Prediction of LncRNA Subcellular Localization with Deep Learning from Sequence Features

Brian Gudenas

February 16, 2018

# Prediction of LncRNA Subcellular Localization with Deep Learning from Sequence Features

library(readr)  
library(stringr)  
library(sleuth)  
library(dplyr)  
library(RUVSeq)  
library(RColorBrewer)  
library(pheatmap)

## Load Metadata

samples = read\_tsv("./Data/Meta/metadata.tsv")  
colnames(samples) = make.names(colnames(samples))  
Pair = read.table("./Data/Meta/Pair\_map.txt", sep="\t", header = TRUE)  
  
samples = samples[!is.na(match(samples$File.accession, Pair$Pair1)), ]  
  
samples = samples[samples$Biosample.subcellular.fraction.term.name == "nucleus" | samples$Biosample.subcellular.fraction.term.name == "cytosol", ]  
  
samples$Library.depleted.in[is.na(samples$Library.depleted.in)] = "rRNA"  
samples$Lib\_method = paste0(samples$Library.made.from,"\_", samples$Library.depleted.in)  
  
s2c = dplyr::select(samples, sample = File.accession, condition = Biosample.subcellular.fraction.term.name, cell = Biosample.term.name, platform = Platform, Lib = Lib\_method, reads = Read.length, warning = Audit.WARNING)  
  
## Remove any cell types with less than 4 total samples  
s2c = s2c[is.na(match(s2c$cell, names(table(s2c$cell))[table(s2c$cell) < 4] ) ), ]  
  
filespath = file.path(getwd(), "/Data/Kallisto/Quant", s2c$sample)  
s2c = dplyr::mutate(s2c, path = filespath)  
  
## Total samples  
nrow(s2c)

## [1] 93

## Samples per fraction  
table(s2c$condition)

##   
## cytosol nucleus   
## 45 48

## Samples per Library type  
table(s2c$Lib)

##   
## polyadenylated mRNA\_rRNA RNA\_polyadenylated mRNA, rRNA   
## 62 23   
## RNA\_rRNA   
## 8

## Cell types  
table(s2c$cell)

##   
## A549 endothelial cell of umbilical vein   
## 4 7   
## GM12878 H1-hESC   
## 8 4   
## HeLa-S3 HepG2   
## 7 16   
## HT1080 IMR-90   
## 4 4   
## K562 keratinocyte   
## 16 7   
## MCF-7 SK-MEL-5   
## 4 4   
## SK-N-DZ SK-N-SH   
## 4 4

## Kallisto.o\* is the standard output from the kallisto quant call for all samples  
output = read.delim(file = "./Data/Kallisto/Kallisto.o2139569", sep = "\n")  
pairmap = read.delim(file ="./Data/Meta/Pair\_map.txt")  
  
alignments = output[grepl("processed", output[,1]), ]  
  
pairmap$total\_reads = as.numeric(str\_trim(str\_replace\_all(unlist(lapply(str\_split(unlist(lapply(str\_split(unlist(lapply(str\_split(alignments, ", "), "[[", 1)), " reads"), "[[", 1)), "processed"),"[[",2)),",","")))  
pairmap$reads\_aligned = as.numeric(str\_replace\_all(unlist(lapply(str\_split(unlist(lapply(str\_split(alignments, ", "), "[[", 2)), " reads"), "[[", 1)), ",",""))  
  
pairmap$percent\_aligned = pairmap$reads\_aligned/pairmap$total\_reads\*100  
s2c$total\_reads = pairmap$total\_reads[match(s2c$sample, pairmap$Pair1)]  
s2c$reads\_aligned = pairmap$reads\_aligned[match(s2c$sample, pairmap$Pair1)]  
  
