Cancer Colorectal PLS-DA

Guillem Soler y Rubén Ribes

7/5/2022

PLS-DA-CANCER COLORECTAL

In this appendix we will analyse the results of running the PLS-DA technique on the 4 different preprocessing on the data we have on the CANCER COLORECTAL disease. The procedure followed on the 4 preprocessed samples is the same as the one explained in the main report, so we will not repeat the explanation. In all the pre-processing of the data we have, we are going to divide into train set and test set to perform the validation of the model, in our case, the test partition corresponds to 20% of the total observations we have.

```
library(phyloseq)

load("DatosSinOutlier.RData")

library(DataExplorer)
library(rmarkdown)
library(dplyr)
library(FactoMineR)
library(factoextra)
library(ropls)
library(pls)
library(caret)
library(ROCR)
library(MLmetrics)
library(vip)
```

Loading of data set S1.

We import the data. As can be seen, we centre and scale the data and then apply the PLS-DA model to the data. We also perform a transformation on the preprocessed TSS data, this transformation allows a better interpretation of the data.

```
species = Datos_Taxa$CancerColorectal

sample = as.data.frame(Datos_sample$CancerColorectal)
a = factor(sample$group, levels=c("control","crc"),labels=c("healthy","ill"))

tss = as.data.frame(log(as.data.frame(t(DatosFinal$CancerColorectal$S1_NORMAL$S1_TSS_
CANCERCOL)) * 10e6 + 1))
clr = as.data.frame(t(DatosFinal$CancerColorectal$S1_NORMAL$S1_CLR_CANCERCOL))

tss = as.data.frame(scale(tss))
clr = as.data.frame(scale(clr))

clr$disease = a
tss$disease = a
```

S1_CLR

Train AND test

We separate the data into train and test, the percentage chosen for the train is 80%, with this percentage we have 98 observations for train and 23 observations for test.

```
set.seed(280)
filas.clr = createDataPartition(clr$disease, p=0.8, list=FALSE)
clrTrain = clr[filas.clr,]
clrTest = clr[-filas.clr,]

dim(clrTest)
```

```
## [1]  23 494
```

```
dim(clrTrain)
```

```
## [1]  98 494
```

Pls-da

The first PLS we performed on the S1 dataset with the CLR pre-processing obtained an AUC of 0.73, this PLS was performed with all the variables of the model and without previous cross-validation.

```
set.seed(280)

library(vip)



plsprobs = plsda(x=select(clrTrain,-disease), y=clrTrain$disease, type="prob", probMe
thod = "softmax")

pred = predict(plsprobs, newdata=select(clrTest,-disease) , type="prob")

pred = as.data.frame(pred)
colnames(pred) = c("healthy","ill")
prediction = prediction(pred$ill,clrTest$disease)
perf = performance(prediction,"auc")
as.numeric(perf@y.values)
```

```
## [1] 0.7301587
```

Pls-da according to the VIP

We make a first loop by means of which we are going to try to improve the results obtained by means of the pls-da on the whole set of variables. The loops performed have been different, in this case, it can be seen that we are only going to reach the first 25 variables, i.e., the loop will start by performing a pls-da with two OTUS

and will add OTUS one by one until reaching 25. These 25 variables are ordered according to their importance in the model. This loop will be performed for all datasets, with the aim of achieving the best possible model.

```
set.seed(280)

library(vip)

resultados = data.frame()

plsprobs = plsda(x=select(clrTrain,-disease), y=clrTrain$disease, type="prob", probMe
thod = "softmax")

g = vip(plsprobs, num_features = 25)
v = g$data$Variable

for (i in 2:length(v)) {
  print(i)
  otus = v[1:i]
  newdata = as.data.frame(clrTrain[otus])
  newdata$disease = clrTrain$disease

  #we train the models with cv with the selected variables based on the auc and take
 the best one.
  tr_fit = trainControl(method= 'repeatedcv',number = 10, repeats = 10, classProbs =
TRUE,summaryFunction = twoClassSummary)
  model = train(disease~., data=newdata, trControl = tr_fit,method = 'pls',metric =
'ROC')
  c = model$results$ncomp[which(model$results$ROC==max(model$results$ROC))]


  #We train the one that gives us the best result with the whole training set and pre
dict the test set. We keep the results obtained
  plsprobs = plsda(x=select(newdata,-disease), y=newdata$disease, ncomp = c, type="pr
ob", probMethod = "softmax")
  pred = predict(plsprobs, newdata=clrTest[otus], type="prob")
  pred = as.data.frame(pred)
  colnames(pred) = c("healthy","ill")
  prediction = prediction(pred$ill,clrTest$disease)
  perf = performance(prediction,"auc")
  resultados = rbind(resultados, data.frame(importantOtus=i,ncomp=c,auc=as.numeric(pe
rf@y.values)))


}
```

```
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
## [1] 11
## [1] 12
## [1] 13
## [1] 14
## [1] 15
## [1] 16
## [1] 17
## [1] 18
## [1] 19
## [1] 20
## [1] 21
## [1] 22
## [1] 23
## [1] 24
## [1] 25
```

```
print(resultados)
```

```
##    importantOtus ncomp       auc
## 1              2     1 0.4047619
## 2              3     1 0.5595238
## 3              4     3 0.6547619
## 4              5     1 0.7579365
## 5              6     1 0.7619048
## 6              7     1 0.8333333
## 7              8     1 0.7936508
## 8              9     1 0.7539683
## 9             10     1 0.6904762
## 10            11     1 0.6984127
## 11            12     1 0.7857143
## 12            13     1 0.7857143
## 13            14     1 0.7301587
## 14            15     1 0.7698413
## 15            16     1 0.7698413
## 16            17     1 0.7619048
## 17            18     1 0.7857143
## 18            19     1 0.7857143
## 19            20     1 0.7936508
## 20            21     1 0.7301587
## 21            22     1 0.7301587
## 22            23     1 0.7301587
## 23            24     1 0.7539683
## 24            25     1 0.7777778
```

Using the work we have done, we managed to increase the AUC considerably, as can be seen, we achieved an AUC of 0.833. In the code that we have done we modify different aspects to achieve the best combination. First we order the variables from most to least important using the vip , and then we make pls models increasingly the number of variables. First the two most important ones, then the three most important ones, and so on . In this case, the number of variables that give us the best result is 7 and the components used by model are 1.

```
set.seed(280)

library(vip)

resultados = data.frame()

plsprobs = plsda(x=select(clrTrain,-disease), y=clrTrain$disease, type="prob", probMe
thod = "softmax")

g = vip(plsprobs, num_features = 7)
v = g$data$Variable

  otus = v[1:7]
  newdata = as.data.frame(clrTrain[otus])
  newdata$disease = clrTrain$disease


  tr_fit = trainControl(method= 'repeatedcv',number = 10, repeats = 10, classProbs =
TRUE,summaryFunction = twoClassSummary)
  model = train(disease~., data=newdata, trControl = tr_fit,method = 'pls',metric =
'ROC')
  plsprobs = plsda(x=select(newdata,-disease), y=newdata$disease, ncomp = c, type="pr
ob", probMethod = "softmax")
  pred = predict(plsprobs, newdata=clrTest[otus], type="prob")
  pred = as.data.frame(pred)
  colnames(pred) = c("healthy","ill")
  prediction = prediction(pred$ill,clrTest$disease)
  perf = performance(prediction,"auc")
  resultados = rbind(resultados, data.frame(importantOtus=7,ncomp=1,auc=as.numeric(pe
rf@y.values)))


resultados
```

```
##   importantOtus ncomp       auc
## 1             7     1 0.8333333
```

F1-score

We calculate the f1-score, so we have another value on which to evaluate the model. We see that the statistic is a little lower than the AUC, but nevertheless it is a very similar value.

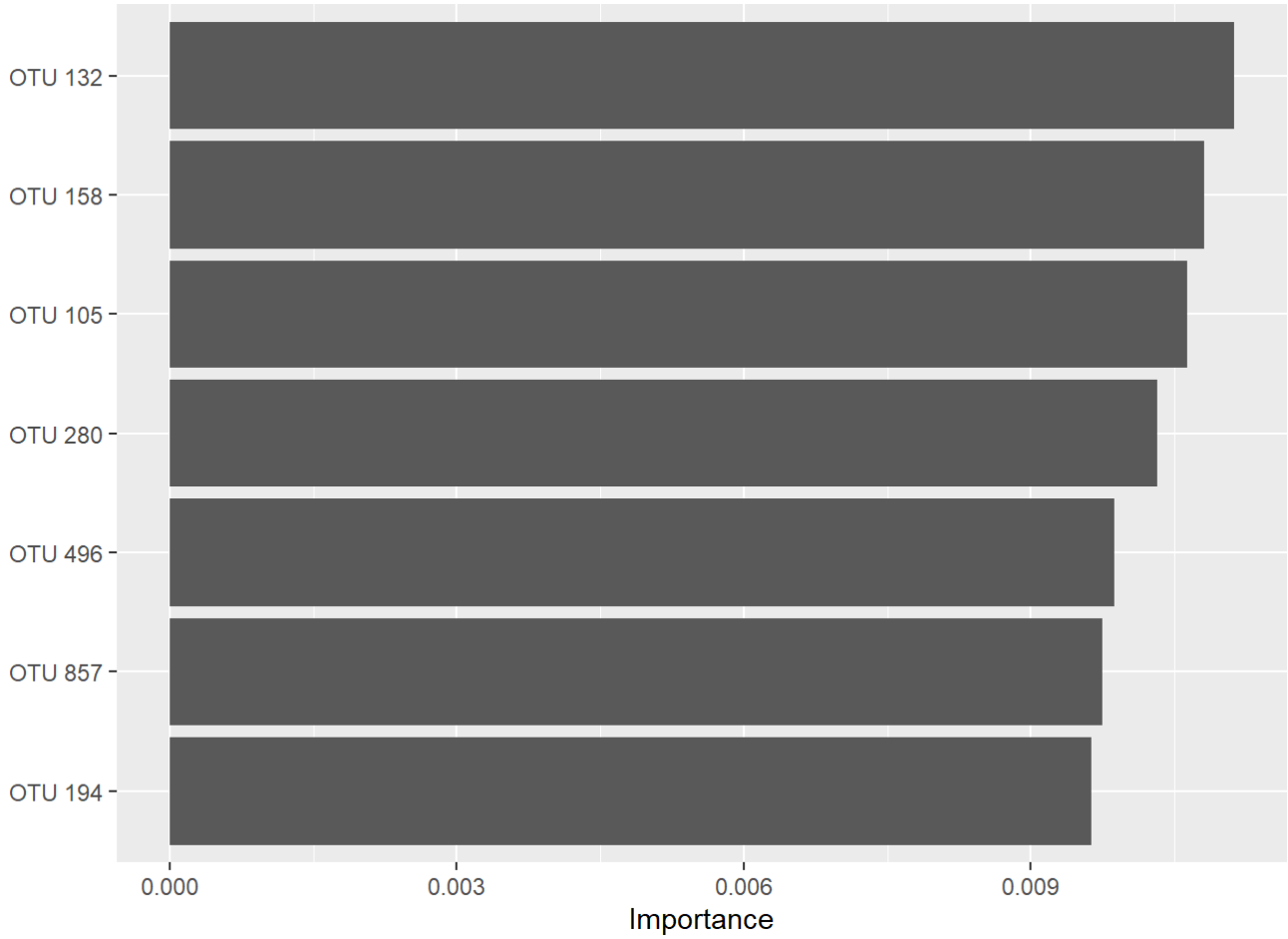
```
plsfinal = plsda(x=select(newdata,-disease), y=newdata$disease, ncomp = 1, type="clas
s", probMethod = "softmax")

pred2 = predict(plsfinal, newdata= clrTest[v[1:7]])
f1 = F1_Score(clrTest$disease, pred2)
f1
```

```
## [1] 0.7777778
```

VIP CHART.

