sPLSDA and multiblock sPLSDA (DIABLO) using mixOmics

2022-10-28

First we will import mixOmics package and some others for this markdown to work

```
library(knitr)
library(mixOmics)
library(devtools)
library(factoextra)
```

Load Data

We'll use a subset of data from The Cancer Genome Atlas (TCGA). It is one of the largest collections of multi-omics data sets for more than 33 different types of cancer for 20 000 individual tumor samples.

As independent variable (X), we'll load in the dataset of miRNA expression levels for 150 samples and their corresponding tumour subtypes (Her2, Basal, or LumA) as categorical outcomes (Y). We'll use this subset to train our model and another to test later to test it.

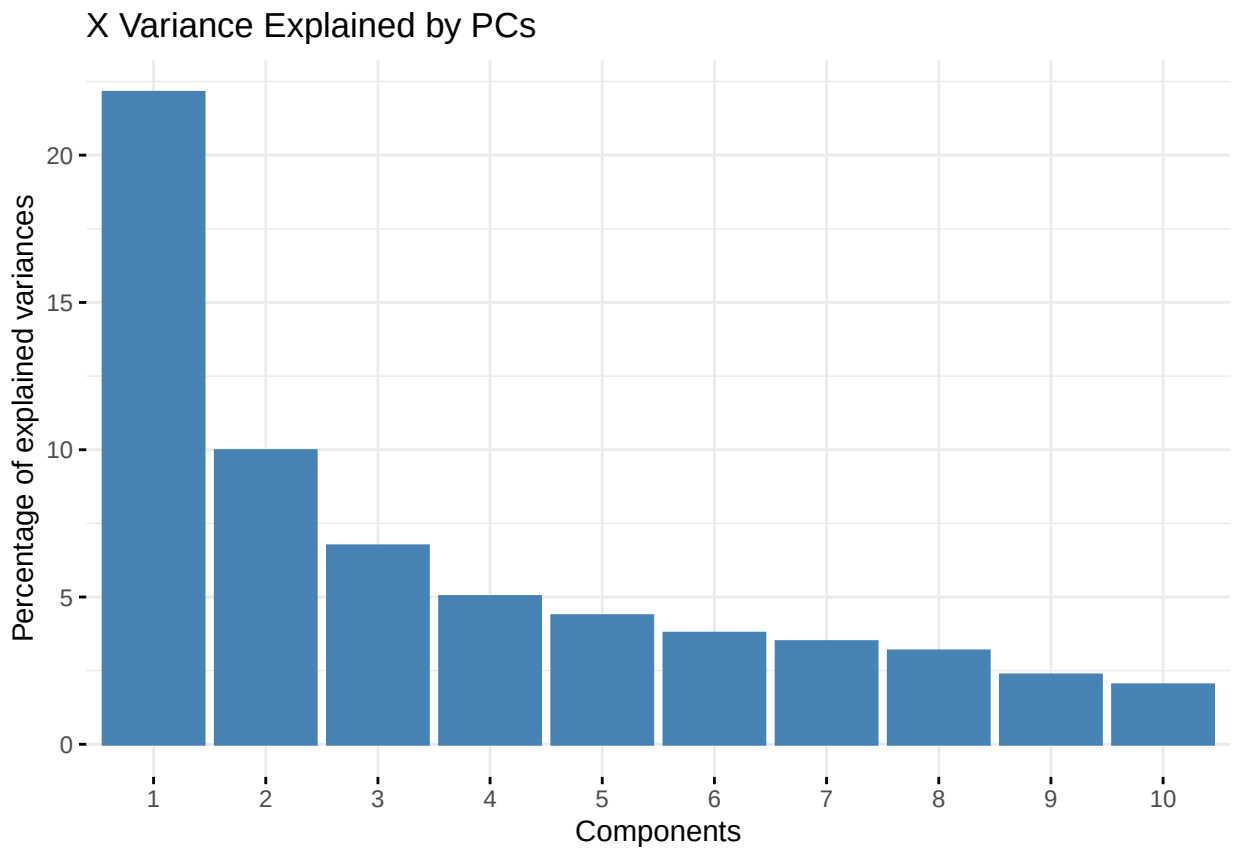
```
data("breast.TCGA")

X <- breast.TCGA$data.train$mirna # use miRNA expression data of 184 miRNA as X matrix
Y <- breast.TCGA$data.train$subtype # use tumour subtype as the Y matrix

head(X[,1:5]) # Columns are features (miRNAs) and rows are samples (individuals)
```

```
##      hsa-let-7a-1 hsa-let-7a-2 hsa-let-7a-3 hsa-let-7b hsa-let-7c
## A0FJ     11.83582     12.85105     11.91881   14.80138   10.93568
## A13E     12.89793     13.90087     12.91273   14.71551   12.03184
## A0G0     12.30773     13.29032     12.30062   15.06619   10.93393
## A0SX     12.03929     13.01081     12.08141   14.62003   11.47153
## A143     13.39097     14.38982     13.42228   15.30561   10.14690
## A0DA     12.34037     13.35986     12.39320   14.85705   11.37053
```