## total aligned reads  
sum(s2c$reads\_aligned)

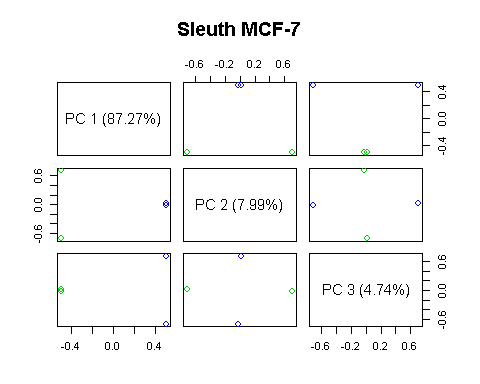
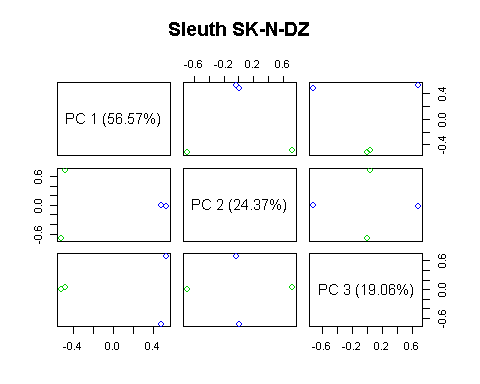
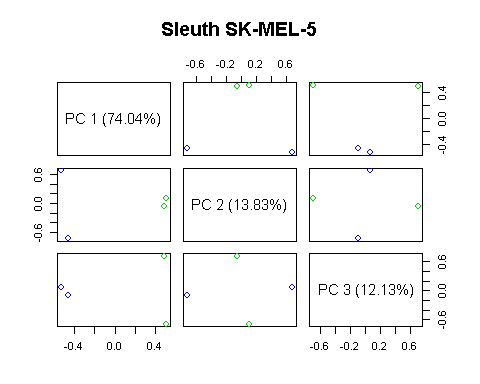
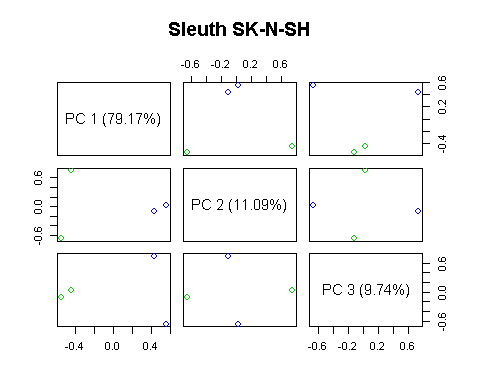
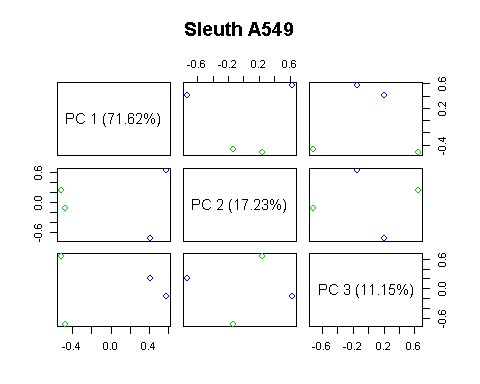
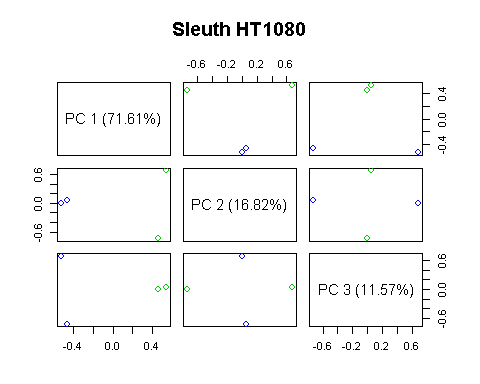
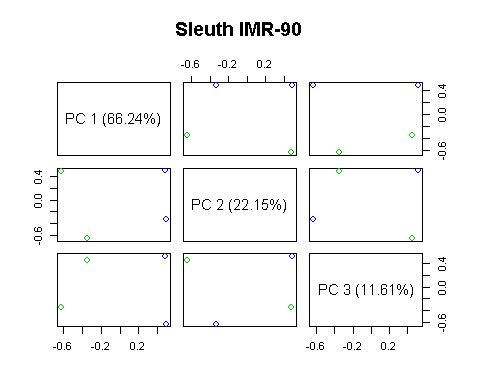
## [1] 5979055020

rm(alignments, samples, pairmap, Pair, output)  
write.csv(s2c, "./Data/Meta/Sample\_Meta.csv")

## DTE by cell type

Perform differential transcript expression analysis for each cell type. If a cell type contains more than a single RNA library construction protocol then we utilize library type as a covariate in the DTE test.

master =c()  
for ( i in unique(s2c$cell)){  
 s2c\_tmp = dplyr::filter(s2c, cell == i) %>% arrange(sample)  
   
 so = sleuth\_prep(s2c\_tmp, transformation\_function = function(x) log2(x+0.5), extra\_bootstrap\_summary = TRUE)  
 lds = plot\_pc\_variance(so)  
  
plot\_pca(so, color\_by = "condition", text\_labels = TRUE) + xlab(paste("PC1","%Var =", round(lds$data[1,2],2))) + ylab(paste("PC2","Var =", round(lds$data[2,2],2)))  
   
## if more than 1 RNA-seq library protocols were used account for covariate  
if (length(unique(s2c\_tmp$Lib)) > 1 ) {  
   
 so <- sleuth\_fit(so, ~ Lib + condition, 'full')  
 so <- sleuth\_fit(so, ~ Lib, 'reduced')  
 so = sleuth\_wt(so, "conditionnucleus")  
 wt\_results = sleuth\_results(so, 'conditionnucleus')  
 res =wt\_results[!is.na(wt\_results$pval), ]  
 res$target\_id = unlist(lapply(str\_split(res$target\_id, "\\."), "[[", 1))  
 out\_name = paste0("./Data/tables/DTE","\_",i, ".tsv")  
 write.table(res, out\_name, sep = "\t", quote = FALSE, row.names = FALSE)  
 res$cell = i  
   
 master = rbind(master,res)  
   
