sPLSDA and multiblock sPLSDA (DIABLO) using mixOmics

2022-10-28

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

```
rm(list = ls()) # Make sure starting with a clean environment
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, mixOmics also has a `pca()` function that will do this for us.

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

PCs = pca.miRNA$x # Gives us 150 PCs

fviz_eig(pca.miRNA, geom = "bar", main="X Variance Explained by PCA Components", xlab = "Component", yla
```

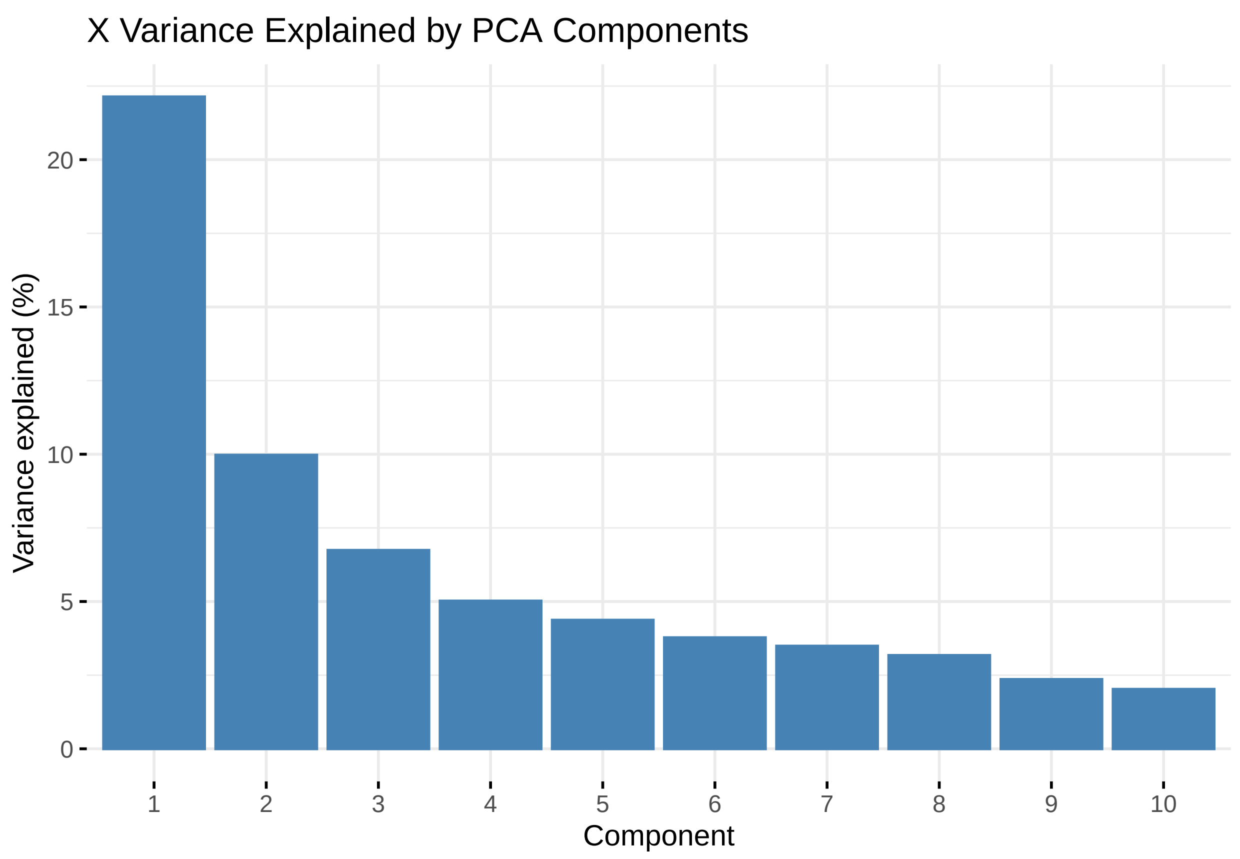


Figure 1: Scree plot of variance explained by each first 10 components.

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

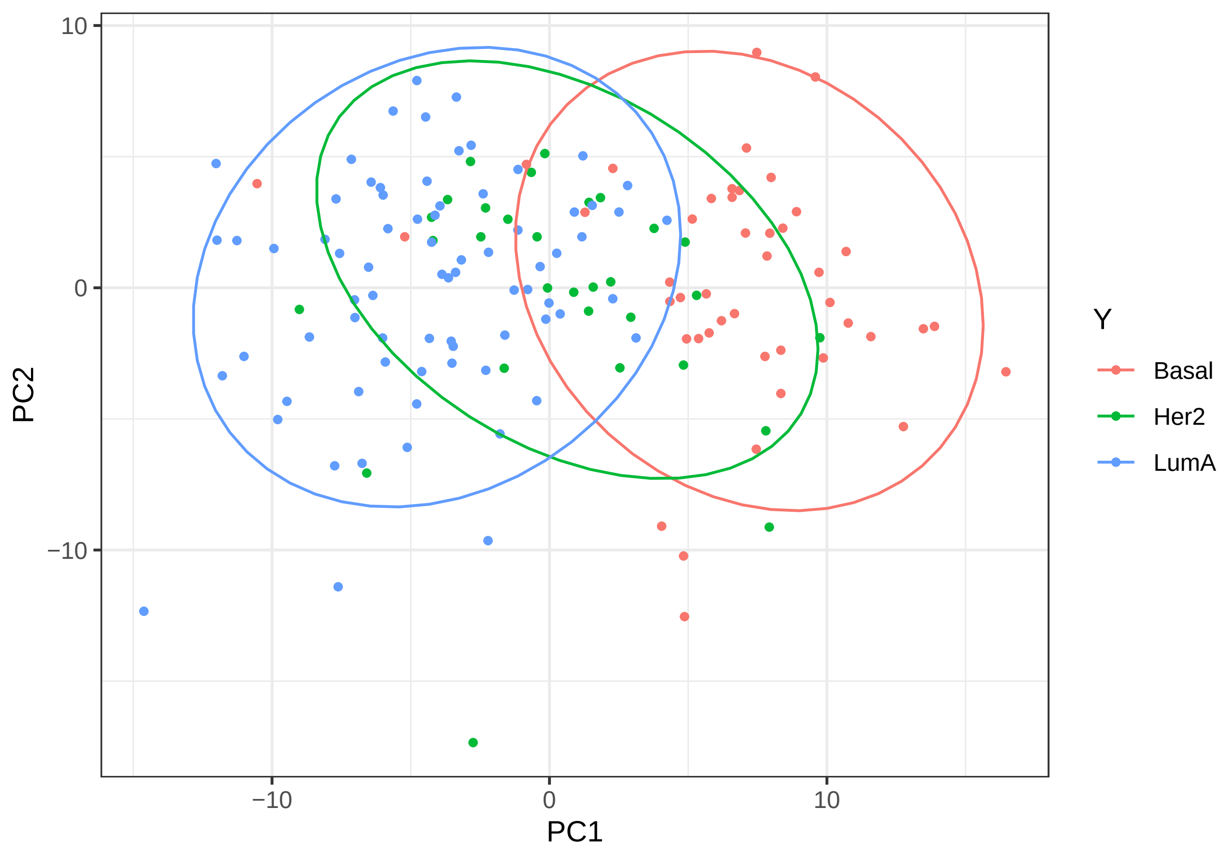


Figure 2: Clustering of tumour types in PC1 and PC2.

