**Original Code for Figure 5H, I**

a <- read\_excel("C:/Users/ryanpm/Desktop/lilycohorttotalv2.xlsx")

#colorOne = '#FFA500'

#colorTwo = '#4B9CD3'

#colorThree = '#595750'

#colorFour = '#FF7F50'

colorFive = '#4682b4'

colorSix = '#FF0000'

#valueOne = "SARS-2"

#valueTwo = "GFP"

#valueThree = "PBS"

#valueFour = "Breakthrough"

valueFive = "baseline"

valueSix = "post-boost"

lily <- read\_excel("C:/Users/ryanpm/Desktop/lilycohorttotalv2.xlsx")

#define data set

data <- lily[c(1:48,65:112),3:95]

Variants <- lily[c(1:48,65:112),c(1,3:95)]

#which(lily$VariantID=="Omicron"|Luminex$VariantID=="Delta")

# data<-delta\_0\_22

#column where Data starts

colStart <-1

colProtect <-95

lassoTresh = 0.4

# Make new matrix of Antibody data

#X <- as.matrix(as.data.frame(lapply(data[, colStart:(ncol(data))], as.numeric)))

X <- as.matrix(data)

#Log transforming the X matrix

#X<-log10(X)

#X[X == -Inf] <- 0

#X[X == 'is.NA'] <- 0

#X[X == 'is.NaN'] <- 0

#Z-score the log10 transformed data

X <- scale(X, center = TRUE, scale = TRUE)

#Set up factor based on the protection data

y <- factor(data[,colProtect], levels = unique(data[,colProtect]))

y <- factor(Variants$Timepoint)

#The options for lasso list

#n\_trials - number time you do lasso selection

#threshold = number of lasso selects that a variable needs to be selected

#

opts\_sel <- list(n\_trials = 5, threshold = lassoTresh, return\_count = FALSE)

#Run Lasso with the options we selected and repeated the number of times we choose

#sel\_features <- select\_lasso(X, y)

sel\_features <- select\_repeat(X, y, selector = select\_lasso, options = opts\_sel)

print(sel\_features)

X\_sel<-X[,sel\_features]

# input for opts\_plot

my\_colors = list(Group = c( "baseline" = colorFive, "post-boost" = colorSix))

# input for opts\_plot - needed for visualize scores

df\_features <- data.frame(name = colnames(X))

df\_features$label <- df\_features$name

opts\_plot <- list(df\_features = df\_features,

loading\_alpha = 0.5,# transparency for the loadings

score\_alpha = 0.5,# transparency for the scores

LV\_ind = c(1,2), # which LVs to plot

colors = my\_colors,

size =4,

y\_name = "Group")

#Run PLS-DA model based on the lasso selected features above

# model\_acc<-plsda\_accuracy(X, y, sel\_features)

#n\_LV = the number of latent variables - 2 coordinates (LV1, LV2)

#change X\_sel to X to run non-lasso data

model <- train\_ropls(X\_sel, y, options = list(n\_LV = 2))

# model\_acc<-plsda\_accuracy(data,colStart, colProtect, lassoTresh, sel\_features)

# model<-model\_acc[[1]]

# X\_sel <- X[,sel\_features]

#Makes the score plot - Separation plot

plt\_scores <- visualize\_ropls\_scores(model, y, options = opts\_plot)

print(plt\_scores)

#Makes the loadings plot - LV1

opts\_plot$X <- X

opts\_plot$y <- y

opts\_plot$LV\_ind <- 1 #Which LV are you plotting

opts\_plot$mark\_enrichment <- unname(model@vipVn) #Color coding the bars based on which group has a higher median of that feature

plt\_loadings\_bar <- visualize\_ropls\_loadings\_bar(model, options = opts\_plot)

print(plt\_loadings\_bar)

#Makes the loadings plot - For LV2

opts\_plot$X <- X

opts\_plot$y <- y

opts\_plot$LV\_ind <- 2 #Which LV are you plotting

opts\_plot$mark\_enrichment <- T #Color coding the bars based on which group has a higher median of that feature

plt\_loadings\_bar <- visualize\_ropls\_loadings\_bar(model, options = opts\_plot)

print(plt\_loadings\_bar)