This graph allows us to observe which variables have been selected for the model. By generating this type of graph, we will be able to observe whether the most important variables that make up the different models coincide.

```
plsprobs = plsda(x=select(clrTrain,-disease), y=clrTrain$disease, type="prob", probMe
thod = "softmax")
vip(plsprobs, num_features = 7)
```



Species

In this table we can observe the characteristics of the variables that appear in the model, in this case there are 7 variables. It should be noted that the predominant class is Bacteroidia, followed by Clostridia. On the other hand, we can see how the family, genus and species vary according to the OTU, with no repeated values.

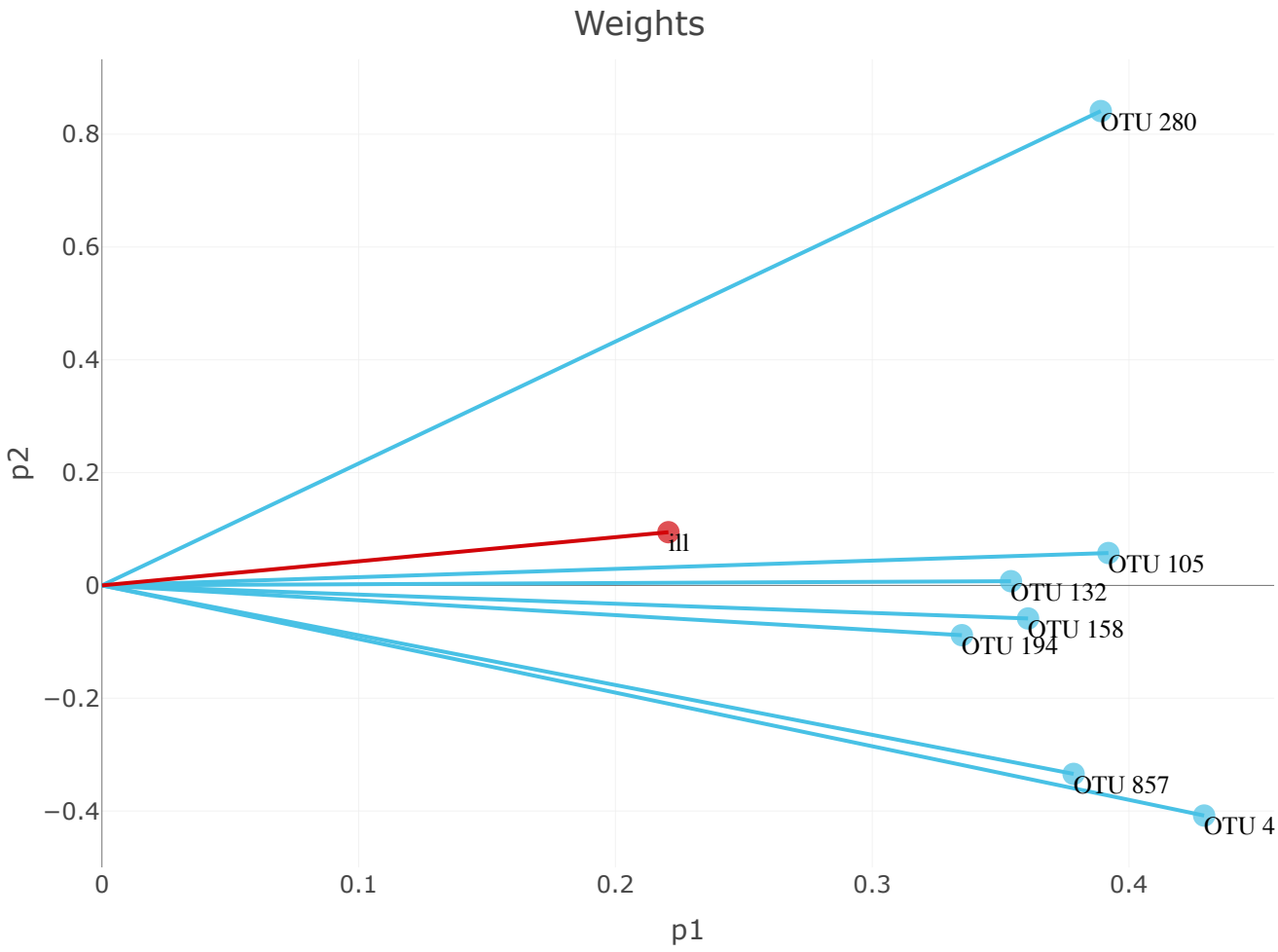
```
specis = species[v]
as.data.frame(specis)
```

```
##             King        Phylum         Class           Order
## OTU 132 Bacteria Bacteroidetes   Bacteroidia   Bacteroidales
## OTU 158 Bacteria Bacteroidetes   Bacteroidia   Bacteroidales
## OTU 105 Bacteria Bacteroidetes   Bacteroidia   Bacteroidales
## OTU 280 Bacteria    Firmicutes    Clostridia   Clostridiales
## OTU 496 Bacteria  Fusobacteria Fusobacteriia Fusobacteriales
## OTU 857 Bacteria    Firmicutes    Clostridia   Clostridiales
## OTU 194 Bacteria Bacteroidetes   Bacteroidia   Bacteroidales
##                         Family                Genus                      Species
## OTU 132  Bacteroidales_noname Bacteroidales_noname Bacteroidales_bacterium_ph8
## OTU 158    Porphyromonadaceae        Porphyromonas       Porphyromonas_somerae
## OTU 105        Bacteroidaceae          Bacteroides        Bacteroides_fragilis
## OTU 280        Clostridiaceae          Clostridium       Clostridium_symbiosum
## OTU 496      Fusobacteriaceae        Fusobacterium     Fusobacterium_nucleatum
## OTU 857 Peptostreptococcaceae   Peptostreptococcus Peptostreptococcus_stomatis
## OTU 194         Rikenellaceae            Alistipes             Alistipes_shahii
##         Type
## OTU 132 <NA>
## OTU 158 <NA>
## OTU 105 <NA>
## OTU 280 <NA>
## OTU 496 <NA>
## OTU 857 <NA>
## OTU 194 <NA>
```

Weights Chart.

To ensure better interpretability of the Weights chart we will use 2 components to perform the pls-da.

```
plsprobs = plsda(x=select(newdata,-disease), y=newdata$disease, ncomp = 2, type="pro
b", probMethod = "softmax")

library(plotly)

w = plsprobs$loading.weights
c = plsprobs$Yloading[2,]

data = rbind(w,c)
rownames(data) = c(rownames(w),"ill")
colnames(data) = c("p1","p2")
data = as.data.frame(data)
data = data %>% mutate(type = as.factor(c(rep(1,nrow(data)-1),2)))

t <- list(
  family = "sans serif",
  size = 13,
  color = toRGB("black"))

p = plot_ly(data, x=~p1, y=~p2, color=~type, colors = c("#48C1E5","#D20409"), showleg
end = F, text = rownames(data))
p = p %>% add_markers(size=0.5) %>% add_segments(x = 0, xend = ~p1, y = 0, yend = ~p
2)
p1 = p %>% add_text(textfont = t, textposition = "bottom right") %>% layout(title =
"Weights")
p1
```

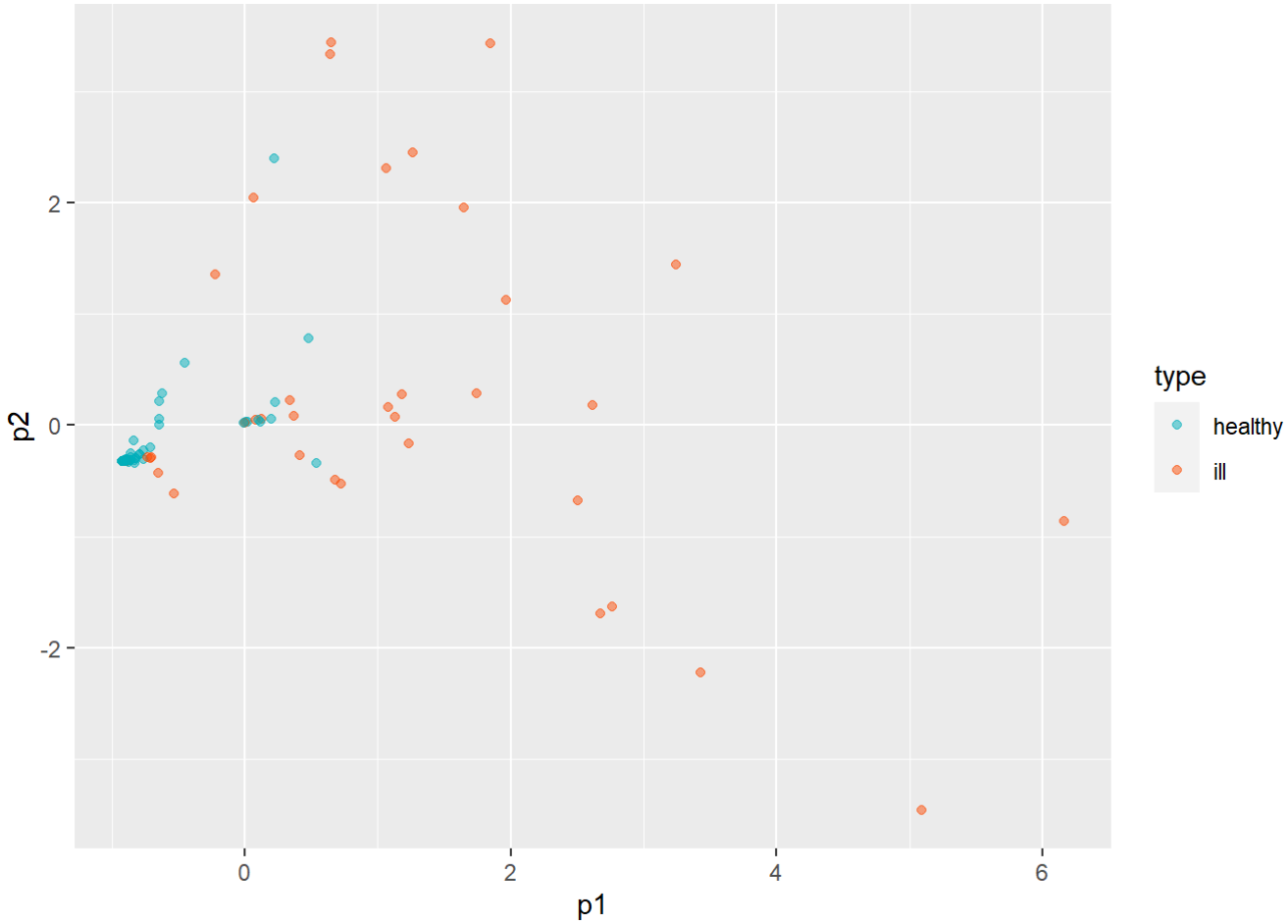


In this graph we can see which OTUS make the occurrence of the disease higher. That is, individuals with these organisms have a higher a priori probability of presenting the disease (cancer colorectal in this case).

