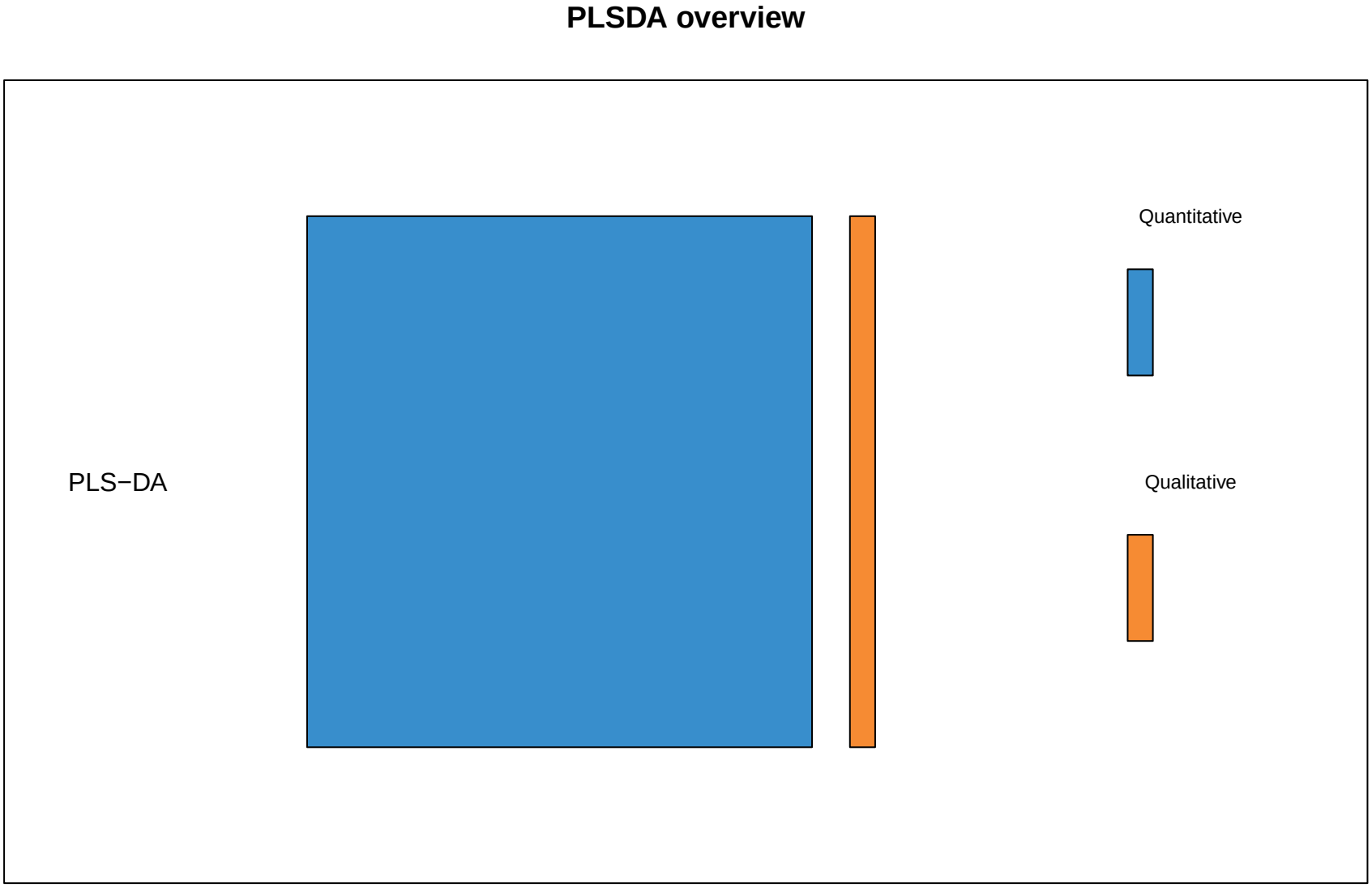
PLS - Discriminant Analysis (PLS-DA) and Sparse PLS-DA

B. Liquet

Introduction



Biological question

I am analysing a single data set (e.g. transcriptomics data) and I would like to classify my samples into known groups and predict the class of new samples. In addition, I am interested in identifying the key variables that drive such discrimination.*

The srbct study

The data are directly available in a processed and normalised format from the `mixOmics` package. The Small Round Blue Cell Tumours (SRBCT) dataset includes the expression levels of 2,308 genes measured on 63 samples.

- The samples are classified into four classes as follows: 8 Burkitt Lymphoma (BL), 23 Ewing Sarcoma (EWS), 12 neuroblastoma (NB), and 20 rhabdomyosarcoma (RMS).

The srbct dataset contains the following:

gene: a data frame with 63 rows and 2308 columns. The expression levels of 2,308 genes in 63 subjects.

class: a class vector containing the class tumour of each individual (4 classes in total).

gene.name: a data frame with 2,308 rows and 2 columns containing further information on the genes.

To illustrate PLS-DA, we will analyse the gene expression levels of `srbct$gene` to discriminate the 4 groups of tumours.

Principle of sparse PLS-DA

Although Partial Least Squares was not originally designed for classification and discrimination problems, it has often been used for that purpose.

> The response matrix ‘Y‘ is qualitative and is internally recoded as a dummy block matrix that records the membership of each observation, i.e. each of the response categories are coded via an indicator variable.

The PLS regression (now PLS-DA) is then run as if Y was a continuous matrix.

Principle of sparse PLS-DA

Sparse PLS-DA performs variable selection and classification in a one step procedure. sPLS-DA is a special case of sparse PLS described previously, where ℓ_1 penalization is applied on the loading vectors associated to the X data set.

