Supplementary Material

EBIC prerequisites

EBIC has two software requirements. First, the system needs to support C++11. This means that GCC 4.6 or newer compiler should be installed. Second, CUDA Toolkit needs to be installed. This limits the usage of the software to only those devices that support CUDA. In future we consider implementing either sequential version of the software, or one that could be run on devices that don't support CUDA.

Installation instructions

EBIC is implemented in C++11 and CUDA. Therefore, a GPU device that supports CUDA needs to be supported in order to run the software. Please follow installation instructions as well as required links are maintained at `https://github.com/EpistasisLab/ebic` in order to install and run the software.

Linux users

1. First, please confirm that your GPU can run CUDA.
2. Please follow *CUDA Installation Guide for Linux* in order to install CUDA.
3. After downloading EBIC the software, please change directory to *EBIC_DIR/ebic/ebic* and type *make*. An *ebic* file should be build and ready for use.

Root access is not required in order to run the software, but may be required to install CUDA.

Windows users

1. First, please confirm that your GPU can run CUDA.
2. Either Microsoft Visual Studio or MinGW need to be installed in order to use the software. Please follow the specific instructions for installing one of those libraries.
3. Please follow *CUDA Installation Guide for Microsoft Windows* in order to install CUDA Toolkit.
4. After downloading the software, please change directory to *EBIC_DIR/ebic/ebic* and type

```
nvcc -O3 -std=c++11 --expt-extended-lambda -Xcompiler -fopenmp --default-stream per-thread
     main.cxx evolutions.cxx dataIO.cxx ebic.cu -Llib -lcuda -o ebic
```

   An *ebic* binary should be build and be ready for use.

Input file

The input file for EBIC requires to have both row and column headers. The top-left header (specifying the row headers name) is optionally. EBIC supports different delimiters: the values in the file could be separated with either commas, tabs, spaces or semicolons. Mind that the identifiers of rows and columns may not have in their names any of those characters. For examples of the input files, please check input1.txt, input2.txt and input3.txt available in the repository.

Output files

By default, for the input file name "input.txt" three output files are built: **input.txt-blocks.txt**, **input.txt-res.txt** and **input.txt-bicl.txt**. If any of those files are redundant to your purposes, please comment corresponding lines in main.cxx that start with 'print'. Please mind that the numbering in EBIC starts with 0, not 1 (this applies both to the specific bicluster as well as indices of rows or columns).

In the first output file (**input-blocks.txt**) the lines starting with BC begin the section of each bicluster and are followed by bicluster identifier (starting from zero). The square brackets indicate the size of the bicluster (rows x columns) . The following lines contain the content of the bicluster: the row and column headers as well as the values extracted from the input matrix.

```
BC0000 [9 x 6]
Rows/Columns col4 col18 ...
row2  0.435 -0.214 ...
row3 -0.315  0.641 ...
...   ...    ...

BC0001  [14 x 6]
Rows/Columns col4 col7 ...
...
```

The **input.txt-res.txt** file starts with the number indicating the number of detected biclusters. Then, subsequent lines begin with word 'Bicluster' and in round brackets contain two lists with square brackets separated by a coma (',').

```
#N
Bicluster([4, 18 ... ],[2, 3, ...])
Bicluster([5, 10 ... ],[4, 7, ...])
```

The last file, **input.txt-r.txt** implements a structure required by *biclust::Biclust* class. The file begins with the line containing the name of the method. Biclusters are stored in three consecutive lines. The first one contains the size of each of biclusters (number of rows and number of columns). The next two lines contain respectively the headers of rows and columns.

```
EBIC
6 4
row_4 row_18 row_39 row_40 row_60 row_62
col_2 col_3 col_5 col_8
7 3
row_5 row_10 row_11 row_17 row_20 row_37 row_42
col_4 col_7 col_9
...
```

How to use the software

EBIC allows to tune the method to the preferences of the user. As of the version of *ebic-0.6* the following options are available:

```
  -h,--help                   Print this help message and exit
  -i,--input TEXT             input file
  -o,--output TEXT            output file
  -n,--iterations INT         number of iterations [default: 5000]
  -b,--biclusters INT         number of biclusters [100]
  -x,--overlap FLOAT          overlap threshold [0.75]
  -g,--gpus INT               number of gpus [1]
  -a,--approx FLOAT           approximate trends allowance [0.85]
  -t,--negative-trends INT    are negative trends enabled [1]
  -m,--missing-value          numeric representation of missing value [MAX]
  -l,--log                    is logging enabled [false]
```

Only a single flag is required in order to run the software (**-i** or alternatively **–input**). EBIC requires to specify an input file; the rest of the flags are set to their defaults. Please mind, that the order of the flags in the command is not important.

