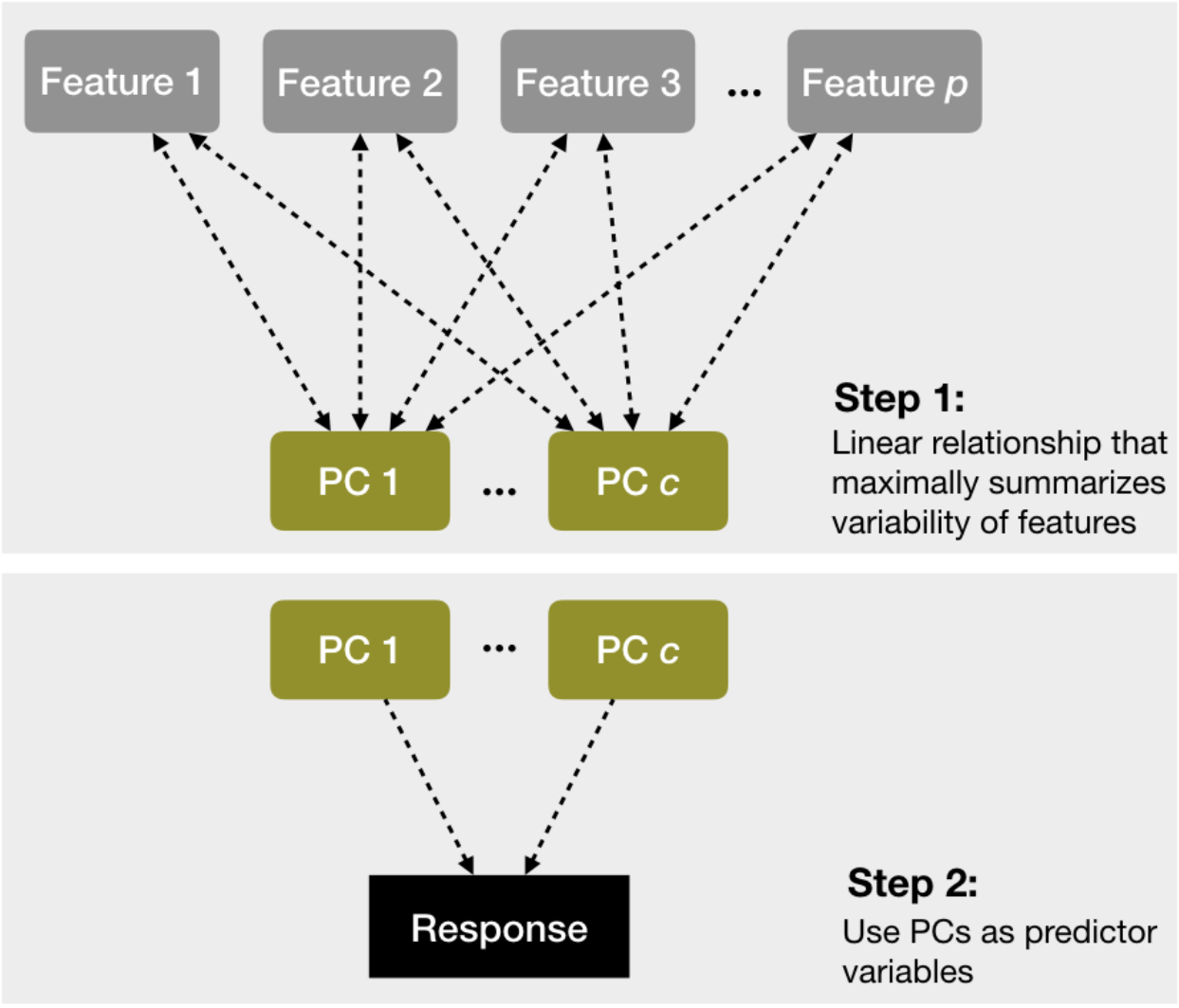
Principal Component Regression (PCR)

Outline of Lecture

- Principal Component Regression
- Partial Least Square (PLS1)
- Partial Least Square (PLS2): multivariate outcome
- Partial Least Square Discriminante Analysis: PLS-DA
- Application

Principal Component Regression



Data set: Liver toxicity

Liver toxicity study in which 64 male rats were exposed to non-toxic (50 or 150 $\mathrm{mg/kg}$), moderately toxic (1500$\mathrm{mg/kg}$) or severely toxic (2000 mg/kg) doses of acetaminophen (paracetamol)

Necropsy was performed at 6, 18, 24 and 48 hours after exposure and the mRNA was extracted from the liver.

10 **clinical measurements** of markers for liver injury are available for each subject.

The microarray data contain expression levels of 3,116 genes.

Liver toxicity contains the following:

- gene : A data frame with 64 rows (rats) and 3116 columns (gene expression levels),
- clinic : A data frame with 64 rows (same rats) and 10 columns (10 clinical variables),
- treatment : A data frame with 64 rows and 4 columns, describe the different treatments, such as doses of acetaminophen and times of necropsy.

Goal: Analyse these two data sets (genes and clinical measurements) using PCR, PLS1, then PLS2 with a regression mode to explain or predict the clinical variables with respect to the gene expression levels.

```
library(mixOmics)
data(liver.toxicity)
X <- liver.toxicity$gene
Y <- liver.toxicity$clinic
y <- liver.toxicity$clinic[, "ALB.g.dL."]
dataXY <- cbind(X,y)
```

Implement PCR with [pls package]

Similar syntax to lm formula but specify the number of PCs (ncomp)

```
library(pls)
pcr_fit <- pcr(y ~ ., ncomp = 2, scale = TRUE, data = dataXY)
summary(pcr_fit)
```

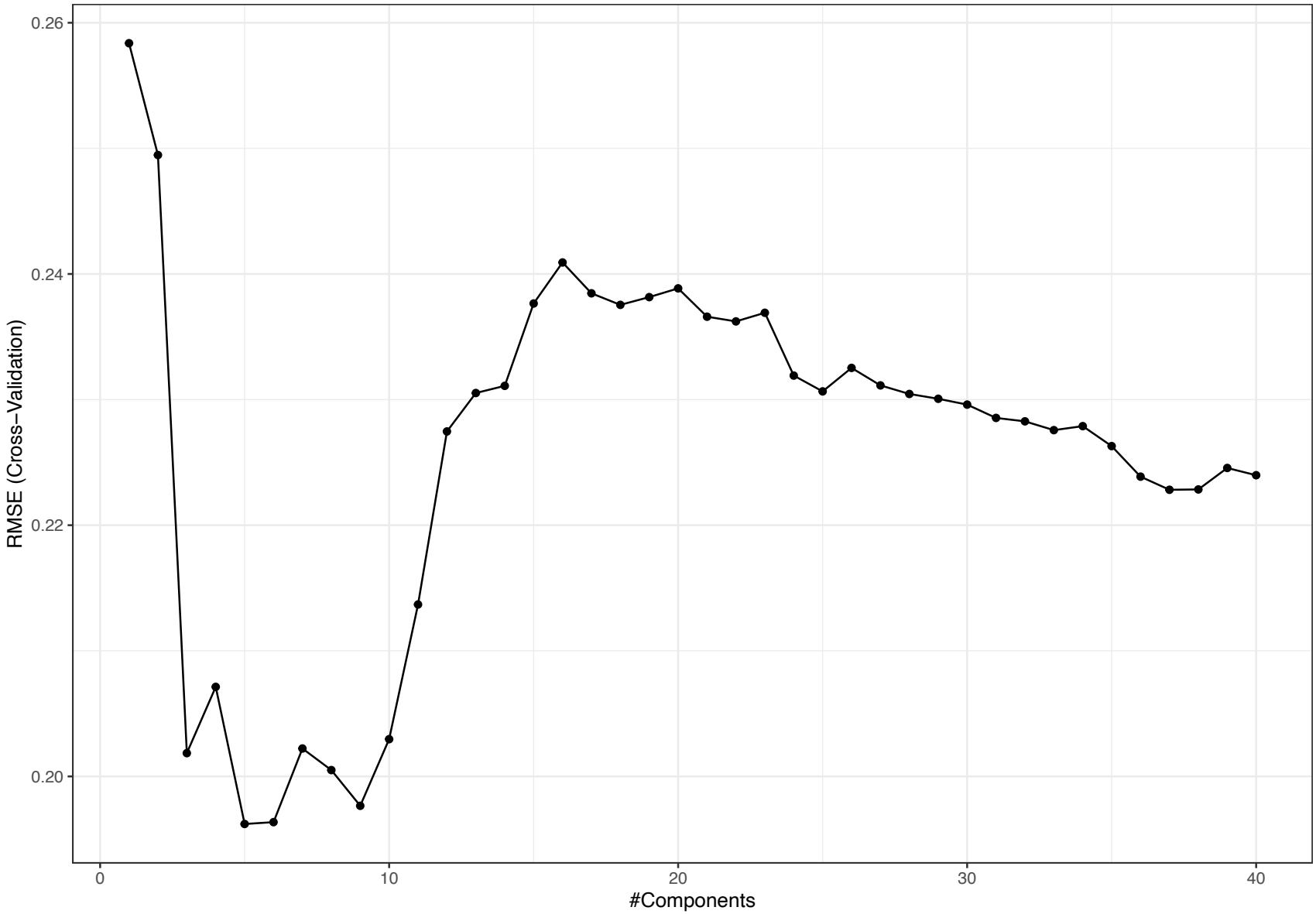
```
## Data:    X dimension: 64 3116
##  Y dimension: 64 1
## Fit method: svdpc
## Number of components considered: 2
## TRAINING: % variance explained
##    1 comps  2 comps
## X  28.07378   42.936
## y   0.07181    8.978
```

Tuning PCR with caret

To perform PCR **we need to tune the number of principal components**

- Tune # components in PCR with `caret`
- `train` with 10-fold CV using `pcr` from `pls`

```
set.seed(2013)
library(caret)
cv_model_pcr <- train(
  y ~ .,
  data = dataXY,
  method = "pcr", #<<
  trControl = trainControl(method = "cv", number = 5),
  preProcess = c("center", "scale"), #<<
  tuneLength = 40)
ggplot(cv_model_pcr) + theme_bw()
```


0.26
0.24
0.22
0.20
RMSE (Cross−Validation)
0
10
20
30
40
#Components

Tuning PCR with caret

By default returns model with minimum CV error as `finalModel`

```
summary(cv_model_pcr$finalModel)
```

```
## Data:    X dimension: 64 3116
##  Y dimension: 64 1
## Fit method: svdpc
## Number of components considered: 5
## TRAINING: % variance explained
##            1 comps  2 comps  3 comps  4 comps  5 comps
## X         28.07378   42.936    50.90    56.28    60.83
## .outcome   0.07181    8.978    41.87    46.97    49.76
```

Summary of PCR for prediction

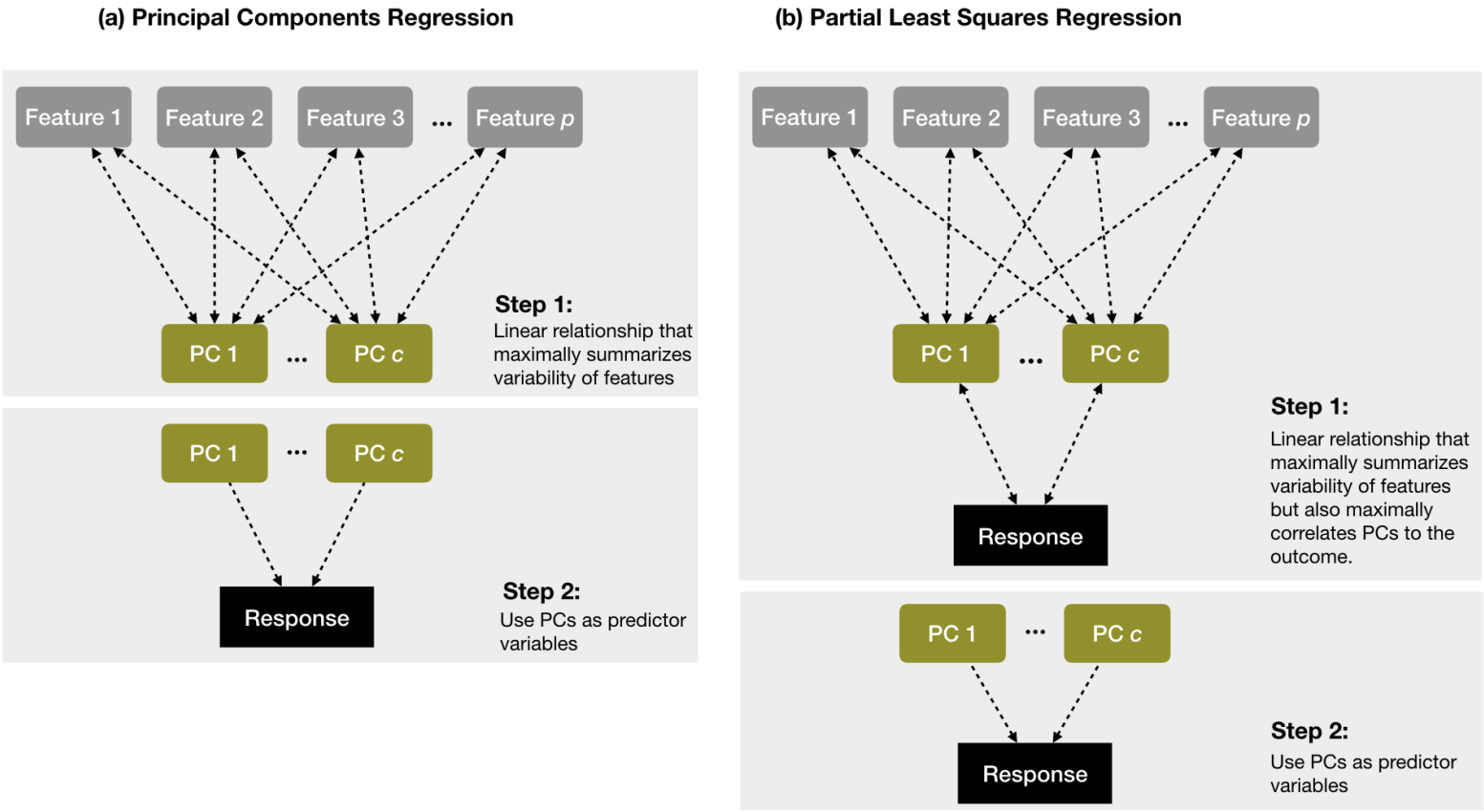
- **Step 1**: Unsupervised learning using PCA
  - create a set of uncorrelated artificial new features
  - Each new feature are a linear combination of the original feature