#VIP Plot

#vipdataframe<-data.frame(feature = V, score = model@vipVn)

vipdataframe<-data.frame(feature = names(model@vipVn), score = unname(model@vipVn))

vipdataframe<-vipdataframe[order(vipdataframe$score),]

vipdataframe$mark <- NA

for (ind\_feat in 1:nrow(vipdataframe)) {

tmp\_mean <- rep(NA, length = nlevels(y))

for (ind\_class in 1:nlevels(y)) {

tmp\_mean[ind\_class] <- mean(X[which(y == levels(y)[ind\_class]),

which(colnames(X) == vipdataframe$feature[ind\_feat])])

}

vipdataframe$mark[ind\_feat] <- levels(y)[which.max(tmp\_mean)]

}

vipdataframe$mark <- factor(vipdataframe$mark, levels = levels(y))

plt\_vip\_bar<-ggplot(vipdataframe, aes(x=score, y=feature, fill=mark)) +

xlab("")+

ylab("")+

ggtitle("VIP scores")+

geom\_bar(stat = "identity", width=0.3)+

scale\_fill\_manual(breaks = c(valueFive, valueSix), values = c(colorFive, colorSix))+

theme\_bw() +

ggplot2::theme(plot.title=element\_text(size=28, hjust=0.5),

axis.text.x=element\_text(size=12),

axis.text.y=element\_text(size=12),

axis.title.x=element\_text(size=12),

axis.title.y=element\_text(size=12),

legend.position = "none")

print(plt\_vip\_bar)

**Output files**

**PLS-DA Lasso - Figure 5H**

Chart, scatter chart

Description automatically generated

**Latent Variable 1 Scores**

Chart, bar chart

Description automatically generated

**Latent Variable 2 Scores**

Chart, bar chart

Description automatically generated

**VIP Scores – Figure 5I**

Chart, bar chart

Description automatically generated

**Validation of Model**

Diagram

Description automatically generated

**Original Code for Figure 5J, K**

lily <- read\_excel("C:/Users/ryanpm/Desktop/lilycohortfcrv2.xlsx")

#define data set

data <- lily[c(1:48,65:112),3:34]

Variants <- lily[c(1:48,65:112),c(1,3:34)]

#which(lily$VariantID=="Omicron"|Luminex$VariantID=="Delta")

# data<-delta\_0\_22

#column where Data starts

colStart <-1

colProtect <-34

lassoTresh = 0.2

# Make new matrix of Antibody data

#X <- as.matrix(as.data.frame(lapply(data[, colStart:(ncol(data))], as.numeric)))

X <- as.matrix(data)

#Log transforming the X matrix

#X<-log10(X)

#X[X == -Inf] <- 0

#X[X == 'is.NA'] <- 0

#X[X == 'is.NaN'] <- 0

#Z-score the log10 transformed data

X <- scale(X, center = TRUE, scale = TRUE)

#Set up factor based on the protection data

y <- factor(data[,colProtect], levels = unique(data[,colProtect]))

y <- factor(Variants$Timepoint)

#The options for lasso list

#n\_trials - number time you do lasso selection

#threshold = number of lasso selects that a variable needs to be selected

#

opts\_sel <- list(n\_trials = 5, threshold = lassoTresh, return\_count = FALSE)

#Run Lasso with the options we selected and repeated the number of times we choose

#sel\_features <- select\_lasso(X, y)

sel\_features <- select\_repeat(X, y, selector = select\_lasso, options = opts\_sel)

print(sel\_features)

X\_sel<-X[,sel\_features]

# input for opts\_plot

my\_colors = list(Group = c( "baseline" = colorFive, "post-boost" = colorSix))

# input for opts\_plot - needed for visualize scores

df\_features <- data.frame(name = colnames(X))

df\_features$label <- df\_features$name

opts\_plot <- list(df\_features = df\_features,

loading\_alpha = 0.5,# transparency for the loadings

score\_alpha = 0.5,# transparency for the scores

LV\_ind = c(1,2), # which LVs to plot

colors = my\_colors,

size =4,

y\_name = "Group")

