Data plan for expProb

Bingfang

2/6/2025

## 1. Descriptions

The ultimate goal of this project is to identify a list of truly expressed genes in a specific cell type, which will be used for the functional analysis of biological processes and pathways in that cell type.

In previous works, I applied a method developed by Grakskit et al. (Statistics, 2023) to calculate the expression probability (expProb) for each gene in each state (“null”, “off”, “on”) in each lung cell type (see lung\_geneExp\_barcodeV4.Rmd).

To validate expProb, I calculated the expression probability of “on” state for NS-Forest genes (including marker genes and binary genes) in their targeted cell type—the cell type where they were identified as marker genes. However, about 25% of marker genes have “on” state expression probabilities lower than 75% in their targeted cell type (see lung\_geneExp\_barcodeV5.Rmd).

When calculating the expression probability of “on” state for differentially expressed (DE) genes. About 10% of DE genes have “on” state expression probabilities lower than 75% in the cell type where they are known to be expressed, see lung\_geneExp\_barcodeV6.Rmd).

It is a more stringent test to validate expProb using NS-Forest genes than using DE genes. Understanding why a subset of NS-Forest genes has predicted low expression probabilities could help us determine whether to use this expProb model to generate the expressed gene list for our knowledge graph and whether modification of model can improve the performance.

## 2. Data analysis plan

### 2.1 The purposes of this Rmarkdown

The question is: Why does a subset of NS-Forest genes have a low expression probability in their target cell type? The hypothesis is that since NS-Forest marker genes are only expressed in one or a few cell types, the model might classify their expression as “off” when their expression levels are low. The purposes of this Rmarkdown is to investigate if the model recall and precision can be improved when the combined probability of “on and off” states is used to identify expressed genes?

### 2.2 The rationales behind each analysis step

1. Check the “on” state probability and calculate the confusion matrix with three probability cut-off (0.75, 0.90, 0.95). Between recall and precision, we prioritize precision since we need to avoid false positives (FP). However, The precision can not be calculated accurately here as some marker genes are also expressed in cell types other than their target cell types.
2. Check the “on and off” state probability and calculate the confusion matrix with three probability cut-off. Compare the results with step 1.
3. Merge the tables from steps 1 and 2. Manually check 10 genes that have low expression probability in the “on” state. Looking for the explanation why the expression probability is low for this subset of markers.
4. Artificially set π-null and π-off to 0.33 if the “on and off” state probability remains lower than 75% for NS-Forest markers. Investigate if the pre-trained data have wrong prior probability.

## 3. Set environment

#install.packages("sads")   
library(sads)  
#install.packages("BiocParallel")  
library(BiocParallel)  
library(Seurat)  
library(org.Hs.eg.db)  
#install.packages("Matrix")  
library(Matrix)  
# package for goenrichment  
library(clusterProfiler)  
library(AnnotationDbi)  
# package for Venn diagram  
library(gplots, lib.loc = "/usr/local/apps/R/4.4/site-library\_4.4.2")  
library(tidyverse)  
# package for plot  
install.packages("ggplot2")   
library(ggplot2)  
# package for confusion matrix  
install.packages('caret')  
library(caret)  
  
  
# Get gene-specific distributions  
# Publication:https://academic.oup.com/biostatistics/article/23/4/1150/6622011  
# https://github.com/igrabski/scRNAseq-cell-type  
load('params\_EM\_81020.rda')   
pi.all.g <- params[[1]] # π-null,π-off, probability of null and off state.   
g.on.all.g <- params[[2]] # μ0j shift from μ0   
g.off.all.g <- params[[3]] # μ1j shift from μ1  
a.all.g <- params[[4]] # αj⁠  
sigma.all.g <- params[[5]] # σ0j,σ1j  
mu.g<-c(-12.73153,-8.301349) # global means for gene rates on a log scale, (μ0 , μ1)

## 4. Data

### load HLCA core data from cellXgene

## Read .rds file  
#hlca = readRDS("/gpfs/gsfs10/users/xubr/cellXgene/scRNAseq-cell-type/data/b351804c-293e-4aeb-9c4c-043db67f4540\_hlca\_core.rds")  
  
