Notebook1_DIABLO

May 8, 2024

1 DIABLO, Integration of multi-omics data

This is an adaptation of this vignette: https://bioconductor.org/packages/release/bioc/vignettes/mixOmics/inst/c
mixOmics package tutorials here: http://mixomics.org/

Data were preprocessed for participants first measurement of each omic.

There is a lot of fine tuning that could be done if we want to showcase this analysis.

[1]:

```
suppressPackageStartupMessages(library(tidyverse))
suppressPackageStartupMessages(library(mixOmics))
suppressPackageStartupMessages(library(plyr))
suppressPackageStartupMessages(library(caTools))
suppressPackageStartupMessages(library(caret))
suppressPackageStartupMessages(library("BiocParallel"))

set.seed(99)
```

[2]:

```
options(jupyter.plot_scale=1,
        width=200,
        repr.matrix.max.cols=200,
        repr.matrix.max.rows=Inf)
```

[8]:

```
getwd()
```

'/home/sagemaker-user/Aging_Workshop_24/Session2'

[10]:

```
# Read in baseline measures
### NOTE DIABLO can be run with repeated measures
platform <- "SageMaker"
if ("SageMaker" == platform) {
    omicsDir  <- "/home/sagemaker-user/Aging_Workshop_24/data"
    frailtyDir <- "/home/sagemaker-user/Aging_Workshop_24/data/frailty"
} else {
    omicsDir <- "../WGCNA/"
    frailtyDir <- "/shared-libs/useful-files/frailty_measures_kanelab/
  ↪FI_workshop_040124"
}
prots <- read_delim(file.path(omicsDir,"prot_baseline.csv"), delim=",")
```

```r
mets <- read_delim(file.path(omicsDir,"mets_baseline.csv"), delim=",")
clin <- read_delim(file.path(omicsDir,"clinical_baseline.csv"), delim=",")

print(dim(mets))
print(dim(prots))
print(dim(clin))

frailty <- read_delim(file.path(frailtyDir,"combination_fi_040124.csv"),␣
 ↪delim=",")
```

```
Rows: 2842 Columns: 1196
  Column specification



Delimiter: ","
chr    (1): public_client_id
dbl (1195): CAM_O00533(CHL1), CAM_O14786(NRP1), CAM_O15031(PLXNB2),
CAM_O75015(FCGR3B), CAM_O75023(LILRB5), CAM_O95445(APOM), CAM_P00441(SOD1),
CAM_P00915(CA1), CAM_P01033(TIMP1), CAM_P01034(CST3)…

  Use `spec()` to retrieve the full column specification for this
data.
  Specify the column types or set `show_col_types = FALSE` to quiet
this message.
Warning message:
"One or more parsing issues, call `problems()` on your data frame for
details, e.g.:
  dat <- vroom(…)
  problems(dat)"
Rows: 2033 Columns: 1051
  Column specification



Delimiter: ","
chr    (1): public_client_id
dbl (1045): 35(S-1-pyrroline-5-carboxylate), 50(spermidine),
55(1-methylnicotinamide), 62(12,13-DiHOME), 71(5-hydroxyindoleacetate),
93(alpha-ketoglutarate), 98(kynurenate), 111(3-hydroxyisobutyra…
lgl    (5): 100001317(pioglitazone), 100001538(lidocaine),
100002719(cotinine N-oxide), 100004177(hydroxypioglitazone (M-IV)),
100008943(pregabalin)

  Use `spec()` to retrieve the full column specification for this
data.
  Specify the column types or set `show_col_types = FALSE` to quiet
```

```
this message.
Rows: 4879 Columns: 129
  Column specification



Delimiter: ","
chr   (1): public_client_id
dbl (128): A/G RATIO, ADIPONECTIN, SERUM, ALAT (SGPT), ALBUMIN,
ALKALINE PHOSPHATASE, ANTIOXID CAP, TOTAL, ARACHIDONIC ACID, ARSENIC, BLOOD,
ASAT (SGOT), BASOPHILS, BASOPHILS ABSOLUTE, BILIRUBIN, …

  Use `spec()` to retrieve the full column specification for this
data.
  Specify the column types or set `show_col_types = FALSE` to quiet
this message.
```

```
[1] 2033 1051
[1] 2842 1196
[1] 4879  129
```

```
New names:
• `` -> `…1`
Rows: 3090 Columns: 83
  Column specification



Delimiter: ","
chr  (3): public_client_id, sex, race
dbl (80): …1, self_fi, self_fi_sum, num_na_self, days_in_program,
age, Dise1, Dise2, Dise3, Dise4, Dise5, Dise6, Dise7, Dise8, Dise9, Dise10,
Dise11, Dise12, Dise13, Dise14, Dise15, SAT_1, SAT_2…

  Use `spec()` to retrieve the full column specification for this
data.
  Specify the column types or set `show_col_types = FALSE` to quiet
this message.
```

[11]:

```
# Light filtering of missing values per row/colum

mets_filt <- mets[, colMeans(is.na(mets)) <= .15]
prots_filt <- prots[, colMeans(is.na(prots)) <= .15]
clin_filt <- clin[, colMeans(is.na(clin)) <= .15]
print(dim(mets_filt))
print(dim(prots_filt))
print(dim(clin_filt))
```

```r
mets_filt <- mets_filt[rowMeans(is.na(mets_filt)) <= .15,]
prots_filt <- prots_filt[rowMeans(is.na(prots_filt)) <= .15,]
clin_filt <- clin_filt[rowMeans(is.na(clin_filt)) <= .15,]
print(dim(mets_filt))
print(dim(prots_filt))
print(dim(clin_filt))


## Diablo uses NIPALS for imputation.
## Diablo centers and scaled data.
```

```
[1] 2033  779
[1] 2842  275
[1] 4879   48
[1] 2009  779
[1] 2842  275
[1] 4828   48
```

[12]:

```r
# Merge to get participants with all measures
#put all data frames into list
df_list <- list(frailty, mets_filt, prots_filt, clin_filt)

#merge all data frames in list
combined_df <- df_list %>% reduce(inner_join, by = "public_client_id")

dim(combined_df)
```

1. 891 2. 1182

[13]:

```r
# Split into "blocks" for DIABLO
metabolites <- combined_df[,colnames(combined_df) %in% colnames(mets_filt)]
proteins <- combined_df[,colnames(combined_df) %in% colnames(prots_filt)]
clinical <- combined_df[,colnames(combined_df) %in% colnames(clin_filt)]
frailty <- combined_df[, c("public_client_id", "sex", "age", "race", "self_fi",␣
  ↪"lab_fi", "merge_fi")]
# DIABLO can only be run with categorical variables
# Split frailty measures into quintiles
frailty <- frailty %>%
  dplyr::mutate(lab_quantile = dplyr::ntile(lab_fi, 5),
                self_quantile = dplyr::ntile(self_fi, 5),
                merge_quantile = dplyr::ntile(merge_fi, 5))
```

[14]:

```r
round(cor(frailty$self_quantile, frailty$lab_quantile, method='s'),3)
table(frailty$self_quantile, frailty$lab_quantile)
```

0.342

```
    1  2  3  4  5
1  58 43 35 29 14
2  56 33 40 30 19
3  35 42 37 27 37
4  19 37 39 45 38
5  11 23 27 47 70
```

[15]:

```
round(cor(frailty$merge_quantile, frailty$self_quantile, method='s'),3)
table(frailty$merge_quantile, frailty$self_quantile)
```

0.73

```
     1   2   3   4   5
1  100  60  19   0   0
2   47  59  52  18   2
3   21  37  54  54  12
4   10  17  33  71  47
5    1   5  20  35 117
```

[16]:

```
round(cor(frailty$merge_quantile, frailty$lab_quantile, method='s'),3)
table(frailty$merge_quantile, frailty$lab_quantile)
```

0.825

```
     1   2   3   4   5
1  130  48   1   0   0
2   37  65  65  11   0
3    9  48  61  57   3
4    2  15  41  74  46
5    1   2  10  36 129
```

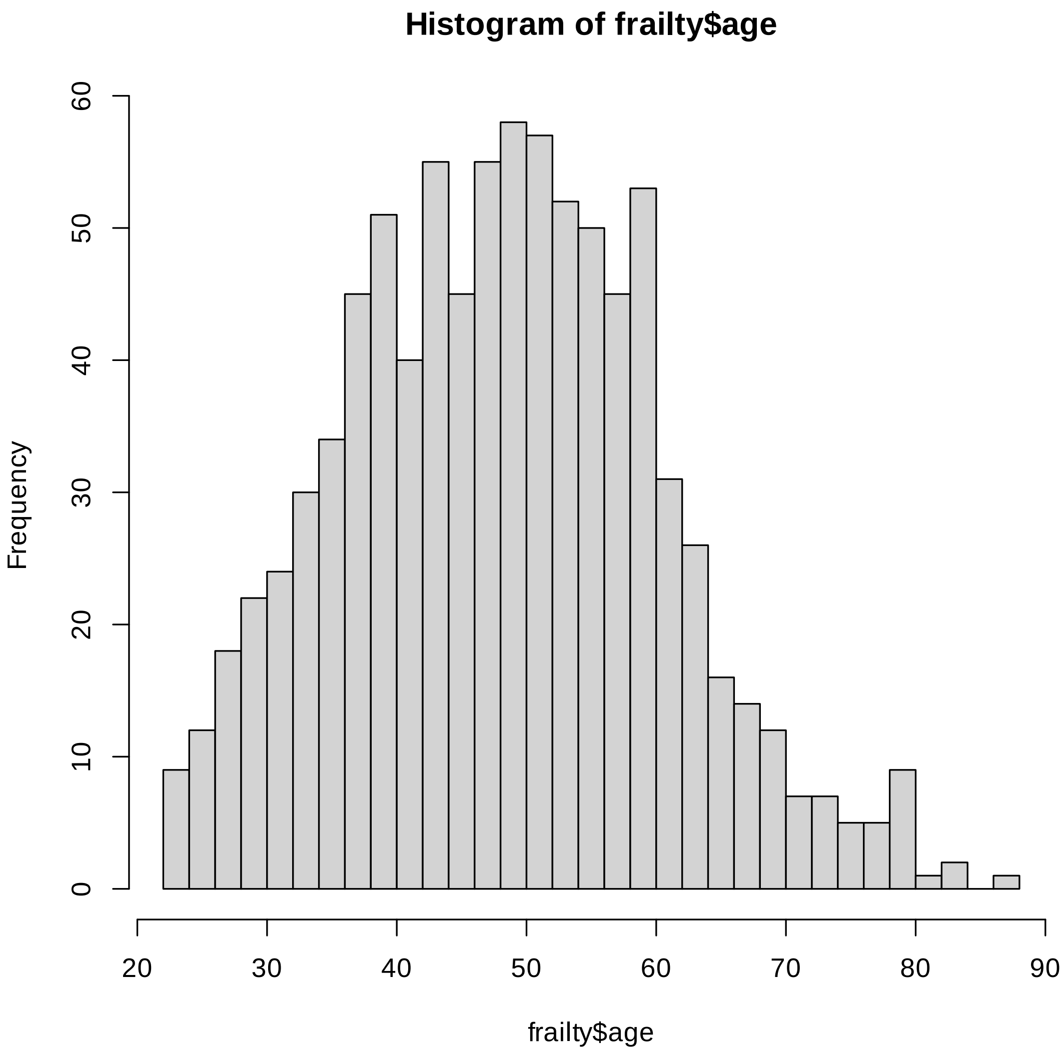
[17]:

```
table(frailty$sex)
```

```
  F   M
572 319
```

[18]:

```
hist(frailty$age, breaks=40)
```



[19]:

```
# Check rows are in the same order
all(frailty$public_client_id == metabolites$public_client_id)
all(frailty$public_client_id == proteins$public_client_id)
all(frailty$public_client_id == clinical$public_client_id)
```

TRUE

TRUE

TRUE

2 Single 'Omics PCA analysis

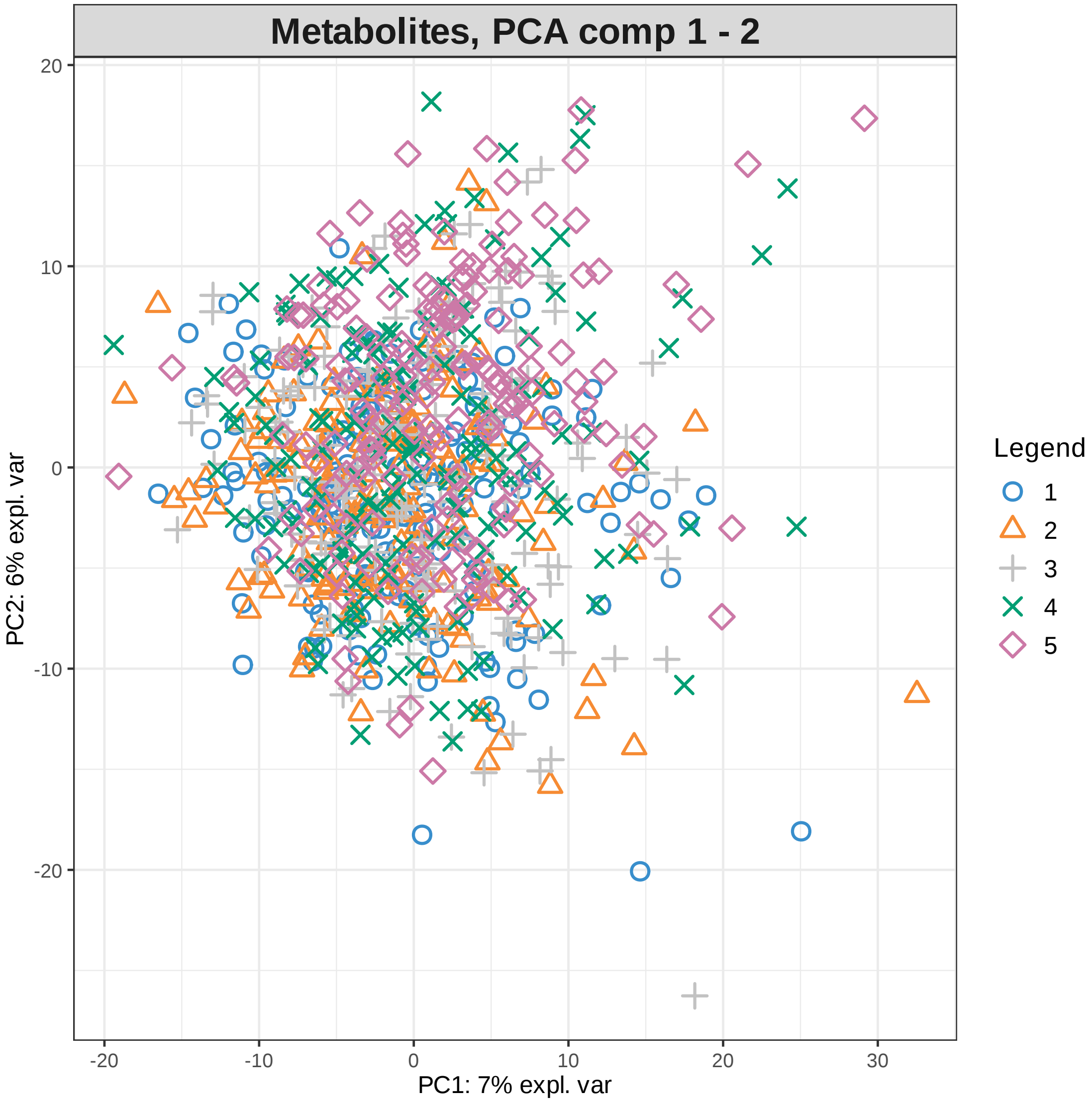
Using full data for the exploratory analysis. We could consider breaking into test/train to get out-of-sample predictions, but the goal here is just to take a quick look at the data so we know what to expect. These PCA plots show that self-reported Frailty Index is hard to predict on the basis of this data; achieving good performance will be difficult.

[20]:

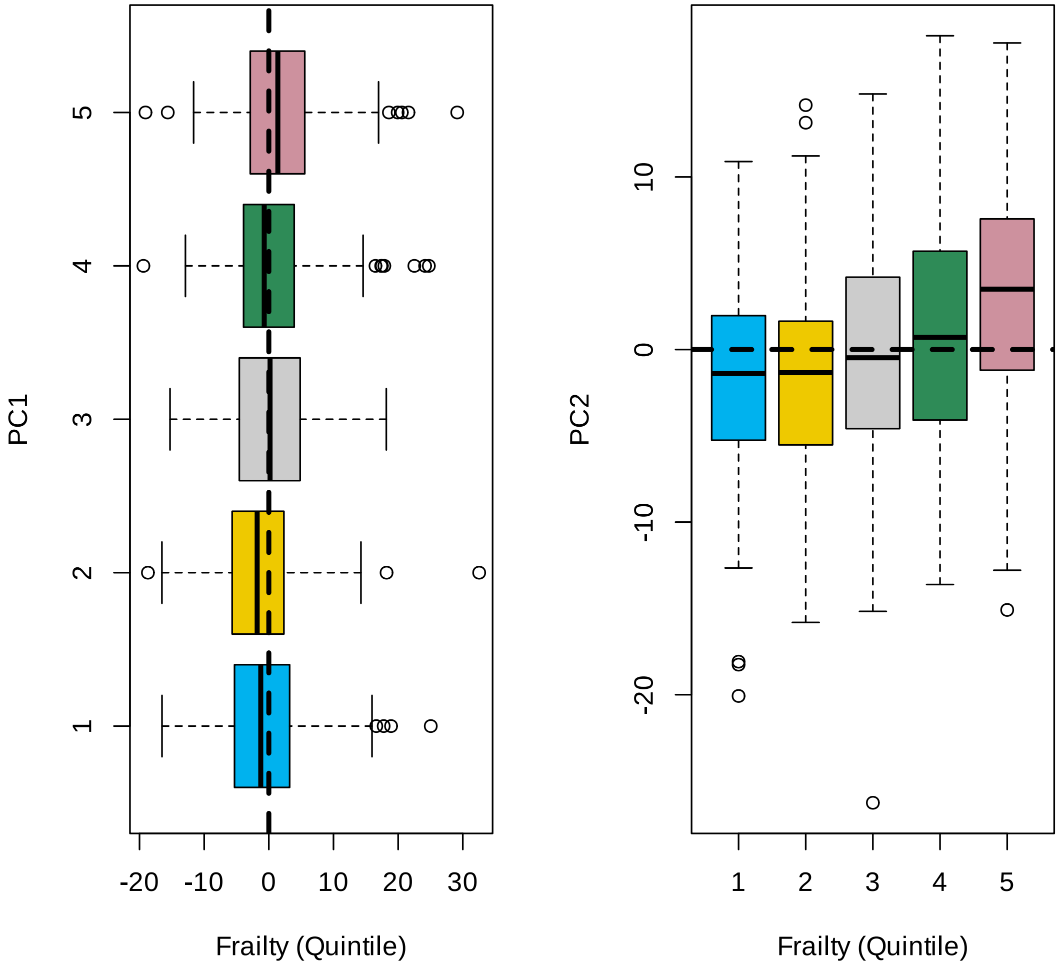
```
Outcome <- as.factor(frailty$self_quantile)
mets_mat <- as.matrix(metabolites[,2:ncol(metabolites)])
prots_mat <- as.matrix(proteins[,2:ncol(proteins)])
clin_mat <- as.matrix(clinical[,2:ncol(clinical)])
```

[21]:

```
pca.mets <- pca(mets_mat, ncomp = 2, scale = TRUE)

plotIndiv(pca.mets, group = Outcome, ind.names = FALSE,
          legend = TRUE,
          title = 'Metabolites, PCA comp 1 - 2')
```