#Run PLS-DA model based on the lasso selected features above

# model\_acc<-plsda\_accuracy(X, y, sel\_features)

#n\_LV = the number of latent variables - 2 coordinates (LV1, LV2)

#change X\_sel to X to run non-lasso data

model <- train\_ropls(X\_sel, y, options = list(n\_LV = 2))

# model\_acc<-plsda\_accuracy(data,colStart, colProtect, lassoTresh, sel\_features)

# model<-model\_acc[[1]]

# X\_sel <- X[,sel\_features]

#Makes the score plot - Separation plot

plt\_scores <- visualize\_ropls\_scores(model, y, options = opts\_plot)

print(plt\_scores)

#Makes the loadings plot - LV1

opts\_plot$X <- X

opts\_plot$y <- y

opts\_plot$LV\_ind <- 1 #Which LV are you plotting

opts\_plot$mark\_enrichment <- unname(model@vipVn) #Color coding the bars based on which group has a higher median of that feature

plt\_loadings\_bar <- visualize\_ropls\_loadings\_bar(model, options = opts\_plot)

print(plt\_loadings\_bar)

#Makes the loadings plot - For LV2

opts\_plot$X <- X

opts\_plot$y <- y

opts\_plot$LV\_ind <- 2 #Which LV are you plotting

opts\_plot$mark\_enrichment <- T #Color coding the bars based on which group has a higher median of that feature

plt\_loadings\_bar <- visualize\_ropls\_loadings\_bar(model, options = opts\_plot)

print(plt\_loadings\_bar)

#VIP Plot

#vipdataframe<-data.frame(feature = V, score = model@vipVn)

vipdataframe<-data.frame(feature = names(model@vipVn), score = unname(model@vipVn))

vipdataframe<-vipdataframe[order(vipdataframe$score),]

vipdataframe$mark <- NA

for (ind\_feat in 1:nrow(vipdataframe)) {

tmp\_mean <- rep(NA, length = nlevels(y))

for (ind\_class in 1:nlevels(y)) {

tmp\_mean[ind\_class] <- mean(X[which(y == levels(y)[ind\_class]),

which(colnames(X) == vipdataframe$feature[ind\_feat])])

}

vipdataframe$mark[ind\_feat] <- levels(y)[which.max(tmp\_mean)]

}

vipdataframe$mark <- factor(vipdataframe$mark, levels = levels(y))

plt\_vip\_bar<-ggplot(vipdataframe, aes(x=score, y=feature, fill=mark)) +

xlab("")+

ylab("")+

ggtitle("VIP scores")+

geom\_bar(stat = "identity", width=0.3)+

scale\_fill\_manual(breaks = c(valueFive, valueSix), values = c(colorFive, colorSix))+