- **Step 2**: Supervised model
  - The artificial new features (PCs) from PCA plays the role of predictors
  - Can tune the number of PC using the response variable

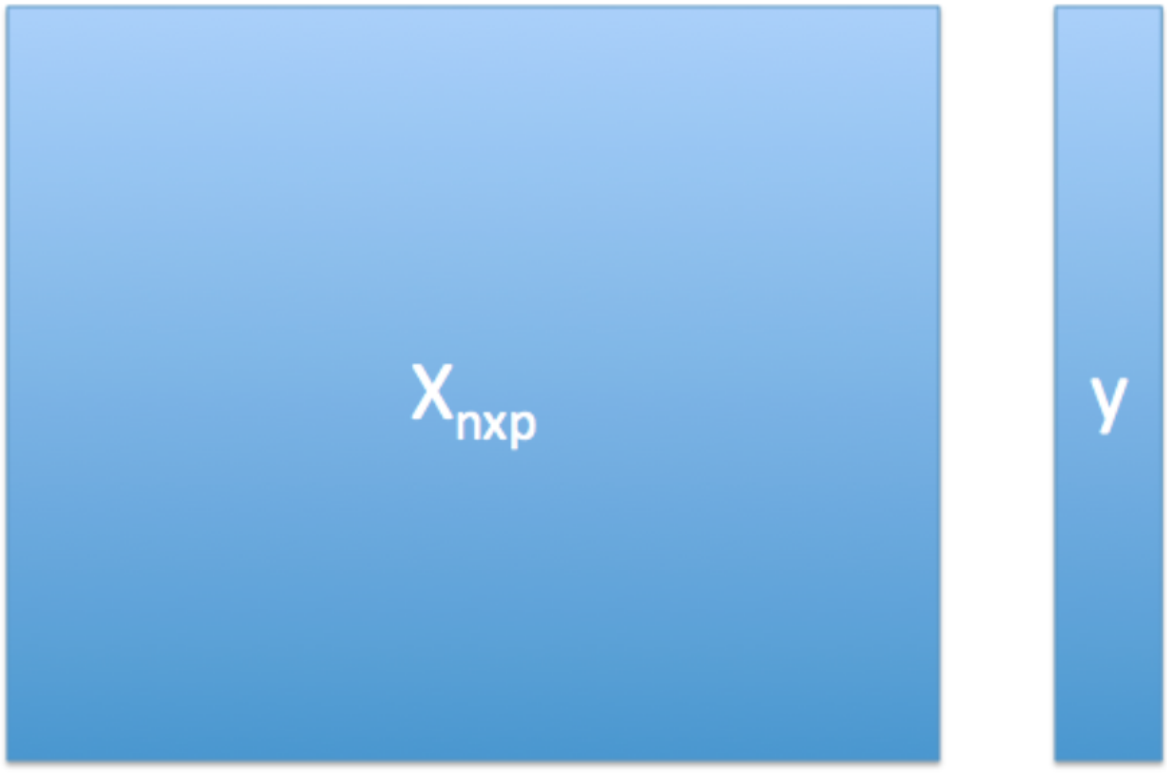
Partial least squares (PLS)

Difference between PCR and PLS

PCR is agnostic of response variable



Univariate case $Y \in \Re^n$: PLS1



PLS as supervised dimension reduction

First principal component in PCA:

$$\widetilde{X}_{(1)} = v_{11}X_{(1)} + v_{21}X_{(2)} + \cdots + v_{p1}X_{(p)}$$

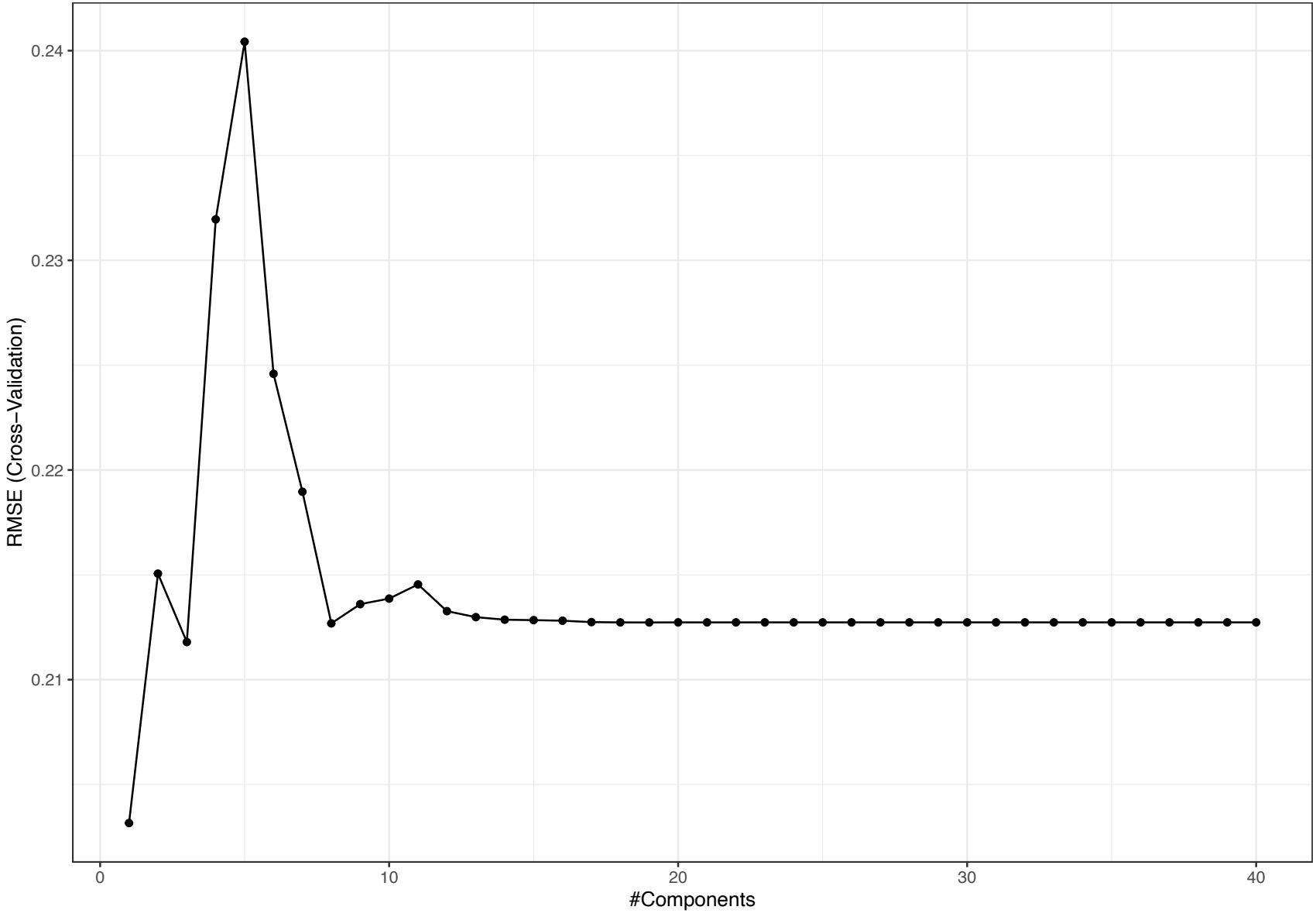
- In **PLS1** we set v_{j1} to the coefficient from **simple linear regression** of y on each $X_{(j)}$
  - Remember this slope is proportional to the correlation! $\widehat{\beta} = r_{X,Y} \cdot \frac{s_Y}{s_X} = \frac{cov(X,Y)}{s_x}$
  - Thus $\widetilde{X}_{(1)}$ in PLS places most weight on variables strongly related to response Y

To compute $\widetilde{X}_{(2)}$ for PLS:

- Regress y on $\widetilde{X}_{(1)}$, the residuals capture signal not explained by $\widetilde{X}_{(1)}$. These residuals are the new outcome y called Deflated y
- Regress each $X_{(j)}$ on $\widetilde{X}_{(1)}$, the residuals capture signal from $X_{(j)}$ not explained by $\widetilde{X}_{(1)}$ called Deflated $X_{(j)}$
- Set v_{j2} to the coefficient from **simple linear regression** of the new y Deflated y on these residuals for each variable Deflated $X_{(j)}$.
- Repeat process until all $\widetilde{X}_{(1)}, \widetilde{X}_{(2)}, \ldots, \widetilde{X}_{(p^*)}$ are computed (**PLS components**), where $p^* < p$ is a tuning parameter.

Tuning PLS with caret

```
set.seed(2013)
cv_model_pls <- train(
  y ~ .,
  data = dataXY,
  method = "pls", #<<
  trControl =
    trainControl(method = "cv", number = 10),
  preProcess = c("center", "scale"),
  tuneLength = 40)
ggplot(cv_model_pls) + theme_bw()
```



Based on this plot we would choose only one dimension

PLS formulation: Univariate case $Y \in \Re^n$

step 1: find $u \in \mathbb{R}^p$ such that $\max_{||u||=1} cov(Xu, Y)$

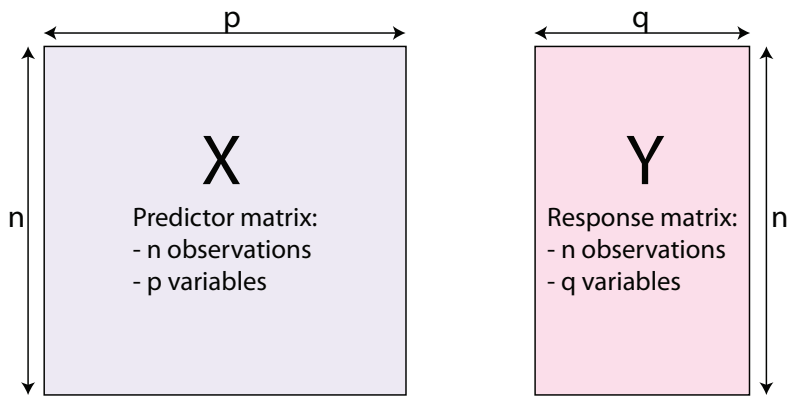
$$\begin{aligned} cov(Xu, Y) &= (Xu)^\top Y = u^\top X \\ &= < u, X^T Y >= ||X^\top Y|| cos(u, X^\top Y) \end{aligned}$$

$\hookrightarrow u = \frac{X^T Y}{||X^T Y||}$

- Thus each element of the vector u is the regression parameter **simple linear regression** of y on each $X_{(j)}$
- **Step 2**: find a new linear combination no correlated to $\xi = Xu$ which explain the residuals $Y - d\xi$ where d is the regression of Y on $\xi = Xu$
- **Deflated step**: $Y \leftarrow Y - d\xi$ and $X \leftarrow X - \xi c^\top$ where c is the regression of X on $\xi = Xu$;
- The columns of the new X are orthogonal (no correlated) to Xu.

PLS: General case

PLS is a family of multivariate statistical techniques based on dimension reduction developed by S. Wold and H. Wold (1966, 1983). It can be seen as a supervised version of PCA.



Modelling Aims

Partial Least Square is also a **Dimension reduction method** with the two following aims:

- Symmetric relationship: analyse the shared information.
- Asymmetric relationship: X = predictors, and Y = response.

