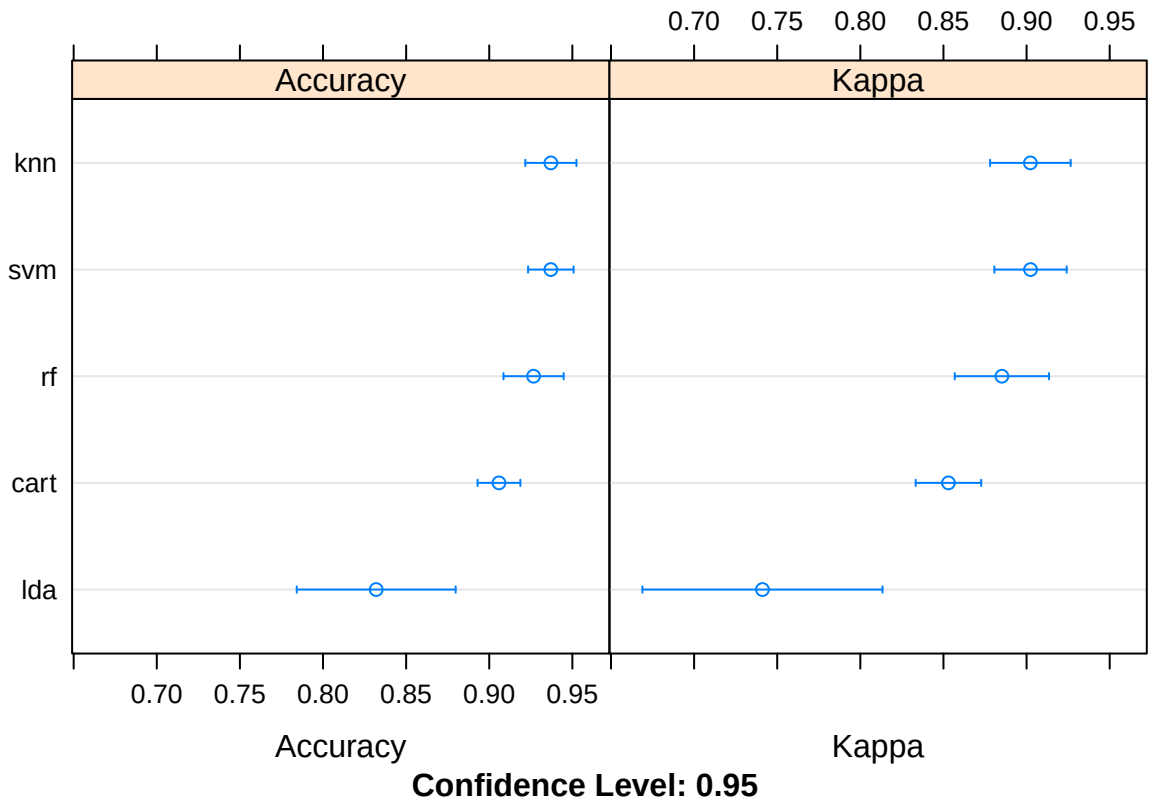
```r
set.seed(8)
fit.rf <- train(type~., data=ML, method="rf", metric=metric, trControl=control)
```

```
## note: only 1 unique complexity parameters in default grid. Truncating the grid to 1 .
```

```r
# Summarize accuracy of models
results <- resamples(list(lda=fit.lda, cart=fit.cart, knn=fit.knn, svm=fit.svm, rf=fit.rf))
summary(results)
```

```
##
## Call:
## summary.resamples(object = results)
##
## Models: lda, cart, knn, svm, rf
## Number of resamples: 10
##
## Accuracy
##           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## lda  0.7462687 0.7670494 0.8496156 0.8319919 0.8690857 0.9420290    0
## cart 0.8769231 0.8962795 0.9083916 0.9058424 0.9219710 0.9275362    0
## knn  0.9104478 0.9230769 0.9331357 0.9370665 0.9552239 0.9710145    0
## svm  0.9104478 0.9233683 0.9328358 0.9370452 0.9517120 0.9710145    0
## rf   0.8805970 0.9121125 0.9319173 0.9266633 0.9407308 0.9565217    0
##
## Kappa
##           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## lda  0.6148123 0.6477719 0.7613814 0.7411277 0.7998193 0.9122695    0
## cart 0.8061871 0.8391606 0.8570670 0.8529961 0.8767642 0.8861762    0
## knn  0.8595878 0.8788339 0.8962441 0.9022970 0.9308077 0.9554264    0
## svm  0.8595878 0.8809922 0.8965890 0.9023873 0.9254630 0.9558259    0
## rf   0.8130450 0.8612922 0.8922280 0.8851578 0.9085183 0.9328358    0
```

```r
# compare accuracy
dotplot(results)
```




```
# summarize best model
print(fit.rf)
```

```
## Random Forest
##
## 670 samples
##   2 predictor
##   5 classes: 'astrocyte', 'excitatory_neuron', 'inhibitory_neuron', 'microglia', 'ODC_PDC'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 603, 605, 605, 601, 603, 604, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.9266633  0.8851578
##
## Tuning parameter 'mtry' was held constant at a value of 2
```