We can clearly observe that there is one type of micro-organisms.The type of OTU that we see is the one that is positively related to the disease, i.e. the greater the quantity or presence of this microorganism, the more likely it is that the disease will develop or be developed. However, within this group we can observe that some OTUs are more positively correlated than others, such as OTUs 105, 132, 158 or 194, which form an angle very close to 0 with the variable 'ill'.

Scores Chart

```
scores = plsprobs$scores
scores = data.frame("p1"=scores[,1], "p2"=scores[,2], "type"=clrTrain$disease)
g1 = ggplot(scores, aes(p1,p2, color=type), colour = c("blue","red")) + geom_point(al
pha = I(0.5)) +
     scale_color_manual(values = c("#00AFBB", "#FC4E07"))
g1
```



We can see from the score graph how both sick (orange) and 'healthy' (blue) individuals are distributed. We can see from the graph that there is a degree of separation, the healthy individuals are in most cases those with negative values on p1. Whereas on p2 they are very evenly distributed around 0, unlike the individuals who have the disease. Those who are very dispersed in the graph.

S1_TSS

Train AND test

We separate the data into train and test, the percentage chosen for train is 80%, with this percentage we have 98 observations for train and 23 observations for test.

```
set.seed(280)
filas.tss = createDataPartition(tss$disease, p=0.8, list=FALSE)


tssTrain =tss[filas.tss,]
tssTest = tss[-filas.tss,]
dim(tssTrain)
```

```
## [1]  98 494
```

```
dim(tssTest)
```

```
## [1]  23 494
```

Pls-da

As with S1_TSSS, we first looked at the AUC of a pls model this is the one that gave us the best result. As can be seen, the AUC value is 0.73 . However, it is still a low value that can be increased.

```
set.seed(280)

library(vip)



plsprobs = plsda(x=select(tssTrain,-disease), y=tssTrain$disease, type="prob", probMe
thod = "softmax")

pred = predict(plsprobs, newdata=select(tssTest,-disease) , type="prob")

pred = as.data.frame(pred)
colnames(pred) = c("healthy","ill")
prediction = prediction(pred$ill,tssTest$disease)
perf = performance(prediction,"auc")
as.numeric(perf@y.values)
```

```
## [1] 0.7380952
```

Pls-da according to VIP

We ran our code to try to find a better model.

```
set.seed(280)

library(vip)

resultados = data.frame()

plsprobs = plsda(x=select(tssTrain,-disease), y=tssTrain$disease, type="prob", probMe
thod = "softmax")

g = vip(plsprobs, num_features = 35)
v = g$data$Variable

for (i in 2:length(v)) {
  print(i)
  otus = v[1:i]
  newdata = as.data.frame(tssTrain[otus])
  newdata$disease = tssTrain$disease

  #we train the models with cv with the selected variables based on the auc and take
 the best one.
  tr_fit = trainControl(method= 'repeatedcv',number = 10, repeats = 10, classProbs =
TRUE,summaryFunction = twoClassSummary)
  model = train(disease~., data=newdata, trControl = tr_fit,method = 'pls',metric =
'ROC')
  c = model$results$ncomp[which(model$results$ROC==max(model$results$ROC))]


  #We train the one that gives us the best result with the whole training set and pre
dict the test set. We keep the results obtained
  plsprobs = plsda(x=select(newdata,-disease), y=newdata$disease, ncomp = c, type="pr
ob", probMethod = "softmax")
  pred = predict(plsprobs, newdata=tssTest[otus], type="prob")
  pred = as.data.frame(pred)
  colnames(pred) = c("healthy","ill")
  prediction = prediction(pred$ill,tssTest$disease)
  perf = performance(prediction,"auc")
  resultados = rbind(resultados, data.frame(importantOtus=i,ncomp=c,auc=as.numeric(pe
rf@y.values)))


}
```

```
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
## [1] 11
## [1] 12
## [1] 13
## [1] 14
## [1] 15
## [1] 16
## [1] 17
## [1] 18
## [1] 19
## [1] 20
## [1] 21
## [1] 22
## [1] 23
## [1] 24
## [1] 25
## [1] 26
## [1] 27
## [1] 28
## [1] 29
## [1] 30
## [1] 31
## [1] 32
## [1] 33
## [1] 34
## [1] 35
```

```
print(resultados)
```

```
##    importantOtus ncomp       auc
## 1              2     1 0.5198413
## 2              3     1 0.5515873
## 3              4     3 0.6547619
## 4              5     2 0.6746032
## 5              6     1 0.7936508
## 6              7     1 0.7539683
## 7              8     1 0.7936508
## 8              9     1 0.7619048
## 9             10     1 0.6904762
## 10            11     1 0.6904762
## 11            12     1 0.7777778
## 12            13     1 0.7777778
## 13            14     1 0.7301587
## 14            15     1 0.7698413
## 15            16     2 0.6984127
## 16            17     2 0.7142857
## 17            18     1 0.7777778
## 18            19     1 0.7777778
## 19            20     1 0.7777778
## 20            21     1 0.7777778
## 21            22     1 0.7142857
## 22            23     1 0.7380952
## 23            24     1 0.7301587
## 24            25     1 0.7301587
## 25            26     1 0.7142857
## 26            27     1 0.7142857
## 27            28     1 0.7142857
## 28            29     1 0.6984127
## 29            30     1 0.6984127
## 30            31     1 0.6984127
## 31            32     1 0.6984127
## 32            33     1 0.6984127
## 33            34     1 0.6984127
## 34            35     1 0.6984127
```

After the execution of our code, we observe that we are able to increase the AUC 0.10. In this case, we see how the best model is with 1 components and 21 variables. The AUC that we obtained is 0.7936.

```
set.seed(280)

library(vip)

resultados = data.frame()

plsprobs = plsda(x=select(tssTrain,-disease), y=tssTrain$disease, type="prob", probMe
thod = "softmax")

g = vip(plsprobs, num_features = 6)
v = g$data$Variable

  otus = v[1:6]
  newdata = as.data.frame(tssTrain[otus])
  newdata$disease = tssTrain$disease



  tr_fit = trainControl(method= 'repeatedcv',number = 10, repeats = 10, classProbs =
TRUE,summaryFunction = twoClassSummary)
  model = train(disease~., data=newdata, trControl = tr_fit,method = 'pls',metric =
'ROC')
  plsprobs = plsda(x=select(newdata,-disease), y=newdata$disease, ncomp = 1, type="pr
ob", probMethod = "softmax")
  pred = predict(plsprobs, newdata=tssTest[otus], type="prob")
  pred = as.data.frame(pred)
  colnames(pred) = c("healthy","ill")
  prediction = prediction(pred$ill,tssTest$disease)
  perf = performance(prediction,"auc")
  resultados = rbind(resultados, data.frame(importantOtus=6,ncomp=1,auc=as.numeric(pe
rf@y.values)))



resultados
```

```
##   importantOtus ncomp       auc
## 1             6     1 0.7936508
```

F1-score

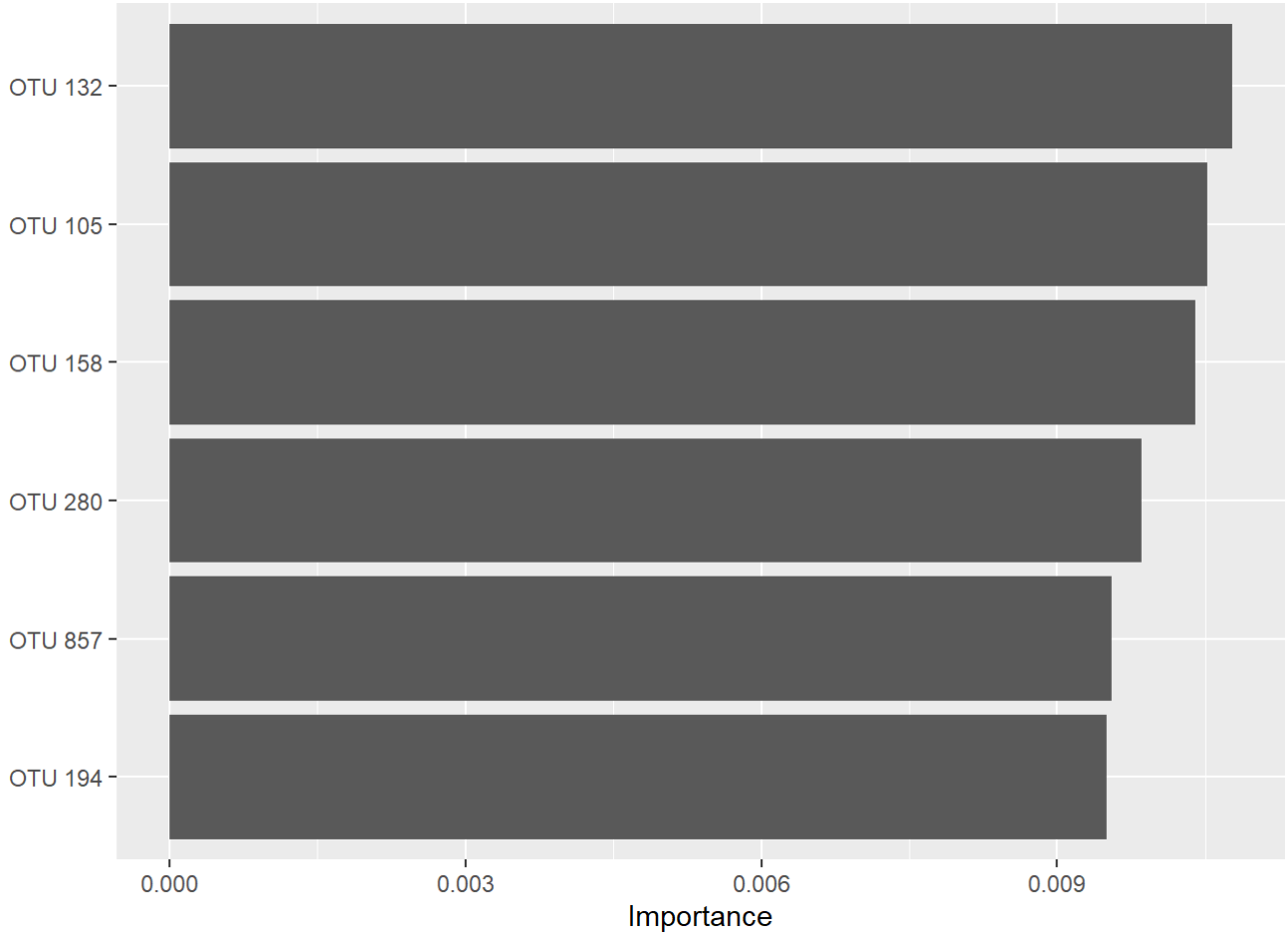
```
plsfinal = plsda(x=select(newdata,-disease), y= newdata$disease, ncomp = 1, type="cla
ss", probMethod = "softmax")

pred2 = predict(plsfinal, newdata= tssTest[v[1:6]])
f1 = F1_Score(tssTest$disease, pred2)
f1
```

```
## [1] 0.8
```

VIP CHART.