- **-i / –input** flag specifies the name of the input file for the analysis.
- **-o / –output** flag determines the base name of the output files. Three files will be created, with -res.txt, -blocks.txt and -r.txt suffixes.
- **-n / –iterations** flag determines how many iterations will be executed. In each iteration of the algorithm a new population of solutions for genetic algorithm is created.
- **-b / –biclusters** flag defines the expected number of biclusters to be returned by the algorithm. The algorithm may still return less solutions.
- **-x / –overlap** flag presents how much series of columns are allowed to overlap with each other. For example: 0.75 means that two solutions returned by the method may share 75% of columns. Row overlap can not be defined by the user.
- **-g / –gpus** flag indicates how many GPUs will be involved in computations. The dataset is split between this number GPUs.
- **-a / –approx** flag specifies the tolerance of approximate patterns. For example, setting this flag to 0.75 means that for each of the biclusters, in each row of the bicluster at least 75% of the values have to be order-preserving (i.e. each consecutive column in the bicluster needs to be greater or equal to the previous one). We don't recommend using values of this parameter below 0.75.
- **-t / –negative-trends** binary flag specifies if each of the biclusters may contain negative correlations of rows. If this value is equal to 1 both negative and positive correlations will be included, otherwise only positive correlations will be considered.
- **-m / –missing-value** flag specifies which value in the matrix will be regarded as a missing value. This means that all comparisons between this value and any other value within the same row will be reported as not satisfied, what impacts the number of rows that are reported within the bicluster. Consequently, the smaller number of rows, the worse quality of the bicluster.
- **-l / –log** flag doesn't require any additional parameters. Setting this flag creates a log of each population in which the actual list of best solutions is outputted.

In order to run EBIC with its default parameters please call the method in the following way (input1.txt is an exemplary dataset for analysis):

```
./ebic -i input1.txt
./ebic -i input1.txt -o output -g 2 -n 10 -b 20 -x 0.5 -a 0.9 -t 0 -l
```

The second line presents an example of using all parameters within the call. The algorithm will be executed on 2 GPUs for 10 iterations, and will return up to 20 biclusters, which columns would be allowed to overlap by no more than 50%. All trends that have at least 90% concordance will be included in each of the bicluster. Negative trends will be rejected. A log file will be generated.

Integration with R and Bioconductor

In this short tutorial we present how to run EBIC from and within R environment.

Our computations will be performed on a popular yeast matrix. The dataset could be loaded with the following command:

```
library(biclust)
data(BicatYeast)
```

In order to execute EBIC, we need to save the matrix to a file. We can use the following command.

```
filename<-'yeast.txt'
write.table(BicatYeast,filename,quote=FALSE, col.names=NA)
```

Notice $col.names = NA$ flag, which is used so that the input file for the algorithm had proper number of columns. We are now ready to launch the analysis using EBIC. We first specify the directory on the file system into which EBIC was downloaded and execute a system command that runs the algorithm:

```
ebic_dir <- "/path/to/ebic/source/executable/"
system(paste(ebic_dir, "ebic -i yeast.txt -a 1", sep="")) #this usually takes ~3-8 minutes
```

We have set EBIC to return only strictly increasing biclusters by setting $a = 1$ flag. Depending on the number of iterations, EBIC analysis may take some time. Usually the algorithms finishes calculations for 5000 iteration within 3-30 minutes for a datasets up to 50.000 rows and 500 columns. The running time depends on setting x parameter with overlap.

Once EBIC completes, three files are generated including an '-r.txt' file, which is required for integration with R and Bioconductor. In order to read the result into *biclust::Biclust* package, we use an *rqubic* library from Bioconductor which offers readBiclusterResults - an R function that is able to parse the file with results from EBIC:

```
library(rqubic)
res <- readBiclusterResults('yeast.txt-r.txt',featureNames=rownames(BicatYeast), sampleNames=colnames(BicatYeast),delimiter=" ")
res
```

