Introduction to MultiVarSel

Marie Perrot-Dockès, Céline Lévy-Leduc, Julien Chiquet

2019-03-20

Introduction and installation

This vignette explains basic usage of **MultiVarSel**, an R package to perform variable selection in the multivariate linear model taking into account the dependence that may exist between the responses. The model can be described as follows :

$$Y = XB + E, \tag{1}$$

where Y is a $n \times q$ matrix of responses, X is a $n \times p$ matrix of covariables, B is a $p \times q$ sparse matrix of coefficients and E is a random error matrix such that $\forall i \in \{1, \cdots, n\}$, $E_i = (E_{i,1}, \ldots, E_{i,q}) \overset{iid}{\sim} \mathcal{N}(0, \Sigma)$. The package consists in estimating Σ beforehand and to plug this estimator in a Lasso criterion, in order to obtain a sparse estimator of the coefficient matrix B.

The package has to be installed and then loaded as follows :

```
require(MultiVarSel)
require(ggplot2)
```

Numerical experiments

We first show an application of our methodology to a simulated data set where the covariance matrix Σ is the covariance matrix of an AR(1) process.

Data generation

We start by generating a random error matrix E as described in the introduction as follows.

```
n <- 30
q <- 1000
p <- 5
rho <- 0.9
sparsity <- 0.01
E <- t(replicate(n, arima.sim(q, model = list(ar = rho, ma = 0))))
```

We then generate a sparse matrix B of coefficients and a matrix of covariables X.

```
s <- round(sparsity * p * q)
ij <- arrayInd(sample(1:(p * q), size = s), c(p, q))
values <- runif(s, 1, 2) * sample(c(-1, 1), s, replace = T)
B <- sparseMatrix(
  i = ij[, 1], j = ij[, 2],
  x = values,
  dims = c(p, q)
)
```

```
X <- matrix(rnorm(n * p), n, p)
Y <- X %*% B + E
```

Whitening test

To apply our methodology we start by estimating the matrix E which is obtained by fitting a linear model with the design matrix X to all the columns of Y as if they were independent:

```
E_hat <- residuals(lm(as.matrix(Y) ~ X + 0))
```

We then use a Portmanteau test to check if each row of this matrix $\widehat{E}$ is a white noise or not.

```
whitening_test(E_hat)
```

```
## [1] 0
```

The *p*-value is smaller than 0.05 hence we reject the hypothesis that each row of the residuals matrix is a white noise which is an expected result since each row of E is an AR(1) process.

Whitening

We then try to remove the dependence among the columns of the residuals matrix by estimating the covariance matrix of the rows of E. To estimate it we try different structures for this covariance. The simplest assumption is that each row of E follows an AR(1) process, we also propose a modelling where each row is an ARMA(p,q) process and a nonparametric one where Σ is only assumed to be Toeplitz. To compare these different dependence modellings we perform a Portmanteau test on the "whitened" matrix $\widehat{E}\widehat{\Sigma}^{-1/2}$, where $\widehat{\Sigma}^{-1/2}$ is the square root of the inverse of the estimation of Σ.

With the following code we test the AR(1), ARMA(1,1) and nonparametric dependence structures :

```
whitening_choice(E_hat, c("AR1", "ARMA", "nonparam"), pAR = 1, qMA = 1)
```

```
##              Pvalue    Decision
## AR1           0.916 WHITE NOISE
## ARMA 1 1       0.92 WHITE NOISE
## nonparam          1 WHITE NOISE
```

We then select the simplest model that allows us to remove the dependence in the data, in that case the AR(1) modelling. We compute the square root of the inverse of the estimator of the covariance matrix of each row of the residuals matrix using the AR(1) modelling as follows:

```
square_root_inv_hat_Sigma <- whitening(E_hat, "AR1", pAR = 1, qMA = 0)
```

In order to whiten the data (remove the dependence), we transform the data as follows:

$$\boldsymbol{Y}\widehat{\boldsymbol{\Sigma}}_q^{-1/2} = \boldsymbol{X}\boldsymbol{B}\widehat{\boldsymbol{\Sigma}}_q^{-1/2} + \boldsymbol{E}\widehat{\boldsymbol{\Sigma}}_q^{-1/2}. \tag{2}$$

The idea is then to use the Lasso criterion introduced by Tibshirani in 1996, and available in the R package `glmnet` on these whitened data. We recall that in the classical linear model

$$\mathcal{Y} = \mathcal{X}\mathcal{B} + \mathcal{E},$$

where $\mathcal{Y}$, $\mathcal{B}$ and $\mathcal{E}$ are vectors and $\mathcal{X}$ is a matrix, the Lasso estimator of $\mathcal{B}$ is defined by

$$\widehat{\mathcal{B}}(\lambda) = \text{Argmin}_{\mathcal{B}}\left\{\|\mathcal{Y} - \mathcal{X}\mathcal{B}\|_2^2 + \lambda\|\mathcal{B}\|_1\right\}.$$