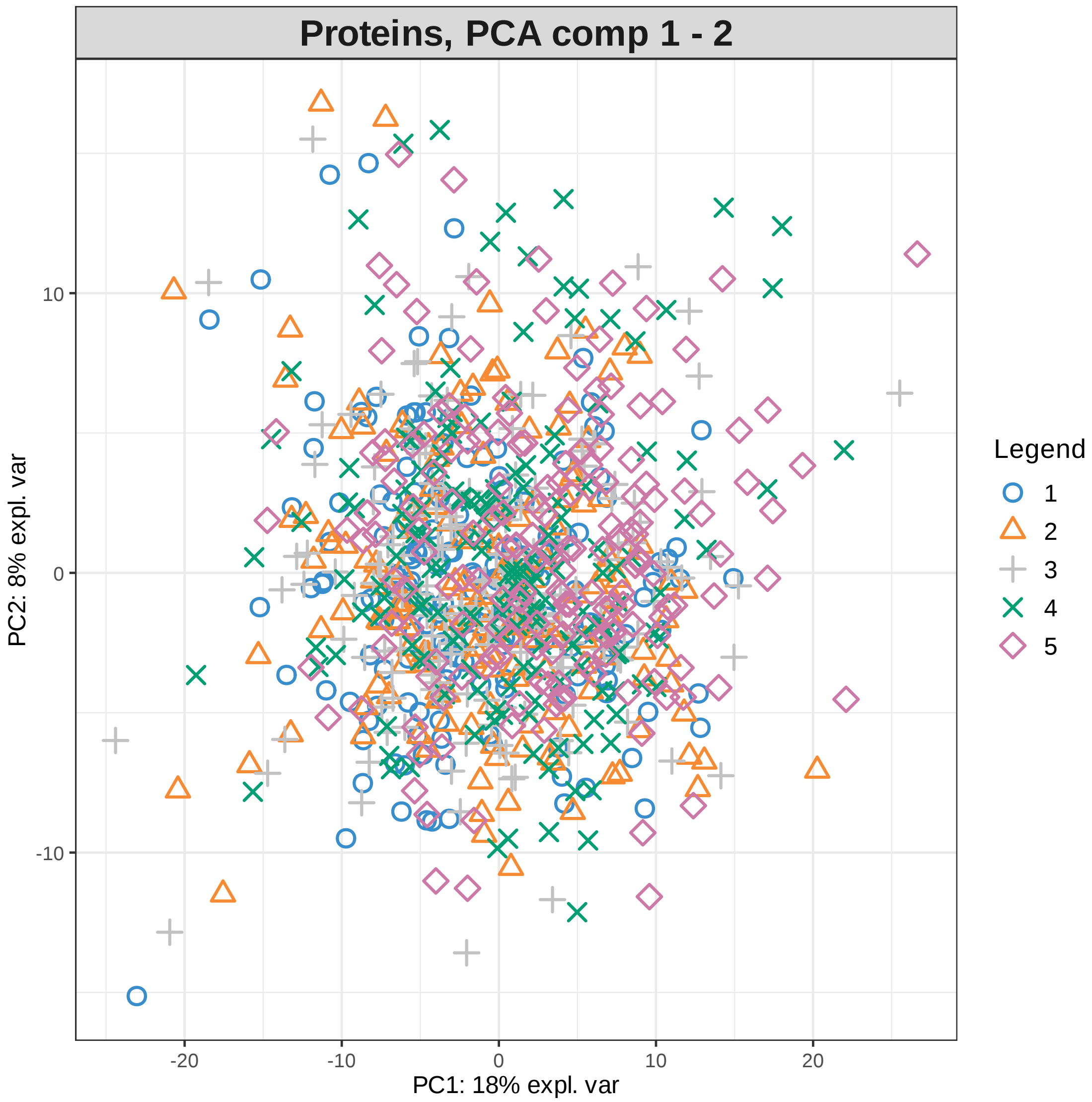


[22]:
```
par(mfrow=c(1,2))
boxplot(split(pca.mets$variates$X[,'PC1'], Outcome),
        ylab="PC1",xlab="Frailty (Quintile)",
        col=c('deepskyblue2','gold2','gray80','seagreen','pink3'))
abline(v=0,lty=2,lwd=3)

boxplot(split(pca.mets$variates$X[,'PC2'], Outcome),
        ylab="PC2",xlab="Frailty (Quintile)",
        col=c('deepskyblue2','gold2','gray80','seagreen','pink3'))
abline(h=0,lty=2,lwd=3)
```



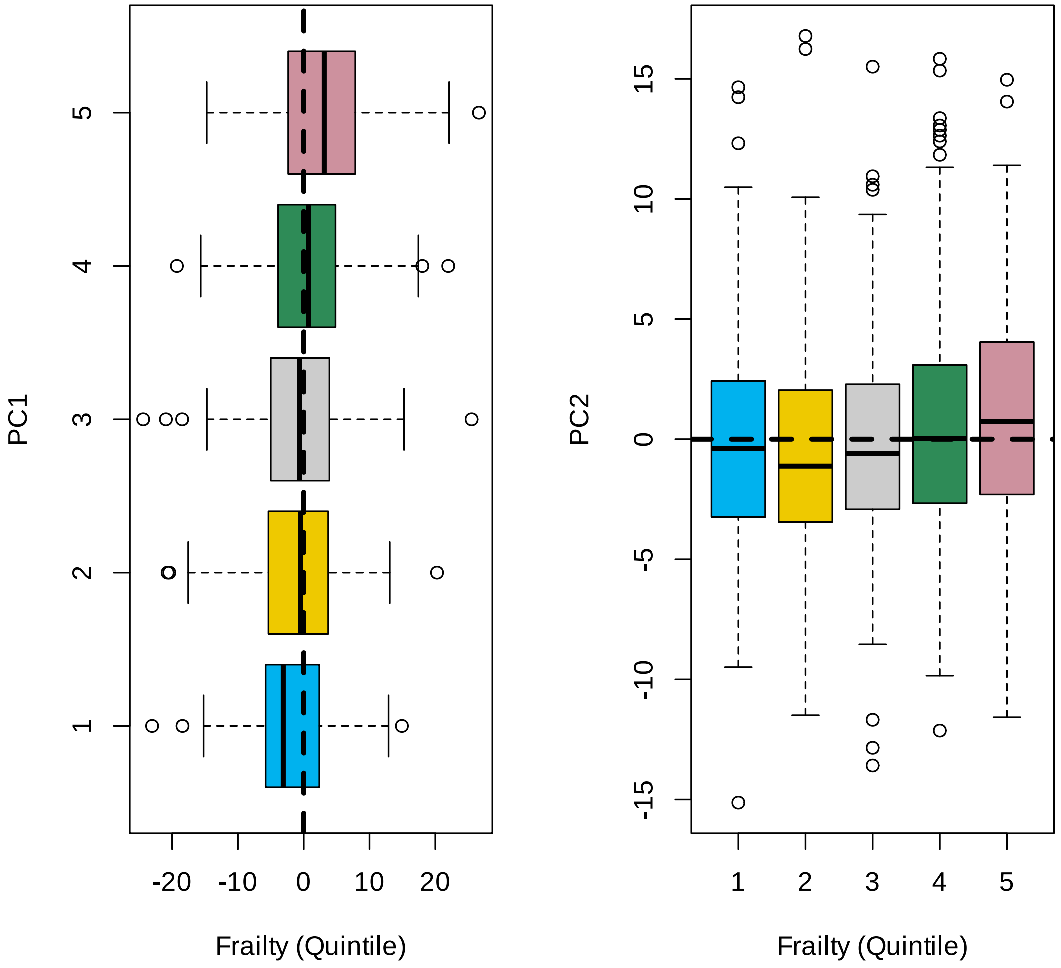
[23]:

```
pca.prots <- pca(prots_mat, ncomp = 2, scale = TRUE)

plotIndiv(pca.prots, group = Outcome, ind.names = FALSE,
          legend = TRUE,
          title = 'Proteins, PCA comp 1 - 2')
```



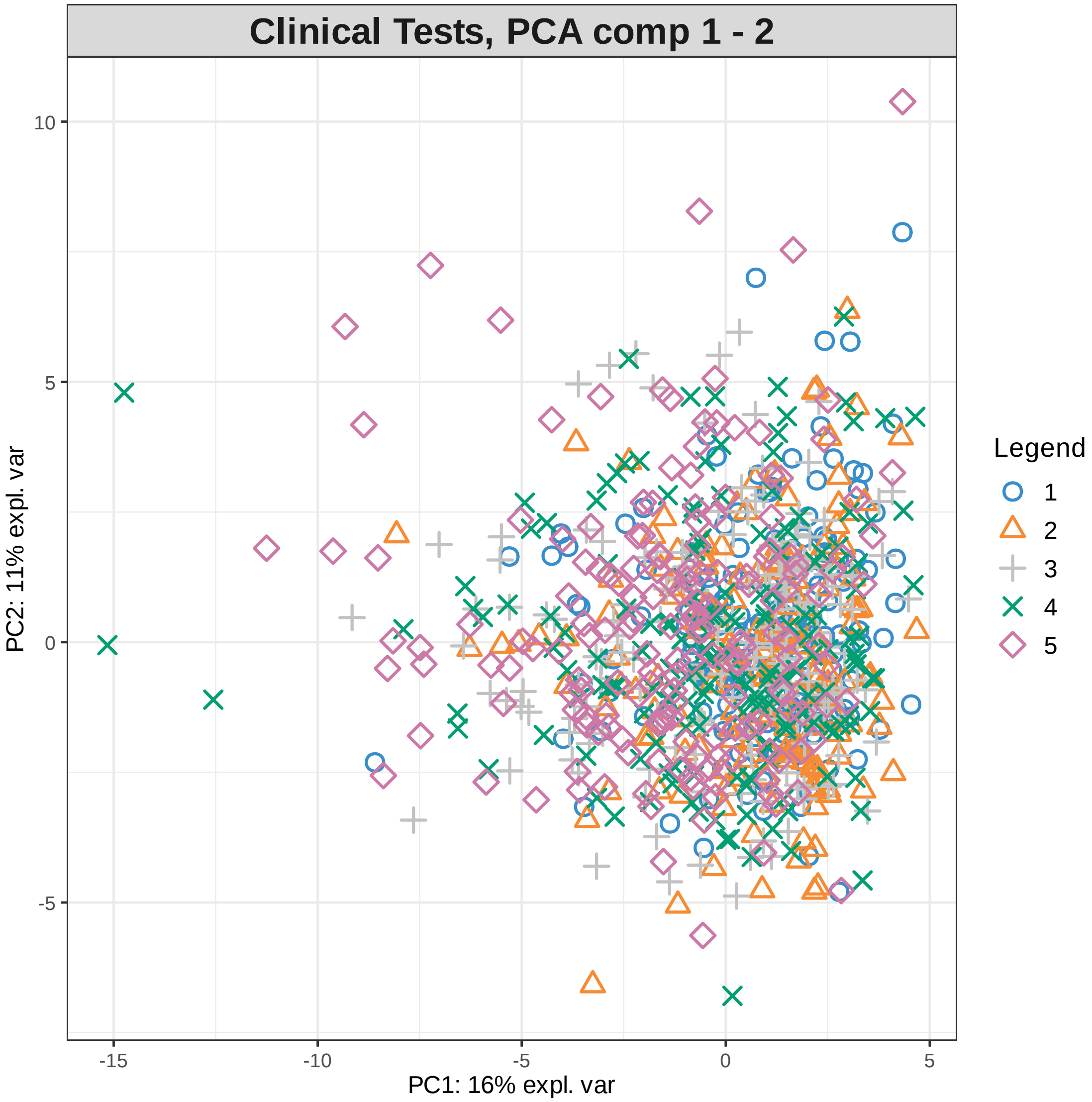
[24]:

```
par(mfrow=c(1,2))
boxplot(split(pca.prots$variates$X[,'PC1'], Outcome),
        ylab="PC1",xlab="Frailty (Quintile)",
        col=c('deepskyblue2','gold2','gray80','seagreen','pink3'))
abline(v=0,lty=2,lwd=3)

boxplot(split(pca.prots$variates$X[,'PC2'], Outcome),
        ylab="PC2",xlab="Frailty (Quintile)",
        col=c('deepskyblue2','gold2','gray80','seagreen','pink3'))
abline(h=0,lty=2,lwd=3)
```

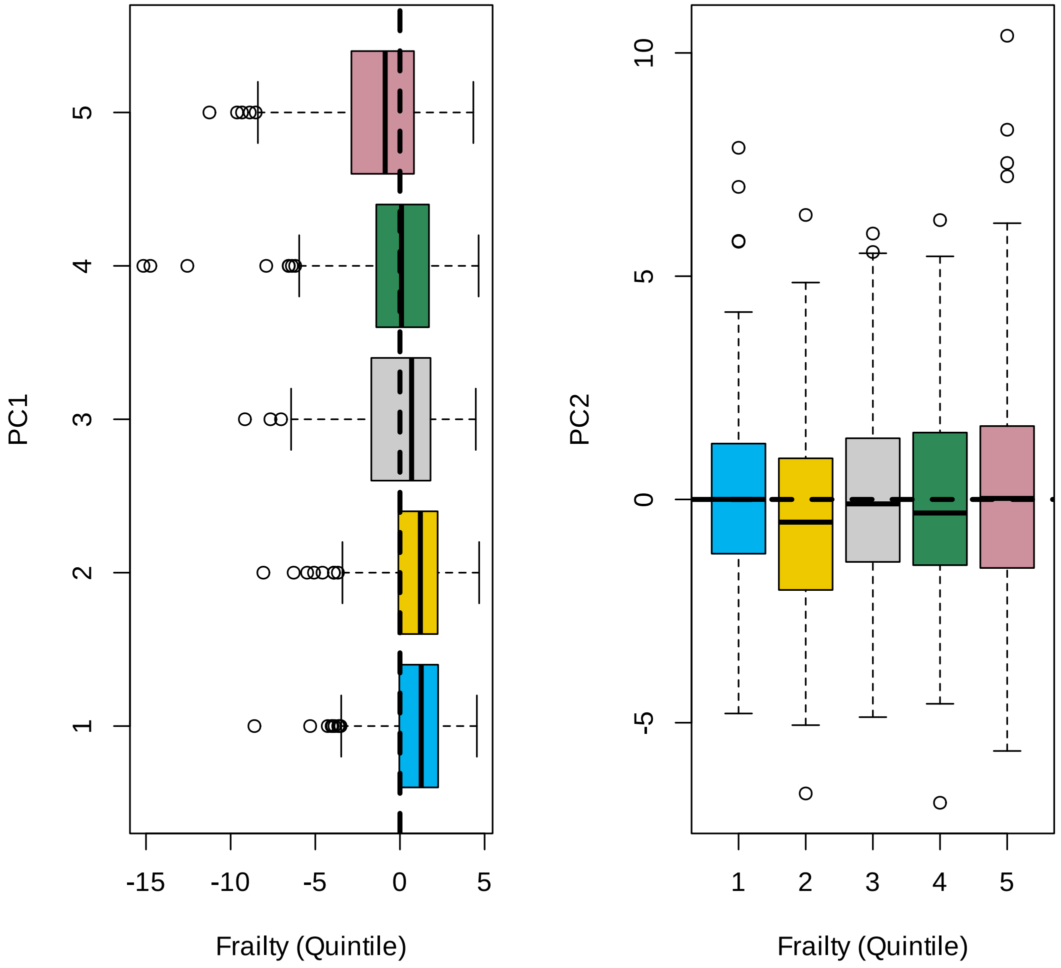


[25]:

```r
pca.clin <- pca(clin_mat, ncomp = 2, scale = TRUE)

plotIndiv(pca.clin, group = Outcome, ind.names = FALSE,
          legend = TRUE,
          title = 'Clinical Tests, PCA comp 1 - 2')
```



```
[26]: par(mfrow=c(1,2))
      boxplot(split(pca.clin$variates$X[,'PC1'], Outcome),
              ylab="PC1",xlab="Frailty (Quintile)",
              col=c('deepskyblue2','gold2','gray80','seagreen','pink3'))
      abline(v=0,lty=2,lwd=3)

      boxplot(split(pca.clin$variates$X[,'PC2'], Outcome),
              ylab="PC2",xlab="Frailty (Quintile)",
              col=c('deepskyblue2','gold2','gray80','seagreen','pink3'))
      abline(h=0,lty=2,lwd=3)
```



3 Single ’Omics PLS-DA

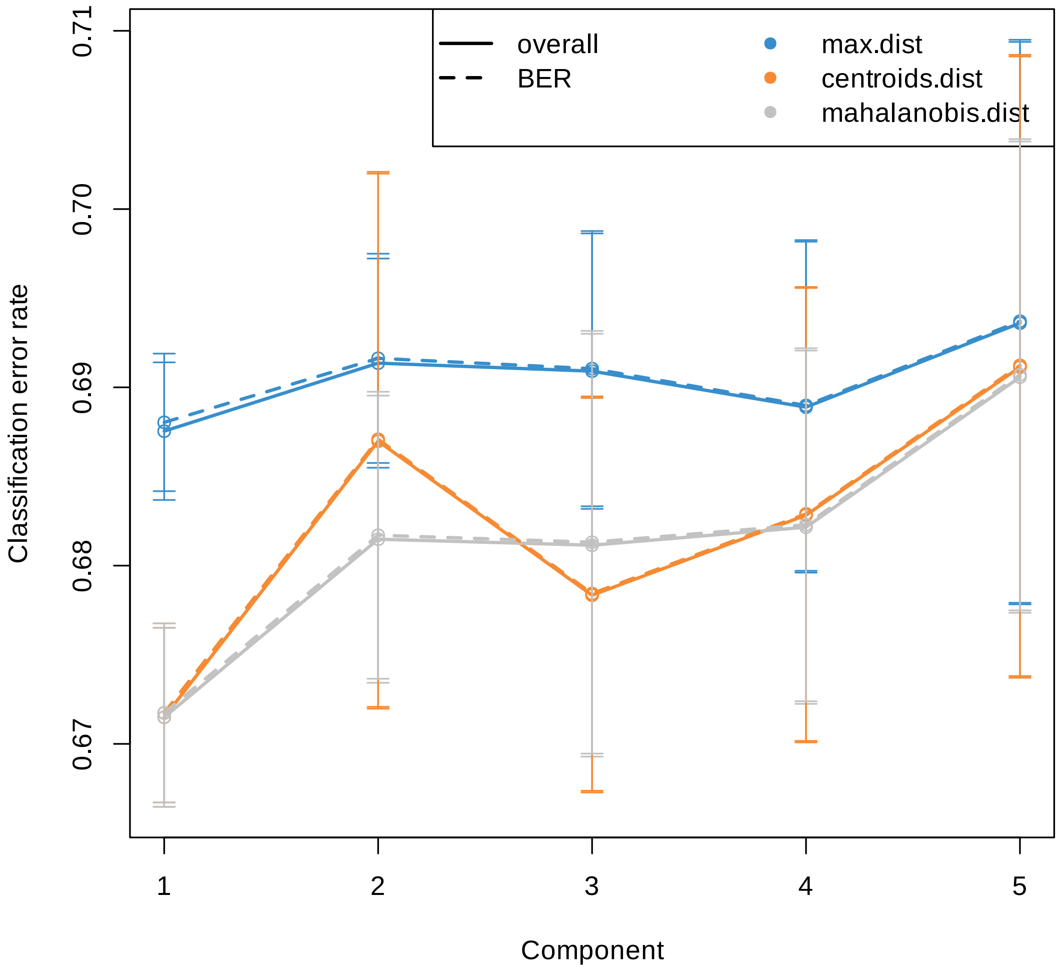
Exploratory data analysis with PCA (above) finds the axes on which the data is most spread out; it allows us to look at the spatial pattern of the data. The outcome labeling each point (quintiles of self-reported Frailty Index, shown by color and marker), however, is not used in PCA; we look at how the outcome correlates (visually) with the spatial pattern.

PLS-DA is similar to PCA, except that it is explicitly trying to spread the spatial pattern of the outcome, rather than the predictive features. The outcome is used to supervise which axis is chosen first, second, etc. This is a first look at how well each of the individual ’omics datasets informs us about the outcome. When we integrate the ’omics data together, we will be looking to take advantage of any differences in what each type of data tells us about the outcome.

3.1 Metabolomics PLS-DA

[27]:
```
plsda.met <- mixOmics::plsda(mets_mat, Outcome, ncomp = 5)

perf.plsda.met  <- mixOmics::perf(plsda.met, validation = 'Mfold', folds = 3,
                  progressBar = TRUE,
                  nrepeat = 10)   ### This is a low number of repeats that␣
 ↪should be increased for a better analysis. Its slow.

plot(perf.plsda.met, sd = TRUE, legend.position = 'horizontal')
```

```
comp 1
  |===============================================================================
==================================================================================
==================================| 100%
comp 2
  |===============================================================================
==================================================================================
==================================| 100%
comp 3
  |===============================================================================
==================================================================================
==================================| 100%
comp 4
  |===============================================================================
==================================================================================
==================================| 100%
comp 5
  |===============================================================================
==================================================================================
==================================| 100%
```



[28]:
```
# Not great BER
perf.plsda.met
```

```
Call:
 perf.mixo_plsda(object = plsda.met, validation = "Mfold", folds = 3, nrepeat =␣
 ↪10, progressBar = TRUE)

 Main numerical outputs:
 --------------------
 Error rate (overall or BER) for each component and for each distance: see␣
 ↪object$error.rate
```

```
Error rate per class, for each component and for each distance: see␣
↪object$error.rate.class
Prediction values for each component: see object$predict
Classification of each sample, for each component and for each distance: see␣
↪object$class
AUC values: see object$auc if auc = TRUE

Visualisation Functions:
--------------------
plot
```

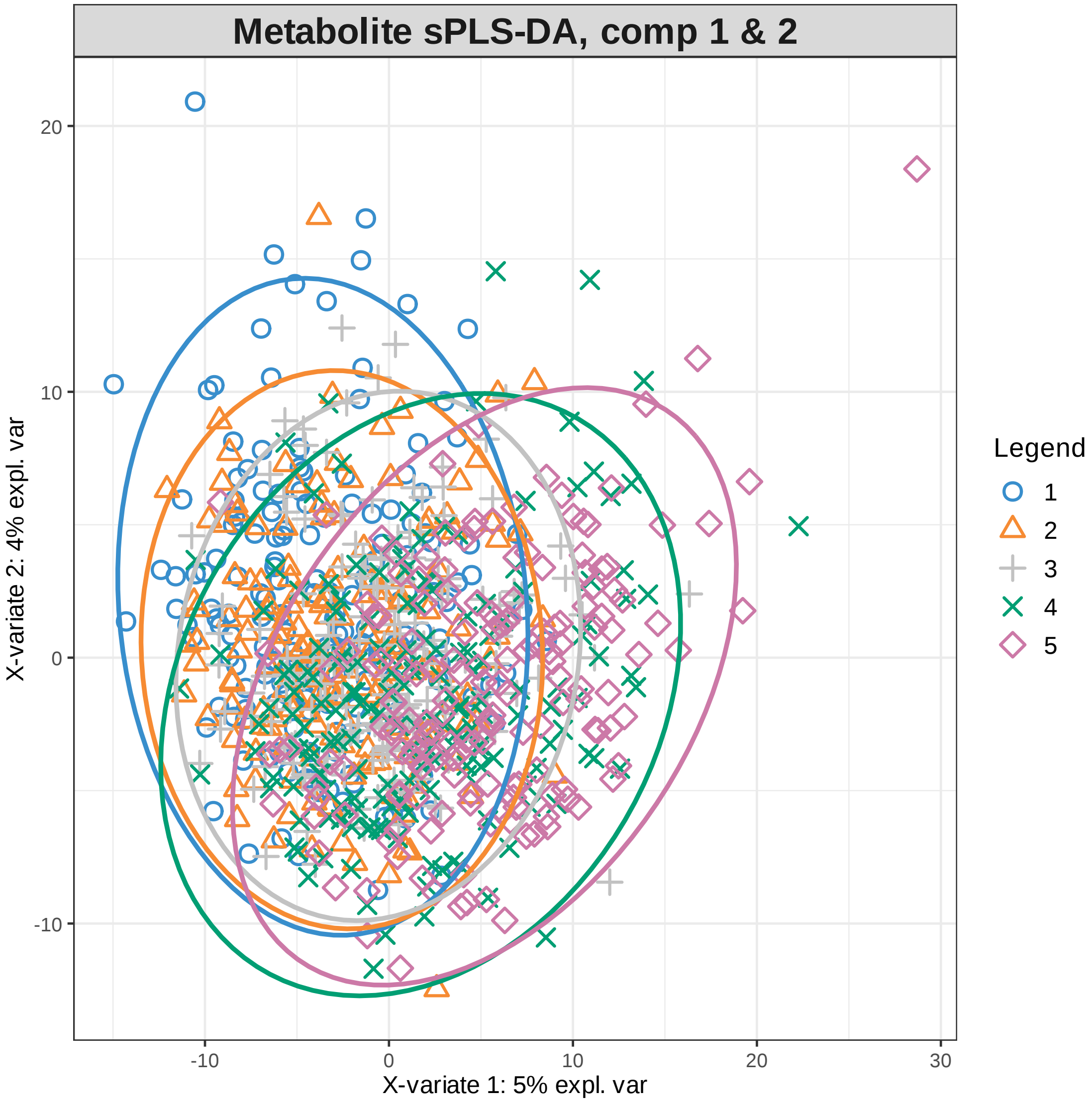
[29]:

```
print(perf.plsda.met$error.rate.class,digits=3)
```

