Quick start guide for ‘bivas’ package

Mingxuan Cai [1], Mingwei Dai [2,4], Jingsi Ming [1],
Heng Peng [1], Jin Liu [3], and Can Yang [4]

[1] Department of Mathematics, Hong Kong Baptist University, Hong Kong.
[2] School of Mathematics and Statistics, Xi’an Jiaotong University, Xi’an, China.
[3] Centre of Quantitative Medicine, Duke-NUS Graduate Medical School, Singapore.
[4] Department of Mathematics, The Hong Kong University of Science and Technology, Hong Kong.

March 19, 2018

1 Overview

This vignette provides an introduction to the ‘bivas’ package. R package ‘bivas’ implements BIVAS model, which is an efficient statistical approach for bi-level variable selection. It provides model parameter estimation as well as statistical inference.

The package can be loaded with the command:

```
R> library("bivas")
```

This vignette is organized as follows. Section 2.1 discusses how to fit BIVAS in linear regression with group structure. Section 2.2 discusses how to fit BIVAS in multi-task learning. Section 2.3 explains command lines for statistical inference for identification of associated variables and groups using BIVAS.

Please feel free to contact Can Yang at macyang@ust.hk for any questions or suggestions regarding the ‘bivas’ package.

2 Workflow

R package bivas provides fitting function for both linear regression with group structure and multi-task learning; it is also designed with an option to execute prarallel computing. In this vignette, we use the simulated groupExample and mtExample in the package.

2.1 Fitting the BIVAS in linear regression with group structure

In this section, we use groupExample data as demonstration. We set the number of groups, variables and sample size to be $K = 100$, $p = 1,000$ and $n = 500$ respectively.

```
R> data(groupExample)
R> group <- groupExample$group
R> X <- groupExample$X
R> y <- groupExample$y
R> length(group)
```

```
[1] 1000
```

```
R> length(unique(group))
```

```
[1] 100
```

```
R> dim(X)
```

```
[1]  500 1000
```

```
R> length(y)
```

```
[1] 500
```

The data to be used are 'X', 'y' and 'group'. 'X' and 'y' are typical design matrix and response vector for linear regression; 'group' is a vector containing the group membership of each variable. The length of 'group' is assumed to be the same as the number of columns of matrix provided to 'X', indicating which group each variable belongs to.

Now we can fit BIVAS with the function:

```
R> fit_bivas <- bivas(y = y, X = X,group = group)
```

if the computation should be parallelized with 2 threads, we can provide the number of threads in an additional argument as follows:

```
R> fit_bivas <- bivas(y = y, X = X, group = group, coreNum = 2)
```

with this, the computation tasks are dynamically allocated to 2 threads in the fitting process.

The returned object is 'fit_bivas' of S3 class containing parameter estimation, posterior inclusion probability, posterior mean and evidence lower bound for each EM procedure. Statistical inference can be made using the command in 2.3.

To evaluate the model perfomance, use

```
R> perf_bivas <- perf.bivas(y = y, X = X, group = group, coreNum = 2, nfolds=10)
```

```
start cv process......... total 10 validation sets
1 -th validation set...
2 -th validation set...
3 -th validation set...
4 -th validation set...
5 -th validation set...
6 -th validation set...
7 -th validation set...
8 -th validation set...
9 -th validation set...
10 -th validation set...
```

```
R> str(perf_bivas)
```

```
List of 3
 $ cvm    : num 3.56
 $ cvsd   : num 0.606
 $ testErr: num [1:10] 3.61 3.4 3.65 3.54 4.19 ...
```

which returns a list of vectors containing the estimates of: cross-validation mean squared error, standard error of mse and the test error for each trial. By default we use 10 fold cross-validation ('nfolds=10').

2.2 Fitting the BIVAS in multi-task learning

For multi-task learning, we use mtExample as demonstration. We set the variables, number of tasks and sample sizes to be $K = 500$, $L = 3$ and $n_1 = 500$, $n_2 = 600$, $n_3 = 500$ respectively.

```
R> data(mtExample)
R> X <- mtExample$X
R> y <- mtExample$y
R> class(X)
```

```
[1] "list"
```

```
R> class(y)
```

```
[1] "list"
```

```
R> sapply(X,ncol)
```

```
[1] 500 500 500
```

```
R> length(X)
```

```
[1] 3
```

```
R> sapply(y,length)
```

```
[1] 500 600 500
```

where the data to be used are are stored in the lists '`X`' and '`y`'. The length of '`X`' is assumed to be the same as the length of '`y`'.

To fit the model, the function '`bivas_mt`' is used:

```
R> fit_bivas_mt <- bivas_mt(y = y, X = X)
```

Again, parallel computing can be called by the following:

```
R> fit_bivas_mt <- bivas_mt(y = y, X = X, coreNum = 2)
```

'`fit_bivas_mt`' is an S3 object containing parameter estimation, posterior inclusion probability, posterior mean and evidence lower bound for each EM procedure.

