Diabetes

Javier Guillén

10/5/2022

In this Rmarkdown we are going to realise the PLS model in the different preprocessing. In each preprocessing, we have selected the otus that maximise the AUC value and we have performed the final PLS (of the preprocessing in question) adding only these otus. We filtered the otus according to their importance (VIP value).

For each best model of each preprocessing, we have made the scores and loadings graphs to observe how the model behaves.

```r
library(phyloseq)

load("DatosSinOutlier.RData")

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

Carga de la base

```
t2d_sample = as.data.frame(Datos_sample$WT2D)
t2d_sample = factor(t2d_sample$disease)

s2_tss = as.data.frame(scale(log(as.data.frame(t(DatosFinal$WT2D$S2_NORMAL$S2_TSS_WT2D)) * 10e6 + 1),center=TRUE,scale=TRUE))

s2_clr = as.data.frame(scale(as.data.frame(t(DatosFinal$WT2D$S2_NORMAL$S2_CLR_WT2D)),center=TRUE,scale=TRUE))

s1_tss = as.data.frame(scale(log(as.data.frame(t(DatosFinal$WT2D$S1_NORMAL$S1_TSS_WT2D)) * 10e6 + 1),center=TRUE,scale=TRUE))

s1_clr = as.data.frame(scale(as.data.frame(t(DatosFinal$WT2D$S1_NORMAL$S1_CLR_WT2D)),center=TRUE,scale=TRUE))
```

Train y test

S1_CLR

The results of this pre-processing are shown in article

```
data = cbind(s1_clr,t2d_sample)
data$t2d_sample=factor(data$t2d_sample)
set.seed(280)
hv_index <- createDataPartition(data$t2d_sample, p = .75, list = FALSE)
tr_s1clr <- data[ hv_index, ]
te_s1clr <- data[-hv_index, ]
```

In this section we will first make an initial pls. We will observe which are the 25 most important otus (according to their vip value). Subsequently we will perform a loop starting with the 2 most important otus and performing the pls, in each iteration we have been adding the next most important otu and performing the model again. Finally we will have checked which is the optimal number of otus to maximise the AUC value.

```r
library(vip)
library(MLmetrics)
resultados = data.frame()

plsprobs = plsda(x=dplyr::select(tr_s1clr ,-t2d_sample), y=tr_s1clr$t2d_sample, ncomp = 1, type="prob", probMethod = "softmax")


v = vip(plsprobs, num_features = 25)$data$Variable
i=2
for (i in 2:length(v)) {
  otus = v[1:i]
  newdata = as.data.frame(tr_s1clr[otus])
  newdata$disease = tr_s1clr$t2d_sample

  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))]




  plsprobs = plsda(x=select(newdata,-disease), y=newdata$disease, ncomp = c, type="prob", probMethod = "softmax")
  pred = predict(plsprobs, newdata=te_s1clr[otus], type="prob")
  pred = as.data.frame(pred)
  colnames(pred) = c("healthy","ill")
  prediction = prediction(pred$ill,as.numeric(te_s1clr$t2d_sample)-1)
  pred2 = predict(plsprobs, newdata=te_s1clr[otus])
  perf <- performance(prediction,"tpr","fpr")
  perf2 = performance(prediction,"auc")
  perf3 = F1_Score(y_pred = pred2, y_true = te_s1clr$t2d_sample)
  resultados = rbind(resultados, data.frame(importantOtus=i,ncomp=c,auc=as.numeric(perf2@y.values),f1score=perf3))


}
print(resultados)
```

```
##    importantOtus ncomp       auc   f1score
## 1              2     1 0.5538462 0.3529412
## 2              3     1 0.6000000 0.3529412
## 3              4     1 0.5153846 0.5185185
## 4              5     1 0.5307692 0.5000000
## 5              6     1 0.5923077 0.5217391
## 6              7     1 0.5846154 0.5600000
## 7              8     1 0.6384615 0.6153846
## 8              9     1 0.6230769 0.5217391
## 9             10     1 0.6692308 0.6400000
## 10            11     2 0.6230769 0.5600000
## 11            12     1 0.6923077 0.5454545
## 12            13     1 0.6923077 0.5833333
## 13            14     1 0.6923077 0.6153846
## 14            15     1 0.7076923 0.6400000
## 15            16     1 0.7923077 0.6363636
## 16            17     1 0.7923077 0.6363636
## 17            18     1 0.7846154 0.6363636
## 18            19     2 0.7000000 0.6086957
## 19            20     1 0.7846154 0.6363636
## 20            21     1 0.7769231 0.6363636
## 21            22     1 0.7307692 0.5833333
## 22            23     1 0.7307692 0.6086957
## 23            24     1 0.7153846 0.6086957
## 24            25     1 0.7461538 0.6086957
```

Looking at the table above, we can conclude that if we include the 16 most important variables in the model, we obtain the highest AUC.

Now we proceed to the scoring plot

Score Plot

```r
library(plotly)

i=16
otus = v[1:i]
newdata = as.data.frame(tr_s1clr[otus])
newdata$disease = tr_s1clr$t2d_sample



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))]




