Weesnaw: Diagnosing Autism

Contents

1.0 Report Background

This report was prepared by your company name. Consultants:

- Sebastian Castillo-Sanchez
- Madison Chamberlain
- Ramin Chowdhury
- Tenzin Tashi

```
# Set to True to Show R code.  Set to False to supress R codes
show<-TRUE
```

2.0 Introduction

Our firm "Weesnaw" was hired as consultants analyze the results of Dr. Hanh, and develop a computational model that analyzes and generates the prediction of Autism Spectrum Disorder (ASD) with biomakrers. We were given a data set of with biomaker data and 67 samples that have ASD. The training data given to us includes a last column determines whether or not sample has ASD or not (has NEU - are neurotypical). Finally, our main task will be to predict which model is best used to classify these points: the Fisher Linear Discriminant Analysis (LDA) Method, or some other method, and with which features.

3.0 Data Description

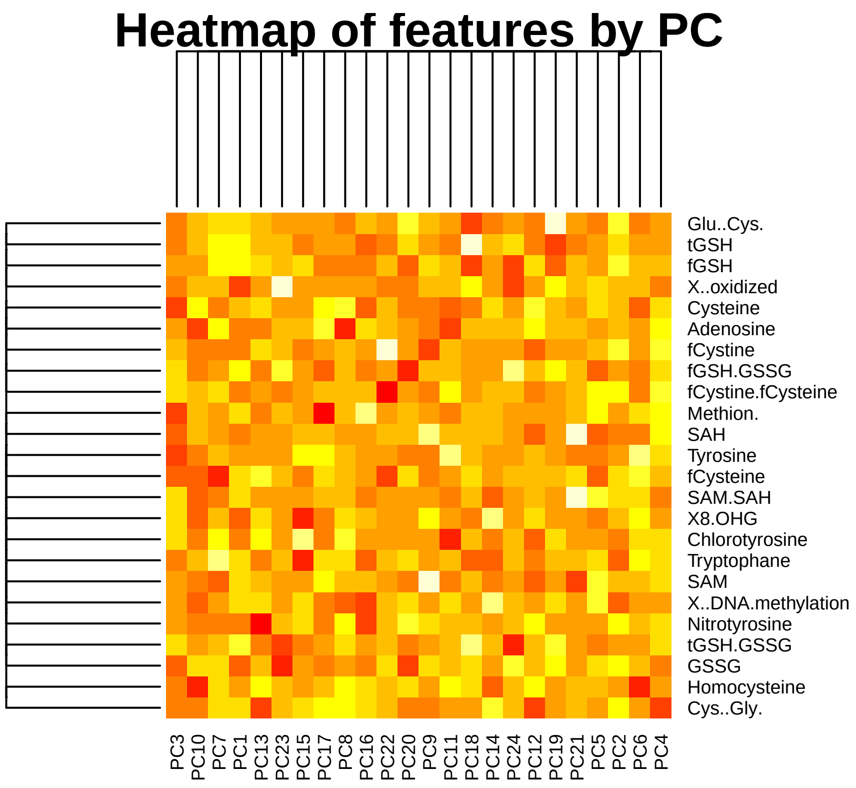
```r
Train.df <- read.csv('~/MATP-4400/data/autism_oxstress2.csv')
Train.df$Group<- factor(Train.df$Group,levels=c('NEU','ASD'))
Test.df<- read.csv('~/MATP-4400/data/autism_oxstress_val2.csv')
Test.df$Group <- factor(Test.df$Group,levels=c('NEU','ASD'))
# ^^ set group labels manually to ensure the underlying factor codes 'NEU' as 0 and 'ASD' as 1.
# This ensures that probabilities close to 1 indicate high likelihood of autisim and probabilities clos
Train.matrix<-as.matrix(Train.df[,-1])
Test.matrix <- as.matrix(Test.df[,-1])

sc_tr <- scale(Train.matrix) # scale tr
means <- attr(sc_tr, 'scaled:center') # get the mean of the columns
stdevs <- attr(sc_tr, 'scaled:scale') # get the std of the columns
sc_tst <- scale(Test.matrix, center=means, scale=stdevs)#scale tst using the means and std of tr
count_pos <- sum(Train.df$Group == 'ASD')
count_neg <- sum(Train.df$Group == 'NEU')
count_neg
```

```
## [1] 98
```

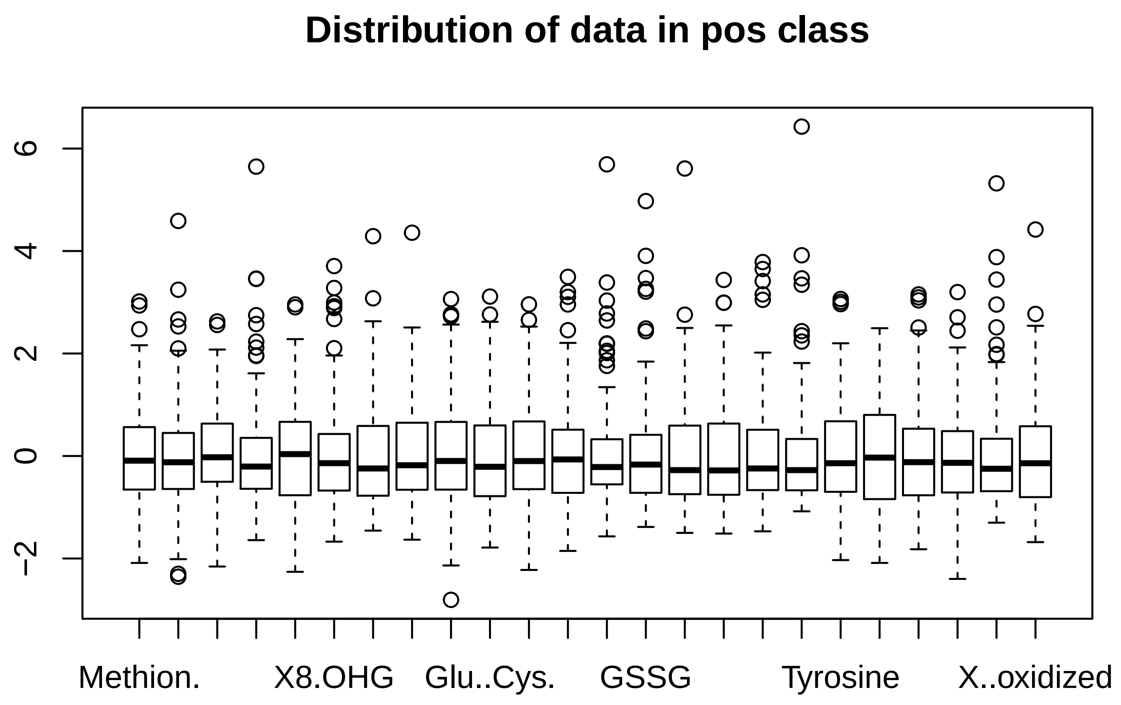
```r
count_pos
```

```
## [1] 67
```

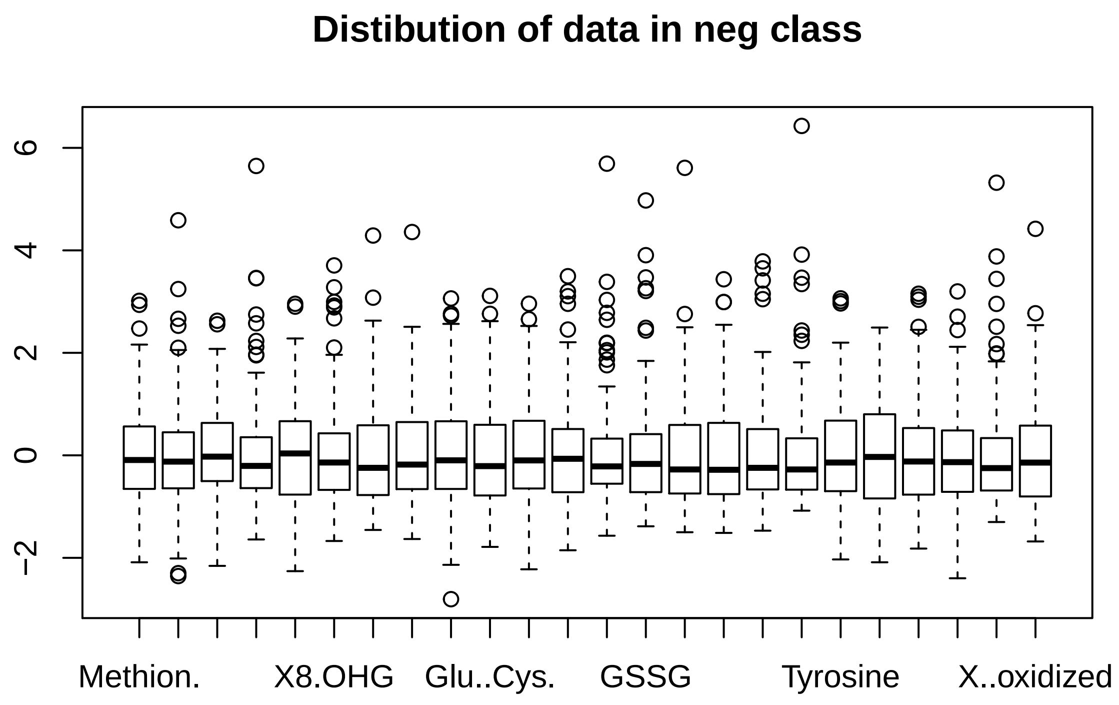
```r
my_pca <- prcomp(sc_tr,retx=TRUE,center=FALSE, scale=FALSE)
heatmap(my_pca$rotation, main = 'Heatmap of features by PC', cexRow = 0.75, cexCol = 0.75 )
```



```r
#my_pca <- prcomp(Train.matrix,retx=TRUE,center=FALSE, scale=FALSE)
#heatmap(my_pca$rotation, main = 'Heatmap of mean of each class', cexRow = 0.75, cexCol = 0.75 )

boxplot(sc_tr, data=count_pos, main="Distribution of data in pos class")
```



```r
boxplot(sc_tr, data=count_neg, main="Distibution of data in neg class")
```