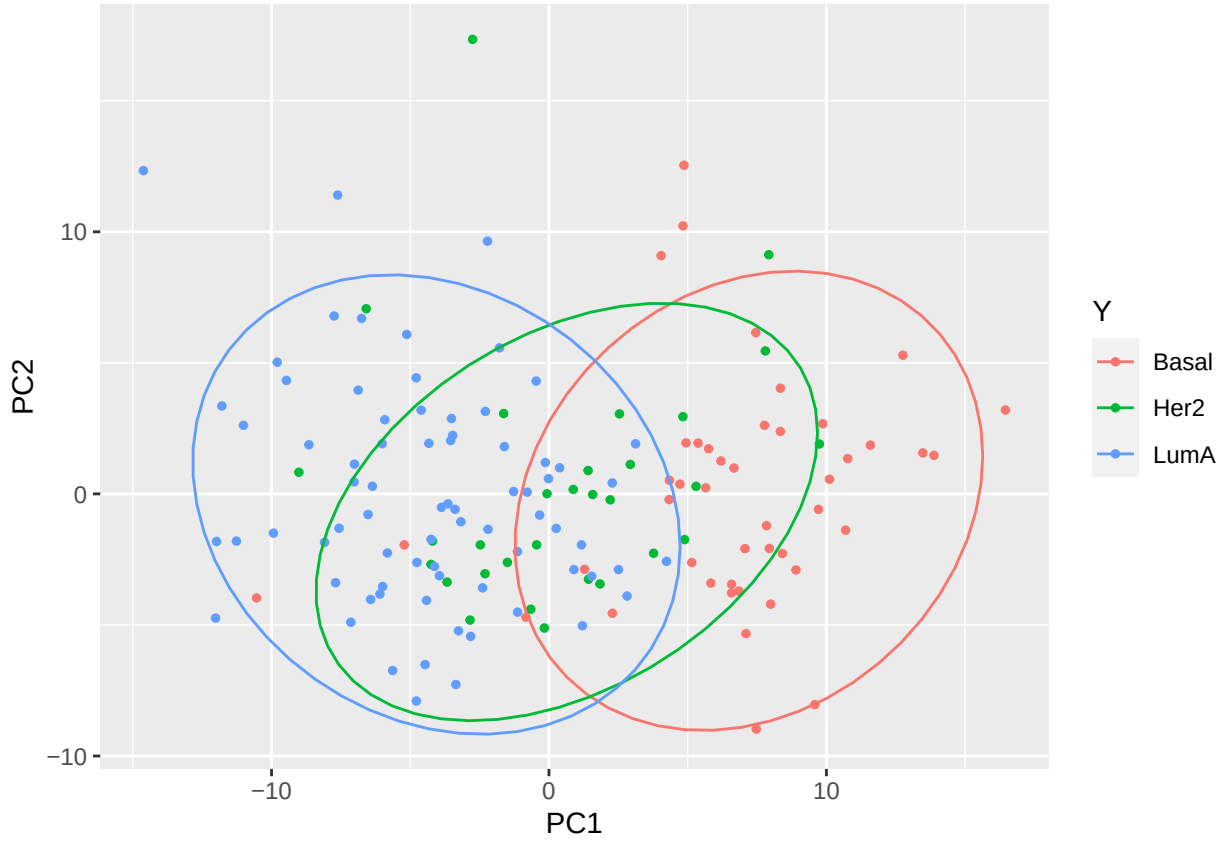
PCA first

Let's use our PCA on our X dataset

```
pca.miRNA <- prcomp(X, center = TRUE, scale = TRUE)

# mixOmics also has a pca() function that will do this for us
PCs = pca.miRNA$x # Gives us 150 PCs

fviz_eig(pca.miRNA, geom = "bar", main="X Variance Explained by PCs", xlab = "Components") # Scree plot
```




```
ggplot(as.data.frame(PCs), aes(y=PC2, x=PC1, colour=Y)) + geom_point(shape=19, size=1) + stat_ellipse()
```




```
# Clustering or classification looks pretty tough with PCA

# prcomp object also gives us the loading vectors if we want them  - they're in this rotation matrix
PCAloadings = pca.miRNA$rotation
head(PCAloadings[1:5,1:5]) # These columns are the loading vectors
```

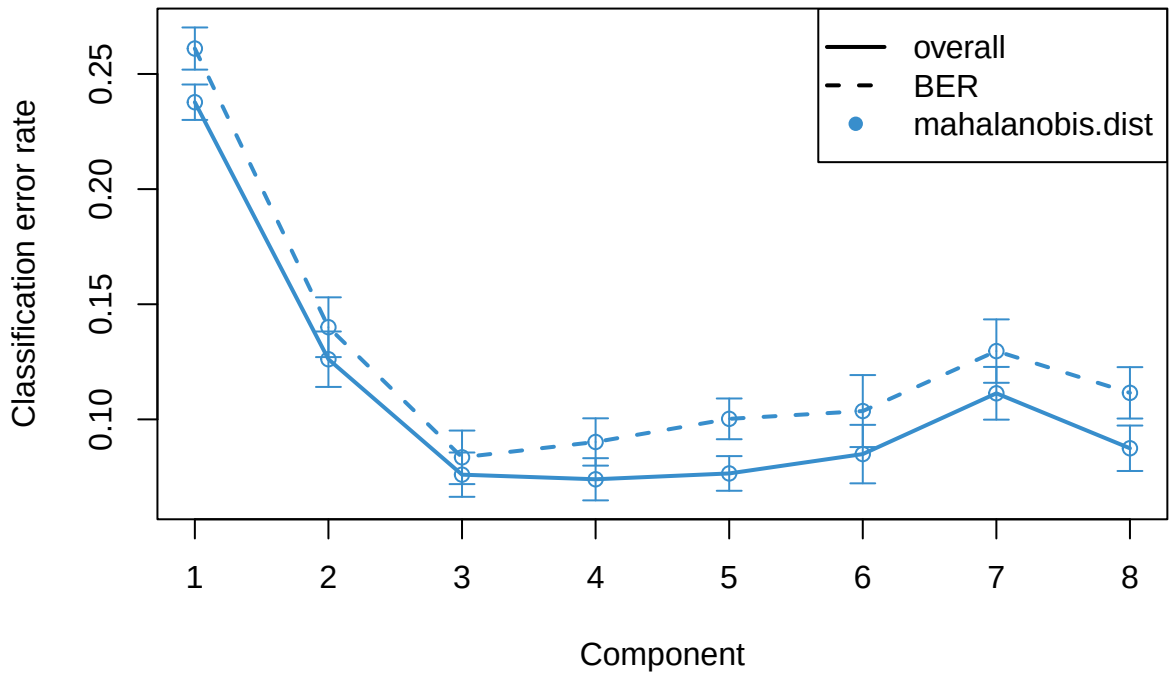
```
##                       PC1         PC2          PC3         PC4         PC5
## hsa-let-7a-1 -0.03715892 -0.05150582 -0.022878107  0.22862602  0.10373115
## hsa-let-7a-2 -0.03762300 -0.05177231 -0.021165738  0.22915339  0.10415396
## hsa-let-7a-3 -0.03643978 -0.05224021 -0.021049146  0.22905973  0.10385496
## hsa-let-7b   -0.06155440 -0.04102471  0.007648534  0.15086006 -0.08763844
## hsa-let-7c   -0.01030703 -0.10794518 -0.031440005 -0.06530195  0.09824572
```

Building sPLSDA model + Evaluation

Next let's use sPLSDA to get components that explain more than X. They explain X, explain Y and explain the relationship between X and Y. In addition we will use the sparse version of PLSDA to select for the most crucial features (miRNA).

```
# Let's feed both X and Y into splsda and at first we should specify how many components we would like
splsda.breast <- splsda(X = X, Y = Y, ncomp = 8)
# since we didn't specify keepX above - this is actually the same as plsda right now

# perf evaluates classification performance for (s)pls(da) objects and creates a perf object
```

```r
perf.splsda.breast <- perf(splsda.breast, folds = 10, nrepeat = 50,
                           dist = "mahalanobis.dist")

# Q2 total is a measure of error in our classification model
plot(perf.splsda.breast, criterion = 'Q2.total')
```