The results are now loaded into R and stored in structure *res*. We may inspect the first bicluster with the following command:

```
bicluster(BicatYeast,res,1)

#$Bicluster1
#            cold_roots_6h salt_root_6h wounding_root_6h wounding_root_24h heat_root_3+3h diurnal_04h.CEL diurnal_08h.CEL diurnal_24h.CEL
ZEA_3h
#264000_at    1.59707080    2.324208     -0.425501760       0.09862368     -0.4841475     -0.6536112     -1.1762706     -1.2080357 -0.42803670
#252193_at    1.99678860    5.117588     -0.079501020       0.06999684     -0.6794046     -1.2252674     -1.4151881     -2.3086370 -0.29660810
#253571_at    0.42187786    4.959305     -0.026344342       0.28586784     -0.5404338     -0.9527876     -1.5250449     -2.0947077 -0.30241370
#255742_at    1.19138120    1.932916      0.120053165       0.22788242     -0.7667831     -0.9822087     -1.4628626     -2.0763514 -0.21945035
#251633_at    0.03153849    4.149996     -0.123590164       0.01615809     -0.1701270     -1.4212694     -2.1016731     -2.3159685 -0.14070208
#245250_at    2.25127770    4.194502      0.217316270       0.28666317     -0.3423099     -1.7095029     -1.7236987     -1.8591344 -0.26921123
#259428_at    1.62924670    1.972585     -0.070572585       0.21250096     -0.3945674     -1.3193880     -1.7259763     -2.1008236 -0.26233903
#253915_at    2.92084720    3.313050     -0.088005710       0.06964143     -0.5728155     -1.4633756     -1.7481755     -2.1463390 -0.44705543
#248448_at    3.71631150    5.079633     -0.001094412       0.07854024     -0.5721630     -0.7232005     -1.0010283     -1.3589410 -0.50274724
#263182_at    1.99260490    3.895353      0.344943580       0.71948530     -0.3811503     -1.2407196     -1.6393700     -2.2424464 -0.33631352
#258947_at    2.00640400    3.895451     -0.096331230       0.55195224     -0.2957309     -1.2812177     -1.9099077     -2.1223881 -0.26843318
#256526_at    1.18674040    4.131755     -0.130111750      -0.00091900     -0.4400624     -1.2023183     -1.6967698     -2.0904887 -0.26639244
#247493_at    0.83023460    4.540572     -0.030311307       0.33022240     -0.4628698     -1.0644313     -1.1842590     -1.9046550 -0.06436941
#245041_at    1.79784400    4.192309      0.656444600       0.88435060     -0.8313448     -1.2584816     -1.3312218     -1.7413750 -0.44494070
#256522_at    1.67509270    1.733241      0.108040270       0.79628050     -0.1260023     -0.8632171     -1.0366577     -1.2912163 -0.06708340
#264758_at    2.48915550    3.110371     -0.234716770       0.05246763     -0.5395883     -0.6690022     -0.7446511     -0.8484708 -0.35105840
#253643_at    2.60801340    3.991088      0.085853934       0.70709914     -0.3878010     -1.3019184     -1.3174548     -1.6750537 -0.28894870
#251507_at    1.49138650    3.700303      0.220464630       0.57720960     -0.2085061     -0.8555255     -0.9268599     -1.7650052 -0.09654838
#258606_at    2.06613660    3.577474     -0.059541310       0.27468148     -0.2598224     -0.5594633     -1.0369704     -1.1449980 -0.10853020
#247240_at    1.84211420    3.606873      0.108546610       0.43564418     -0.3235864     -0.6151341     -0.6151341     -0.8751234 -0.05110991
#267028_at    1.57848310    2.546768      0.147567780       0.15801702     -0.3219629     -0.8790145     -1.1985899     -1.3656100 -0.29762423
#246777_at    1.09011940    2.123252      0.281700100       0.40200030     -0.5196271     -0.6953127     -1.0172399     -1.3027053 -0.38921730
#261648_at    2.10104100    2.253659      0.060844470       0.32531124     -0.3792523     -0.8312557     -1.1069973     -1.1366501 -0.30982083
#251745_at    2.14155340    2.870430      0.181728210       0.31003857     -0.3955855     -1.1328663     -1.3926567     -1.8191866 -0.26330867
#265184_at    1.36044050    2.928029      0.221371170       0.59509360     -0.2575915     -0.6991627     -0.6991627     -0.9269479 -0.08180866
#258436_at    1.75998180    3.602573      0.160219920       0.43551674     -0.2297181     -0.2378955     -0.7973780     -0.8328173 -0.19046270
```

