DIABLO Analysis of Fibrotic ILA - 2 component

Dependencies

```
# Load required libraries
library(tidyverse)    # Includes ggplot2, dplyr, etc.
library(openxlsx)
library(DESeq2)       # For RNA-Seq analysis
library(pheatmap)     # For heatmap visualizations
library(RColorBrewer) # For color palettes
library(reshape2)     # For data reshaping
library(pbapply)      # For progress bar in apply functions
#library(limma)        # For linear modeling
library(data.table)   # For data manipulation
library(car)          # For advanced regression modeling
library(Biobase)      # For Bioconductor data structures
library(mixOmics)
library(BiocParallel)
library(parallel)
detectCores()  # Number of cores available on your machine

# Set global options
options(stringsAsFactors = FALSE)
BPPARAM <- SnowParam(workers = 14)
```

Importing Data

```
methylation <- read.csv("Final_Datasets/methylation_final_subset.csv",
                        row.names = 1,           # Use the first column (BCP_ID) as row names
```

```
                        stringsAsFactors = FALSE, # Keep character columns as characters
                        check.names = FALSE)      # Do not modify column names

proteomic <- read.csv("Final_Datasets/proteomic_final_subset.csv",
                      row.names = 1,
                      stringsAsFactors = FALSE,
                      check.names = FALSE)

rnaseq <- read.csv("Final_Datasets/rnaseq_final_subset.csv",
                   row.names = 1,
                   stringsAsFactors = FALSE,
                   check.names = FALSE)

phenotype <- read.csv("Final_Datasets/phenotype_final_subset.csv",
                      row.names = 1,
                      stringsAsFactors = FALSE,
                      check.names = FALSE)
```

```
methylation <- methylation[match(rownames(phenotype), rownames(methylation)), , drop = FALSE]

proteomic <- proteomic[match(rownames(phenotype), rownames(proteomic)), , drop = FALSE]

rnaseq <- rnaseq[match(rownames(phenotype), rownames(rnaseq)), , drop = FALSE]

all(rownames(methylation) == rownames(phenotype)) # Should return TRUE
```

```
[1] TRUE
```

```
all(rownames(proteomic) == rownames(phenotype))   # Should return TRUE
```

```
[1] TRUE
```

```
all(rownames(rnaseq) == rownames(phenotype))      # Should return TRUE
```

```
[1] TRUE
```

```
X <- list(
  methylation = methylation,
  proteomic = proteomic,
```

```
  rnaseq = rnaseq
)

Y <- as.factor(phenotype$FibroticILA)

design <- matrix(0.5, ncol = length(X), nrow = length(X),
                 dimnames = list(names(X), names(X)))
diag(design) <- 0
```

```
# diablo.tcga <- block.plsda(X, Y, ncomp = 10, design = design)
#
# perf.diablo.tcga = perf(diablo.tcga, validation = 'Mfold', folds = 10, nrepeat = 10)
#
# perf.diablo.tcga$error.rate
#
# plot(perf.diablo.tcga)
#
# perf.diablo.tcga$choice.ncomp$WeightedVote
#
# ncomp <- perf.diablo.tcga$choice.ncomp$WeightedVote["Overall.BER", "centroids.dist"]
```

```
# test.keepX <- list(
#   methylation = c(5:9, seq(10, 25, 5)),
#   proteomic = c(5:9, seq(10, 25, 5)),
#   rnaseq = c(5:9, seq(10, 25, 5)))
#
# tune.result <- tune.block.splsda(
#     X = X,
#     Y = Y,
#     ncomp = 4,
#     test.keepX = test.keepX,
#     design = design,
#     validation = 'Mfold',
#     folds = 10,
#     nrepeat = 10,
#     dist = "mahalanobis.dist",
#     progressBar = TRUE,
#     BPPARAM = BPPARAM
# )
#
#
# list.keepX <- tune.result$choice.keepX
```

```r
# # Save the list.keepX object to an .rds file
# saveRDS(list.keepX, file = "list_keepX.rds")
```

```r
list.keepX <- readRDS("list_keepX_n2.rds")

list.keepX
```

```
$methylation
[1] 25 25

$proteomic
[1] 15  9

$rnaseq
[1] 5 6
```