4.0 Feature Importance using Univariate Logistic Regression

```
# Run logistic regression on variable with name i and store result in matrix res
# Set up res Matrix to hold results
res <- matrix(NA,nrow=ncol(Train.matrix),ncol=4)
rownames(res) <- colnames(Train.matrix)
colnames(res) <- c("Estimate","Std. Error","z value","Pr(>|z|)")

for(j in 1:ncol(Train.matrix)){
  i<-colnames(Train.matrix)[j]
  # Run logistic regression
  mymod <- glm(Group ~ Train.df[ ,i], data = Train.df,  family=binomial())
  res[i,] <- coef(summary(mymod))[2,]
  summary(mymod)
}
#res                                commented bc printing res at the end below

resPos <- matrix(NA, nrow=0, ncol=4)
resNeg <- matrix(NA, nrow=0, ncol=4)
BestFeatures <- matrix(NA, nrow=0, ncol=4)

colnames(resPos) <- c("Estimate","Std. Error","z value","Pr(>|z|)")
colnames(resNeg) <- c("Estimate","Std. Error","z value","Pr(>|z|)")
colnames(BestFeatures) <- c("Estimate","Std. Error","z value","Pr(>|z|)")
```

```r
posRowNames <- character(length = 0)
negRowNames <- character(length = 0)
BFRowNames <- character(length = 0)

for(k in 1:nrow(res)){
  if(res[k,1]>0){
    posLine <- matrix(res[k,], ncol=4)
    resPos <- rbind(resPos,posLine)
    posRowNames <- c(posRowNames,rownames(res)[k])
  }
  if(res[k,1]<=0){
    negLine <- matrix(res[k,], ncol=4)
    resNeg <- rbind(resNeg,negLine)
    negRowNames <- c(negRowNames,rownames(res)[k])
  }
  if(res[k,4]<=0.002){
    BFLine <- matrix(res[k,], ncol=4)
    BestFeatures <- rbind(BestFeatures,BFLine)
    BFRowNames <- c(BFRowNames,rownames(res)[k])
  }
}

rownames(resPos) <- posRowNames
#resPos                                commented bc printing resPos at the end below

rownames(resNeg) <- negRowNames
#resNeg                                commented bc printing resNeg at the end below

rownames(BestFeatures) <- BFRowNames
#BestFeatures                            commented bc printing resNeg at the end below


# printing all 3 so far
res
```

```
##                         Estimate   Std. Error   z value     Pr(>|z|)
## Methion.             -0.31337149  0.061670207 -5.081408 3.746474e-07
## SAM                  -0.07331479  0.018177035 -4.033374 5.498157e-05
## SAH                   0.13012047  0.040462306  3.215844 1.300615e-03
## SAM.SAH              -0.59722801  0.158605547 -3.765493 1.662211e-04
## X..DNA.methylation -1.15573148  0.229567154 -5.034394 4.793631e-07
## X8.OHG               77.21338590 13.010000336  5.934926 2.939785e-09
## Adenosine            10.71451880  2.992836455  3.580055 3.435220e-04
## Homocysteine          0.31235862  0.148240856  2.107102 3.510873e-02
## Cysteine             -0.04766407  0.010560765 -4.513316 6.382187e-06
## Glu..Cys.            -1.29559783  0.351083124 -3.690288 2.240003e-04
## Cys..Gly.            -0.03831276  0.023705798 -1.616177 1.060561e-01
## tGSH                 -0.86295932  0.172600257 -4.999757 5.740265e-07
## fGSH                 -2.49788900  0.536741509 -4.653803 3.258689e-06
## GSSG                 19.83013861  3.451322328  5.745664 9.156094e-09
## fGSH.GSSG            -0.42596155  0.069497242 -6.129186 8.832956e-10
## tGSH.GSSG            -0.13223744  0.021729699 -6.085563 1.160830e-09
## Chlorotyrosine        0.06442460  0.011092354  5.808019 6.321633e-09
## Nitrotyrosine       0.03401429  0.006034522  5.636616 1.734242e-08
## Tyrosine           -0.02077731  0.012113715 -1.715189 8.631063e-02
## Tryptophane        -0.02792591  0.018504986 -1.509102 1.312727e-01
## fCystine            0.10915304  0.023238068  4.697165 2.637970e-06
## fCysteine          -0.08926212  0.037445265 -2.383803 1.713479e-02
## fCystine.fCysteine  2.53876388  0.511503505  4.963336 6.929247e-07
## X..oxidized        37.56650887  5.691938548  6.599950 4.112969e-11
```

```
resPos
```

```
##                       Estimate   Std. Error  z value     Pr(>|z|)
## SAH                 0.13012047  0.040462306 3.215844 1.300615e-03
## X8.OHG             77.21338590 13.010000336 5.934926 2.939785e-09
## Adenosine          10.71451880  2.992836455 3.580055 3.435220e-04
## Homocysteine        0.31235862  0.148240856 2.107102 3.510873e-02
## GSSG               19.83013861  3.451322328 5.745664 9.156094e-09
## Chlorotyrosine      0.06442460  0.011092354 5.808019 6.321633e-09
## Nitrotyrosine       0.03401429  0.006034522 5.636616 1.734242e-08
## fCystine            0.10915304  0.023238068 4.697165 2.637970e-06
## fCystine.fCysteine  2.53876388  0.511503505 4.963336 6.929247e-07
## X..oxidized        37.56650887  5.691938548 6.599950 4.112969e-11
```

```
resNeg
```

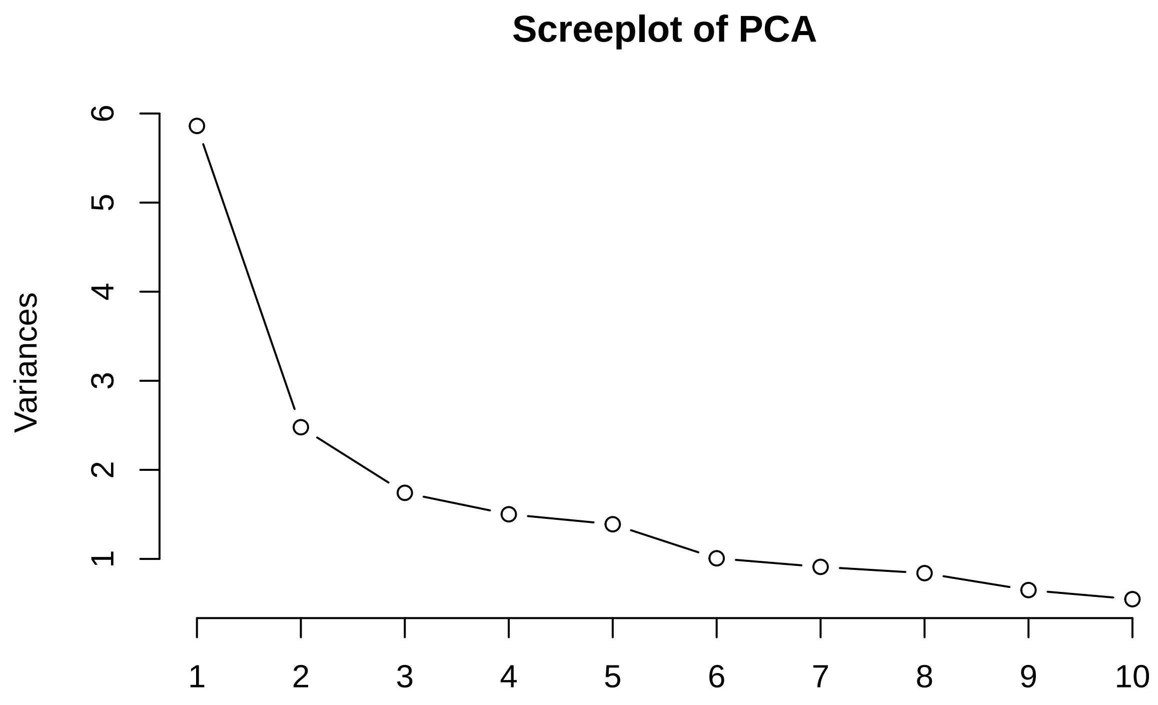
```
##                       Estimate Std. Error   z value     Pr(>|z|)
## Methion.           -0.31337149 0.06167021 -5.081408 3.746474e-07
## SAM                -0.07331479 0.01817704 -4.033374 5.498157e-05
## SAM.SAH            -0.59722801 0.15860555 -3.765493 1.662211e-04
## X..DNA.methylation -1.15573148 0.22956715 -5.034394 4.793631e-07
## Cysteine           -0.04766407 0.01056077 -4.513316 6.382187e-06
## Glu..Cys.          -1.29559783 0.35108312 -3.690288 2.240003e-04
## Cys..Gly.          -0.03831276 0.02370580 -1.616177 1.060561e-01
## tGSH               -0.86295932 0.17260026 -4.999757 5.740265e-07
## fGSH               -2.49788900 0.53674151 -4.653803 3.258689e-06
## fGSH.GSSG          -0.42596155 0.06949724 -6.129186 8.832956e-10
## tGSH.GSSG          -0.13223744 0.02172970 -6.085563 1.160830e-09
## Tyrosine           -0.02077731 0.01211372 -1.715189 8.631063e-02
## Tryptophane        -0.02792591 0.01850499 -1.509102 1.312727e-01
## fCysteine          -0.08926212 0.03744526 -2.383803 1.713479e-02
```

```
BestFeatures
```