Clustering or classification looks pretty tough with these PCs

The `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]) # 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 (miRNAs that matter most to determining tumour subtype).

Let's feed both X and Y into splsda and at first we should specify how many components we would like to come up with, here 8 is used (just arbitrary)

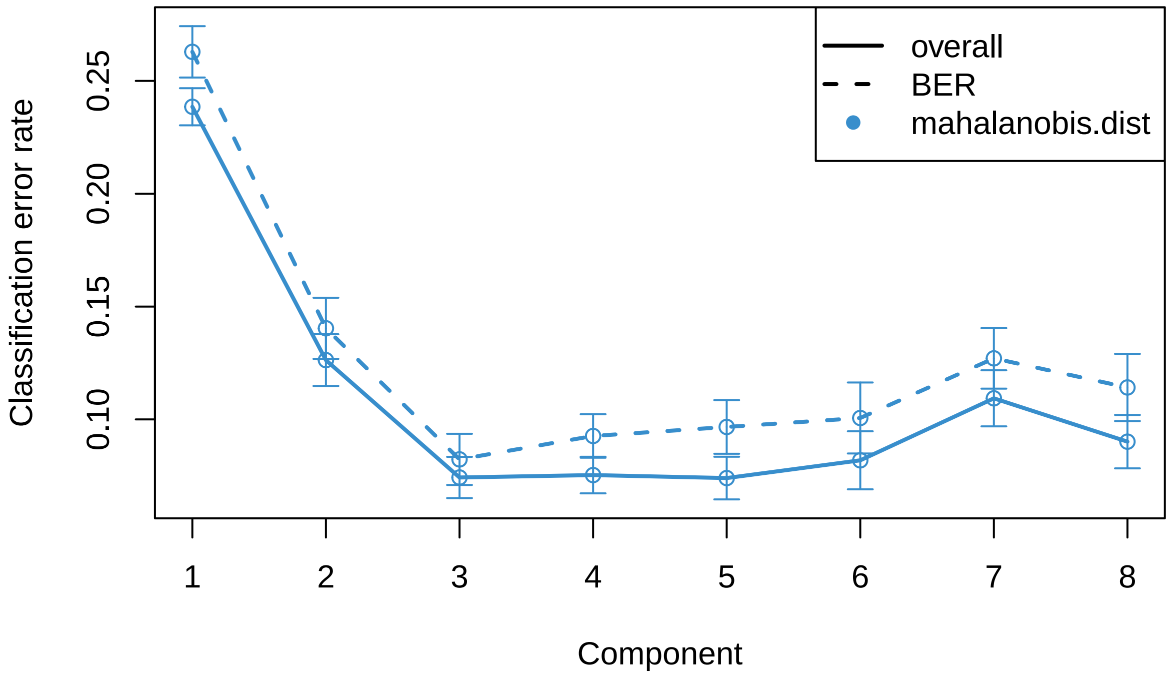
```r
splsda.breast <- splsda(X = X, Y = Y, ncomp = 8)
```

Since we didn't specify a `keepX` argument above - this is actually the same as plsda for 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")

plot(perf.splsda.breast, criterion = 'Q2.total') # Plot the perf object
```



`Q2 total` is a measure of error in our classification model. BER is balanced error rate, most appropriate when there are class imbalances (as we have here)

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 feature selection - right now we are using them all like PCA.

Tuning Our Model

Let's tune some of these hyperparameters (primarily ncomp, keepX). We'll optimize over a range of keepX values (5-120 in intervals of 5).

```r
list.keepX <- c(seq(5, 120, 5))
```

```r
# Optimize over possible keepX values
tune.splsda.breast <- tune.splsda(X, Y, ncomp = 8,
                                  test.keepX = list.keepX,
                                  nrepeat = 10, folds = 10)

plot(tune.splsda.breast) # mixOmics makes visualization easy
```

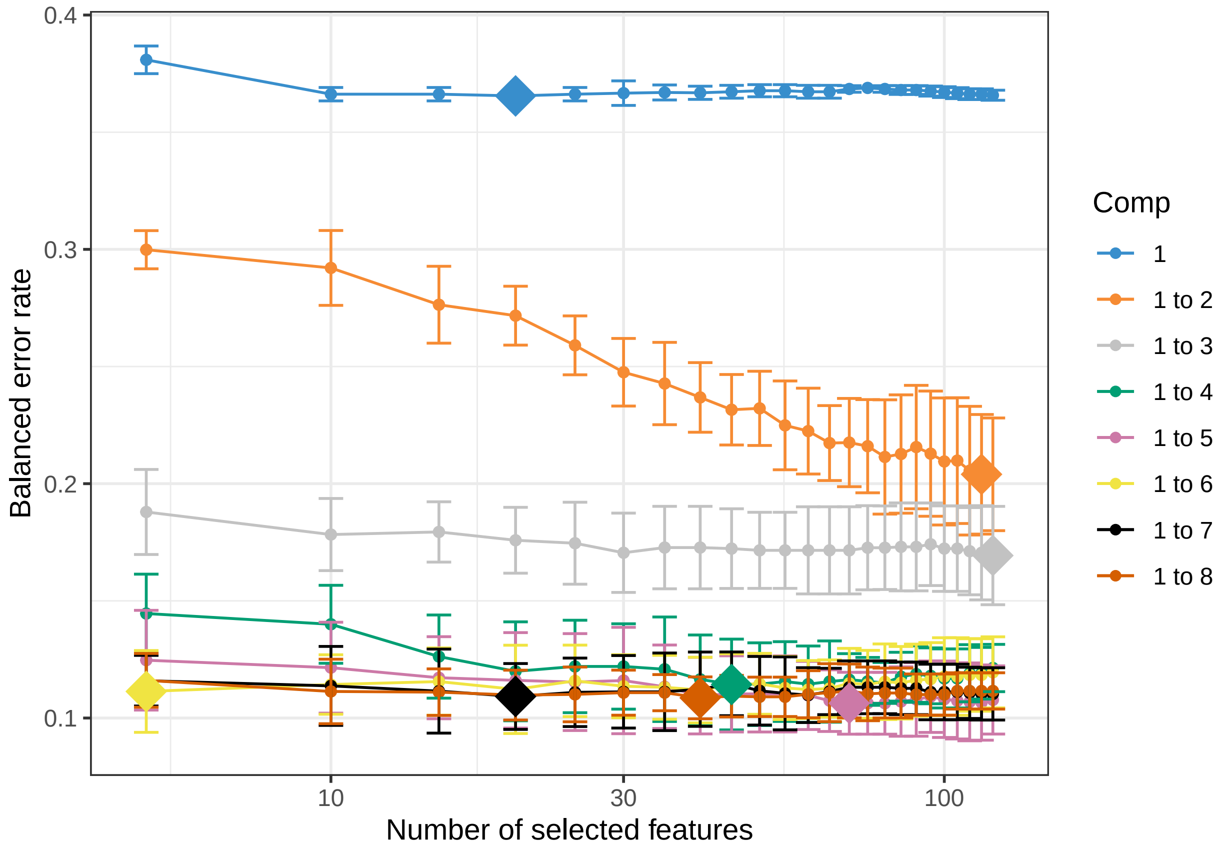


Figure 3: Tuning of model by examining balanced classification error as more components are added to the model.

The tuning involves 8 models that vary in their number of components (ncomp) where fewer is better, and examining how error rate changes as number of features (keepX) varies. Regardless of how many features are included in the model, a model based on 1 component (ncomp=1) has consistently higher error rate, the model based on the first 3 components (ncomp=3) has consistently lower error but all model based on 4 or more components (ncomp>=4) consistently have the lowest error rate. A model based on the first 2 components is the only that appears to improve significantly as more features are included in the model.

Newly Tuned Model + Evaluation

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

We have to extract the optimal `keepX` and the optimal `ncomp`. This now counts as *sparse* PLS-DA because we have feature selection.