## Preview the dataset  
#summary(hlca@assays$RNA@meta.features$feature\_type)  
#head(hlca@assays$RNA@meta.features$feature\_name)  
#table(hlca@meta.data$cell\_type\_ontology\_term\_id)  
#table(hlca@meta.data$cell\_type)  
#table(hlca@meta.data$ann\_finest\_level)

### load data of NS-forest marker and binary marker genes, calculate prior probability.

# The full table of gene symbol and Ensemble ID  
gencode<-read.csv("./data/gencode.v47.annotation.csv")  
gene\_Ensemble <-gencode %>% select(c(`gene\_name`, "ENSEMBLID"))  
  
# load NS-forest marker gene and binary marker gene  
#list.files("./data/")  
binary\_gene<-read.csv("./data/hlcacore\_ann\_finest\_label\_all\_supplementary.csv", sep=",")  
NSmarker\_gene<-read.csv("./data/hlcacore\_ann\_finest\_level.markers.csv", sep = ',')  
  
# change format of NSmarker\_gene  
#NSmarker\_gene$NSForest\_markers[1]  
NSmarker\_gene$NSForest\_markers <-gsub("]", "",NSmarker\_gene$NSForest\_markers)  
NSmarker\_gene$NSForest\_markers <-gsub("\\[","",NSmarker\_gene$NSForest\_markers)  
NSmarker\_gene$NSForest\_markers <-gsub(" ","",NSmarker\_gene$NSForest\_markers)  
NSmarker\_gene$NSForest\_markers <-gsub("'","",NSmarker\_gene$NSForest\_markers)  
NSmarker\_gene <- NSmarker\_gene %>%  
 separate\_rows(NSForest\_markers, sep = ",")  
  
# merge to obtain ENSEMBLID  
NSmarker<-merge(NSmarker\_gene,gene\_Ensemble, by.x = "NSForest\_markers", by.y = "gene\_name", all.x = TRUE)   
geneList\_bi<-merge(binary\_gene,gene\_Ensemble, by.x = "binary\_genes", by.y = "gene\_name", all.x = TRUE)   
#gencode[gencode$gene\_name == "C1orf194",] ## 4 NS-forest marker don't have EnsemblID  
  
  
# generate a data frame of π-null,π-off, which is the probabilities of null and off states.   
pi\_df<-as.data.frame(pi.all.g)  
pi\_df$`ENSEMBLID`<-rownames(pi\_df)  
  
# Obtain prior probability of expression for the NS-forest marker genes   
prior\_marker<-merge(NSmarker, pi\_df, by = "ENSEMBLID", all.x = TRUE)   
prior\_marker$prior\_prob\_off <- prior\_marker$V1 + prior\_marker$V2  
prior\_marker$prior\_prob\_on <- 1-(prior\_marker$V1 + prior\_marker$V2)  
  
# visualize prior probabilities for the NS-forest marker genes   
prior\_marker<-prior\_marker %>% select(c("ENSEMBLID","NSForest\_markers", "clusterName","clusterSize","f\_score","marker\_count","binary\_genes","V1","V2","prior\_prob\_off","prior\_prob\_on"))  
#hist(as.numeric(prior\_marker$prior\_prob\_on),main = "", xlab = "Prior probability of expression for the marker genes")  
  
# Obtain prior probability of expression for the NS-forest binary genes   
prior\_binary<-merge(geneList\_bi, pi\_df, by = "ENSEMBLID", all.x = TRUE)   
prior\_binary$prior\_prob\_off <- prior\_binary$V1 + prior\_binary$V2  
prior\_binary$prior\_prob\_on <- 1-(prior\_binary$V1 + prior\_binary$V2)  
  
# visualize prior probabilities for binary genes  
#hist(as.numeric(prior\_binary$prior\_prob\_on),main = "", xlab = "Prior probability of expression of the Binary genes")

Notes:

HLCA and cellRef are combined datasets. Will integration methods affect the proportion/rate (𝜆) of individual gene expression counts in total gene expression counts in a cell type?