 } else {  
   
 Expr = so$obs\_norm %>% dplyr::select(target\_id, sample, est\_counts)  
 Emat = tidyr::spread(Expr, key = sample, est\_counts)  
 rownames(Emat) = unlist(lapply(str\_split(Emat[,1],"\\|"), "[[",1))  
 Emat = as.matrix(Emat[,-1])  
 #dim(Emat)  
 ## need to round est\_counts to integer for RUVseq  
 Emat = round(Emat)  
   
 # Filter transcripts   
 filter <- apply(Emat, 1, function(x) length(x[x > 5]) >= 2) ##remove non-expressed transcripts (5 counts in 2 samples)  
 filtered <- Emat[filter,]   
 EDASeq::plotPCA(filtered, col=as.numeric(as.factor(s2c\_tmp$condition))+2, cex=1.2, k=3, pch = as.numeric(as.factor(s2c\_tmp$Lib)), main = paste("Sleuth",i) )  
   
 so <- sleuth\_fit(so, ~ condition, 'full')  
 so <- sleuth\_fit(so, ~ 1, 'reduced')  
 so <- sleuth\_lrt(so, 'reduced', 'full')  
 so = sleuth\_wt(so, "conditionnucleus")  
 wt\_results = sleuth\_results(so, 'conditionnucleus')  
 res =wt\_results[!is.na(wt\_results$pval), ]  
 res$target\_id = unlist(lapply(str\_split(res$target\_id, "\\."), "[[", 1))  
 out\_name = paste0("./Data/tables/DTE","\_",i, ".tsv")  
 write.table(res, out\_name, sep = "\t", quote = FALSE, row.names = FALSE)  
 res$cell = i  
  
 master = rbind(master,res)  
 }  
  
}



write.csv(master, "./Data/tables/Master.csv", quote = FALSE, row.names = FALSE)  
rm(Emat,Expr,filtered,Pair,pairmap, res, samples, wt\_results, alignments, filter, set,set1,so, output, s2c\_tmp, filespath, i, out\_name, wts,x)

## LncRNA Filter

master = read.csv("./Data/tables/Master.csv", header = TRUE, stringsAsFactors = FALSE)  
  
annots = rtracklayer::readGFF("./Data/Annotation/gencodev27lncRNAs.gtf", version = 2L)  
annots = dplyr::filter(annots, type == "transcript")  
# remove trailing decimal  
annots$transcript\_id = lapply(strsplit(annots$transcript\_id, "\\."), "[[", 1)  
  
lnc\_match = match(master$target\_id, annots$transcript\_id)  
  
lncRNAs = master[!is.na(lnc\_match), ]  
  
## set up Expression matrix lncRNAs x cell\_type  
mat = matrix(nrow = length(unique(lncRNAs$target\_id)), ncol = length(unique(lncRNAs$cell)), data = NA)  
colnames(mat) = unique(lncRNAs$cell)  
rownames(mat) = unique(lncRNAs$target\_id)  
  
## Loop through and add fold-changes, leave NA if not detected  
for (i in 1:nrow(lncRNAs)){  
 l2fc = lncRNAs$b[i]  
 target = lncRNAs$target\_id[i]  
 celltype = lncRNAs$cell[i]  
   
 mat[rownames(mat) == target, colnames(mat) == celltype] = l2fc  
}  
  
wts = table(s2c$cell)/nrow(s2c)  
wts = wts[order(names(wts)) ]  
  
mat = mat[ , order(colnames(mat))]  
  
df = data.frame(target\_id = rownames(mat))  
df$l2fc = NA  
  
for (i in 1:nrow(mat)){  
   
 df$l2fc[i] = weighted.mean(mat[i,], wts, na.rm = TRUE)  
}

## Feature Extraction

source("./src/DeepLncRNA.R")  
mart = biomaRt::useMart("ensembl", dataset = "hsapiens\_gene\_ensembl")  
trans = GetSeq(df$target\_id, mart)  
trans = cbind(df$l2fc[match(trans$ensembl\_transcript\_id, df$target\_id)], trans)  
colnames(trans)[1] = "l2fc"  
df = FeatureExtract(trans)

## Create training dataset

rownames(df) = df$ensembl\_transcript\_id  
df = df %>% dplyr::select( -c(cdna, ensembl\_transcript\_id))  
  
drop\_vars = colnames(df[ ,4:ncol(df)])[colSums(df[ ,4:ncol(df)])==0]  
df = df %>% dplyr::select( -one\_of(drop\_vars))  
saveRDS(df, "./Data/feature\_set.rds")  
  
df = df[df$l2fc < 0 | df$l2fc > 2.8, ]  
df$Loc = "Nuclear"  
df$Loc[df$l2fc < 0] = "Cytosol"  
df$Loc = as.factor(df$Loc)  
  
set.seed(54321)  
  
spec = c(train = .7, test = .15, validate = .15)  
  
g = sample(cut(  
 seq(nrow(df)),   
 nrow(df)\*cumsum(c(0,spec)),  
 labels = names(spec)  
))  
  
res = split(df, g)  
  
lapply(res, dim)