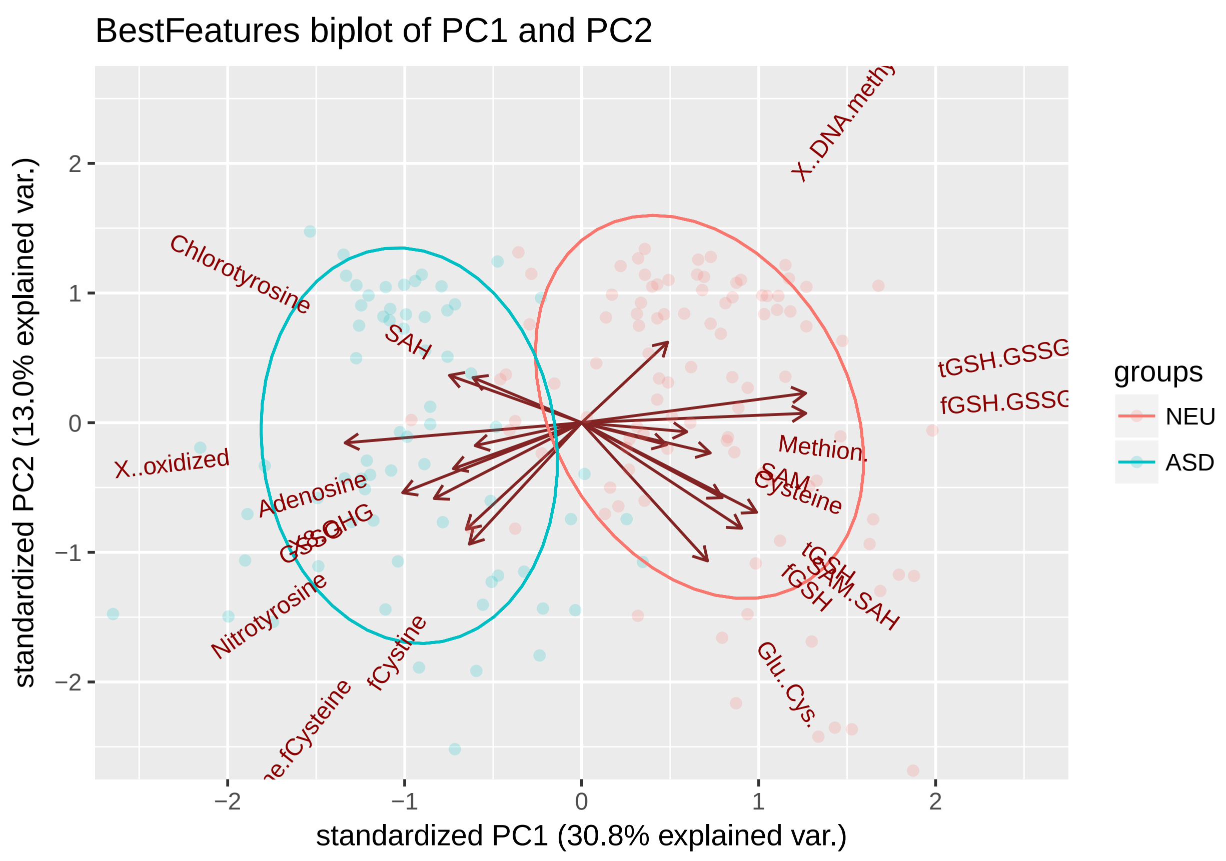
```
##                       Estimate   Std. Error   z value     Pr(>|z|)
## Methion.           -0.31337149  0.061670207 -5.081408 3.746474e-07
## SAM                -0.07331479  0.018177035 -4.033374 5.498157e-05
## SAH                 0.13012047  0.040462306  3.215844 1.300615e-03
## SAM.SAH            -0.59722801  0.158605547 -3.765493 1.662211e-04
## X..DNA.methylation -1.15573148  0.229567154 -5.034394 4.793631e-07
## X8.OHG             77.21338590 13.010000336  5.934926 2.939785e-09
## Adenosine          10.71451880  2.992836455  3.580055 3.435220e-04
## Cysteine           -0.04766407  0.010560765 -4.513316 6.382187e-06
## Glu..Cys.          -1.29559783  0.351083124 -3.690288 2.240003e-04
## tGSH               -0.86295932  0.172600257 -4.999757 5.740265e-07
## fGSH               -2.49788900  0.536741509 -4.653803 3.258689e-06
## GSSG               19.83013861  3.451322328  5.745664 9.156094e-09
## fGSH.GSSG          -0.42596155  0.069497242 -6.129186 8.832956e-10
## tGSH.GSSG          -0.13223744  0.021729699 -6.085563 1.160830e-09
## Chlorotyrosine      0.06442460  0.011092354  5.808019 6.321633e-09
## Nitrotyrosine       0.03401429  0.006034522  5.636616 1.734242e-08
## fCystine            0.10915304  0.023238068  4.697165 2.637970e-06
## fCystine.fCysteine  2.53876388  0.511503505  4.963336 6.929247e-07
## X..oxidized        37.56650887  5.691938548  6.599950 4.112969e-11
```

5.0 PCA Analysis

```
trainBF <- sc_tr[,rownames(BestFeatures)]

# trainBF <- matrix(NA, nrow=nrow(sc_tr), ncol=0)
# trBFcol <- character(length = 0)
#
# for(i in 1:nrow(BestFeatures)){
#   for(j in 1:ncol(sc_tr)){
#     if(rownames(BestFeatures)[i] == colnames(sc_tr)[j]){
#       trainBF <- cbind(trainBF, sc_tr[,j])
#       trBFcol <- c(trBFcol, colnames(sc_tr)[j])
#     }
#   }
# }
# colnames(trainBF) <- trBFcol
#trainBF


set.seed(300)
my.pca <- prcomp(trainBF, retx = TRUE, center = FALSE, scale = FALSE)

screeplot(my.pca, type="lines", main = "Screeplot of PCA")
```



```r
#km <- kmeans(trainBF, 4)

#Creating biplot
p <- ggbiplot(my.pca,
              choices=c(1,2),
              alpha=.2,
              varname.adjust=3,
              labels=rownames(trainBF),
              groups=Train.df$Group,
              ellipse=2)
p + ggtitle('BestFeatures biplot of PC1 and PC2') +  coord_cartesian(xlim=c(-2.5,2.5), ylim=c(-2.5,2.5)
```



6.0 LDA Model

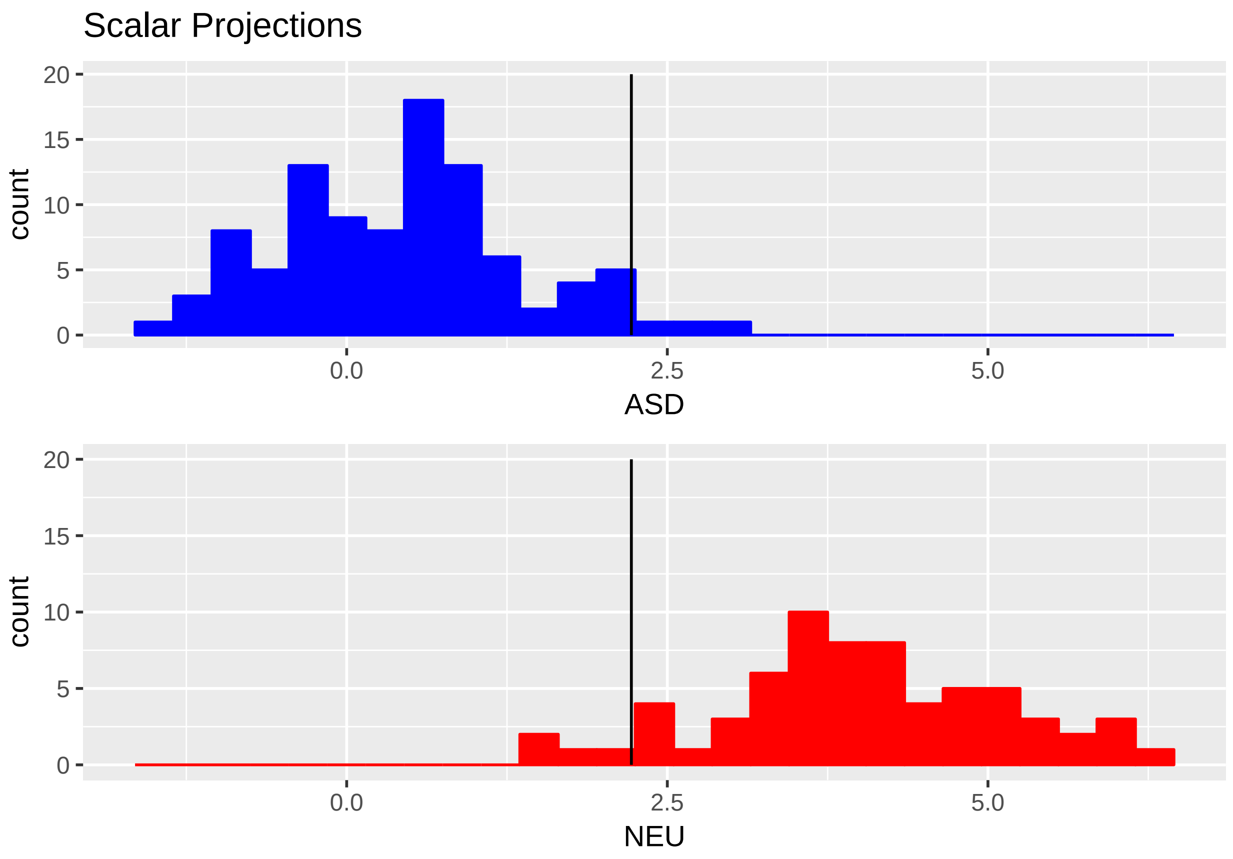
```
papervar <-cbind("X..DNA.methylation","X8.OHG","Glu..Cys.","fCystine.fCysteine","X..oxidized","Chloroty

paperdf <- Train.df[,papervar]
papermatrix <- as.matrix(paperdf)

lda.fit <- lda(Group ~ ., cbind(paperdf,Train.df["Group"]), prior=c(1,1)/2)

#Calculate the LDA threshold from the means and the normal vector.
thresh <- ((lda.fit$means[1,] +lda.fit$means[2,])/2)%*%lda.fit$scaling

#Compute the scalar projections of each class on the separating hyperplane.
projtrain1 <- papermatrix%*%as.matrix(lda.fit$scaling)
pplustrain1  <- projtrain1[Train.df$Group[ ]=='ASD'] #All the class 1 projections
pminustrain1 <- projtrain1[Train.df$Group[ ]=='NEU'] #All the class -1 projections

#% correctly classified as ASD
sum(pplustrain1>thresh[1])/length(pplustrain1)
```

```
## [1] 0.9402985
```

```
#% correctly classified as NEU
sum(pminustrain1<thresh[1])/length(pminustrain1)
```

```
## [1] 0.9591837
```

```
histopair(pminustrain1,pplustrain1,yy=c(0,20),thresh,label1="ASD",label2="NEU", bwid=0.3)
```