Principle of sparse PLS-DA

```
library(mixOmics)
data(srbct)
X <- srbct$gene
Y <- srbct$class
summary(Y)
```

```
EWS  BL  NB RMS
 23   8  12  20
```

```
dim(X); length(Y)
```

```
[1]   63 2308
```

```
[1] 63
```

Quick start

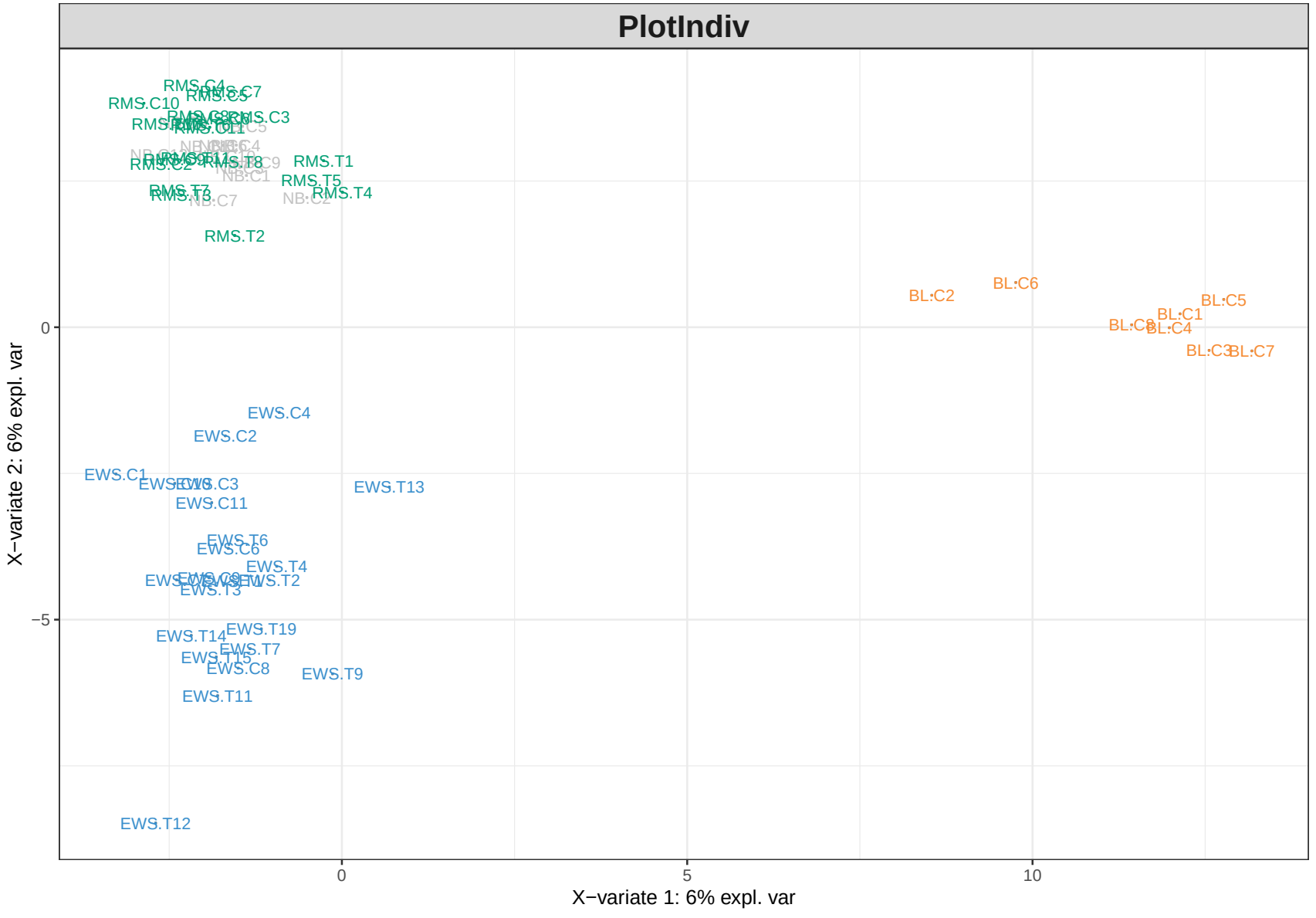
For a quick start we arbitrarily set the number of variables to select to 50 on each of the 3 components of PLS-DA.

```
# 1 Run the method
MyResult.splsda <- splsda(X, Y, keepX = c(50,50))
# 2 Plot the samples
plotIndiv(MyResult.splsda)
```