```r
optimal.keepX <- tune.splsda.breast$choice.keepX
optimal.ncomp <-  length(optimal.keepX) # extract optimal number of components
```

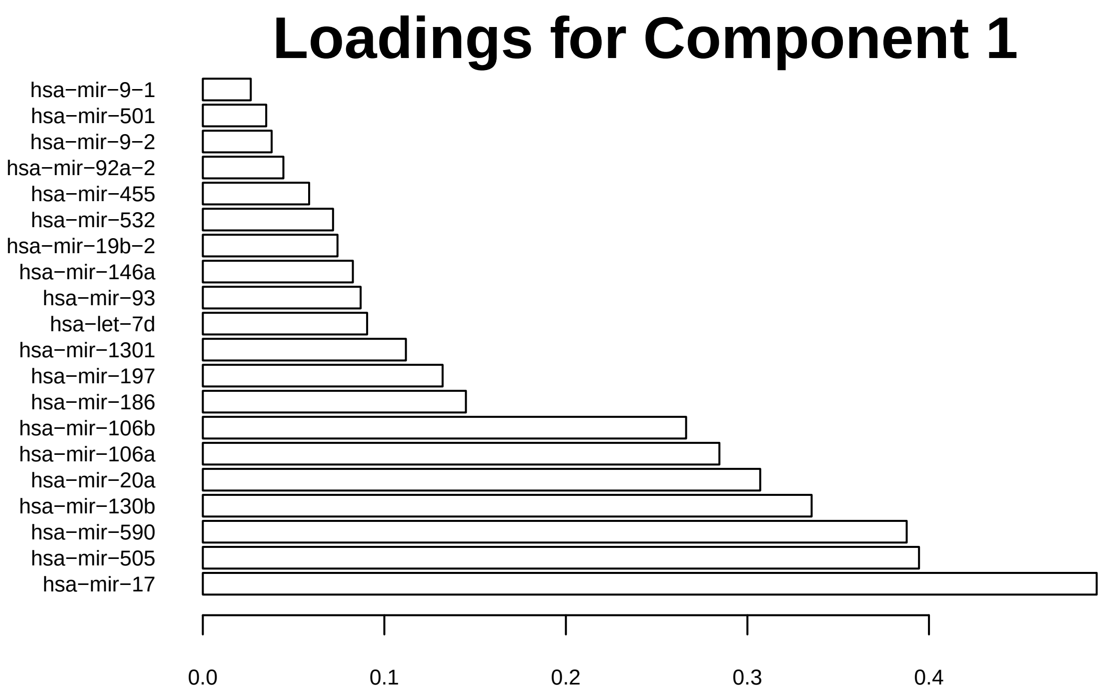
```r
final.splsda.breast <- splsda(X, Y, ncomp = optimal.ncomp,
                         keepX = optimal.keepX)

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

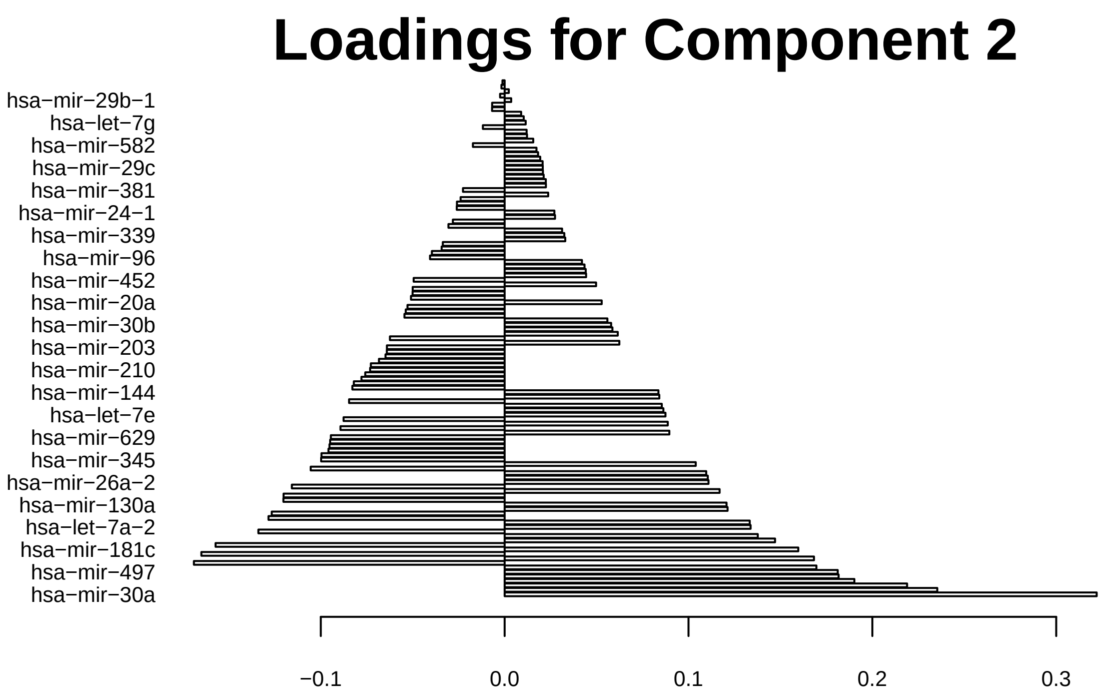
Let's Visualize the process!