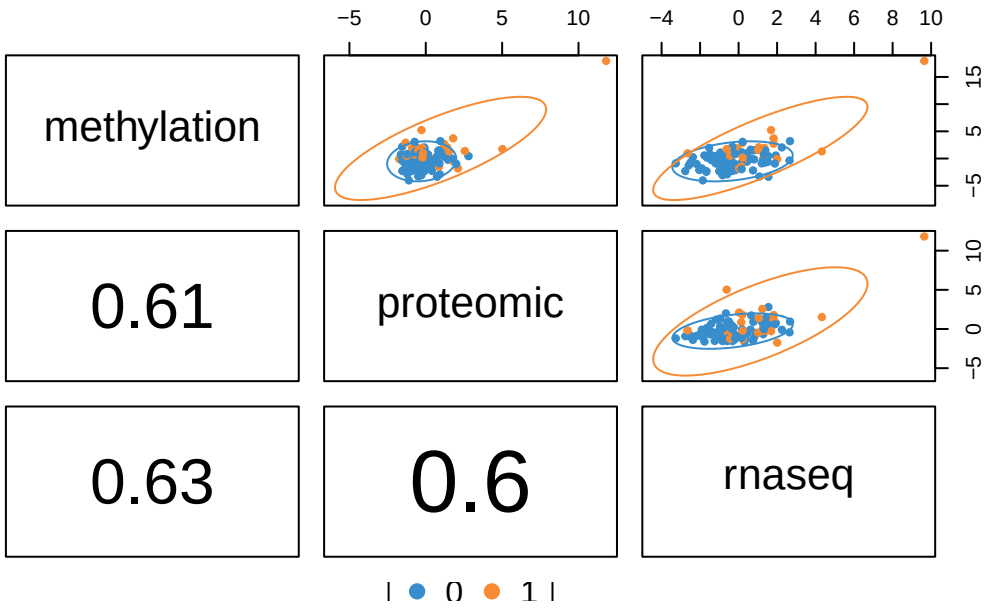
```r
diablo.tcga <- block.splsda(X = X, Y = Y, ncomp = 2, keepX = list.keepX, design = design)
```

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

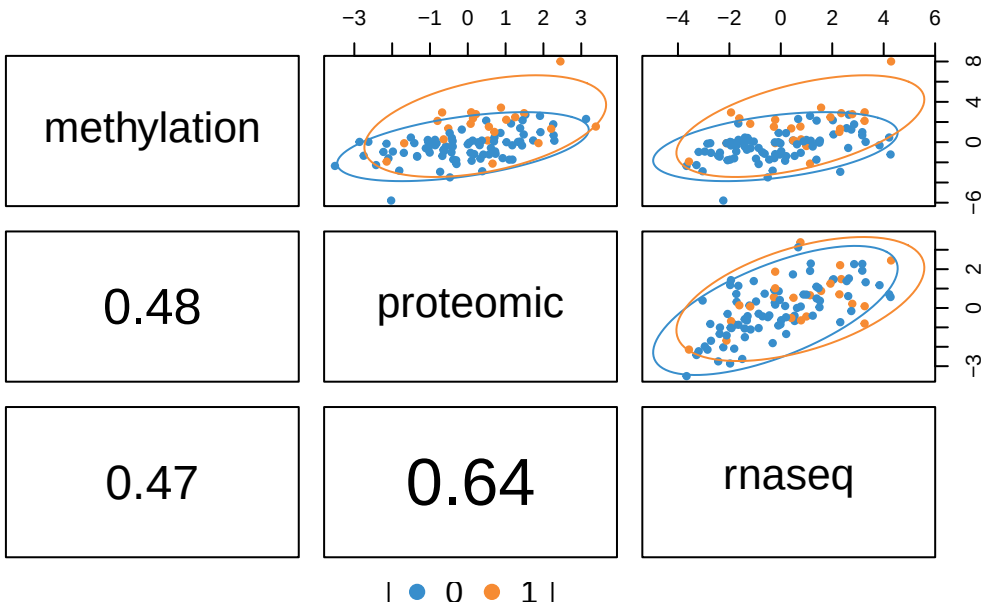
```r
diablo.tcga$design
```

```
            methylation proteomic rnaseq Y
methylation         0.0       0.5    0.5 1
proteomic           0.5       0.0    0.5 1
rnaseq              0.5       0.5    0.0 1
Y                   1.0       1.0    1.0 0
```