Modelling Aims

When Y is only one column (univariate case), PLS can be summarized as

- Dimension reduction method: p dimension space $\Rightarrow$ H dimension space ($H << p$)
- PLS looks the best components the most correlated to the response variable
- The PLS components are linear combinations of the variables

$$\widetilde{X}_{(k)} = u_{1,k} \times X_{(1)} + u_{2,k} \times X_{(2)} + \ldots + u_{p,k} \times X_{(p)}$$

- It is a supervised approach

Modelling Aims

In more general case (Y multivariate $q > 1$):

- PLS finds pairs of latent (score) vectors $\xi_{(k)} = Xu_{(k)}$, $\omega_{(k)} = Yv_{(k)}$

$$\xi_{(k)} = u_{1,k} \times \text{gene}_{(1)} + u_{2,k} \times \text{gene}_{(2)} + \cdots + u_{p,k} \times \text{gene}_{(p)}$$
$$\omega_{(k)} = v_{1,k} \times \text{pheno}_{(1)} + v_{2,k} \times \text{pheno}_{(2)} + \cdots + v_{p,k} \times \text{pheno}_{(p)}$$

- Symmetric relationship. Analyse the shared information.
- Asymmetric relationship. There is a set of response and predictor variables that can be used for prediction.

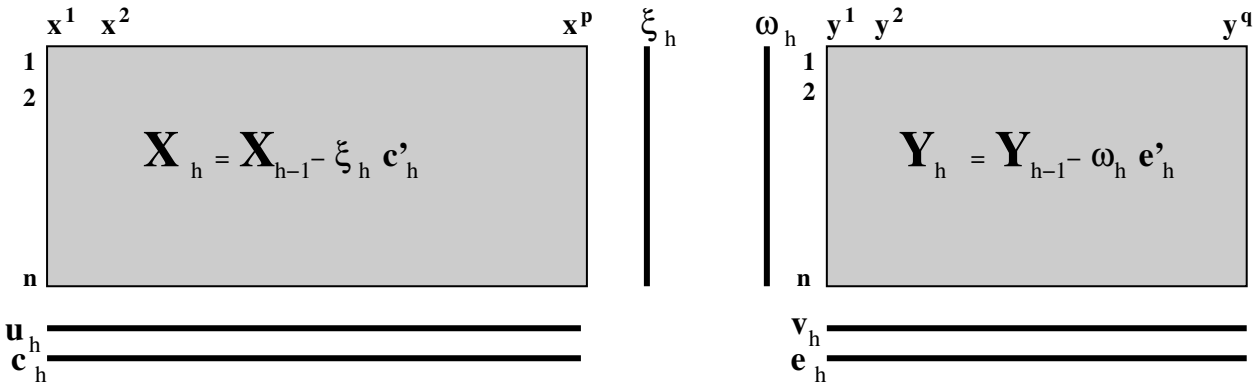
Objective function:

$$\max_{\|u_{(h)}\|=1, \|v_{(h)}\|=1} cov(X_h u_{(h)}, Y_h v_{(h)}) \qquad h = 1 \ldots H$$

Principle:

- Iterative procedure $\mapsto$ orthogonal component (latent variable $\xi_{(h)} = X_h u_{(h)}$).
- successive local regressions on the latent variables.
- X and Y are successively deflated.

PLS: regression mode (multivariate case: Y $(n \times q)$)



For each iteration h, $h = 1..H$, decompose X and Y into:

1. Loadings vectors $u_{(h)}$ and $v_{(h)}$, p- and q- dimensional vectors
2. Latent variables $\xi_{(h)}$ and $\omega_{(h)}$, n-dimensional vectors
3. Regression of X_{h-1} and Y_{h-1} on $\xi_{(h)}$ and $\omega_{(h)}$, reg. coeff. $c_{(h)}$ and $e_{(h)}$
4. Residual matrices: deflation step of X_{h-1} and Y_{h-1}

Algorithm: regression mode

Objective function:

$$\max_{\|u_{(h)}\|=1, \|v_{(h)}\|=1} cov(X_h u_{(h)}, Y_h v_{(h)}) \qquad h = 1 \ldots H$$

Start: set w to the first column of Y

1. $u = \frac{X^T w}{w^T w}$, scale u to one. u is the loading vector associated to X
2. $\xi = Xu$ is the latent variable associated to X
3. $v = \frac{Y^T \xi}{\xi^T \xi}$, scale v to one. v is the loading vector associated to Y
4. $w = Yv$ is the latent variable associated to Y.
5. If convergence then 6 else 1
6. $c = \frac{X^T \xi}{\xi^T \xi}$, $e = \frac{Y^T \xi}{\xi^T \xi}$ are the partial regression coefficients from the regression of X (Y) onto ξ.
7. Deflation step: Compute the residual matrices $X \leftarrow X - \xi c^T$ and $Y \leftarrow Y - \xi e^T$

PLS family

PLS = Partial Least Squares or Projection to Latent Structures $ $\ Four main methods coexist in the literature:

(i) Partial Least Squares Correlation (PLSC) also called PLS-SVD;

(ii) PLS in mode A (PLS-W2A, for Wold's Two-Block, Mode A PLS);

(iii) PLS in mode B (PLS-W2B) also called Canonical Correlation Analysis (CCA);

(iv) Partial Least Squares Regression (PLSR, or PLS2).

- (i),(ii) and (iii) are symmetric while (iv) is asymmetric.
- Different objective functions to optimise.
- Good news: all use the singular value decomposition (SVD).

PLS connected to Singular Value Decomposition (SVD)

Let a matrix $M : p \times q$ of rank r:

$$M = U\Delta V^T = \sum_{l=1}^{r} \delta_l u_{(l)} v_{(l)}^T,$$

- $U = (u_{(l)}) : p \times p$ and $V = (v_{(l)}) : q \times q$ are two orthogonal matrices which contain the normalised left (resp. right) singular vectors
- $\Delta = \mathrm{diag}(\delta_1, \ldots, \delta_r, 0, \ldots, 0)$: the ordered singular values $\delta_1 \geqslant \delta_2 \geqslant \cdots \geqslant \delta_r > 0$.

Connexion between SVD and maximum covariance

We were able to describe the optimization problem of the **four** PLS methods as:

$$(u^*, v^*) = \underset{\|u\|_2=\|v\|_2=1}{\operatorname{argmax}} \; Cov(X_{h-1}u, Y_{h-1}v), \qquad h = 1, \ldots, H$$

Matrices X_h and Y_h are obtained recursively from X_{h-1} and Y_{h-1}.

The four methods differ by the deflation process, chosen so that the above scores or weight vectors satisfy given constraints.

The solution at step h is obtained by computing **only the first** triplet $(\delta_1, u_{(1)}, v_{(1)}$ of singular elements of the SVD of $M_{h-1} = X_{h-1}^T Y_{h-1}$:

$$(u^*, v^*) = (u_{(1)}, v_{(1)})$$

PLS in practice: the `nutrimouse` study

The ‘nutrimouse‘ study contains the expression levels of genes potentially involved in nutritional problems and the concentrations of hepatic fatty acids for forty mice. The data sets come from a nutrigenomic study in the mouse, in which the effects of five regimens with contrasted fatty acid compositions on liver lipids and hepatic gene expression in mice were considered.

PLS in practice: the `nutrimouse` study

Two sets of variables were measured on 40 mice:

- `gene`: the expression levels of 120 genes measured in liver cells, selected among (among about 30,000) as potentially relevant in the context of the nutrition study.
- `lipid`: concentration (in percentage) of 21 hepatic fatty acids measured by gas chromatography.
- `diet`: a 5-level factor. Oils used for experimental diets preparation were corn and colza oils (50/50) for a reference diet (REF), hydrogenated coconut oil for a saturated fatty acid diet (COC), sunflower oil for an Omega6 fatty acid-rich diet (SUN), linseed oil for an Omega3-rich diet (LIN) and corn/colza/enriched fish oils for the FISH diet (43/43/14).
- `genotype` 2-levels factor indicating either wild-type (WT) and PPARα -/- (PPAR).

PLS in practice: the `nutrimouse` study

To illustrate PLS, we will integrate the gene expression levels (`gene`) with the concentrations of hepatic fatty acids (`lipid`).

Set up the data

We first set up the data as `X` expression matrix and `Y` as the lipid abundance matrix. We also check that the dimensions are correct and match:

```
library(mixOmics)
data(nutrimouse)
X <- nutrimouse$gene
Y <- nutrimouse$lipid
dim(X); dim(Y)
```

```
## [1]  40 120
```

```
## [1] 40 21
```

Quick start

```
MyResult.pls <- pls(X,Y, ncomp=10)
plotIndiv(MyResult.pls)
```

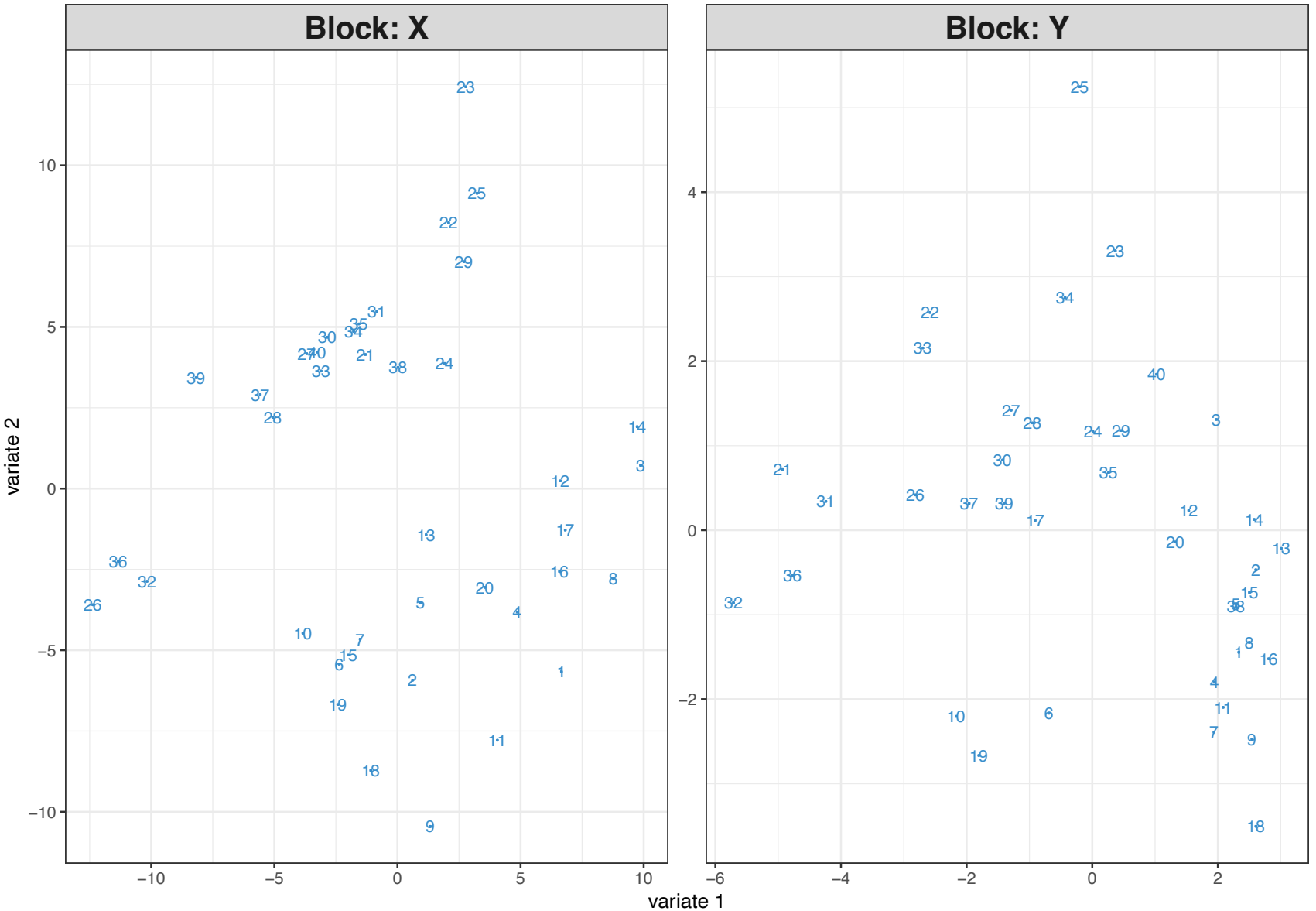
```
plotVar(MyResult.pls)
```

If you were to run `pls` with minimal code, you would be using the following default values:

- `ncomp = 2`: the first two PLS components are calculated and are used for graphical outputs;
- `scale = TRUE`: data are scaled (variance = 1, strongly advised here);
- `mode = "regression"`: by default a PLS regression mode should be used.