Now let's use the visualizations simplified by mixOmics to see the process and results. What loading vectors go into making the first 3 components?

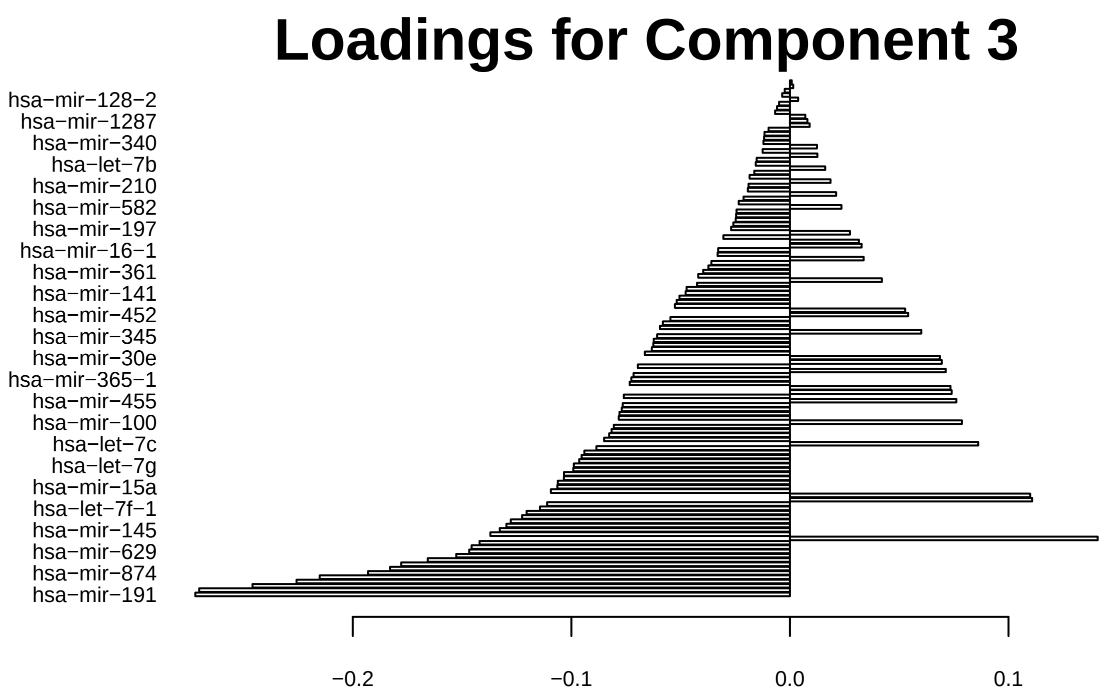
```r
plotLoadings(final.splsda.breast, comp = 1, title = "Loadings for Component 1")
```



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



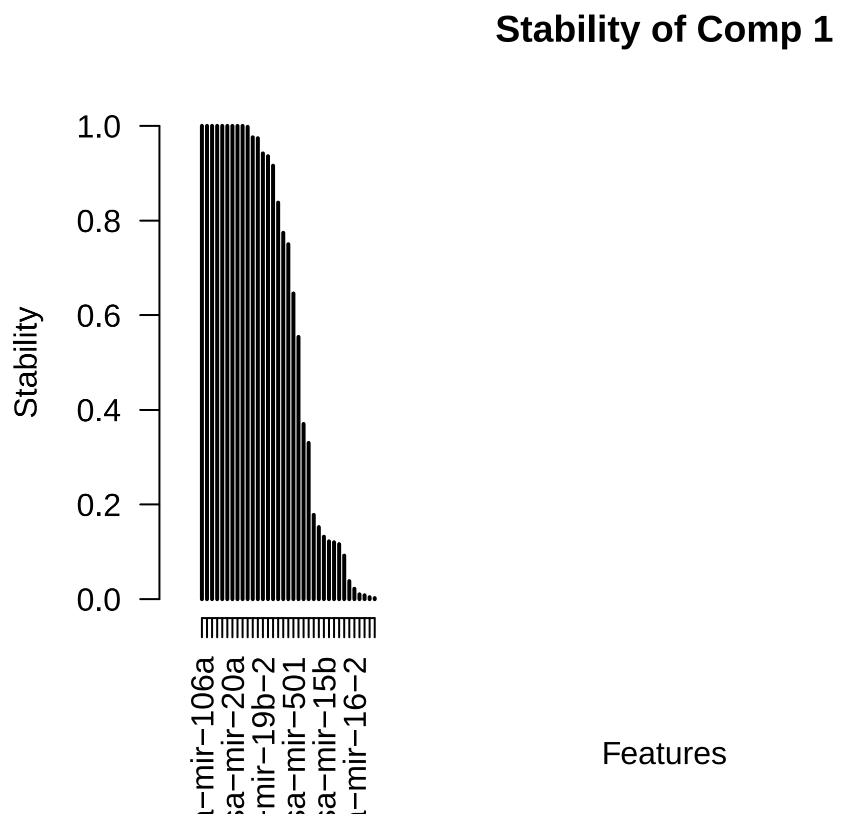
```r
plotLoadings(final.splsda.breast, comp = 3, title = "Loadings for Component 3")
```