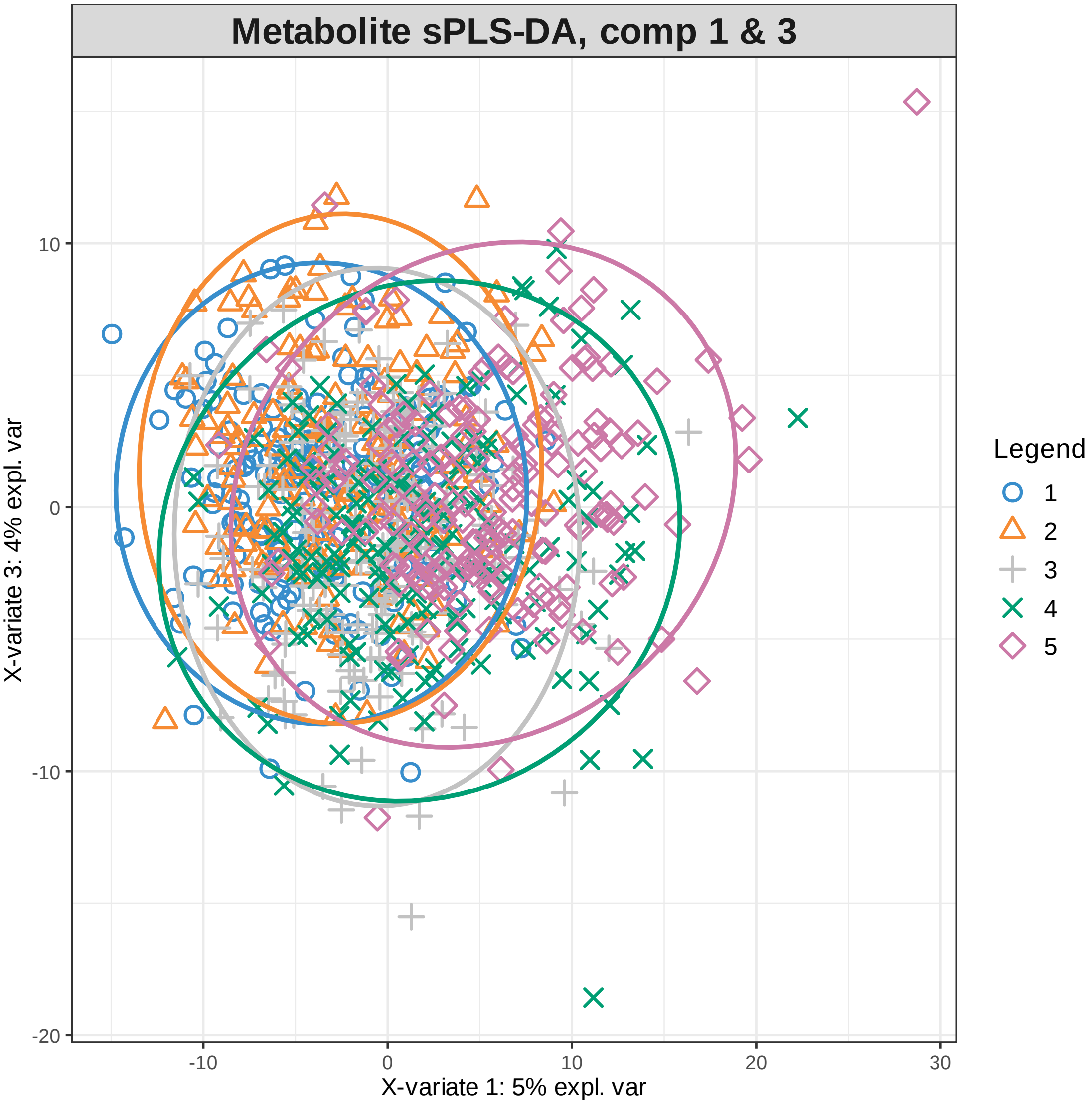
```
$max.dist
  comp1 comp2 comp3 comp4 comp5
1 0.249 0.451 0.567 0.604 0.602
2 1.000 0.848 0.778 0.787 0.805
3 1.000 0.967 0.922 0.898 0.875
4 0.999 0.897 0.769 0.741 0.749
5 0.192 0.296 0.419 0.416 0.438

$centroids.dist
  comp1 comp2 comp3 comp4 comp5
1 0.457 0.595 0.596 0.618 0.605
2 0.885 0.766 0.734 0.742 0.754
3 0.778 0.823 0.806 0.797 0.802
4 0.806 0.805 0.772 0.776 0.792
5 0.433 0.447 0.484 0.482 0.503

$mahalanobis.dist
  comp1 comp2 comp3 comp4 comp5
1 0.457 0.483 0.534 0.559 0.570
2 0.885 0.826 0.788 0.802 0.821
3 0.778 0.876 0.875 0.872 0.875
4 0.806 0.852 0.783 0.754 0.766
5 0.433 0.371 0.427 0.425 0.421
```

[30]:

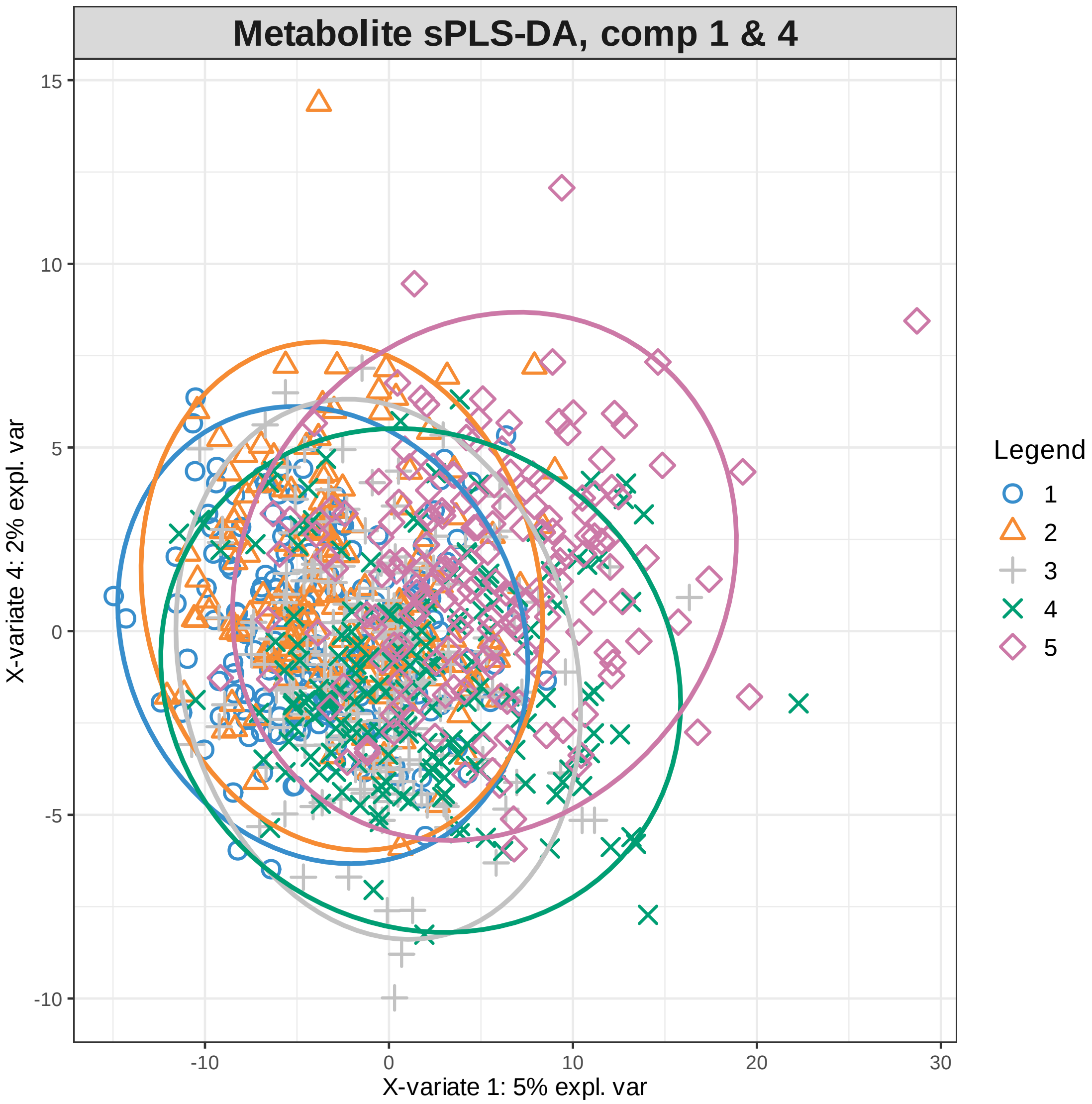
```
plotIndiv(plsda.met, comp = c(1,2), # plot samples from final model
          group = Outcome, ind.names = FALSE, # colour by class label
          ellipse = TRUE, legend = TRUE, # include 95% confidence ellipse
          title = 'Metabolite sPLS-DA, comp 1 & 2')
```



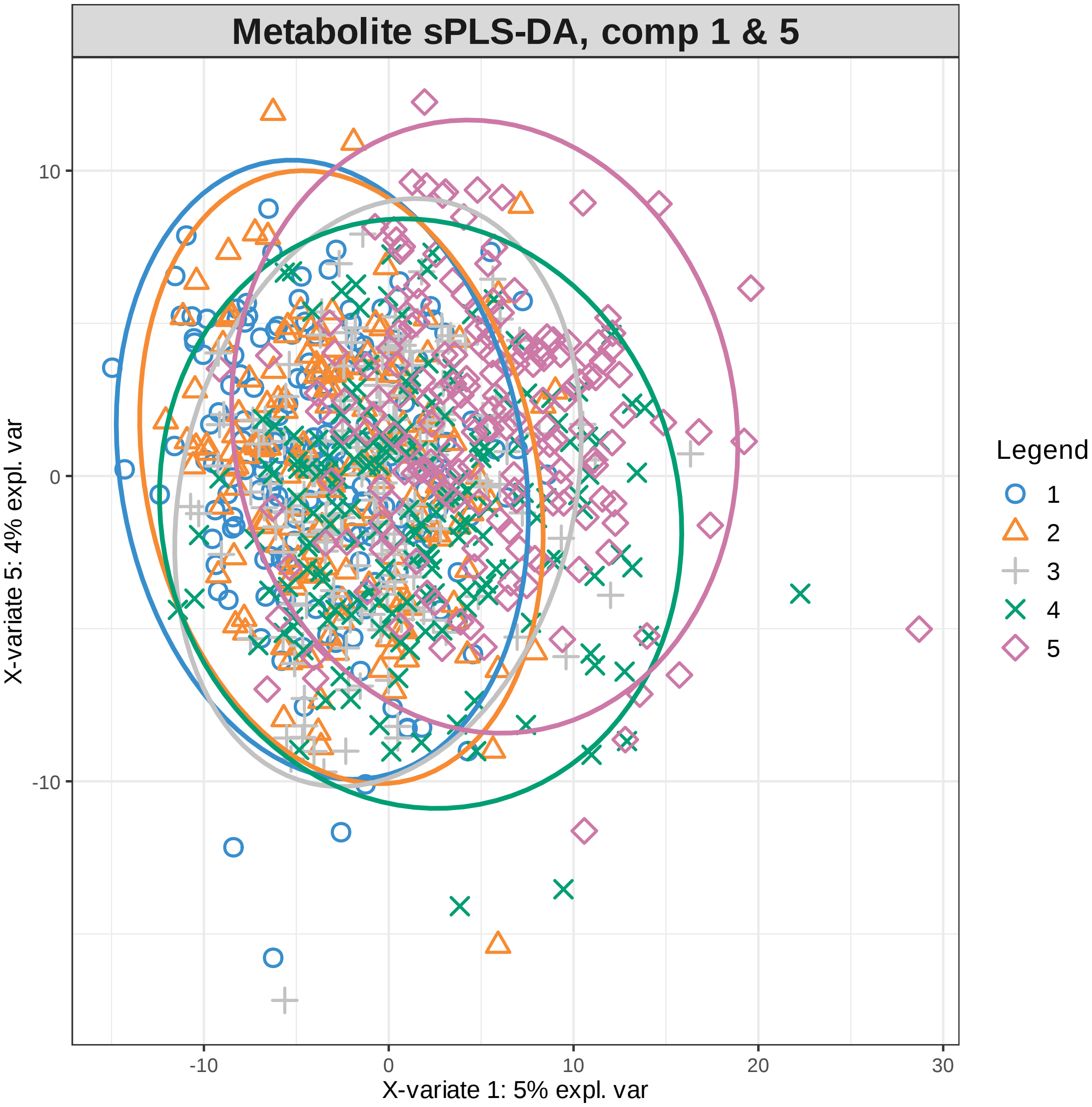
```
[31]: plotIndiv(plsda.met, comp = c(1,3), # plot samples from final model
                group = Outcome, ind.names = FALSE, # colour by class label
                ellipse = TRUE, legend = TRUE, # include 95% confidence ellipse
                title = 'Metabolite sPLS-DA, comp 1 & 3')
```



```
[32]: plotIndiv(plsda.met, comp = c(1,4), # plot samples from final model
                group = Outcome, ind.names = FALSE, # colour by class label
                ellipse = TRUE, legend = TRUE, # include 95% confidence ellipse
                title = 'Metabolite sPLS-DA, comp 1 & 4')
```

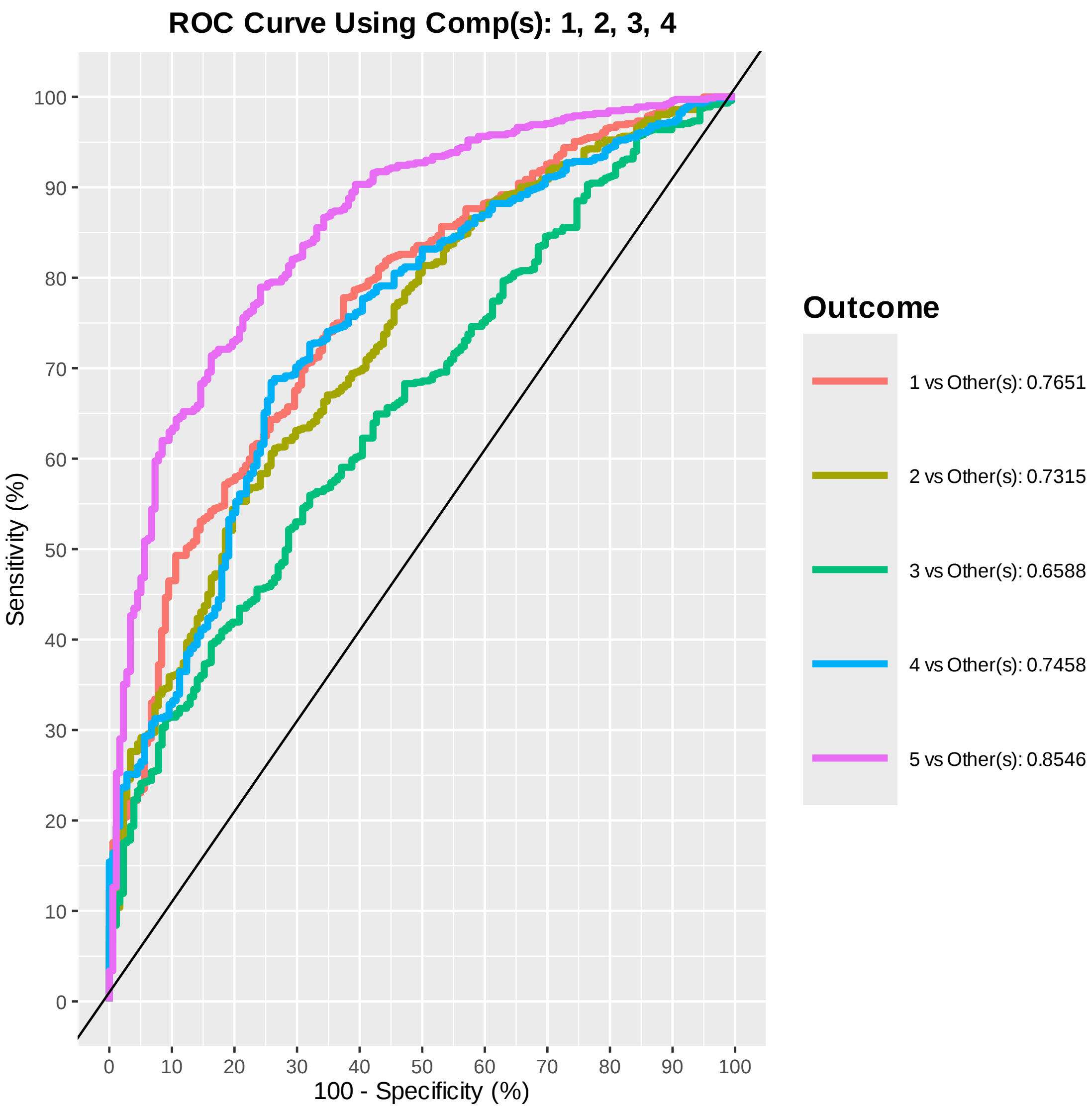


```
[33]: plotIndiv(plsda.met, comp = c(1,5), # plot samples from final model
                group = Outcome, ind.names = FALSE, # colour by class label
                ellipse = TRUE, legend = TRUE, # include 95% confidence ellipse
                title = 'Metabolite sPLS-DA, comp 1 & 5')
```



[34]:

```
# Component 4 appears to add to the ability to separate Q4 from Q3 and Q5; the␣
↪value of Component 5 is less clear
met.auroc <- auroc(plsda.met, roc.comp = 4, print = FALSE)
```