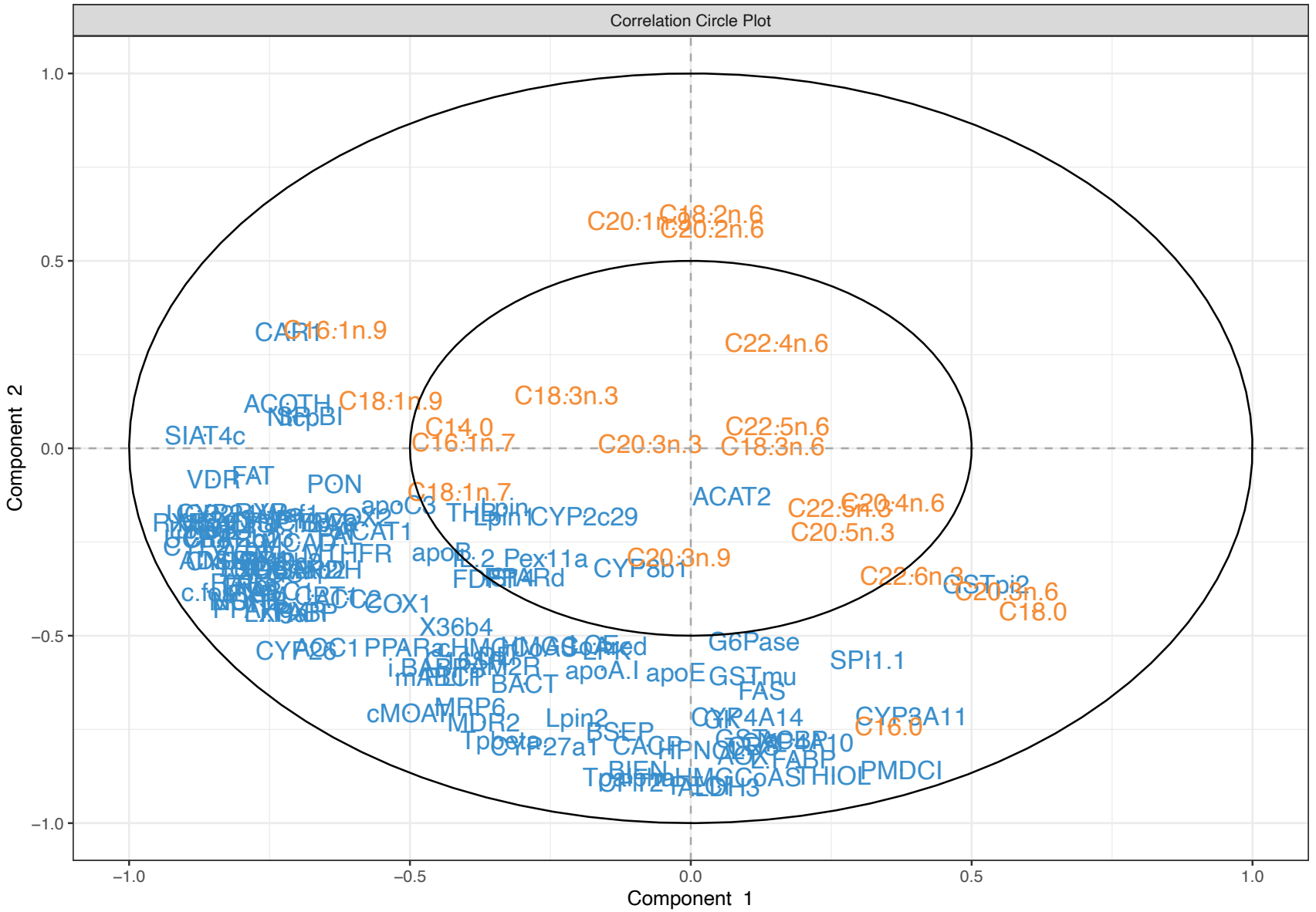
Plot the samples

```
plotIndiv(MyResult.pls)
```



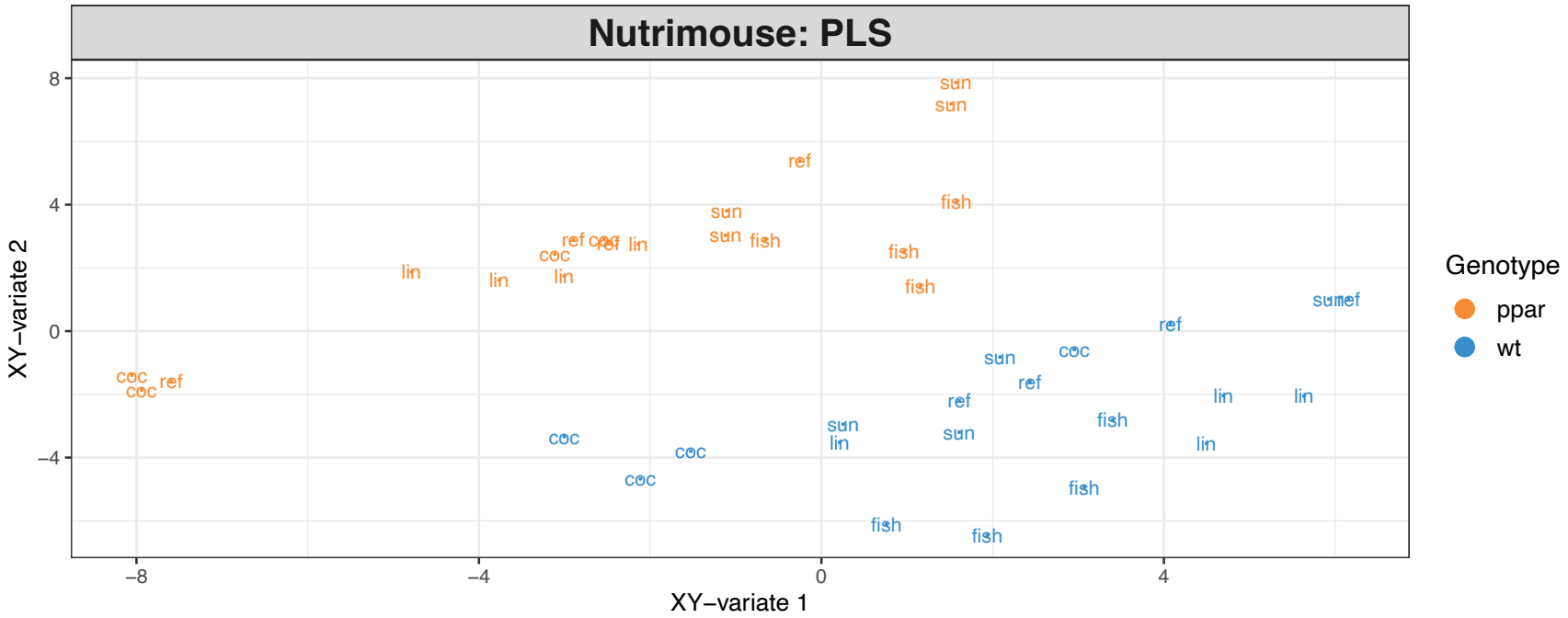
Plot the variables

```
plotVar(MyResult.pls)
```



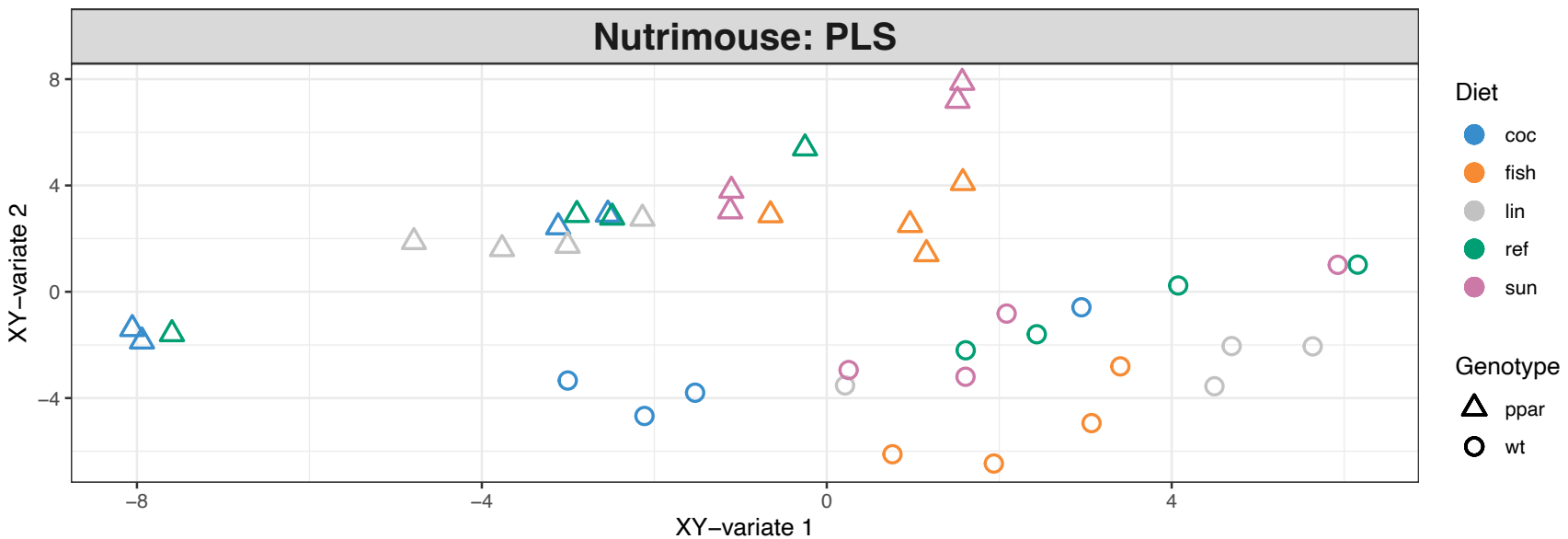
Customize sample plots

```
plotIndiv(MyResult.pls, group = nutrimouse$genotype,
          rep.space = "XY-variate", legend = TRUE,
          legend.title = 'Genotype',
          ind.names = nutrimouse$diet,
          title = 'Nutrimouse: PLS')
```



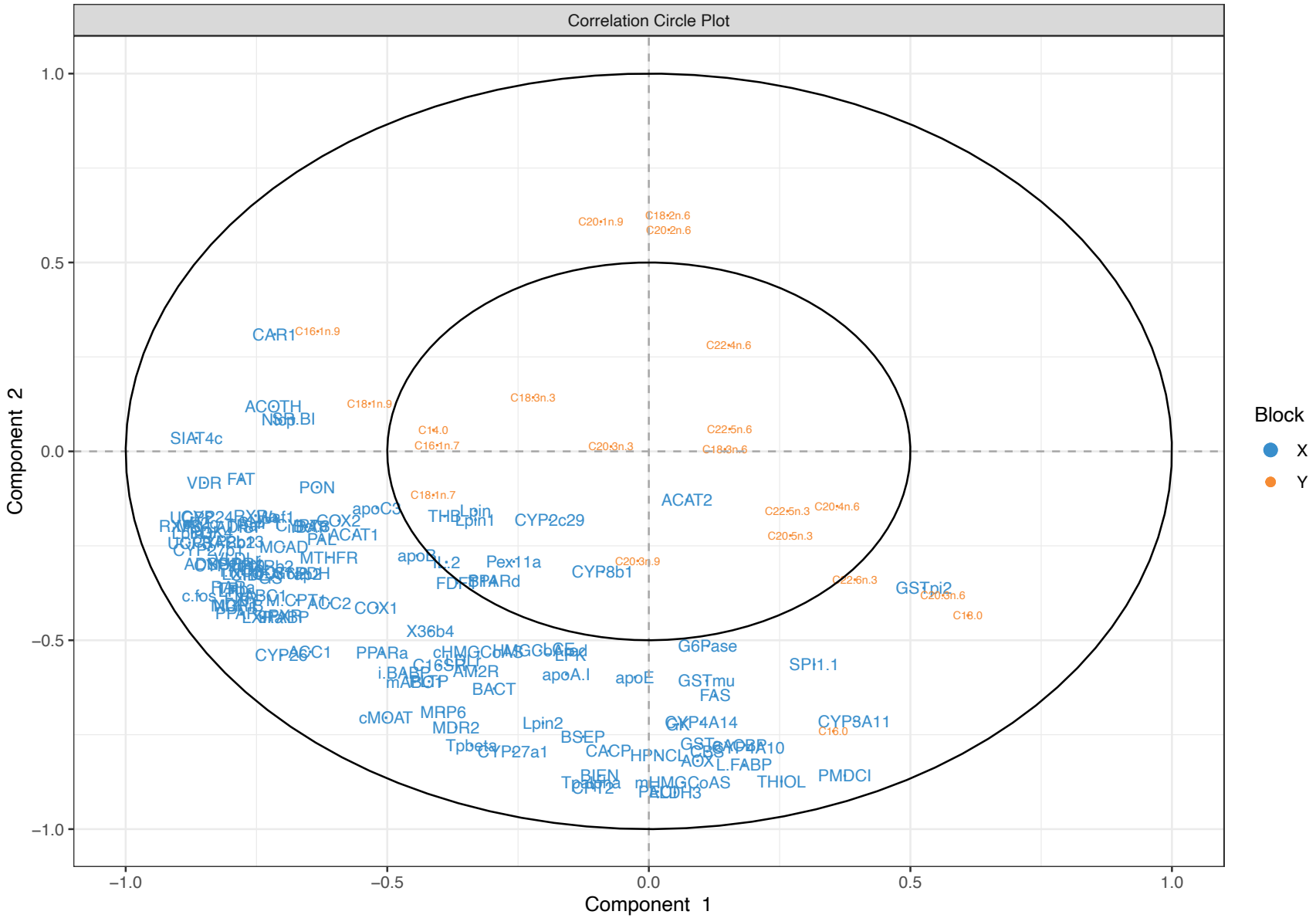
Customize sample plots

```
plotIndiv(MyResult.pls, group=nutrimouse$diet,
          pch = nutrimouse$genotype,
          rep.space = "XY-variate",  legend = TRUE,
          legend.title = 'Diet', legend.title.pch = 'Genotype',
          ind.names = FALSE,
          title = 'Nutrimouse: PLS')
```



Customize variable plots

```
plotVar(MyResult.pls, cex=c(3,2), legend = TRUE)
```



Customize variable plots

In this example, the figure is difficult to interpret and we would prefer to use a sparse vesrion of PLS to selecet the most important variable.