plsprobs = plsda(x=select(newdata,-disease), y=newdata$disease, ncomp = 2, type="prob", probMethod = "softmax")
scores = plsprobs$scores
scores = data.frame("pc1"=scores[,1], "pc2"=scores[,2], "type"=tr_s1clr$t2d_sample)
g1 = ggplot(scores, aes(pc1,pc2, color=type)) + geom_point(alpha = I(0.6)) +
     scale_color_manual(values = c("blue", "red"))
g1
```

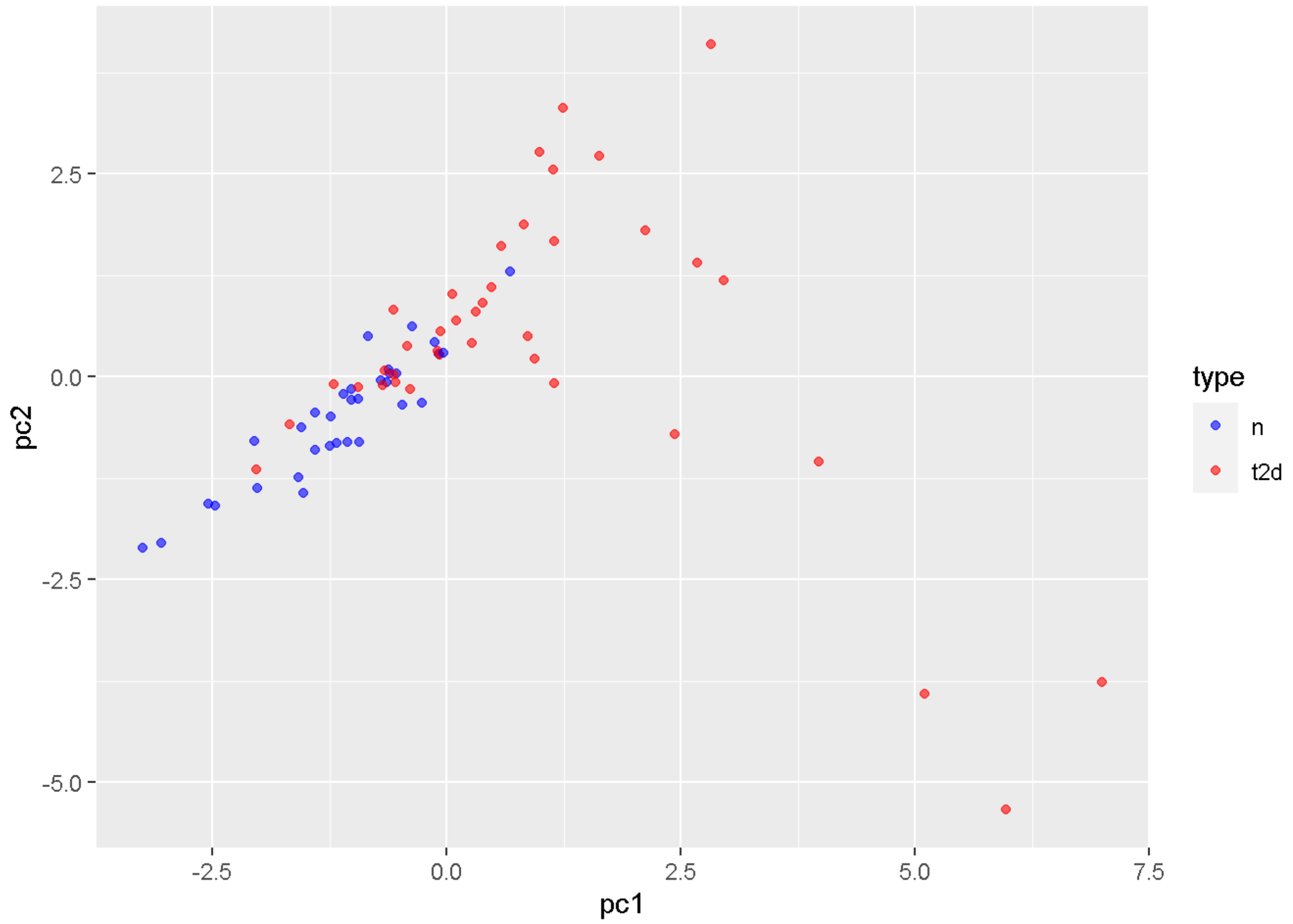
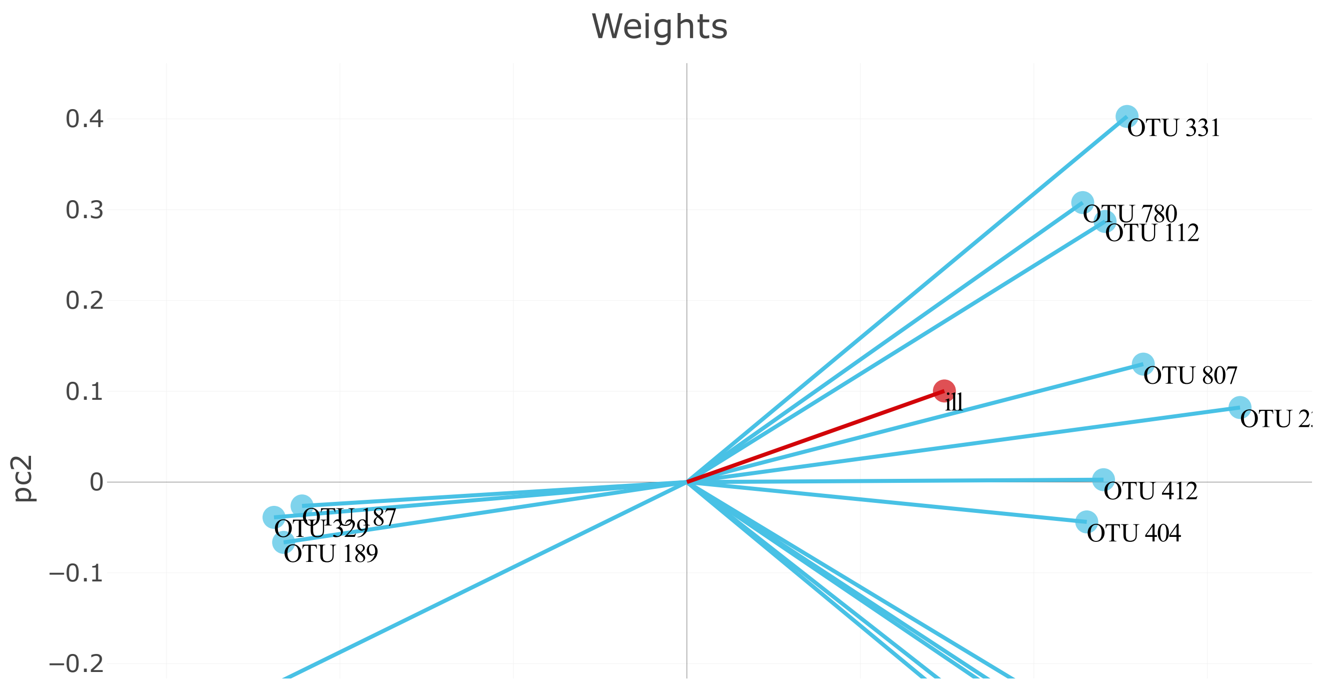
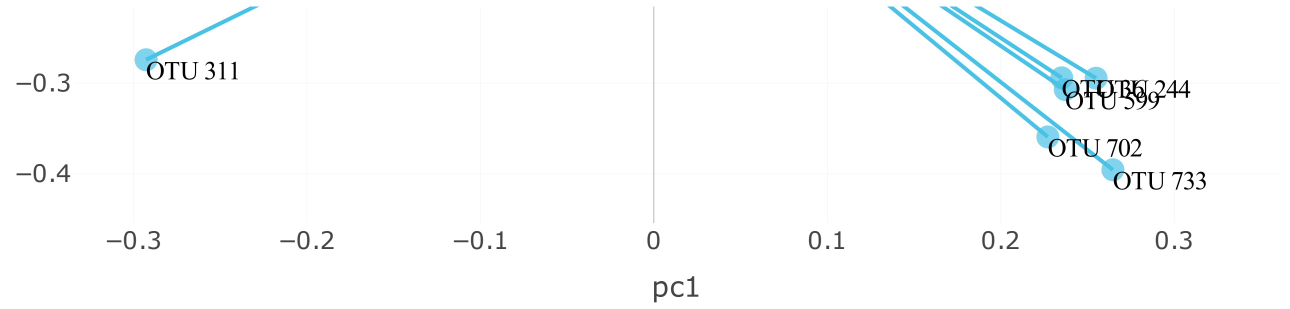


Gráfico loadings