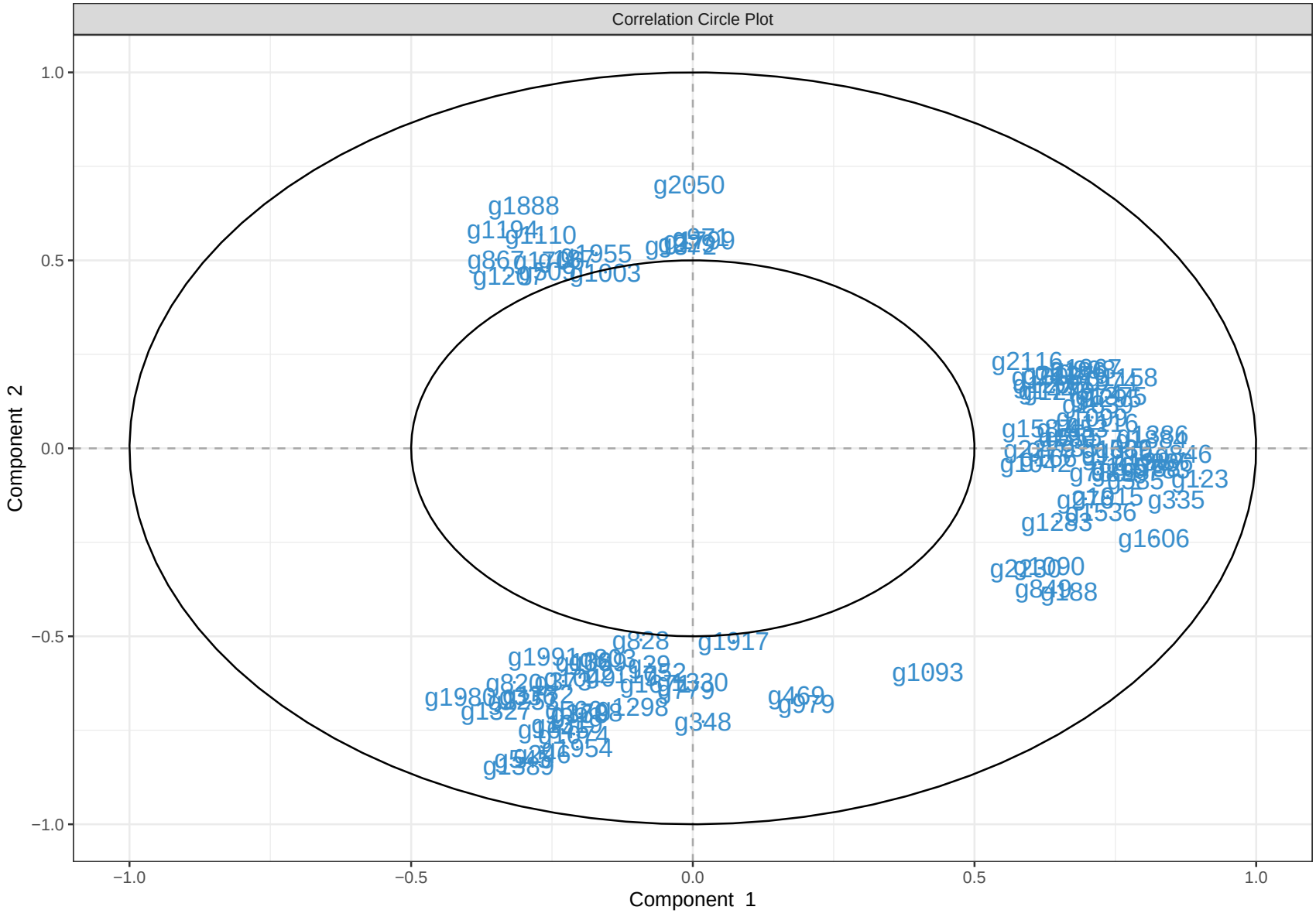
```
# 3 Plot the variables
plotVar(MyResult.splsda)
```

```
# Selected variables on component 1
selectVar(MyResult.splsda, comp=1)$name
```

Plot samples



Plot variables



Comments

- We can observe a clear discrimination between the BL samples and the others on the first component (x-axis), and EWS vs the others on the second component (y-axis).
- Remember that this discrimination spanned by the first two PLS-DA components is obtained based on a subset of 100 variables (50 selected on each component).
- From the `plotIndiv` the axis labels indicate the amount of variation explained per component.
- Note that the interpretation of this amount is *not* the same as in PCA. In PLS-DA, the aim is to maximise the covariance between `X` and `Y`, not only the variance of `X` as it is the case in PCA!

PLS-DA

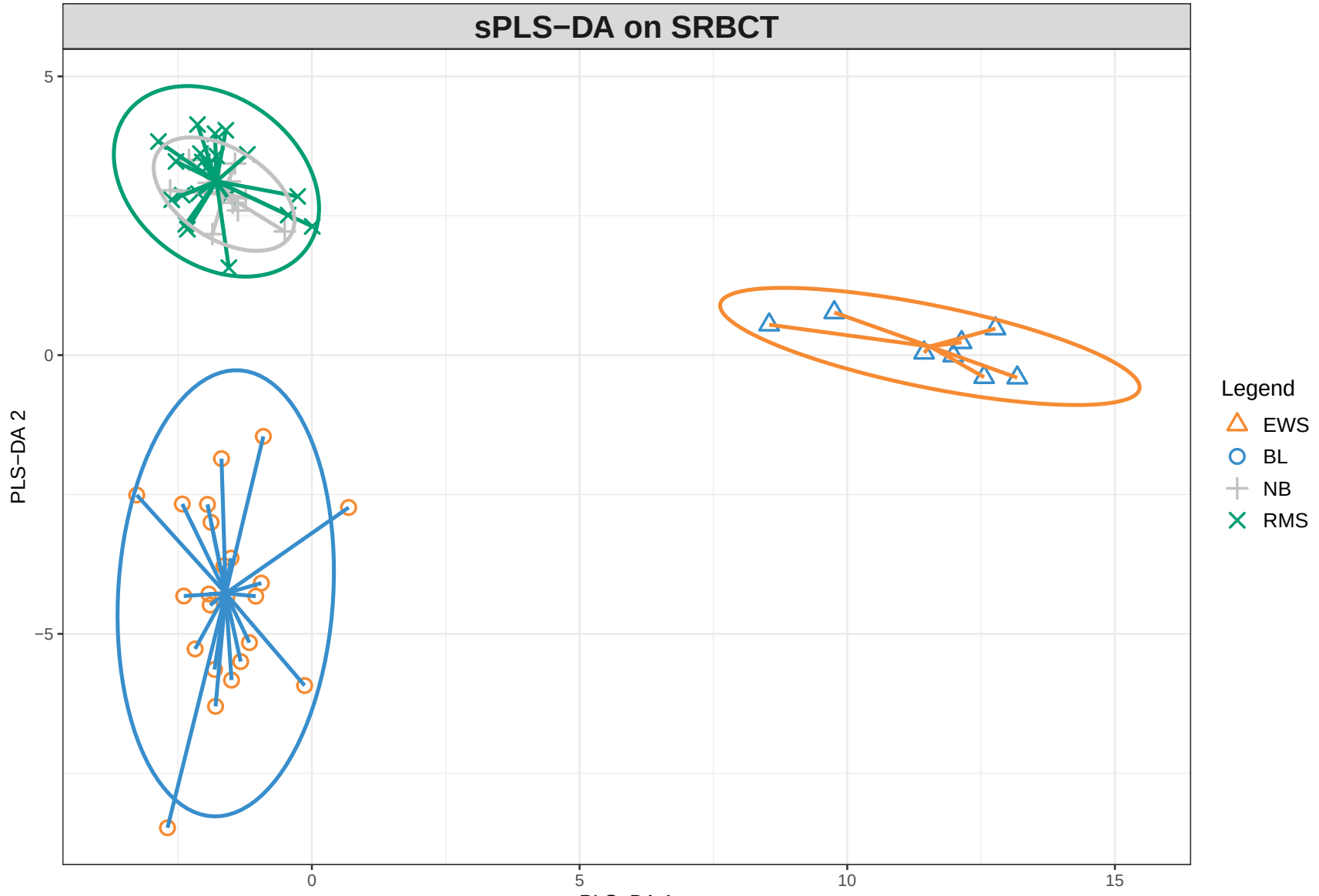
PLS-DA without variable selection can be performed as:

```
MyResult.plsda <- plsda(X,Y) # 1 Run the method
plotIndiv(MyResult.plsda)    # 2 Plot the samples
```

```
plotVar(MyResult.plsda)      # 3 Plot the variables
```