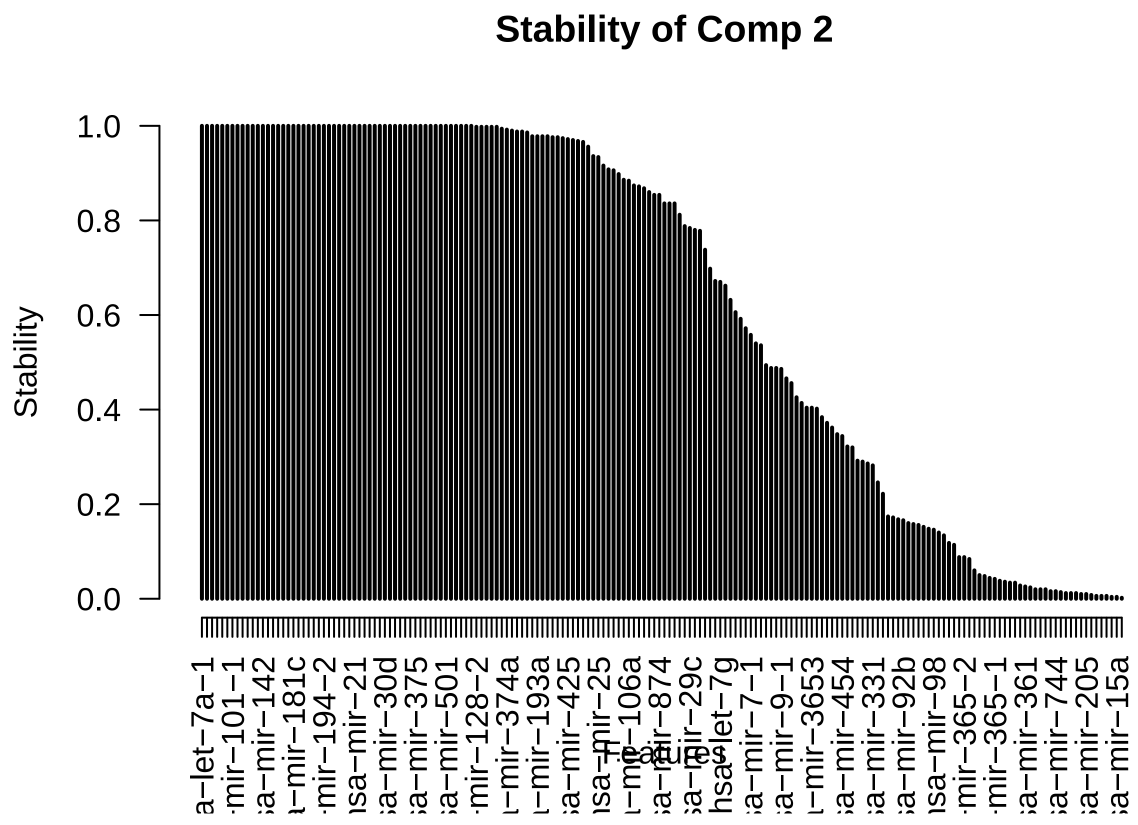
Let's visualize how stable the model is.

How often it is selecting the same features as most informative through folds/repeats.

```
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))
```



```r
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))
```

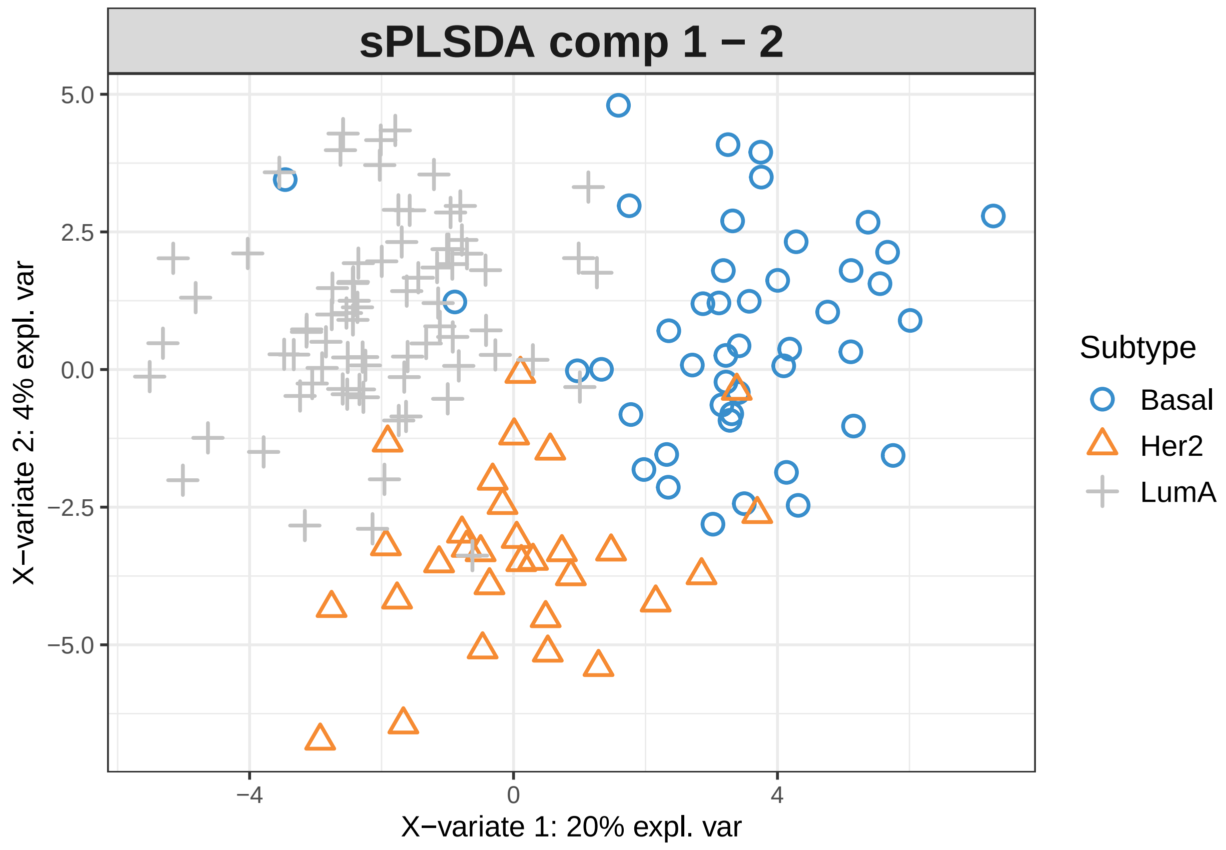


Let's visualize the results!