## $train  
## [1] 6074 1584  
##   
## $test  
## [1] 1302 1584  
##   
## $validate  
## [1] 1302 1584

#Check splits dont contain duplicates == FALSE  
table(duplicated(c(rownames(res$train), rownames(res$test), rownames(res$validate))))

##   
## FALSE   
## 8678

# Save final dataframes  
saveRDS(res, "./Data/Training\_frames.rds")

## Deep Learning

library(h2o)  
localH2O = h2o.init(ip="localhost", port = 54321, startH2O = TRUE, nthreads = 24, max\_mem\_size = "84G")  
train\_hex = as.h2o(res$train[ ,-1])  
valid\_hex = as.h2o(res$validate[ ,-1])  
test\_hex = as.h2o(res$validate[ ,-1])  
  
hyper\_params <- list(  
 activation=c("RectifierWithDropout"),  
 hidden =list( c(250,125,75), c(100,50,25 ), c(64, 32, 16)),  
 input\_dropout\_ratio=c(0, 0.1,0.2, 0.3, 0.4),  
 l1 = c(1e-6,1e-5,1e-4, 1e-3, 0),  
 l2 = c(1e-6,1e-5,1e-4, 1e-3, 0)  
 )  
   
search\_criteria = list(strategy = "RandomDiscrete", max\_models = 800, seed=54321, stopping\_rounds=10, stopping\_tolerance=1e-3)  
   
 dl\_random\_grid <- h2o.grid(  
 algorithm="deeplearning",  
 grid\_id = "dl\_grid",  
 training\_frame=train\_hex,  
 validation\_frame = valid\_hex,  
 x=1:1582,  
 y=1583,  
 epochs= 400,  
 stopping\_metric="misclassification",  
 stopping\_tolerance=1e-3,   
 stopping\_rounds=10,  
 hyper\_params = hyper\_params,  
 search\_criteria = search\_criteria  
 )  
   
grid <- h2o.getGrid("dl\_grid",sort\_by="max\_per\_class\_error", decreasing=FALSE)  
   
best\_model <- h2o.getModel(grid@model\_ids[[1]])  
   
h2o.saveModel(best\_model, path="./Model/", force=TRUE)  
h2o.shutdown(FALSE)

library(randomForest)  
library(doParallel)  
library(caret)  
library(pROC)  
  
cluster <- makeCluster(detectCores())   
registerDoParallel(cluster)  
  
svm\_radial <- caret::train(Loc ~.,  
 data = res$train,  
 method = "svmRadial",  
 preProcess = c("center", "scale"),  
 trControl = trainControl(classProbs = TRUE),  
 allowParralel = TRUE,  
 tuneLength = 5)  
  
rf <- caret::train(Loc ~ ., data = res$train, method = "rf",  
 allowParralel = TRUE,  
 tuneLength = 5,  
 ntree = 101)  
  
stopCluster(cluster)  
registerDoSEQ()  
  
 save.image("./Data/ML\_models.RData")

## Machine learning model comparison on validation set

load(file = "./Data/ML\_models.RData")  
library(h2o)  
library(caret)  
  
localH2O = h2o.init(ip="localhost", port = 54321, startH2O = TRUE, nthreads = 1, max\_mem\_size = "2G")

##   
## H2O is not running yet, starting it now...  
##   
## Note: In case of errors look at the following log files:  
## C:\Users\Brian\AppData\Local\Temp\RtmpaeE3kF/h2o\_Brian\_started\_from\_r.out  
## C:\Users\Brian\AppData\Local\Temp\RtmpaeE3kF/h2o\_Brian\_started\_from\_r.err  
##   
##   
## Starting H2O JVM and connecting: . Connection successful!  
##   
## R is connected to the H2O cluster:   
## H2O cluster uptime: 2 seconds 44 milliseconds   
## H2O cluster version: 3.16.0.2   
## H2O cluster version age: 2 months and 26 days   
## H2O cluster name: H2O\_started\_from\_R\_Brian\_evq212   
## H2O cluster total nodes: 1   
## H2O cluster total memory: 1.78 GB   
## H2O cluster total cores: 4   
## H2O cluster allowed cores: 1   
## H2O cluster healthy: TRUE   
## H2O Connection ip: localhost   
## H2O Connection port: 54321   
## H2O Connection proxy: NA   
## H2O Internal Security: FALSE   
## H2O API Extensions: Algos, AutoML, Core V3, Core V4   
## R Version: R version 3.3.2 (2016-10-31)

#DNN = h2o.loadModel("./Data/dl\_grid\_model\_190")  
DNN = h2o.loadModel("./Data/Models/dlgrid\_model\_483")  
train\_hex = as.h2o(res$train)

##   
 |   
 | | 0%  
 |   
 |=================================================================| 100%

valid\_hex = as.h2o(res$validate)