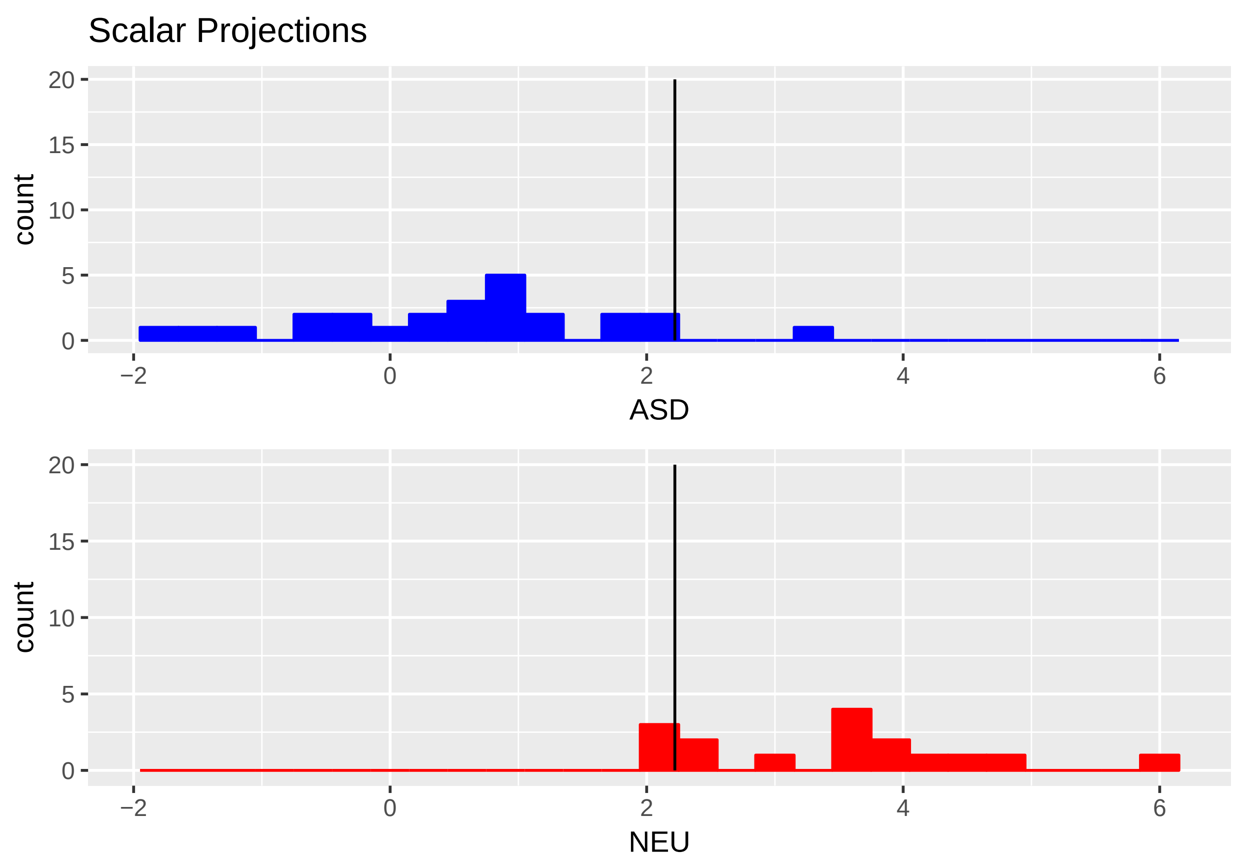
```
#Compute the scalar projections of each class on the separating hyperplane for the testing data.
papertest <- as.matrix(Test.df[,papervar])
projtest1 <- papertest%*%as.matrix(lda.fit$scaling)
pplustest1  <- projtest1[Test.df$Group[ ]=='ASD'] #All the class 1 projections
pminustest1 <- projtest1[Test.df$Group[ ]=='NEU'] #All the class -1 projections

#% correctly classified as ASD
sum(pplustest1>thresh[1])/length(pplustest1)
```

```
## [1] 0.8125
```

```
#% correctly classified as NEU
sum(pminustest1<thresh[1])/length(pminustest1)
```

```
## [1] 0.96
```

```
histopair(pminustest1,pplustest1,yy=c(0,20),thresh,label1="ASD",label2="NEU", bwid=0.3)
```



```
#LOO analysis
ypredict <- c(1:nrow(paperdf))

for (i in 1:nrow(paperdf)){
  paperdfloo <- paperdf[-i,]
  paperloo.matrix <- as.matrix(paperdf)
  loo <- paperloo.matrix[i,]
  paperloo.matrix <- paperloo.matrix[-i,]
  Trainloo.df <- Train.df[-i,]

  ldaloo.fit <- lda(Group ~ ., cbind(paperdfloo,Trainloo.df["Group"]), prior=c(1,1)/2)
  thresh <- ((ldaloo.fit$means[1,] +ldaloo.fit$means[2,])/2)%*%ldaloo.fit$scaling

  sproj <- loo%*%as.matrix(ldaloo.fit$scaling)

  if(sproj>thresh)
    ypredict[i]='ASD'
  else
    ypredict[i]='NEU'
}

#show percentage of correctly classified ASD points
correctpos = sum(Train.df[ypredict[]=='ASD',"Group"]=='ASD')
correctpos/sum(Train.df[,"Group"]=='ASD')
```

```
## [1] 0.9104478
```

```
#show percentage of correctly classified ASD points
correctneg = sum(Train.df[ypredict[]=='NEU',"Group"]=='NEU')
correctneg/sum(Train.df[,"Group"]=='NEU')
```

```
## [1] 0.9591837
```

```
#show total percentage correct
correct = sum(ypredict[]==Train.df["Group"])
correct/length(ypredict)
```

```
## [1] 0.9393939
```