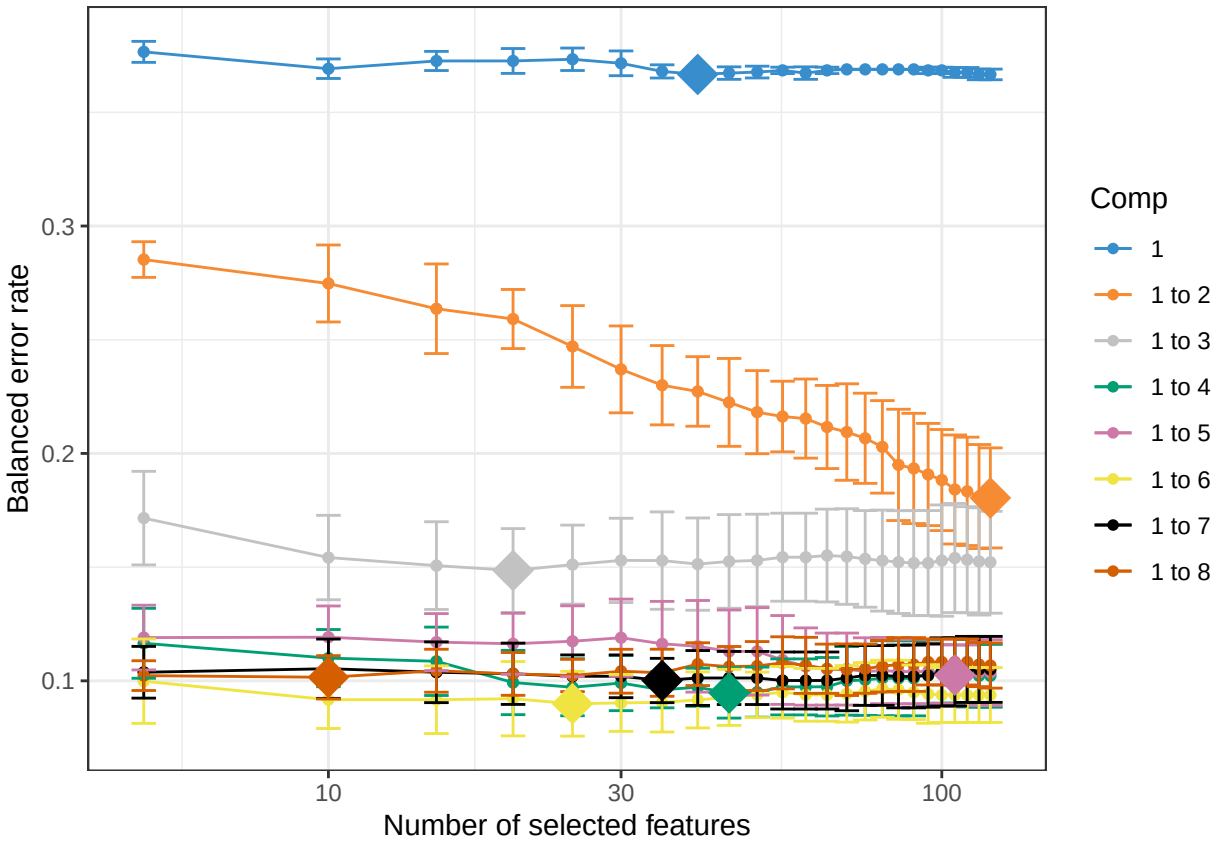



```r
# BER is balanced error rate for when there are different proportions of each class (as we have here)
```

Tuning Our Model

But we haven't really TUNED our model - our choice of using ncomp = 8 was arbitrary and not informed by the data or the regression. We also haven't really used the 'sparse' part of this method yet because we haven't added any selection pressure on the number of features to use - right now we are using them all like PCA. Let's tune some of these hyperparameters (primarily ncomp, keepX)

```r
# Let's tune the model over possible keepX from 5-120 in intervals of 5
list.keepX <- c(seq(5, 120, 5))

tune.splsda.breast <- tune.splsda(X, Y, ncomp = 8,
                                  test.keepX = list.keepX,
                                  nrepeat = 10, folds = 10)

# Again mixOmics helps make visualization easy
plot(tune.splsda.breast)
```



Newly Tuned Model + Evaluation

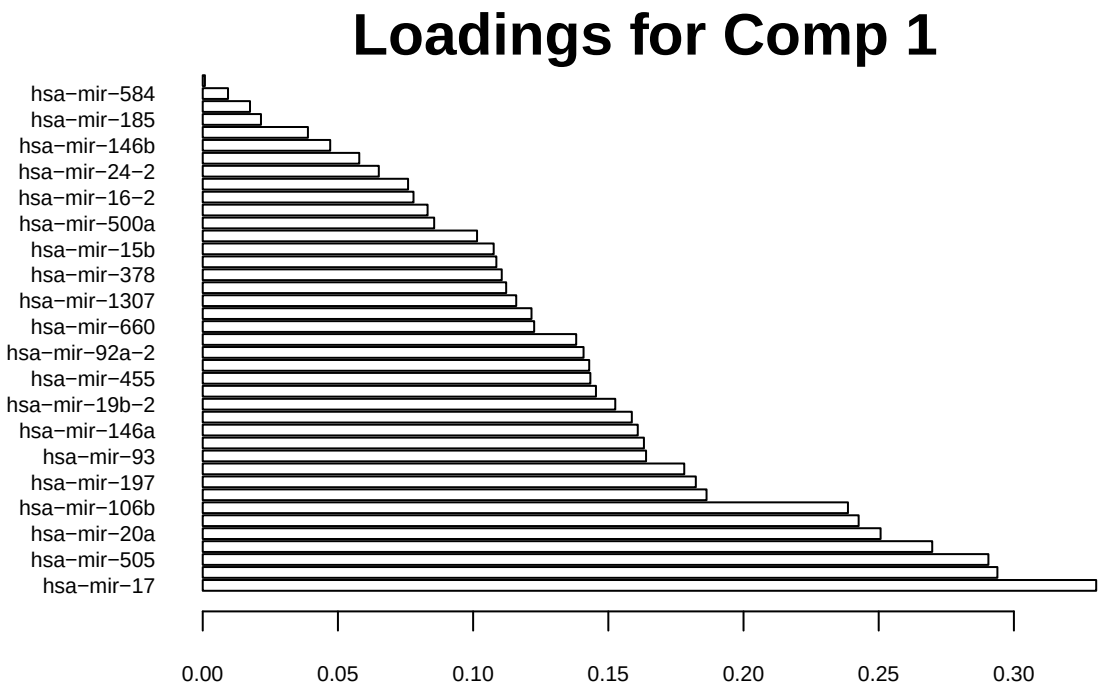
Now let's build the new model based on the hyperparameters we just tuned.

```
# extract optimal number of features to use for X dataframe and the optimal ncomp
optimal.keepX <- tune.splsda.breast$choice.keepX
optimal.ncomp <-  length(optimal.keepX) # extract optimal number of components

final.splsda.breast <- splsda(X, Y, ncomp = optimal.ncomp,
                          keepX = optimal.keepX)
# This is now sparse because we have selection pressure for X features

perf.final.splsda.breast <- perf(final.splsda.breast, dist = "mahalanobis.dist",
                                 folds = 10, nrepeat = 50)
```

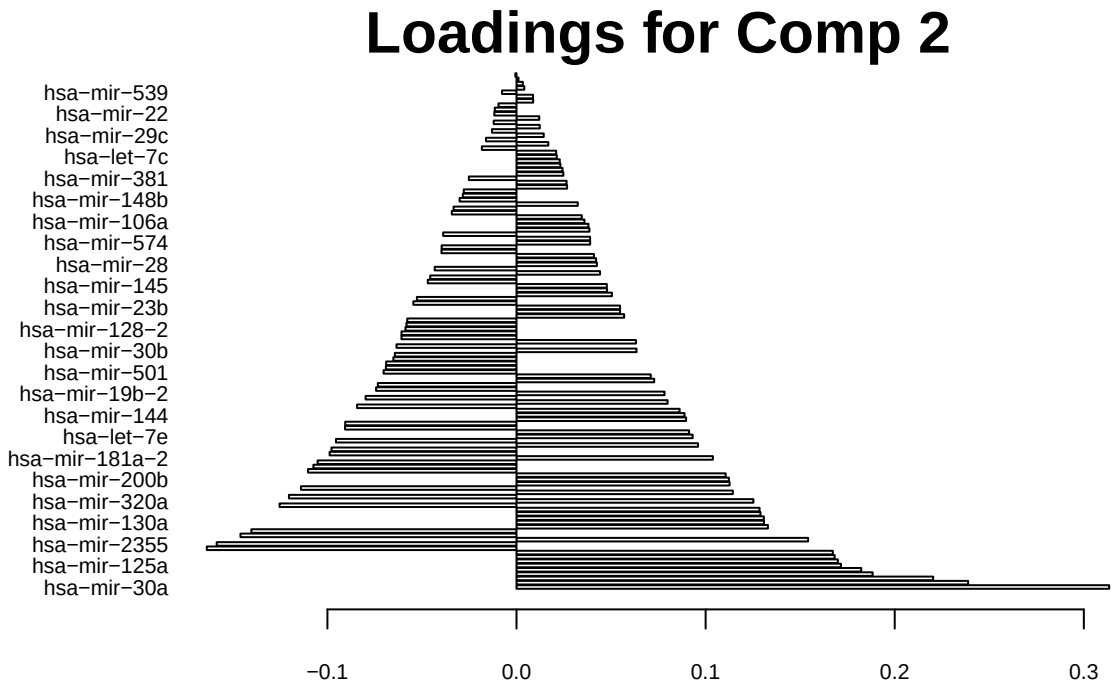
Let's Visualize the process!