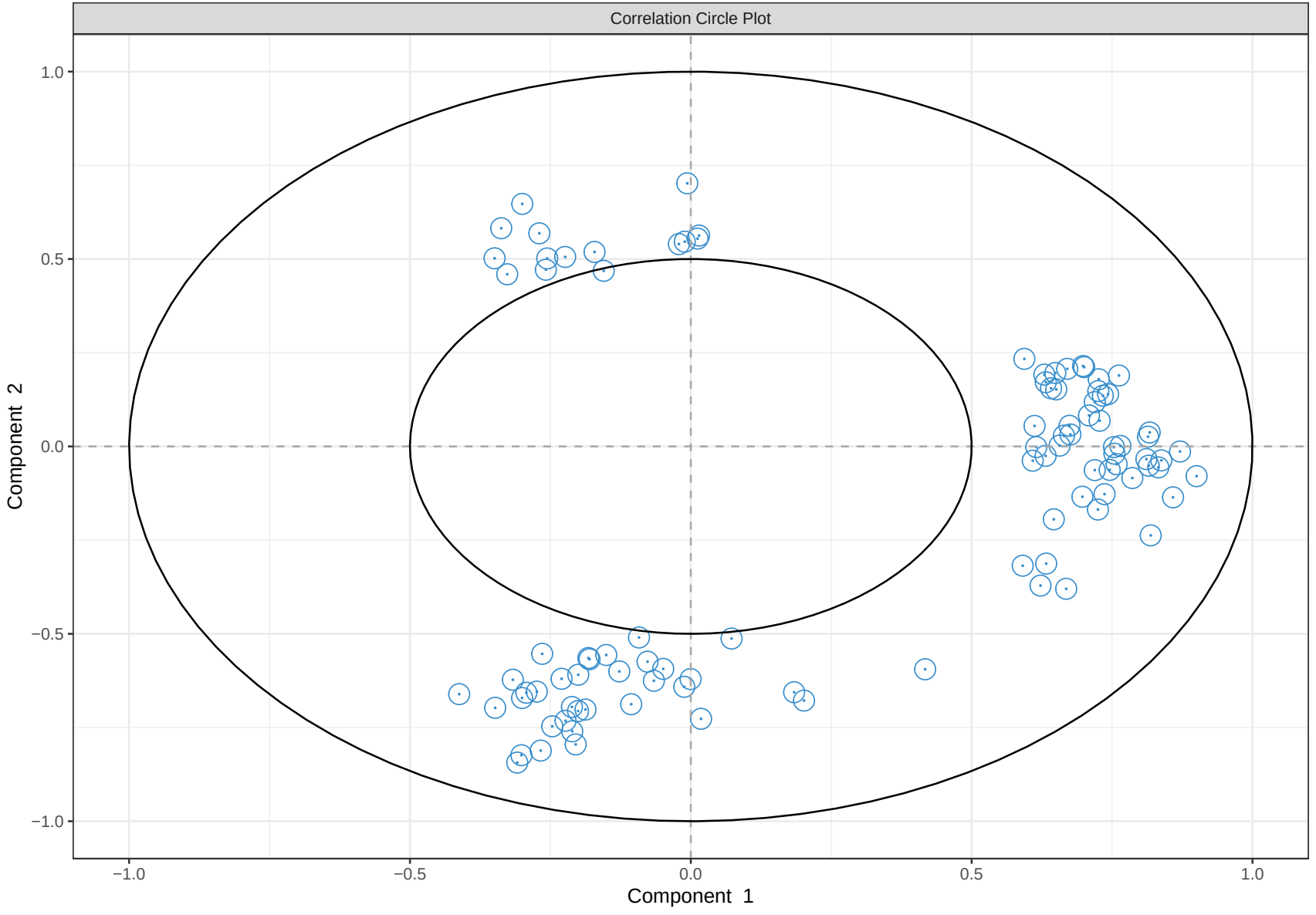
Customize sample plots

```
plotIndiv(MyResult.splsda, ind.names = FALSE, legend=TRUE,
          ellipse = TRUE, star = TRUE, title = 'sPLS-DA on SRBCT',
          X.label = 'PLS-DA 1', Y.label = 'PLS-DA 2')
```