7.0 Investigation of Alternative Models

```
# Run multiple logistic regression on data in data frame Train.df
# Using only variables in papervars.  Note this is LR model for unscaled data.  This shouldn't be one o
# You need to do the one for scaled data.
papervar <-c("X..DNA.methylation","X8.OHG","Glu..Cys.",
             "fCystine.fCysteine","X..oxidized","Chlorotyrosine","tGSH.GSSG")
fulldat <- cbind.data.frame(Group=Train.df$Group, Train.matrix[ ,papervar])
mymod <- glm(Group~.,data=fulldat,family=binomial())
# Predict all the data in Train.df
result <- predict(mymod,data=fulldat, type='response')
trainpred<-matrix(NA,nrow=length(result),ncol=1)
thresh<- 0.5
trainpred[result<=thresh] <- 'NEU'
trainpred[result>thresh] <- 'ASD'
# This commands make group names are in correct order
trainpred<- factor(trainpred,levels=c('NEU','ASD'))
trainactual<-Train.df$Group
table(trainactual,trainpred)
```

```
##            trainpred
## trainactual NEU ASD
##         NEU  94   4
##         ASD   4  63
```

```
summary(mymod)
```

```
##
## Call:
## glm(formula = Group ~ ., family = binomial(), data = fulldat)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.93063  -0.07596  -0.00486   0.07095   2.63023
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)           3.48484    5.49542   0.634  0.52599
## X..DNA.methylation   -1.90347    0.66093  -2.880  0.00398 **
## X8.OHG               70.17294   25.69929   2.731  0.00632 **
## Glu..Cys.            -1.54711    0.99413  -1.556  0.11965
## fCystine.fCysteine    2.01268    0.86865   2.317  0.02050 *
## X..oxidized         5.72356   15.76845   0.363  0.71662
## Chlorotyrosine      0.07436    0.02800   2.655  0.00792 **
## tGSH.GSSG          -0.16208    0.08630  -1.878  0.06035 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 222.880  on 164  degrees of freedom
## Residual deviance:  34.026  on 157  degrees of freedom
## AIC: 50.026
##
## Number of Fisher Scoring iterations: 8
```

```
datacomb<- cbind.data.frame(Group = Train.df$Group, Train.matrix[,papervar])
multmod<- glm(Group~.,data = datacomb, family = binomial())
#prediction of data in Train.df
pred<- predict(multmod, data = datacomb, type = 'response')
trainpred <- matrix(NA, nrow = length(pred), ncol= 1)
t <- 0.5
trainpred[pred<= t] <- 'NEU'
trainpred[pred>t] <- 'ASD'
#Making group names in the correct order
trainpred <- factor(trainpred, levels = c('NEU', 'ASD'))
correctrain <- Train.df$Group
table(correctrain, trainpred)
```

```
##            trainpred
## correctrain NEU ASD
##         NEU  94   4
##         ASD   4  63
```

```
summary(multmod)
```

```
##
## Call:
## glm(formula = Group ~ ., family = binomial(), data = datacomb)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.93063  -0.07596  -0.00486   0.07095   2.63023
##
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)         3.48484    5.49542   0.634  0.52599
## X..DNA.methylation -1.90347    0.66093  -2.880  0.00398 **
## X8.OHG             70.17294   25.69929   2.731  0.00632 **
## Glu..Cys.          -1.54711    0.99413  -1.556  0.11965
## fCystine.fCysteine  2.01268    0.86865   2.317  0.02050 *
## X..oxidized         5.72356   15.76845   0.363  0.71662
## Chlorotyrosine      0.07436    0.02800   2.655  0.00792 **
## tGSH.GSSG          -0.16208    0.08630  -1.878  0.06035 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 222.880  on 164  degrees of freedom
## Residual deviance:  34.026  on 157  degrees of freedom
## AIC: 50.026
##
## Number of Fisher Scoring iterations: 8
```

```
#SVM (weesnaw!!!!)
papervar <-cbind("X..DNA.methylation","X8.OHG","Glu..Cys.","fCystine.fCysteine","X..oxidized","Chloroty

paperdf <- Train.df[,papervar]
papermatrix <- as.matrix(paperdf)

svm.fit <- svm(Group ~ ., cbind(paperdf,Train.df["Group"]), prior=c(1,1)/2)

trainpredict <- as.matrix(predict(svm.fit,as.matrix(Train.df[,papervar])))
testpredict <- as.matrix(predict(svm.fit,as.matrix(Test.df[,papervar])))

#show percentage of correctly classified ASD training points
correctpos2 = sum(Train.df[trainpredict[]=='ASD',"Group"]=='ASD')
correctpos2/sum(trainpredict[]=='ASD')
```

```
## [1] 0.984375
```

```
#show percentage of correctly classified NEU training points
correctneg2 = sum(Train.df[trainpredict[]=='NEU',"Group"]=='NEU')
correctneg2/sum(trainpredict[]=='NEU')
```

```
## [1] 0.960396
```

```
#percentage correct on all training data
sum(trainpredict==Train.df["Group"])/length(trainpredict)
```

```
## [1] 0.969697
```

```
#show percentage of correctly classified ASD training points
correctpos3 = sum(Test.df[testpredict[]=='ASD',"Group"]=='ASD')
correctpos3/sum(testpredict[]=='ASD')
```

```
## [1] 0.9230769
```

```
#show percentage of correctly classified NEU training points
correctneg3 = sum(Test.df[testpredict[]=='NEU',"Group"]=='NEU')
correctneg3/sum(testpredict[]=='NEU')
```

```
## [1] 0.8571429
```

```
#percentage correct on all testing data
sum(testpredict==Test.df["Group"])/length(testpredict)
```

```
## [1] 0.8780488
```

```
#yeah that plsda
papervar <-cbind("X..DNA.methylation","X8.OHG","Glu..Cys.","fCystine.fCysteine","X..oxidized","Chloroty

paperdf <- Train.df[,papervar]
papermatrix <- as.matrix(paperdf)
```

```r
plsda.fit <- plsda(papermatrix, as.factor(Train.df[,"Group"]), probMethod = "Bayes")

trainpredict3 <- as.matrix(predict(plsda.fit, papermatrix))
testpredict3 <- as.matrix(predict(plsda.fit, as.matrix(Test.df[,papervar])))

#show percentage of correctly classified ASD training points
correctpos5 = sum(Train.df[trainpredict3[]=='ASD',"Group"]=='ASD')
correctpos5/sum(trainpredict3[]=='ASD')
```

```
## [1] 0.9137931
```

```r
#show percentage of correctly classified NEU training points
correctneg5 = sum(Train.df[trainpredict3[]=='NEU',"Group"]=='NEU')
correctneg5/sum(trainpredict3[]=='NEU')
```

```
## [1] 0.8691589
```

```r
#percentage correct on all training data
sum(trainpredict3==Train.df["Group"])/length(trainpredict3)
```

```
## [1] 0.8848485
```

```r
#show percentage of correctly classified ASD training points
correctpos6 = sum(Test.df[testpredict3[]=='ASD',"Group"]=='ASD')
correctpos6/sum(testpredict3[]=='ASD')
```

```
## [1] 0.8125
```

```r
#show percentage of correctly classified NEU training points
correctneg6 = sum(Test.df[testpredict3[]=='NEU',"Group"]=='NEU')
correctneg6/sum(testpredict3[]=='NEU')
```

```
## [1] 0.88
```

```r
#percentage correct on all testing data
sum(testpredict3==Test.df["Group"])/length(testpredict3)
```

```
## [1] 0.8536585
```

```r
confusionMatrix(trainpredict3,as.factor(Train.df[,"Group"]))
```

```
## Warning in confusionMatrix.default(trainpredict3, as.factor(Train.df[,
## "Group"])): Levels are not in the same order for reference and data.
## Refactoring data to match.
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction NEU ASD
##        NEU  93  14
##        ASD   5  53
##
##                Accuracy : 0.8848
##                  95% CI : (0.826, 0.9292)
##     No Information Rate : 0.5939
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.7561
##  Mcnemar's Test P-Value : 0.06646
```

```
##
##             Sensitivity : 0.9490
##             Specificity : 0.7910
##          Pos Pred Value : 0.8692
##          Neg Pred Value : 0.9138
##              Prevalence : 0.5939
##          Detection Rate : 0.5636
##    Detection Prevalence : 0.6485
##       Balanced Accuracy : 0.8700
##
##        'Positive' Class : NEU
##
```

```
confusionMatrix(testpredict3,as.factor(Test.df[,"Group"]))
```

```
## Warning in confusionMatrix.default(testpredict3, as.factor(Test.df[,
## "Group"])): Levels are not in the same order for reference and data.
## Refactoring data to match.
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction NEU ASD
##        NEU  22   3
##        ASD   3  13
##
##                Accuracy : 0.8537
##                  95% CI : (0.7083, 0.9443)
##     No Information Rate : 0.6098
##     P-Value [Acc > NIR] : 0.0006365
##
##                   Kappa : 0.6925
##  Mcnemar's Test P-Value : 1.0000000
##
##             Sensitivity : 0.8800
##             Specificity : 0.8125
##          Pos Pred Value : 0.8800
##          Neg Pred Value : 0.8125
##              Prevalence : 0.6098
##          Detection Rate : 0.5366
##    Detection Prevalence : 0.6098
##       Balanced Accuracy : 0.8462
##
##        'Positive' Class : NEU
##
```

8.0 Feature Challenge

```
#using the univariate logistic regression in part 4, find the next 7 best features
nonpaperbest<-BestFeatures[setdiff(rownames(BestFeatures), papervar),]
sorted <- nonpaperbest[,order('Pr(>|z|)')]
sorted[1:7]
```

```
##    Methion.         SAM         SAH      SAM.SAH   Adenosine    Cysteine
## -0.31337149 -0.07331479  0.13012047 -0.59722801 10.71451880 -0.04766407
##        tGSH
## -0.86295932
```

```
npapervar <- cbind("Methion.","SAM","SAH","SAM.SAH","Adenosine","Cysteine","tGSH")

#do svm on this
npaperdf <- Train.df[,npapervar]
npapermatrix <- as.matrix(npaperdf)

svm.fit2 <- svm(Group ~ ., cbind(npaperdf,Train.df["Group"]), prior=c(1,1)/2)

trainpredict2 <- as.matrix(predict(svm.fit2,as.matrix(Train.df[,npapervar])))
testpredict2 <- as.matrix(predict(svm.fit2,as.matrix(Test.df[,npapervar])))

#show percentage of correctly classified ASD training points
correctpos3 = sum(Train.df[trainpredict2[]=='ASD',"Group"]=='ASD')
correctpos3/sum(trainpredict2[]=='ASD')
```

```
## [1] 0.8412698
```

```
#show percentage of correctly classified NEU training points
correctneg3 = sum(Train.df[trainpredict2[]=='NEU',"Group"]=='NEU')
correctneg3/sum(trainpredict2[]=='NEU')
```

```
## [1] 0.8627451
```

```
#percentage correct on all training data
sum(trainpredict2==Train.df["Group"])/length(trainpredict2)
```

```
## [1] 0.8545455
```

```
#show percentage of correctly classified ASD training points
correctpos4 = sum(Test.df[testpredict2[]=='ASD',"Group"]=='ASD')
correctpos4/sum(testpredict2[]=='ASD')
```

```
## [1] 0.9230769
```

```
#show percentage of correctly classified NEU training points
correctneg4 = sum(Test.df[testpredict2[]=='NEU',"Group"]=='NEU')
correctneg4/sum(testpredict2[]=='NEU')
```

```
## [1] 0.8571429
```

```
#percentage correct on all testing data
sum(testpredict2==Test.df["Group"])/length(testpredict2)
```

