visCorVar

Maxime BRUNIN

27 juillet 2020

Contents

Introduction **2**

Data integration **2**

Pre-processing **2**

Correlation circle **3**

Network **4**

Introduction

The package visCorVar enables to visualize results from data integration with the function block.splsda of the package mixOmics. The data integration is performed for different types of omics data (transcriptomics, metabolomics, transcriptomics) in order to select high correlated variables of different blocks and the predict the class membership of a new sample. A function is provided to determine the blocks whose correlation circles can be overlaid. The package visCorVar performs the overlaying of correlation circles of blocks and the zoom in a rectangle of correlation circle to select subsets of relevant correlated variables. Finally, a network in graphml format is built from selected block variables, variables of interest (optional) and response variables. Links are drawn between variables if these variables are correlated. The network can be visualized with Cytoscape.

Data integration

Before using the package visCorVar, a data integration has to be performed with function block.splsda (package mixOmics).

```
res_data_integration = block.splsda(X = list_X,
                                    Y = factor_Y,
                                    ncomp = ncomp,
                                    keepX = keepX,
                                    design = design,
                                    scheme = scheme,
                                    mode = mode,
                                    max.iter = max_iter)
```

The paramter X contains the blocks measured on the same samples. The parameter Y contains the condition associated with each sample. The parameter ncomp is the number of components to compute. The parameter keepX contains for each block and each component the number of selected block variables. The parameter design is a matrix indicating which blocks are connected. For the block.splsda function, the default value of the parameter scheme and mode is “horst” and “regression” respectively. The parameter max.iter is the maximum number of iterations of the algorithm.

Pre-processing

The pre-processing enables to determine the blocks whose correlation circles can be overlaid and to compute the correlations between the components and the selected block variables, the response variables and variables of interest (optional). This computation of the correlations are necessary to plot the correlation circle and to create a network. This pre-processing is done by the function matCorAddVar. In this section, some parameters are detailed and an example of a call of the function matCorAddVar is given.

Each column of the matrix block_Y is associated with a condition (in this example, these conditions are A and B), this column defined a response variable. For the function block.splsda, the response variable si equal to 1 if the sample is in this condition and is equal to 0 otherwise. The table below shows an example of block_Y :

A	B
1	0
0	1
0	1
1	0
1	0
0	1

A	B
1	0
0	1
1	0
1	0
1	0
1	0
1	0
1	0
0	1
1	0
0	1
0	1
0	1
1	0
0	1
1	0
1	0
1	0

The variables of interest are variables that the user want to have in the network. The variables of interest have to be block variables. The table below shows an example of variables of interest.

var
var_block1_3
var_block1_5
var_block1_6
var_block2_13
var_block2_17
var_block_301

```
data(res_data_integration)
comp = 1:2
cutoff_comp = 0.8
res_matCorAddVar = matCorAddVar(res_block_splsda = res_data_integration,
                                block_Y = block_Y,
                                cutoff_comp = cutoff_comp,
                                var_interest = var_interest,
                                comp = comp)
```

The description of the input parameters is provided in the help of the function matCorAddVar.

Correlation circle

If it is possible to overlay the correlation circles, the function circleCor performs the overlaying of blocks and a zoom in a rectangle of correlation circle. Only the selected block variables whose correlation with comp[1] component or comp[2] component is greater than cutoff in absolute value are plotted in the correlation circle. The function circleCor returns a subset of relevant correlated variables contained in the rectangle.

```
library(RColorBrewer)
list_cor_comp_selected_var_resp_var = res_matCorAddVar$list_cor_comp_selected_var_resp_var
```

```
list_vec_index_block_select = res_matCorAddVar$list_vec_index_block_select
mat_cor_comp1 = res_matCorAddVar$mat_cor_comp1
mat_cor_comp2 = res_matCorAddVar$mat_cor_comp2
names_blocks = c("X1", "X3")
names_response_variables = c("A", "B")
comp = 1:2
vec_col = colorRampPalette(brewer.pal(9, "Spectral"))(dim(mat_cor_comp1)[1] + 1)

names_block_variables=circleCor(list_dataframe_cor_comp_var_global=list_cor_comp_selected_var_resp_var,
                                list_vec_index_block_select = list_vec_index_block_select,
                                mat_cor_comp1 = mat_cor_comp1,
                                mat_cor_comp2 = mat_cor_comp2,
                                names_blocks = names_blocks,
                                names_response_variables = names_response_variables,
                                comp = comp,
                                cutoff = 0.85,
                                min.X = -1,
                                max.X = 1,
                                min.Y = -1,
                                max.Y = 1,
                                vec_col = vec_col,
                                rad.in = 0.5,
                                cex = 0.7,
                                cex_legend = 0.8,
                                pos = c(1.2, 0),
                                pch = 20)
```

The description of the input parameters is provided in the help of the function circleCor.

Network

A network can be created with the function networkVar and can be exported in the graphml format for visualization with Cytoscape. An approximation of the correlation between two variables (called similarity between two variables) is associated with the link between two variables. Before creating the network with the function networkVar, similarity matrices have to be computed with the function computeMatSimilarity.

```
comp = 1:2
cutoff_comp = 0.8
res_compute_mat_similarity = computeMatSimilarity(res_matCorAddVar = res_matCorAddVar)
```

The description of the input parameters is provided in the help of the function computeMatSimilarity

The nodes of this network are the response variables, the variables of interest (optional) and the selected block variables whose similarity with another variable is greater than a threshold in absolute value. This network gives a insight of correlated variables.

```
names_blocks = c("X1", "X3")
names_response_variables = c("A", "B")
comp = 1:2
names_resp_var2 = c("A")
res_networkVar = networkVar(res_compute_mat_similarity = res_compute_mat_similarity,
                            names_block_variables = names_block_variables,
                            names_response_variables = names_resp_var2,
                            cutoff = 0)
#> Warning in networkVar(res_compute_mat_similarity = res_compute_mat_similarity, : 1 variables of inte
```

```
#>                  are not block variables. These variables will not be in the network.
```

The description of the input parameters is provided in the help of the function networkVar.