```r
#Aqui hacemos los graficos del mejor pls
w = plsprobs$loading.weights
c = plsprobs$Yloading[2,]

data = rbind(w,c)
rownames(data) = c(rownames(w),"ill")
colnames(data) = c("pc1","pc2")
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=~pc1, y=~pc2, color=~type, colors = c("#48C1E5","#D20409"), showlegend = F, text = rownames(data))
p = p %>% add_markers(size=0.5) %>% add_segments(x = 0, xend = ~pc1, y = 0, yend = ~pc2)
p1 = p %>% add_text(textfont = t, textposition = "bottom right") %>% layout(title = "Weights")
p1
```





The explanation of both graphs can be found in the article

In the following pre-processing we have done the same process, so we will only focus on the explanation of the loading and scoring plot.

S1_TSS

```
data = cbind(s1_tss,t2d_sample)
data$t2d_sample=factor(data$t2d_sample)
set.seed(280)
hv_index <- createDataPartition(data$t2d_sample, p = .75, list = FALSE)
tr_s1tss <- data[ hv_index, ]
te_s1tss <- data[-hv_index, ]
```

```r
library(vip)

resultados = data.frame()

plsprobs = plsda(x=dplyr::select(tr_s1tss ,-t2d_sample), y=tr_s1tss$t2d_sample, ncomp = 1, type="prob", probMethod = "softmax")


v = vip(plsprobs, num_features = 25)$data$Variable
i=2
for (i in 2:length(v)) {
  otus = v[1:i]
  newdata = as.data.frame(tr_s1tss[otus])
  newdata$disease = tr_s1tss$t2d_sample


  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))]



  plsprobs = plsda(x=select(newdata,-disease), y=newdata$disease, ncomp = c, type="prob", probMethod = "softmax")
  pred = predict(plsprobs, newdata=te_s1tss[otus], type="prob")
  pred = as.data.frame(pred)
  colnames(pred) = c("healthy","ill")
  prediction = prediction(pred$ill,as.numeric(te_s1tss$t2d_sample)-1)
  pred2 = predict(plsprobs, newdata=te_s1tss[otus])
  perf <- performance(prediction,"tpr","fpr")
  perf2 = performance(prediction,"auc")
  perf3 = F1_Score(y_pred = pred2, y_true = te_s1tss$t2d_sample)
  resultados = rbind(resultados, data.frame(importantOtus=i,ncomp=c,auc=as.numeric(perf2@y.values),f1score=perf3))


}
print(resultados)
```

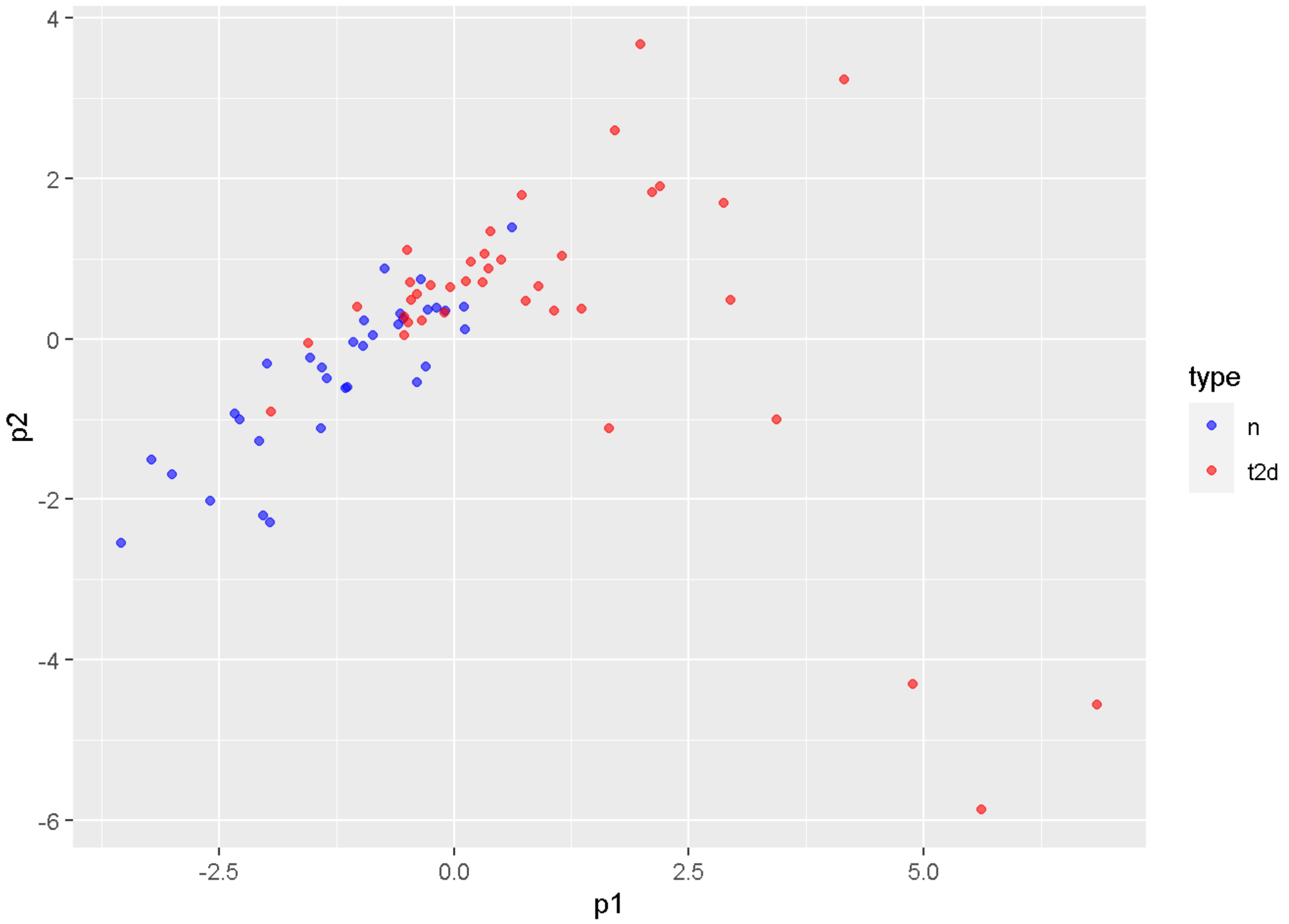
```
##    importantOtus ncomp       auc   f1score
## 1              2     1 0.5538462 0.3529412
## 2              3     1 0.6076923 0.3333333
## 3              4     1 0.6076923 0.6153846
## 4              5     2 0.5769231 0.5000000
## 5              6     1 0.5769231 0.5000000
## 6              7     3 0.6615385 0.5833333
## 7              8     1 0.6230769 0.4545455
## 8              9     1 0.6692308 0.5454545
## 9             10     1 0.6692308 0.6400000
## 10            11     1 0.6769231 0.6400000
## 11            12     1 0.6769231 0.6400000
## 12            13     1 0.6769231 0.6400000
## 13            14     1 0.6923077 0.6666667
## 14            15     1 0.7076923 0.6400000
## 15            16     1 0.7692308 0.6363636
## 16            17     2 0.6923077 0.5000000
## 17            18     1 0.7461538 0.6363636
## 18            19     1 0.7538462 0.6363636
## 19            20     2 0.7000000 0.5454545
## 20            21     2 0.7000000 0.6666667
## 21            22     2 0.7000000 0.6666667
## 22            23     3 0.6538462 0.6923077
## 23            24     3 0.6615385 0.6923077
## 24            25     2 0.6846154 0.6400000
```

Scoring plot