## 5. Functions for calculating rate,three state probabilities

Functions from the 2022 Biostatistics paper

# Train reference data  
trainReference <- function(ref,pi.all=pi.all.g,mu=mu.g,a=a.all.g,sigma.all=sigma.all.g,  
 g.on.all=g.on.all.g,g.off.all=g.off.all.g,discrim\_only=FALSE) {  
 # Subset to genes present in both reference data and knowledge base  
 common <- intersect(names(ref),rownames(pi.all))  
 if (discrim\_only) {  
 common <- intersect(common,discrim.g)  
 }  
 N <- sum(ref)  
 ref <- ref[common]  
 pi.all2 <- pi.all[common,]  
 a.all2 <- a[common]  
 sigma.all2 <- sigma.all[common,]  
 g.on.all2 <- g.on.all[common]  
 g.off.all2 <- g.off.all[common]  
   
 # Compute mixing probability for each component  
 prob.exp <- pi.all2[,1]\*dnbinom(ref,1,1/((N/(a.all2))+1))  
 prob.ln1 <- sapply(common,function(j)   
 pi.all2[j,2]\*sads::dpoilog(ref[j],mu[1]+g.off.all2[j]+log(N),sigma.all2[j,1]))  
 prob.ln2 <- sapply(common,function(j)   
 (1-pi.all2[j,1]-pi.all2[j,2])\*sads::dpoilog(ref[j],mu[2]+g.on.all2[j]+log(N),sigma.all2[j,2]))   
 return(data.frame(rate=ref/N,exp=prob.exp/(prob.exp+prob.ln1+prob.ln2),ln1=prob.ln1/(prob.exp+prob.ln1+prob.ln2),ln2=prob.ln2/(prob.exp+prob.ln1+prob.ln2)))  
}  
  
  
# Train all reference  
trainAllReference <- function(data,labels,pi.all=pi.all.g,mu=mu.g,a=a.all.g,sigma.all=sigma.all.g,  
 g.on.all=g.on.all.g,g.off.all=g.off.all.g,discrim\_only=FALSE) {  
 cell.types <- unique(labels)  
 dataSums <- lapply(cell.types,function(c) rowSums(data[,labels==c]))  
  
 d.list <- bplapply(dataSums,trainReference,pi.all=pi.all,mu=mu,a=a,sigma.all=sigma.all,  
 g.on.all=g.on.all,g.off.all=g.off.all,discrim\_only=discrim\_only)  
 names(d.list) <- cell.types  
 return(d.list)  
}  
  
  
# getBarcode of on state  
getBarcode<-function(d.list) {  
 barcode <- sapply(d.list,function(x) 1-x[,2]-x[,3]) # ### Barcode is ln2=prob.ln2/(prob.exp+prob.ln1+prob.ln2) if greater than 0  
 barcode[barcode<0] <- 0  
 colnames(barcode) <- names(d.list)  
 rownames(barcode) <- rownames(d.list[[1]])  
 return(barcode)  
}  
  
# get probability of on and off state  
getBarcode2 <- function(d.list) {  
 barcode <- sapply(d.list,function(x) 1-x[,2])  
 barcode[barcode<0] <- 0  
 colnames(barcode) <- names(d.list)  
 rownames(barcode) <- rownames(d.list[[1]])  
 return(barcode)  
}

## 6. Functions for calculating cell type-specific expression probability for lung datasets

Three functions to generate expression probability for cell types

## for cellRef, meta\_feature is "celltype\_level3\_fullname"  
## for hlca, meta\_feature is "ann\_finest\_level"  
## Run individual cell type.  
generateCellType\_barcode <- function(seurat\_dataset,dataset\_name, cellType, cut\_off){  
 # subset by cell type  
 if(dataset\_name=="hlca") {  
 data\_subset <- subset(seurat\_dataset, subset = ann\_finest\_level == cellType)  
 } else if(dataset\_name=="cellRef"){  
 data\_subset <- subset(seurat\_dataset, subset = celltype\_level3\_fullname == cellType)  
 }  
 matrix<- as.matrix(data\_subset@assays$RNA@counts)  
  
 labels<-c(rep(cellType,dim(matrix)[2]))  
   
 # calculte rate and probability  
 data.d <- trainAllReference(matrix,labels,discrim\_only=FALSE)  
  
 # get barcode and gene list  
 barcodes <- getBarcode2(data.d)  
 geneList <-rownames(barcodes)[barcodes > cut\_off]  
 length(geneList)  
 write.table(geneList, file = paste0(cellType, cut\_off, ".tsv"), sep = "\n", row.names = FALSE, col.names = FALSE, quote = FALSE)  
 return(barcodes)  
   