Now we can visualize the bicluster. A great tool for this is offered by *gplots* library and is called *heatmap.2*.

```
library(gplots)
heatmap.2(bicluster(BicatYeast,res,1)[[1]], Colv=FALSE, Rowv=FALSE,dendrogram="none",density.info=c("none"), trace="none", margins=c(10,10), srtCol=45)
```

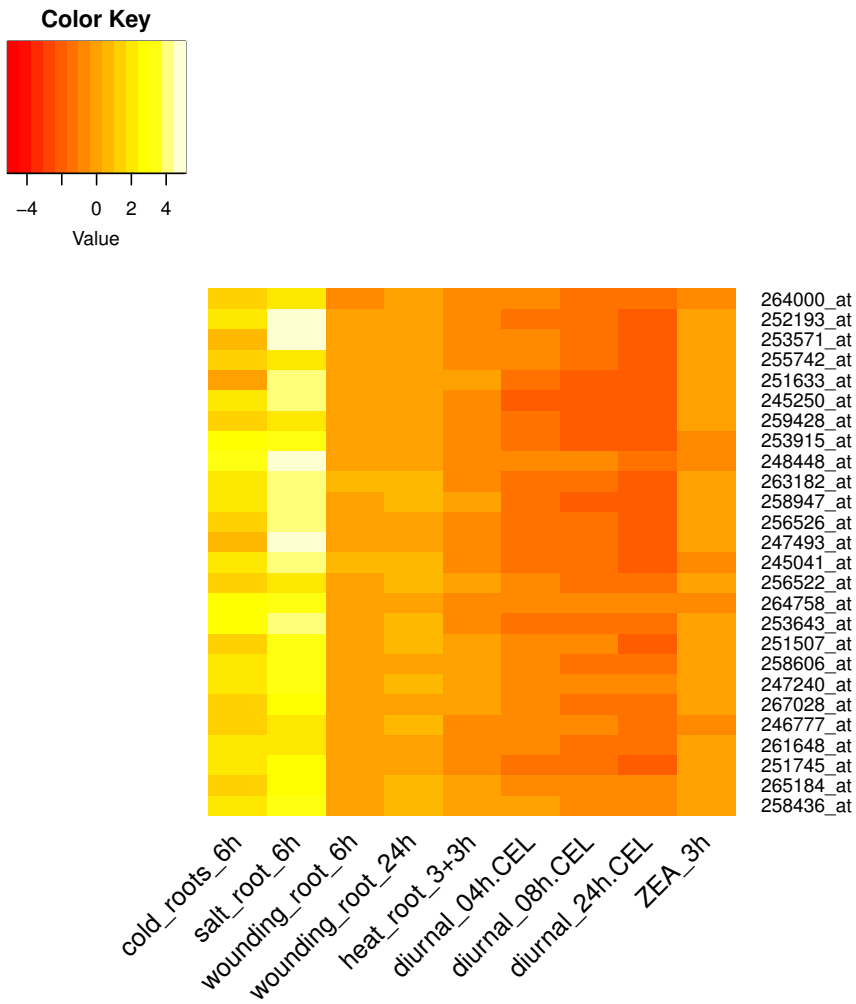

Fig. 1. Heatmap of the first bicluster.

The results could also be plotted using a parallel coordinates chart.

```
parallelCoordinates(BicatYeast, res, 1)
```

Both charts present monotonously increasing patterns found by EBIC on subset of dataset. Notice that gene expression profile is going to be reordered in .blocks file to reflect the monotonous relation.