This graph allows us to observe which variables have been selected for the model. By generating this type of graph, we will be able to observe whether the most important variables that make up the different models coincide.

```
plsprobs = plsda(x=select(tssTrain,-disease), y=tssTrain$disease, type="prob", probMe
thod = "softmax")
vip(plsprobs, num_features = 6)
```



Species

In this table we can observe the characteristics of the variables that appear in the model, in this case there are 6 variables. It is worth noting that the predominant class is Bacteroidia, followed by Clostridia. On the other hand, we can see how the family, genus and species vary according to the OTU, with no repeated values. We can also see how the variables coincide with the variables of the best S1-CLR model.R.

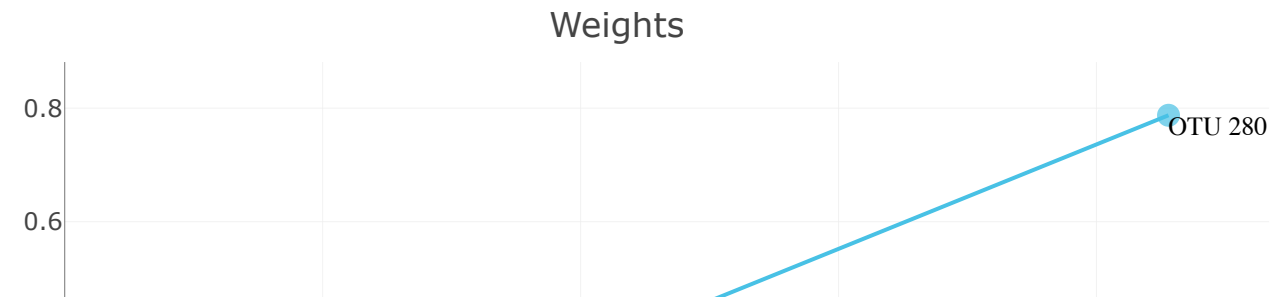
```
specis = species[v]
as.data.frame(specis)
```

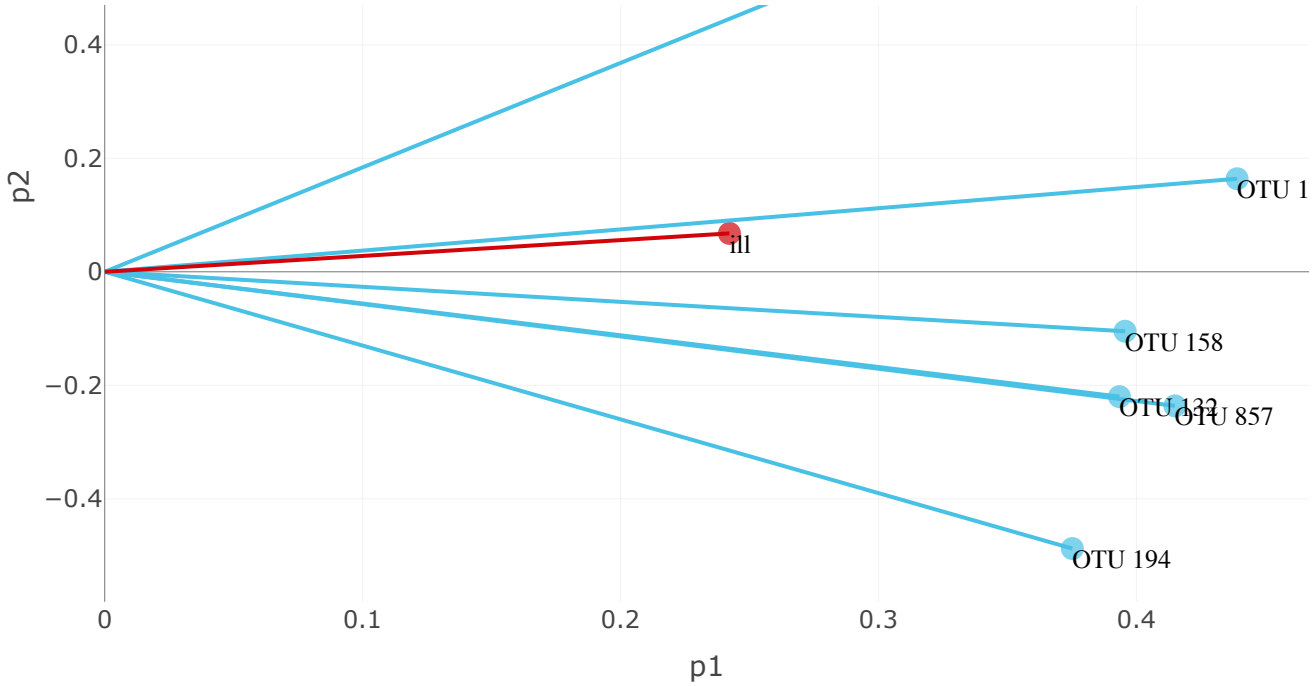
```
##             King        Phylum       Class         Order                 Family
## OTU 132 Bacteria Bacteroidetes Bacteroidia Bacteroidales  Bacteroidales_noname
## OTU 105 Bacteria Bacteroidetes Bacteroidia Bacteroidales        Bacteroidaceae
## OTU 158 Bacteria Bacteroidetes Bacteroidia Bacteroidales    Porphyromonadaceae
## OTU 280 Bacteria    Firmicutes  Clostridia Clostridiales        Clostridiaceae
## OTU 857 Bacteria    Firmicutes  Clostridia Clostridiales Peptostreptococcaceae
## OTU 194 Bacteria Bacteroidetes Bacteroidia Bacteroidales         Rikenellaceae
##                        Genus                     Species Type
## OTU 132 Bacteroidales_noname Bacteroidales_bacterium_ph8 <NA>
## OTU 105          Bacteroides        Bacteroides_fragilis <NA>
## OTU 158        Porphyromonas       Porphyromonas_somerae <NA>
## OTU 280          Clostridium       Clostridium_symbiosum <NA>
## OTU 857   Peptostreptococcus Peptostreptococcus_stomatis <NA>
## OTU 194            Alistipes            Alistipes_shahii <NA>
```

Weights Chart

As in the previous case, we performed the pls-da with 2 components to make the Weights and scores graphs.

```
plsprobs = plsda(x=select(newdata,-disease), y= newdata$disease, ncomp = 2, type="cla
ss", probMethod = "softmax")


library(plotly)

w = plsprobs$loading.weights
c = plsprobs$Yloading[2,]

data = rbind(w,c)
rownames(data) = c(rownames(w),"ill")
colnames(data) = c("p1","p2")
data = as.data.frame(data)
data = data %>% mutate(type = as.factor(c(rep(1,nrow(data)-1),2)))

t <- list(
  family = "sans serif",
  size = 13,
  color = toRGB("black"))

p = plot_ly(data, x=~p1, y=~p2, color=~type, colors = c("#48C1E5","#D20409"), showleg
end = F, text = rownames(data))
p = p %>% add_markers(size=0.5) %>% add_segments(x = 0, xend = ~p1, y = 0, yend = ~p
2)
p1 = p %>% add_text(textfont = t, textposition = "bottom right") %>% layout(title =
"Weights")
p1
```





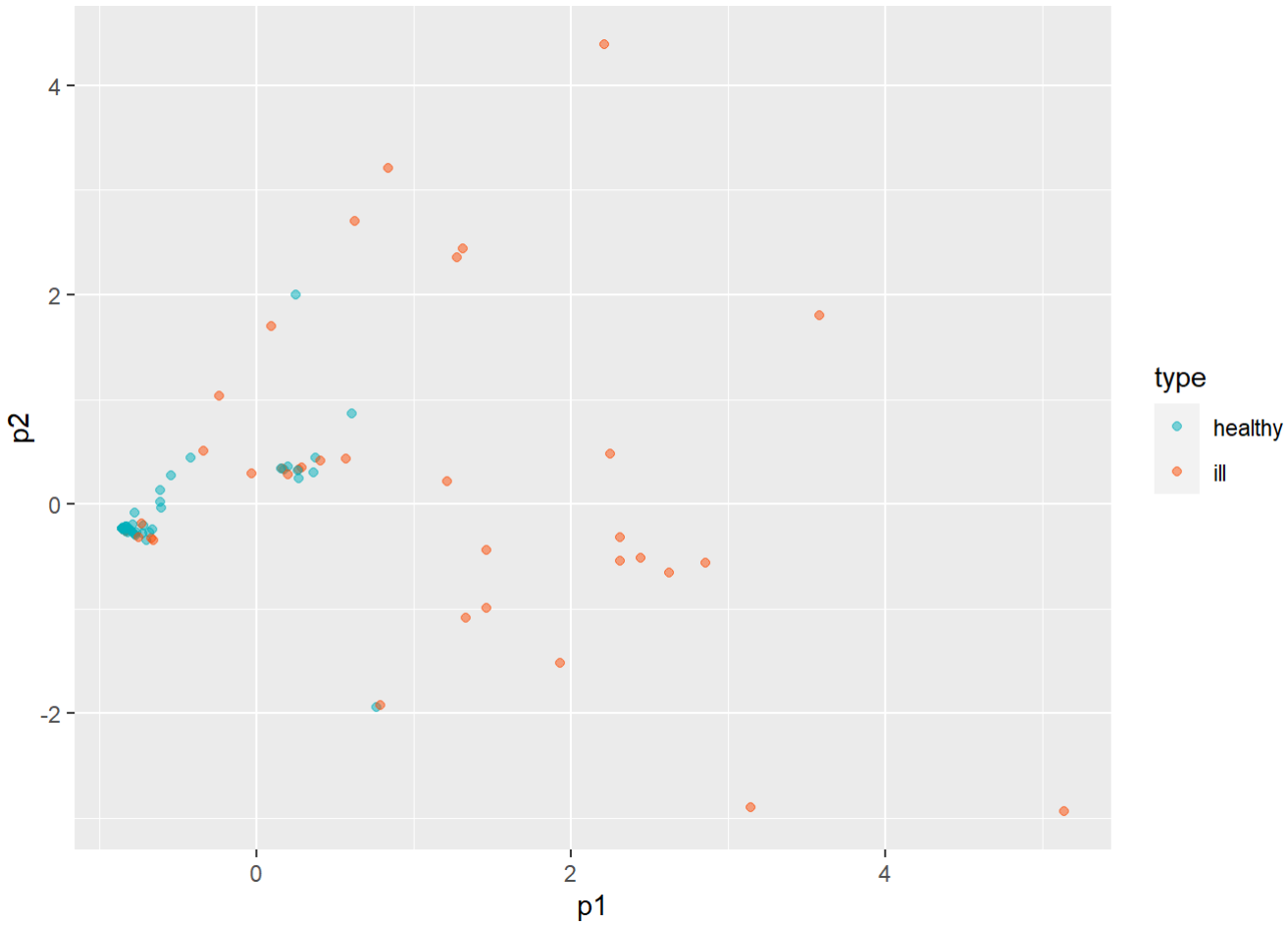
In this graph we can see which OTUS make the occurrence of the disease higher. That is, individuals with these organisms have a higher a priori probability of presenting the disease.