Now let's use the visualizations simplified by mixOmics to visualize the process and results.

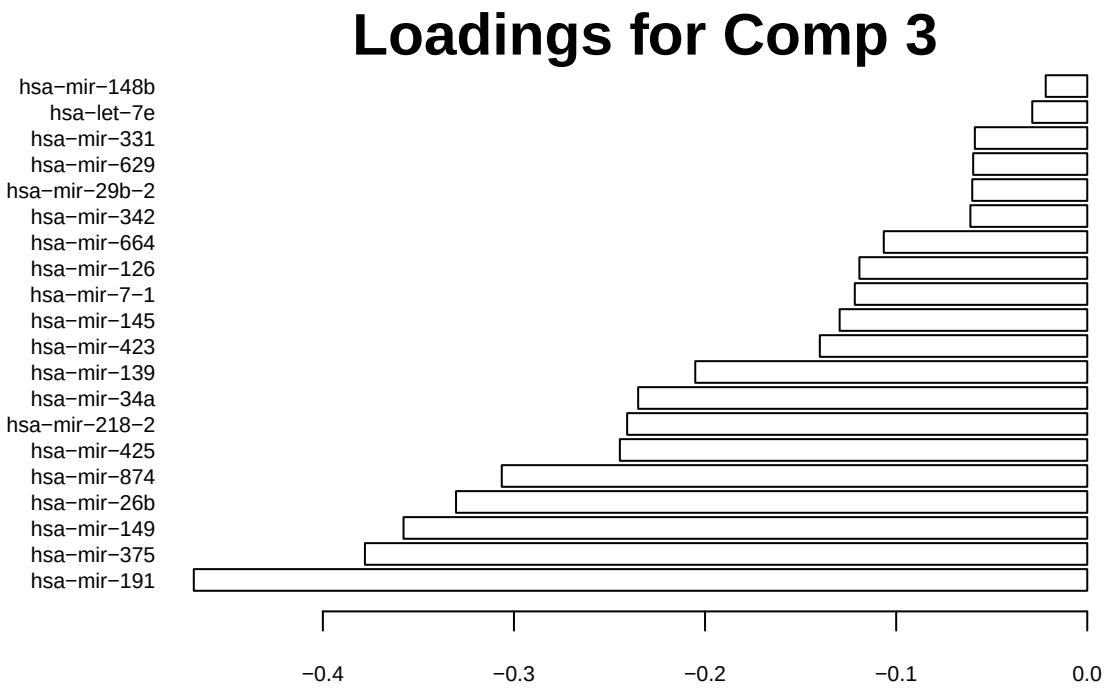
```
# Let's visualize what loading vectors went into making the model for the first 3 comp
plotLoadings(final.splsda.breast, comp = 1, title = "Loadings for Comp 1")
```



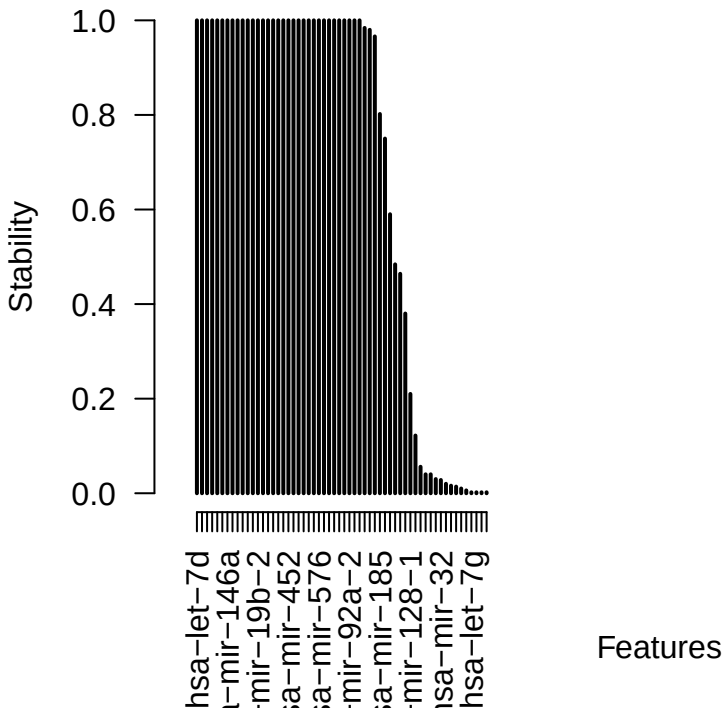

```
plotLoadings(final.splsda.breast, comp = 2, title = "Loadings for Comp 2")
```