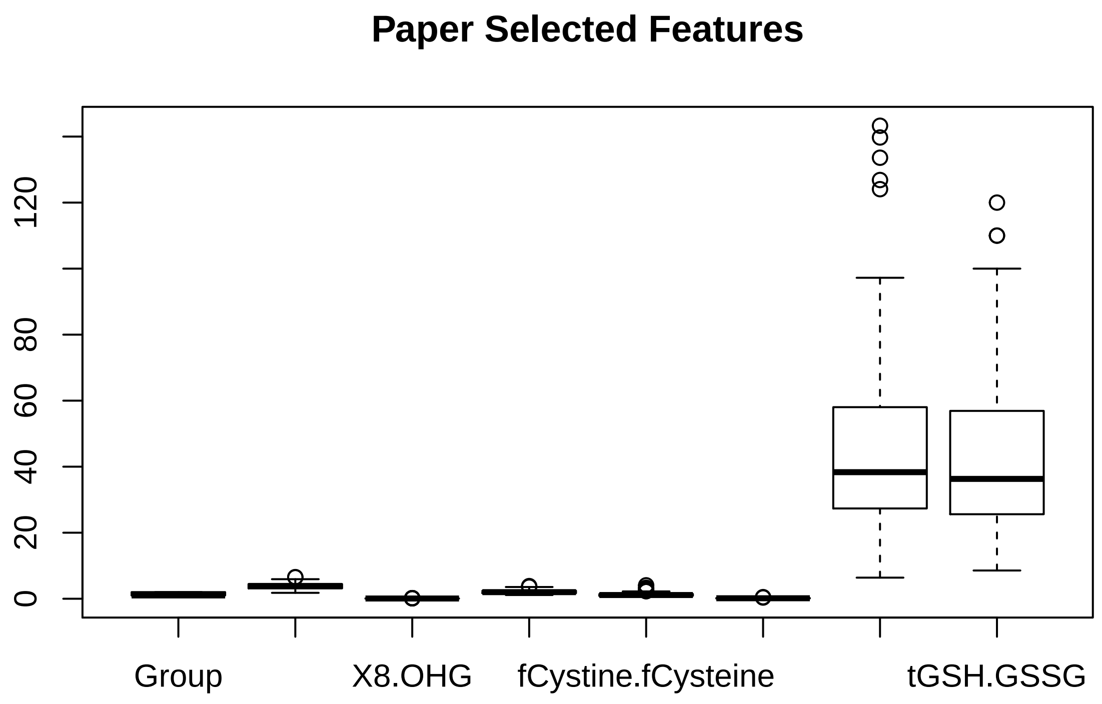
```
## [1] 0.8780488
```

9.0 Additional Analysis and Visualizations

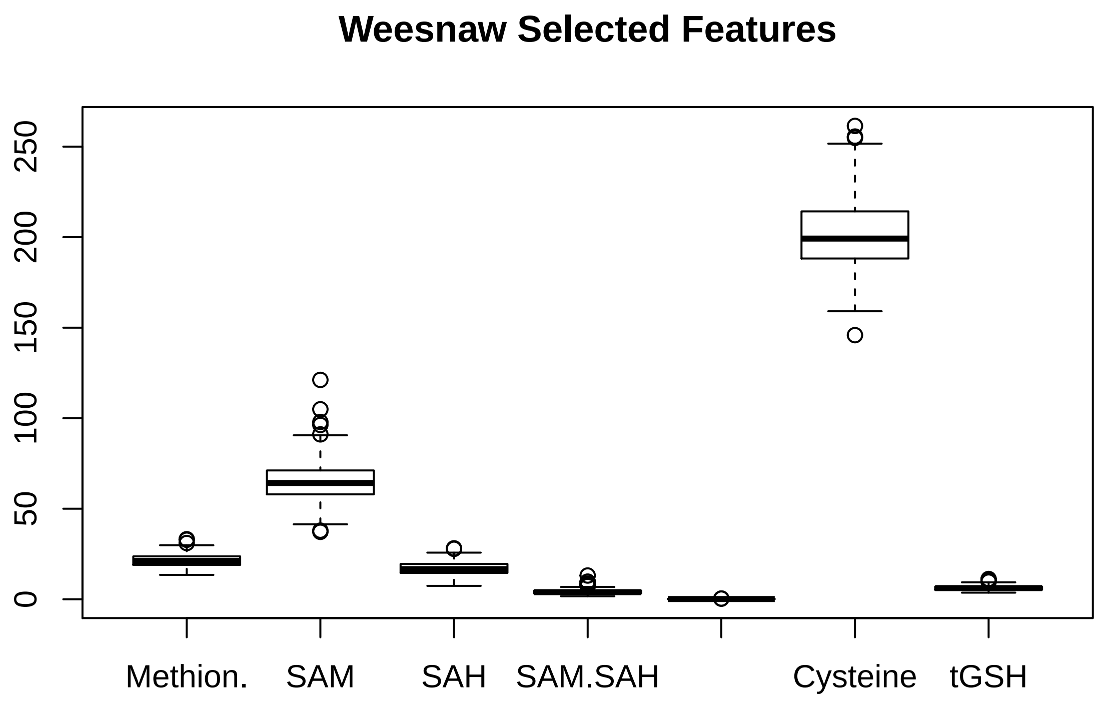
Each group member should do additional analyses or visualization using one or more R commands not covered in class. Indicate the R commands you used. Do the analysis. Discuss the results

9.1 Additional Results from Tenzin

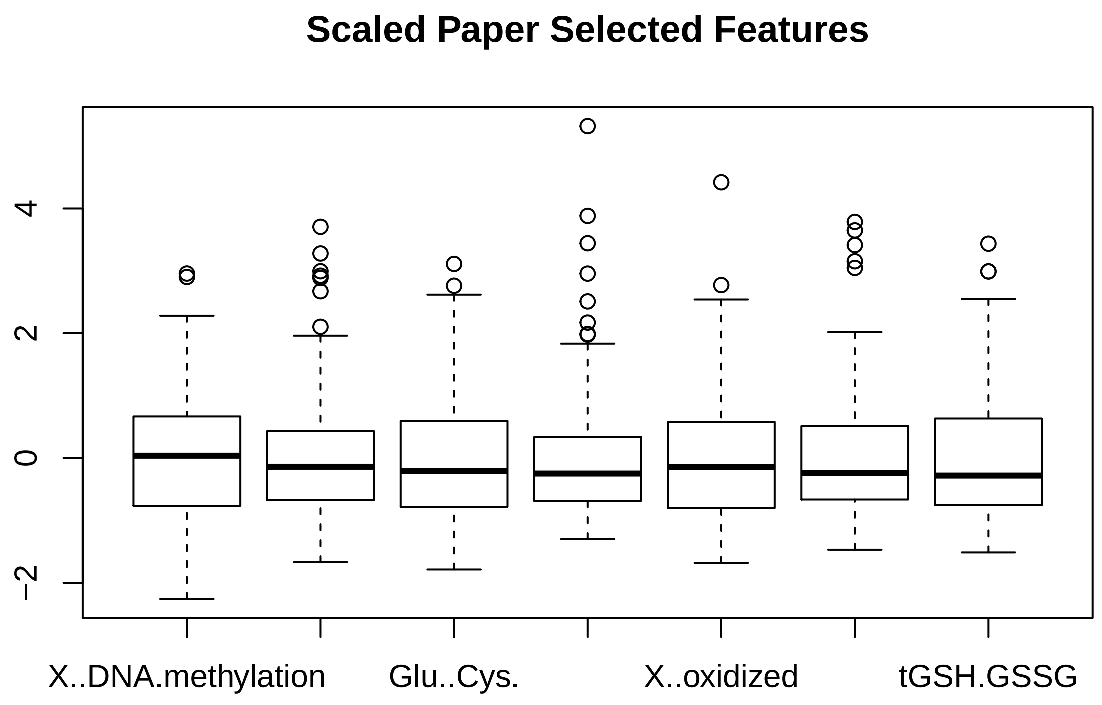
```r
# In attempts to see what differences there exist between
# the features that Dr. Hahn selected, and we(esnaw) selected
# I have plotted boxplots of both sets of features, scaled
# and unscaled.
# As expected, the boxplots show relatively similar trends
# which makes sense as the models using the two sets were
# still relativly similar in accuracy
# However, Dr. Hahn's paper features seem to have more
# consistency:
# in the unscaled plots, there is less variation in the
# range of values, and in the scaled plots, there are
# fewer outliers, especially below the first quartile

boxplot(fulldat, main="Paper Selected Features")
```



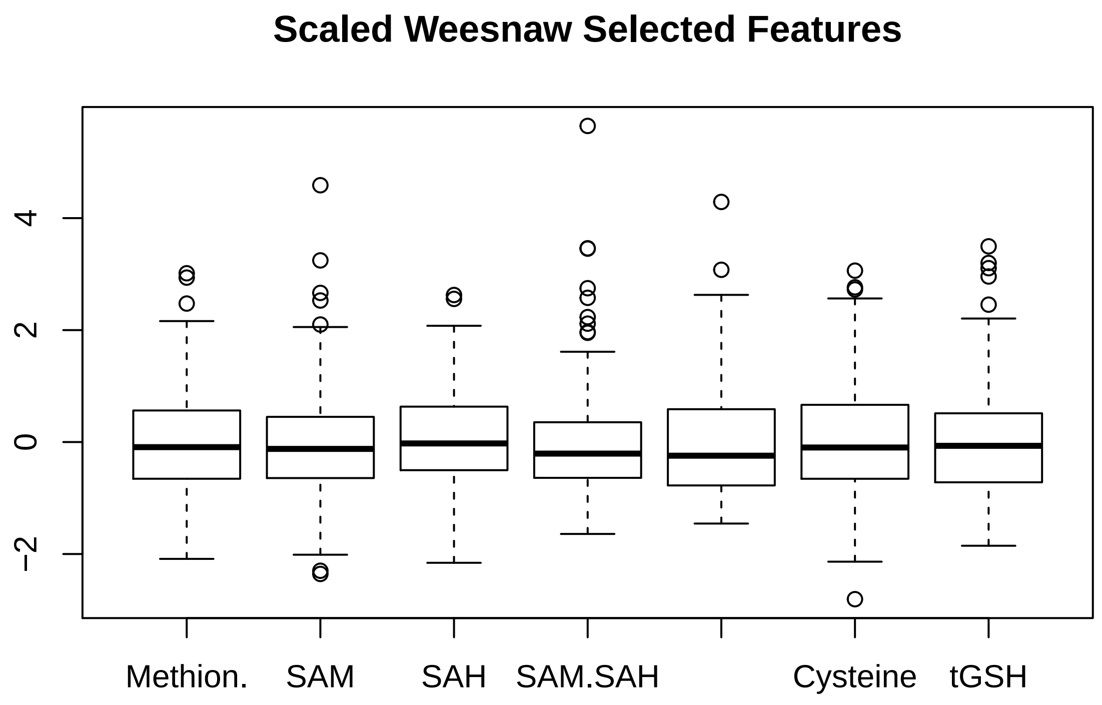
```r
boxplot(npaperdf, main="Weesnaw Selected Features")
```



```
sc_fulldat <- scale(fulldat[,-1])
sc_npaperdf <- scale(npaperdf)

boxplot(sc_fulldat, main="Scaled Paper Selected Features")
```