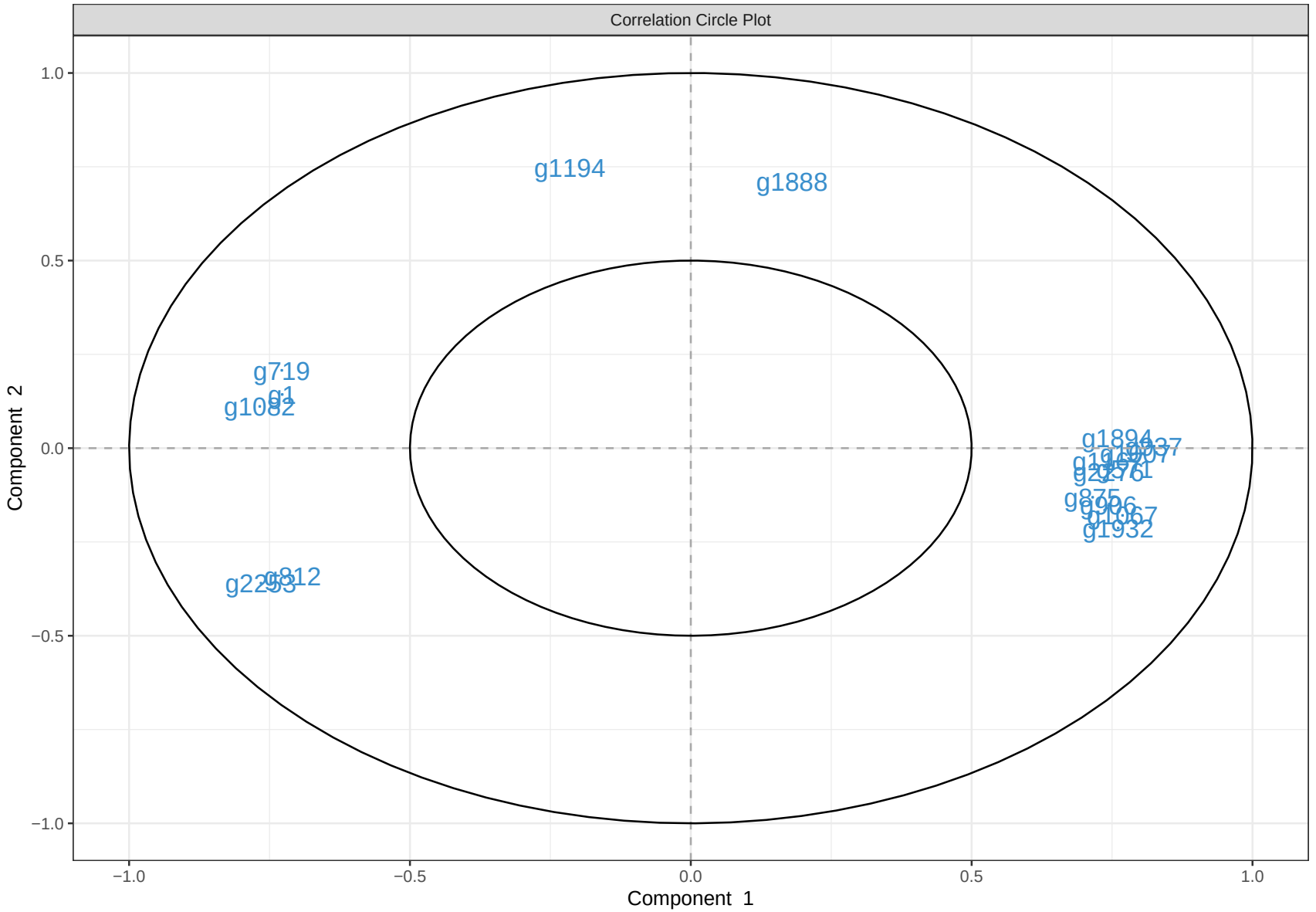
Customize variable plots

```
plotVar(MyResult.splsda, var.names=FALSE)
```



Customize variable plots

```
plotVar(MyResult.plsda, cutoff=0.7)
```



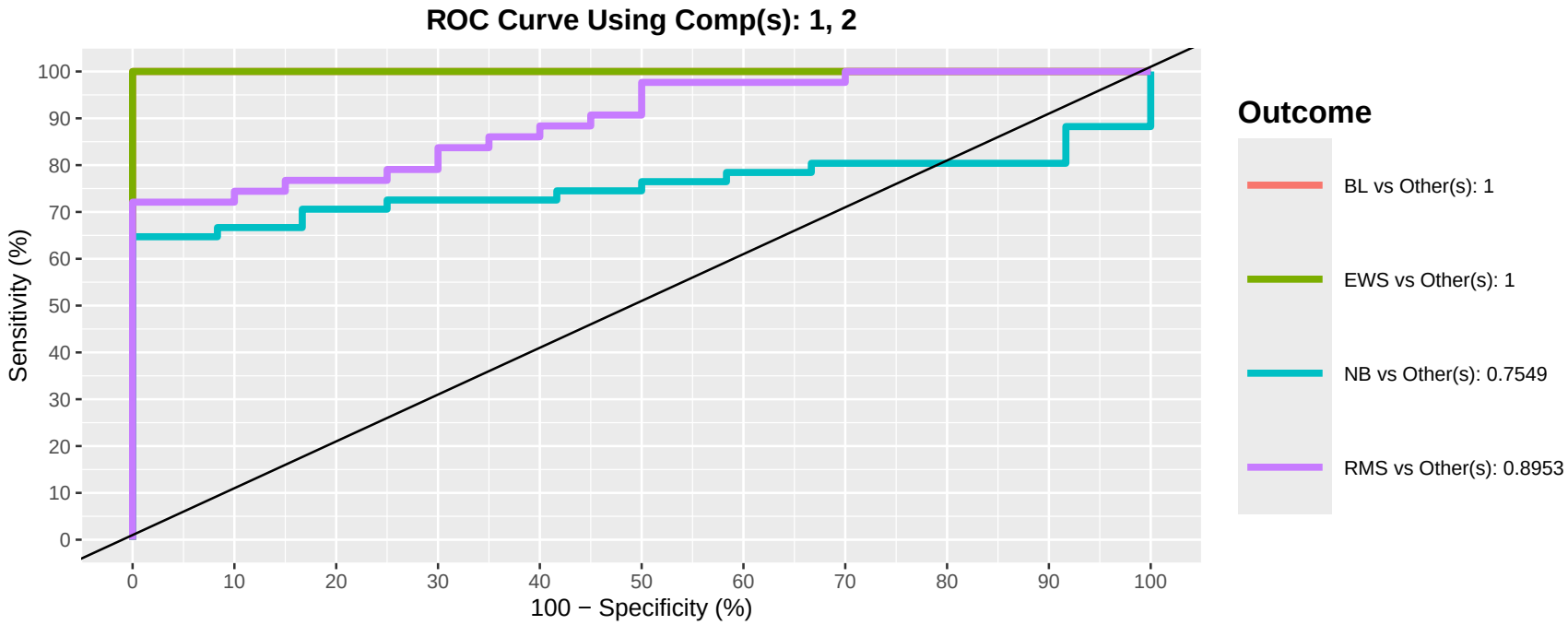
In this particular case, no variable selection was performed. Only the display was