```
library(plotly)

i=19
otus = v[1:i]
newdata = as.data.frame(tr_s1tss[otus])
newdata$disease = tr_s1tss$t2d_sample



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))]
c
```

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

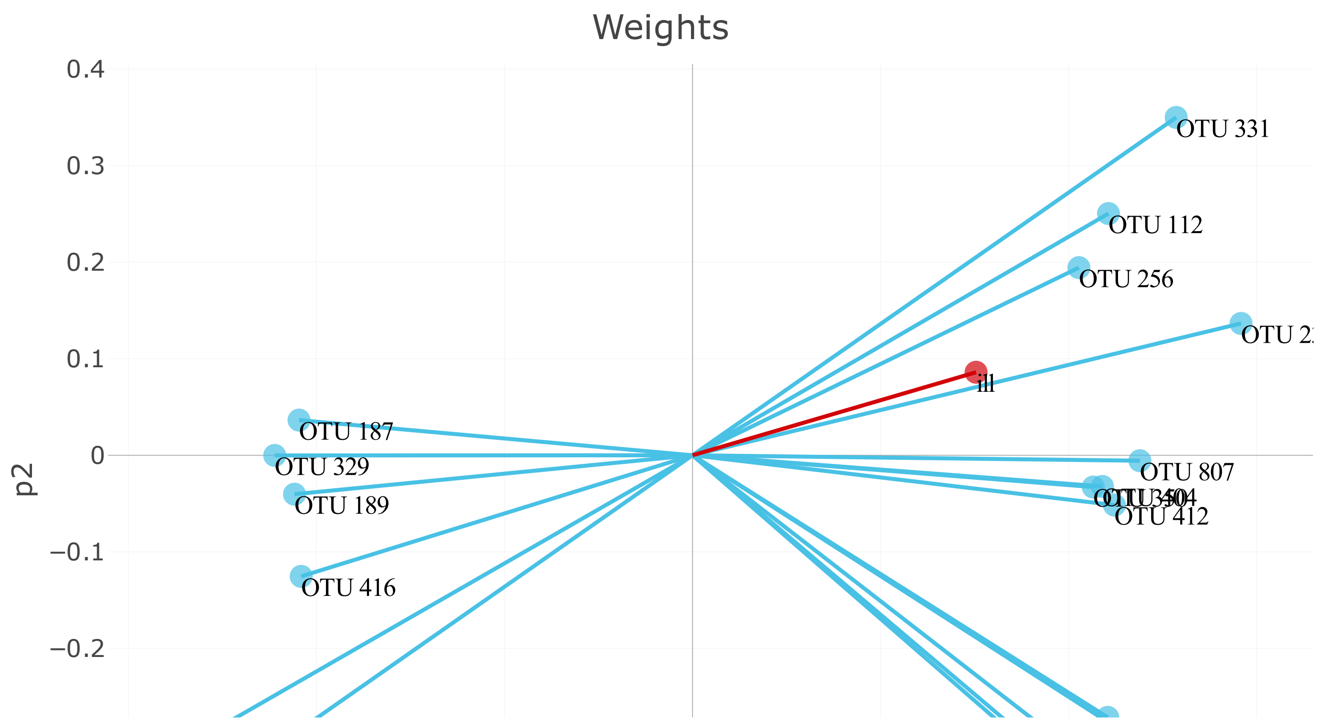
```
plsprobs = plsda(x=select(newdata,-disease), y=newdata$disease, ncomp = 2, type="prob", probMethod = "softmax")
scores = plsprobs$scores
scores = data.frame("p1"=scores[,1], "p2"=scores[,2], "type"=tr_s1tss$t2d_sample)
g1 = ggplot(scores, aes(p1,p2, color=type)) + geom_point(alpha = I(0.6)) +
     scale_color_manual(values = c("blue", "red"))
g1
```

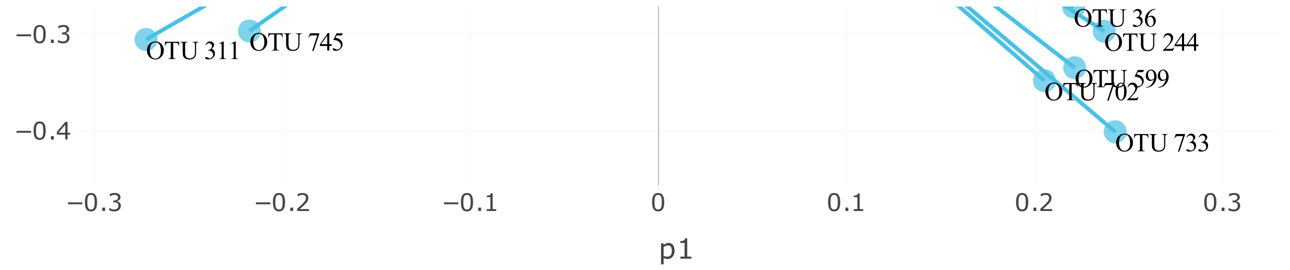


In the S1-clr preprocessing we have used the 19 most important variables and we observe that the AUC value has dropped a little. However, we can see that the score plot is very similar, in fact we find no significant differences, the points follow the same distribution.

Loading plot