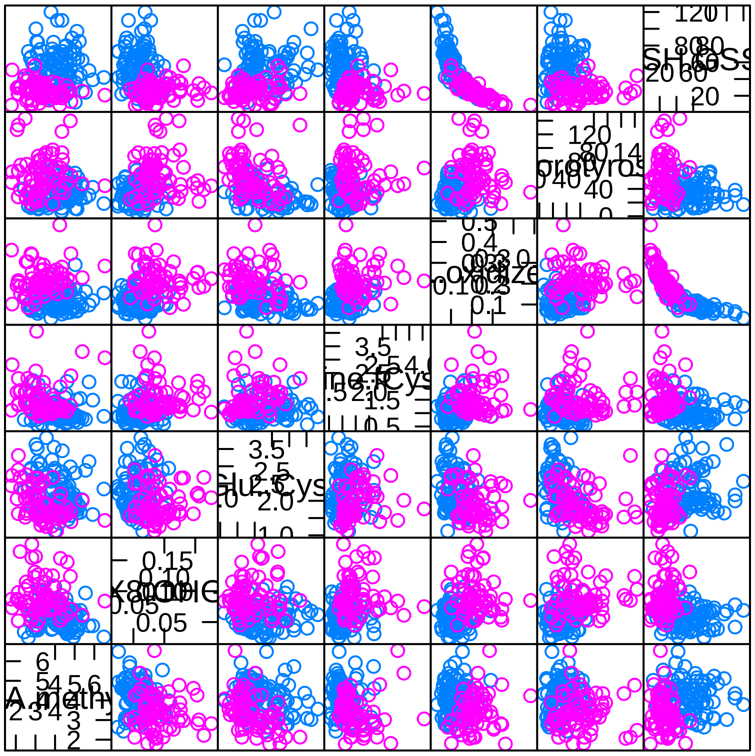


```r
boxplot(sc_npaperdf, main="Scaled Weesnaw Selected Features")
```



# 9.2 Additional Results from Ramin

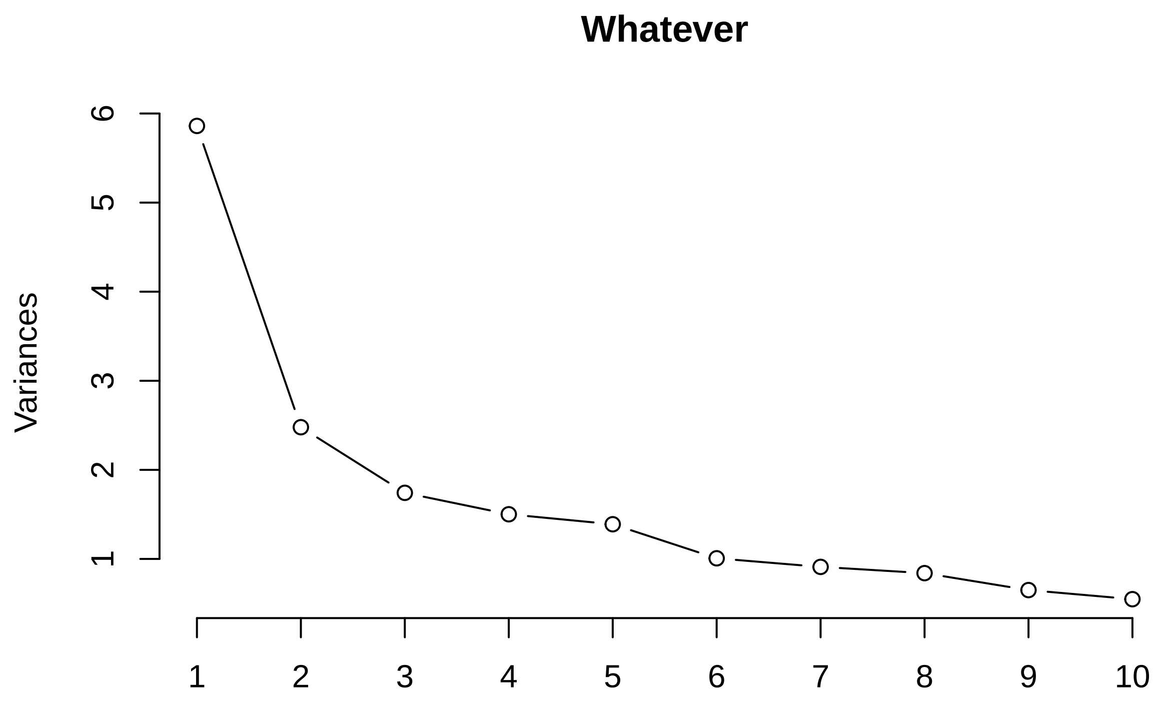
```
# Here we can see how when we plot the 7
# paper variables against each other most
# of them appear to be very separable

featurePlot(x=paperdf, y=Train.df[,"Group"], plot="pairs")
```



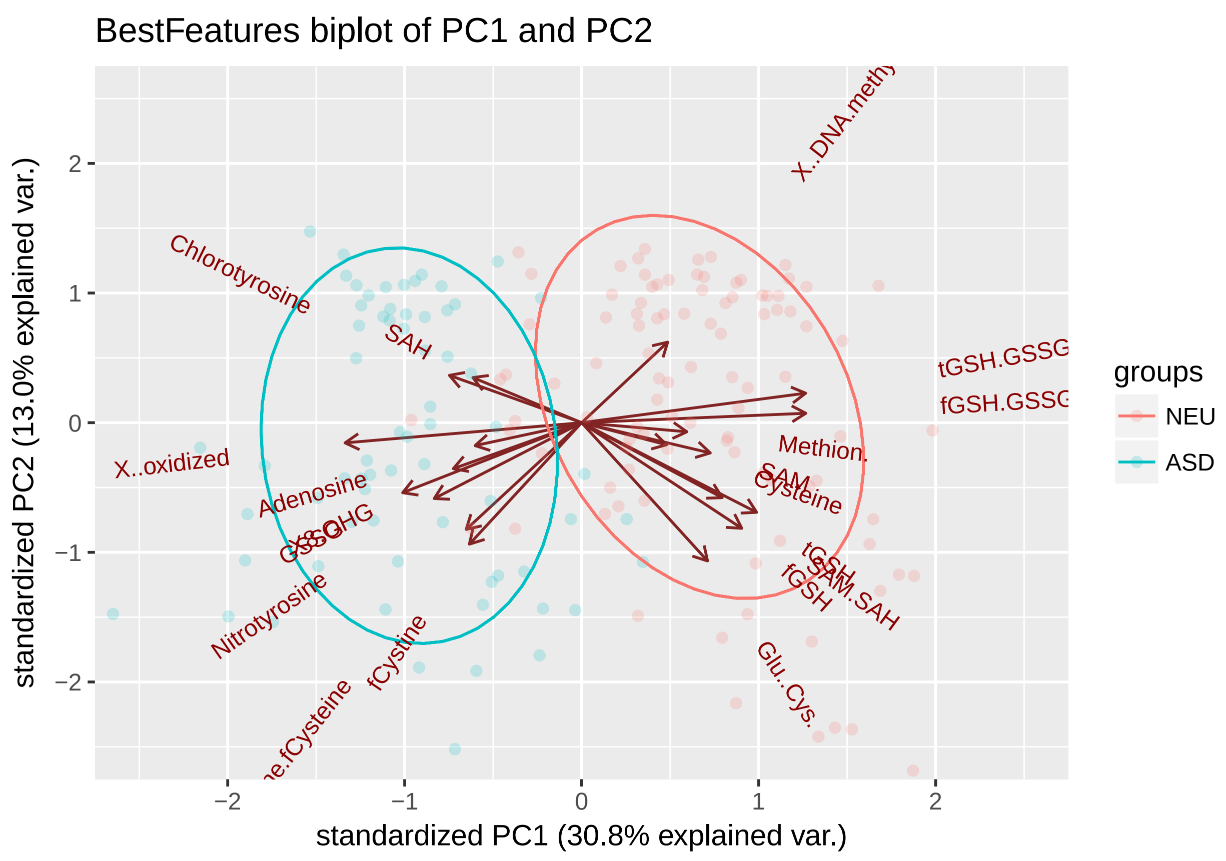
Scatter Plot Matrix

## 9.3 Additional Results from Madison

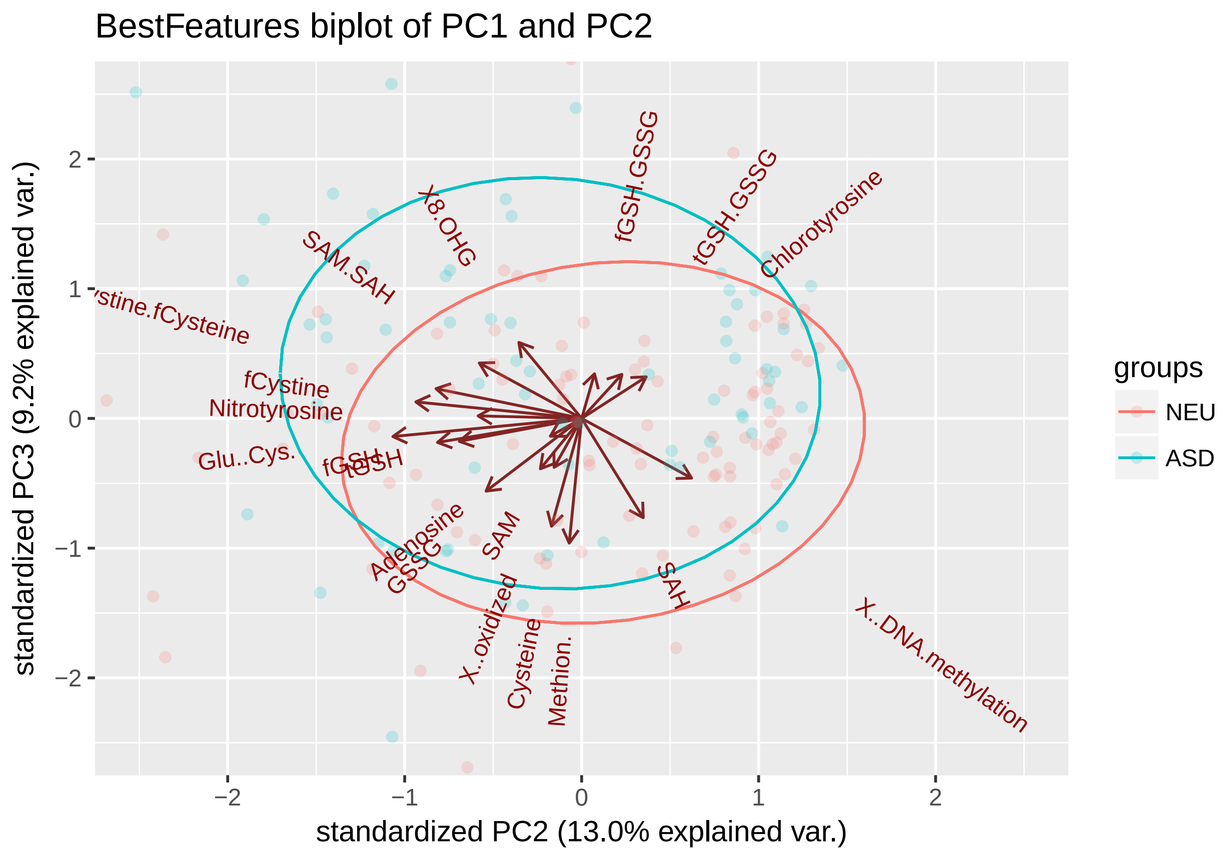
```
# Between principle components 1&2 we can see
# that the classfication between patients
# classified as "NEU" and patients classified
# as "ASD" is well separated. Meaning there is
# a clear distinction between what variables
# describe the classification. However, we can
# see between principles 2&3 and 3&4, since the
# two classifications ovelap, it's hard to distinguish
# which variables are important for classification of NEU and ASD

my.pca <- prcomp(trainBF, retx = TRUE, center = FALSE, scale = FALSE)
screeplot(my.pca, type="lines", main = "Whatever")
```