theme\_bw() +

ggplot2::theme(plot.title=element\_text(size=28, hjust=0.5),

axis.text.x=element\_text(size=12),

axis.text.y=element\_text(size=12),

axis.title.x=element\_text(size=12),

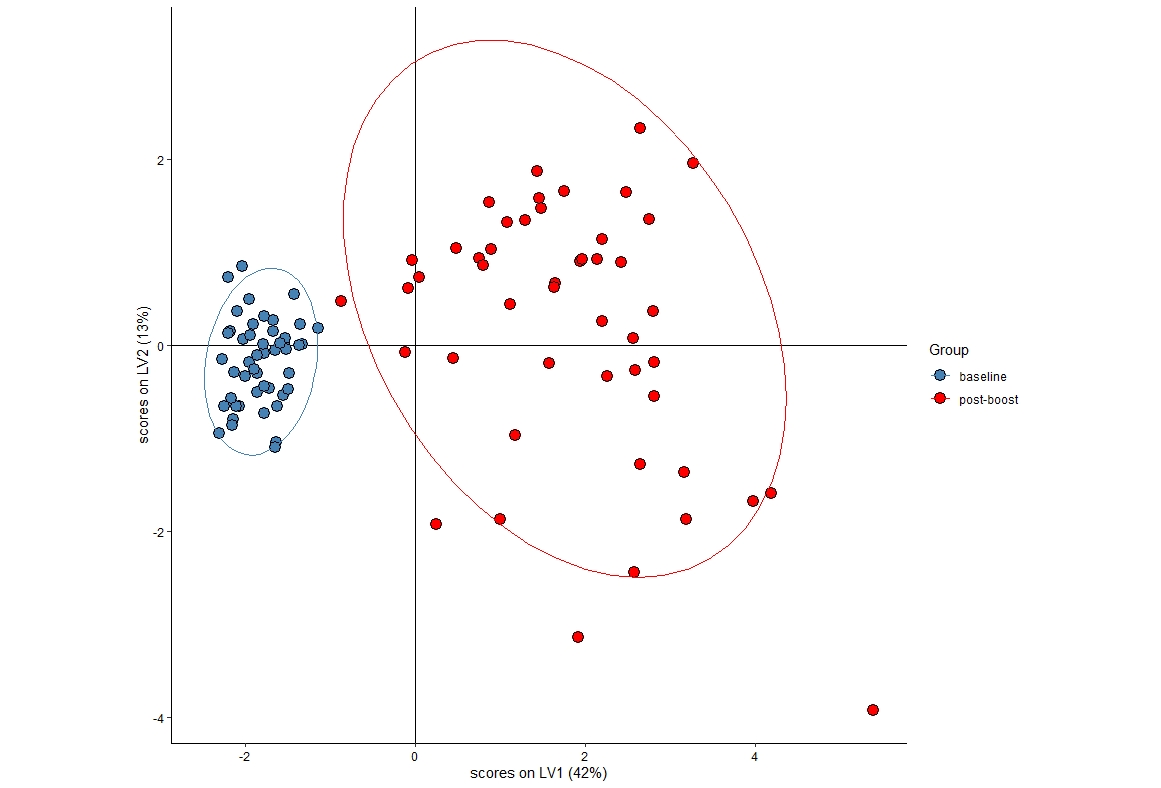
axis.title.y=element\_text(size=12),

legend.position = "none")

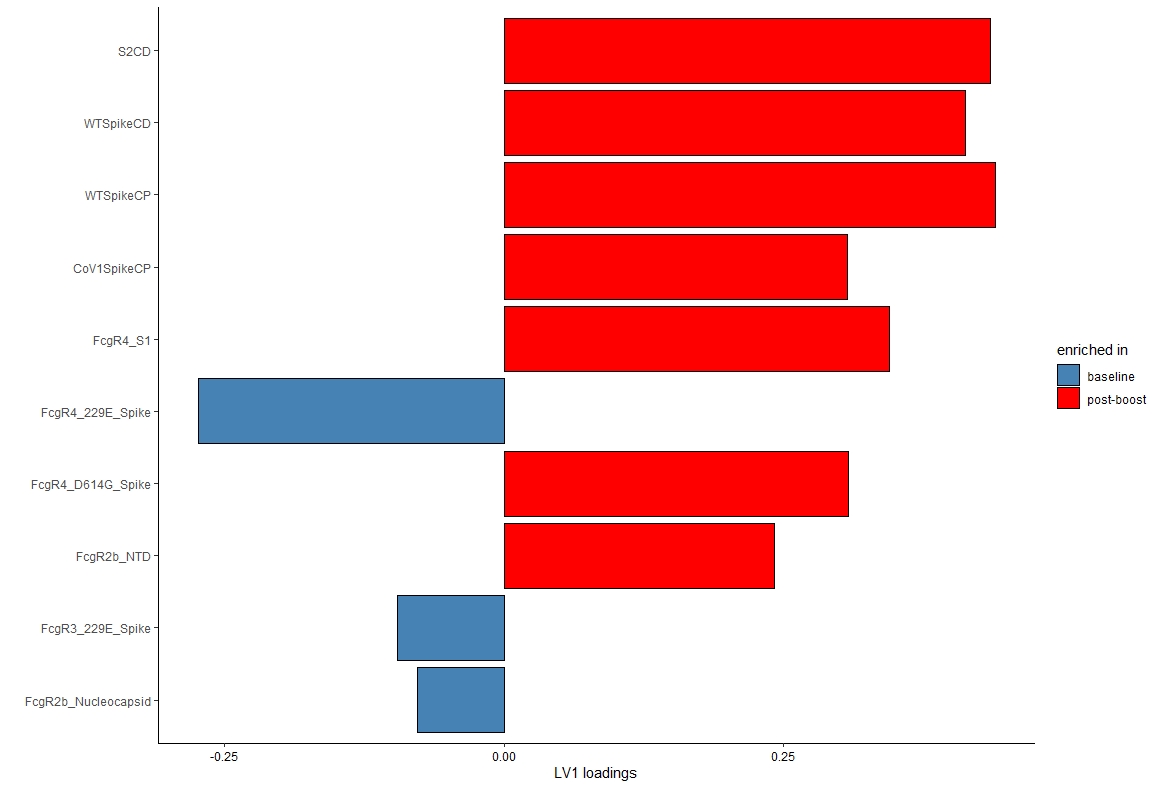
print(plt\_vip\_bar)