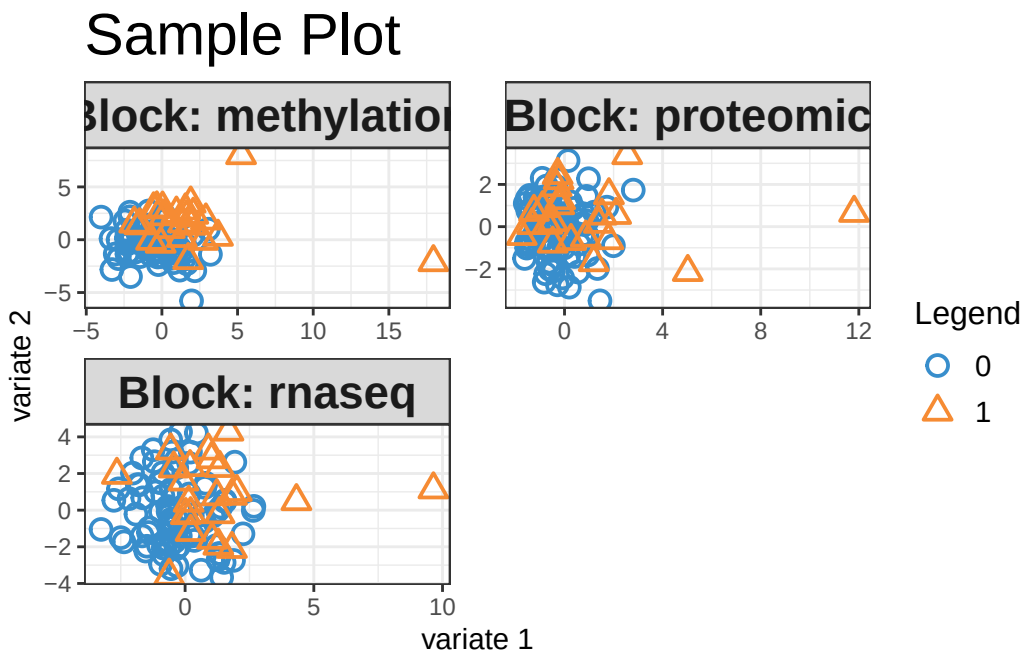
```r
plotDiablo(diablo.tcga, ncomp = 1)
```



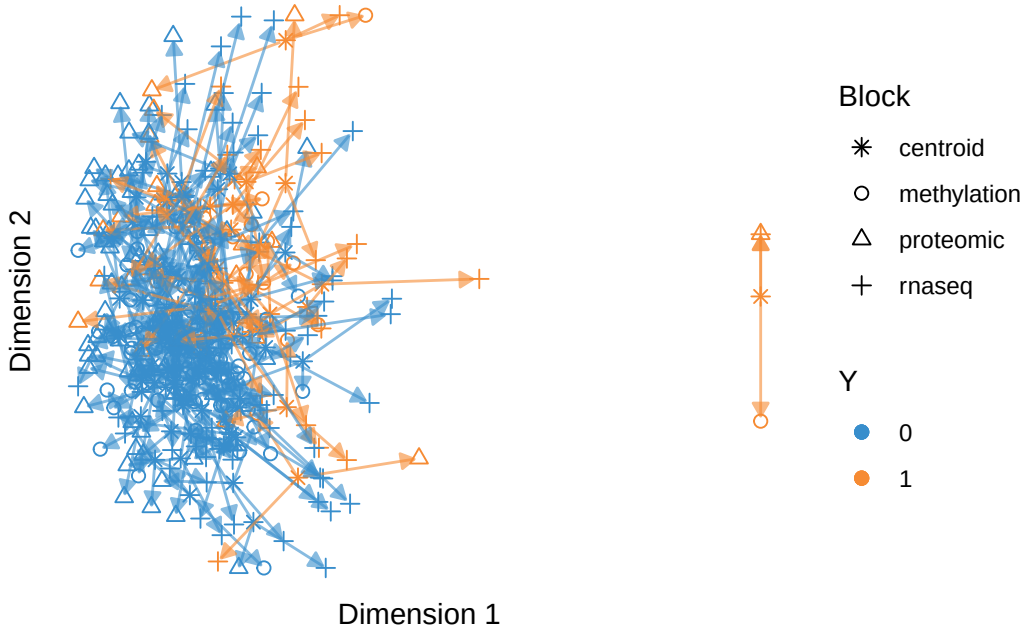

```
plotDiablo(diablo.tcga, ncomp = 2)
```