##   
 |   
 | | 0%  
 |   
 |=================================================================| 100%

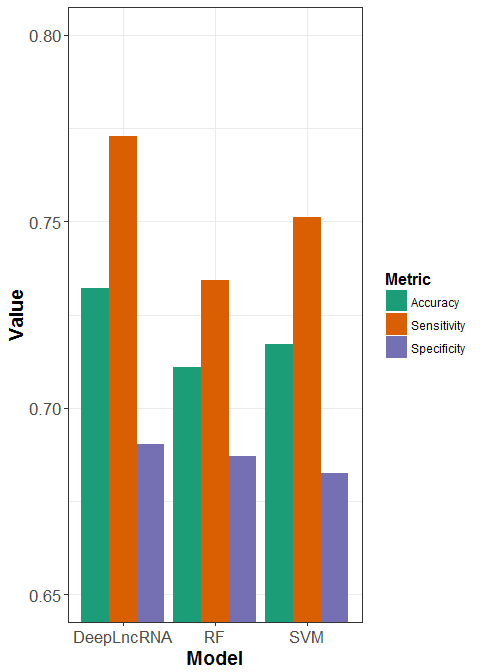
test\_hex = as.h2o(res$test)

##   
 |   
 | | 0%  
 |   
 |=================================================================| 100%

## Make plot of validation accuracies  
rf\_vals = predict(rf, res$validate)  
rf\_conf = confusionMatrix(rf\_vals, res$validate$Loc, positive = "Nuclear")  
  
svm\_vals = predict(svm\_radial, res$validate)  
svm\_conf = confusionMatrix(svm\_vals, res$validate$Loc, positive = "Nuclear")  
  
DNN\_vals = predict(DNN, valid\_hex)

##   
 |   
 | | 0%  
 |   
 |=================================================================| 100%

DNN\_conf = confusionMatrix(as.vector(DNN\_vals$predict), res$validate$Loc, positive = "Nuclear")  
  
  
GetMets = function(confMat){  
 ## get acc, sens, spec from caret confusion matrix  
 acc = confMat$overall[1]  
 spec = confMat$table[1,1]/(confMat$table[1,1] + confMat$table[2,1])  
 sens = confMat$table[2,2]/(confMat$table[2,2] + confMat$table[1,2])  
 mets = c(acc,sens,spec)  
 names(mets) = c("Accuracy","Sensitivity","Specificity")  
 return(mets)  
}  
  
valid\_mets = data.frame(RF = GetMets(rf\_conf), SVM = GetMets(svm\_conf), DeepLncRNA = GetMets(DNN\_conf), Metrics = names(GetMets(DNN\_conf))) %>%   
 tidyr::gather(valid\_mets, Metrics)  
colnames(valid\_mets) = c("Metric","Model","Value")  
  
ggplot(data = valid\_mets, aes(as.factor(Model), Value, fill = Metric) ) +  
 geom\_bar( stat = "identity", position = "dodge") +  
 scale\_fill\_brewer(palette = "Dark2") +  
 coord\_cartesian(ylim=c(0.65,0.8)) +  
 xlab("Model") +  
 theme\_bw() +   
 theme(axis.text=element\_text(size=13),  
 axis.title=element\_text(size=14,face="bold"), legend.title = element\_text(size=12,face="bold"))



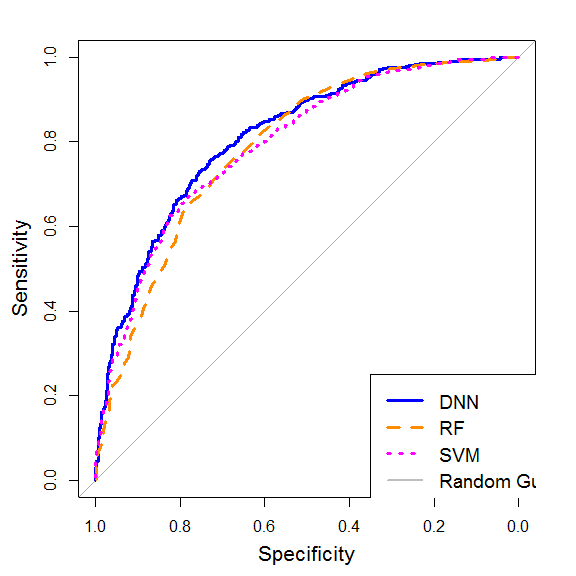
ggsave("./Figures/Model\_Comparison.pdf")

## Model evalutation on test set

dnn\_test = predict(DNN, test\_hex)

##   
 |   
 | | 0%  
 |   
 |=================================================================| 100%

rf\_test = predict(rf, res$test, "prob")  
svm\_test = predict(svm\_radial, res$test, "prob")  
  
  
dnn\_roc = pROC::roc(predictor=as.vector(dnn\_test$Nuclear),  
 response=res$test$Loc,  
 levels=levels(res$test$Loc) )  
  
plot(dnn\_roc, col = "blue", cex.lab = 1.3, cex.axis = 1, lwd=3)  
  
rf\_roc =pROC::roc(predictor=as.vector(rf\_test$Nuclear),  
 response=res$test$Loc,  
 levels=levels(res$test$Loc) )  
  
plot(rf\_roc, add = TRUE, col = "darkorange",lty=2, lwd = 3)  
  
svm\_roc = pROC::roc(predictor=as.vector(svm\_test$Nuclear),  
 response=res$test$Loc,  
 levels=levels(res$test$Loc) )  
plot(svm\_roc, add = TRUE, col = "magenta",lty =3, lwd = 3)  
  
  
legend(0.35, 0.25, legend=c("DNN", "RF", "SVM","Random Guess"),  
 col=c("blue", "darkorange","magenta","grey"), lty = c(1, 2,3,1), lwd = c(3,3,3,2), cex = 1.2)