A cut-off can be set to display only the variables that mostly contribute to the definition of each component. Those variables should be located towards the circle of radius 1, far from the centre.

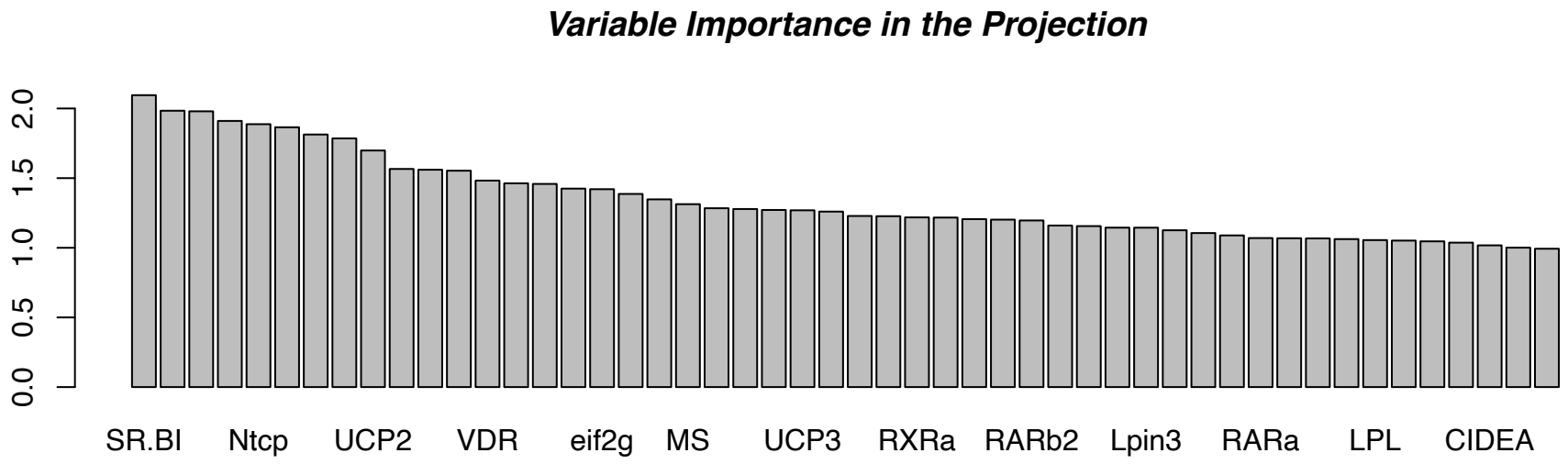
```
plotVar(MyResult.pls, cutoff=0.5)
```

In this particular case, no variable selection was performed. Only the display was altered to show a subset of variables.

Variable Importance in the Projection (VIP)

Variable importance in projection (VIP) coefficients reflect the relative importance of each X variable for each X variate in the prediction model.

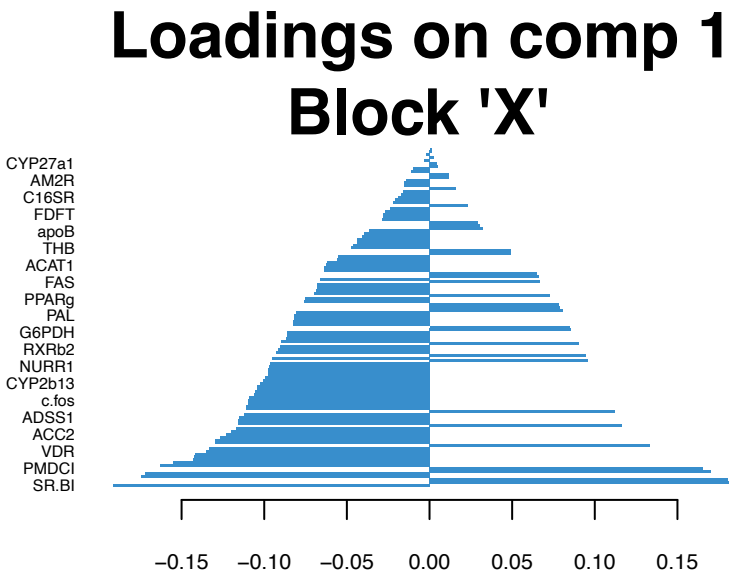
```
my.vip <- sort(vip(MyResult.pls)[,1],decreasing = TRUE)
barplot(my.vip[1:50],
beside = FALSE,
ylim = c(0, max(my.vip)), legend = rownames(my.vip)[1:50],
main = "Variable Importance in the Projection", font.main = 4)
```

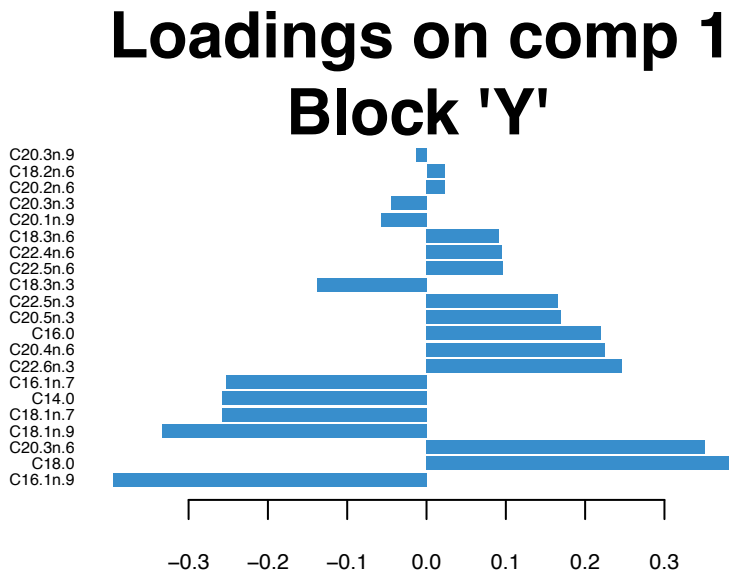


Loading plots

The loading plots help visualise the coefficients assigned to each selected variable on each component:

```
plotLoadings(MyResult.pls, comp = 1, size.name = rel(0.5))
```





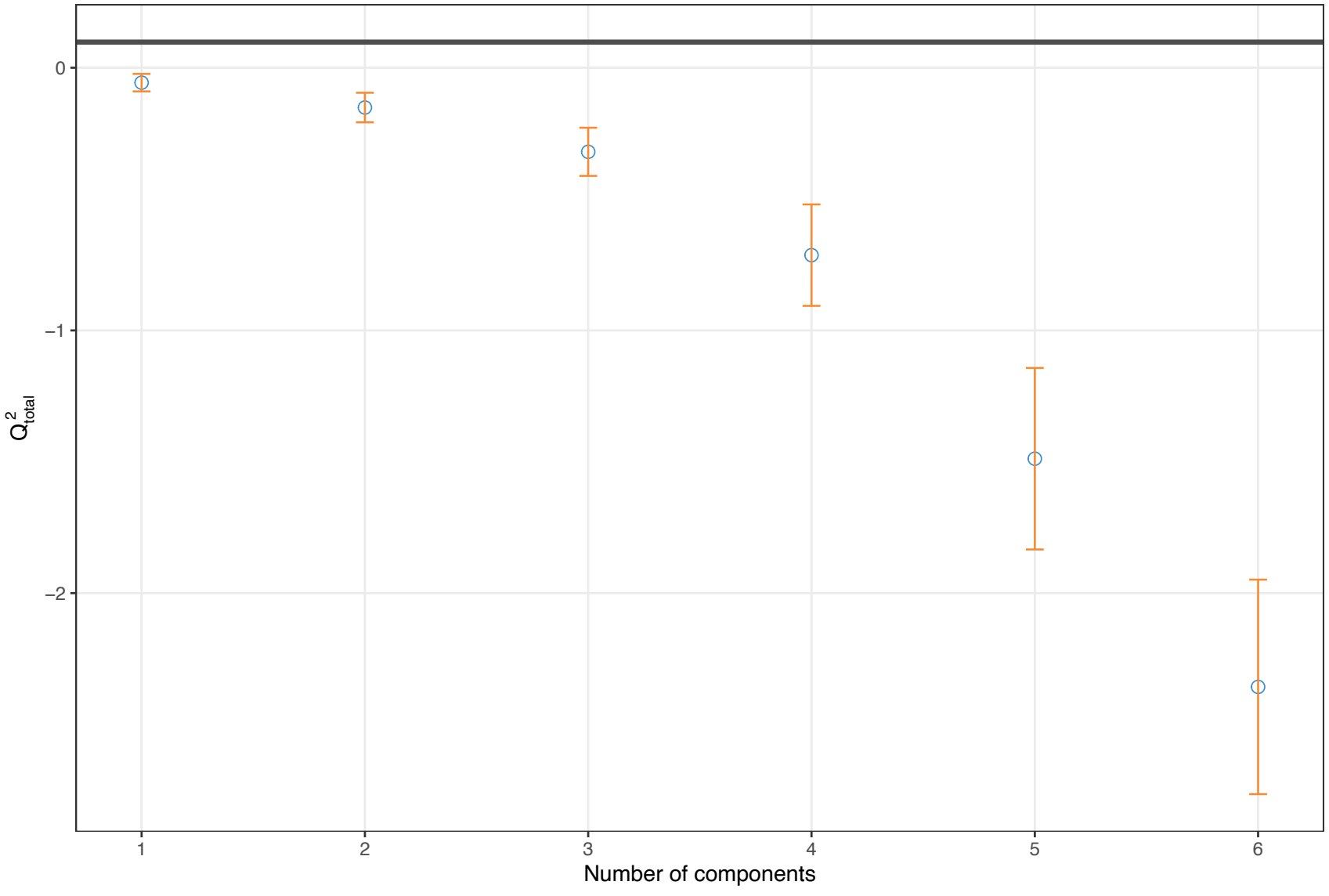
Tuning parameters and numerical outputs

- choose the number of components to retain `ncomp`.
- `perf` function and repeated k-fold cross-validation to calculate the Q^2 criterion used in the SIMCA-P software.
- The rule of thumbs is that a PLS component should be included in the model if its value is $\leqslant 0.0975$. Here we use 5-fold CV repeated 10 times.
- We run a PLS model with a sufficient number of components first, then run `perf` on the object.

```
MyResult.pls <- pls(X,Y, ncomp = 6)
set.seed(30) # for reproducbility
perf.pls <- mixOmics::perf(MyResult.pls, validation = "Mfold", folds = 5,
                  progressBar = FALSE, nrepeat = 10)
```

Q^2

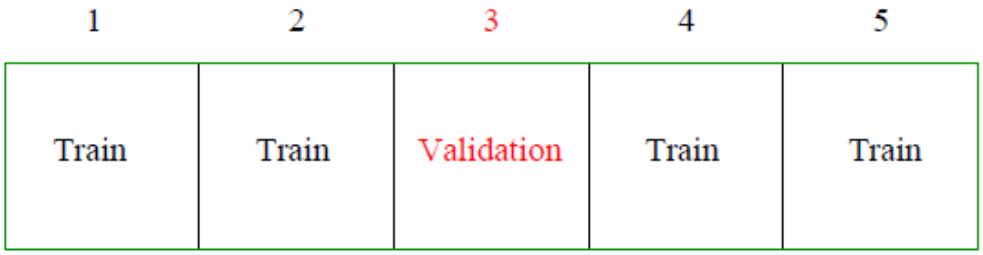
```
plot(perf.pls, criterion = 'Q2.total')
```



Reminder of Cross-Validation

- One idea is to split the data set into two fractions, then use one portion to fit the model and the other to evaluate how well the estimated modelpredicted the observations in the second portion.
- The problem with this solution is that we rarely have so much data that we can freely part with half of it solely for the purpose of choosing tuning parameters.
- To finesse this problem, cross-validation splits the data into K folds, fits the data on $K - 1$ of the folds, and evaluates risk on the fold that was left out.