}  
  
  
## Run multiple cell types  
runBatch<-function(seurat\_dataset,dataset\_name,cell\_type\_list,cut\_off\_list){  
 dic ={}  
 for(cellType in cell\_type\_list){  
   
 for (cut\_off in cut\_off\_list) {  
   
 code<-generateCellType\_barcode(seurat\_dataset, dataset\_name,cellType, cut\_off)  
 print(head(code))  
 dic[[cellType]]<-as.data.frame(code)   
 }  
 }  
 return(dic)  
}  
  
  
## Check cell type-specific gene expression probability (ExpProb)  
# output the expression probability for a gene in a specific cell type  
checkGene <- function(id,sym, barcode\_dic, marker, data\_name){  
 df\_cell\_type\_gene<-data.frame(matrix(vector(), 0, 6,  
 dimnames=list(c(), c("id", "sym", "barcode","marker", "cellType", "data"))),  
 stringsAsFactors=F)  
 for(i in 1:length(barcode\_dic)){  
 barcode<-barcode\_dic[[i]][id,]  
 df = data.frame(id=id,sym=sym, barcode=barcode, marker=marker,cellType=names(barcode\_dic)[i], data=data\_name)  
 df\_cell\_type\_gene <- rbind(df\_cell\_type\_gene, df)  
 }  
 return(df\_cell\_type\_gene)  
}

## 7. Validate the expression probabilities using know expressed genes

### 7.1. Check the “on” state probability of NS-forest markers and calculate the confusion matrix.

The following confusion matrix are based on assumption that marker genes expressed and only expressed in their targeted cell type. Since many markers expressed in more than their targeted cell type. FP counts were not correct. Therefore, recall is trustworthy, the precision is not correct.

#### 7.1.1 Probabilty cut off 0.75

on\_hlca\_post <- read.table("marker\_hlca\_prior\_post.tsv", sep = "\t", header = TRUE, stringsAsFactors = FALSE)  
  
on\_hlca\_post\_75<-on\_hlca\_post %>% mutate(expected\_value= ifelse(targeted=="TRUE", 1,0 ),  
 predicted\_value= ifelse(expProb >0.75, 1,0 ))  
on\_hlca\_post\_clean <- on\_hlca\_post\_75 %>% drop\_na()  
common\_levels <- union(levels(factor(on\_hlca\_post\_clean$predicted\_value)),   
 levels(factor(on\_hlca\_post\_clean$expected\_value)))  
  
on\_hlca\_post\_clean$predicted\_value <- factor(on\_hlca\_post\_clean$predicted\_value, levels = common\_levels)  
on\_hlca\_post\_clean$expected\_value <- factor(on\_hlca\_post\_clean$expected\_value, levels = common\_levels)  
  
#Creating confusion matrix  
on\_matrix <- confusionMatrix(data=on\_hlca\_post\_clean$predicted\_value, reference = on\_hlca\_post\_clean$expected\_value,positive = "1")  
  
on\_matrix

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 6765 33  
## 1 1215 100  
##   
## Accuracy : 0.8462   
## 95% CI : (0.8381, 0.854)  
## No Information Rate : 0.9836   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.1117   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.75188   
## Specificity : 0.84774   
## Pos Pred Value : 0.07605   
## Neg Pred Value : 0.99515   
## Prevalence : 0.01639   
## Detection Rate : 0.01233   
## Detection Prevalence : 0.16209   
## Balanced Accuracy : 0.79981   
##   
## 'Positive' Class : 1   
##

#### 7.1.2 Probabilty cut off 0.90

on\_hlca\_post\_90<-on\_hlca\_post %>% mutate(expected\_value= ifelse(targeted=="TRUE", 1,0 ),  
 predicted\_value= ifelse(expProb >0.9, 1,0 ))  
on\_hlca\_post\_clean <- on\_hlca\_post\_90 %>% drop\_na()  
  
on\_hlca\_post\_clean$predicted\_value <- factor(on\_hlca\_post\_clean$predicted\_value, levels = common\_levels)  
on\_hlca\_post\_clean$expected\_value <- factor(on\_hlca\_post\_clean$expected\_value, levels = common\_levels)  
  
#Creating confusion matrix  
on\_matrix <- confusionMatrix(data=on\_hlca\_post\_clean$predicted\_value, reference = on\_hlca\_post\_clean$expected\_value,positive = "1")  
on\_matrix