We can clearly observe that there is one type of micro-organisms.

The type of OTU that we see is the one that is positively related to the disease, i.e. the greater the quantity or presence of this microorganism, the more likely it is that the disease will develop. However, within this group we can see that some OTUS are more positively correlated than others, such as OTUS 105 or 158, which form an angle very close to 0 with the variable ‘ill’. This graph is very similar to the graph observed in the s1-clr model.

Scores Chart.

```
scores = plsprobs$scores
scores = data.frame("p1"=scores[,1], "p2"=scores[,2], "type"=tssTrain$disease)
g1 = ggplot(scores, aes(p1,p2, color=type), colour = c("blue","red")) + geom_point(al
pha = I(0.5)) +
     scale_color_manual(values = c("#00AFBB", "#FC4E07"))
g1
```



We can see from the score graph how both sick (orange) and ‘healthy’ (blue) individuals are distributed. It can be seen from the graph that there is a degree of separation, the healthy individuals are in most cases those with negative values on p1. Whereas on p2 they are very evenly distributed around 0, unlike the individuals who have the disease. These are widely dispersed in the graph.

As with the weights graph, the S1-CLR and S1-TSS score graphs are similar in many respects, with a very similar distribution.

Loading the S2 dataset

We import the data. As can be seen, we centre and scale the data to later carry out the PLS-DA and we also carry out a transformation on the pre-processed TSS data, this transformation allows us a better interpretation of the data.

S2_CLR

```
sample = as.data.frame(Datos_sample$CancerColorectal)
a = factor(sample$group, levels=c("control","crc"),labels=c("healthy","ill"))

tss = as.data.frame(log(as.data.frame(t(DatosFinal$CancerColorectal$S2_NORMAL$S2_TSS_
CANCERCOL)) * 10e6 + 1))
clr = as.data.frame(t(DatosFinal$CancerColorectal$S2_NORMAL$S2_CLR_CANCERCOL))

tss = as.data.frame(scale(tss))
clr = as.data.frame(scale(clr))

clr$disease = a
tss$disease = a
```

Train AND test

We separate the data into train and test, the percentage chosen for train is 80%, with this percentage we have 98 observations for train and 23 observations for test.

```
set.seed(280)
filas.clr = createDataPartition(clr$disease, p=0.8, list=FALSE)
clrTrain = clr[filas.clr,]
clrTest = clr[-filas.clr,]
dim(clrTrain)
```

```
## [1]  98 185
```

```
dim(clrTest)
```

```
## [1]  23 185
```

Pls-da

The first PLS we performed on the S1 dataset with the CLR pre-processing obtained an AUC of 0.73, this PLS was performed with all the variables of the model and without previous cross-validation.

```
set.seed(280)

library(vip)



plsprobs = plsda(x=select(clrTrain,-disease), y=clrTrain$disease, type="prob", probMe
thod = "softmax")

pred = predict(plsprobs, newdata=select(clrTest,-disease) , type="prob")

pred = as.data.frame(pred)
colnames(pred) = c("healthy","ill")
prediction = prediction(pred$ill,clrTest$disease)
perf = performance(prediction,"auc")
as.numeric(perf@y.values)
```

```
## [1] 0.7380952
```

Pls-da according VIP.

We run the loop to try to find the best model.

```
set.seed(280)

library(vip)

resultados = data.frame()

plsprobs = plsda(x=select(clrTrain,-disease), y=clrTrain$disease, type="prob", probMe
thod = "softmax")

g = vip(plsprobs, num_features = 25)
v = g$data$Variable

for (i in 2:length(v)) {
  print(i)
  otus = v[1:i]
  newdata = as.data.frame(clrTrain[otus])
  newdata$disease = clrTrain$disease

  #we train the models with cv with the selected variables based on the auc and take
 the best one.
  tr_fit = trainControl(method= 'repeatedcv',number = 10, repeats = 10, classProbs =
TRUE,summaryFunction = twoClassSummary)
  model = train(disease~., data=newdata, trControl = tr_fit,method = 'pls',metric =
'ROC')
  c = model$results$ncomp[which(model$results$ROC==max(model$results$ROC))]



  #We train the one that gives us the best result with the whole training set and pre
dict the test set. We keep the results obtained
  plsprobs = plsda(x=select(newdata,-disease), y=newdata$disease, ncomp = c, type="pr
ob", probMethod = "softmax")
  pred = predict(plsprobs, newdata=clrTest[otus], type="prob")
  pred = as.data.frame(pred)
  colnames(pred) = c("healthy","ill")
  prediction = prediction(pred$ill,clrTest$disease)
  perf = performance(prediction,"auc")
  resultados = rbind(resultados, data.frame(importantOtus=i,ncomp=c,auc=as.numeric(pe
rf@y.values)))



}
```

```
## [1] 2
## [1] 3
## [1] 4
```

```
## Warning in plsda.default(x = select(newdata, -disease), y = newdata$disease, :
## A value single ncomp must be specified. max(ncomp) was used. Predictions can be
## obtained for values <= ncomp
```

```
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
## [1] 11
## [1] 12
## [1] 13
## [1] 14
## [1] 15
## [1] 16
## [1] 17
## [1] 18
## [1] 19
## [1] 20
## [1] 21
## [1] 22
## [1] 23
## [1] 24
## [1] 25
```

```
print(resultados)
```

```
##    importantOtus ncomp       auc
## 1              2     1 0.6269841
## 2              3     2 0.6468254
## 3              4     2 0.6587302
## 4              4     3 0.6587302
## 5              5     2 0.6587302
## 6              6     1 0.7619048
## 7              7     1 0.7619048
## 8              8     2 0.8650794
## 9              9     2 0.8015873
## 10            10     2 0.7857143
## 11            11     1 0.6984127
## 12            12     1 0.7777778
## 13            13     1 0.7460317
## 14            14     1 0.7857143
## 15            15     1 0.7777778
## 16            16     1 0.7936508
## 17            17     1 0.8015873
## 18            18     1 0.7857143
## 19            19     1 0.7698413
## 20            20     1 0.7698413
## 21            21     1 0.7698413
## 22            22     1 0.7857143
## 23            23     1 0.8333333
## 24            24     1 0.7698413
## 25            25     1 0.7936508
```

With the S2 dataset and the CLR preprocessing we found the best model, with an AUC of 0.86 with 2 components and only 8 variables.

```
set.seed(280)

library(vip)

resultados = data.frame()

plsprobs = plsda(x=select(clrTrain,-disease), y=clrTrain$disease, type="prob", probMe
thod = "softmax")

g = vip(plsprobs, num_features = 8)
v = g$data$Variable

  otus = v[1:8]
  newdata = as.data.frame(clrTrain[otus])
  newdata$disease = clrTrain$disease



  tr_fit = trainControl(method= 'repeatedcv',number = 10, repeats = 10, classProbs =
TRUE,summaryFunction = twoClassSummary)
  model = train(disease~., data=newdata, trControl = tr_fit,method = 'pls',metric =
'ROC')
  plsprobs = plsda(x=select(newdata,-disease), y=newdata$disease, ncomp = 2, type="pr
ob", probMethod = "softmax")
  pred = predict(plsprobs, newdata=clrTest[otus], type="prob")
  pred = as.data.frame(pred)
  colnames(pred) = c("healthy","ill")
  prediction = prediction(pred$ill,clrTest$disease)
  perf = performance(prediction,"auc")
  resultados = rbind(resultados, data.frame(importantOtus=8,ncomp=2,auc=as.numeric(pe
rf@y.values)))



resultados
```

```
##   importantOtus ncomp       auc
## 1             8     2 0.8650794
```

F1-score.

```
plsfinal = plsda(x=select(newdata,-disease), y=newdata$disease, ncomp = 2, type="clas
s", probMethod = "softmax")

pred2 = predict(plsfinal, newdata= clrTest[v[1:8]])
F1_Score(clrTest$disease, pred2)
```

```
## [1] 0.8235294
```