In order to be able to use the Lasso criterion we will apply the *vec* operator to (2)

$$
\begin{aligned}
\mathcal{Y} = vec(\boldsymbol{Y}\widehat{\boldsymbol{\Sigma}}^{-1/2}) &= vec(\boldsymbol{X}\boldsymbol{B}\widehat{\boldsymbol{\Sigma}}^{-1/2}) + vec(\boldsymbol{E}\widehat{\boldsymbol{\Sigma}}^{-1/2}) \\
&= ((\widehat{\boldsymbol{\Sigma}}^{-1/2})' \otimes \boldsymbol{X})vec(\boldsymbol{B}) + vec(\boldsymbol{E}\widehat{\boldsymbol{\Sigma}}^{-1/2}) \\
&= \mathcal{X}\mathcal{B} + \mathcal{E}.
\end{aligned}
$$

Variable selection

The Lasso criterion applied to $\mathcal{Y} = vec(\boldsymbol{Y}\widehat{\boldsymbol{\Sigma}}^{-1/2})$ will provide an estimation of the non null positions of $\mathcal{B} = vec(\boldsymbol{B})$ and hence the non null positions of B. In order to avoid false positive positions we add a stability selection step. These different steps (whitening, vectorization, Lasso, stability selection) are implemented in the function `variable_selection` of the R package `MultiVarSel`.

```
Frequencies <- variable_selection(
  Y = Y, X = X, square_root_inv_hat_Sigma,
  nb_repli = 1000, parallel = FALSE
)
```

Parallel computing is also supported by this function. To make it work, users must download the package doMC which is not available on Windows platforms (it is on others).

```
require(doMC)
```

```
## Loading required package: doMC
```

```
## Loading required package: iterators
```

```
registerDoMC(cores = 4)
Frequencies <- variable_selection(
  Y = Y, X = X, square_root_inv_hat_Sigma,
  nb_repli = 1000, parallel = TRUE
)
```

In the previous command line, `nb_replis` corresponds to the number of replications which is used in the stability selection. The following plot displays the frequencies at which each coefficient of B is considered as being non null.

```
p <- ggplot(
  data = Frequencies[Frequencies$frequency >= 0.95, ],
  aes(x = Names_of_Y, y = Names_of_X,
    color = frequency, fill = frequency )) +
  geom_tile(size = 0.75) +
  scale_color_gradient2(midpoint = 0.95, mid = "orange") +
  scale_fill_gradient2(midpoint = 0.95, mid = "orange") +
  theme_bw() +
  theme(axis.text.x = element_text(angle = 90)) +
  ylab("Levels of X") + xlab("Names of Y")
p
```
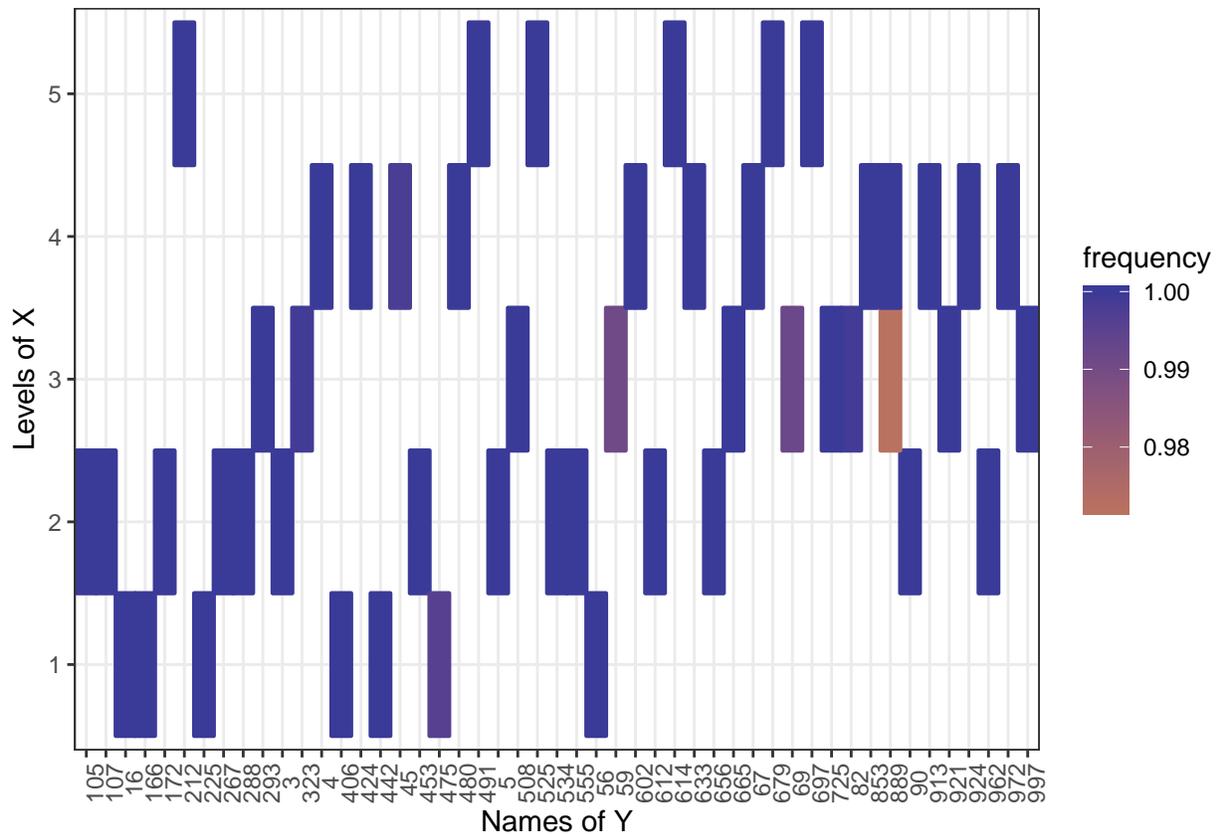
If we take a threshold of 0.95, meaning that we keep as non null values only the ones that are considered as non null in more than 95% of the times we have a True Positive Rate equal to 1 and a False Positive Rate equal to 0.

