Example 4 - DIABLO

Maxime Hervé, Florence Nicolè and Kim-Anh Lê Cao

16/10/2017

Contents

Packages to load

```
library(mixOmics)      # Functions needed: block.plsda, plotDiablo, circosPlot, cimDiablo,
                       #                   network
library(RVAideMemoire) # Functions needed: DIABLO.cv, DIABLO.test, MVA.synt, MVA.plot
```

Data loading

Perianths

The first dataset is composed of 15 rows (15 samples, 5 per genotype) and 41 columns (the genotype and the absolute concentration of 40 metabolites).

```
tab.Perianths <- read.table("Example 4 - Perianths.txt",header=TRUE)
```

Anthers

The second dataset is composed of the same 15 rows and 44 columns (the genotype and the absolute concentration of 43 metabolites).

```
tab.Anthers <- read.table("Example 4 - Anthers.txt",header=TRUE)
```

The grouping factor (genotype) can be taken from any of the two datasets since rows are identical:

```
Genotype <- tab.Anthers$Genotype
```

Pre-treatment

Both datasets are autoscaled:

```
Chemistry.Perianths <- scale(tab.Perianths[,2:41])
Chemistry.Anthers <- scale(tab.Anthers[,2:44])
```

R-related step: all datasets are input in a single list.

```
Chemistry <- list(Perianths=Chemistry.Perianths,Anthers=Chemistry.Anthers)
```

Analysis

We fit the DIABLO model. Since we have 3 groups, we use 2 components:

```
DIABLO <- block.plsda(Chemistry,Genotype,ncomp=2)
```

It is mandatory to validate the model before any interpretation from graphical outputs. We first estimate the classification error rate by cross-validation:

```
DIABLO.cv(DIABLO)
```

```
        Cross validation

Model: DIABLO
5-fold validation
Validation repeated 10 times
2 components

Classification criterion: Mahalanobis distance

Mean (standard error) classification error rate (%): 0 (0)
```

We test the significance of the discrimination:

```
# This may take several mminutes to run
# Remove progress=FALSE to see computation progress
DIABLO.test(DIABLO,progress=FALSE)
```

```
    Permutation test based on cross-validation

data:  DIABLO
DIABLO (2 components)
999 permutations
CER = 0.01, p-value = 0.001
```

The discrimination is significant, indicating that there is a significant difference between the three genotypes.

How well are components from each block correlated?

```
MVA.synt(DIABLO)
```

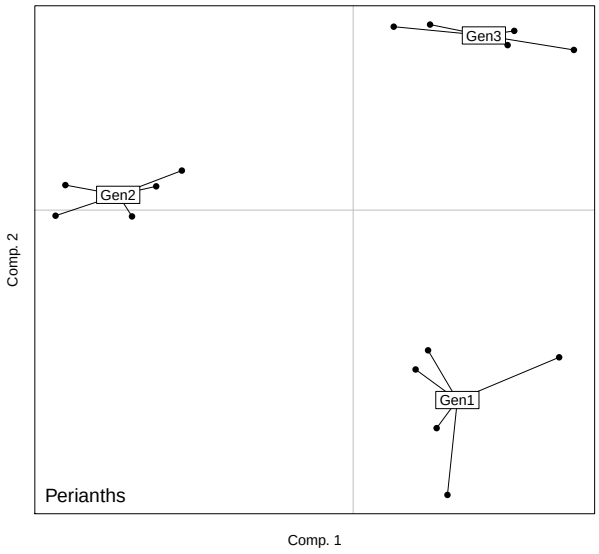
```
Criterion: inter-block correlations - Axes 1
       Block1
Block2 0.9612
```

```
Criterion: inter-block correlations - Axes 2
       Block1
Block2 0.9509

Criterion: intra-block total variance (%) - Block 1
 Axis Proportion Cumulative
    1      31.78      31.78
    2      20.87      52.64

Criterion: intra-block total variance (%) - Block 2
 Axis Proportion Cumulative
    1      32.62      32.62
    2      34.91      67.54
```

The components of the two datasets are highly positively correlated (0.96 for the first pair, 0.95 for the second pair). Those correlations are shown later in the graphical outputs.