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 6980 38  
## 1 1000 95  
##   
## Accuracy : 0.8721   
## 95% CI : (0.8646, 0.8793)  
## No Information Rate : 0.9836   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.1293   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.71429   
## Specificity : 0.87469   
## Pos Pred Value : 0.08676   
## Neg Pred Value : 0.99459   
## Prevalence : 0.01639   
## Detection Rate : 0.01171   
## Detection Prevalence : 0.13497   
## Balanced Accuracy : 0.79449   
##   
## 'Positive' Class : 1   
##

#### 7.1.3 Probabilty cut off 0.95

on\_hlca\_post\_95<-on\_hlca\_post %>% mutate(expected\_value= ifelse(targeted=="TRUE", 1,0 ),  
 predicted\_value= ifelse(expProb >0.95, 1,0 ))  
on\_hlca\_post\_clean <- on\_hlca\_post\_95 %>% drop\_na()  
  
on\_hlca\_post\_clean$predicted\_value <- factor(on\_hlca\_post\_clean$predicted\_value, levels = common\_levels)  
on\_hlca\_post\_clean$expected\_value <- factor(on\_hlca\_post\_clean$expected\_value, levels = common\_levels)  
  
#Creating confusion matrix  
on\_matrix <- confusionMatrix(data=on\_hlca\_post\_clean$predicted\_value, reference = on\_hlca\_post\_clean$expected\_value,positive = "1")  
on\_matrix

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 7168 45  
## 1 812 88  
##   
## Accuracy : 0.8944   
## 95% CI : (0.8875, 0.901)  
## No Information Rate : 0.9836   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.146   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.66165   
## Specificity : 0.89825   
## Pos Pred Value : 0.09778   
## Neg Pred Value : 0.99376   
## Prevalence : 0.01639   
## Detection Rate : 0.01085   
## Detection Prevalence : 0.11093   
## Balanced Accuracy : 0.77995   
##   
## 'Positive' Class : 1   
##

### 7.3 Estimate expected positive count based on observation of NS-forest markers

The above confusion matrices are based on assumption that marker genes expressed and only expressed in their targeted cell type. However, the marker gene combinations given by NS-Forest often have more than one marker gene, which implies that marker genes are expressed in more than one cell type. Therefore, we expect that multiple cell types express marker genes. I estimated the expected positive count based on observation of NS-forest markers.

If x is the number of cell types in which a marker gene is expressed on average, then two-marker gene expressed in one of 61 cell types follows the probability formula:(x/61) \* (x/61)

In the lung HLCA dataset, two-marker gene combinations are needed to classify each of 61 cell types, therfore (x/61) \* (x/61) = 1, and x = sqrt(61)=7.81.This means, on average, a marker gene is expressed in 7.81 cell types.

Given 133 marker genes, the expected positive\_count\_expected = 7.81 \* 133 =1038.73

### 7.4 Recall and adjust precision for three cut off

#### 0.75 cut-off:

recall:0.75

From the confusion matrix, the predicted positive\_count\_predicted = 1215 + 100=1315

adjusted Positive Predictive Value (Precision) = 1038.73/(1215 + 100)\*0.75 =0.59

#### 0.90 cut-off:

recall:0.71

From the confusion matrix, the predicted positive\_count\_predicted = 1000 + 95=1095

adjusted Positive Predictive Value (Precision) = 1038.73/(1000 + 95)\*0.71 =0.67

#### 0.95 cut-off:

recall:0.66

From the confusion matrix, the predicted positive\_count\_predicted = 812+88=1315

adjusted Positive Predictive Value (Precision) = 1038.73/(812+88)\*0.66 =0.76

### 7.2. Check the “on and off” state probability of NS-forest markers and calculate the confusion matrix. Compare the results with step 1.