Other useful plots

ROC

As PLS-DA acts as a classifier, we can plot a ROC Curve to complement the sPLS-DA classification performance results detailed latter. The AUC is calculated from training cross-validation sets and averaged.

```
auc.plsda <- auroc(MyResult.splsda)
```



Variable selection outputs

First, note that the number of variables to select on each component does not need to be identical on each component, for example:

```
MyResult.splsda2 <- splsda(X,Y, ncomp=3, keepX=c(15,10,5))
```

Selected variables are listed in the `selectVar` function:

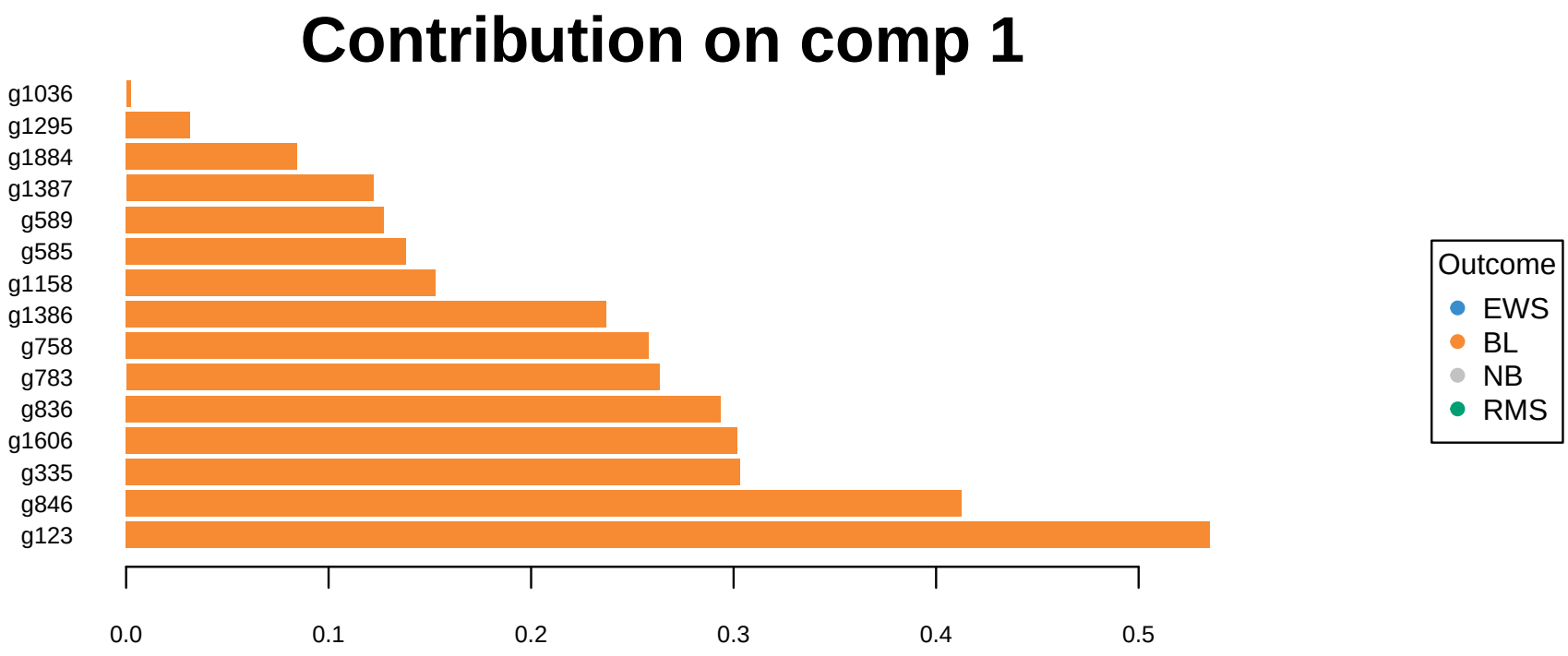
```
selectVar(MyResult.splsda2, comp=1)$value
```

```
       value.var
g123  0.53516982
g846  0.41271455
g335  0.30309695
g1606 0.30194141
g836  0.29365241
g783  0.26329876
g758  0.25826903
g1386 0.23702577
g1158 0.15283961
g585  0.13838913
g589  0.12738682
g1387 0.12202390
g1884 0.08458869
g1295 0.03150351
```

plotLoadings

Selected variables can be visualised in `plotLoadings` with the arguments `contrib = 'max'` that is going to assign to each variable bar the sample group colour for which the mean (`method = 'mean'`) is maximum. See `example(plotLoadings)` for other options (e.g. min, median)

```
plotLoadings(MyResult.splsda2, contrib = 'max', method = 'mean')
```



Comments

- Interestingly from this plot, we can see that all selected variables on component 1 are highly expressed in the BL (orange) class.
- Setting `contrib = 'min'` would highlight that those variables are lowly expressed in the NB grey class, which makes sense when we look at the sample plot.
- Since 4 classes are being discriminated here, samples plots in 3d may help interpretation:

```
plotIndiv(MyResult.splsda2, style="3d")
```

Tuning parameters and numerical outputs

For this set of methods, three parameters need to be chosen:

1 - The number of components to retain `ncomp`. The rule of thumb is usually $K - 1$ where K is the number of classes, but it is worth testing a few extra components.

2 - The number of variables `keepX` to select on each component for sparse PLS-DA,

3 - The prediction distance to evaluate the classification and prediction performance of PLS-DA.