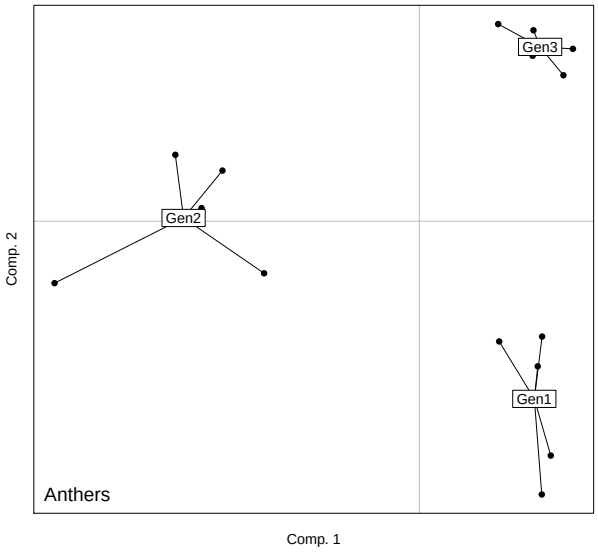
We draw the score plot of the first block (Perianths):

```
MVA.plot(DIABLO,fac=Genotype,space=1,drawextaxes=FALSE,main="Perianths")
```



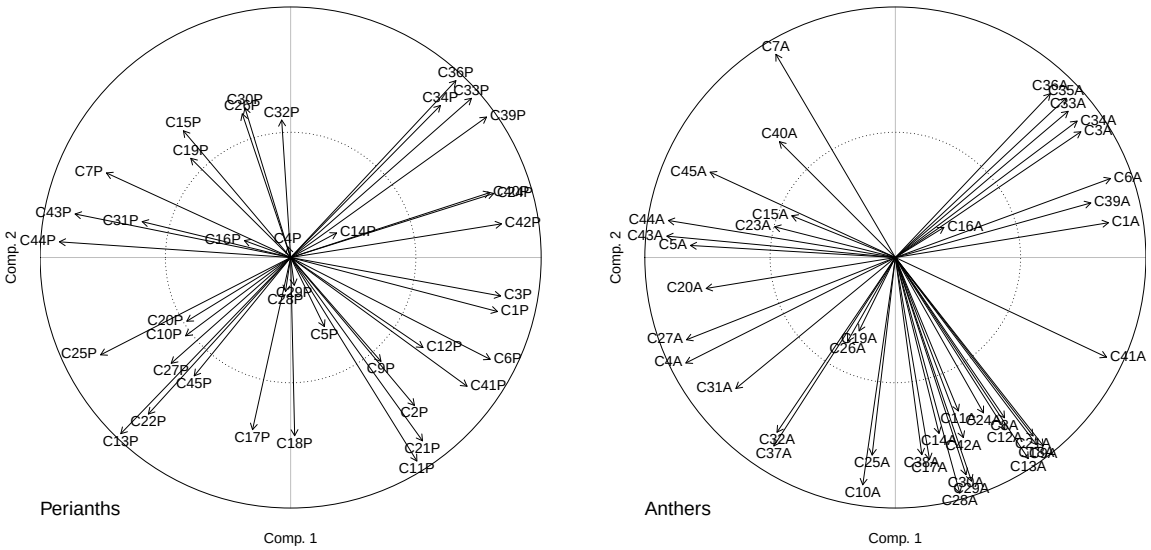
Then the score plot of the second block (Anthers):

```
MVA.plot(DIABLO,fac=Genotype,space=2,drawextaxes=FALSE,main="Anthers")
```



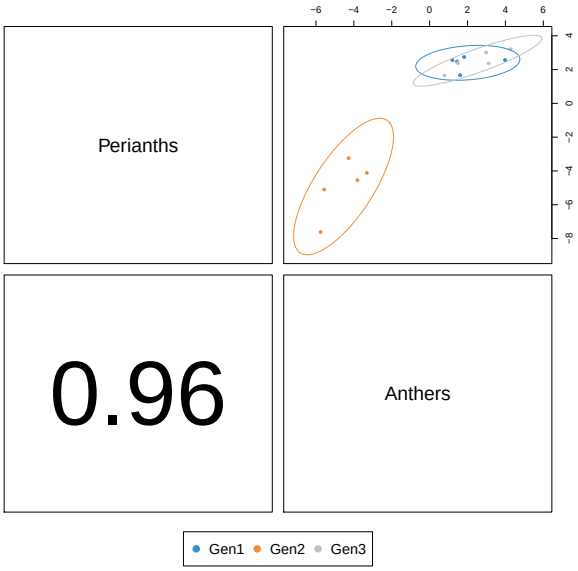
We draw the correlation circle plot of the two blocks to identify the discriminant and correlated compounds:

```
par(mfrow=c(1,2))
MVA.plot(DIABLO,"corr",space=1,main="Perianths")
MVA.plot(DIABLO,"corr",space=2,main="Anthers")
```