```
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"), showlegend = F, text = rownames(data))
p = p %>% add_markers(size=0.5) %>% add_segments(x = 0, xend = ~p1, y = 0, yend = ~p2)
p1 = p %>% add_text(textfont = t, textposition = "bottom right") %>% layout(title = "Weights")
p1
```





Again in the loading plot we observe the same trend. We see no significant differences with respect to the clr processing.

With this we can conclude that we do not find very significant differences between the clr and tss normalisations within the s1 preprocessing.

S2_CLR

Now we are going to perform the same process in preprocessing S2. First we will do it by taking the normalisation clr

```
data = cbind(s2_clr,t2d_sample)
data$t2d_sample=factor(data$t2d_sample)
set.seed(280)
hv_index <- createDataPartition(data$t2d_sample, p = .75, list = FALSE)
tr_s2clr <- data[ hv_index, ]
te_s2clr <- data[-hv_index, ]
```

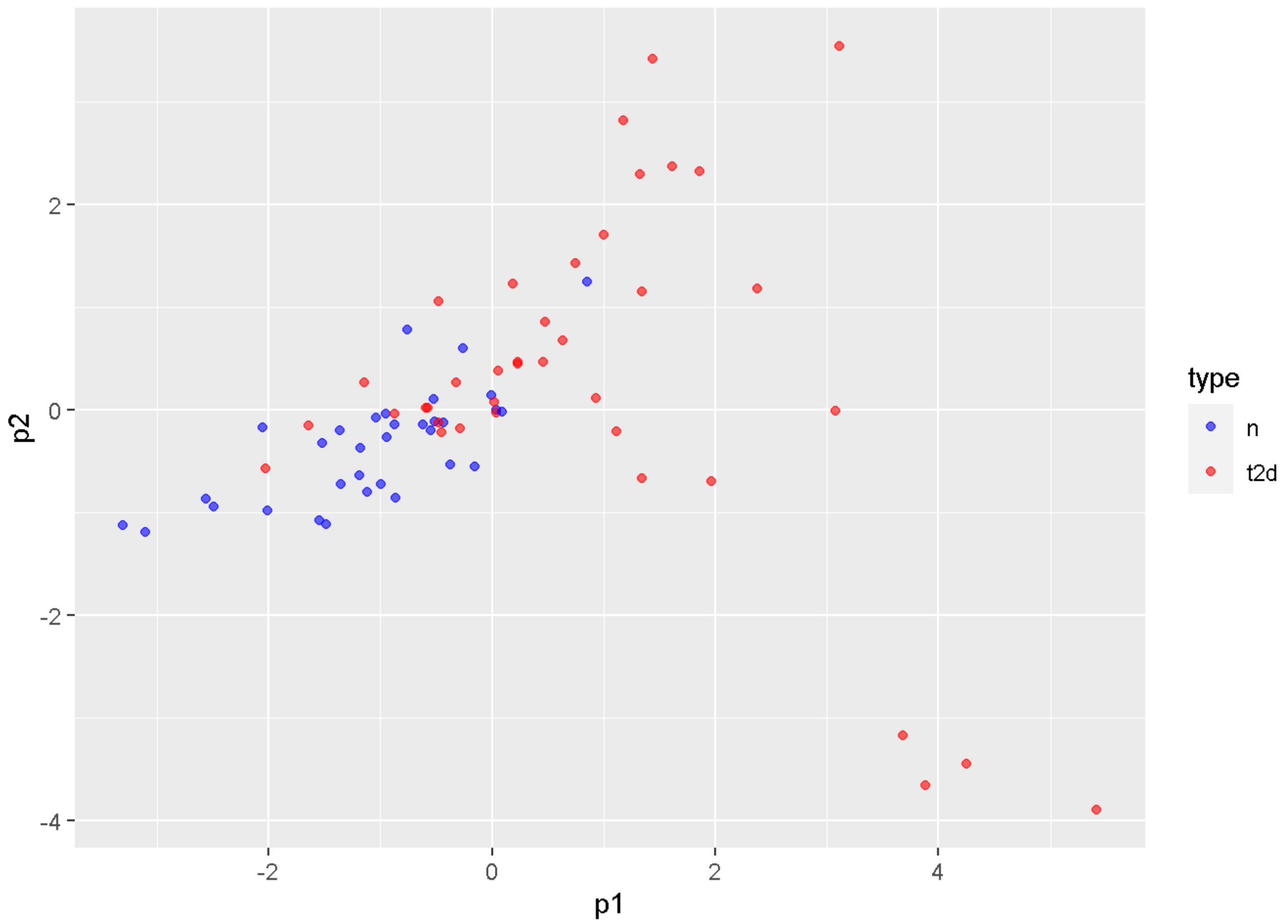
```r
library(vip)

resultados = data.frame()

plsprobs = plsda(x=dplyr::select(tr_s2clr ,-t2d_sample), y=tr_s2clr$t2d_sample, ncomp = 1, type="prob", probMethod = "softmax")


v = vip(plsprobs, num_features = 25)$data$Variable
i=2
for (i in 2:length(v)) {
  otus = v[1:i]
  newdata = as.data.frame(tr_s2clr[otus])
  newdata$disease = tr_s2clr$t2d_sample


  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))]



  plsprobs = plsda(x=select(newdata,-disease), y=newdata$disease, ncomp = c, type="prob", probMethod = "softmax")
  pred = predict(plsprobs, newdata=te_s2clr[otus], type="prob")
  pred = as.data.frame(pred)
  colnames(pred) = c("healthy","ill")
  prediction = prediction(pred$ill,as.numeric(te_s2clr$t2d_sample)-1)
  pred2 = predict(plsprobs, newdata=te_s2clr[otus])
  perf <- performance(prediction,"tpr","fpr")
  perf2 = performance(prediction,"auc")
  perf3 = F1_Score(y_pred = pred2, y_true = te_s2clr$t2d_sample)
  resultados = rbind(resultados, data.frame(importantOtus=i,ncomp=c,auc=as.numeric(perf2@y.values),f1score=perf3))