Species

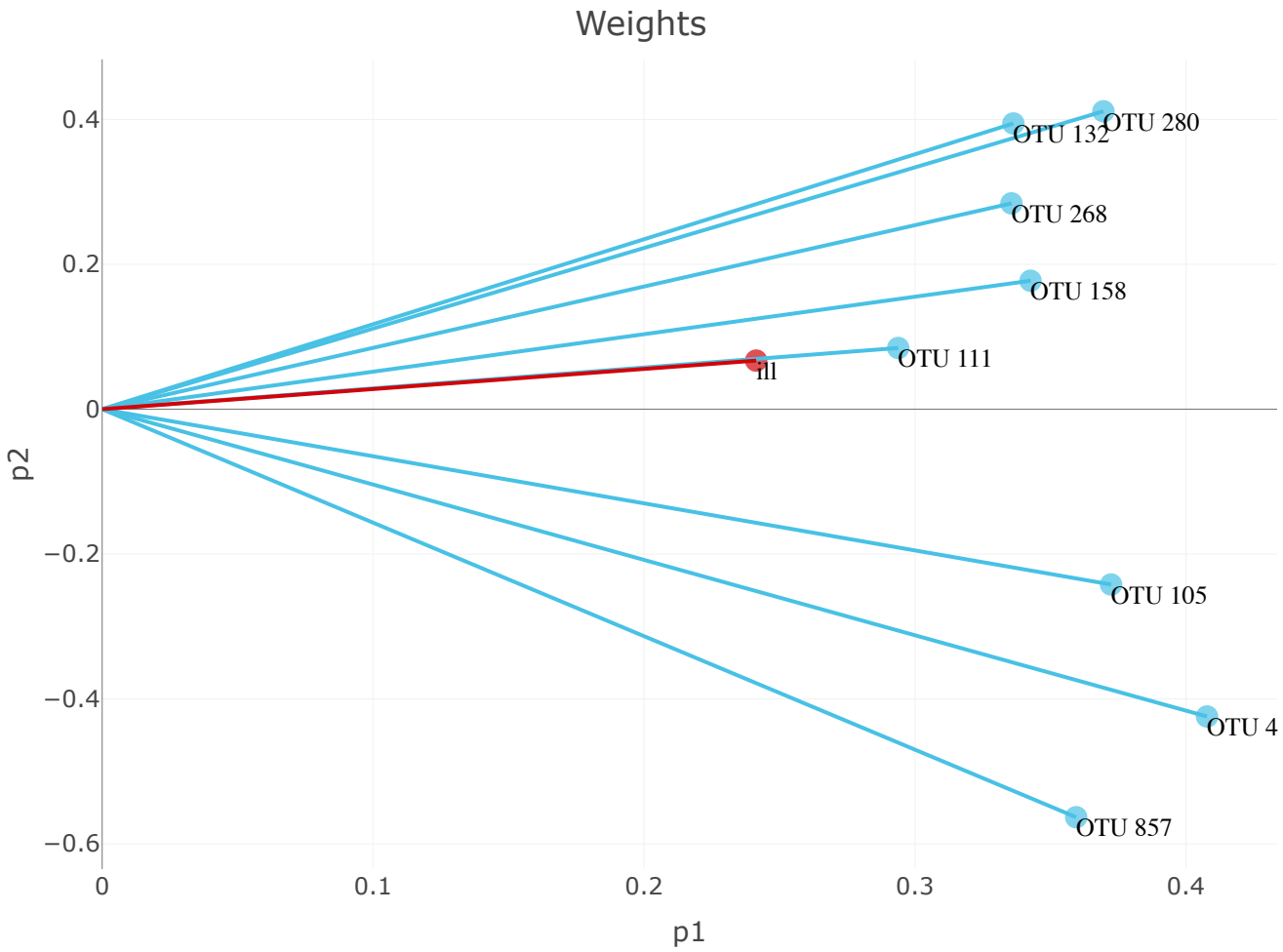
In this table we can observe the characteristics of the variables that appear in the model, in this case there are 8 variables. It is worth noting that the predominant class is Bacteroidia, followed by Clostridia. In this case there are 4 OTUs of the Bacteroidia class and 3 of Clostridia, and finally there is also an OTU of the Fusobacteriia class.

```
specis = as.data.frame(species[v])
specis
```

```
##             King        Phylum         Class           Order
## OTU 105 Bacteria Bacteroidetes   Bacteroidia   Bacteroidales
## OTU 280 Bacteria    Firmicutes    Clostridia   Clostridiales
## OTU 857 Bacteria    Firmicutes    Clostridia   Clostridiales
## OTU 132 Bacteria Bacteroidetes   Bacteroidia   Bacteroidales
## OTU 158 Bacteria Bacteroidetes   Bacteroidia   Bacteroidales
## OTU 496 Bacteria  Fusobacteria Fusobacteriia Fusobacteriales
## OTU 268 Bacteria    Firmicutes    Clostridia   Clostridiales
## OTU 111 Bacteria Bacteroidetes   Bacteroidia   Bacteroidales
##                        Family               Genus                      Species
## OTU 105        Bacteroidaceae          Bacteroides        Bacteroides_fragilis
## OTU 280        Clostridiaceae          Clostridium       Clostridium_symbiosum
## OTU 857 Peptostreptococcaceae   Peptostreptococcus Peptostreptococcus_stomatis
## OTU 132  Bacteroidales_noname Bacteroidales_noname Bacteroidales_bacterium_ph8
## OTU 158    Porphyromonadaceae        Porphyromonas       Porphyromonas_somerae
## OTU 496      Fusobacteriaceae        Fusobacterium     Fusobacterium_nucleatum
## OTU 268        Clostridiaceae          Clostridium       Clostridium_hathewayi
## OTU 111        Bacteroidaceae          Bacteroides          Bacteroides_nordii
##         Type
## OTU 105 <NA>
## OTU 280 <NA>
## OTU 857 <NA>
## OTU 132 <NA>
## OTU 158 <NA>
## OTU 496 <NA>
## OTU 268 <NA>
## OTU 111 <NA>
```

Weights Chart

```
library(plotly)

w = plsprobs$loading.weights
c = plsprobs$Yloading[2,]

data = rbind(w,c)
rownames(data) = c(rownames(w),"ill")
colnames(data) = c("p1","p2")
data = as.data.frame(data)
data = data %>% mutate(type = as.factor(c(rep(1,nrow(data)-1),2)))

t <- list(
  family = "sans serif",
  size = 13,
  color = toRGB("black"))

p = plot_ly(data, x=~p1, y=~p2, color=~type, colors = c("#48C1E5","#D20409"), showleg
end = F, text = rownames(data))
p = p %>% add_markers(size=0.5) %>% add_segments(x = 0, xend = ~p1, y = 0, yend = ~p
2)
p1 = p %>% add_text(textfont = t, textposition = "bottom right") %>% layout(title =
"Weights")
p1
```

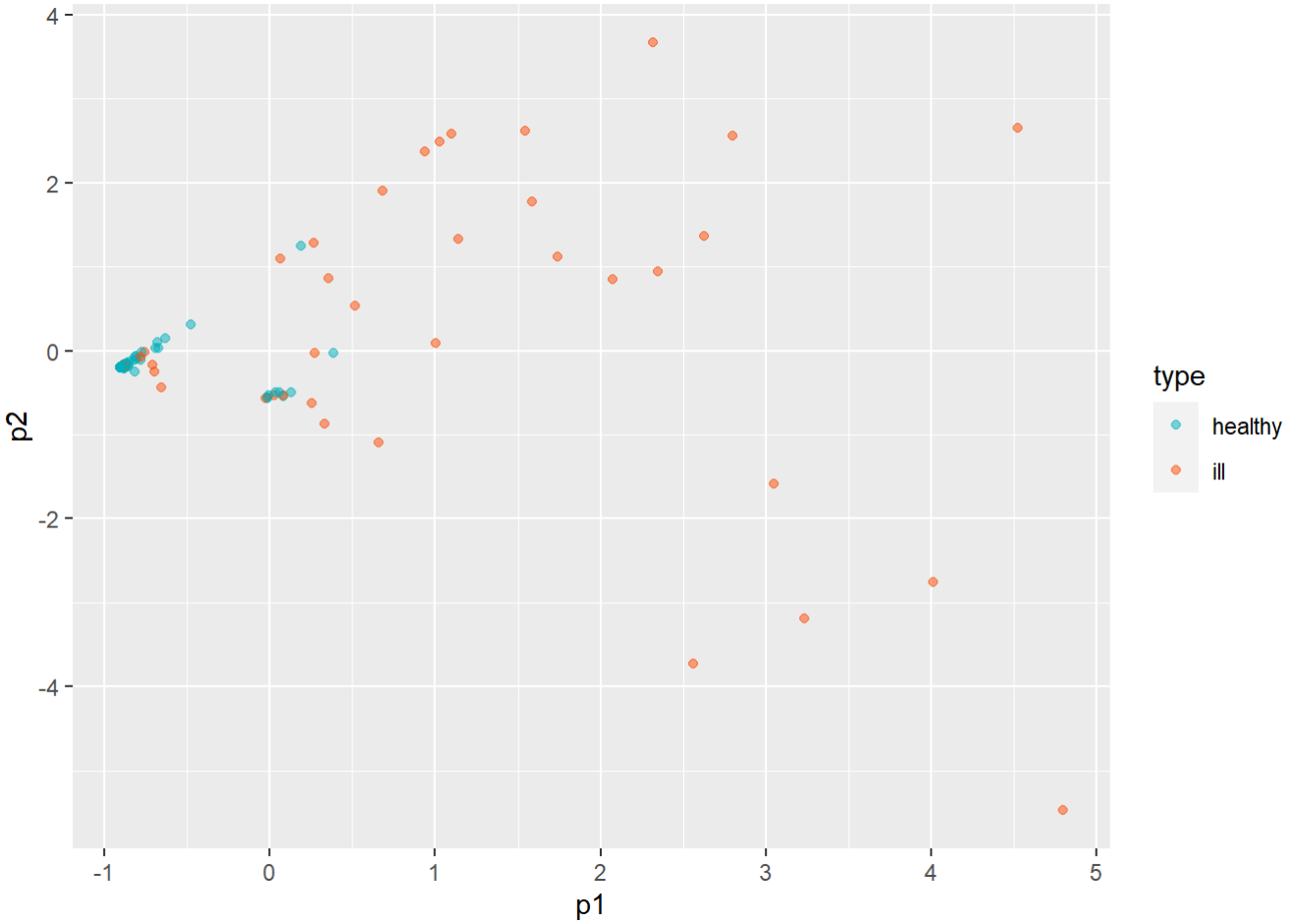


We can clearly observe that there is one type of micro-organisms.The type of OTU that we see is the one that is positively related to the disease, which means that a greater quantity or presence of this microorganism makes it more likely that the disease will develop. However, within this group we can observe that some OTUs

are more positively correlated than others, such as OTUs 158 or 111, which form an angle very close to 0 with the variable 'ill'.

Scores Chart.

```
scores = plsprobs$scores
scores = data.frame("p1"=scores[,1], "p2"=scores[,2], "type"=clrTrain$disease)
g1 = ggplot(scores, aes(p1,p2, color=type), colour = c("blue","red")) + geom_point(al
pha = I(0.5)) +
     scale_color_manual(values = c("#00AFBB", "#FC4E07"))
g1
```



We can see from the score plot how both sick (orange) and 'healthy' (blue) individuals are distributed. In this particular case, we can see how the different individuals are distributed in components 1 and 2 of the pls-da.

Although the individuals are mixed, we can see how they differ in different aspects. Most of the healthy individuals are found in negative values with respect to component p1, and with respect to component p2 they are observed agglutinated in values very close to 0. On the other hand, the values that refer to the sick individuals are very distributed throughout the graph. Of these individuals, it should be noted that the vast majority are concentrated in high values for p2.

S2_TSS

Train AND test

We separate the data into train and test, the percentage chosen for train is 80%, with this percentage we have 98 observations for train and 23 observations for test.

```
set.seed(280)
filas.tss = createDataPartition(tss$disease, p=0.8, list=FALSE)

tssTrain =tss[filas.tss,]
tssTest = tss[-filas.tss,]
dim(tssTrain)
```

```
## [1]  98 185
```

```
dim(tssTest)
```

```
## [1]  23 185
```

Pls-da

As with S2_TSSS, we first looked at the AUC of a pls model this is the one that gave us the best result. As can be seen, the AUC value is 0.75 . However, it is still a low value that can be increased.

```
set.seed(280)

library(vip)



plsprobs = plsda(x=select(tssTrain,-disease), y=tssTrain$disease, type="prob", probMe
thod = "softmax")

pred = predict(plsprobs, newdata=select(tssTest,-disease) , type="prob")

pred = as.data.frame(pred)
colnames(pred) = c("healthy","ill")
prediction = prediction(pred$ill,tssTest$disease)
perf = performance(prediction,"auc")
as.numeric(perf@y.values)
```

```
## [1] 0.7539683
```

Pls-da according VIP.