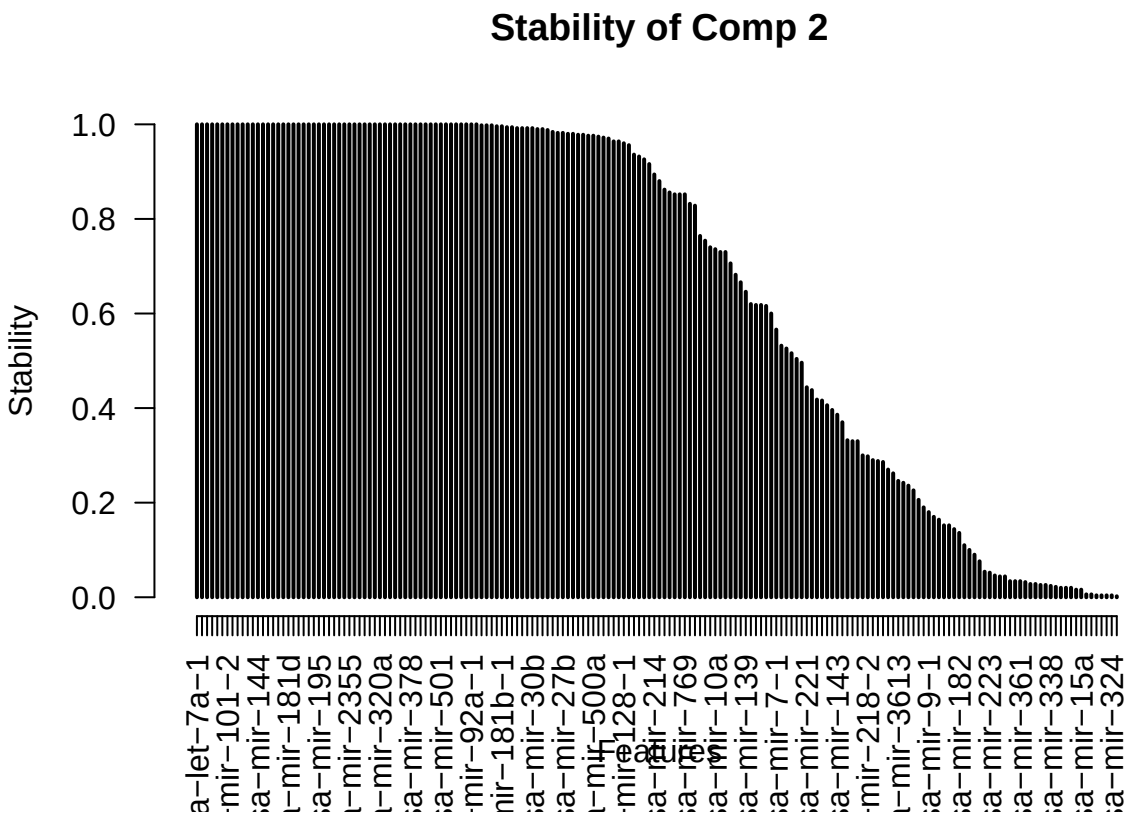

```
plotLoadings(final.splsda.breast, comp = 3, title = "Loadings for Comp 3")
```




```r
# Let's visualize how stable the model is - how often it is selecting each feature through folds/repeat
plot(perf.final.splsda.breast$features$stable$comp1, type = 'h',
     ylab = 'Stability',
     xlab = 'Features',
     main = 'Stability of Comp 1', las =2,
     xlim = c(0, 184),
     ylim = c(0, 1))
```

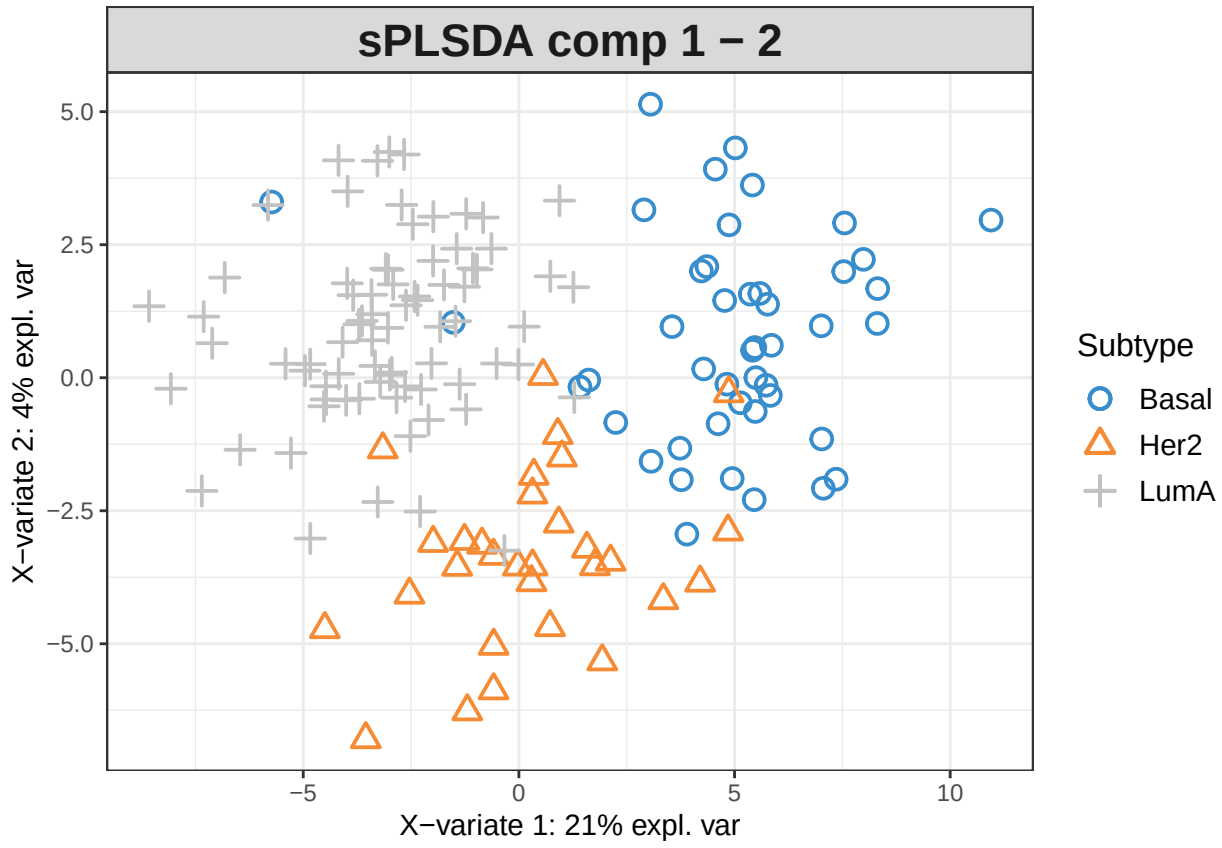



```
plot(perf.final.splsda.breast$features$stable$comp2, type = 'h',
     ylab = 'Stability',
     xlab = 'Features',
     main = 'Stability of Comp 2', las =2,
     xlim = c(0, 184),
     ylim = c(0, 1))
```