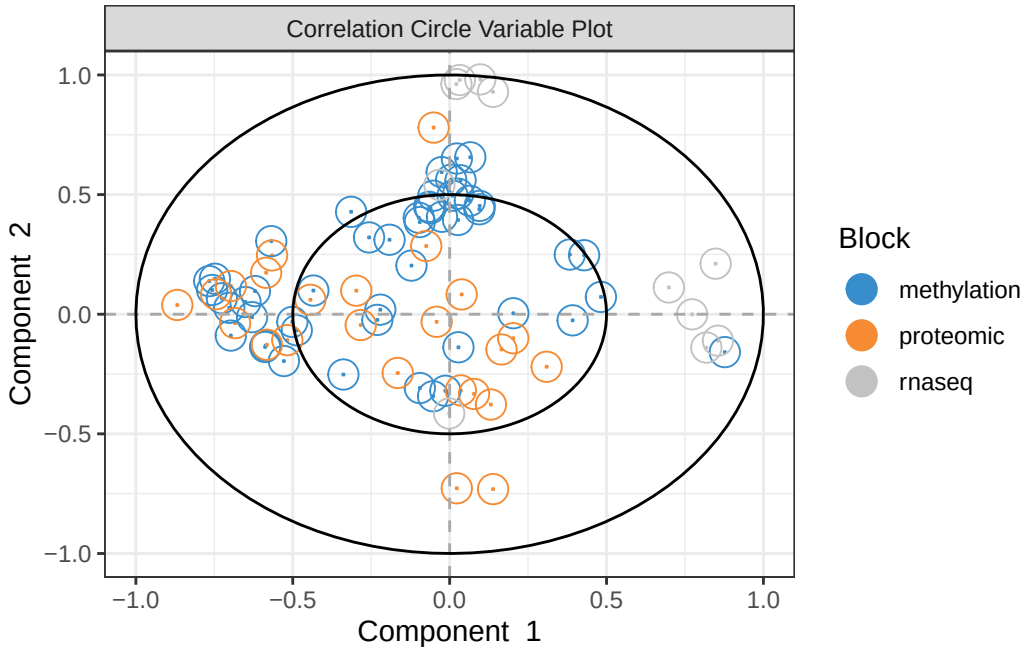
```
plotIndiv(diablo.tcga, ind.names = FALSE, legend = TRUE, title = 'Sample Plot')
```



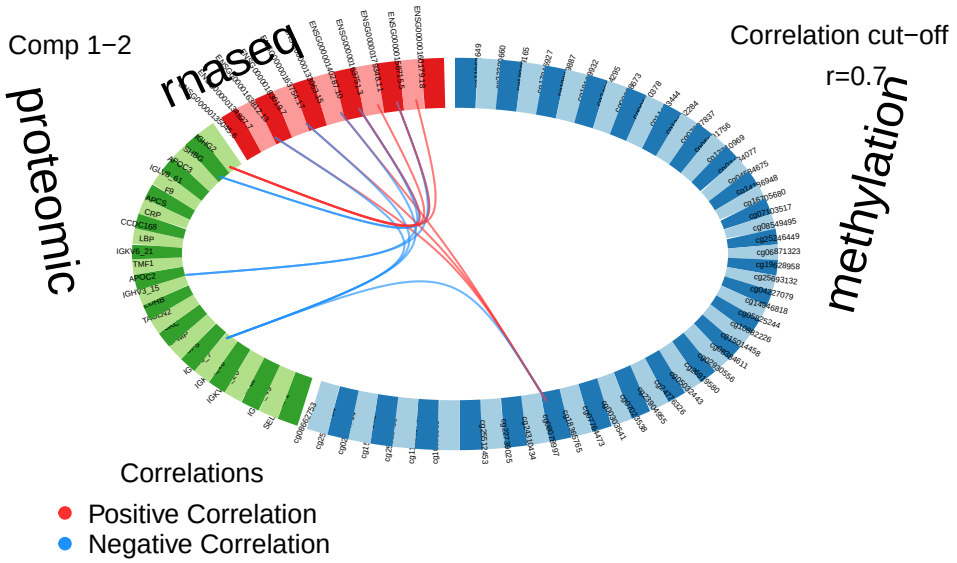

```
plotArrow(diablo.tcga, ind.names = FALSE, legend = TRUE,
          title = 'Arrow Plot')
```




```
plotVar(diablo.tcga, var.names = FALSE, legend = TRUE, title = 'Correlation Circle Variable
```