**Output Files**

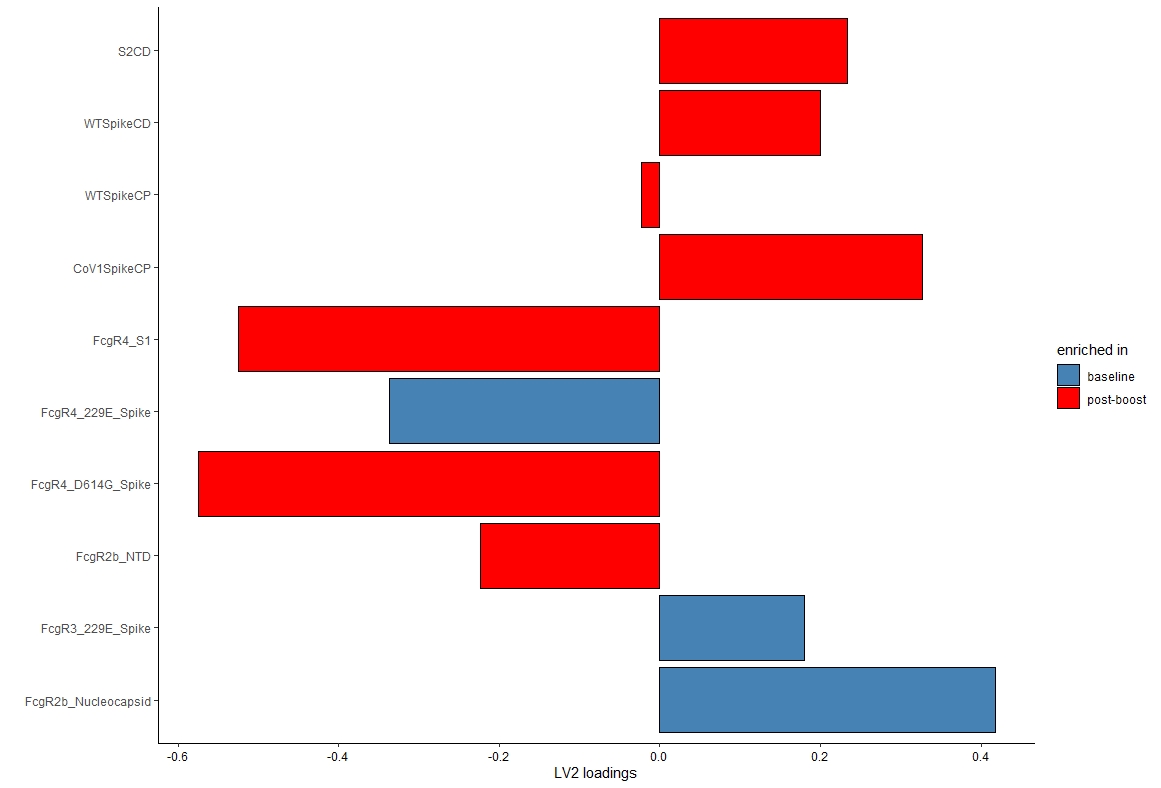
**PLS-DA – Figure 5J**



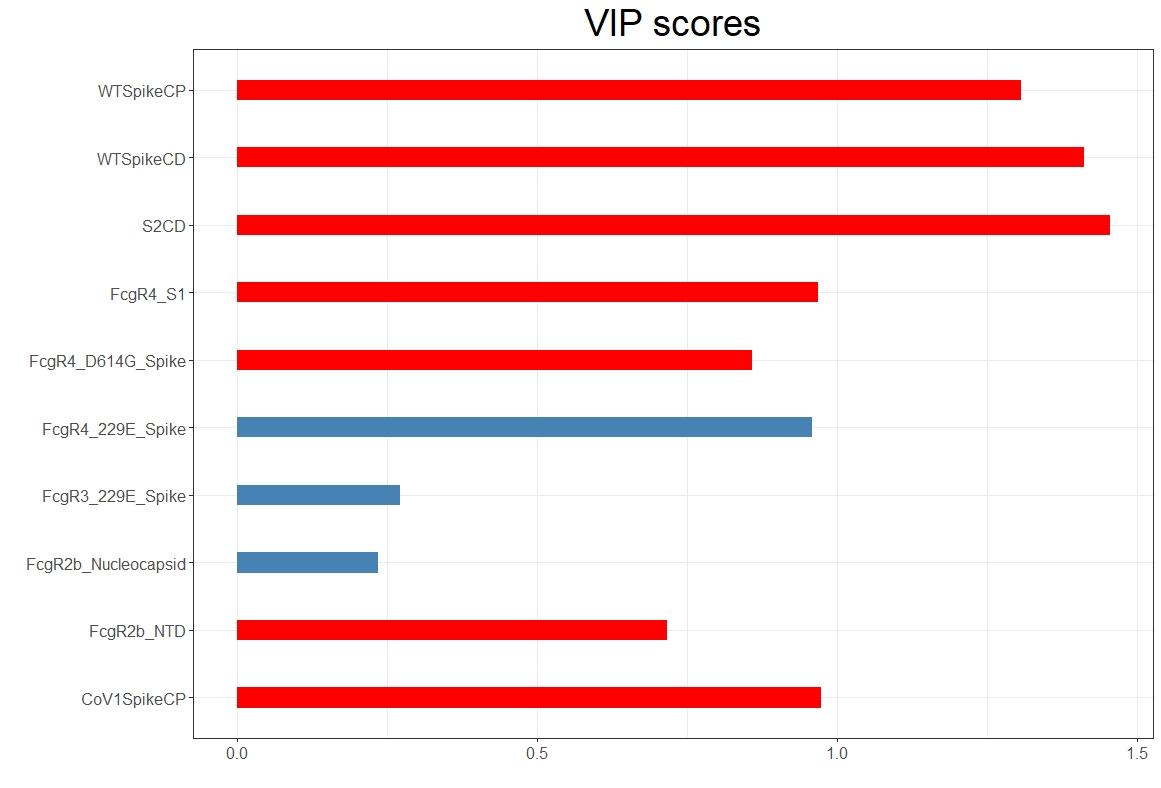
**Latent Variable 1 Scores**



**Latent Variable 2 Scores**



**VIP Scores – Figure 5K**



**Validation of Model**

Chart

Description automatically generated with medium confidence

**Model Validation Code**

set.seed(123)

opts\_sel <- list(threshold = 0.3, n\_trials = 10, return\_count = FALSE)

sel\_features <- select\_repeat(X, y, selector = select\_lasso, options = opts\_sel)

select <- function(X, y) { return(select\_repeat(X, y, selector = select\_lasso, options = opts\_sel)) }

method = list(select = select,

train = train\_ropls,

predict = predict\_ropls,

score = score\_accuracy)

#opts\_val <- list(n\_folds = 10, rf\_trials = 0, pt\_trials = 0)

opts\_val <- list(n\_folds = 5, rf\_trials = 50, pt\_trials = 50)

vals <- validate\_repeat(X, y, method, opts\_val, n\_trials = 10)