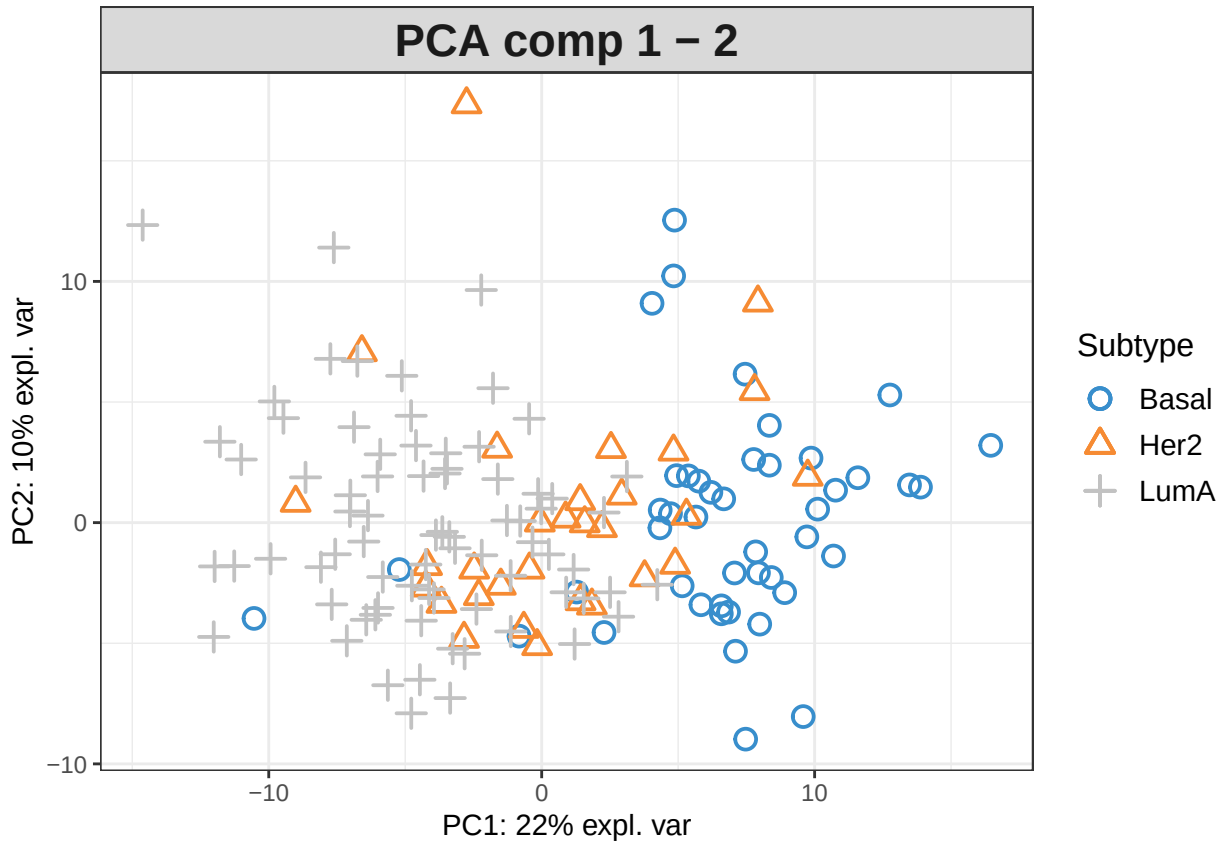


Let's visualize the results!

```
# We can also plot our PCA but have to use the mixOmics PCA to plot with plotIndiv()
pca.miRNA = pca(X, center = TRUE, scale = TRUE)

plotIndiv(final.splsda.breast, ind.names = FALSE,
          rep.space = "X-variate",
          group = breast.TCGA$data.train$subtype, # colour by group
          col.per.group = color.mixo(1:3),
          legend = TRUE, legend.title = 'Subtype', title = 'sPLSDA comp 1 - 2')
```




```
# Let's compare plots
plotIndiv(pca.miRNA, comp = c(1, 2), ind.names = FALSE,
          group = breast.TCGA$data.train$subtype,
          legend = TRUE, legend.title = 'Subtype', title = 'PCA comp 1 - 2')
```




```
# Some more complex visualizations
# col.tox <- color.mixo(as.numeric(as.factor(c(1,3)))) # create set of colours
# library(rgl) # we'll use this to make a 3D plot
# plotIndiv(final.splsda.breast, ind.names = FALSE, rep.space = "XY-variate", axes.box = "both", style =

# Some plots that only really appropriate when using multiblock.splsda
```

Predictive Modelling

Now let's use the model to predict category when we only have the X input matrix (this testing dataset has 70 samples)

```
predict.model = predict(final.splsda.breast, breast.TCGA$data.test$mirna)

confusion.mat <- get.confusion_matrix(
                truth = breast.TCGA$data.test$subtype,
                predicted = predict.model$MajorityVote$mahalanobis.dist[,5])
kable(confusion.mat)
```

	predicted.as.Basal	predicted.as.Her2	predicted.as.LumA
Basal	18	3	0
Her2	0	13	1
LumA	0	3	32

Same Methods but Multi-block with DIABLO

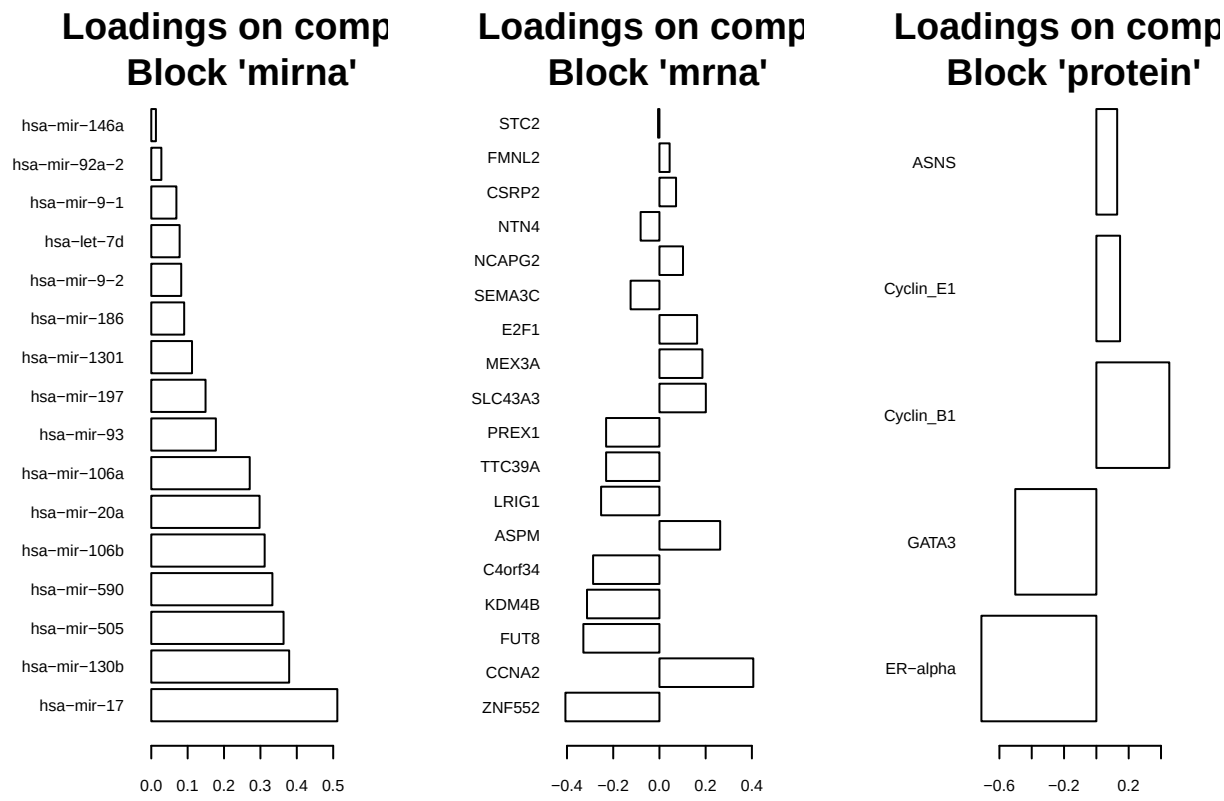
If we want to perform sPLSDA on N-integrated datasets (omics data) we can multiblock.splsda, the mixOmics framework is called DIABLO. The procedure is very similar but we add datasets to our X input. Below I have included the code for an example DIABLO.