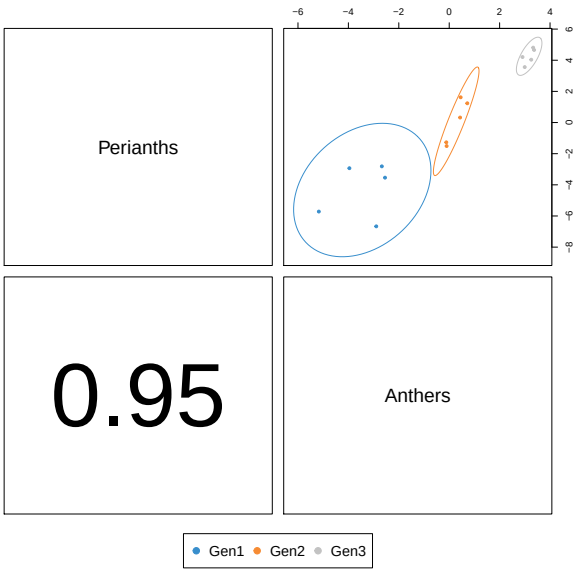


Another way to look at the the integrative power of DIABLO is to display how correlated the components are, and whether they are able to discriminate the different groups:

```
# First pair of components
plotDiablo(DIABLO,ncomp=1)
```



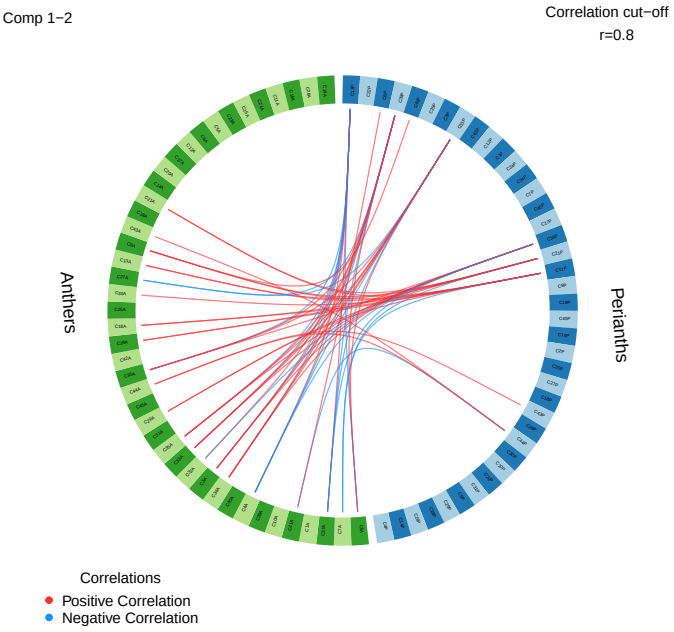

```
# Second pair of components
plotDiablo(DIABLO,ncomp=2)
```



We observe here that the first set of components discriminates Genotype 2 *vs.* the other genotypes, while the second set of components discriminates all three genotypes. The high correlation coefficients indicate that the method is able to extract correlated information between the two datasets.

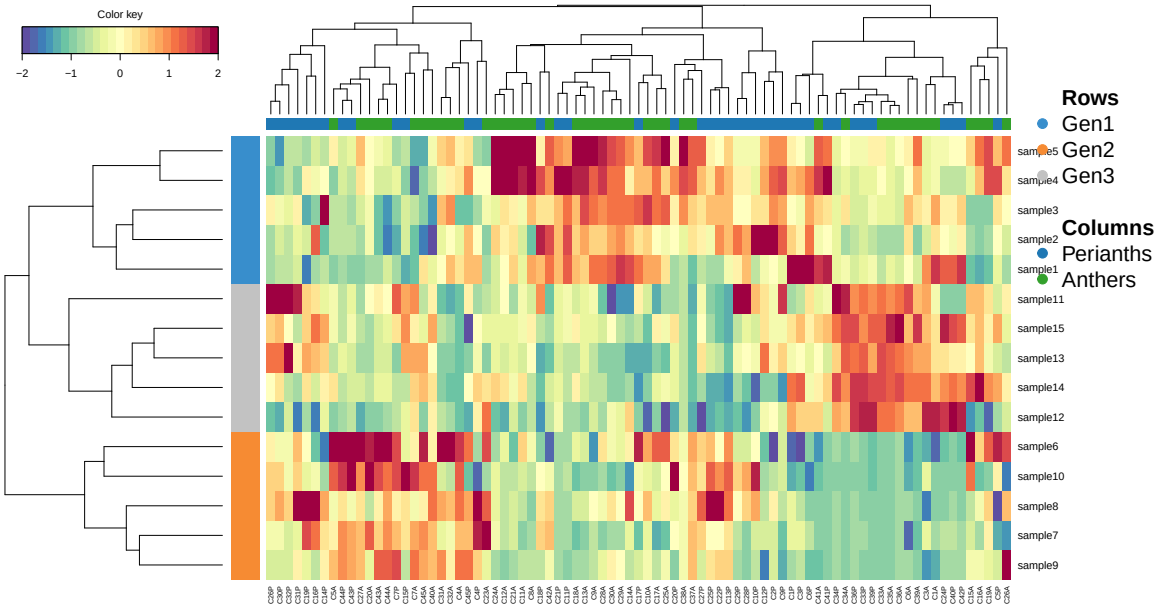
Other useful representations are available, such as the 'circos plots' where only pairwise correlations >= 0.8 are represented (this threshold can be set differently). This plot enables to visualize the correlation structure between variables from the two datasets (represented in the quadrants) and the nature of the correlation (positive or negative):

```
class(DIABLO)[1] <- "block.splsda" # R-related step
circosPlot(DIABLO,cutoff=0.8,line=FALSE)
```