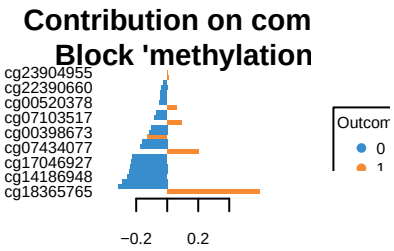


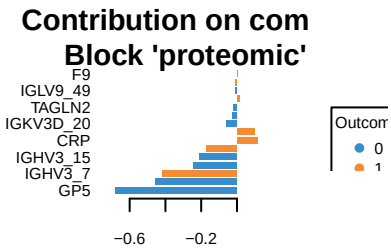
```
circosPlot(diablo.tcga, cutoff = 0.7, title = 'Circos Plot', size.labels = 1.5)
```

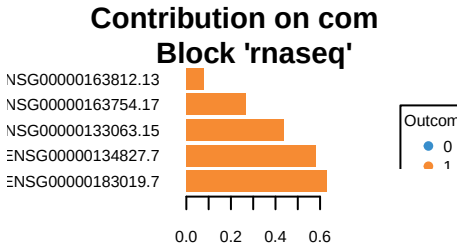



```
# pdf("network_plot.pdf", width = 850, height = 1100)
# network(diablo.tcga, blocks = c(1, 2, 3), cutoff = 0.76, color.node = c('darkorchid', 'brow
# dev.off()
```

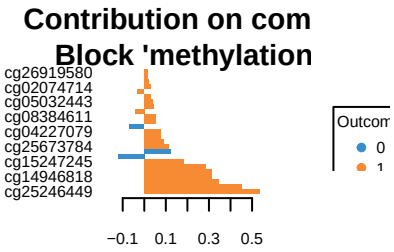
```
plotLoadings(diablo.tcga, comp = 1, contrib = 'max', method = 'median')
```

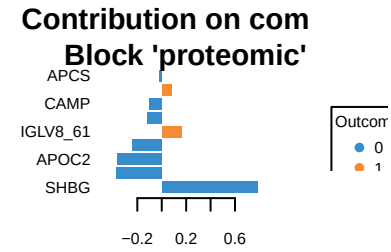


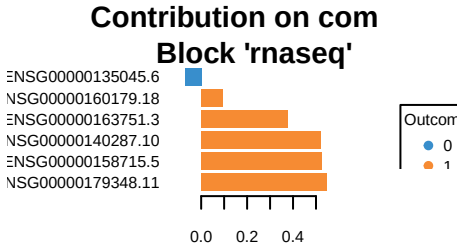





```
plotLoadings(diablo.tcga, comp = 2, contrib = 'max', method = 'median')
```