## Test set performance metrics

rf\_test = predict(rf, res$test)  
rf\_conf = confusionMatrix(rf\_test, res$test$Loc, positive = "Nuclear")  
  
svm\_test = predict(svm\_radial, res$test)  
svm\_conf = confusionMatrix(svm\_test, res$test$Loc, positive = "Nuclear")  
  
DNN\_test = predict(DNN, test\_hex)

##   
 |   
 | | 0%  
 |   
 |=================================================================| 100%

DNN\_conf = confusionMatrix(as.vector(DNN\_test$predict), res$test$Loc, positive = "Nuclear")  
  
test\_mets = data.frame(t(data.frame(RF = GetMets(rf\_conf), SVM = GetMets(svm\_conf), DNN = GetMets(DNN\_conf)) ))  
test\_mets$AUC = c(rf\_roc$auc, svm\_roc$auc, dnn\_roc$auc)  
  
rf\_mcc = mltools::mcc(preds = as.numeric(rf\_test)-1, actuals = as.numeric(res$test$Loc)-1)  
svm\_mcc = mltools::mcc(preds = as.numeric(svm\_test)-1, actuals = as.numeric(res$test$Loc)-1)  
dnn\_mcc = mltools::mcc(preds = as.numeric(as.factor(as.vector(DNN\_test$predict)))-1, actuals = as.numeric(res$test$Loc)-1)  
  
test\_mets$MCC = c(rf\_mcc, svm\_mcc, dnn\_mcc)  
print(round(test\_mets, 3))

## Accuracy Sensitivity Specificity AUC MCC  
## RF 0.716 0.726 0.707 0.788 0.433  
## SVM 0.713 0.721 0.706 0.795 0.427  
## DNN 0.733 0.807 0.662 0.812 0.473

write.csv(round(test\_mets, 3), "./Data/test\_metrics.csv")  
h2o.shutdown(FALSE)

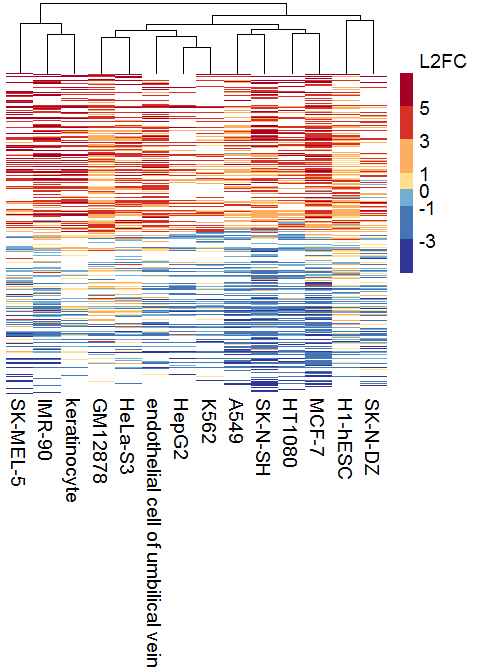
## [1] TRUE

## Differentially localized lncRNA heatmap

mat\_match = match(rownames(mat), rownames(df))  
mat = mat[!is.na(mat\_match), ]  
dim(mat)

## [1] 8678 14

div\_cols = unlist(list(color = brewer.pal(11, "RdYlBu")))[11:1]  
## truncate ends of dist to avoid saturation for visualization  
mat2 = mat  
mat2[mat2 >= 7] = 7  
mat2[mat2 <= -5] = -5  
mat\_breaks = c(min(mat2, na.rm = TRUE), -3, -1, 0, 1, 3, 5 ,max(mat2, na.rm = TRUE))  
div\_cols = div\_cols[c(1,2,3,7,8,10,11)]  
  
mat2 = mat2[order(rowMeans(mat2, na.rm = TRUE), decreasing = TRUE), ]  
  
#pdf("./Figures/lncRNA\_heatmap.pdf")  
# fig looks diff in rmarkdown, plot to PDF for published version  
pheatmap(mat2, cluster\_rows = FALSE , cluster\_cols = TRUE, labels\_row = "", breaks = mat\_breaks, color = div\_cols, legend\_breaks = c(-3, -1,0, 1, 3,5, max(mat2, na.rm = TRUE)), legend\_labels = c(-3, -1, 0, 1, 3,5, "L2FC\n"), cex = 1.2 )



#dev.off()