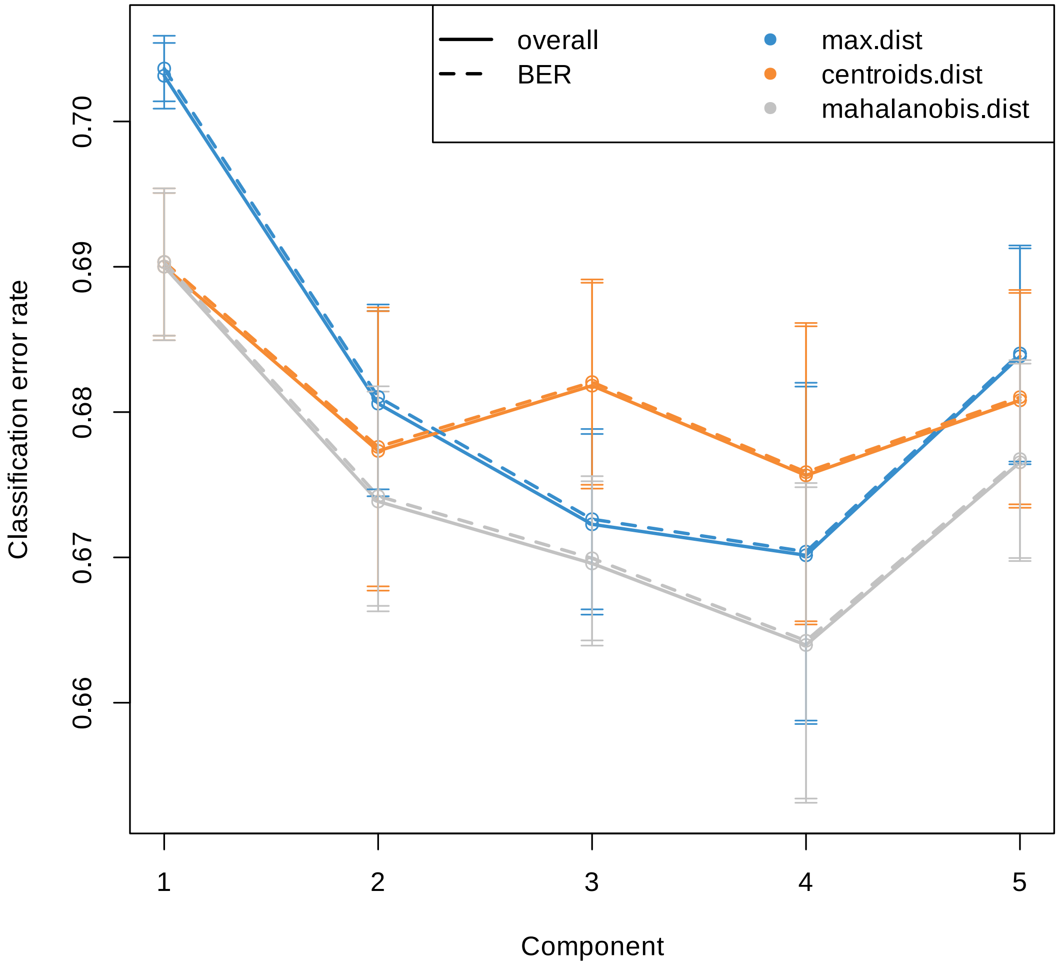


3.2 Proteomics PLS-DA

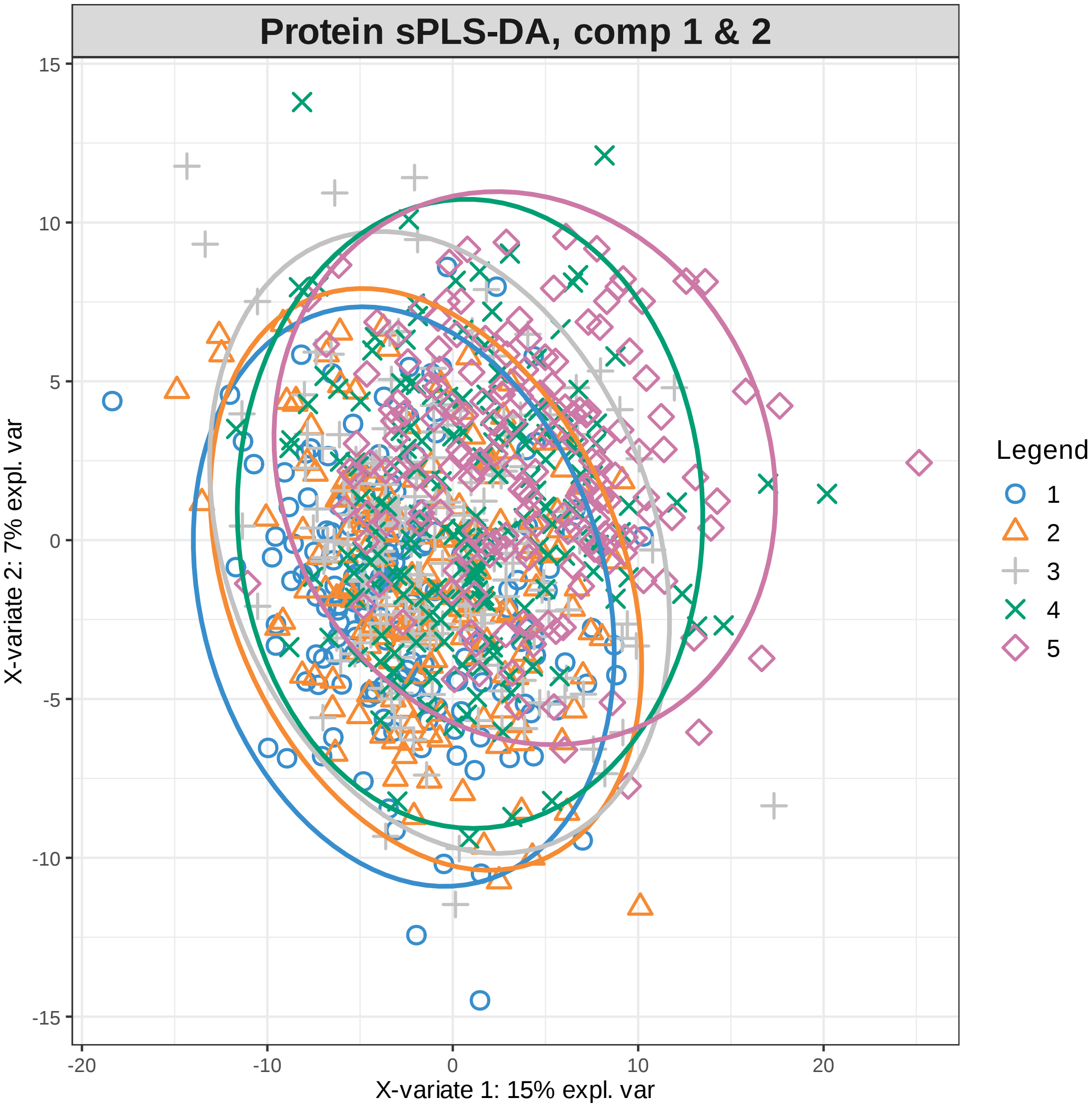
[35]:

```
plsda.prots <- mixOmics::plsda(prots_mat, Outcome, ncomp = 5)

perf.plsda.prots  <- mixOmics::perf(plsda.prots, validation = 'Mfold', folds =␣
  ↪3,
                  progressBar = TRUE,
                  nrepeat = 10)   ### This is a low number of repeats that␣
  ↪should be increased for a better analysis. Its slow.

plot(perf.plsda.prots, sd = TRUE, legend.position = 'horizontal')
```

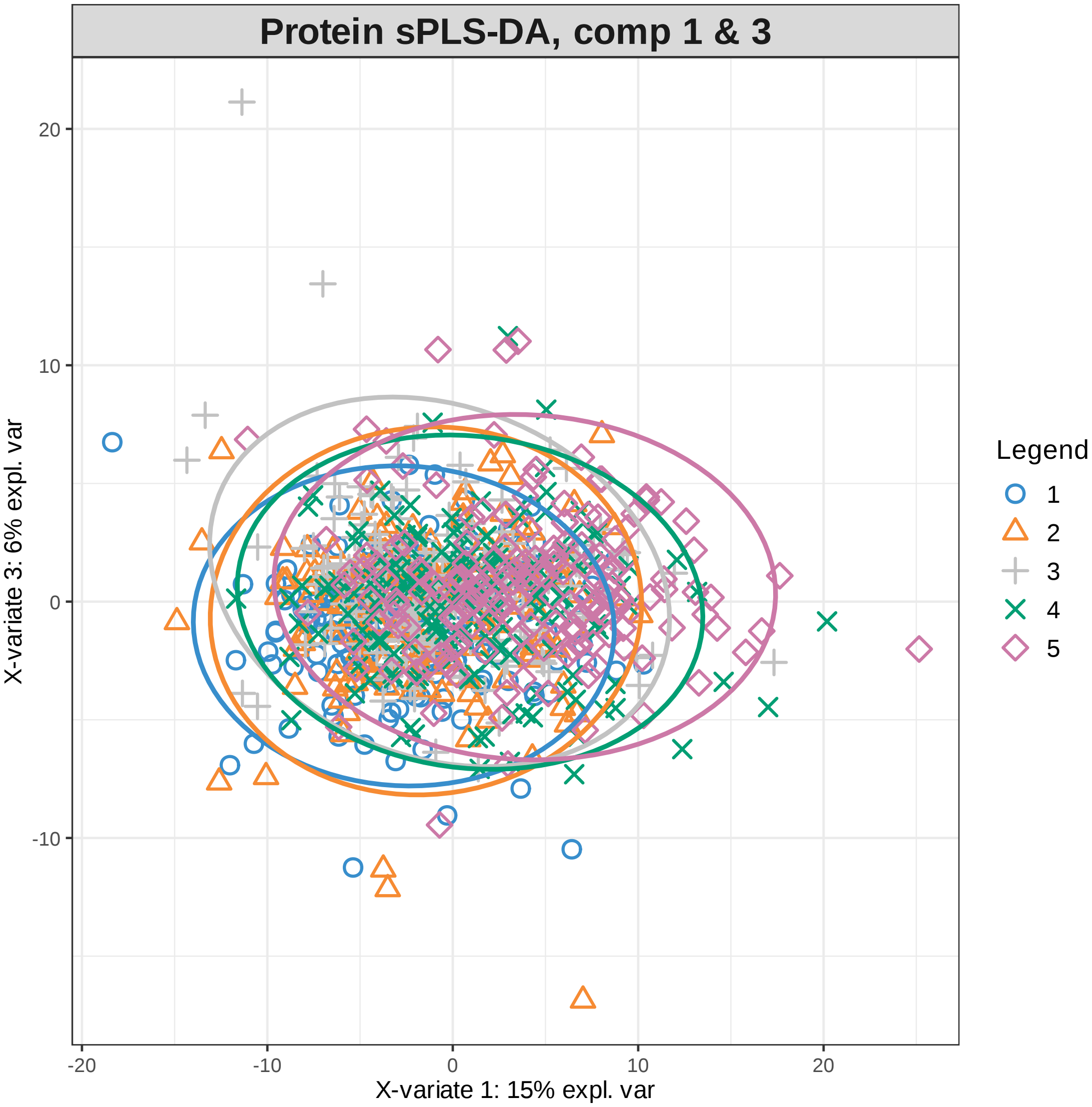
```
comp 1
  |=============================================================================
================================================================================
=================================| 100%
comp 2
  |=============================================================================
================================================================================
=================================| 100%
comp 3
  |=============================================================================
================================================================================
=================================| 100%
comp 4
  |=============================================================================
================================================================================
=================================| 100%
comp 5
  |=============================================================================
================================================================================
=================================| 100%
```



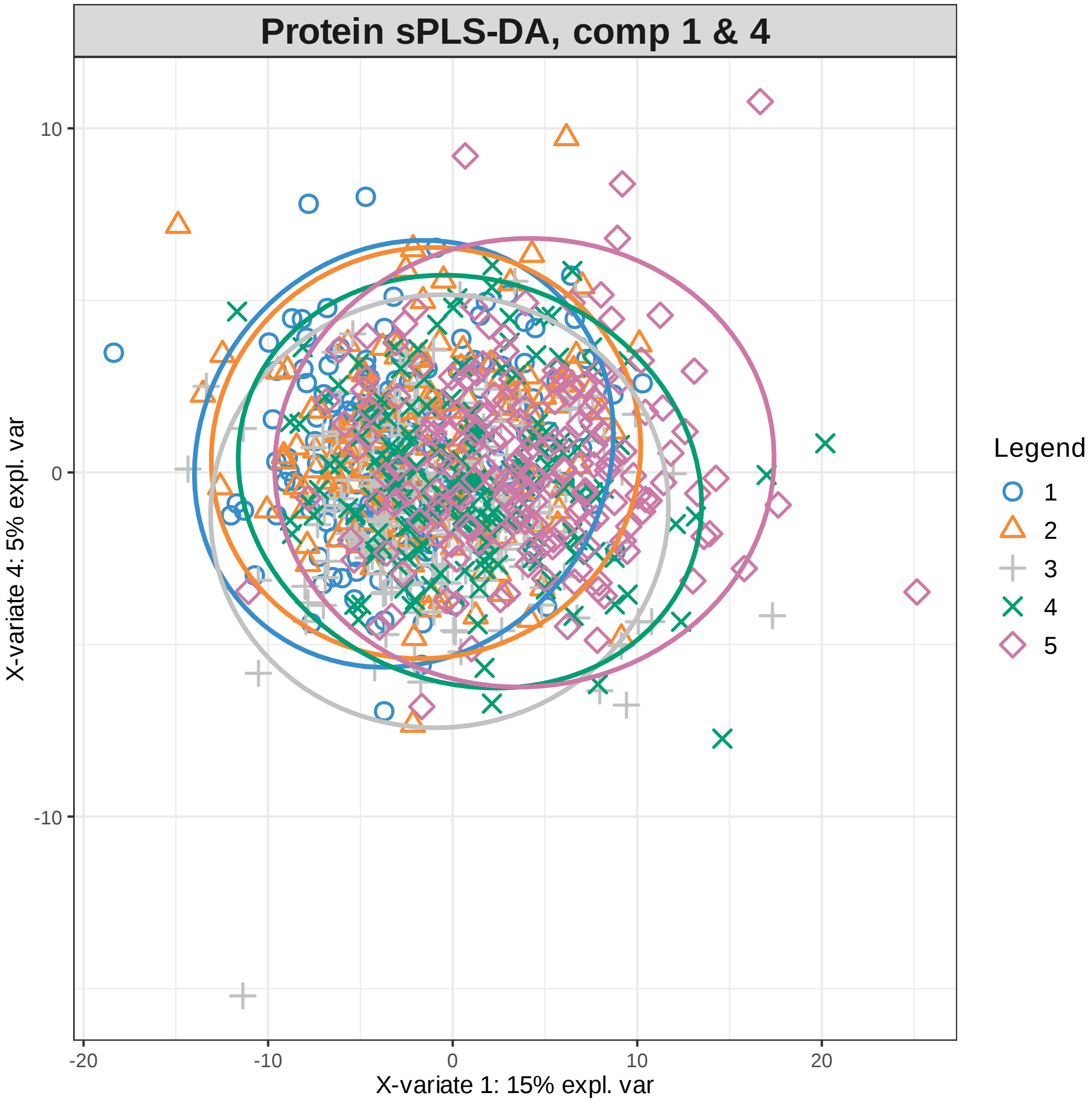
[36]:
```
plotIndiv(plsda.prots, comp = c(1,2), # plot samples from final model
          group = Outcome, ind.names = FALSE, # colour by class label
          ellipse = TRUE, legend = TRUE, # include 95% confidence ellipse
          title = 'Protein sPLS-DA, comp 1 & 2')
```



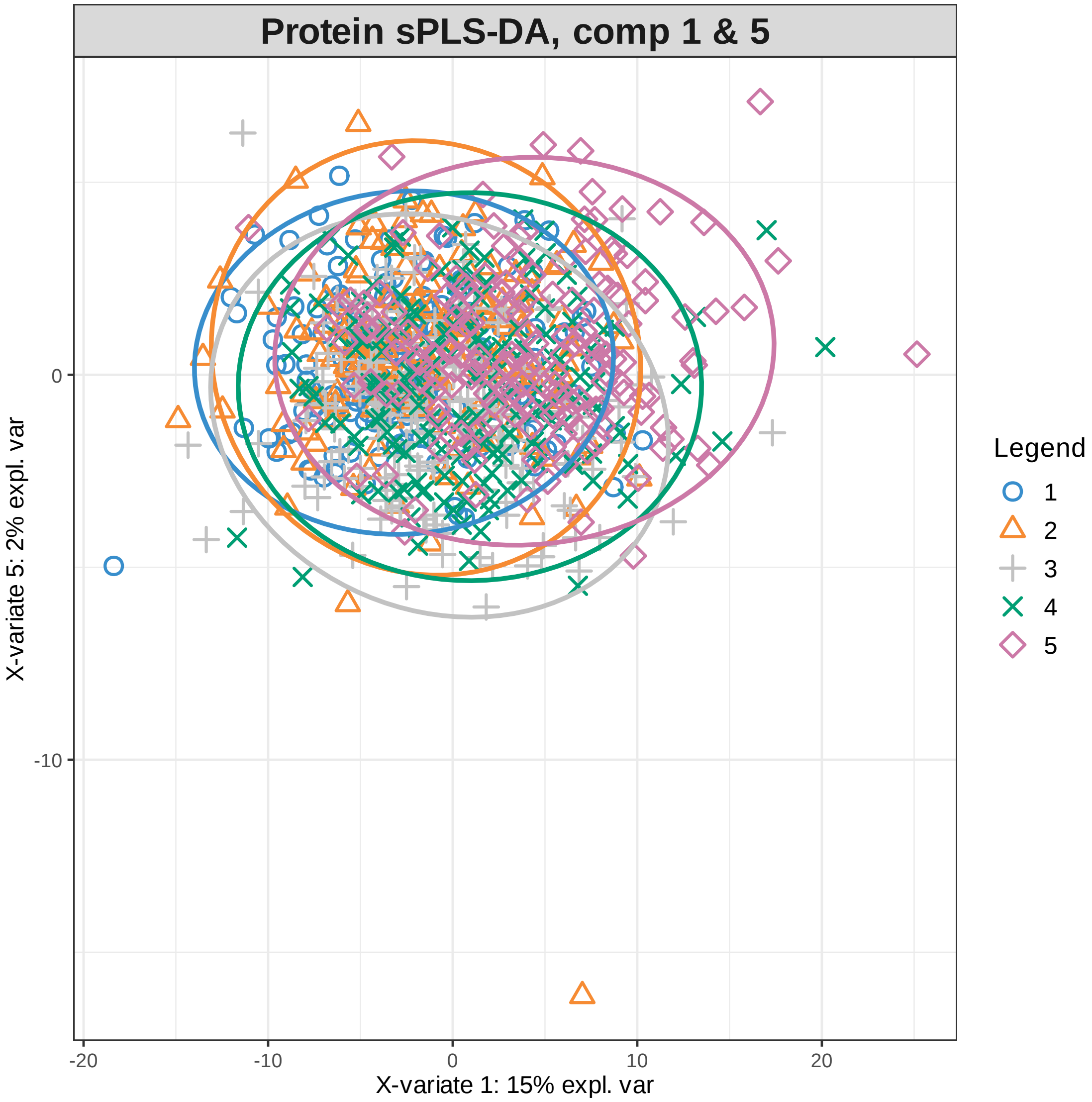
```
[37]: plotIndiv(plsda.prots, comp = c(1,3), # plot samples from final model
                group = Outcome, ind.names = FALSE, # colour by class label
                ellipse = TRUE, legend = TRUE, # include 95% confidence ellipse
                title = 'Protein sPLS-DA, comp 1 & 3')
```



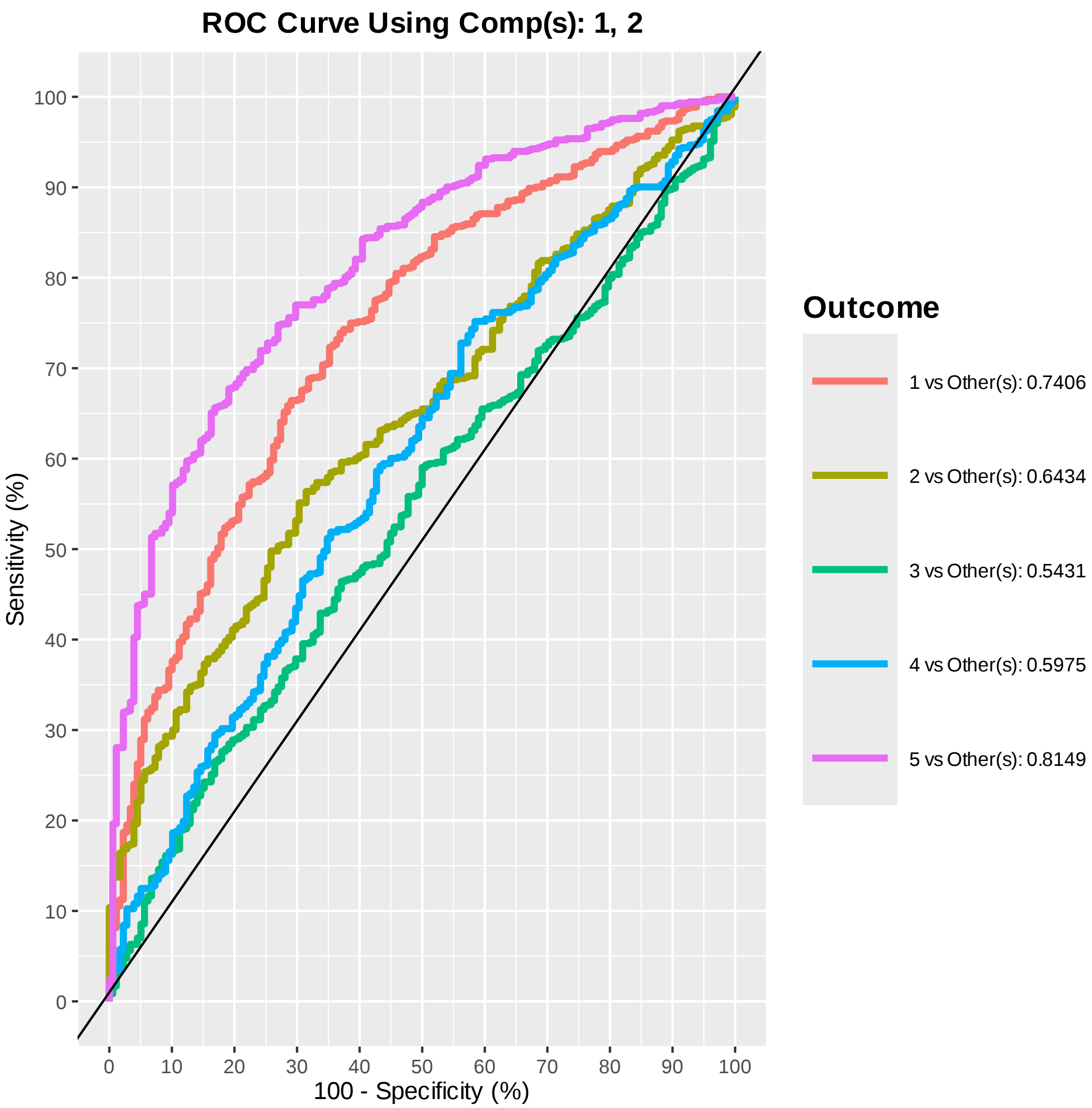
```
[38]: plotIndiv(plsda.prots, comp = c(1,4), # plot samples from final model
                group = Outcome, ind.names = FALSE, # colour by class label
                ellipse = TRUE, legend = TRUE, # include 95% confidence ellipse
                title = 'Protein sPLS-DA, comp 1 & 4')
```



```
[39]: plotIndiv(plsda.prots, comp = c(1,5), # plot samples from final model
                 group = Outcome, ind.names = FALSE, # colour by class label
                 ellipse = TRUE, legend = TRUE, # include 95% confidence ellipse
                 title = 'Protein sPLS-DA, comp 1 & 5')
```



```
[40]: prots.auroc <- auroc(plsda.prots, roc.comp = 2, print = FALSE)
```