## cell type name  
#unique(cellRef$celltype\_level3\_fullname)  
#unique(hlca$ann\_finest\_level)  
  
## run all cell-type in HLCA  
seurat\_dataset<-hlca  
dataset\_name <-"hlca"  
cell\_type\_list <- unlist(unique(hlca$ann\_finest\_level))  
cut\_off\_list <- c(0.75)  
  
#barcode\_dic2<-runBatch(seurat\_dataset,dataset\_name,cell\_type\_list,cut\_off\_list)  
  
#saveRDS(barcode\_dic2, file = "hlca\_barcode\_all\_low.rds")  
barcode\_dic2<-load(hlca\_barcode\_all\_low.rds)  
  
# maker prior and post probability  
prior\_marker <- prior\_marker[!is.na(prior\_marker$ENSEMBLID), ]  
data\_name<-rep("hlca", nrow(prior\_marker))  
  
df\_hlca\_gene<-checkGene(prior\_marker$ENSEMBLID,prior\_marker$NSForest\_markers, barcode\_dic=barcode\_dic2, prior\_marker$clusterName, data\_name)  
  
df\_hlca\_post<-merge(df\_hlca\_gene,prior\_marker, by.x = "id", by.y = "ENSEMBLID", all.x = TRUE)  
df\_hlca\_post<-df\_hlca\_post %>% mutate(target=ifelse(df\_hlca\_post$`clusterName` == df\_hlca\_post$`cellType`, TRUE, FALSE))  
write.table(df\_hlca\_post, 'hlca\_prior\_post\_ExpProb\_low.tsv', sep = "\t",row.names = FALSE,col.names = TRUE)

#### 7.2.1 probability cut off 0.75

on\_off\_post <- read.table('hlca\_prior\_post\_ExpProb\_low.tsv', sep = "\t", header = TRUE, stringsAsFactors = FALSE)  
  
on\_off\_post\_75<-on\_off\_post %>% mutate(expected\_value= ifelse(target=="TRUE", 1,0 ),  
 predicted\_value= ifelse(barcode >0.75, 1,0 ))  
on\_off\_post\_clean <- on\_off\_post\_75 %>% drop\_na()  
common\_levels <- union(levels(factor(on\_off\_post\_clean$predicted\_value)),   
 levels(factor(on\_off\_post\_clean$expected\_value)))  
  
on\_off\_post\_clean$predicted\_value <- factor(on\_off\_post\_clean$predicted\_value, levels = common\_levels)  
on\_off\_post\_clean$expected\_value <- factor(on\_off\_post\_clean$expected\_value, levels = common\_levels)  
  
#Creating confusion matrix  
on\_off\_matrix <- confusionMatrix(data=on\_off\_post\_clean$predicted\_value, reference = on\_off\_post\_clean$expected\_value,positive = "1")  
  
on\_off\_matrix

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 3235 0  
## 1 4745 133  
##   
## Accuracy : 0.4151   
## 95% CI : (0.4044, 0.4259)  
## No Information Rate : 0.9836   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.0219   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 1.00000   
## Specificity : 0.40539   
## Pos Pred Value : 0.02727   
## Neg Pred Value : 1.00000   
## Prevalence : 0.01639   
## Detection Rate : 0.01639   
## Detection Prevalence : 0.60126   
## Balanced Accuracy : 0.70269   
##   
## 'Positive' Class : 1   
##

#### 7.2.2 probability cut off 0.9

on\_off\_post\_90<-on\_off\_post %>% mutate(expected\_value= ifelse(target=="TRUE", 1,0 ),  
 predicted\_value= ifelse(barcode >0.9, 1,0 ))  
on\_off\_post\_clean <- on\_off\_post\_90 %>% drop\_na()  
  
on\_off\_post\_clean$predicted\_value <- factor(on\_off\_post\_clean$predicted\_value, levels = common\_levels)  
on\_off\_post\_clean$expected\_value <- factor(on\_off\_post\_clean$expected\_value, levels = common\_levels)  
  
#Creating confusion matrix  
on\_off\_matrix <- confusionMatrix(data=on\_off\_post\_clean$predicted\_value, reference = on\_off\_post\_clean$expected\_value,positive = "1")  
on\_off\_matrix

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 3567 0  
## 1 4413 133  
##   
## Accuracy : 0.4561   
## 95% CI : (0.4452, 0.467)  
## No Information Rate : 0.9836   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.0258   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 1.00000   
## Specificity : 0.44699   
## Pos Pred Value : 0.02926   
## Neg Pred Value : 1.00000   
## Prevalence : 0.01639   
## Detection Rate : 0.01639   
## Detection Prevalence : 0.56034   
## Balanced Accuracy : 0.72350   
##   
## 'Positive' Class : 1   
##