Let's add mRNA expression levels and proteomics to the input data. We will import omics datasets on breast cancers as independent variables and their subtypes (Her2, Basal, or LumA) as categorical outcomes just as before

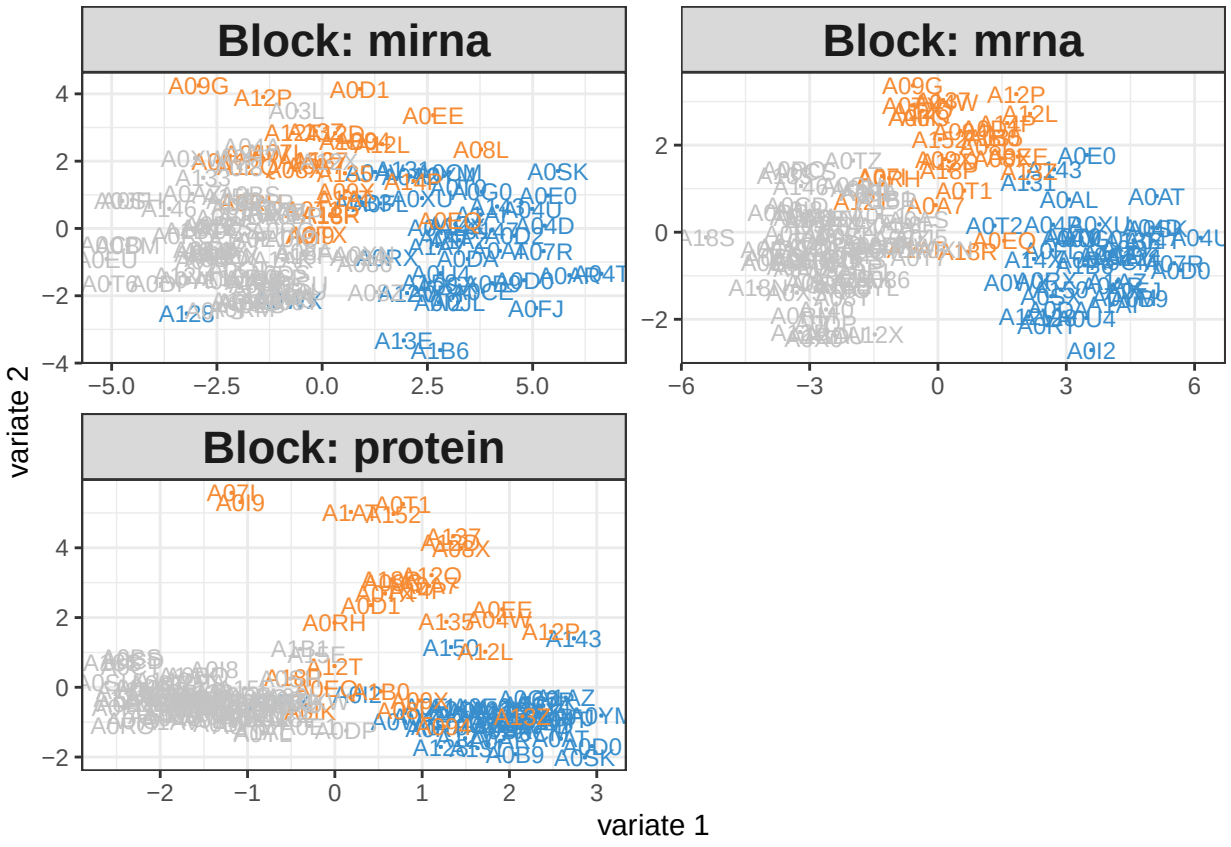
```
# use the mirna, mrna and protein expression levels as predictive datasets
# note that each dataset is measured across the same individuals (samples)
X1 <- breast.TCGA$data.train$mirna
X2 <- breast.TCGA$data.train$mrna
X3 <- breast.TCGA$data.train$protein
X_all <- list(mirna = X1, mrna = X2, protein = X3)
Y_all <- breast.TCGA$data.train$subtype # use the subtype as the outcome variable
```

```
list.keepX = list(mirna = c(16, 17), mrna = c(18,5), protein = c(5,5))
result.sparse.diablo.breast <-  block.splsda(X_all, Y_all, keepX = list.keepX, ncomp = 5)
```

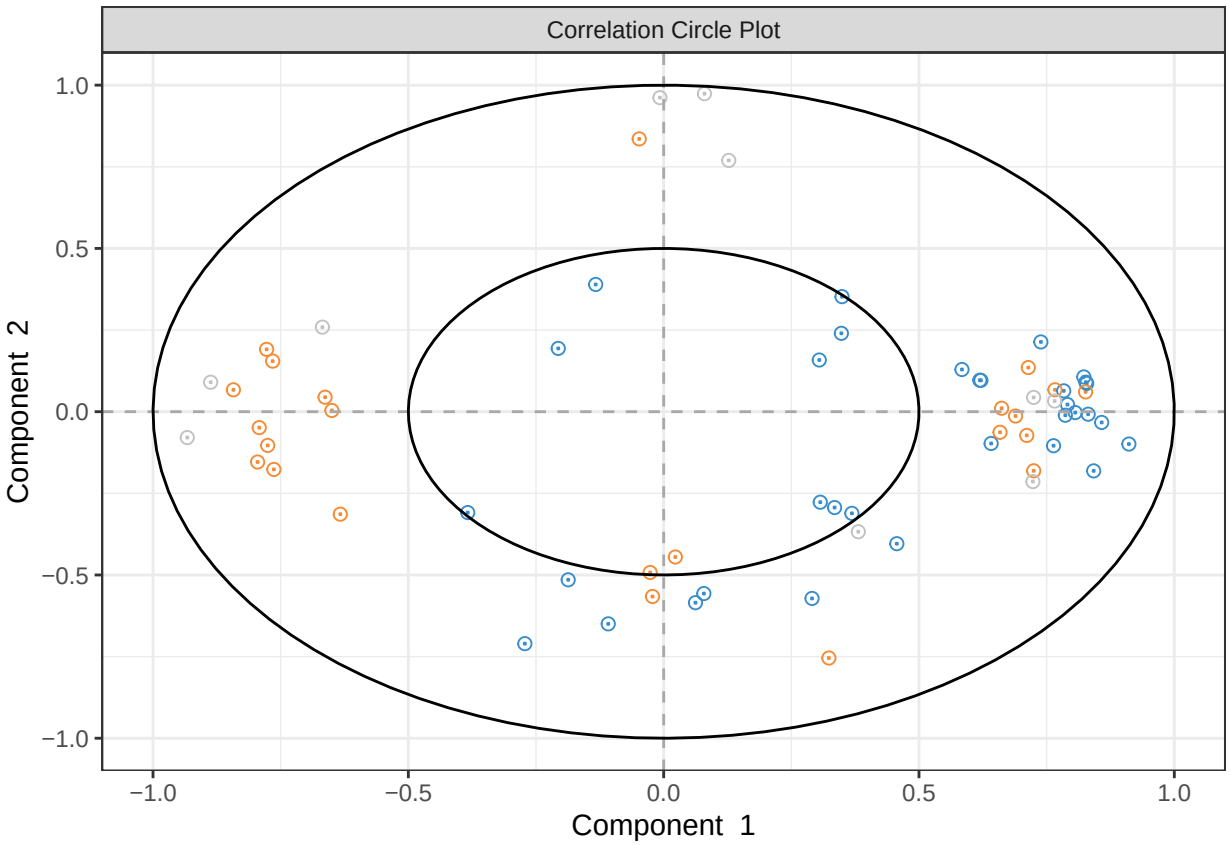
```
# plot the contributions of each feature to each component 1
plotLoadings(result.sparse.diablo.breast, ncomp = 1)
```