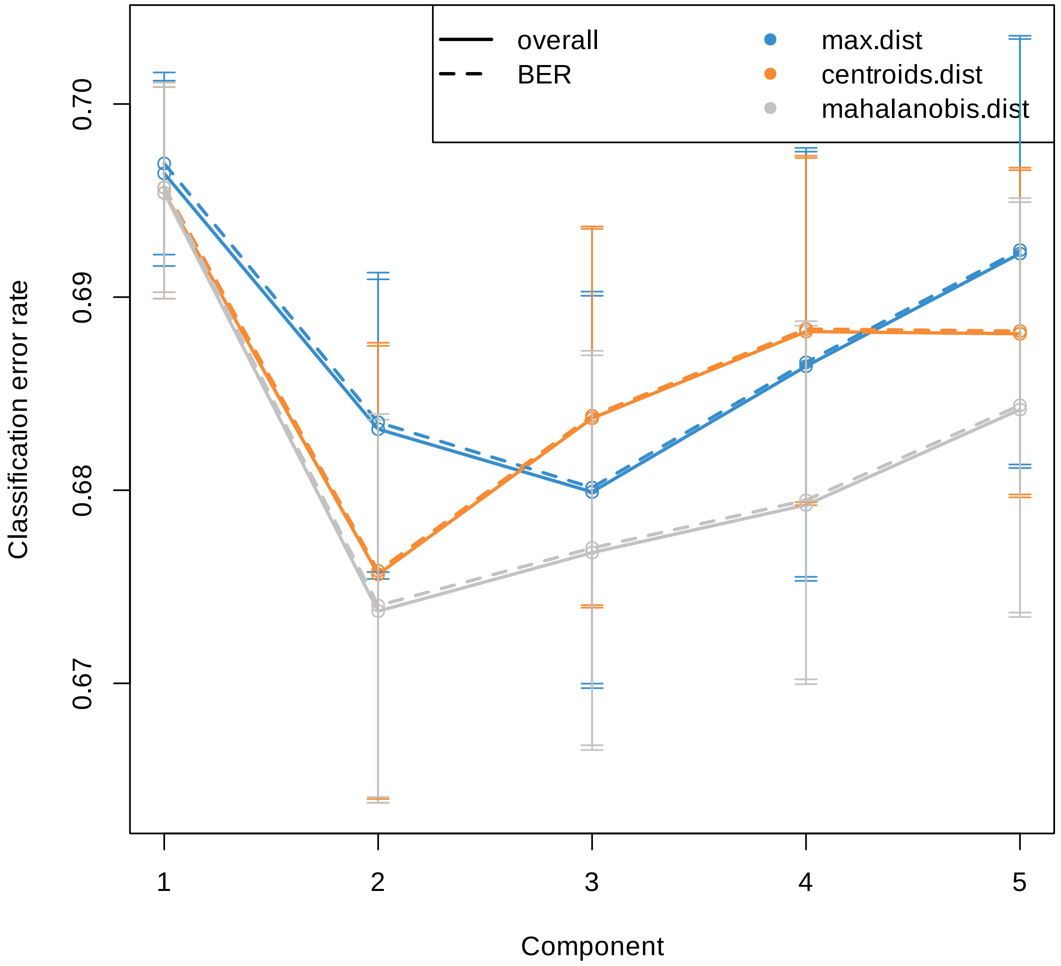


3.3 Clinical Tests PLS-DA

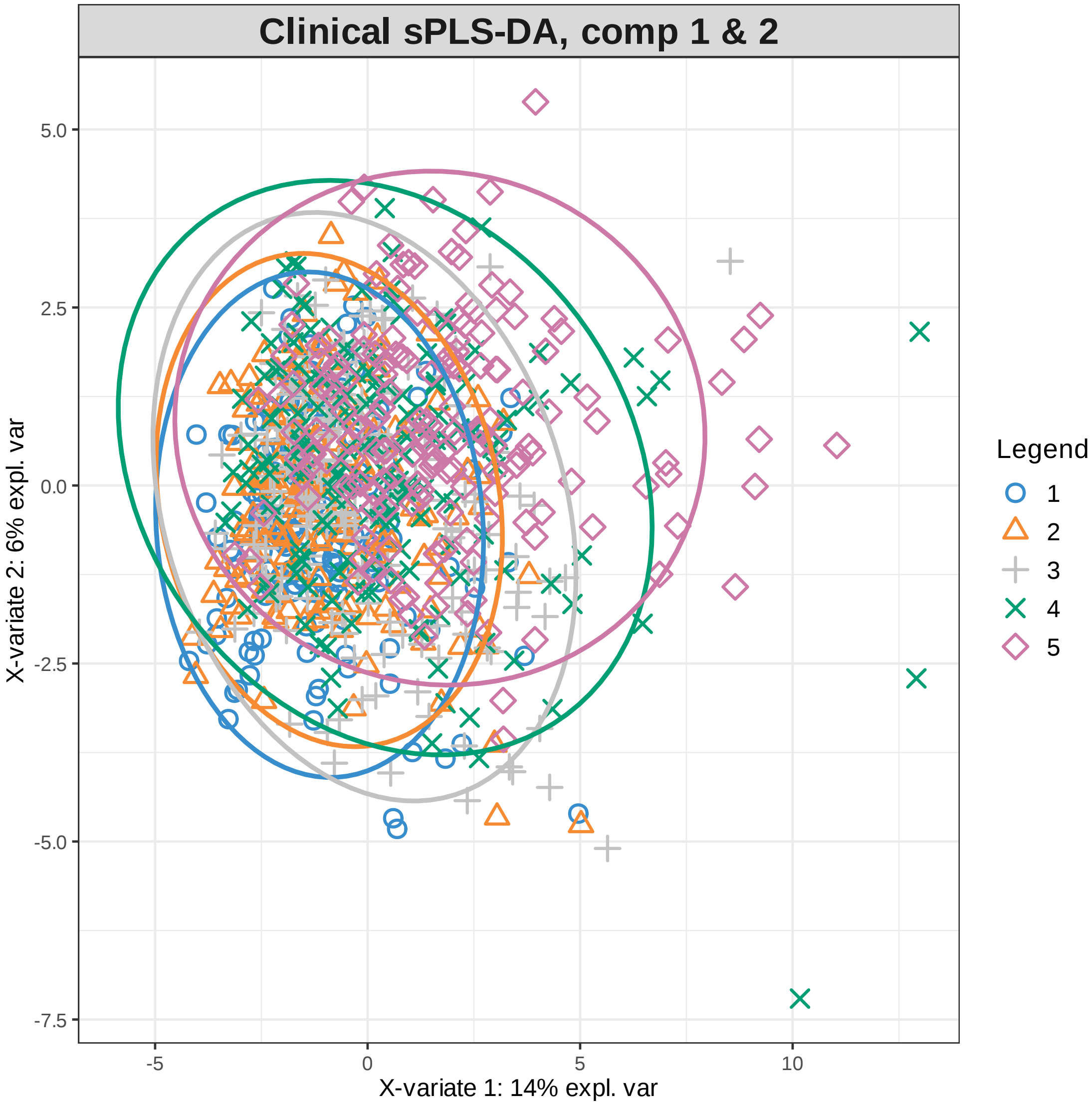
[41]:
```
plsda.clin <- mixOmics::plsda(clin_mat, Outcome, ncomp = 5)

perf.plsda.clin  <- mixOmics::perf(plsda.clin, validation = 'Mfold', folds = 3,
                    progressBar = TRUE,
                    nrepeat = 10)   ### This is a low number of repeats that␣
 ↪should be increased for a better analysis. Its slow.

plot(perf.plsda.clin, sd = TRUE, legend.position = 'horizontal')
```

```
comp 1
```

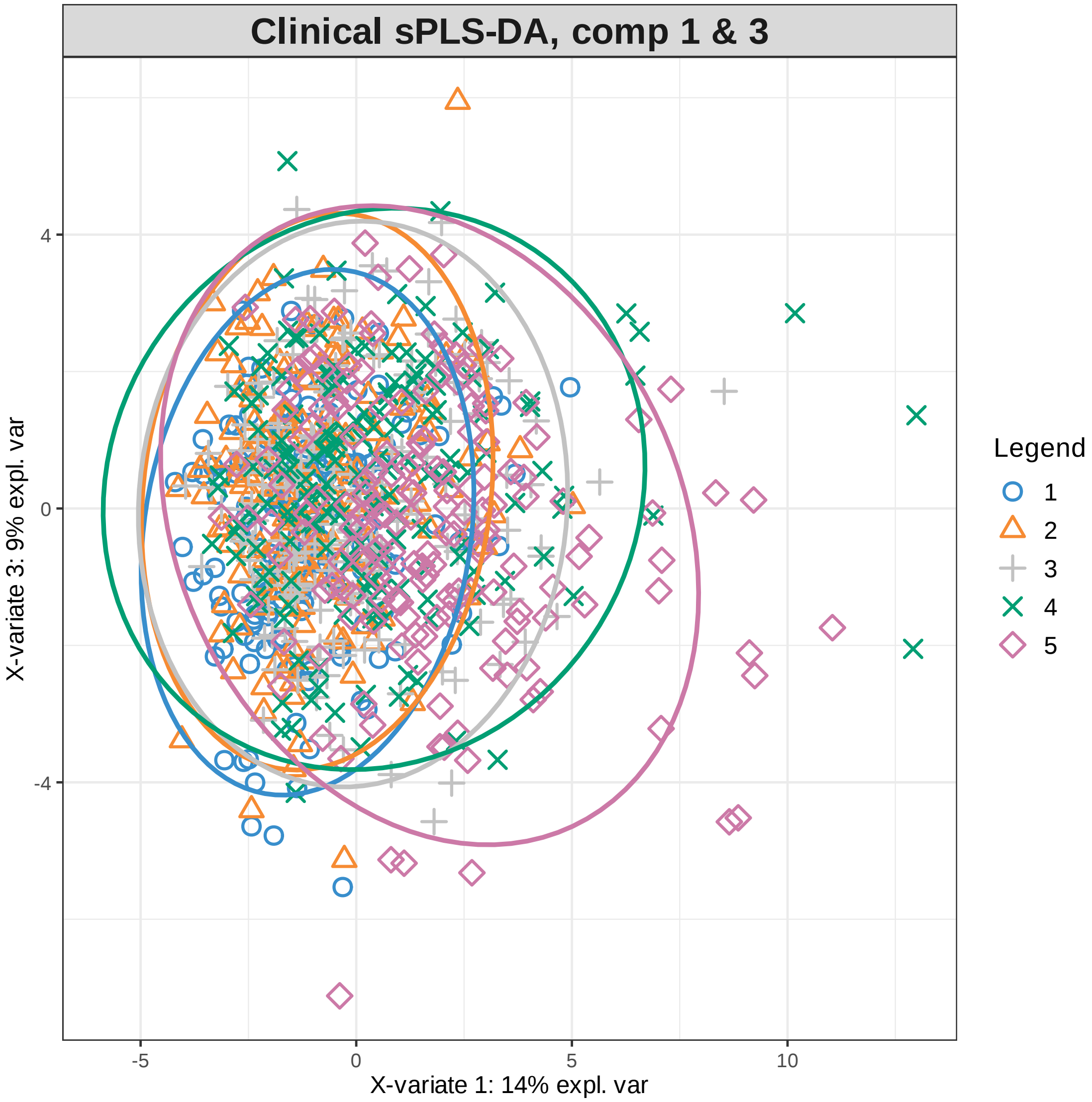
```
  |==============================================================================
=================================================================================
=================================| 100%
comp 2
  |==============================================================================
=================================================================================
=================================| 100%
comp 3
  |==============================================================================
=================================================================================
=================================| 100%
comp 4
  |==============================================================================
=================================================================================
=================================| 100%
comp 5
  |==============================================================================
=================================================================================
=================================| 100%
```



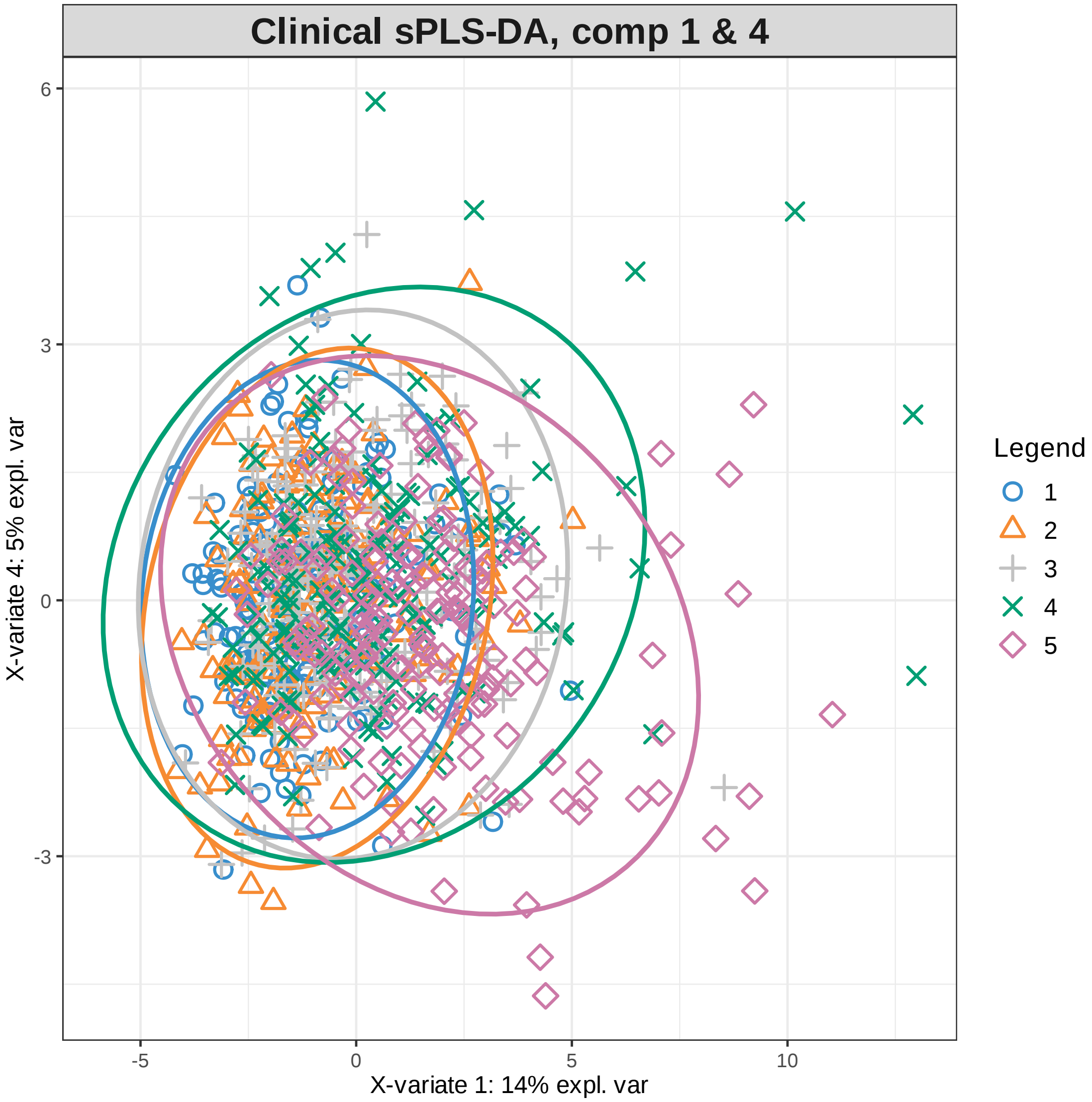
```
[42]: plotIndiv(plsda.clin, comp = c(1,2), # plot samples from final model
                group = Outcome, ind.names = FALSE, # colour by class label
                ellipse = TRUE, legend = TRUE, # include 95% confidence ellipse
                title = 'Clinical sPLS-DA, comp 1 & 2')
```



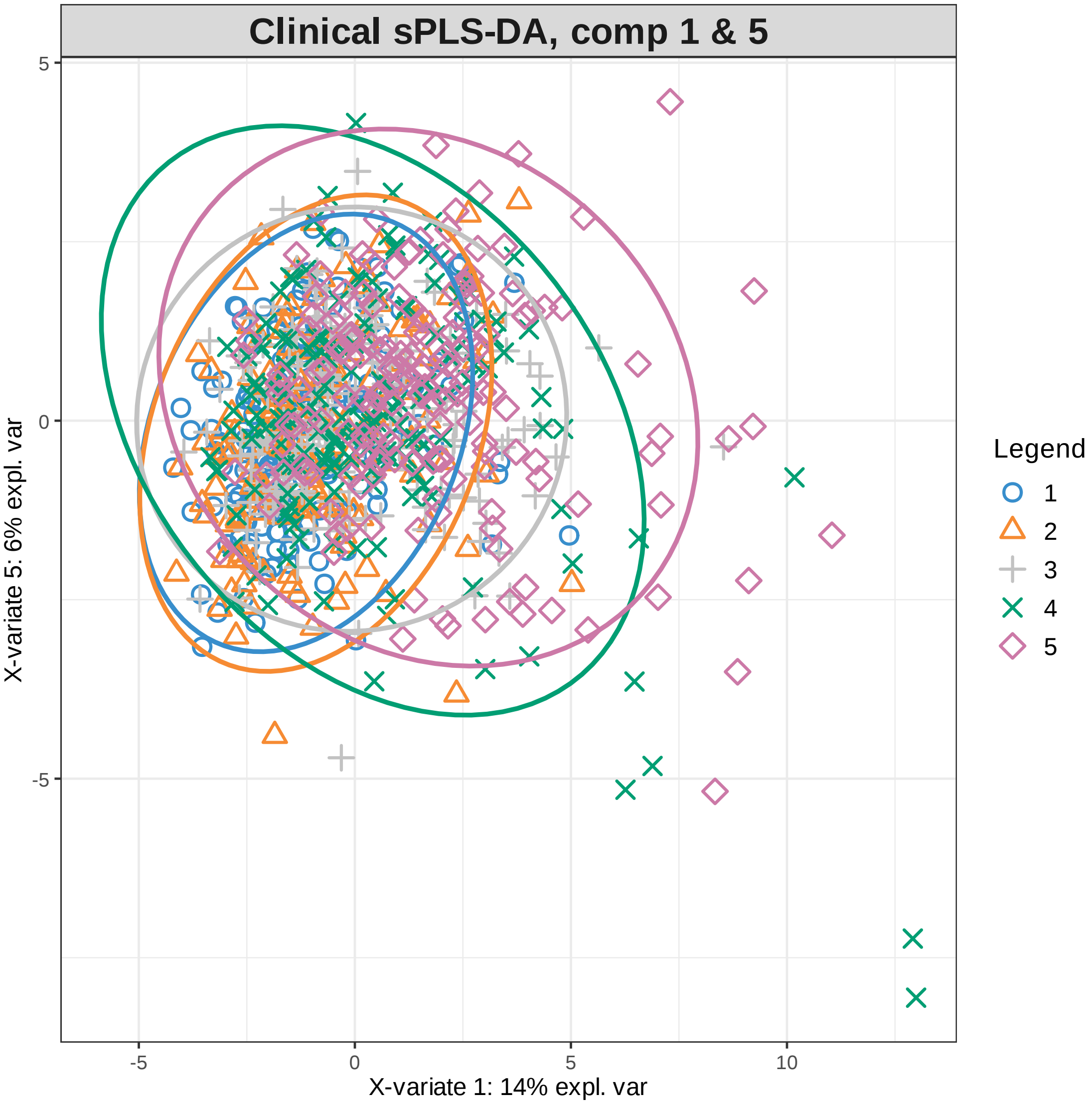
```
[43]: plotIndiv(plsda.clin, comp = c(1,3), # plot samples from final model
                group = Outcome, ind.names = FALSE, # colour by class label
                ellipse = TRUE, legend = TRUE, # include 95% confidence ellipse
                title = 'Clinical sPLS-DA, comp 1 & 3')
```