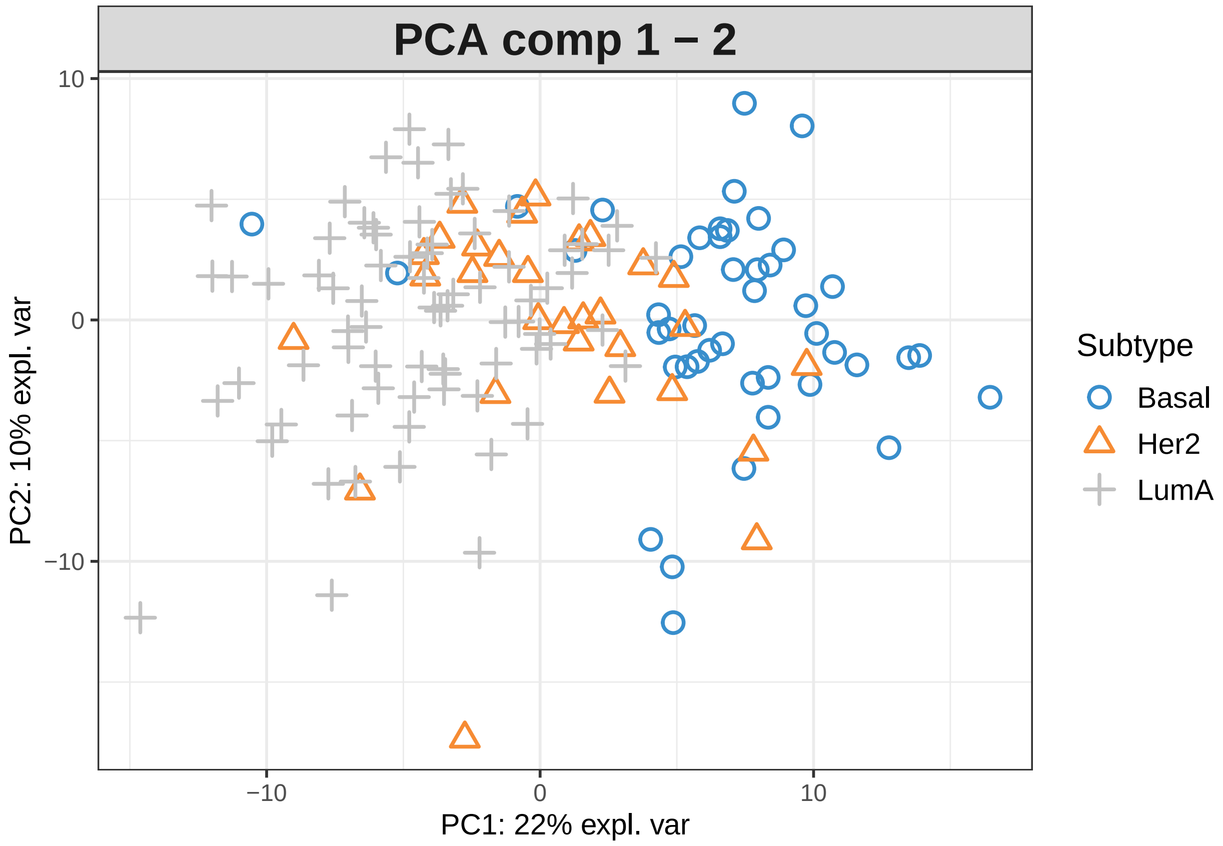
Compare the plot of sPLS-DAs component 1&2 vs PCAs component 1&2 (here using mixOmics `plotIndiv()` function)

```r
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')
```



```r
# 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')
```



Much better clustering with sPLS-DA than PCA.

Some more complex visualizations can also be made relatively easily, excluded here for brevity.

```
# 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 tumour category when we only have the X input matrix (miRNA expression levels). This test set 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	4	31

Same Methods but Multi-block with DIABLO

If we want to perform sPLSDA on N-integrated datasets (omics data) we can use `multiblock.splsda`, the mixOmics framework is called DIABLO. The procedure is very similar but we add parallel datasets of another modality, like mRNA or protein expression levels, to our X input.

Example DIABLO

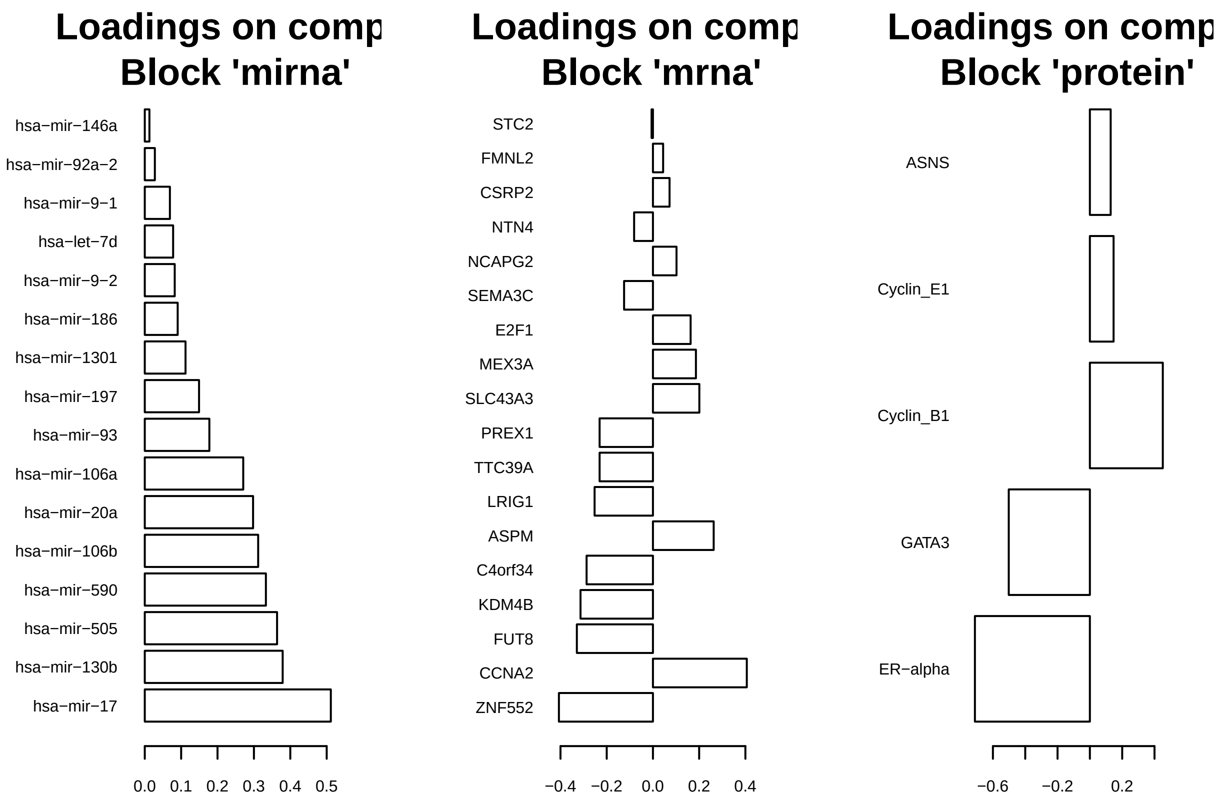
Let's add mRNA expression levels and proteomics to the input data. We will import omics datasets on breast cancers as independent variables and their tumour subtypes (Her2, Basal, or LumA) as categorical outcomes just as before. Note: each dataset contains measures from the same individuals.