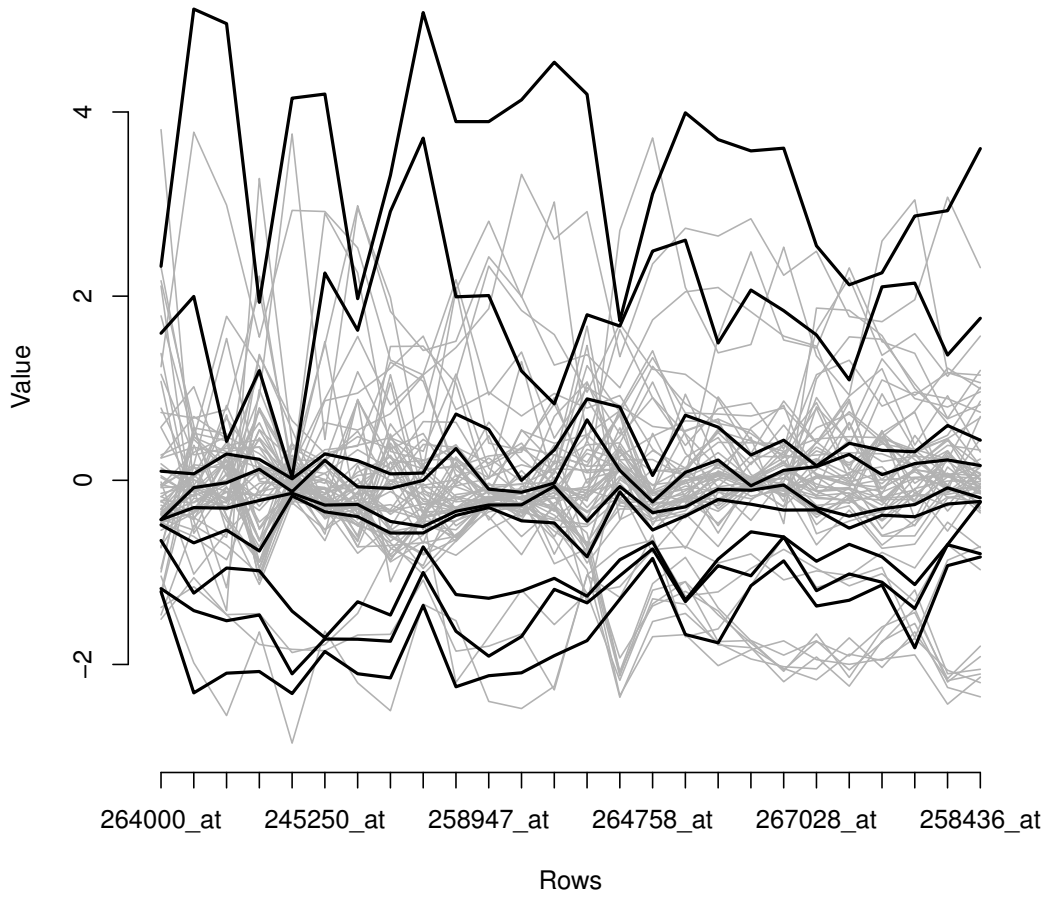

Fig. 2. Parallel coordinates with gene expression of the first bicluster.

We may also create a graph with row captions. We take advantage of *QUBIC* and *qgraph* libraries.

```
library(qgraph)
library(QUBIC)
net1 <- qunetwork(BicatYeast, res, number = 2, group = 2, method= "spearman")
# Extract edge coloring:
g <- qgraph(cbind(1:6,1:6,c(-0.9, -0.6, -0.3, 0.3, 0.6,0.9)), DoNotPlot=TRUE)
# Set colors and width of the edges
col <- g$graphAttributes$Edges$color
lwd <- g$graphAttributes$Edges$width

# Create a plot:
qgraph(net1[[1]], groups= net1[[2]], layout= "spring", minimum=0.6, color = cbind(rainbow(length(net1[[2]]) - 1), "gray", edge.label= FALSE))
# Add a legend:
legend(x=1.35,y=-0.02,title="Correlation value:",legend=c(-0.9, -0.6, -0.3, 0.3, 0.6,0.9),col=col,lwd=lwd,bty="n")
```

Solid green edges of the graph prove that all the genes (located in the nodes of the graph) are very highly correlated with each other according to Spearman's rho.