K-fold cross validation

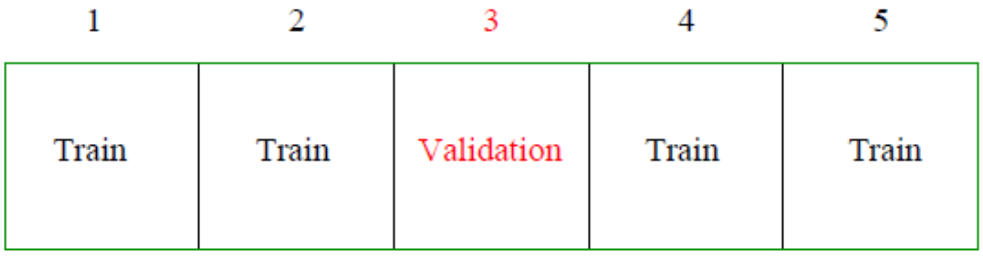


Let $k : 1, \ldots, N \rightarrow 1, \ldots, K$ the function indicating the partition to which observation i is allocated:

$$CV = \frac{1}{n} \sum_{i=1}^{n} (y^{(i)} - \widehat{f}^{-k(i)}(X^{(i)}))^2$$

where $\widehat{f}^{-k(i)}$ is the prediction of the subject i based on a model fitted with the $k(i)$th part of the data removed.

M-K-fold cross validation



Let $k : 1, \ldots, N \rightarrow 1, \ldots, K$ the function indicating the partition to which observation i is allocated:

$$CV = \frac{1}{M} \sum \left(\frac{1}{n} \sum_{i=1}^{n} (y^{(i)} - \widehat{f}^{-k(i)}(X^{(i)}))^2 \right)$$

where $\widehat{f}^{-k(i)}$ is the prediction of the subject i based on a model fitted with the $k(i)$th part of the data removed.

Cross-validation: Leave One Out (loo)

- $\widehat{Y}_i$ is the prediction of the i-th observation obtained with the model fitted on all the observation.
- $\widehat{Y}_i^{(-i)}$ is the prediction of the i-th observation obtained with the model fitted on all the observation except the i-th observation.

The cross-validation approach compares predictions $\widehat{Y}_i^{(-i)}$ to observations Y_i.

Choice of the number of latent variables H using Q^2_H

Determine $\widehat{H}$ by cross-validation

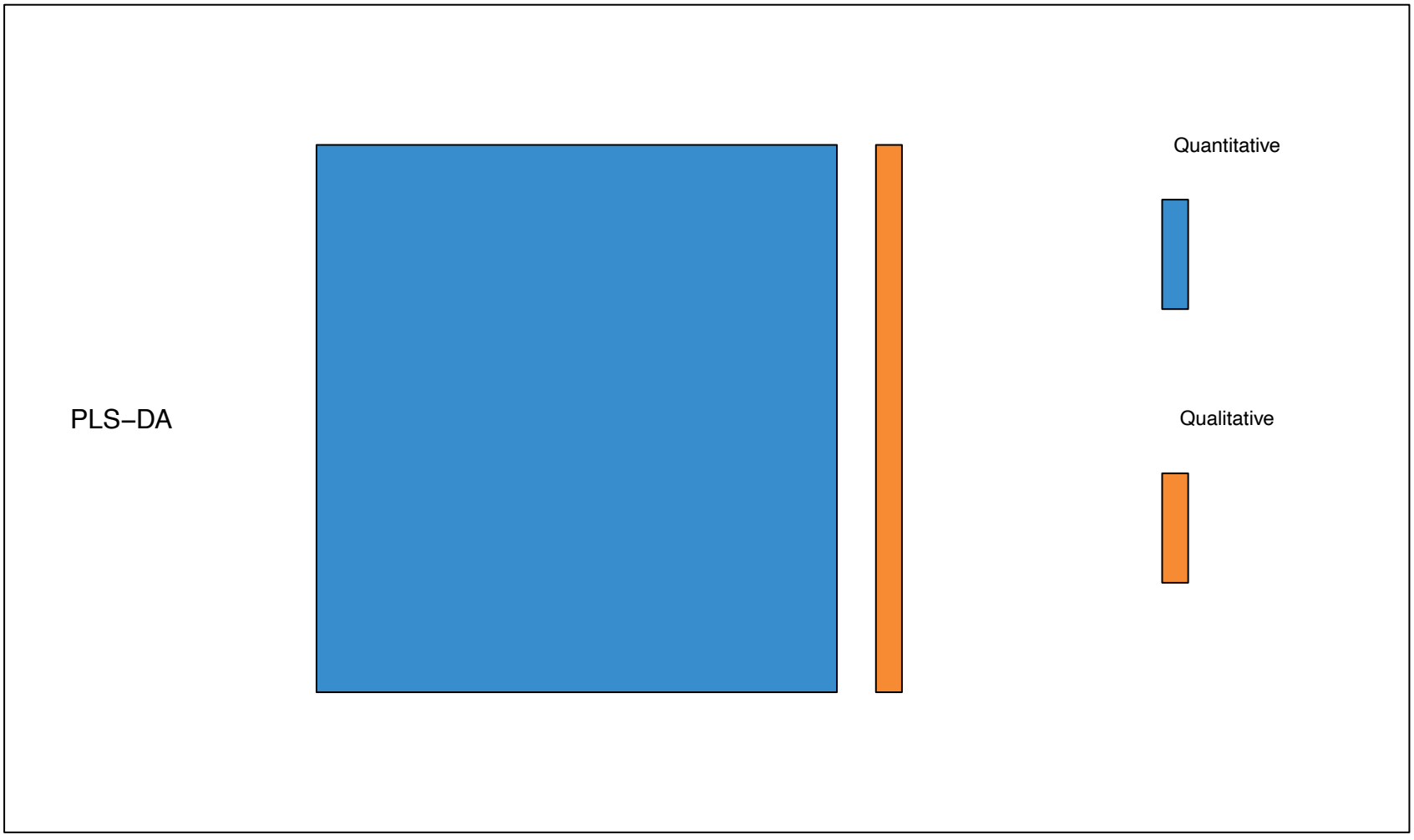
For each $H = 1 \ldots n$:

1. Evaluate $\widehat{Y}_i^H$ and $\widehat{Y}_i^{H(-i)}$
2. Evaluate *Residual Sum of Squares* : $RSS_H = \sum_{i=1}^{n} \left(Y_i - \widehat{Y}_i^H \right)^2$
3. Evaluate *PRediction Error Sum of Squares* :
   $PRESS_H = \sum_{i=1}^{n} \left(Y_i - \widehat{Y}_i^{H(-i)} \right)^2$
4. Evaluate $Q^2_H = 1 - \frac{PRESS_H}{RSS_{H-1}}$

PLS - Discriminant Analysis (PLS-DA)

Introduction

PLSDA overview



Motivation

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.

Partial Least-Squares Discriminant Analysis (PLS-DA) is a popular machine learning as a useful feature selector and classifier.