## Genomic predictions

Gtrans = GetSeq(annots$transcript\_id, mart)  
Gtrans = Gtrans[is.na(match(Gtrans$ensembl\_transcript\_id, rownames(res$train))), ]  
Gdf = FeatureExtract(Gtrans)  
preds = DeepLncRNA(Gdf)

##   
## H2O is not running yet, starting it now...  
##   
## Note: In case of errors look at the following log files:  
## C:\Users\Brian\AppData\Local\Temp\Rtmpe6nqnW/h2o\_Brian\_started\_from\_r.out  
## C:\Users\Brian\AppData\Local\Temp\Rtmpe6nqnW/h2o\_Brian\_started\_from\_r.err  
##   
##   
## Starting H2O JVM and connecting: . Connection successful!  
##   
## R is connected to the H2O cluster:   
## H2O cluster uptime: 2 seconds 30 milliseconds   
## H2O cluster version: 3.16.0.2   
## H2O cluster version age: 2 months and 25 days   
## H2O cluster name: H2O\_started\_from\_R\_Brian\_xtp717   
## H2O cluster total nodes: 1   
## H2O cluster total memory: 0.89 GB   
## H2O cluster total cores: 4   
## H2O cluster allowed cores: 1   
## H2O cluster healthy: TRUE   
## H2O Connection ip: localhost   
## H2O Connection port: 54321   
## H2O Connection proxy: NA   
## H2O Internal Security: FALSE   
## H2O API Extensions: Algos, AutoML, Core V3, Core V4   
## R Version: R version 3.3.2 (2016-10-31)   
##   
##   
 |   
 | | 0%  
 |   
 |=================================================================| 100%  
##   
 |   
 | | 0%  
 |   
 |=================================================================| 100%

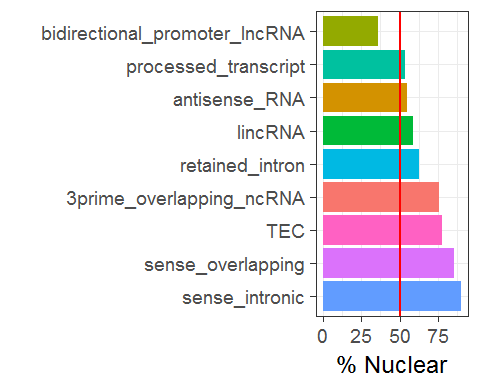
Gdf = cbind(preds, Gdf)  
Gdf$transcript\_biotype = Gtrans$transcript\_biotype  
dim(Gdf)

## [1] 21816 1632

## check overall predictions  
table(preds$predict)

##   
## Cytosol Nuclear   
## 8877 12939

pos = Gdf %>%  
 group\_by(transcript\_biotype) %>%  
 summarise(avg = (sum(predict == "Nuclear")/n())\*100, count = n()) %>%  
 filter(count > 20) %>%  
 arrange(desc(avg)) %>%  
 dplyr::select(transcript\_biotype)  
pos = as.character(pos$transcript\_biotype)  
  
Gdf %>%   
 group\_by(transcript\_biotype) %>%   
 summarise(avg = (sum(predict == "Nuclear")/n())\*100, count = n()) %>%   
 filter(count > 20) %>%   
 arrange(desc(avg)) %>%   
 ggplot( aes(x = as.factor(transcript\_biotype), avg, fill = as.factor(transcript\_biotype))) +  
 geom\_bar(stat="identity", position = "dodge") +  
 xlab("") +  
 ylab("% Nuclear") +  
 coord\_flip() +  
 geom\_hline(yintercept = 50, col = "red", lwd = 1) +  
 scale\_x\_discrete(limits = pos) +  
 guides(fill=FALSE) +  
 theme\_bw(base\_size = 18)



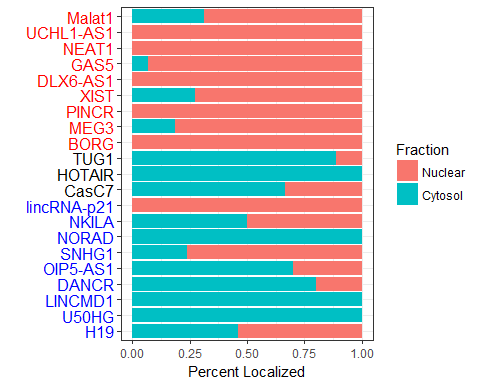
ggsave("./Figures/genomic\_biotype\_loc.pdf")

## Predictions on lncRNAs with known localizations

library(dplyr)  
library(reshape2)  
library(ggplot2)  
# annots ------------------------------------------------------------------  
known = read.csv("./Data/annotated\_lncRNAs.csv")  
  
trans = Gene2Transcript(known$ID, mart)  
seqs = GetSeq(trans$ensembl\_transcript\_id, mart)  
  
mousemart = biomaRt::useMart("ensembl", dataset = "mmusculus\_gene\_ensembl")  
transM = Gene2Transcript(known$ID[known$Species == "Mouse"], mart = mousemart)  
seqM = GetSeq(transM$ensembl\_transcript\_id, mart = mousemart)  
  