An exemple in metabolomics

In this section we study a LC-MS (Liquid Chromatography-Mass Spectrometry) data set made of African copals samples. The samples correspond to ethanolic extracts of copals produced by trees belonging to two genera Copaifera (C) and Trachylobium (T) with a second level of classification coming from the geographical provenance of the Copaifera samples (West (W) or East (E) Africa). Since all the Trachylobium samples come from East Africa, we can use the modeling proposed in (1) where X is a one-way ANOVA design matrix with 3 levels. Our goal is to identify the most important features (the m/z values) characterizing the different levels. In order to speed up the computation time, we decided to only study the 200 first columns of the responses matrix.

```
data("copals_camera")
Y <- scale(Y[, 1:200])
```

In order to analyze the whole responses matrix the last command line has to be replaced by

```
Y <- scale(Y)
```

We build the design matrix as follows

```
X <- model.matrix(~ group + 0)
```

We start by computing the residuals of the one-way ANOVA model.

4

```
E_hat <- residuals(lm(as.matrix(Y) ~ X + 0))
```

Then we test if the columns of the residuals are independent using the Portmanteau test.

```
whitening_test(E_hat)
```

```
## [1] 5.676735e-229
```

The *p*-value is smaller than 0.05 and thus the hypothesis that each row of E is a white noise is rejected. We try our different covariance modellings for the residuals and see if one manages to remove the dependence among the columns of the residuals matrix by using a Portmanteau test.

```
whitening_choice(E_hat, c("AR1", "nonparam", "ARMA"), pAR = 1, qMA = 1)
```

```
##             Pvalue      Decision
## AR1              0 NO WHITE NOISE
## nonparam     0.992    WHITE NOISE
## ARMA 1 1     0.653    WHITE NOISE
```

From this result, we observe that the non parametric modelling is the only one which removes the dependence.

In this application, the design matrix X is the design matrix of a one-way ANOVA. In that scenario we recommend to use the argument `group` = "your qualitative variable" in the `variable_selection` function. This argument will ensure that in the cross-validation the different fold are homogeneously distributed among the levels of the qualitative variable.

```
S12_inv <- whitening(E_hat, "nonparam", pAR = 1, qMA = 0)
Frequencies <-
  variable_selection(
    Y = Y,
    X = X,
    S12_inv,
    nb_repli = 1000,
    parallel = TRUE
  )
```