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 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
```

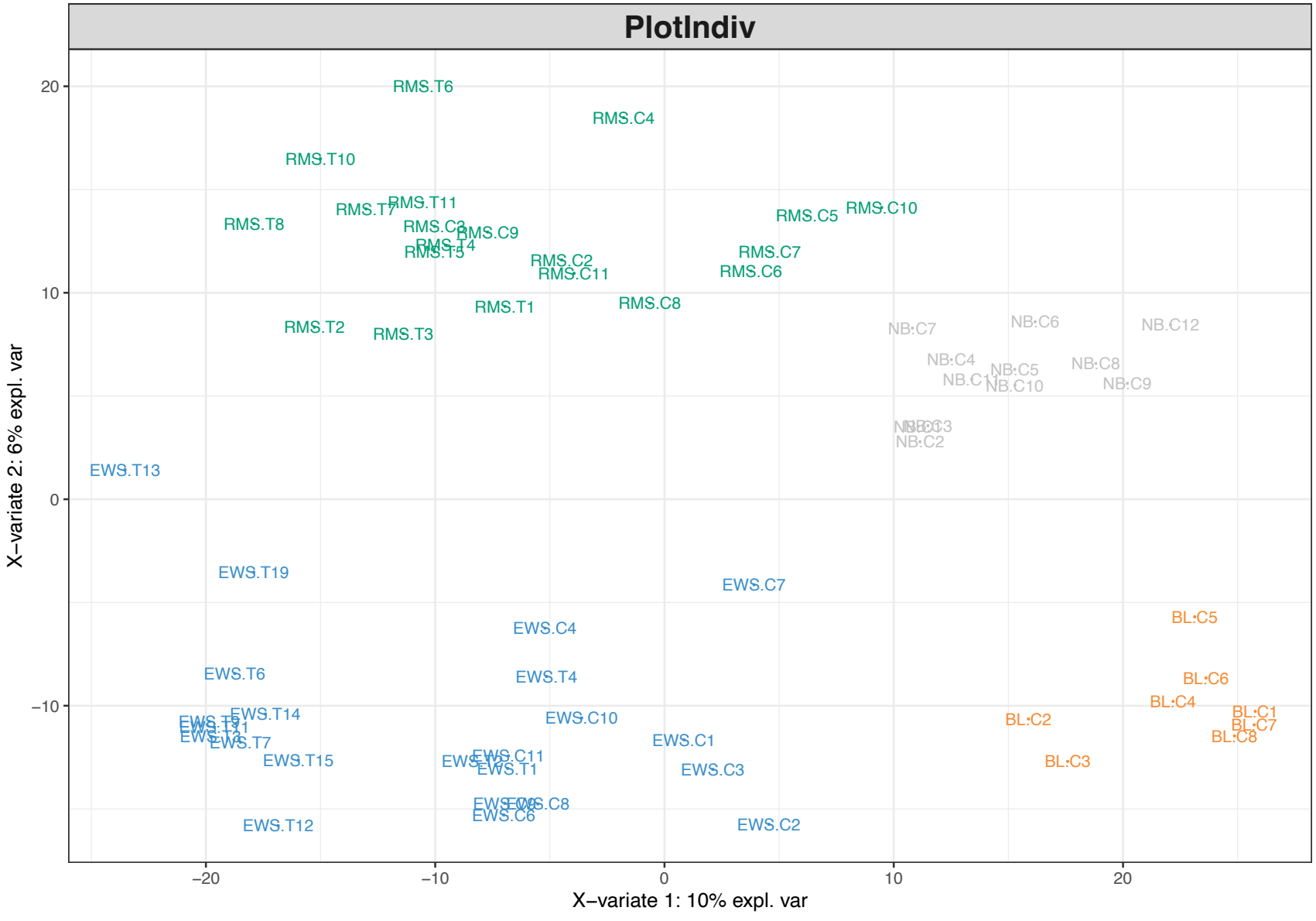
```
## [1] 63
```

Quick start

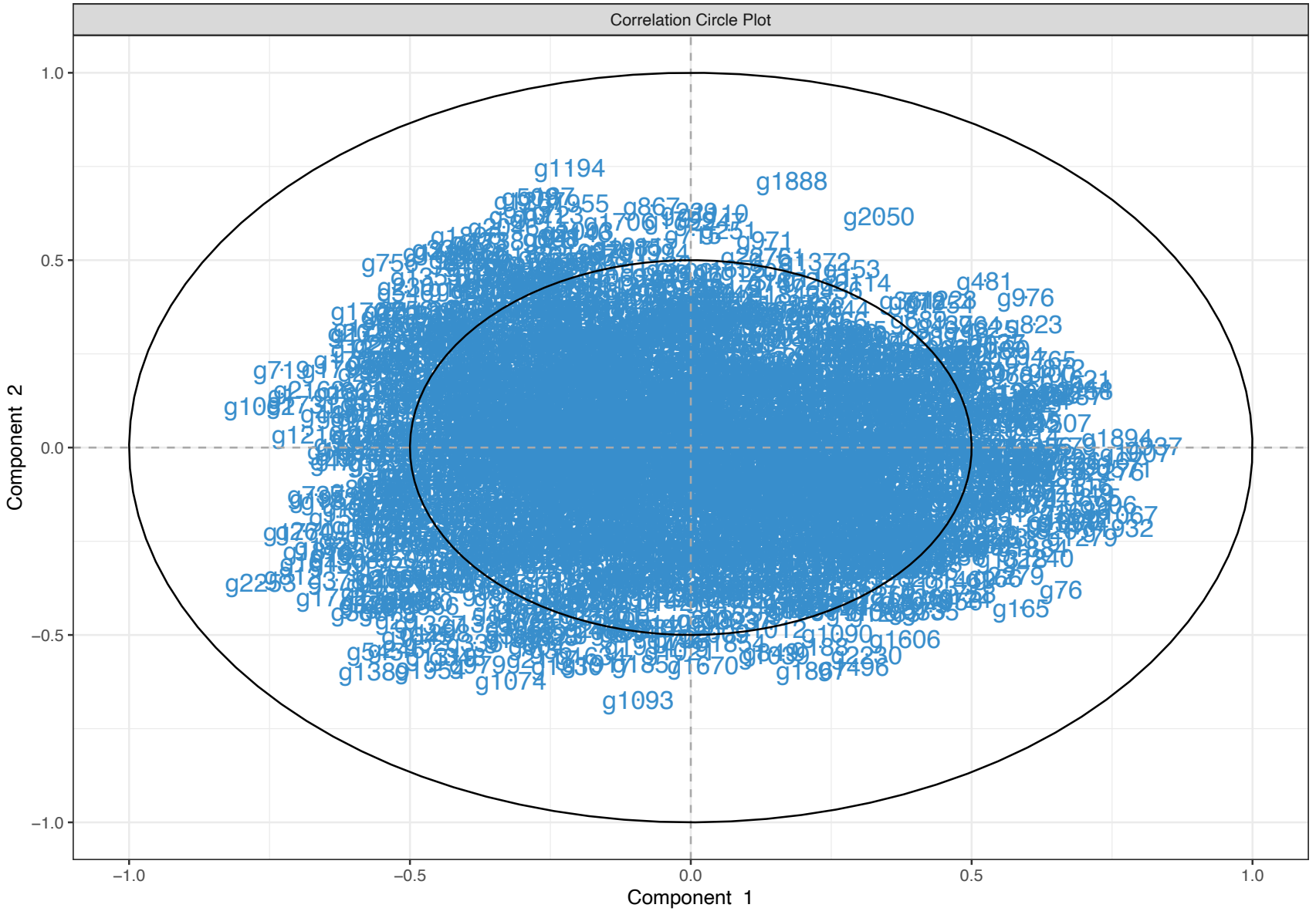
```
# 1 Run the method
MyResult.plsda <- mixOmics::plsda(X, Y, ncomp=10)
# 2 Plot the samples
plotIndiv(MyResult.plsda)
```

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

Plot samples



Plot variables

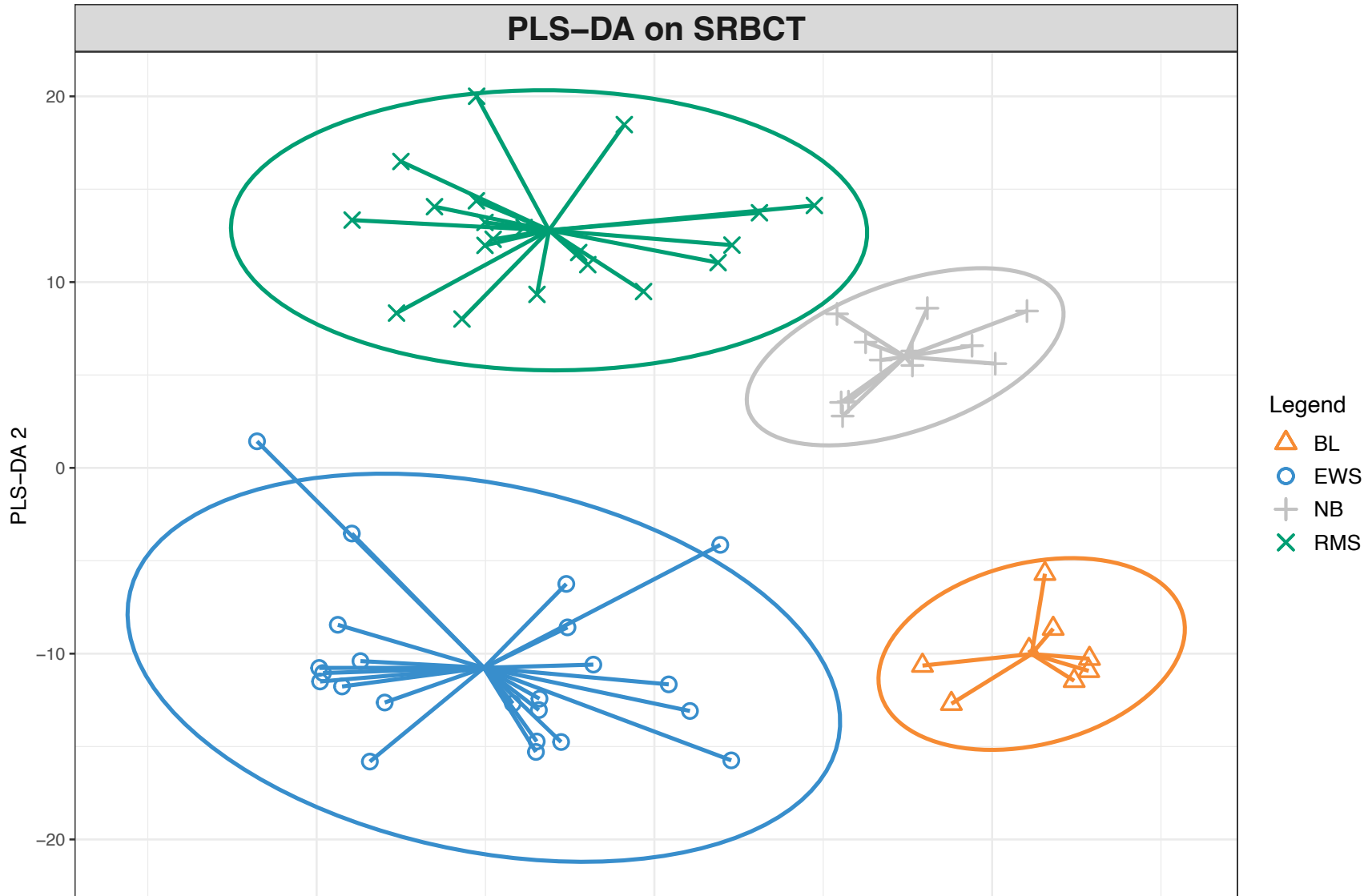


Comments

- 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).
- 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!

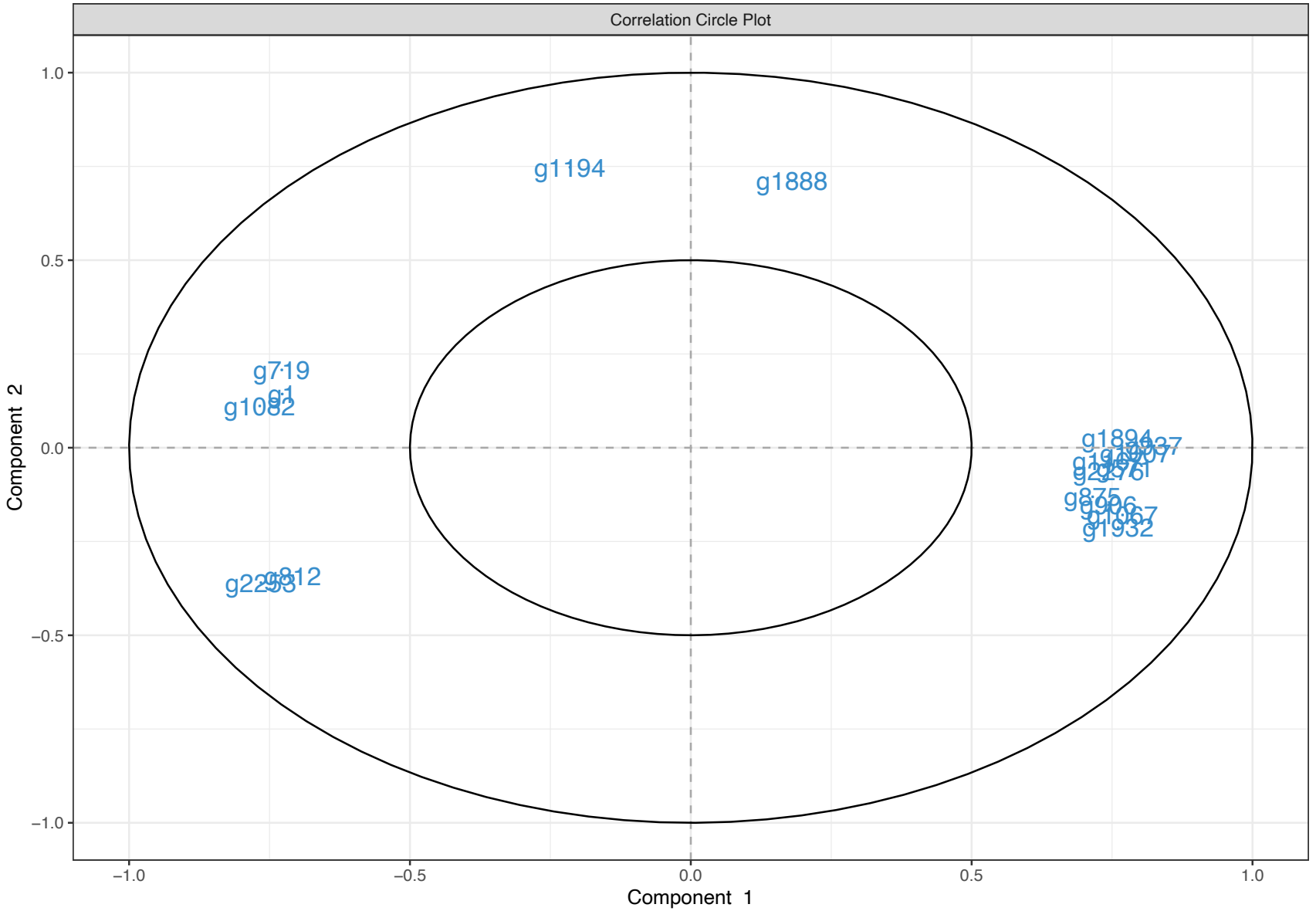
Customize sample plots

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



Customize variable plots

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

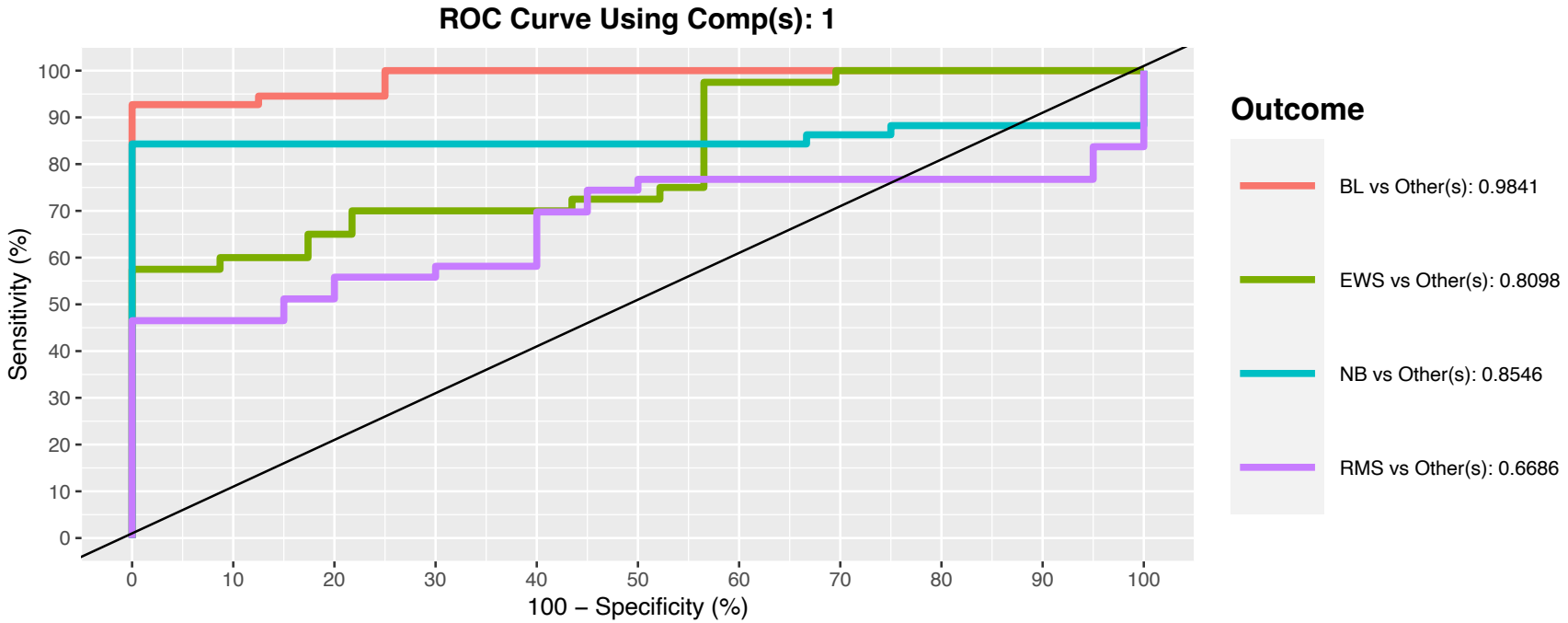


Only the display was altered to show a subset of variables

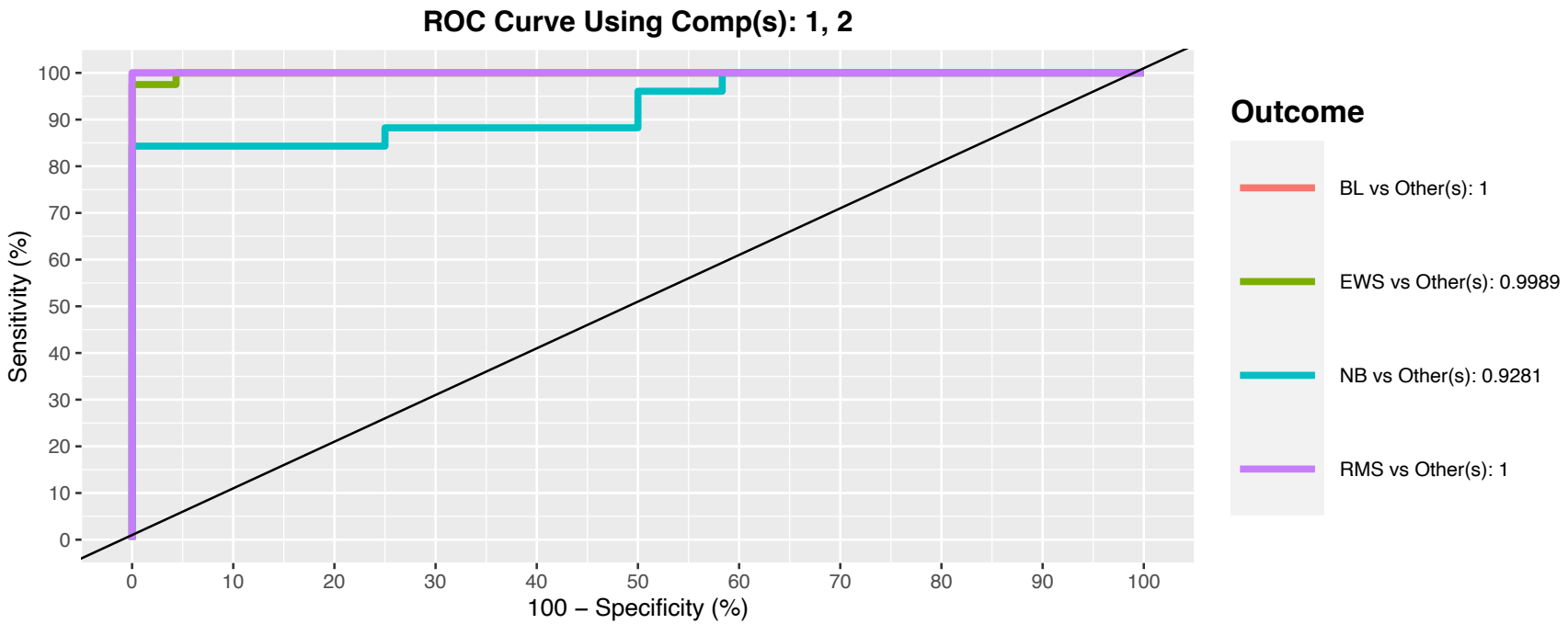
ROC

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