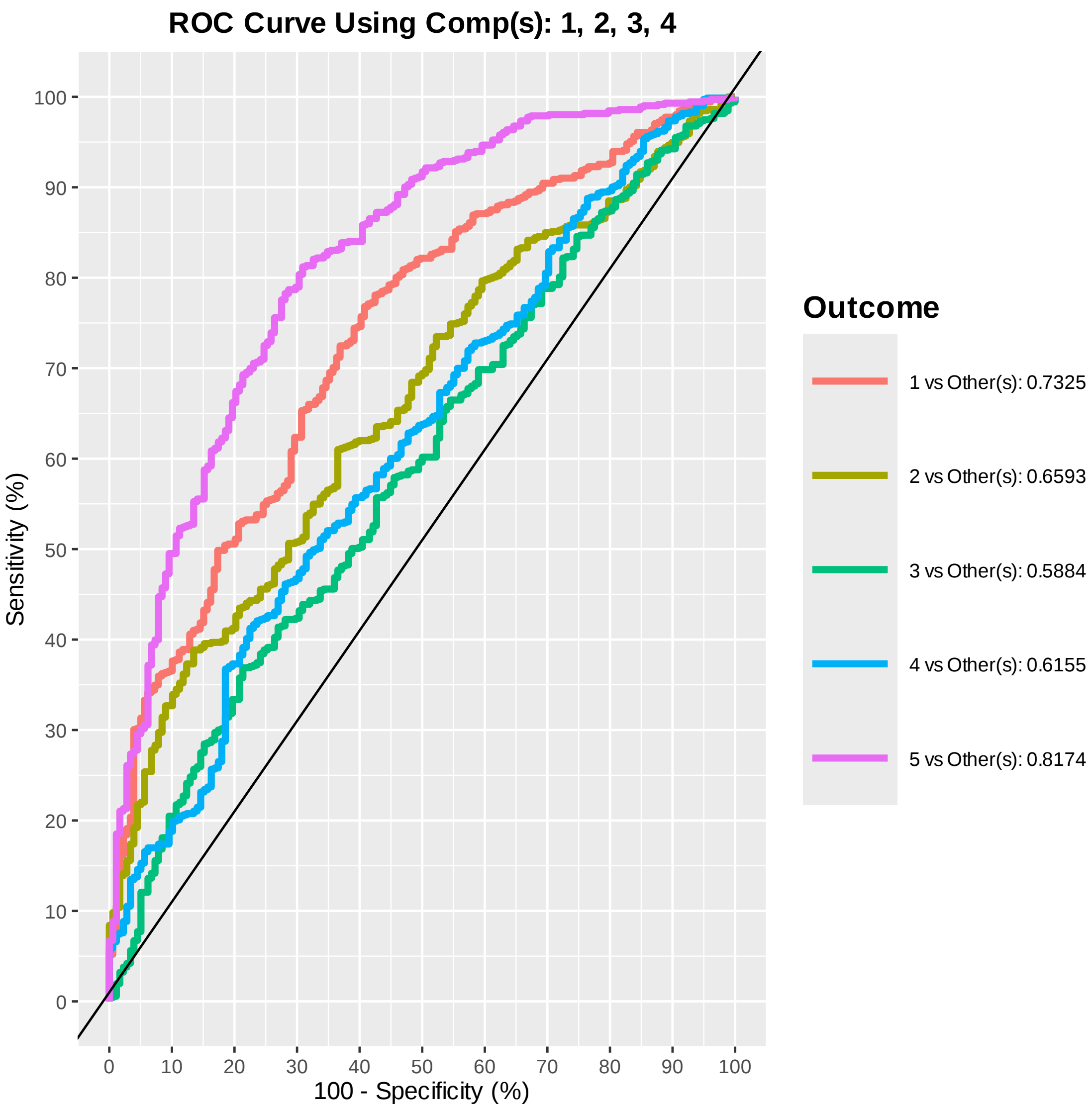
```
[44]: plotIndiv(plsda.clin, comp = c(1,4), # plot samples from final model
                group = Outcome, ind.names = FALSE, # colour by class label
                ellipse = TRUE, legend = TRUE, # include 95% confidence ellipse
                title = 'Clinical sPLS-DA, comp 1 & 4')
```



```
[45]: plotIndiv(plsda.clin, comp = c(1,5), # plot samples from final model
                group = Outcome, ind.names = FALSE, # colour by class label
                ellipse = TRUE, legend = TRUE, # include 95% confidence ellipse
                title = 'Clinical sPLS-DA, comp 1 & 5')
```



```
[46]: clin.auroc <- auroc(plsda.clin, roc.comp = 4, print = FALSE)
```



```
[47]: # Summary of the ability of each 'omic type to classify each quintile of␣
        ↪Self-Reported Frailty Index
      auroc.table <- cbind(
          met.auroc[['Comp4']][,'AUC'],
          prots.auroc[['Comp2']][,'AUC'],
          clin.auroc[['Comp3']][,'AUC'])
      dimnames(auroc.table) <- list(SelfFI = c('Q1','Q2','Q3','Q4','Q5'),
                   Block = c("Metabolites","Proteins","Clinical"))
      cat(noquote("Area Under ROC, predicting each quintile\n"))
      print(auroc.table)
```

```
Area Under ROC, predicting each quintile
      Block
```

```
SelfFI Metabolites Proteins Clinical
    Q1      0.7651   0.7406   0.7327
    Q2      0.7315   0.6434   0.6570
    Q3      0.6588   0.5431   0.5573
    Q4      0.7458   0.5975   0.5955
    Q5      0.8546   0.8149   0.8050
```

4 Full-strength DIABLO: Multiblock sPLS-DA

[48]:
```
X <- list(metabolite = mets_mat,
          protein = prots_mat,
          clinical = clin_mat)
```

[49]:
```
# Initial design with correlatin of .10
design <- matrix(0.1, ncol = length(X), nrow = length(X),
                dimnames = list(names(X), names(X)))
diag(design) <- 0
design
```

A matrix: 3 × 3 of type dbl

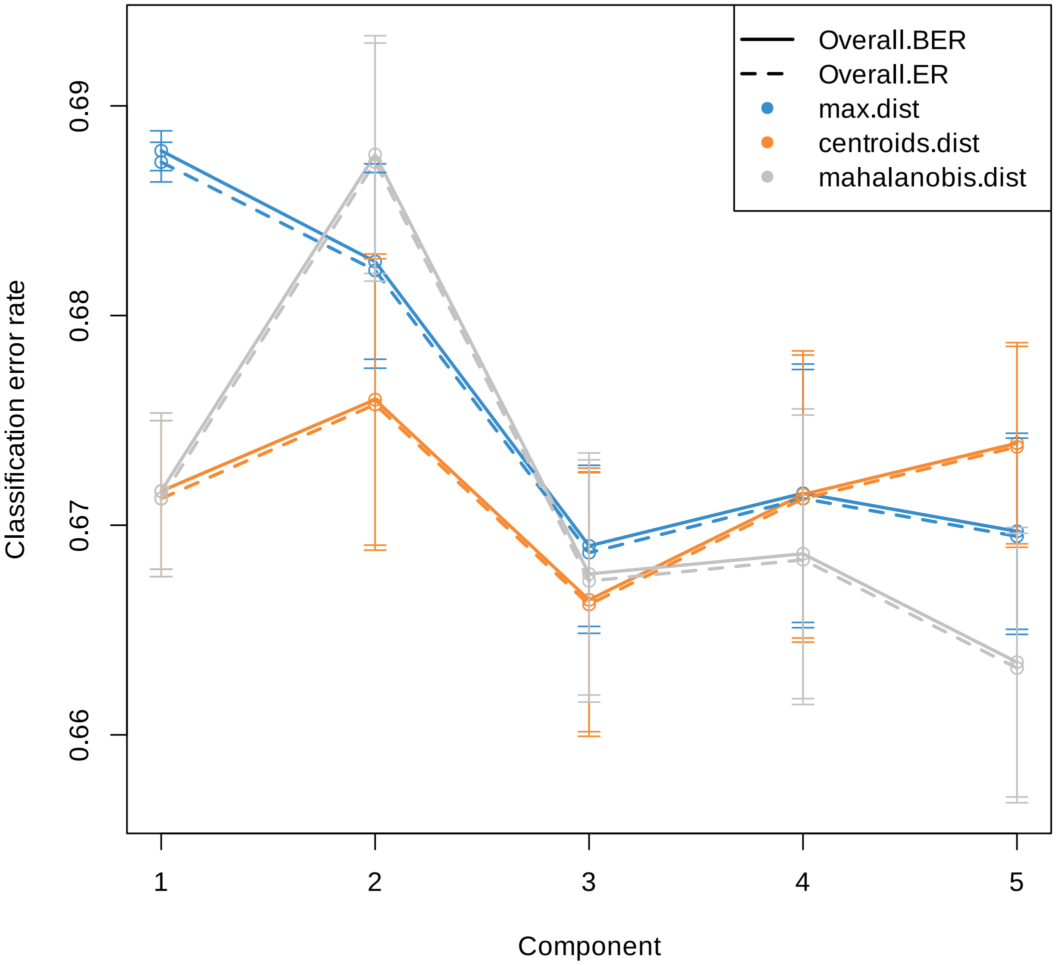
	metabolite	protein	clinical
metabolite	0.0	0.1	0.1
protein	0.1	0.0	0.1
clinical	0.1	0.1	0.0

[50]:
```
# For reference this is a highly correlated data set
# Requiring lower correlation in the design leads to higher prediction
res1.pls <- pls(mets_mat, prots_mat, ncomp = 1)
cat(noquote(paste("cor(PLS.Metabolomics, PLS.Proteomics) =",round(cor(res1.
 ↪pls$variates$X, res1.pls$variates$Y)[1,1], 3),"\n")))

res2.pls <- pls(mets_mat, clin_mat, ncomp = 1)
cat(noquote(paste("cor(PLS.Metabolomics, PLS.Clinical)   =",round(cor(res2.
 ↪pls$variates$X, res2.pls$variates$Y)[1,1], 3),"\n")))

res3.pls<- pls(prots_mat, clin_mat, ncomp = 1)
cat(noquote(paste("cor(PLS.Proteomics,   PLS.Clinical)   =",round(cor(res3.
 ↪pls$variates$X, res3.pls$variates$Y)[1,1],3),"\n")))
```

```
cor(PLS.Metabolomics, PLS.Proteomics) = 0.682
cor(PLS.Metabolomics, PLS.Clinical)   = 0.879
cor(PLS.Proteomics,   PLS.Clinical)   = 0.649
```

[51]:
```
# This takes a 20 min to run!
diablo.selfFI <- block.plsda(X, Outcome, ncomp = 5, design = design)

perf.diablo.selfFI = mixOmics::perf(diablo.selfFI, validation = 'Mfold',
                                     progressBar = TRUE,
                                           folds = 10, nrepeat = 10)

# Plot of the error rates based on weighted vote
plot(perf.diablo.selfFI)
```

```
Design matrix has changed to include Y; each block will be
            linked to Y.


Performing repeated cross-validation with nrepeat = 10…
  |=====================================================================
=====================================================================
============================|  100%
```



[52]:
```
perf.diablo.selfFI$choice.ncomp$WeightedVote
```

A matrix: 2 × 3 of type dbl

	max.dist	centroids.dist	mahalanobis.dist
Overall.ER	3	1	5
Overall.BER	3	1	5

[64]:
```
# ncomp <- perf.diablo.selfFI$choice.ncomp$WeightedVote["Overall.BER",␣
 ↪"centroids.dist"]
ncomp <- 4
```

4.1 Tuning the sparsity of the components

[65]:
```
# Variable tuning - the number of features to include in each component
# Set the search grid with value of 5 and reduce later

startTime <- Sys.time()
test.keepX <- list(metabolite = c(seq(3, 12, 3)),
                   protein = c(seq(3, 12, 3)),
                   clinical = c(seq(3, 9, 3)))

tune.diablo.selfFI <- tune.block.splsda(X, Outcome, ncomp = 2,
                              test.keepX = test.keepX, design = design,
                              validation = 'Mfold', folds = 10, nrepeat = 1,␣
 ↪### Should update nrepeats with a final model
                              BPPARAM = BiocParallel::SnowParam(workers = 16),
                              dist = "centroids.dist")

list.keepX <- tune.diablo.selfFI$choice.keepX

endTime <- Sys.time()
print(endTime - startTime)
```

```
Design matrix has changed to include Y; each block will be
            linked to Y.


You have provided a sequence of keepX of length: 4 for block metabolite and 4
for block protein and 3 for block clinical.
This results in 48 models being fitted for each component and each nrepeat, this
may take some time to run, be patient!


Time difference of 10.01568 mins
```

[66]:
```
print(list.keepX)
```

```
$metabolite
[1] 9 9

$protein
[1] 9 9

$clinical
[1] 6 6
```

4.1.1 Final model

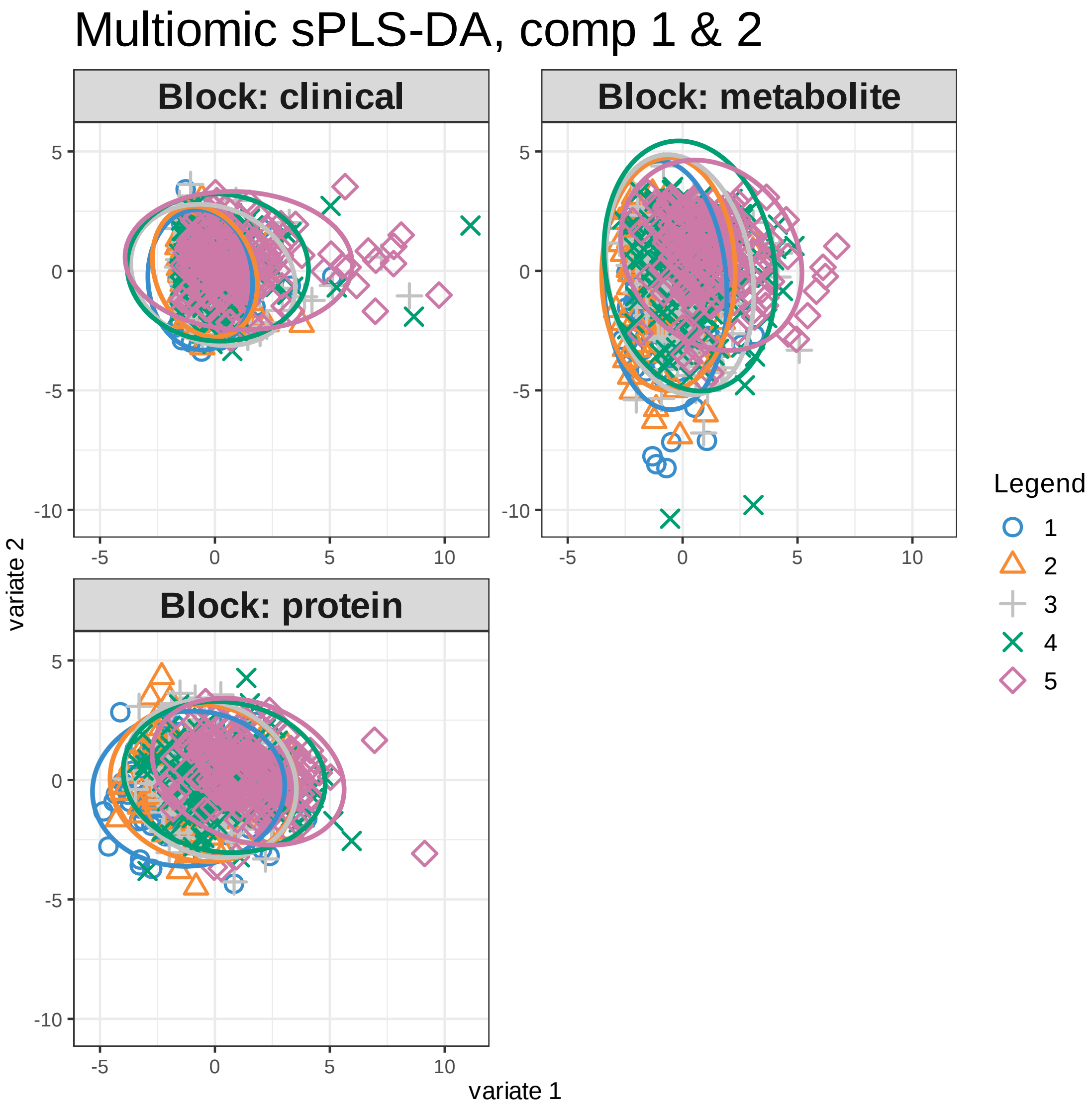
[67]:

```r
diablo.selfFI.final <- block.splsda(X, Outcome, ncomp = ncomp,
                            keepX = list.keepX, design = design)
```

```
Design matrix has changed to include Y; each block will be
            linked to Y.
```

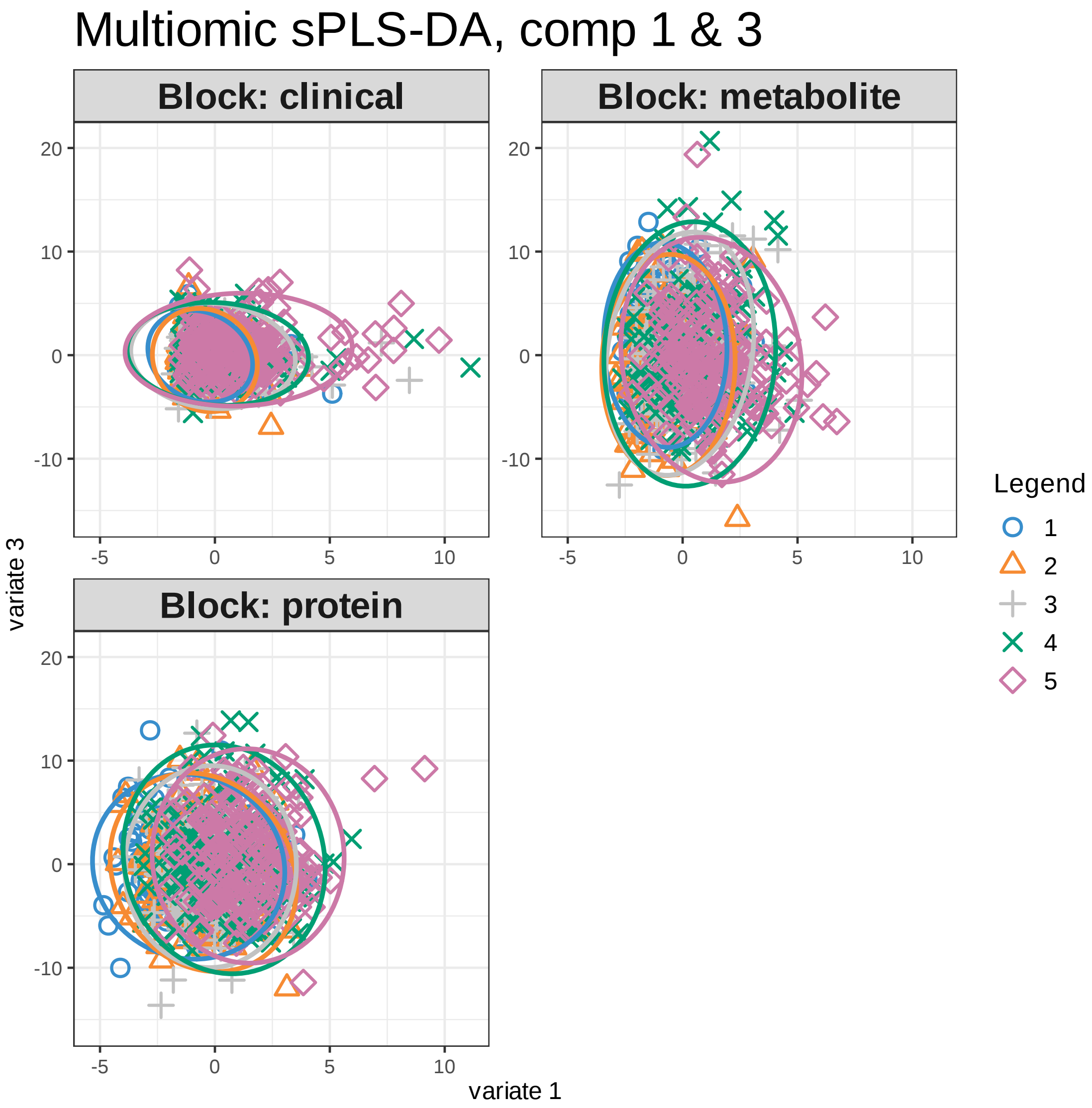
[68]:

```r
plotIndiv(diablo.selfFI.final, comp = c(1,2), # plot samples from final model
          group = Outcome, ind.names = FALSE,
          ellipse = TRUE, legend = TRUE,
          title = 'Multiomic sPLS-DA, comp 1 & 2')
```

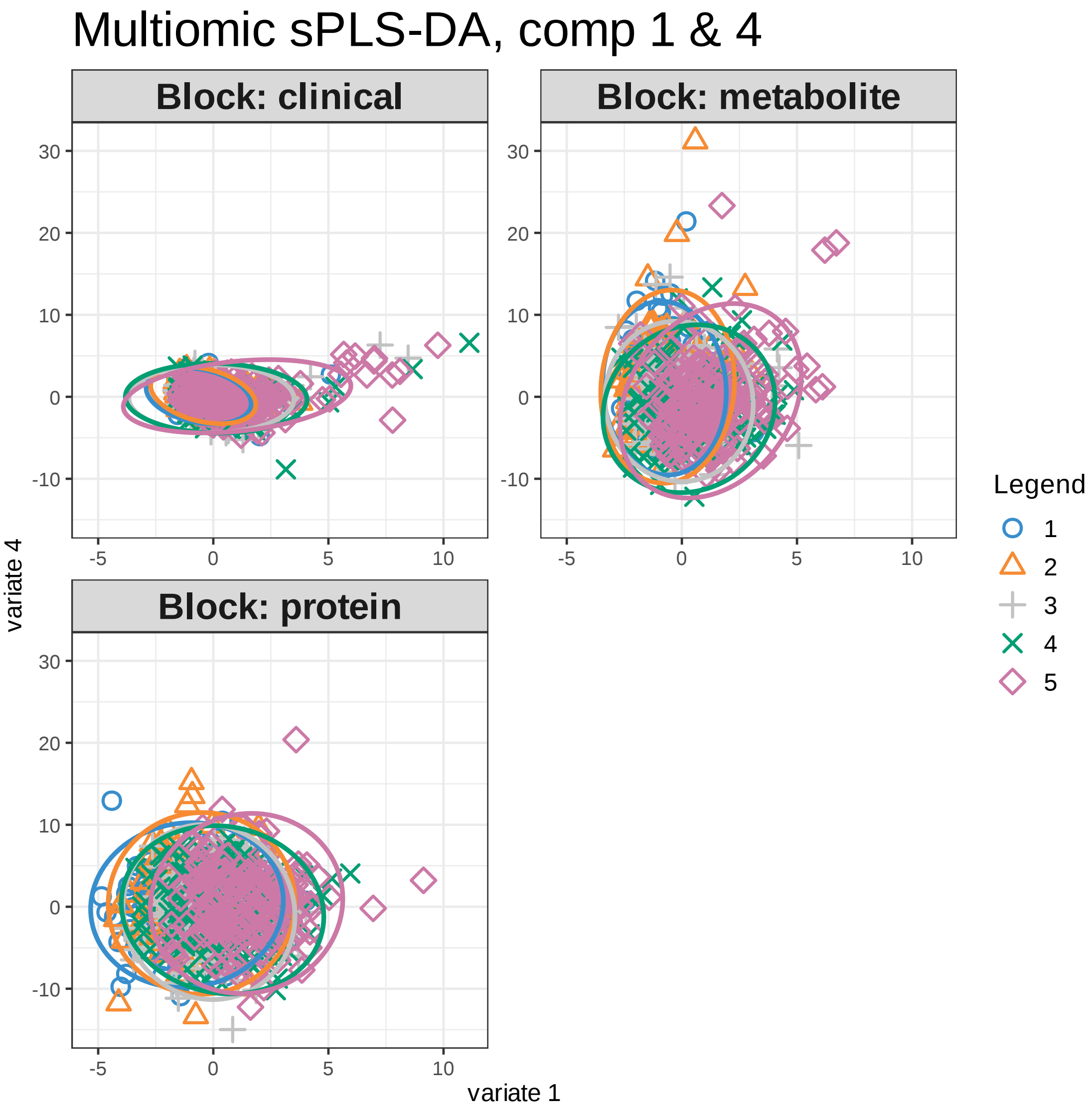


[69]:

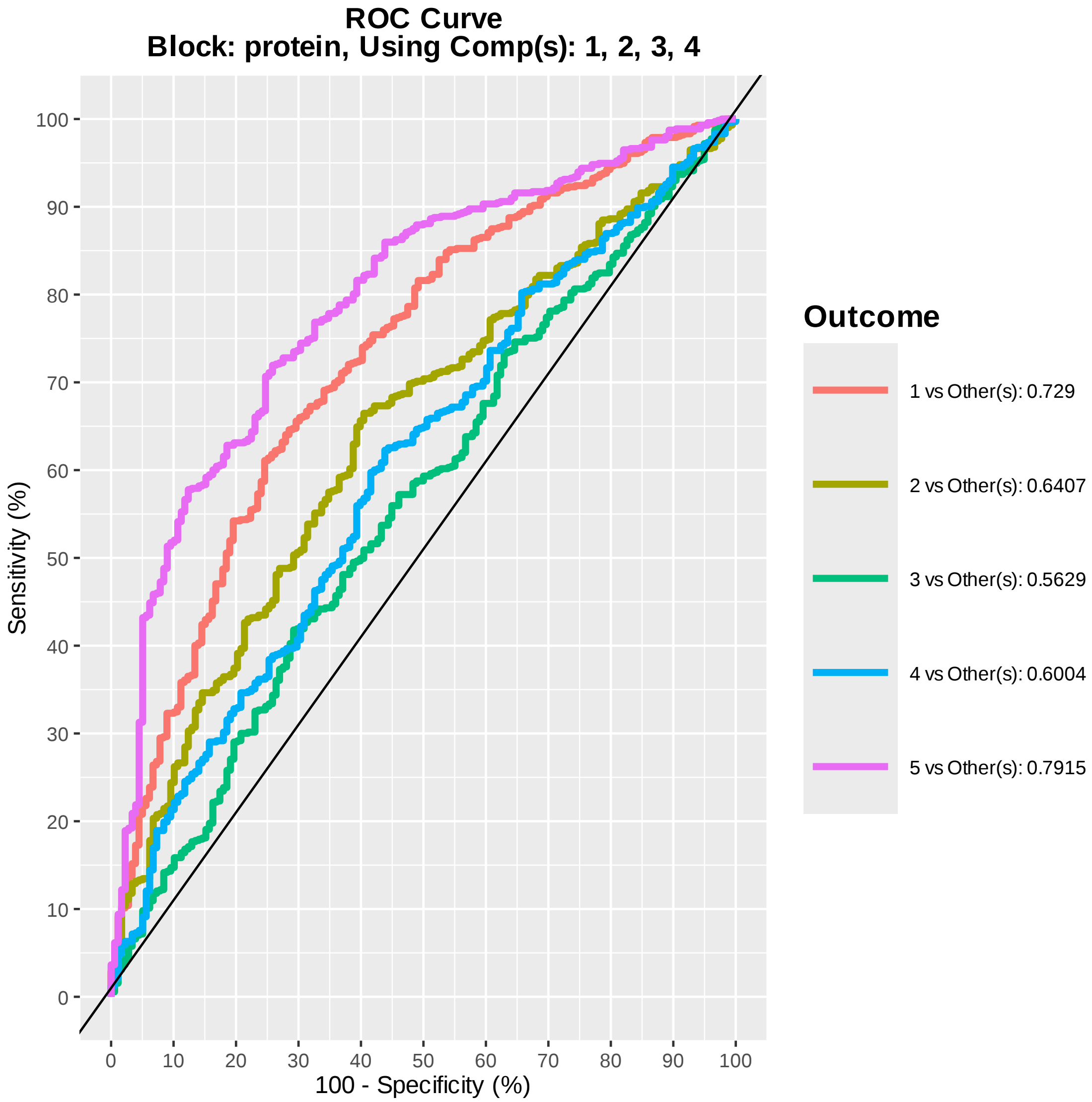
```
plotIndiv(diablo.selfFI.final, comp = c(1,3), # plot samples from final model
          group = Outcome, ind.names = FALSE,
          ellipse = TRUE, legend = TRUE,
          title = 'Multiomic sPLS-DA, comp 1 & 3')
```



```
[70]: plotIndiv(diablo.selfFI.final, comp = c(1,4), # plot samples from final model
                group = Outcome, ind.names = FALSE,
                ellipse = TRUE, legend = TRUE,
                title = 'Multiomic sPLS-DA, comp 1 & 4')
```

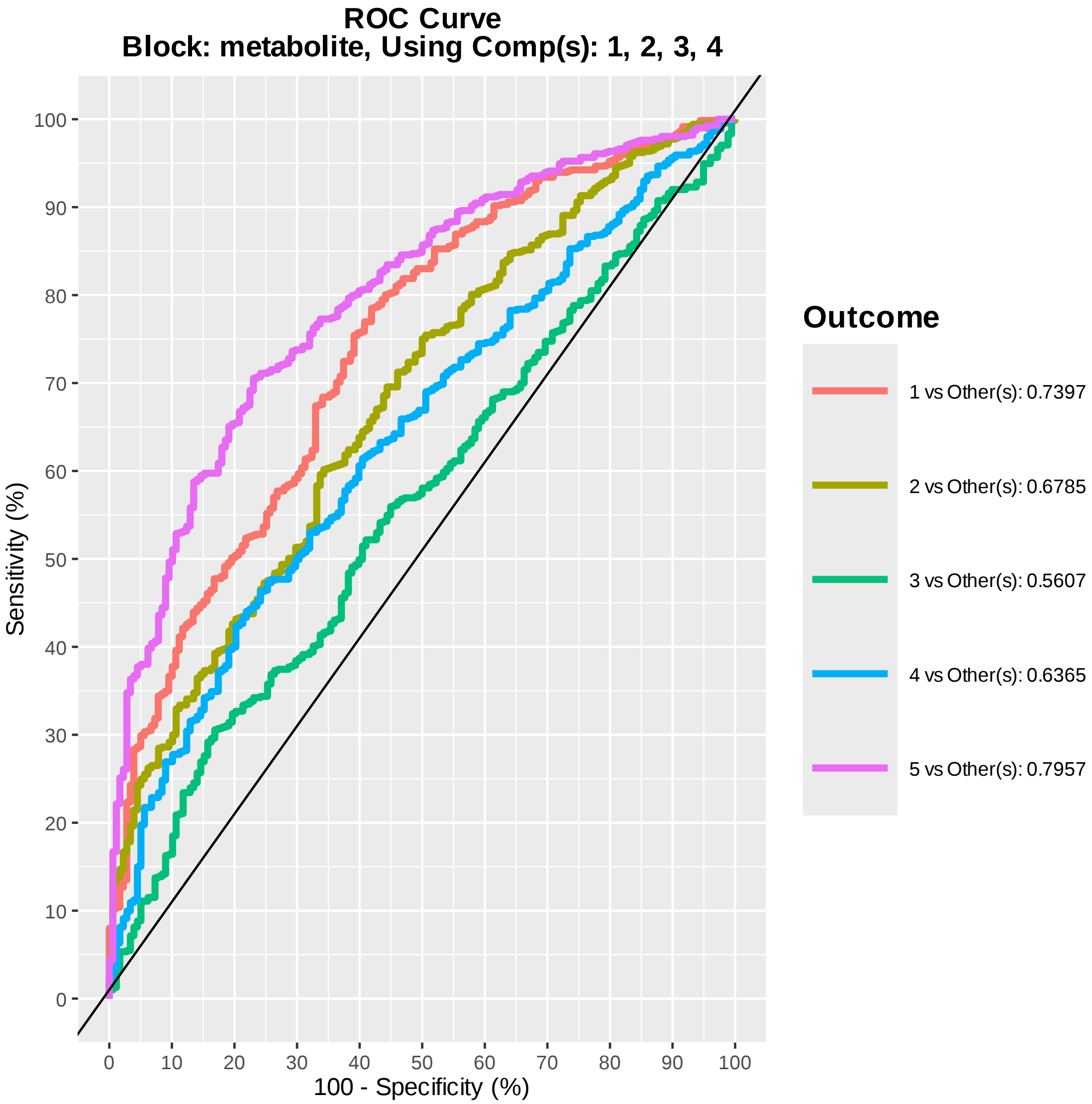


[71]:
```
auc.diablo.prots <- auroc(diablo.selfFI.final, roc.block = "protein", roc.comp␣
↪= 4, print = FALSE)
```

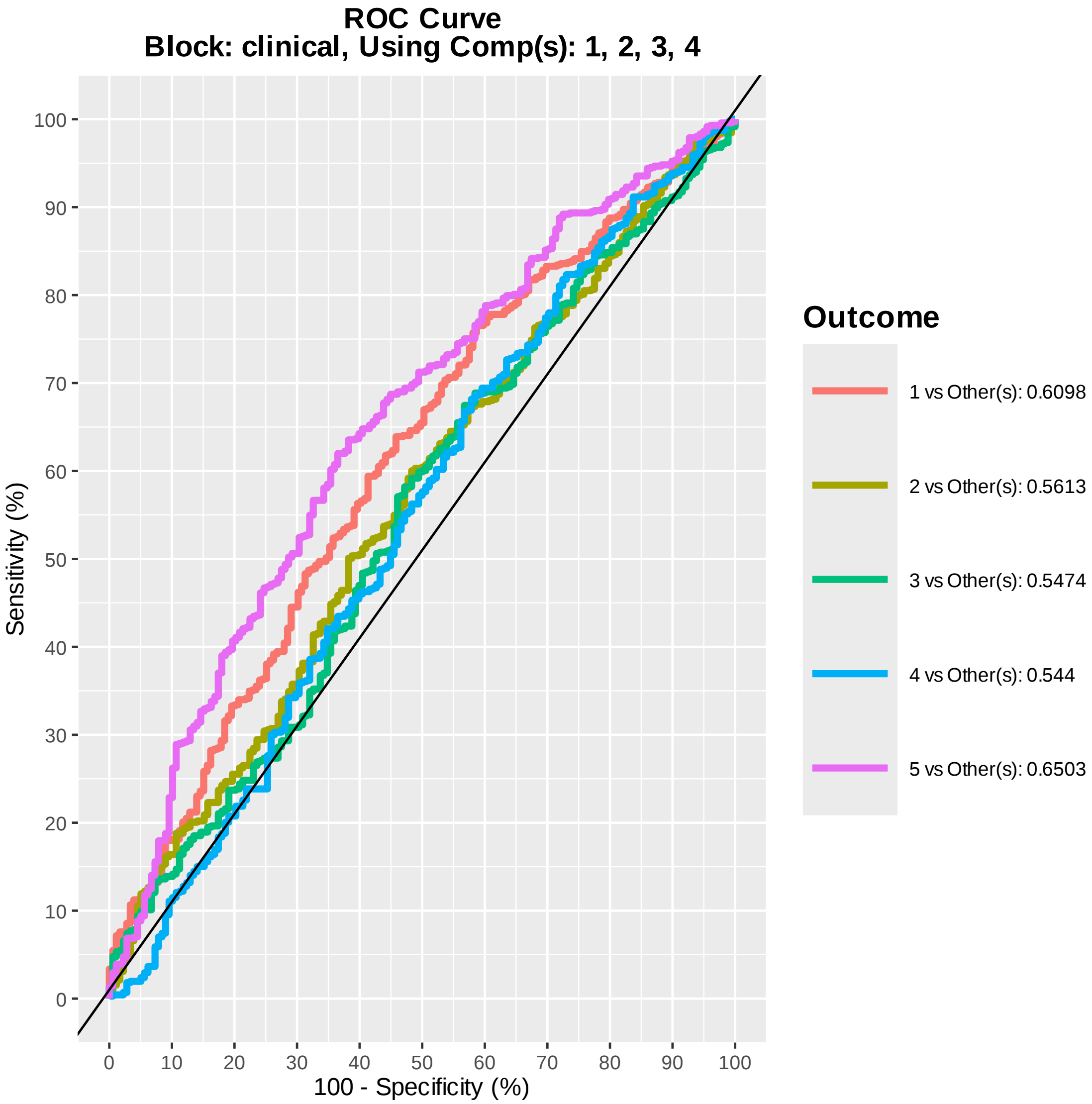


[72]:

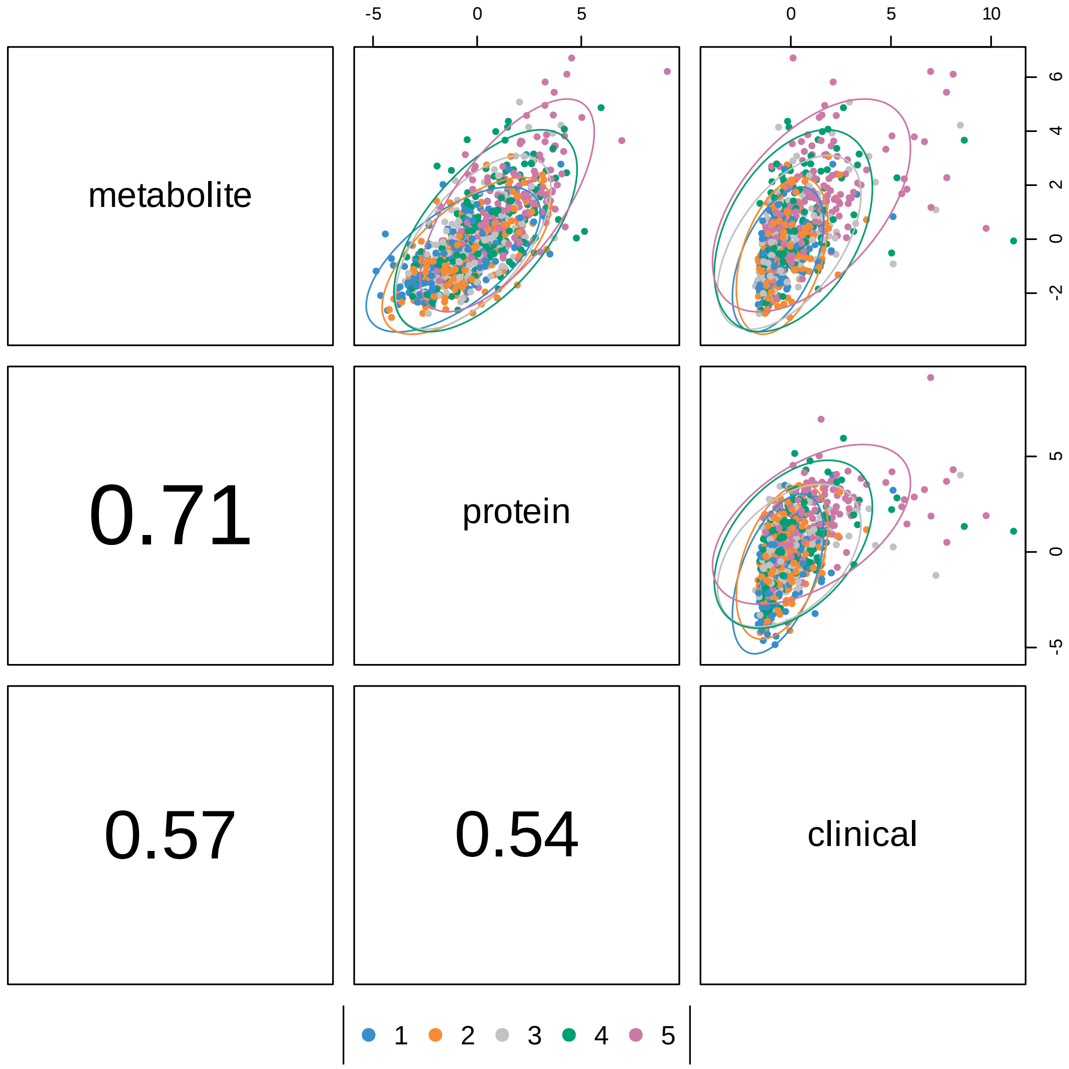
```
auc.diablo.met <- auroc(diablo.selfFI.final, roc.block = "metabolite", roc.comp␣
 ↪= 4, print = FALSE)
```



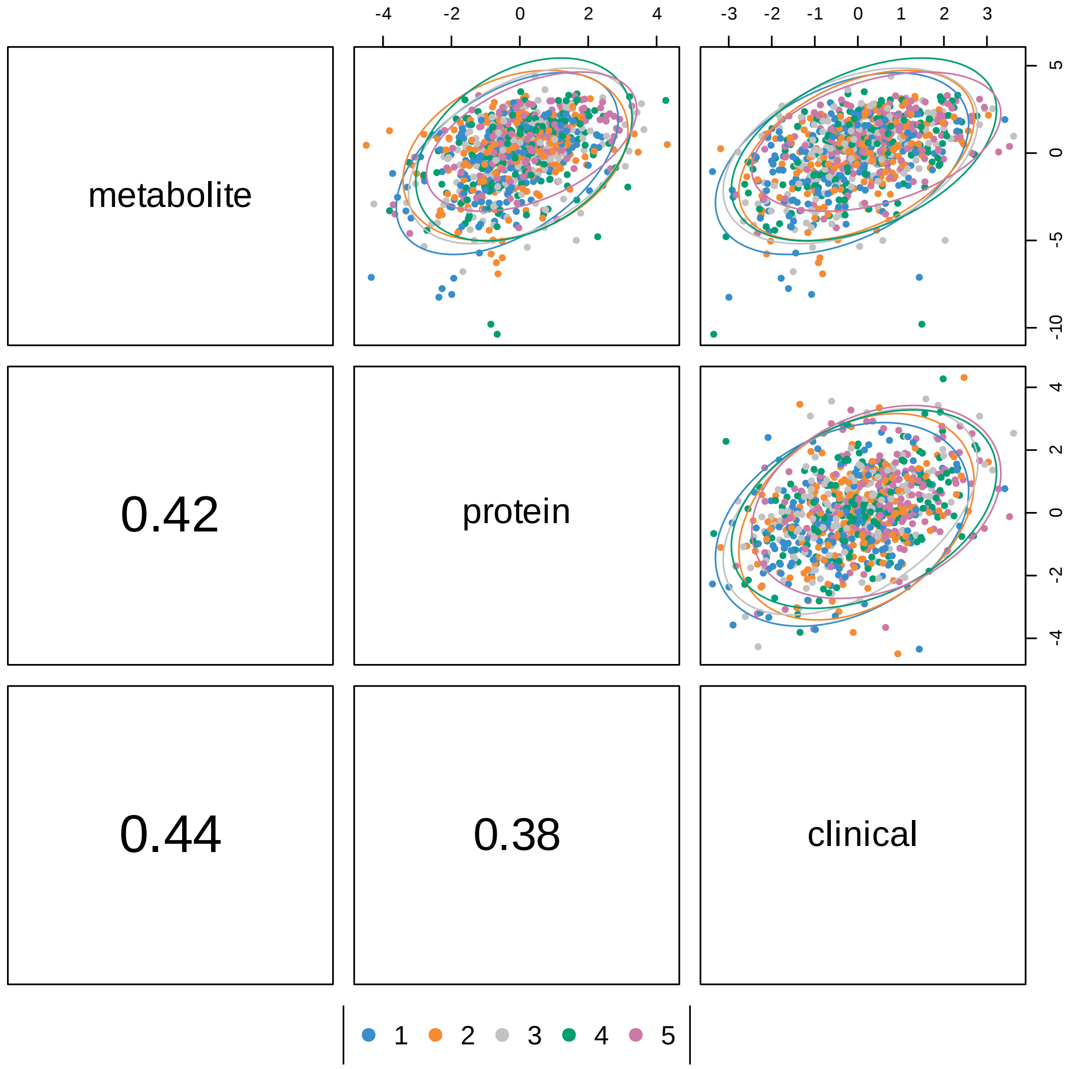
```
[73]: auc.diablo.clin <- auroc(diablo.selfFI.final, roc.block = "clinical", roc.comp␣
      ↪= 4, print = FALSE)
```