Validating the Best Model

```
# validation of model
predictions.rf <- predict(fit.rf, validation)
confusionMatrix(predictions.rf, validation$type)
```

```
## Confusion Matrix and Statistics
##
##                   Reference
## Prediction          astrocyte excitatory_neuron inhibitory_neuron microglia
##   astrocyte                13                 0                 0         0
##   excitatory_neuron         0                82                 2         0
##   inhibitory_neuron         1                 3                46         0
##   microglia                 0                 0                 1         6
##   ODC_PDC                   0                 0                 0         0
##                   Reference
## Prediction          ODC_PDC
##   astrocyte               0
##   excitatory_neuron       0
##   inhibitory_neuron       1
##   microglia               0
##   ODC_PDC                11
##
## Overall Statistics
##
##                Accuracy : 0.9518
##                  95% CI : (0.9073, 0.979)
##     No Information Rate : 0.512
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9244
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: astrocyte Class: excitatory_neuron
## Sensitivity                   0.92857                   0.9647
## Specificity                   1.00000                   0.9753
## Pos Pred Value                1.00000                   0.9762
## Neg Pred Value                0.99346                   0.9634
## Prevalence                    0.08434                   0.5120
## Detection Rate                0.07831                   0.4940
## Detection Prevalence          0.07831                   0.5060
## Balanced Accuracy             0.96429                   0.9700
##                      Class: inhibitory_neuron Class: microglia Class: ODC_PDC
## Sensitivity                            0.9388          1.00000        0.91667
## Specificity                            0.9573          0.99375        1.00000
## Pos Pred Value                         0.9020          0.85714        1.00000
## Neg Pred Value                         0.9739          1.00000        0.99355
## Prevalence                             0.2952          0.03614        0.07229
## Detection Rate                         0.2771          0.03614        0.06627
## Detection Prevalence                   0.3072          0.04217        0.06627
## Balanced Accuracy                      0.9480          0.99687        0.95833
```

Discussion

Using the first two tSNE dimensions, I trained different classifiers using the 'caret' package in R to identify astrocyte cells against the other cell types. The k-nearest neighbor, support-vector networks, and random

forest models had the greatest accuracy (over 90%) for the training data set. The kappa statistic is also high for each model. Kappa is a more robust measure than accuracy because it considers the probability of an agreement happening by chance. Furthermore, it gives a better estimate of model's performance when the distribution of classes is skewed. Ultimately, I chose the random forest model to test the validation data set, which returned an accuracy greater than 95%. There is high specificity and sensitivity for every class.

Support-vector machines are better at handling outliers and provides information at the boundaries, but because our clusters are distinct, I removed this model from consideration. K-nearest neighbors requires the creation of a 'similarity' space and then determines a class by finding the nearest neighbor in that space. This makes KNN susceptible to noisy data, which we found is true from the mislabeled cells in question 1. Random forest is the best because it accumulates information from several decision tree algorithms with different subsets of the training data. This combats overfitting. A potential downside to random forest is that it cannot predict outside the range in the training data.

Future Directions

The first analysis I would perform given more time would be to differential accessibility analysis between the clusters. Assuming that each cell type has a different pattern of accessible chromatin region, we could possibly correlate the pattern to the cell type based on our cluster identification (using the provided labels). This analysis will tell us which regions of the chromatin are likely being actively transcribe for each cell type. We should expect to see for astrocytes regions of the chromosome that contain markers for mature astrocytes, such as Aldh1L1, AldoC, and Glt1. If we see a region that is unique for astrocytes, we could perform knockout studies to understand the importance of those regions for astrocyte development and/or function.

Conversely, if already know what pattern there should be for a given cell type, we could use that information to identify the cluster and compare that with the provided labels. This might help us resolve why there is a group of excitatory neurons clustering together with the inhibitory neurons.

If we find the genes associated with the significantly differential regions between clusters, we could perform a GO and KEGG analysis. From this analysis, we would see the function of genes and pathways being upregulated or downregulated. Glial cells, for example, should have a higher expression of genes linked to myelin compared to neuronal cells. In particular, we can scan through these differentially accessible regions for motifs to which regulatory elements bind. Correlating this information with cell type would reveal whether there are cell-type specific regulatory elements.

Another possible analysis would be cell trajectory which predicts the direction of changes from one cell state to another. It would be interesting to see how glial cells or astrocyte differentiate from progenitor cells over pseudotime.