The following plot displays the selected coefficients in B that is the selected features (m/z values) characterizing the different levels when the threshold is equal to 0.999.

```
Frequencies$Names_of_Y <- as.numeric(gsub("X", "", Frequencies$Names_of_Y))

p <- ggplot(
  data = Frequencies[Frequencies$frequency >= 0.999, ],
  aes(x = Names_of_Y, y = Names_of_X, color = Names_of_X, fill = Names_of_X)
) +
  geom_point() + theme_bw() +
  theme(text = element_text(face = "bold", size = 12))
p
```
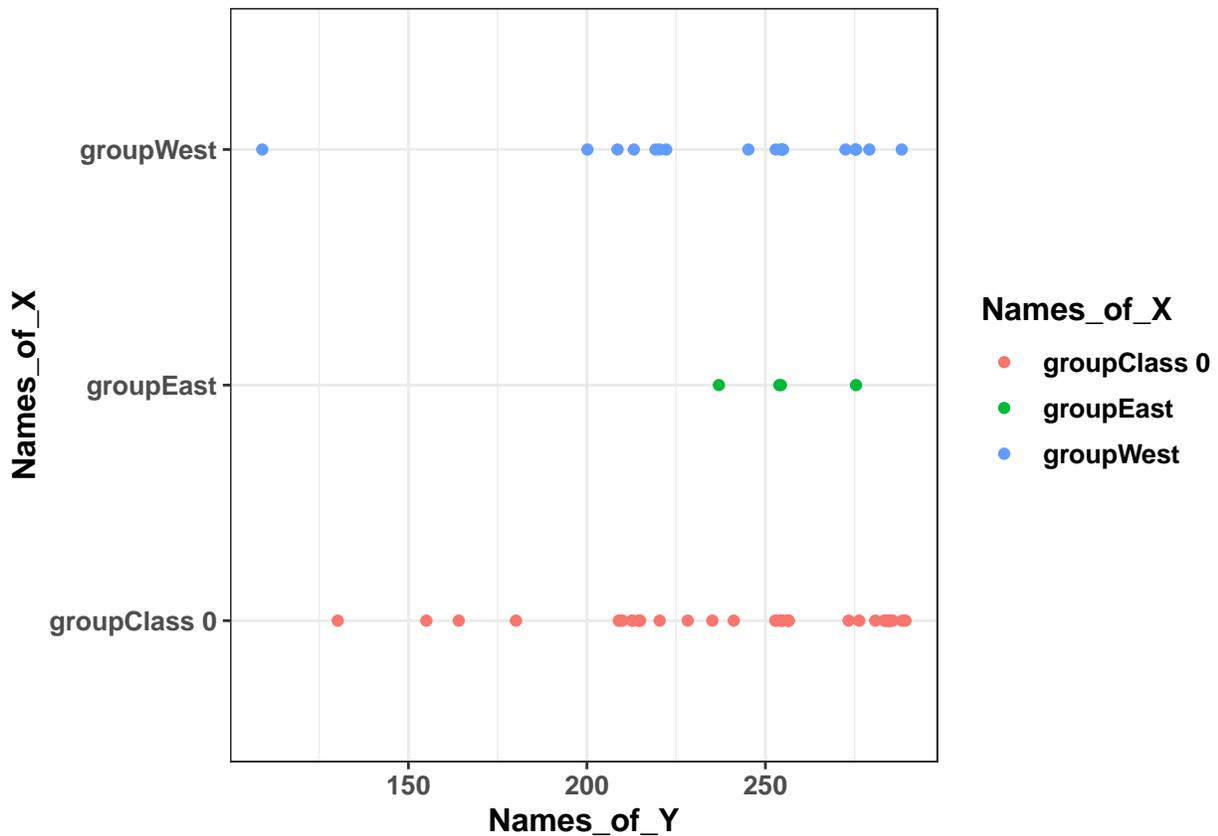
Session Info

```
## R version 3.5.2 (2018-12-20)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 16.04.6 LTS
##
## Matrix products: default
## BLAS: /usr/lib/libblas/libblas.so.3.6.0
## LAPACK: /usr/lib/lapack/liblapack.so.3.6.0
##
## locale:
##  [1] LC_CTYPE=fr_FR.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=fr_FR.UTF-8        LC_COLLATE=fr_FR.UTF-8
##  [5] LC_MONETARY=fr_FR.UTF-8    LC_MESSAGES=fr_FR.UTF-8
##  [7] LC_PAPER=fr_FR.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=fr_FR.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] parallel  stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
## [1] doMC_1.3.5       iterators_1.0.10  ggplot2_3.1.0     MultiVarSel_1.1.3
## [5] glmnet_2.0-16    foreach_1.4.4     Matrix_1.2-14
```

```
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_1.0.0        compiler_3.5.2  pillar_1.3.1     plyr_1.8.4
##  [5] bindr_0.1.1       tools_3.5.2     digest_0.6.18    evaluate_0.12
##  [9] tibble_2.0.1      gtable_0.2.0    lattice_0.20-35  pkgconfig_2.0.2
## [13] rlang_0.3.1       rstudioapi_0.9.0 yaml_2.2.0      xfun_0.4
## [17] bindrcpp_0.2.2    withr_2.1.2     stringr_1.3.1    dplyr_0.7.8
## [21] knitr_1.21        grid_3.5.2      tidyselect_0.2.4 glue_1.3.0
## [25] R6_2.3.0          rmarkdown_1.11  purrr_0.2.5      magrittr_1.5
## [29] scales_1.0.0      codetools_0.2-15 htmltools_0.3.6 assertthat_0.2.0
## [33] colorspace_1.4-0  labeling_0.3    stringi_1.2.4    lazyeval_0.2.1
## [37] munsell_0.5.0     crayon_1.3.4
```