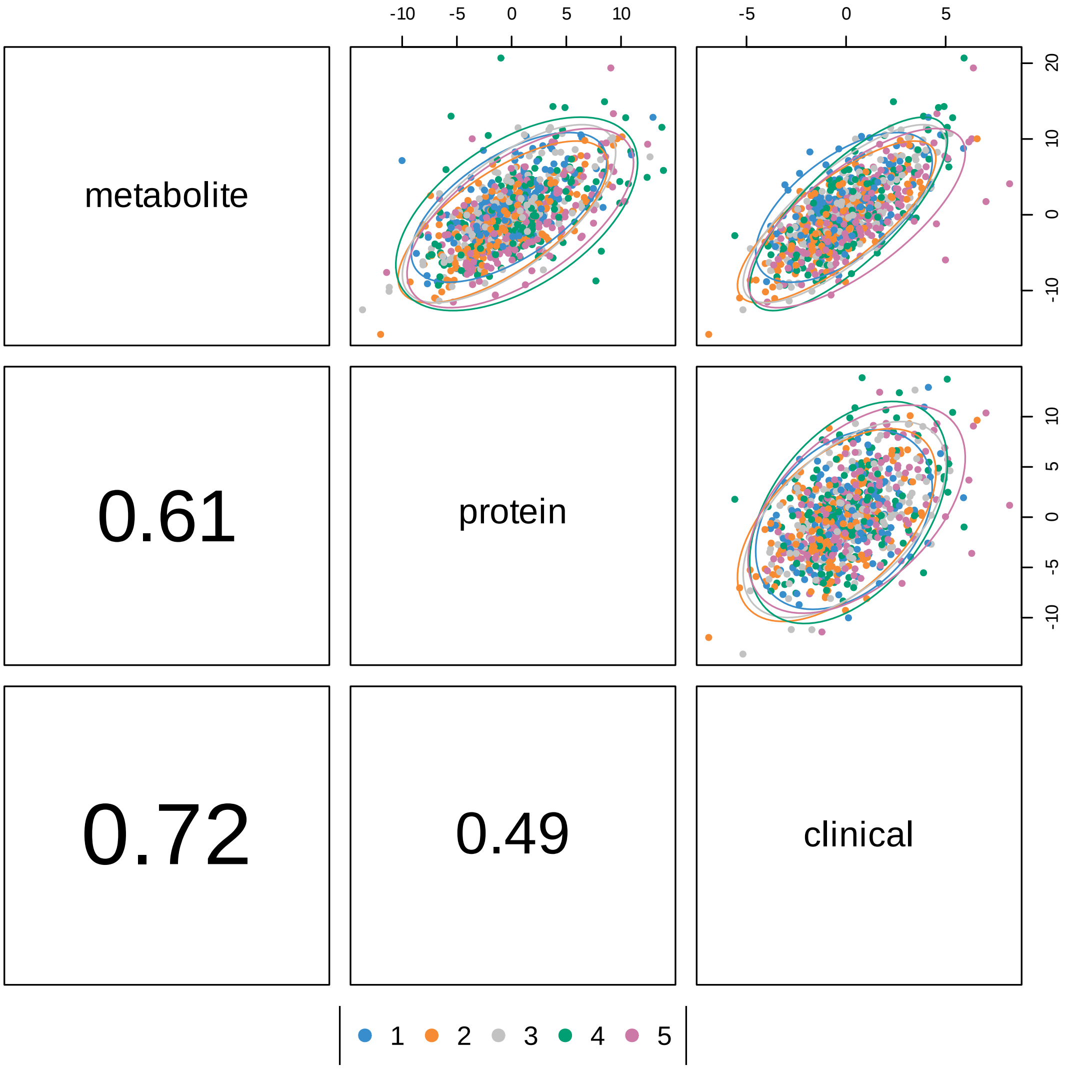
[74]:
```
plotDiablo(diablo.selfFI.final, ncomp = 1)
```



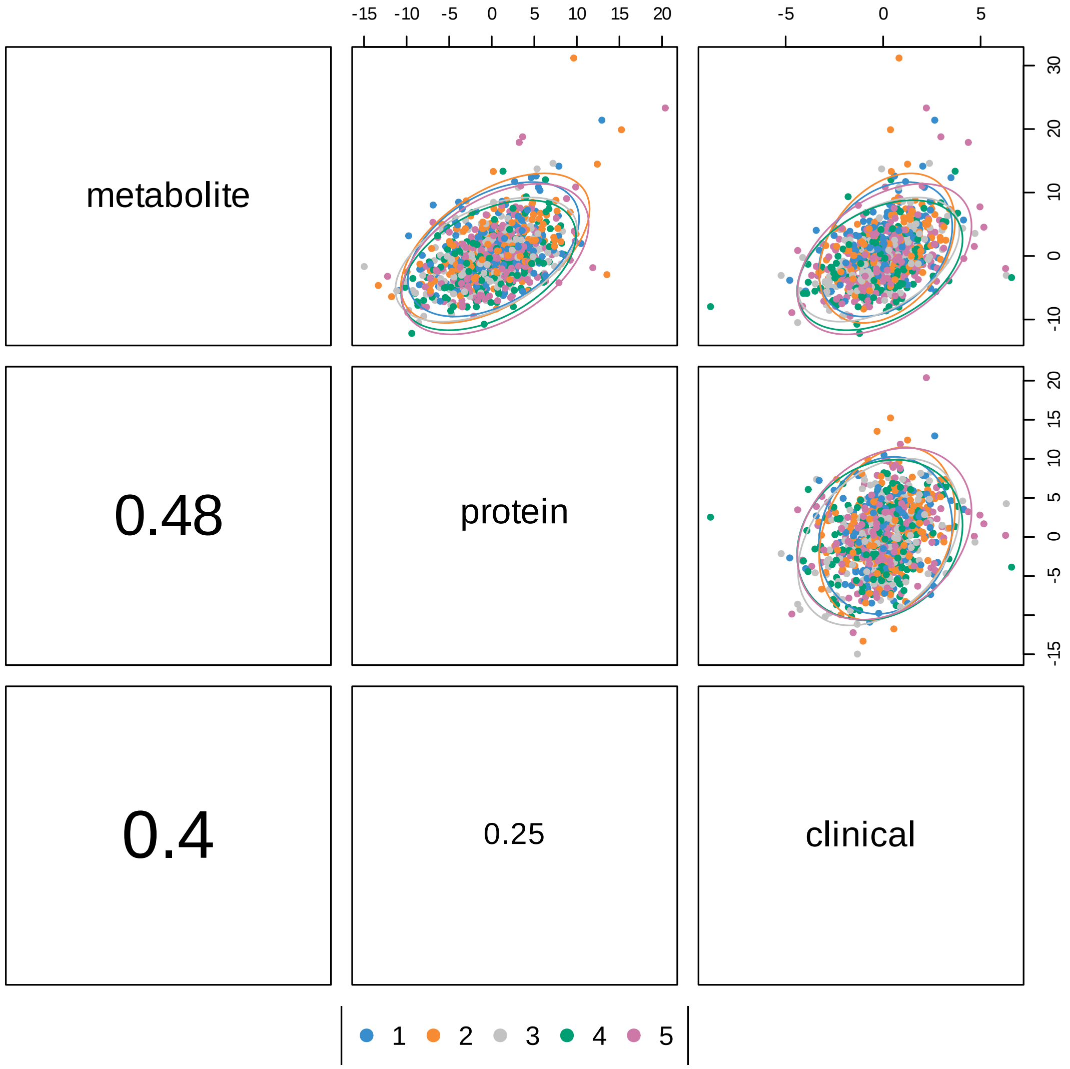
```
[75]: plotDiablo(diablo.selfFI.final, ncomp = 2)
```



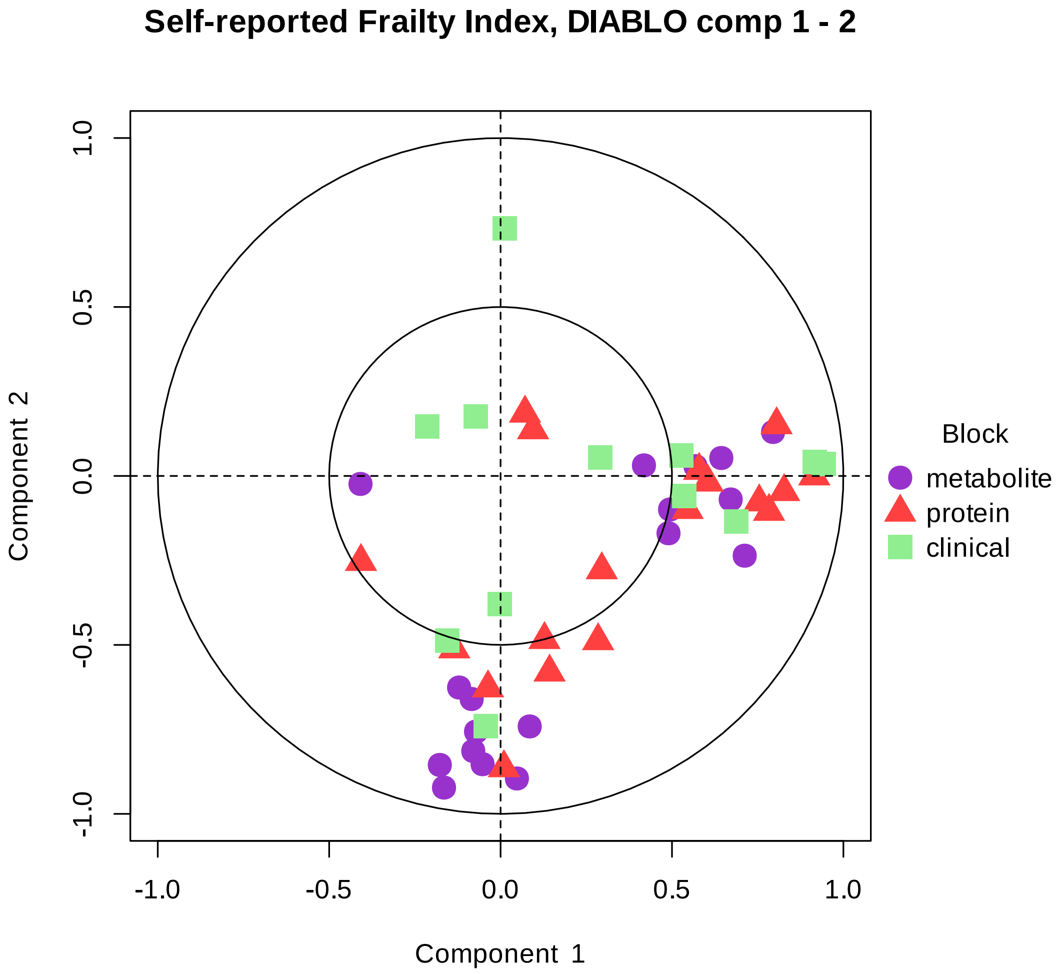
[76]:
```
plotDiablo(diablo.selfFI.final, ncomp = 3)
```



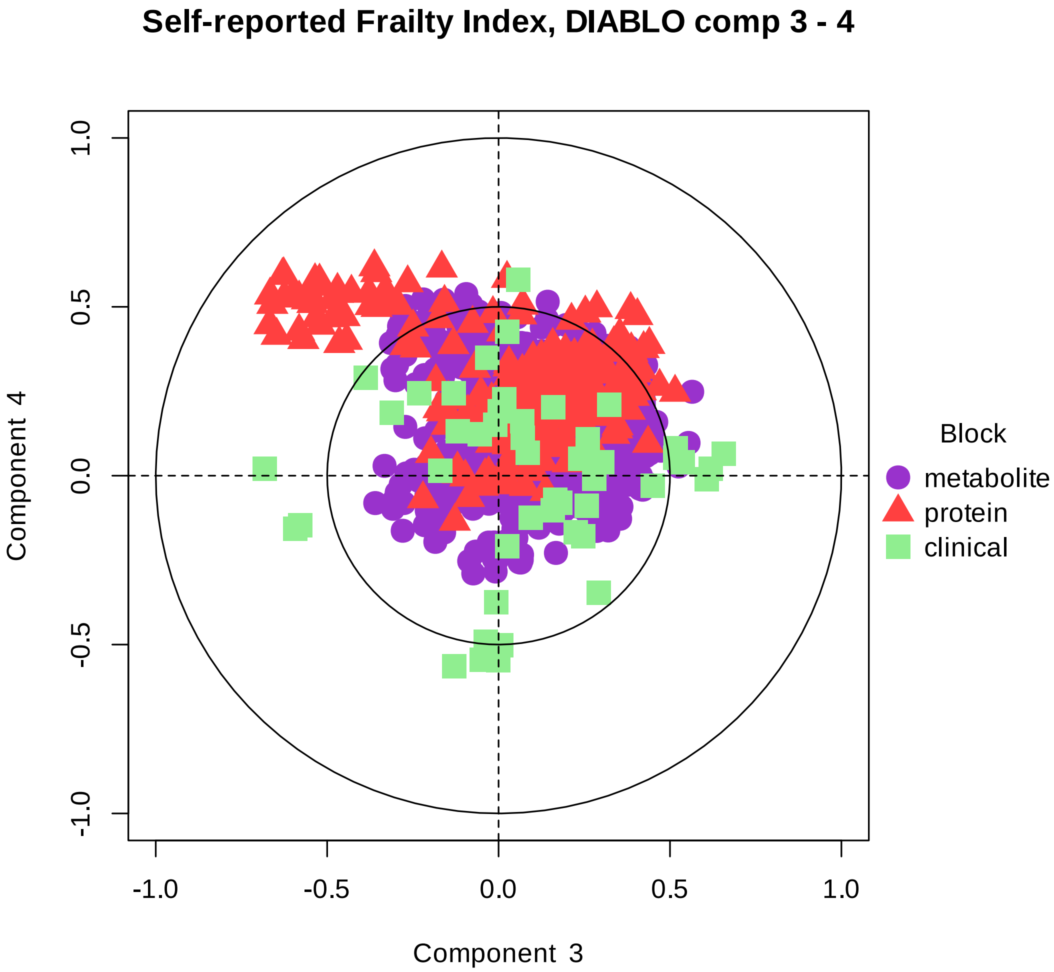
[77]:
```
plotDiablo(diablo.selfFI.final, ncomp = 4)
```



[78]:
```
plotVar(diablo.selfFI.final, var.names = FALSE, style = 'graphics', legend =␣
 ↪TRUE,
        pch = c(16, 17, 15), cex = c(2,2,2),  comp=c(1,2),
        col = c('darkorchid', 'brown1', 'lightgreen'),
        title = 'Self-reported Frailty Index, DIABLO comp 1 - 2')
```



```
[79]: plotVar(diablo.selfFI.final, var.names = FALSE, style = 'graphics', legend = TRUE,
              pch = c(16, 17, 15), cex = c(2,2,2), comp=c(3,4),
              col = c('darkorchid', 'brown1', 'lightgreen'),
              title = 'Self-reported Frailty Index, DIABLO comp 3 - 4')
```



[80]:

```
norm <- function(v) { sqrt(sum(v*v)) }
threshold <- 0.25
for (omic in names(diablo.selfFI.final$loadings)[1:3]) {
    r <- unlist(apply(diablo.selfFI.final$loadings[[omic]], 1, norm))
    print(diablo.selfFI.final$loadings[[omic]][r > threshold,],digits=3)
    cat(noquote("\n"))
}
```

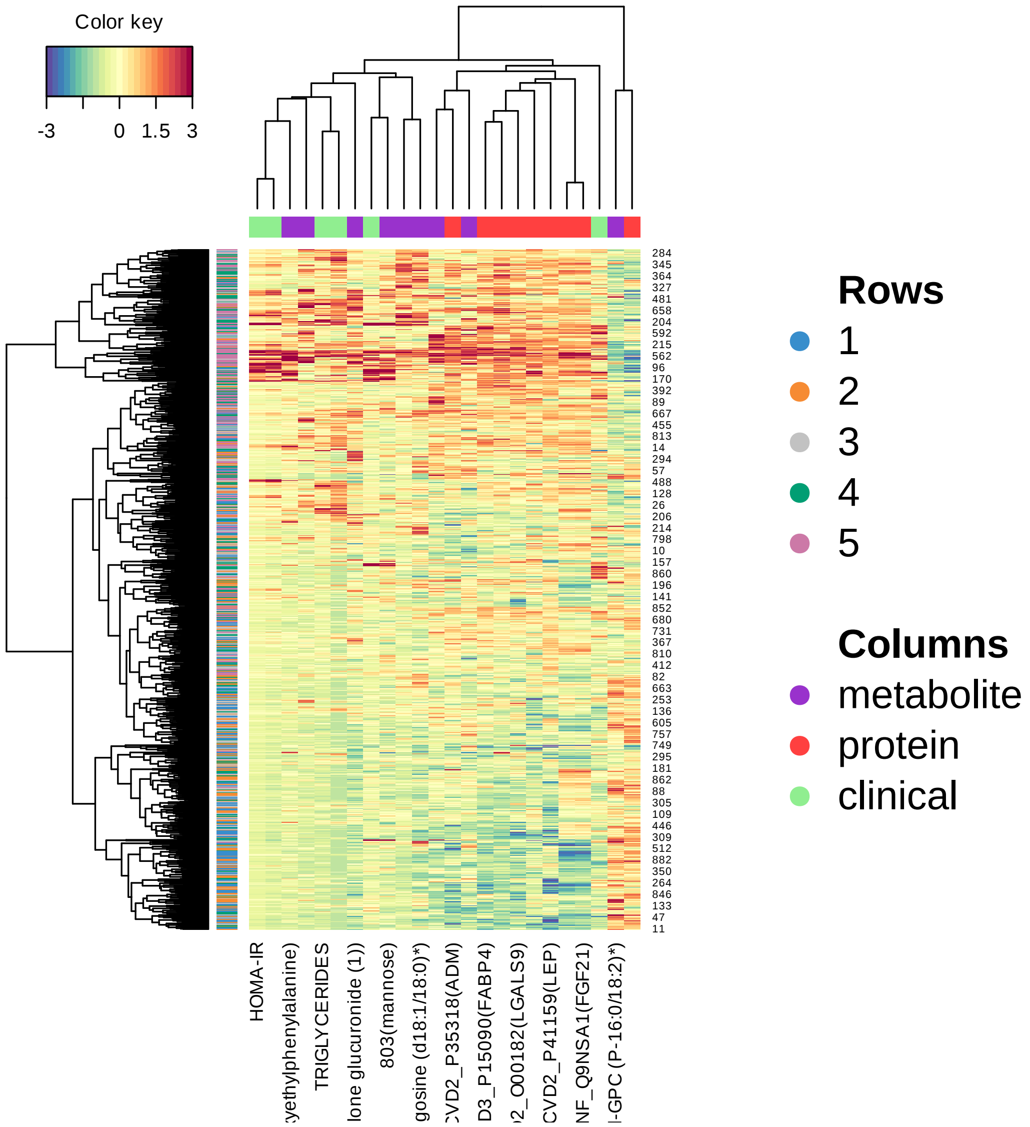
```
                                                      comp1  comp2    comp3
comp4
922(N-stearoyl-sphinganine (d18:0/18:0)*)             0.455  0.000 -0.03706
-0.02158
100000792(dehydroepiandrosterone sulfate (DHEA-S))    0.000 -0.703 -0.01144
```

```
-0.00106
100001994(androstenediol (3beta,17beta) disulfate (2)) 0.000 -0.484 -0.00998
-0.01791
100002067(pregnenediol sulfate (C21H34O5S)*)           0.000 -0.361  0.02108
0.00185
100015971(cortolone glucuronide (1))                   0.382  0.000 -0.00718
0.00812
100020205(hydroxyasparagine**)                         0.730  0.000  0.02651
0.01643

                    comp1  comp2     comp3    comp4
CVD2_O00182(LGALS9) 0.246  0.000  0.059155  0.04585
CVD2_O00253(AGRP)   0.000 -0.849 -0.009975 -0.01444
CVD2_P18510(IL1RN)  0.317  0.000 -0.029262  0.00489
CVD2_P35318(ADM)    0.380  0.000  0.072882 -0.00190
CVD2_P41159(LEP)    0.459  0.000 -0.095473 -0.03382
CVD3_P15090(FABP4)  0.677  0.000 -0.000457  0.01247
CVD3_Q9NQ76(MEPE)   0.000 -0.375  0.008933 -0.04596

                       comp1  comp2     comp3    comp4
ALBUMIN                0.000 -0.377  0.000332  0.11421
BUN/CREAT RATIO        0.000  0.665  0.130878  0.00484
CREATININE ENZ, SER    0.000 -0.557  0.244141 -0.05311
GFR, MDRD              0.000  0.000 -0.344761 -0.12230
GFR, MDRD, AFRICAN AM  0.000  0.000 -0.339979 -0.15279
HDL CHOL DIRECT        0.000  0.000  0.028041  0.38309
HDL PARTICLE NUMBER    0.000  0.000 -0.020439  0.27153
HOMA-IR                0.531  0.000  0.004619  0.11714
INSULIN                0.735  0.000 -0.013823  0.06297
LPIR_SCORE             0.359  0.000 -0.030393 -0.26410
POTASSIUM              0.000 -0.304 -0.082552  0.02027
TRIGLYCERIDES          0.152  0.000  0.034645 -0.22445
Triglyceride HDL Ratio 0.000  0.000  0.022311 -0.26016
UREA NITROGEN          0.000  0.000  0.304412 -0.02366
```

[ ]:

```
cimDiablo(diablo.selfFI.final, color.blocks = c('darkorchid', 'brown1', 'lightgreen'),
          comp = 1, margin=c(8,20), legend.position = "right")
```

```
trimming values to [-3, 3] range for cim visualisation. See 'trim' arg in
?cimDiablo
```



[ ]:

```
perf.diablo.selfFI.final <- perf(diablo.selfFI.final,  validation = 'Mfold',␣
↪folds = 10,
                                 nrepeat = 10, dist = 'centroids.dist')
```

[ ]:

```
plot(perf.diablo.selfFI)
```

[ ]:

```

```