We tested our code to find a better model.

```
set.seed(280)

library(vip)

resultados = data.frame()

plsprobs = plsda(x=select(tssTrain,-disease), y=tssTrain$disease, type="prob", probMe
thod = "softmax")

g = vip(plsprobs, num_features = 35)
v = g$data$Variable

for (i in 2:length(v)) {
  print(i)
  otus = v[1:i]
  newdata = as.data.frame(tssTrain[otus])
  newdata$disease = clrTrain$disease

  #we train the models with cv with the selected variables based on the auc and take
 the best one.
  tr_fit = trainControl(method= 'repeatedcv',number = 10, repeats = 10, classProbs =
TRUE,summaryFunction = twoClassSummary)
  model = train(disease~., data=newdata, trControl = tr_fit,method = 'pls',metric =
'ROC')
  c = model$results$ncomp[which(model$results$ROC==max(model$results$ROC))]



  #We train the one that gives us the best result with the whole training set and pre
dict the test set. We keep the results obtained
  plsprobs = plsda(x=select(newdata,-disease), y=newdata$disease, ncomp = c, type="pr
ob", probMethod = "softmax")
  pred = predict(plsprobs, newdata=tssTest[otus], type="prob")
  pred = as.data.frame(pred)
  colnames(pred) = c("healthy","ill")
  prediction = prediction(pred$ill,tssTest$disease)
  perf = performance(prediction,"auc")
  resultados = rbind(resultados, data.frame(importantOtus=i,ncomp=c,auc=as.numeric(pe
rf@y.values)))


}
```

```
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
## [1] 11
## [1] 12
## [1] 13
## [1] 14
## [1] 15
## [1] 16
## [1] 17
## [1] 18
## [1] 19
## [1] 20
## [1] 21
## [1] 22
## [1] 23
## [1] 24
## [1] 25
## [1] 26
## [1] 27
## [1] 28
## [1] 29
## [1] 30
## [1] 31
## [1] 32
## [1] 33
## [1] 34
## [1] 35
```

```
print(resultados)
```

```
##    importantOtus ncomp       auc
## 1              2     1 0.6269841
## 2              3     2 0.6547619
## 3              4     2 0.6746032
## 4              5     3 0.6666667
## 5              6     1 0.7539683
## 6              7     1 0.7619048
## 7              8     2 0.6746032
## 8              9     2 0.7777778
## 9             10     1 0.7777778
## 10            11     1 0.6904762
## 11            12     1 0.7698413
## 12            13     1 0.7460317
## 13            14     1 0.7777778
## 14            15     1 0.7857143
## 15            16     1 0.7857143
## 16            17     1 0.7857143
## 17            18     1 0.7857143
## 18            19     2 0.7460317
## 19            20     1 0.8253968
## 20            21     1 0.8253968
## 21            22     1 0.8253968
## 22            23     1 0.8253968
## 23            24     1 0.7698413
## 24            25     1 0.7936508
## 25            26     1 0.8015873
## 26            27     1 0.8015873
## 27            28     1 0.7539683
## 28            29     1 0.7857143
## 29            30     1 0.7857143
## 30            31     1 0.7698413
## 31            32     1 0.7460317
## 32            33     1 0.7619048
## 33            34     1 0.7857143
## 34            35     1 0.7857143
```

```
set.seed(280)

library(vip)

resultados = data.frame()

plsprobs = plsda(x=select(tssTrain,-disease), y=tssTrain$disease, type="prob", probMe
thod = "softmax")

g = vip(plsprobs, num_features = 20)
v = g$data$Variable

  otus = v[1:20]
  newdata = as.data.frame(tssTrain[otus])
  newdata$disease = tssTrain$disease


  tr_fit = trainControl(method= 'repeatedcv',number = 10, repeats = 10, classProbs =
TRUE,summaryFunction = twoClassSummary)
  model = train(disease~., data=newdata, trControl = tr_fit,method = 'pls',metric =
'ROC')
  plsprobs = plsda(x=select(newdata,-disease), y=newdata$disease, ncomp = 1, type="pr
ob", probMethod = "softmax")
  pred = predict(plsprobs, newdata=tssTest[otus], type="prob")
  pred = as.data.frame(pred)
  colnames(pred) = c("healthy","ill")
  prediction = prediction(pred$ill,tssTest$disease)
  perf = performance(prediction,"auc")
  resultados = rbind(resultados, data.frame(importantOtus=20,ncomp=1,auc=as.numeric(p
erf@y.values)))


resultados
```

```
##   importantOtus ncomp       auc
## 1            20     1 0.8253968
```

F1-score

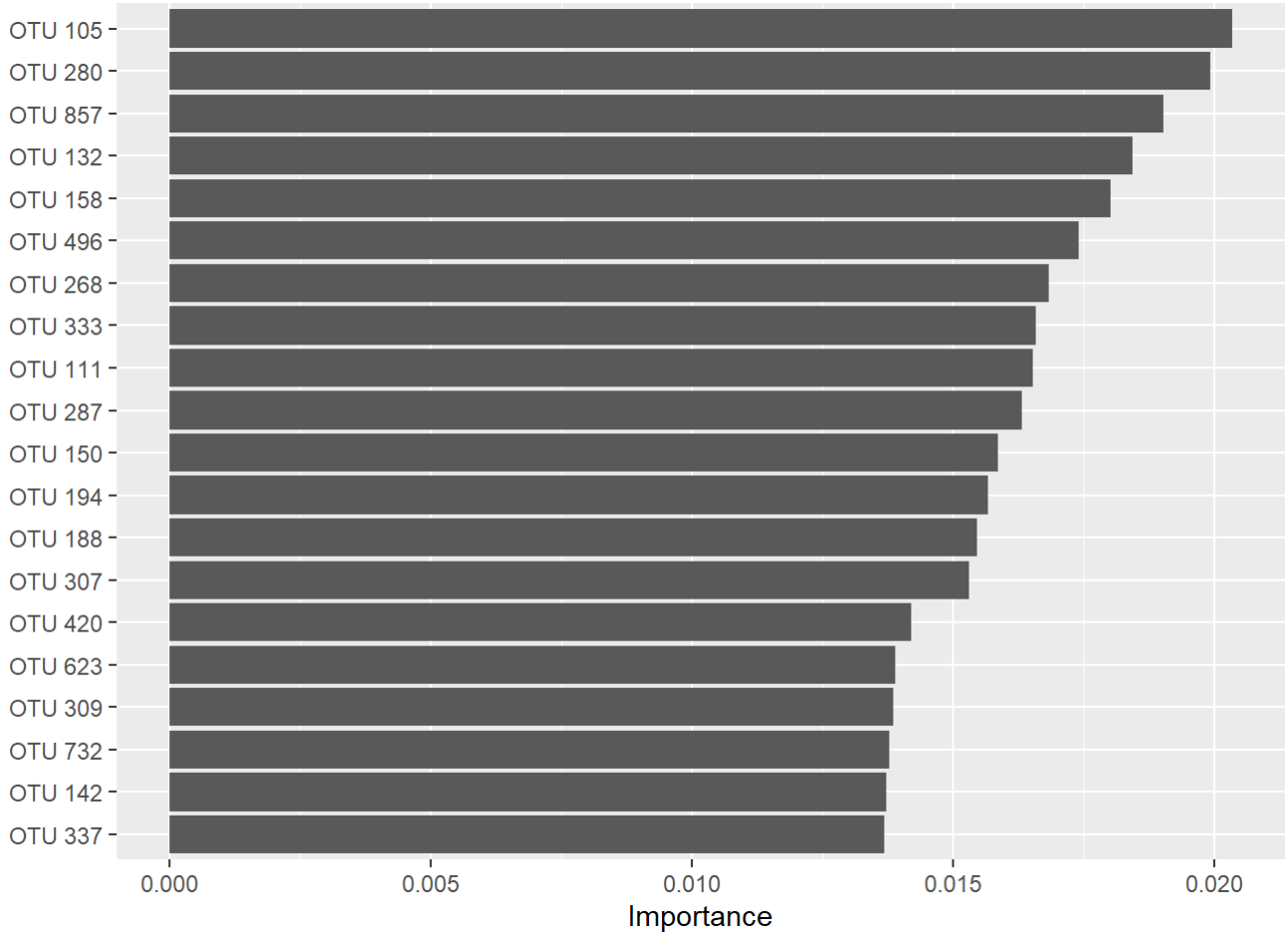
```
plsfinal = plsda(x=select(newdata,-disease), y=newdata$disease, ncomp = 1, type="clas
s", probMethod = "softmax")

pred2 = predict(plsfinal, newdata= tssTest[v[1:20]])
F1_Score(tssTest$disease, pred2)
```

```
## [1] 0.8235294
```

VIP CHART.

This graph allows us to observe which variables have been selected for the model. By generating this type of graph, we will be able to observe whether the most important variables that make up the different models coincide.

```
plsprobs = plsda(x=select(tssTrain,-disease), y=tssTrain$disease, type="prob", probMe
thod = "softmax")
vip(plsprobs, num_features = 20)
```



Species

In this table we can observe the characteristics of the variables that appear in the model, in this case there are 20 variables, however we are only going to show the 10 most important ones.

On the graph it is worth noting that the predominant class is Clostridia, followed by Bacteroidia. These species are the most dominant in all the models carried out, so it is intuited that they will have an important role in the prediction and the model carried out.

```
v
```