}
print(resultados)
```

```
##    importantOtus ncomp       auc   f1score
## 1              2     1 0.5538462 0.3529412
## 2              3     1 0.6000000 0.3529412
## 3              4     1 0.5153846 0.5185185
## 4              5     1 0.5307692 0.5000000
## 5              6     1 0.5923077 0.5217391
## 6              7     1 0.5846154 0.5600000
## 7              8     1 0.6384615 0.6153846
## 8              9     1 0.6230769 0.5217391
## 9             10     2 0.6307692 0.6400000
## 10            11     1 0.6461538 0.5454545
## 11            12     1 0.6461538 0.5833333
## 12            13     2 0.6538462 0.6400000
## 13            14     1 0.7307692 0.6086957
## 14            15     1 0.7307692 0.6086957
## 15            16     1 0.7307692 0.6086957
## 16            17     1 0.7307692 0.6086957
## 17            18     2 0.6461538 0.4761905
## 18            19     1 0.6769231 0.4761905
## 19            20     1 0.7000000 0.5833333
## 20            21     1 0.6846154 0.5454545
## 21            22     1 0.7000000 0.6086957
## 22            23     1 0.7076923 0.6086957
## 23            24     1 0.6615385 0.5714286
## 24            25     1 0.6615385 0.5454545
```

In this case, we find a lower AUC: 0.73 with the 14 best otus.

Scoring plot

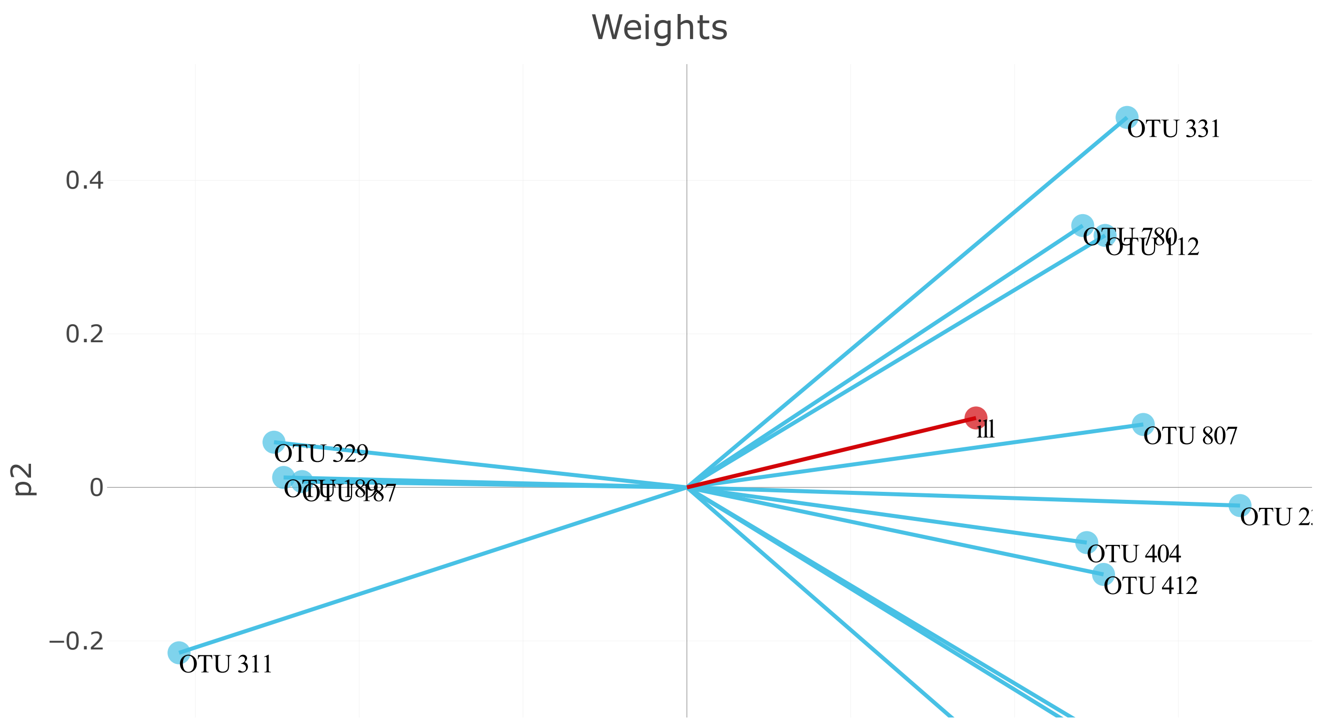
```r
library(plotly)

i=14
otus = v[1:i]
newdata = as.data.frame(tr_s2clr[otus])
newdata$disease = tr_s2clr$t2d_sample



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))]




plsprobs = plsda(x=select(newdata,-disease), y=newdata$disease, ncomp = 2, type="prob", probMethod = "softmax")
scores = plsprobs$scores
scores = data.frame("p1"=scores[,1], "p2"=scores[,2], "type"=tr_s2clr$t2d_sample)
g1 = ggplot(scores, aes(p1,p2, color=type)) + geom_point(alpha = I(0.6)) +
     scale_color_manual(values = c("blue", "red"))
g1
```

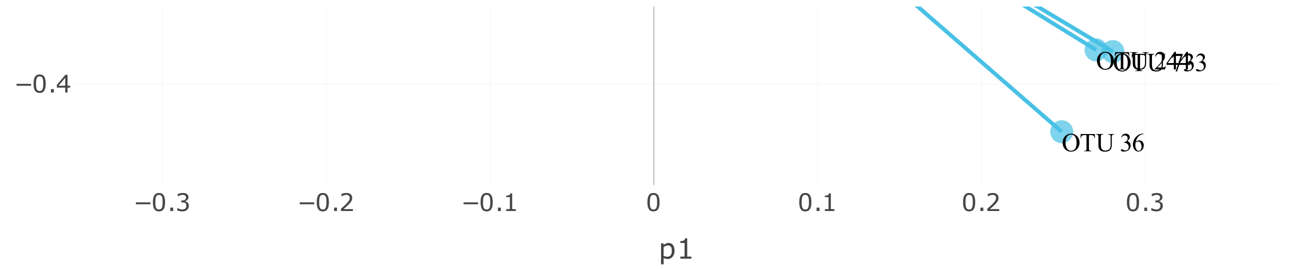


With respect to the score plot, we found no significant differences with respect to the S1 preprocessing. We see again that the points follow the same distribution and there is a certain degree of separation.

Loading plot