To evaluate the model perfomance, use

```
R> perf_bivas_mt <- perf.bivas_mt(y = y, X = X, coreNum = 2, nfolds=10)
```

```
start cv process......... total 10 validation sets
1 -th validation set...
2 -th validation set...
3 -th validation set...
4 -th validation set...
5 -th validation set...
6 -th validation set...
7 -th validation set...
8 -th validation set...
9 -th validation set...
10 -th validation set...
```

```
R> str(perf_bivas)

List of 3
 $ cvm    : num 3.56
 $ cvsd   : num 0.606
 $ testErr: num [1:10] 3.61 3.4 3.65 3.54 4.19 ...
```

which returns a list of vectors containing the estimates of: cross-validation mean squared error, standard error of mse and the test error for each trial. By default we use 10 fold cross-validation (‘`nfolds=10`’).

2.3 Statistical inference, variable selection and prediction

In this section we show how to make statistical inference based on the fitted ‘`bivas`’ object.

To obtain the posterior inclusion probability of groups and variables, use

```
R> pos_bivas <- getPos(fit_bivas,weight=T)
R> str(pos_bivas)

List of 2
 $ var_pos  : num [1:1000, 1] 0.00296 0.00296 0.00298 0.00296 0.00296 ...
 $ group_pos: num [1:100, 1] 0.0078 1 0.0131 0.00607 0.00898 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:100] "1" "2" "3" "4" ...
  .. ..$ : NULL

R> pos_bivas_mt <- getPos(fit_bivas_mt,weight=T)
R> str(pos_bivas_mt)

List of 2
 $ var_pos  : num [1:500, 1:3] 0.0114 0.0265 0.0112 0.026 0.0171 ...
 $ group_pos: num [1:500, 1] 0.937 0.912 0.901 0.903 0.901 ...
```

which returns a list of values between 0 and 1 indicating the posterior inclusion probabilities of groups and variables. If ‘`weight=F`’, the outcomes of all EM procedures will be returned unweighted by the lower bound. For multi-task BIVAS, the variable level results are stored in a K by L matrix.

To obtain the coefficient estimates, use

```
R> coef_bivas <- coef(fit_bivas,weight=T)
R> str(coef_bivas)

List of 2
 $ cov : num [1, 1] 6.41e-17
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr "Intercept"
  .. ..$ : NULL
 $ beta: num [1:1000, 1] -3.37e-05 1.92e-05 -5.93e-04 2.04e-04 -2.22e-04 ...

R> coef_bivas_mt <- coef(fit_bivas_mt,weight=T)
R> str(coef_bivas_mt)
```

```
List of 2
 $ cov : num [1, 1:3] 5.05 5.02 5
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr "Intercept"
  .. ..$ : NULL
 $ beta: num [1:500, 1:3] 0.000394 0.001997 -0.000347 -0.001942 0.001002 ...
```

which returns a list of vectors containing the estimates of fixed effects and random effects. If '`weight=F`', the outcomes of all EM procedures will be returned unweighted by the lower bound. For multi-task BIVAS, the variable level results are stored in a K by L matrix.

To obtain the fdr, apply generic function '`fdr`' to a fitted '`bivas`' object:

```
R> fdr_bivas <- fdr(fit_bivas,FDRset=0.05,control="global")
R> str(fdr_bivas)
```

```
List of 2
 $ FDR     : num [1:1000] 0 0 0 0 0 0 0 0 0 0 ...
 $ groupFDR: Named num [1:100] 0 1 0 0 0 0 0 0 0 0 ...
  ..- attr(*, "names")= chr [1:100] "1" "2" "3" "4" ...
```

```
R> fdr_bivas_mt <- fdr(fit_bivas_mt,FDRset=0.05,control="global")
R> str(fdr_bivas_mt)
```

```
List of 2
 $ FDR     : num [1:500, 1:3] 0 0 0 0 0 0 0 0 0 0 ...
 $ groupFDR: num [1:500] 1 1 0 0 0 0 0 0 0 1 ...
```

which returns a list of binary values indicating the relevant groups and variables. One can specify threshold in FDR control through the argument '`FDRset`' which is 0.05 in our case; and we allow both local ('`control="local"`') and global ('`control="global"`') FDR controls. For multi-task BIVAS, the variable level results are stored in a K by L matrix.

To predict with new data based on fitted model, use

```
R> y_hat <- predict(fit_bivas, X=groupExample$X)
R> str(y_hat)
```

```
 num [1:500, 1] 0.851 -1.257 -1.159 0.867 -2.581 ...
```

```
R> y_hat <- predict(fit_bivas_mt, X=mtExample$X)
R> str(y_hat)
```

```
List of 3
 $ : num [1:500, 1] 3.05 5.18 4.61 5.62 4.64 ...
 $ : num [1:600, 1] 6.13 5.42 3.36 4.9 5.42 ...
 $ : num [1:500, 1] 4.78 6.58 5.94 5.47 5.58 ...
```

which returns a vector of predicted values for group BIVAS or a list of predicted values for each task for multi-task BIVAS.

References

Mingxuan Cai, Mingwei Dai, Jingsi Ming, Heng Peng, Jin Liu and Can Yang. BIVAS: A scalable Bayesian method for bi-level variable selection with applications. 2017. Under review.