#save(vals, file = "//Users/harrybertera/Desktop/Work/Post Doc Projects/Paulina/Glycans/COVID Glycans/R files/AccuracyFile.Rdata")

cv.acc <- 0

for (i in 1:100){

cv.acc <- cv.acc + vals[[i]]$cv\_score

}

cv.acc <- cv.acc/100 ## 57.14286% cv accuracy 2022.2.4 - for V2 Model w/o tech reps

cv.acc

# Crossvalidation Code

#n\_trials, threshold - Lasso inputs, make sure they are the same as when you ran LASSO on actual data

opts\_sel <- list(n\_trials = 10, threshold = 0.3, return\_count = FALSE)

select <- function(X, y) { return(select\_repeat(X, y, selector = select\_lasso, options = opts\_sel)) }

method = list(select=select,

train = train\_ropls,

predict = predict\_ropls,

score = score\_accuracy)

#n\_folds - Number of groups you split the data into (5, 4 groups are used to build and and 1 group is used to test the model)

#rf\_trials - Random Ab feature test instead of Lasso selected features - repeating that 100 times

#pt\_trials - Permutation testing - permute protection data after LASSO selection - repeat 100 times

#opts = list(n\_folds = 5, rf\_trials = 1, pt\_trials = 1)

#opts = list(n\_folds = 5, rf\_trials = 100, pt\_trials = 100)

opts = list(n\_folds = 5, rf\_trials = 50, pt\_trials = 50)

#The actual call to run the validation validate\_repeat

#return\_cv <- validate\_repeat(X, y, method, opts, n\_trials = 1)

#return\_cv <- validate\_repeat(X, y, method, opts, n\_trials = 10)

return\_cv <- validate\_repeat(X, y, method, opts, n\_trials = 10)

visualize\_validate(return\_cv)

# cv\_all=unlist(return\_cv)

# print(jj)

# print(mean(cv\_all))

# Stats Testing - To double Check Model Sigificance

model.acc <- c()

rf <- c()

pl <- c()

for (i in 1:100){

model.acc <- c(model.acc, return\_cv[[i]][1]$cv\_score)

rf <- c(rf, return\_cv[[i]][2]$rf\_scores)

pl <- c(pl, return\_cv[[i]][3]$pt\_scores)

}

#Create a dataframe with all the accuracy values from the model, the model with random features (rf)

#and from permutated labels (pl) and save it as a csv file

model.validation <- data.frame(Model = model.acc, random.features = rf, permuted.labels = pl)

write.csv(model.validation, file = "/Users/harrybertera/Desktop/Work/Post Doc Projects/Paulina/Glycans/COVID Glycans/R files/ValidationCSV.csv", row.names = FALSE)

################

#Make file cord plot input

# calculate correlation

res1 <- rcorr(X, type="spearman")

rvalue<-res1$r

pvalue<-res1$P

# chord

mat<-rvalue

df = data.frame(from = rep(rownames(mat), times = ncol(mat)),

to = rep(colnames(mat), each = nrow(mat)),

value = as.vector(mat),

stringsAsFactors = FALSE)

matp<-pvalue

dfp = data.frame(from = rep(rownames(matp), times = ncol(matp)),

to = rep(colnames(matp), each = nrow(matp)),

value = as.vector(matp),

stringsAsFactors = FALSE)

alllist<-unique(df$to)

length.all<-length(alllist)

indexall<-(length.all\*(length.all-1))/2

indexarray <- c()

for (aa in 1:length.all){

currentarray<-(length.all\*(aa-1)+1):(length.all\*aa)

if (aa>1){

currentminus<-1:(aa-1)

currentarraynew<-currentarray[aa:length(currentarray)]

}else{

currentarraynew<-currentarray

}

indexarray<-append(indexarray,currentarraynew)

}

df<-df[indexarray,]

dfp<-dfp[indexarray,]

whichsame<-which(df$from==df$to)

df<-df[-whichsame,]

dfp<-dfp[-whichsame,]

dfp$value<-p.adjust(dfp$value, method = "BH", n=dim(dfp)[1])

lasso\_row<-which((df$from %in% sel\_features)|(df$to %in% sel\_features))

df<-df[lasso\_row,]

dfp<-dfp[lasso\_row,]

df$p <- NA

df$p <-dfp$value

#Select More significant Nodes

selectSigNodes <-which((abs(df$value) > 0.5)&(df$p < 0.05))

dfSig<-df[selectSigNodes,]

**Univariate Confirmation**