Tuning parameters and numerical outputs

- For **item 1**, the `perf` evaluates the performance of PLS-DA for a large number of components, using repeated k-fold cross-validation.
- For example here we use 3-fold CV repeated 10 times (note that we advise to use at least 50 repeats, and choose the number of folds that are appropriate for the sample size of the data set):

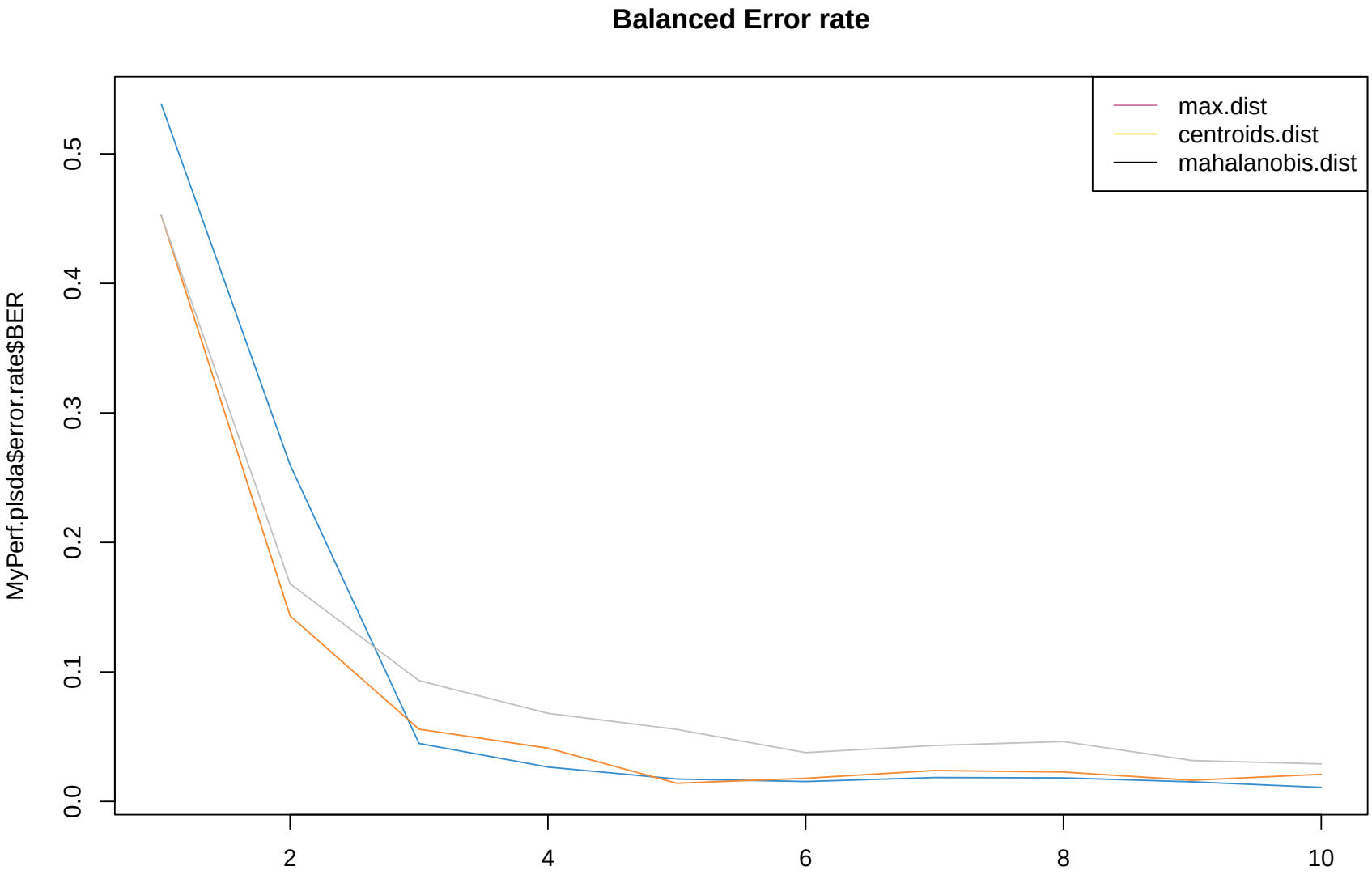
Tuning parameters and numerical outputs

```
MyResult.plsda2 <- plsda(X,Y, ncomp=10)
set.seed(30) # for reproducibility in this vignette, otherwise increase nrepeat
MyPerf.plsda <- perf(MyResult.plsda2, validation = "Mfold", folds = 3,
                  progressBar = FALSE, nrepeat = 10) # we suggest nrepeat = 50

# type attributes(MyPerf.plsda) to see the different outputs
# slight bug in the output function currently see the quick fix below

plot(MyPerf.plsda, col = color.mixo(5:7), sd = TRUE, legend.position = "horizontal",ylim=c(0,0.4))
```

Tuning parameters and numerical outputs



Comments

- The plot outputs the classification error rate, or *Balanced* classification error rate when the number of samples per group is unbalanced, the standard deviation according to three prediction distances.
- Here we can see that for the BER and the maximum distance, the best performance (i.e. low error rate) seems to be achieved for `ncomp = 3`.

Prediction performance

In addition (**item 3** for PLS-DA), the numerical outputs listed here can be reported as performance measures:

```
MyPerf.plsda
```

```
Call:
 perf.mixo_plsda(object = MyResult.plsda2, validation = "Mfold", folds

 Main numerical outputs:
 --------------------
 Error rate (overall or BER) for each component and for each distance:
 Error rate per class, for each component and for each distance: see ob
 Prediction values for each component: see object$predict
 Classification of each sample, for each component and for each distanc
 AUC values: see object$auc if auc = TRUE

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

The number of variables

- Regarding **item 2**, we now use `tune.splsda` to assess the optimal number of variables to select on each component.
- We first set up a grid of `keepX` values that will be assessed on each component, one component at a time.
- Similar to above we run 3-fold CV repeated 10 times with a maximum distance prediction defined as above.