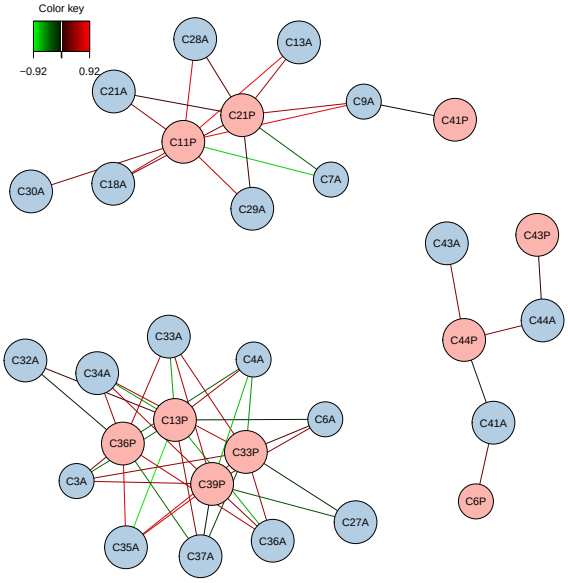
The 'clustered image maps' (CIM) with the default settings Euclidian distance/complete linkage clustering method, enable to visualize the relationship between the samples (in rows) and the compounds (in columns):

```
cimDiablo(DIABLO)
```



Finally the relevance network shows similar information as the circos plot, for those interested in visualizing subgroups of highly correlated variables (correlation threshold >=0.8 here):

```
network(DIABLO,cutoff=0.8)
```



Information on the current R session

```
sessionInfo()
```

```
R version 3.4.0 (2017-04-21)
Platform: x86_64-w64-mingw32/x64 (64-bit)
Running under: Windows 7 x64 (build 7601) Service Pack 1

Matrix products: default

locale:
[1] LC_COLLATE=French_France.1252  LC_CTYPE=French_France.1252
[3] LC_MONETARY=French_France.1252 LC_NUMERIC=C
[5] LC_TIME=French_France.1252

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] RVAideMemoire_0.9-68 mixOmics_6.1.3       ggplot2_2.2.1
[4] lattice_0.20-35      MASS_7.3-47          knitr_1.17

loaded via a namespace (and not attached):
 [1] tidyselect_0.2.2   purrr_0.2.3        reshape2_1.4.2
 [4] splines_3.4.0      colorspace_1.3-2   htmltools_0.3.6
 [7] yaml_2.1.14        mgcv_1.8-22        rlang_0.1.2
[10] nloptr_1.0.4       glue_1.1.1         RColorBrewer_1.1-2
[13] bindrcpp_0.2       plyr_1.8.4         bindr_0.1
[16] stringr_1.2.0      MatrixModels_0.4-1 munsell_0.4.3
[19] gtable_0.2.0       htmlwidgets_0.9    evaluate_0.10.1
```

```
[22] permute_0.9-4      SparseM_1.77       httpuv_1.3.5
[25] quantreg_5.33      pbkrtest_0.4-7     parallel_3.4.0
[28] Rcpp_0.12.13       xtable_1.8-2       corpcor_1.6.9
[31] scales_0.5.0       backports_1.1.1    vegan_2.4-4
[34] jsonlite_1.5       mime_0.5           lme4_1.1-14
[37] ellipse_0.3-8      digest_0.6.12      stringi_1.1.5
[40] dplyr_0.7.4        shiny_1.0.5        grid_3.4.0
[43] rprojroot_1.2      ade4_1.7-8         tools_3.4.0
[46] magrittr_1.5       rgl_0.98.1         lazyeval_0.2.0
[49] tibble_1.3.4       cluster_2.0.6      tidyr_0.7.1
[52] car_2.1-5          pkgconfig_2.0.1    Matrix_1.2-11
[55] assertthat_0.2.0   minqa_1.2.4        rmarkdown_1.6
[58] R6_2.2.2           nnet_7.3-12        igraph_1.1.2
[61] nlme_3.1-131       compiler_3.4.0
```