```
auc.plsda <- auroc(MyResult.plsda,roc.comp=1)
```



```
auc.plsda <- auroc(MyResult.plsda,roc.comp=2)
```



Tuning parameters and numerical outputs

For this method, 2 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 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):

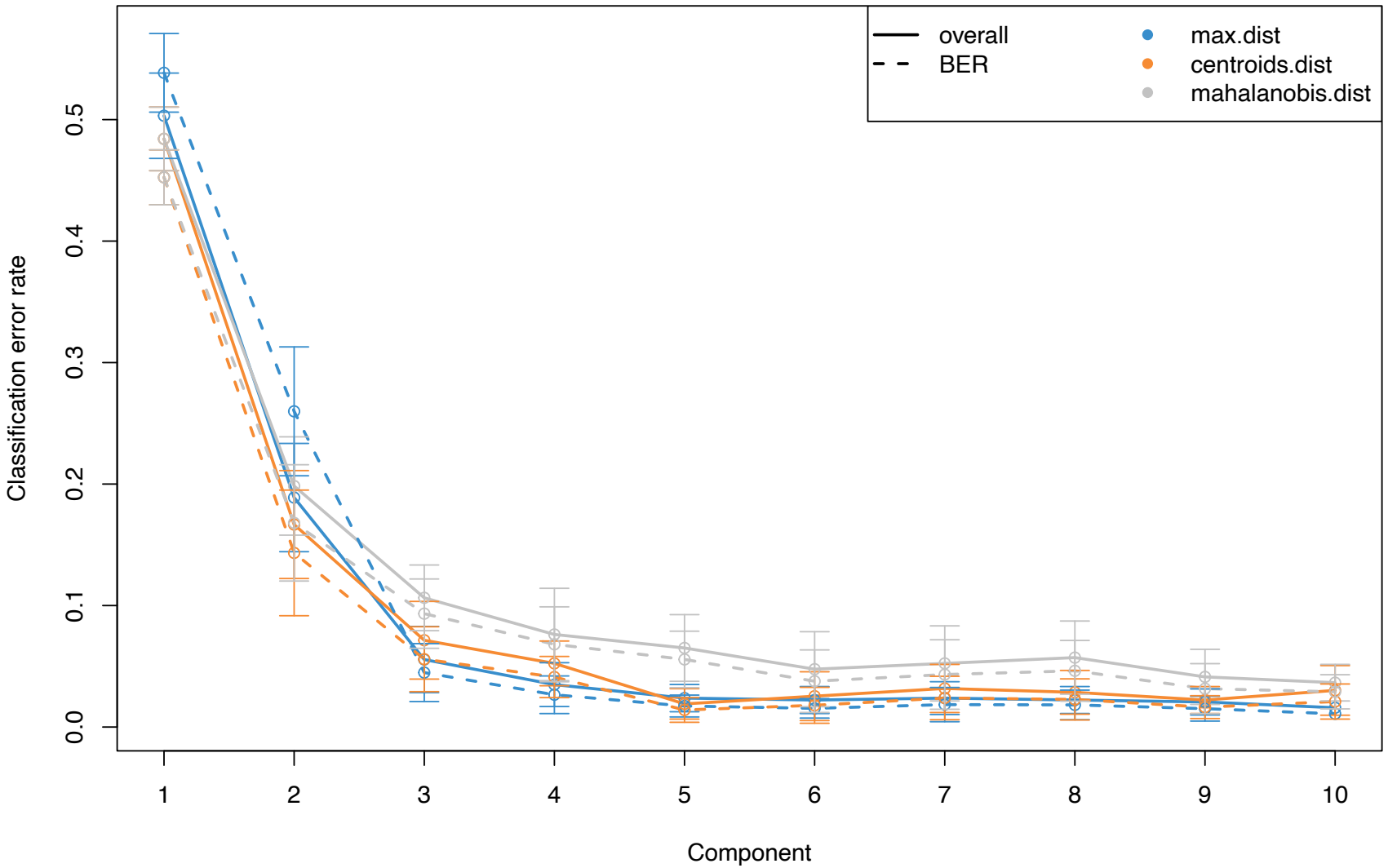
Tuning parameters and numerical outputs

```
plsda.srbct <- mixOmics::plsda(X,Y, ncomp = 10)

set.seed(30) # For reproducibility with this handbook, remove otherwise
perf.plsda.srbct <- mixOmics::perf(plsda.srbct, validation = 'Mfold', folds = 3,
                  progressBar = FALSE,  # Set to TRUE to track progress
                  nrepeat = 10)         # We suggest nrepeat = 50

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

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`.

Classification performance

We can rerun a more extensive performance evaluation with more repeats for our final model:

```
set.seed(30)
final.plsda.srbct <- mixOmics::plsda(X,Y, ncomp = 3)
perf.final.plsda.srbct <- mixOmics::perf(final.plsda.srbct,
            validation = 'Mfold',folds = 3,
            progressBar = FALSE,nrepeat = 10) # we recommend 50
```

- final overall performance for 3 components:

```
perf.final.plsda.srbct$error.rate$BER[, 'max.dist']
```

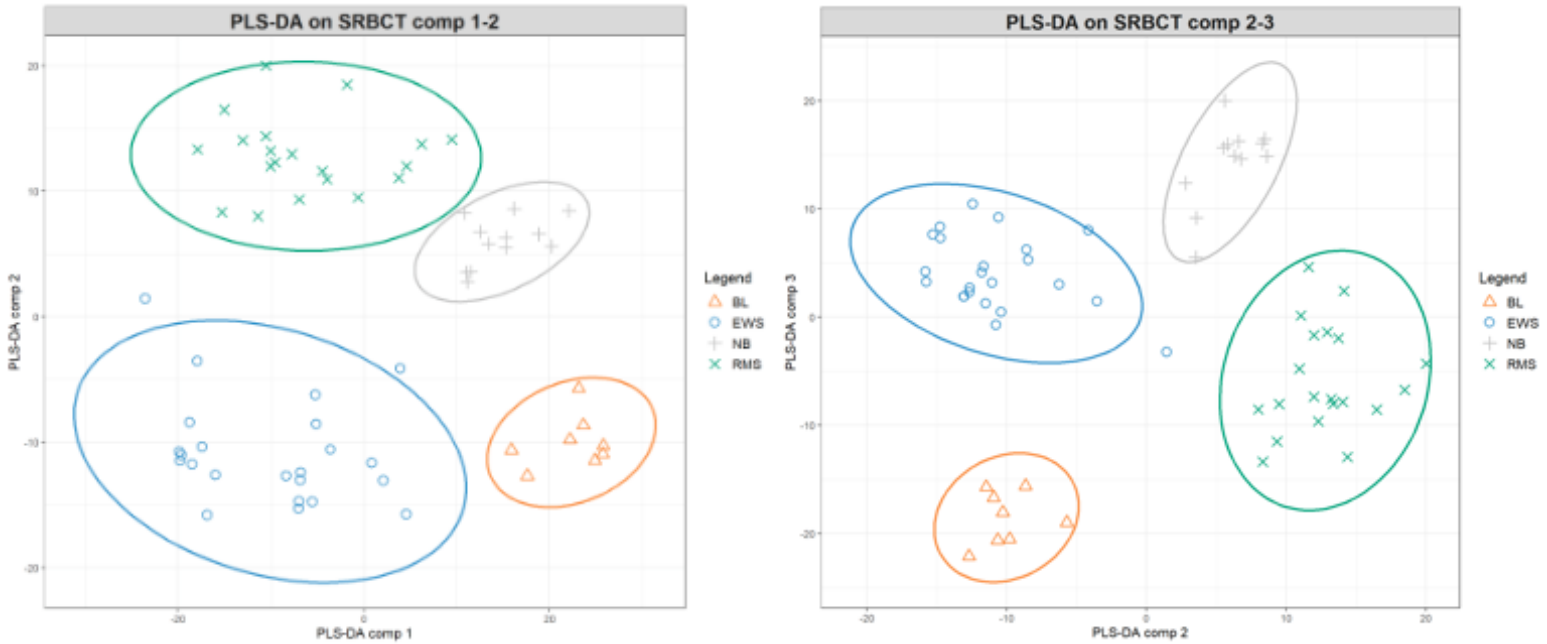
```
##      comp1      comp2      comp3
## 0.53850543 0.25986413 0.04481884
```

- As well as the error rate per class across each component:

```
perf.final.plsda.srbct$error.rate.class$max.dist
```

```
##         comp1      comp2      comp3
## EWS 0.2565217 0.08695652 0.08260870
## BL  0.8125000 0.51250000 0.00000000
## NB  0.3000000 0.37500000 0.04166667
## RMS 0.7850000 0.06500000 0.05500000
```

Final plot


PLS-DA on SRBCT comp 1-2
PLS-DA comp 2
PLS-DA comp 1
Legend
BL
EWS
NB
RMS
PLS-DA on SRBCT comp 2-3
PLS-DA comp 3
PLS-DA comp 2
Legend
BL
EWS
NB
RMS

Take Home Message

- Dimension Reduction approach for 2 blocs of Data

- Supervised method

- Finds successive pairs of latent (score) vector which are most correlated

- Symmetric relationship. Analyse the shared information.

-Asymmetric relationship. There is a set of response and predictor variables that can be used for prediction

- PLS-DA: supervised method for qualitative response variable

- Similar as discriminant analysis