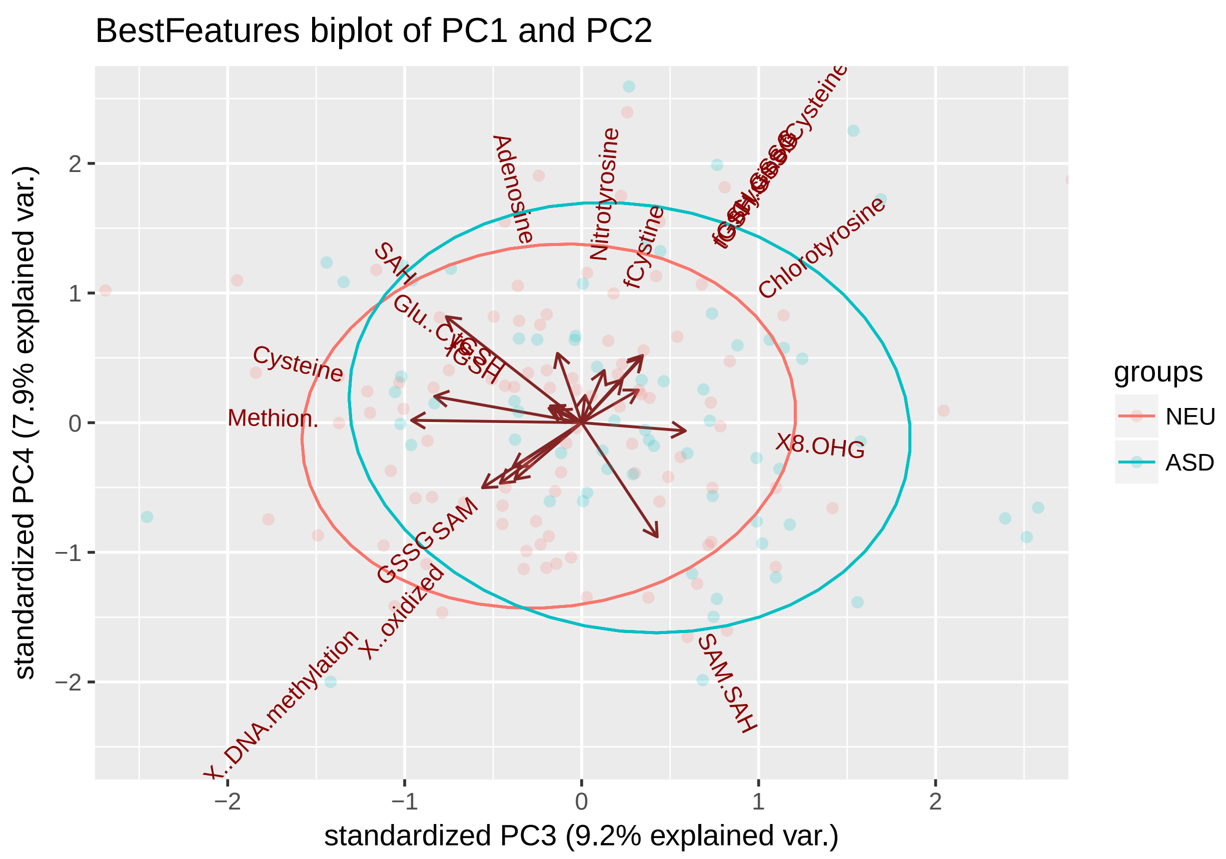
```
#km <- kmeans(trainBF, 4)
#Creating biplot
p <- ggbiplot(my.pca,
            choices=c(1,2),
            alpha=.2,
            varname.adjust=3,
            labels=rownames(trainBF),
            groups=Train.df$Group,
            ellipse=2)
p + ggtitle('BestFeatures biplot of PC1 and PC2') +  coord_cartesian(xlim=c(-2.5,2.5), ylim=c(-2.5,2.5)
```



```
# ^^ convenient function for cbinding and then making the result a data frame
p1 <- ggbiplot(my.pca,
            choices=c(2,3),
            alpha=.2,
            varname.adjust=3,
            labels=rownames(trainBF),
            groups=Train.df$Group,
            ellipse=2)
p1 + ggtitle('BestFeatures biplot of PC1 and PC2') +  coord_cartesian(xlim=c(-2.5,2.5), ylim=c(-2.5,2.5
```



```
p2 <- ggbiplot(my.pca,
            choices=c(3,4),
            alpha=.2,
            varname.adjust=3,
            labels=rownames(trainBF),
            groups=Train.df$Group,
            ellipse=2)
p2 + ggtitle('BestFeatures biplot of PC1 and PC2') +  coord_cartesian(xlim=c(-2.5,2.5), ylim=c(-2.5,2.5
```



## 9.4 Additional Results from Sebastian

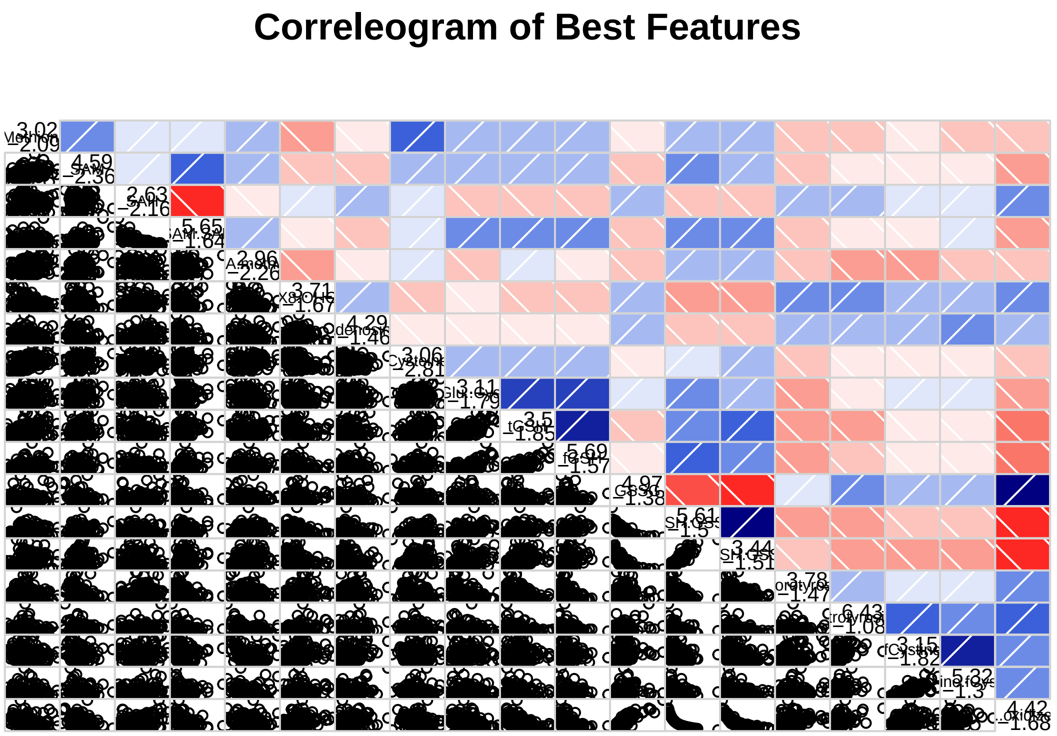
```r
# The upper panel shows the shading to show correlation between the Best Features
# The lower panel shows the different points of the Best Features
# The diagonal panel shows the min and max of the Beat Features

# The results show that Tyrosine is the closest
# Best Feature related to Tryptophane as shown through
# the darker colors (blue). The points on the other hand,
# show the different points relating to the different features.


install.packages("corrgram", dependencies = TRUE)
```

```
## Installing package into '/home/tashit/R/x86_64-pc-linux-gnu-library/3.4'
## (as 'lib' is unspecified)
```

```r
library(corrgram)
```

```
##
## Attaching package: 'corrgram'

## The following object is masked from 'package:plyr':
##
##     baseball
```

```r
#Installed package "corrgram" and created a Correleogram of the Best Features matrix
corrgram(trainBF, order = NULL, lower.panel=panel.pts, text.panel = panel.txt,
         upper.panel = panel.shade, diag.panel = panel.minmax,
         main = "Correleogram of Best Features")
```



# 10.0 Final Predictive Model Weesnaw supports the usage of both the Fisher LDA Model, and the SVM model for this investigation. Both are recommended because of slightly varied results in our analysis, this may work itself out with further testing. The variation was in that while the SVM model proved more accurate on the training set, the LDA model was more accurate on the testing set. Both results are desired as having an accurate model for training can lead to accurate testing identification, and having accurate testing identification is the goal of using such models.

11.0 Conclusion

In the end, Weesnaw has come to the conclusion that in this first round of analysis, that Dr. Hahn has indeed picked out more relevant features than Weesnaw has. This result is not unexpected, but there is always the possibility to improve.