```
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"), showlegend = F, text = rownames(data))
p = p %>% add_markers(size=0.5) %>% add_segments(x = 0, xend = ~p1, y = 0, yend = ~p2)
p1 = p %>% add_text(textfont = t, textposition = "bottom right") %>% layout(title = "Weights")
p1
```





However, there are differences with respect to the loading plot. In this graph we see again that there is a selection of otus that are negatively correlated with the presence of the disease. However, we do not find a clear distinction between micro-organisms that are neutral and micro-organisms that favour the presence of the disease.

S2_TSS

Now we perform the same process in the other normalisation: tss

```
data = cbind(s2_tss,t2d_sample)
data$t2d_sample=factor(data$t2d_sample)
set.seed(280)
hv_index <- createDataPartition(data$t2d_sample, p = .75, list = FALSE)
tr_s2tss <- data[ hv_index, ]
te_s2tss <- data[-hv_index, ]
```

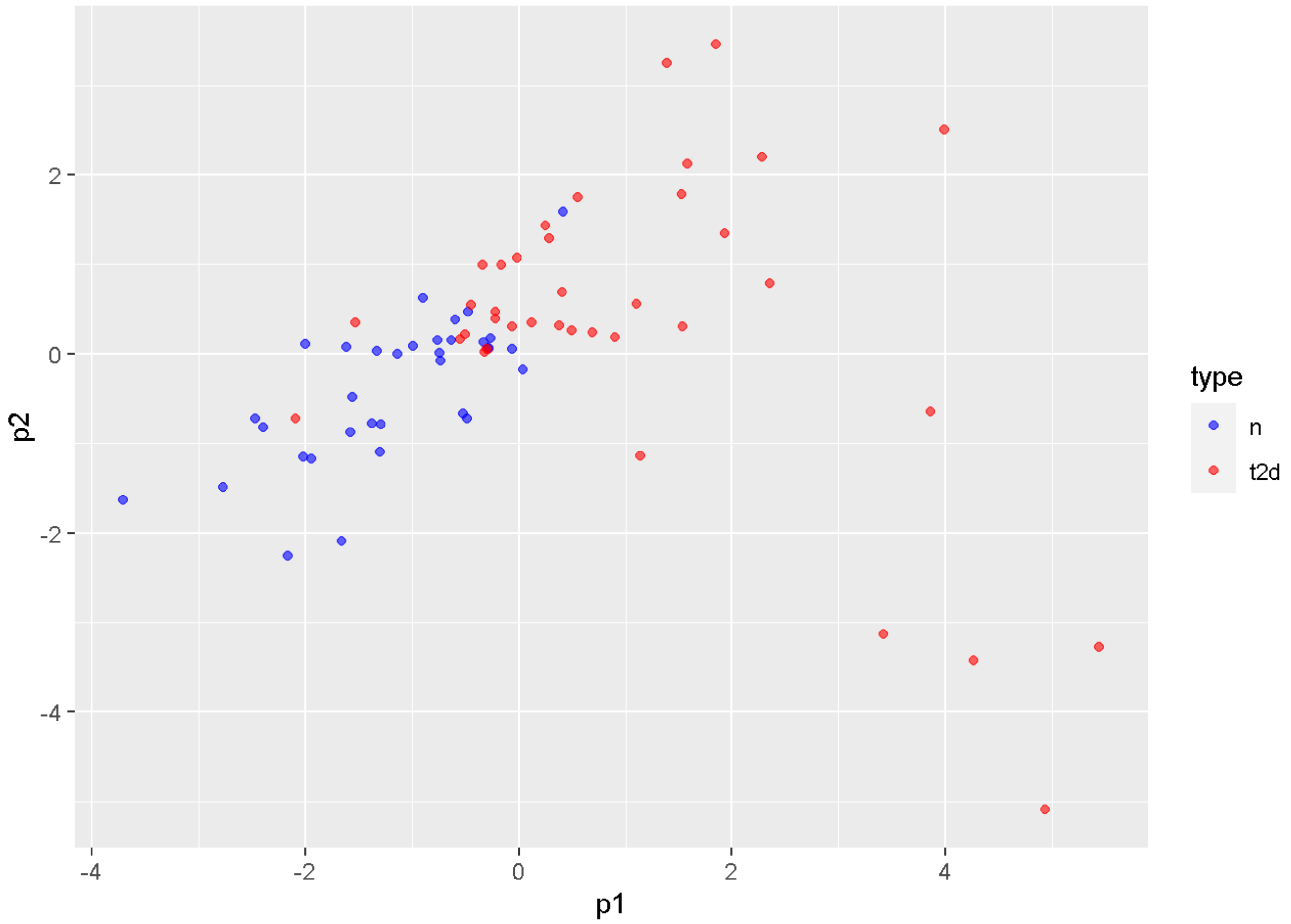
```r
library(vip)

resultados = data.frame()

plsprobs = plsda(x=dplyr::select(tr_s2tss ,-t2d_sample), y=tr_s2tss$t2d_sample, ncomp = 1, type="prob", probMethod = "softmax")


v = vip(plsprobs, num_features = 25)$data$Variable
i=2
for (i in 2:length(v)) {
  otus = v[1:i]
  newdata = as.data.frame(tr_s2tss[otus])
  newdata$disease = tr_s2tss$t2d_sample

  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))]


  plsprobs = plsda(x=select(newdata,-disease), y=newdata$disease, ncomp = c, type="prob", probMethod = "softmax")
  pred = predict(plsprobs, newdata=te_s2tss[otus], type="prob")
  pred = as.data.frame(pred)
  colnames(pred) = c("healthy","ill")
  prediction = prediction(pred$ill,as.numeric(te_s2tss$t2d_sample)-1)
  pred2 = predict(plsprobs, newdata=te_s2tss[otus])
  perf <- performance(prediction,"tpr","fpr")
  perf2 = performance(prediction,"auc")
  perf3 = F1_Score(y_pred = pred2, y_true = te_s2tss$t2d_sample)
  resultados = rbind(resultados, data.frame(importantOtus=i,ncomp=c,auc=as.numeric(perf2@y.values),f1score=perf3))