Borg = seqM  
Borg$ensembl\_transcript\_id = "BORG"  
Borg$cdna = stringr::str\_replace\_all(known$seq[6], "\n","")  
Borg$transcript\_length = nchar(as.character(Borg$cdna))  
Borg$GC\_content = sum(oligonucleotideFrequency(DNAString(Borg$cdna), 1)[c(2,3)])/ sum(oligonucleotideFrequency(DNAString(Borg$cdna), 1))  
  
  
fullseq = rbind(Borg,seqM,seqs)  
fullseq$chromosome\_name[1:2] = 0 ## set to zero bc not on human chroms (mouse lncRNAs)  
fullseq = fullseq %>% filter(transcript\_biotype != "protein\_coding") %>%   
 filter(transcript\_biotype != "nonsense\_mediated\_decay") %>%   
 filter(transcript\_length >= 200)  
  
df = FeatureExtract(fullseq)  
  
trans = rbind(transM, trans, c("BORG","BORG","lincRNA"))  
  
################  
#build new DNN excluding all known LncRNAs from train/valid set  
 res$train = res$train[is.na(match(rownames(res$train), trans$ensembl\_transcript\_id)), ]  
res$validate = res$validate[is.na(match(rownames(res$validate), trans$ensembl\_transcript\_id)), ]  
  
library(h2o)  
localH2O = h2o.init(ip="localhost", port = 54321, startH2O = TRUE, nthreads = -1, max\_mem\_size = "15G")

##   
## H2O is not running yet, starting it now...  
##   
## Note: In case of errors look at the following log files:  
## C:\Users\Brian\AppData\Local\Temp\Rtmp23uYUO/h2o\_Brian\_started\_from\_r.out  
## C:\Users\Brian\AppData\Local\Temp\Rtmp23uYUO/h2o\_Brian\_started\_from\_r.err  
##   
##   
## Starting H2O JVM and connecting: . Connection successful!  
##   
## R is connected to the H2O cluster:   
## H2O cluster uptime: 2 seconds 125 milliseconds   
## H2O cluster version: 3.16.0.2   
## H2O cluster version age: 2 months and 26 days   
## H2O cluster name: H2O\_started\_from\_R\_Brian\_nwr240   
## H2O cluster total nodes: 1   
## H2O cluster total memory: 13.33 GB   
## H2O cluster total cores: 4   
## H2O cluster allowed cores: 4   
## H2O cluster healthy: TRUE   
## H2O Connection ip: localhost   
## H2O Connection port: 54321   
## H2O Connection proxy: NA   
## H2O Internal Security: FALSE   
## H2O API Extensions: Algos, AutoML, Core V3, Core V4   
## R Version: R version 3.3.2 (2016-10-31)

h2o.no\_progress()  
train\_hex = as.h2o(res$train[ ,-1])  
valid\_hex = as.h2o(res$validate[ ,-1])  
  
DNN = h2o.deeplearning(  
 training\_frame=train\_hex,  
 validation\_frame = valid\_hex,  
 x=1:1582,  
 y=1583,  
 epochs= 2000,  
 stopping\_metric="misclassification",  
 stopping\_tolerance=1e-3,   
 stopping\_rounds=10,  
 activation = "RectifierWithDropout",  
 hidden = c(64,32,16),  
 input\_dropout\_ratio = 0.3,  
 l1 = 1e-3,  
 l2 = 1e-5)  
  
preds = predict(DNN, as.h2o(df))  
preds = as.data.frame(preds)  
rownames(preds) = df$ensembl\_transcript\_id  
  
preds$gene = trans$ensembl\_gene\_id[match(rownames(preds), trans$ensembl\_transcript\_id)]  
preds$name = known$Name[match(preds$gene, known$ID)]  
preds$Loc = known$Localized[match(preds$gene, known$ID)]  
  
meds = preds %>%   
 group\_by(gene) %>%   
 summarise(Nuclear = sum(predict == "Nuclear")/n())   
  
meds$Loc = known$Localized[match(meds$gene, known$ID)]  
meds = meds[order(meds$Loc), ]  
meds$Cytosol = 1-meds$Nuclear  
nams= known$Name[match( meds$gene, known$ID)]  
nams = nams[!is.na(nams)]  
meds$Name = nams  
  
medL = melt(meds[,c(2,4,5)], id.vars = "Name")  
  
colvec = rep("red", nrow(meds))  
colvec[meds$Loc == "Cytosol"] = "blue"  
colvec[meds$Loc == "Dual"] = "black"  
  
ggplot(medL, aes(x = Name, value, fill = as.factor(variable))) +  
 geom\_bar(stat="identity") +  
 xlab("") +  
 ylab("Percent Localized") +  
 coord\_flip() +  
 scale\_x\_discrete(limits = meds$Name) +  
 scale\_fill\_discrete(name = "Fraction") +  
 theme\_bw() +  
 theme(axis.text.y=element\_text(colour = colvec, size = 12))



ggsave("./Figures/annotated\_lncRNAs.pdf", width = 5, height = 7, units = "in")