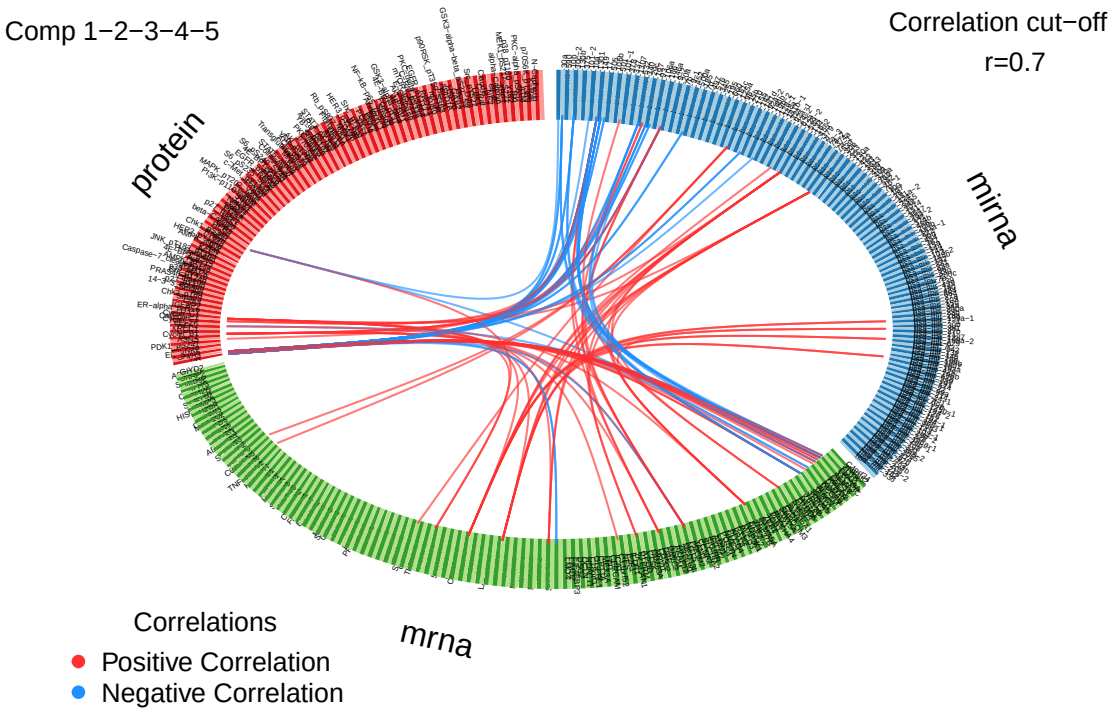
```
plotIndiv(result.sparse.diablo.breast, var.names = FALSE) # plot the samples
```




```
plotVar(result.sparse.diablo.breast, cex = c(2,2,2), var.names = FALSE) # plot the variables
```




```
circosPlot(result.sparse.diablo.breast, cutoff = 0.7)
```




```
pdf("Our_cim_plot_diablo")
cimDiablo(result.sparse.diablo.breast)
```

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

```
dev.off()
```

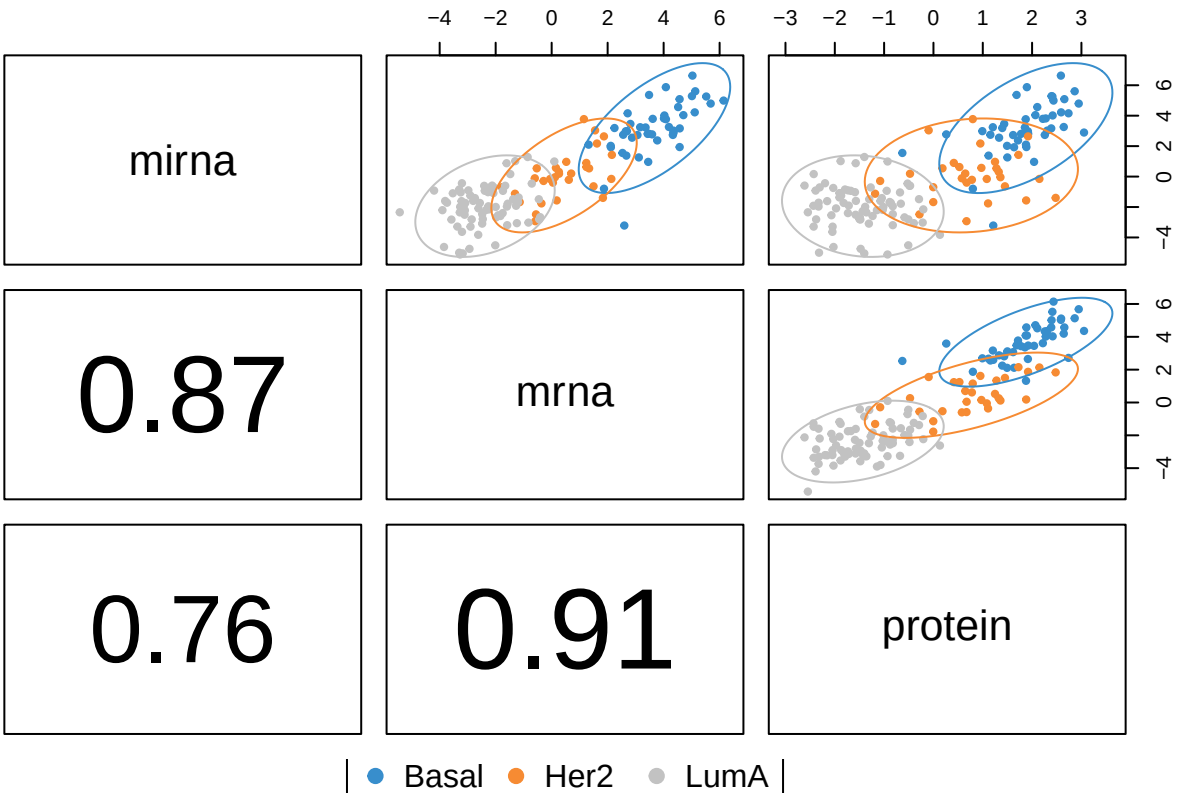
```
## pdf
##   2
```

Multiomics Integrative Predictive Modelling

```
predict.model.diablo = predict(result.sparse.diablo.breast, list(mirna = breast.TCGA$data.test$mirna, mr
```

```
## Warning in predict.block.spls(result.sparse.diablo.breast, list(mirna =
## breast.TCGA$data.test$mirna, : Some blocks are missing in 'newdata'; the
## prediction is based on the following blocks only: mirna, mrna
```

```
# Note protein data is missing for prediction and that is OK. This is explained in warning below

plotDiablo(result.sparse.diablo.breast)
```




```
confusion.mat_diablo <- get.confusion_matrix(
                truth = breast.TCGA$data.test$subtype,
                predicted = predict.model.diablo$WeightedVote$mahalanobis.dist[,5])
kable(confusion.mat_diablo)
```

	predicted.as.Basal	predicted.as.Her2	predicted.as.LumA
Basal	20	1	0
Her2	0	14	0
LumA	0	0	35

And we didn't even properly tune this model! Imagine the possibilities...

End