#### 7.2.3 probability cut off 0.95

on\_off\_post\_95<-on\_off\_post %>% mutate(expected\_value= ifelse(target=="TRUE", 1,0 ),  
 predicted\_value= ifelse(barcode >0.95, 1,0 ))  
on\_off\_post\_clean <- on\_off\_post\_95 %>% drop\_na()  
  
on\_off\_post\_clean$predicted\_value <- factor(on\_off\_post\_clean$predicted\_value, levels = common\_levels)  
on\_off\_post\_clean$expected\_value <- factor(on\_off\_post\_clean$expected\_value, levels = common\_levels)  
  
#Creating confusion matrix  
on\_off\_matrix <- confusionMatrix(data=on\_off\_post\_clean$predicted\_value, reference = on\_off\_post\_clean$expected\_value,positive = "1")  
on\_off\_matrix

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 3747 0  
## 1 4233 133  
##   
## Accuracy : 0.4782   
## 95% CI : (0.4673, 0.4892)  
## No Information Rate : 0.9836   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.0282   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 1.00000   
## Specificity : 0.46955   
## Pos Pred Value : 0.03046   
## Neg Pred Value : 1.00000   
## Prevalence : 0.01639   
## Detection Rate : 0.01639   
## Detection Prevalence : 0.53815   
## Balanced Accuracy : 0.73477   
##   
## 'Positive' Class : 1   
##

### 7.4 Recall and adjust precision for three cut off

#### 0.75 cut-off:

recall:1

adjusted Positive Predictive Value (Precision) = 1038.73/(4745+133)\*1 =0.21

#### 0.90 cut-off:

recall:1

adjusted Positive Predictive Value (Precision) = 1038.73/(4413+133)\*1 =0.23

#### 0.95 cut-off:

recall:1

adjusted Positive Predictive Value (Precision) = 1038.73/(4233+133)\*1 =0.23

### 7.5 Observation and results

Although recall is 100% for NS-forest markers, the number of FP is too high when using combined probability of both on and off states as expression probability.

### 7.3. Merge the tables from steps 1 and 2. Manually check 10 genes that have low expression probability in the “on” state.

# modify column name prepare for merge  
colnames(on\_hlca\_post\_clean)  
on\_off\_post\_narrow<-on\_off\_post\_clean %>% mutate(ENSEMBLId=id,   
 TargetCellType=marker,  
 dataCellSet=cellType,  
 on\_off\_expProb=barcode) %>%  
 select(c("ENSEMBLId","TargetCellType","dataCellSet","on\_off\_expProb"))  
on\_hlca\_post\_narrow<-on\_hlca\_post\_clean %>% select(c("ENSEMBLId","TargetCellType","dataCellSet","expProb"))   
  
  
  
# merge two dataframe  
on\_hlca\_post\_clean$ENSEMBLId\_TargetCellType\_dataCellSet= paste(on\_hlca\_post\_clean$ENSEMBLId,on\_hlca\_post\_clean$TargetCellType,on\_hlca\_post\_clean$dataCellSet)  
prior\_post\_on\_off<-merge(on\_hlca\_post\_clean, on\_off\_post\_narrow, by=c("ENSEMBLId","TargetCellType","dataCellSet"), x.all=TRUE)  
  
# remove duplication  
prior\_post\_on\_off <- prior\_post\_on\_off[!duplicated(prior\_post\_on\_off), ]  
  
  
write.table(prior\_post\_on\_off, 'hlca\_prior\_post\_on\_off.tsv', sep = "\t",row.names = FALSE,col.names = TRUE)   
  
  
# manually check 10 genes with low ExpProb

### 7.4 Artificially set π-null and π-off to 0.33 if the “on and off” state probability remains lower than 75% for NS-Forest markers. Check if the pre-trained data have wrong prior probability.

All NS-Forest markers have “on and off” state probability equal to 1. This step is not necessary.

## 8 Conclusion

The expProb model defines the expression of ~25% of NS-forest marker genes in the “off” state, likely because their expression levels are low. However, using a combined “on and off” state to identify expressed genes will result in a high false positive count and low precision.

Plans for next steps:

1.Make distribution plots for the markers with wrong expProb. 2.Check all prior and post parameter measure for the markers with wrong expProb, especially lamda. 3.Understand the training code and see if there is a room to modification. 4.Re-visit supplemental methods. 5.Set the target recall and precision.