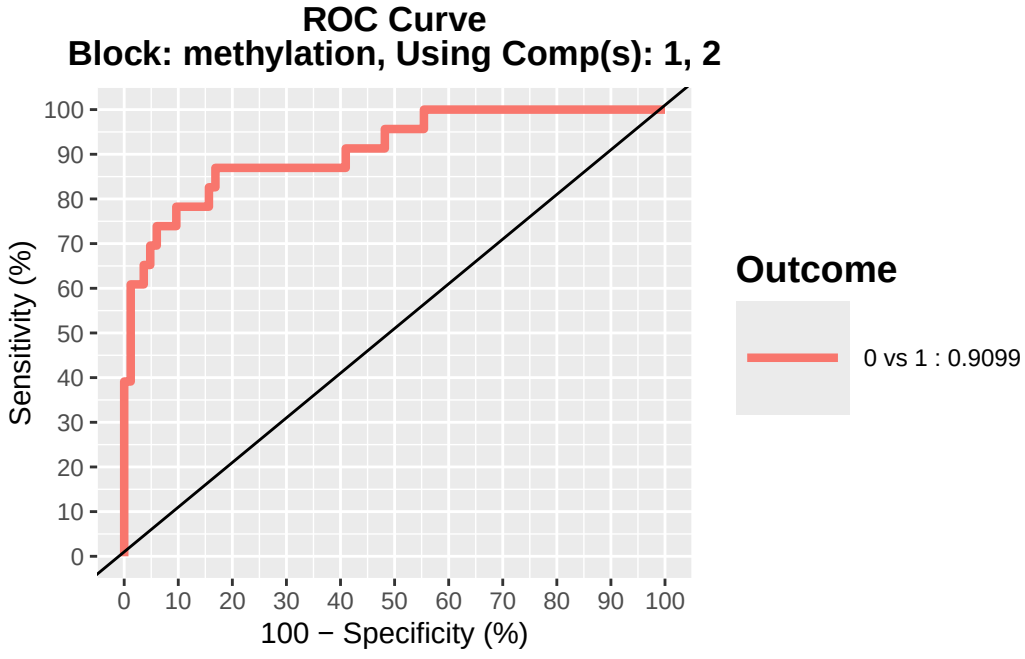
```
# cimDiablo(diablo.tcga, color.blocks = c('darkorchid', 'brown1', 'lightgreen'),
#           comp = 2, legend.position = "right")
```

```
# # Set desired width, height, and resolution
# # in pixels
# img_res <- 100     # in ppi
#
# # plotDiablo
# for (i in 1:4) {
#   png(filename = paste0("graphs/plotDiablo_n", i, ".png"),
#       width = img_width, height = img_height, res = img_res)
#   plotDiablo(diablo.tcga, ncomp = i)
#   dev.off()
# }
#
# # plotIndiv
# png(filename = "graphs/plotIndiv_n4.png", res = img_res)
# plotIndiv(diablo.tcga, ind.names = FALSE, legend = TRUE, title = 'Sample Plot')
# dev.off()
#
#
# # plotArrow
# png(filename = "graphs/plotArrow_n4.png", res = img_res)
# plotArrow(diablo.tcga, ind.names = FALSE, legend = TRUE, title = 'Arrow Plot')
# dev.off()
#
# # plotVar
# png(filename = "graphs/plotVar_n4.png", res = img_res)
# plotVar(diablo.tcga, var.names = FALSE, legend = TRUE, title = 'Correlation Circle Variable
# dev.off()
#
# # circosPlot
# png(filename = "graphs/circosPlot_n4.png", res = img_res)
# circosPlot(diablo.tcga, cutoff = 0.7, title = 'Circos Plot', size.labels = 1.5)
# dev.off()
#
# # network
# network(diablo.tcga, blocks = c(1,2,3),
#         cutoff = 0.88,
#         color.node = c('darkorchid', 'brown1', 'lightgreen'),
#         save = 'png', name.save = 'graphs/network_n4.png'
# )
```

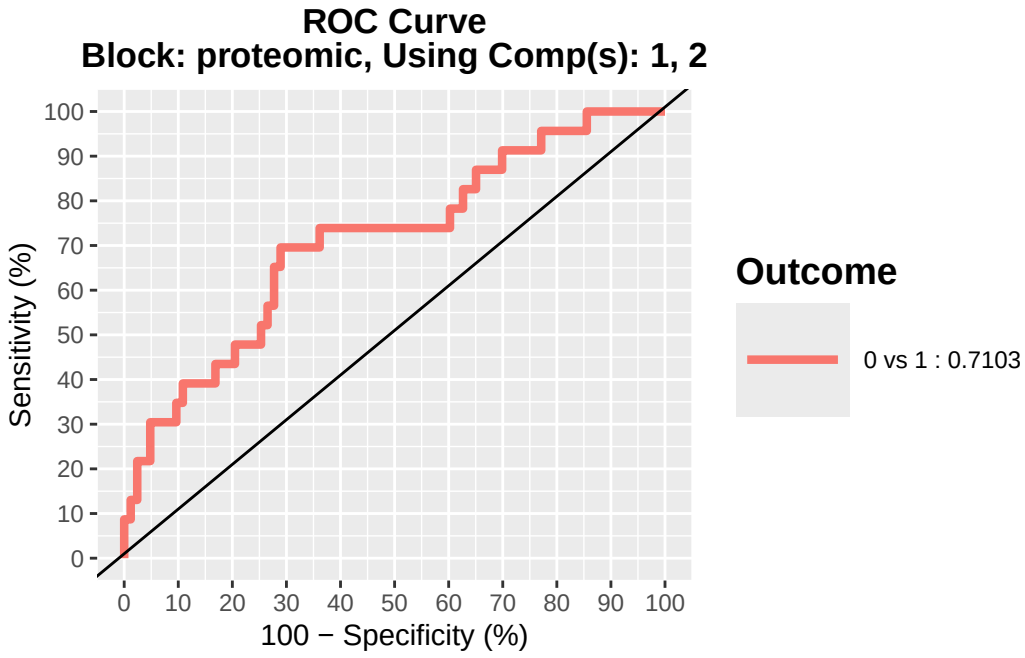
```
#
# # plotLoadings
# png(filename = "graphs/plotLoadings_n4.png", res = img_res)
# plotLoadings(diablo.tcga, comp = 4, contrib = 'max', method = 'median')
# dev.off()
#
#
# # cimDiablo
# png(filename = "graphs/cimDiablo_n4.png", res = img_res)
# cimDiablo(diablo.tcga, color.blocks = c('darkorchid', 'brown1', 'lightgreen'),
#           comp = 4, margin = c(8,20), legend.position = "right")
# dev.off()
```

```
# perf.diablo.tcga <- perf(diablo.tcga,  validation = 'Mfold', folds = 10,
#                          nrepeat = 10, dist = 'centroids.dist')
#
# perf.diablo.tcga$MajorityVote.error.rate
#
# perf.diablo.tcga$WeightedVote.error.rate
```