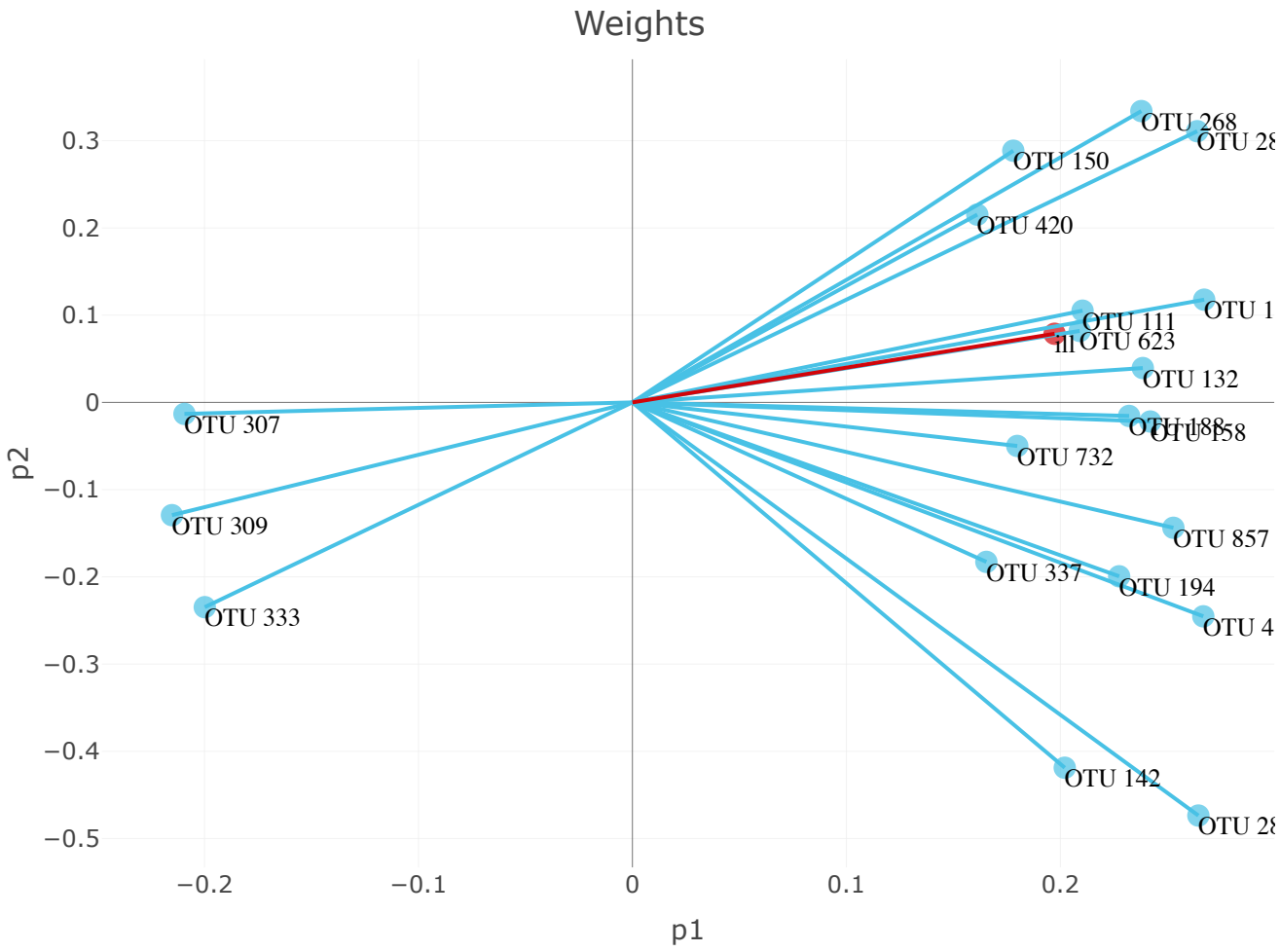
```
##  [1] "OTU 105" "OTU 280" "OTU 857" "OTU 132" "OTU 158" "OTU 496" "OTU 268"
##  [8] "OTU 333" "OTU 111" "OTU 287" "OTU 150" "OTU 194" "OTU 188" "OTU 307"
## [15] "OTU 420" "OTU 623" "OTU 309" "OTU 732" "OTU 142" "OTU 337"
```

```
specis = species[v[1:10]]
as.data.frame(specis)
```

```
##             King        Phylum         Class           Order
## OTU 105 Bacteria Bacteroidetes   Bacteroidia   Bacteroidales
## OTU 280 Bacteria    Firmicutes    Clostridia   Clostridiales
## OTU 857 Bacteria    Firmicutes    Clostridia   Clostridiales
## OTU 132 Bacteria Bacteroidetes   Bacteroidia   Bacteroidales
## OTU 158 Bacteria Bacteroidetes   Bacteroidia   Bacteroidales
## OTU 496 Bacteria  Fusobacteria Fusobacteriia Fusobacteriales
## OTU 268 Bacteria    Firmicutes    Clostridia   Clostridiales
## OTU 333 Bacteria    Firmicutes    Clostridia   Clostridiales
## OTU 111 Bacteria Bacteroidetes   Bacteroidia   Bacteroidales
## OTU 287 Bacteria    Firmicutes    Clostridia   Clostridiales
##                                         Family                Genus
## OTU 105                         Bacteroidaceae          Bacteroides
## OTU 280                         Clostridiaceae          Clostridium
## OTU 857                  Peptostreptococcaceae   Peptostreptococcus
## OTU 132                   Bacteroidales_noname Bacteroidales_noname
## OTU 158                      Porphyromonadaceae        Porphyromonas
## OTU 496                        Fusobacteriaceae        Fusobacterium
## OTU 268                          Clostridiaceae          Clostridium
## OTU 333                        Lachnospiraceae         Butyrivibrio
## OTU 111                         Bacteroidaceae          Bacteroides
## OTU 287 Clostridiales_Family_XI_Incertae_Sedis           Parvimonas
##                             Species Type
## OTU 105        Bacteroides_fragilis <NA>
## OTU 280       Clostridium_symbiosum <NA>
## OTU 857 Peptostreptococcus_stomatis <NA>
## OTU 132 Bacteroidales_bacterium_ph8 <NA>
## OTU 158       Porphyromonas_somerae <NA>
## OTU 496     Fusobacterium_nucleatum <NA>
## OTU 268       Clostridium_hathewayi <NA>
## OTU 333      Butyrivibrio_crossotus <NA>
## OTU 111          Bacteroides_nordii <NA>
## OTU 287     Parvimonas_unclassified <NA>
```

Weights Chart.

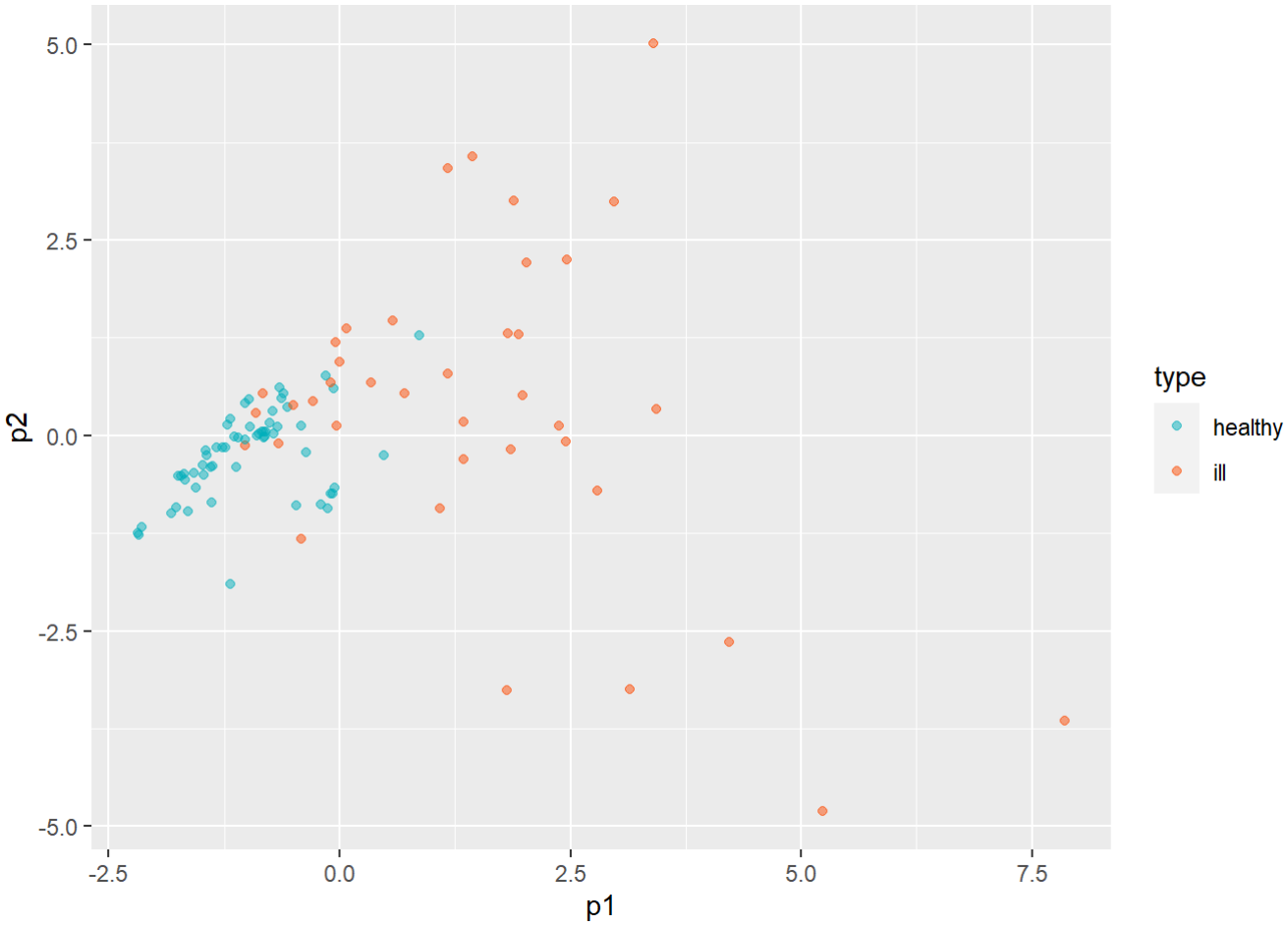
```
library(plotly)

plsprobs = plsda(x=select(newdata,-disease), y=newdata$disease, ncomp = 2, type="clas
s", probMethod = "softmax")

w = plsprobs$loading.weights
c = plsprobs$Yloading[2,]

data = rbind(w,c)
rownames(data) = c(rownames(w),"ill")
colnames(data) = c("p1","p2")
data = as.data.frame(data)
data = data %>% mutate(type = as.factor(c(rep(1,nrow(data)-1),2)))

t <- list(
  family = "sans serif",
  size = 13,
  color = toRGB("black"))

p = plot_ly(data, x=~p1, y=~p2, color=~type, colors = c("#48C1E5","#D20409"), showleg
end = F, text = rownames(data))
p = p %>% add_markers(size=0.5) %>% add_segments(x = 0, xend = ~p1, y = 0, yend = ~p
2)
p1 = p %>% add_text(textfont = t, textposition = "bottom right") %>% layout(title =
"Weights")
p1
```



The S2-TSS Weights graph is more difficult to interpret because it is the one with the largest number of variables. However, we observe a clear difference with respect to the previous graphs. The graph shows two types of relationships, variables that are negatively related to the disease and others that are positively related. The variables that are negatively related to the disease are 3: OTU 307, OTU 309 and OTU 333. On the other hand, we continue to observe a large number of variables that are positively related to the disease. Those that form an angle very close to 0 degrees, such as OTU 111 or OTU 105, stand out.

Scores Chart.

```
scores = plsprobs$scores
scores = data.frame("p1"=scores[,1], "p2"=scores[,2], "type"=tssTrain$disease)
g1 = ggplot(scores, aes(p1,p2, color=type), colour = c("blue","red")) + geom_point(al
pha = I(0.5)) +
     scale_color_manual(values = c("#00AFBB", "#FC4E07"))
g1
```



Probably thanks to the fact that in this model we have a greater number of variables, we can observe the Scores graph a little better. In this case, the clusters of healty individuals around 0 of component 2 are no longer observed. However, it can still be seen how healty individuals are located to the right of 0 with respect to component 1, while ill individuals are located to the left.

Conclusion.

In conclusion, we can say that in this disease we have found a similar AUC in all the preprocessed models. The differences found between the different models are not very significant. However, it can be observed that the CLR preprocessed transformation gives better results than the TSS. On the other hand, the differences between the S1 and S2 models are not observed, as both produce models with a very similar AUC, always

around 80%. Moreover, in both cases, the models provided by our code consist of the same variables. That is, the CLR and TSS preprocessing provide us with the same models in S1 and S2, so it seems that this transformation does not have any differences.