}
print(resultados)
```

```
##    importantOtus ncomp       auc   f1score
## 1              2     1 0.5461538 0.3529412
## 2              3     1 0.6076923 0.3333333
## 3              4     3 0.6538462 0.6400000
## 4              5     2 0.5769231 0.5000000
## 5              6     1 0.5769231 0.5000000
## 6              7     3 0.6615385 0.6400000
## 7              8     2 0.6076923 0.6153846
## 8              9     2 0.6076923 0.6153846
## 9             10     1 0.6230769 0.6153846
## 10            11     1 0.6230769 0.5217391
## 11            12     1 0.6153846 0.5217391
## 12            12     2 0.6153846 0.5217391
## 13            13     1 0.6461538 0.6400000
## 14            14     1 0.7384615 0.6086957
## 15            15     1 0.7153846 0.6666667
## 16            16     1 0.7076923 0.6666667
## 17            17     1 0.7153846 0.6666667
## 18            18     1 0.7230769 0.6666667
## 19            19     1 0.7538462 0.6400000
## 20            20     1 0.7153846 0.6666667
## 21            21     2 0.6923077 0.6400000
## 22            22     2 0.6923077 0.6400000
## 23            23     2 0.6692308 0.5833333
## 24            24     2 0.6692308 0.5217391
## 25            25     2 0.6692308 0.5217391
```

n this case, we find a AUC: 0.75 with the 19 best otus.

Scoring plot

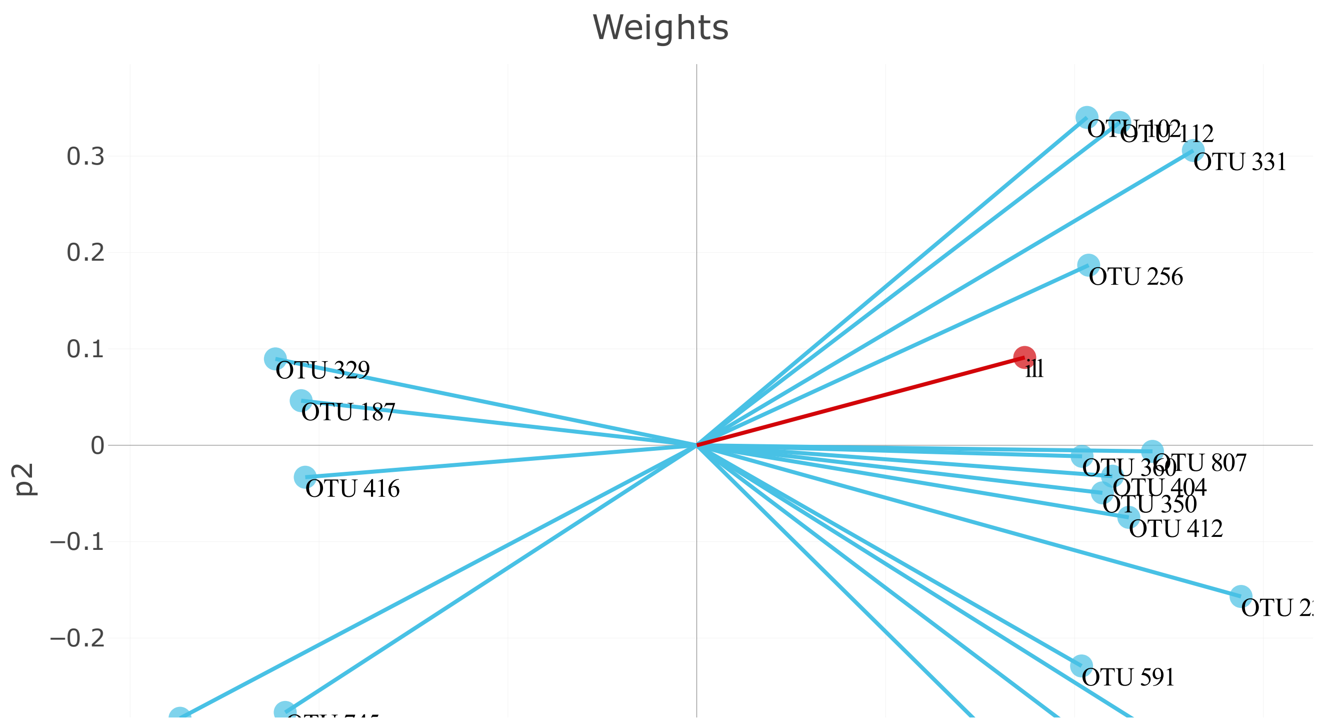
```r
library(plotly)

i=19
otus = v[1:i]
newdata = as.data.frame(tr_s2tss[otus])
newdata$disease = tr_s2tss$t2d_sample



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))]




plsprobs = plsda(x=select(newdata,-disease), y=newdata$disease, ncomp = 2, type="prob", probMethod = "softmax")
scores = plsprobs$scores
scores = data.frame("p1"=scores[,1], "p2"=scores[,2], "type"=tr_s2tss$t2d_sample)
g1 = ggplot(scores, aes(p1,p2, color=type)) + geom_point(alpha = I(0.6)) +
     scale_color_manual(values = c("blue", "red"))
g1
```

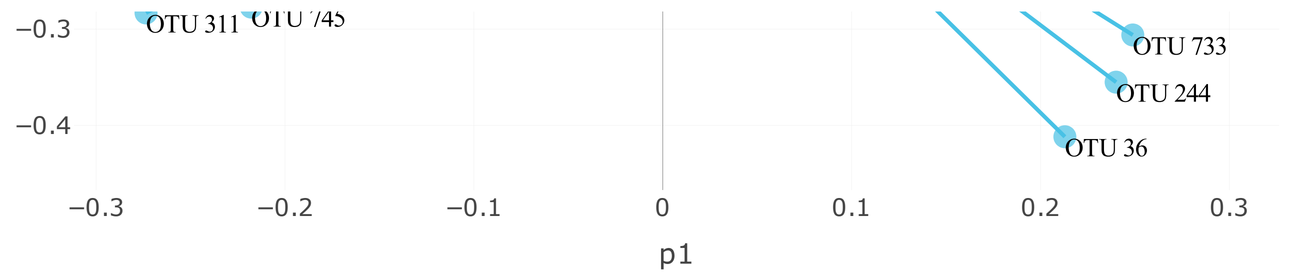


Regarding the scores graph we can observe the same as in the clr normalisation. So we could conclude that the separation of individuals is not significantly different between preprocessed s1 and s2.

Loading plot

```
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"), showlegend = F, text = rownames(data))
p = p %>% add_markers(size=0.5) %>% add_segments(x = 0, xend = ~p1, y = 0, yend = ~p2)
p1 = p %>% add_text(textfont = t, textposition = "bottom right") %>% layout(title = "Weights")
p1
```





Finally, with respect to the loading plot we observe the same peculiarity as in the tss normalisation.

So we can conclude that probably the preprocessed s1 groups the important otus better than the preprocessed one.

As a final conclusion, we have observed that S1 obtains a better AUC than S2 and also manages to separate the otus better according to their correlation with the presence of the disease.