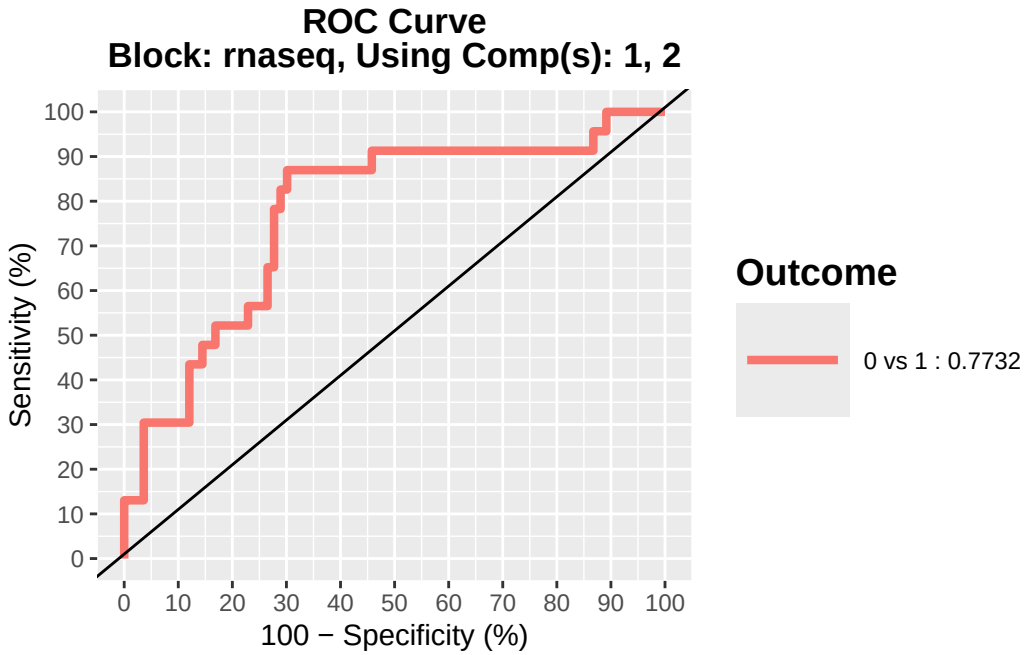
```
auc.diablo.tcga <- auroc(diablo.tcga, roc.block = "methylation", roc.comp = 2,
                  print = FALSE)
```



```
auc.diablo.tcga <- auroc(diablo.tcga, roc.block = "proteomic", roc.comp = 2,
                         print = FALSE)
```




```
auc.diablo.tcga <- auroc(diablo.tcga, roc.block = "rnaseq", roc.comp = 2,
                         print = FALSE)
```


ROC Curve
Block: rnaseq, Using Comp(s): 1, 2
Sensitivity (%)
100
90
80
70
60
50
40
30
20
10
0
0
10
20
30
40
50
60
70
80
90
100
100 − Specificity (%)
Outcome
0 vs 1 : 0.7732