```
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 tumour subtype as the outcome variable
```

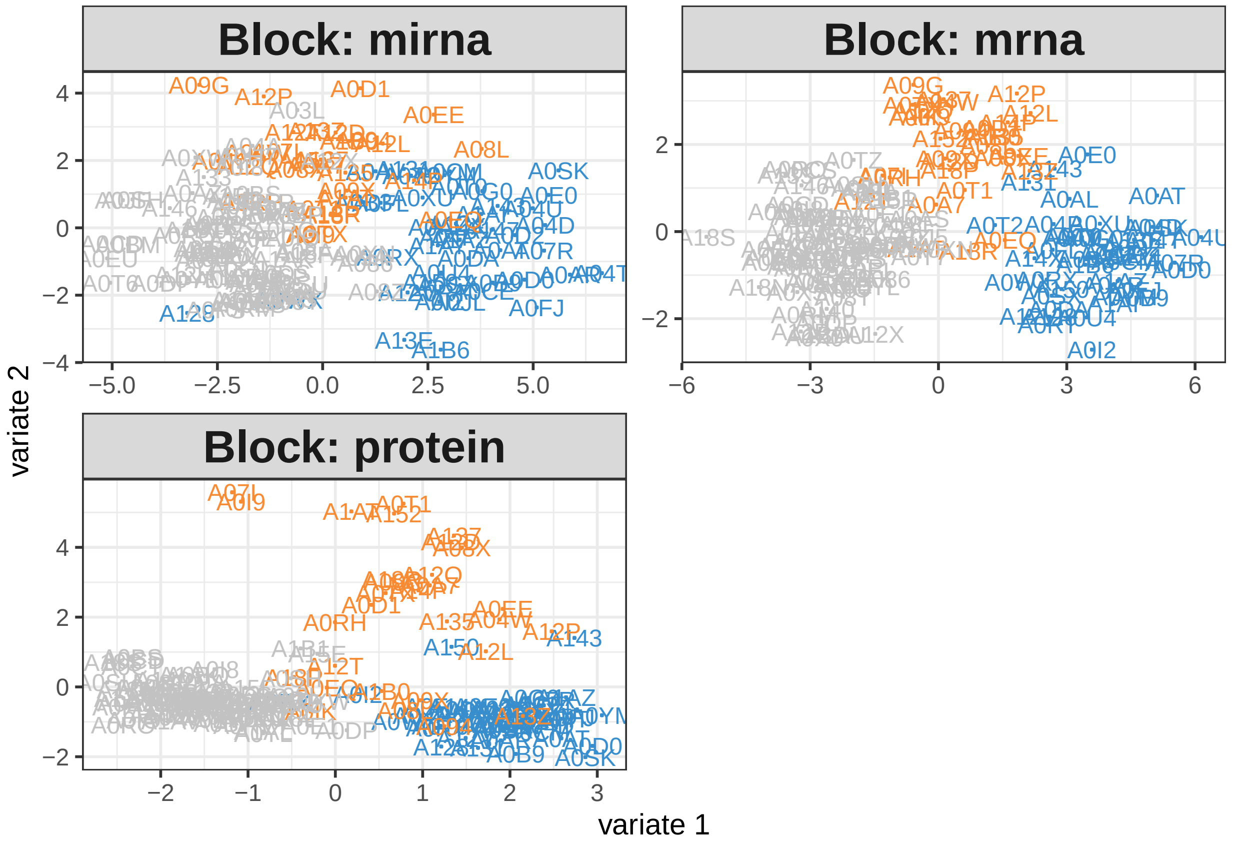
```
list.keepX = list(mirna = c(16, 17), mrna = c(18,5), protein = c(5,5))
result.sparse.diablo.breast <-  block.splsda(X_all, Y_all, keepX = list.keepX, ncomp = 5)
```

Plot the contributions of each feature to each component 1

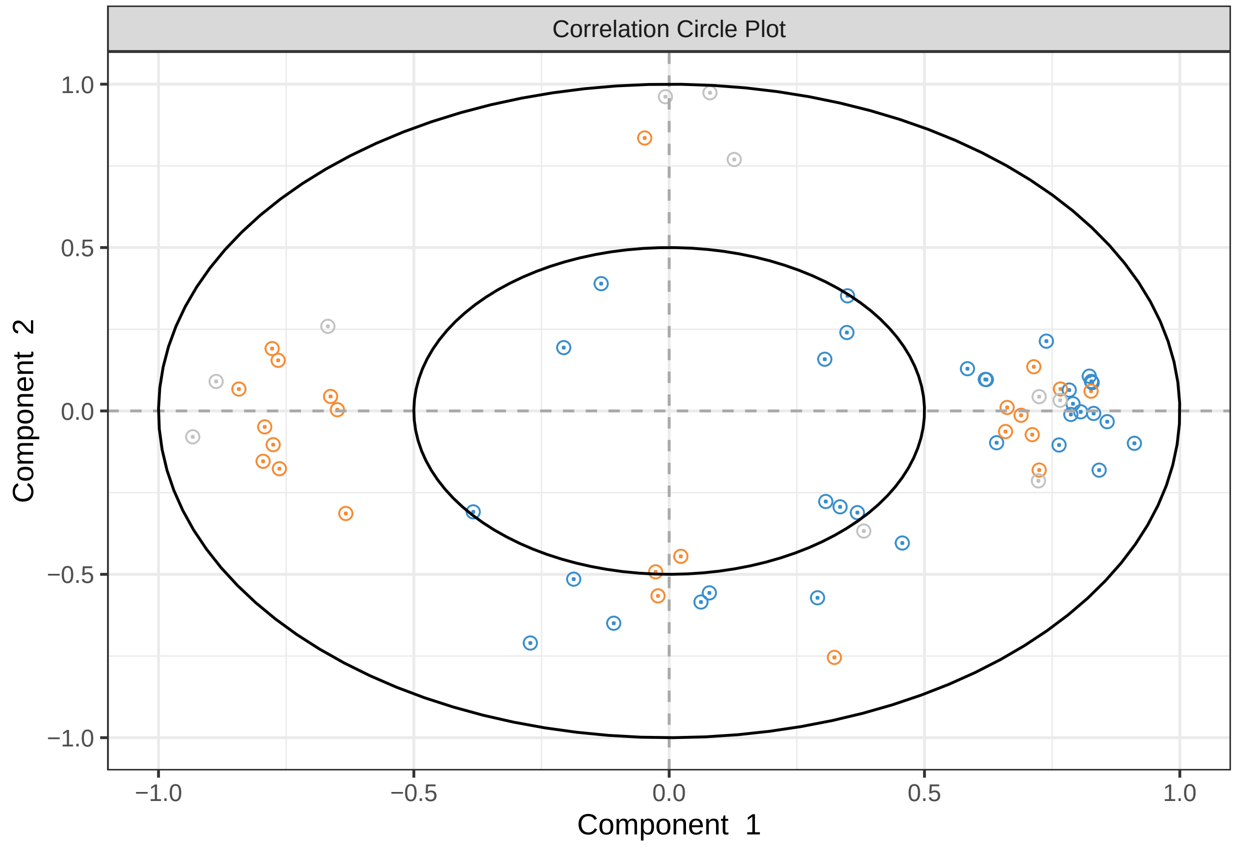
```
plotLoadings(result.sparse.diablo.breast, ncomp = 1)
```



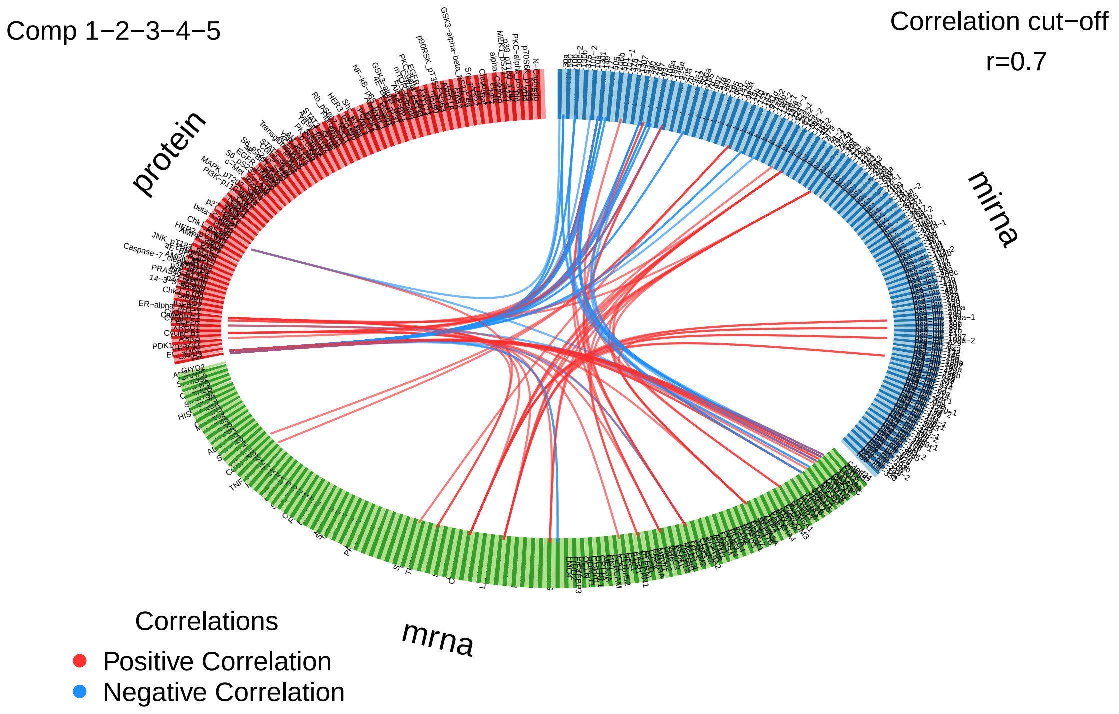
```r