```
list.keepX <- c(5:10,  seq(20, 100, 10))
list.keepX # to output the grid of values tested
```

```
 [1]   5   6   7   8   9  10  20  30  40  50  60  70  80  90 100
```

```
set.seed(30) # for reproducbility
tune.splsda.srbct <- tune.splsda(X, Y, ncomp = 3,
                          validation = 'Mfold',
                          folds = 3, dist = 'max.dist', progressBar = FALSE,
                          measure = "BER", test.keepX = list.keepX,
                          nrepeat = 10)   # we suggest nrepeat = 50
```

Comments

We can then extract the classification error rate averaged across all folds and repeats for each tested `keepX` value, the optimal number of components (see `?tune.splsda` for more details), the optimal number of variables to select per component which is summarised in a plot where the diamond indicated the optimal `keepX` value:

```
error <- tune.splsda.srbct$error.rate
ncomp <- tune.splsda.srbct$choice.ncomp$ncomp
# optimal number of components based on t-tests on the error rate
ncomp
```

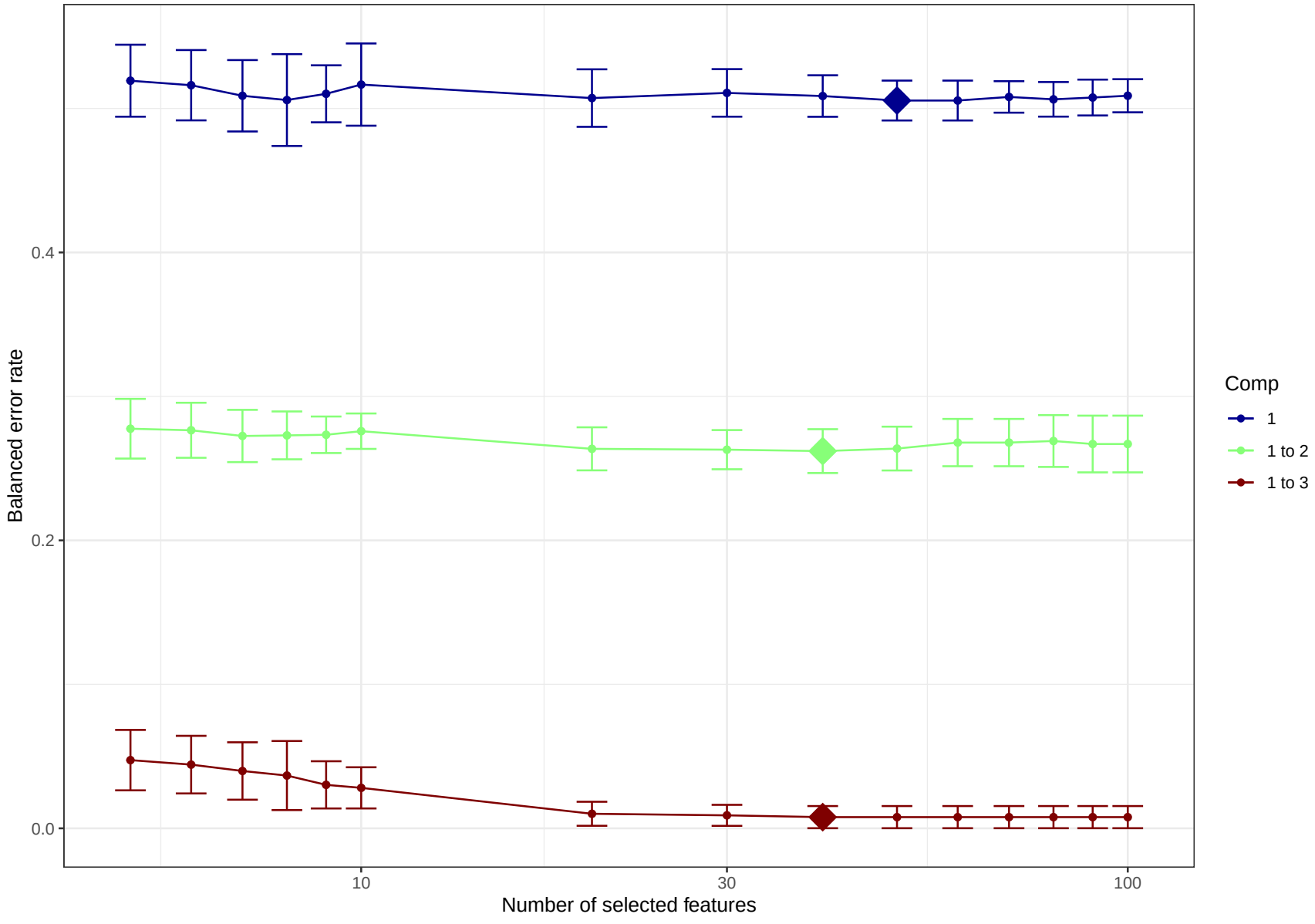
```
[1] 3
```

```
select.keepX <- tune.splsda.srbct$choice.keepX[1:ncomp]
# optimal number of variables to select
select.keepX
```

```
comp1 comp2 comp3
   50    40    40
```

Tune

```
plot(tune.splsda.srbct, col = color.jet(ncomp))
```



Final model

Based on those tuning results, we can run our final and tuned sPLS-DA model:

```
MyResult.splsda.final <- splsda(X, Y, ncomp = ncomp, keepX =
                                  select.keepX)
plotIndiv(MyResult.splsda.final, ind.names = FALSE, legend=TRUE,
          ellipse = TRUE, title="SPLS-DA, Final result")
```

Final comment

- Additionally we can run ‘perf‘ for the final performance of the sPLS-DA model. Also note that ‘perf‘ will output ‘features‘ that lists the frequency of selection of the variables across the different folds and different repeats.

- This is a useful output to assess the confidence of your final variable selection, see a more [detailed example here](http://mixomics.org/case-studies/splsda-srbct/).

Take Home Message: PLS-DA and sPLS-DA

- Dimension Reduction approach

- supervised method for qualitative response variable

- Similar as discriminant analysis

- sparse version enables us variable selection