plotIndiv(result.sparse.diablo.breast, var.names = FALSE) # plot the samples
```



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



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



Multiomics Integrative Predictive Modelling

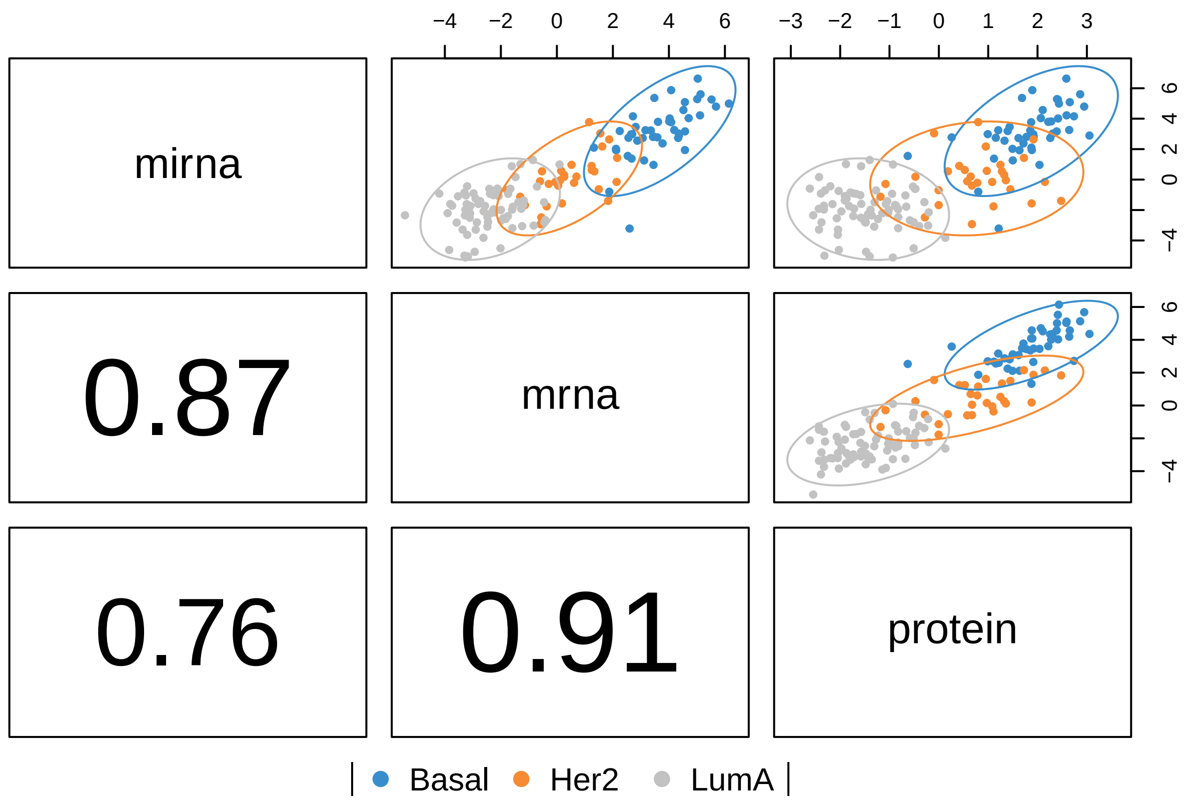
```
predict.model.diablo = predict(result.sparse.diablo.breast, list(mirna = breast.TCGA$data.test$mirna, mr
```

```
## 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)
```



```r
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