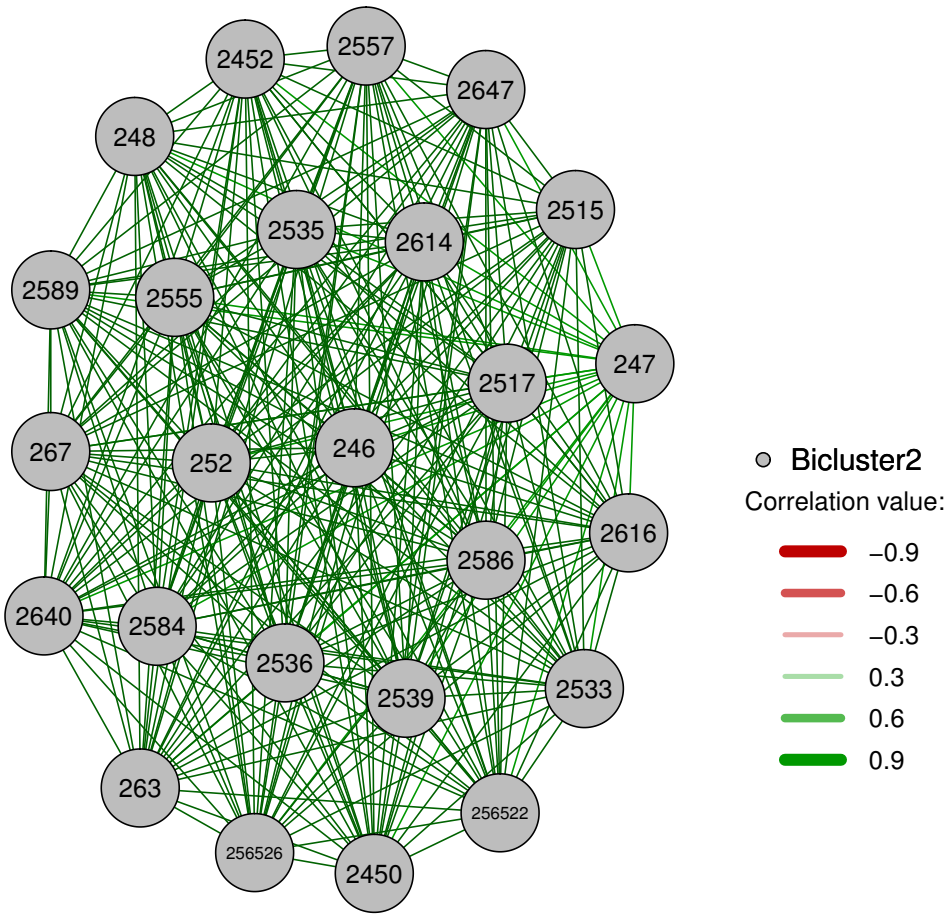

Fig. 3. Graph of correlations between the rows of the second bicluster

Methylation example

We start with loading all required libraries for annotaions, quering Gene Expression Omnibus, performing biclustering and gene enrichment analysis.

```
library(IlluminaHumanMethylation450kanno.ilmn12.hg19)
library(missMethyl)
library(org.Hs.eg.db)
library(GEOquery)
library(biclust)
library(rqubic)
library(GOstats)
```

We download a GSE dataset with methylation data. The dataset would normally need preprocessing and normalization in order to be suitable for the analysis. One of the options is to download raw data and perform preprocessing, the other is to use the data that has already been preprocessed and normalized.

```
# geo<-getGEO("GSE84493")
```

As the authors of the accession have already provided preprocessed dataset, we will base on their method of preprocessing (Dasen). We can either download full accession (a couple of GB of space), by executing:

```
#geo<-getGEOSuppFiles("GSE84493")
```

or simply a file in which we are interested in (please notive, that the size of the following file is over 1GB).

```
url='https://ftp.ncbi.nlm.nih.gov/geo/series/GSE84nnn/GSE84493/suppl//GSE84493_Matrix_processed.txt.gz?tool=geoquery'
download.file(url, 'GSE84493_Matrix_processed.txt.gz')
data <- read.table(gzfile('GSE84493/GSE84493_Matrix_processed.txt.gz'), row.names=1, header=T, sep="\t")
```

The downloaded matrix has alternating columns that contain processed values and their p-values. We're removing the columns corresponding to p-values, as they would bias our results. A filtering based on p-values and setting a missing value representative could also be used in here. We save the extracted data into a regular file in order to load it with EBIC. We set the required path to EBIC and perform analysis with the software:

```
data <- data[c(T,F)]
filename <- 'GSE84493.txt'
write.table(data, filename, quote=F, append=T, col.names=NA)
ebic_dir <- "/where/ebic/executable/is/located/ebic/"
system(paste(ebic_dir, "ebic -i ", filename, sep=""))
```

The results from EBIC are saved into a file, which is compatible with *Biclust* class from *biclust* package. Loading of the results to R is very simple:

```
res <- rqubic::readBiclusterResults(paste(filename,'-r.txt',sep=''), featurenames<-rownames(data), samplenames<-colnames(data), delimiter=' ')
```

The next step includes mapping rows of the matrix into genes.

```
annotation <- getAnnotation(IlluminaHumanMethylation450kanno.ilmn12.hg19)
allcpgs <- rownames(annotation)
universe=Lkeys(org.Hs.egSYMBOL)
```

Finally, we check if the results are bio-meaningful by performing gene set enrichment analysis using R package GOstats. We also perform

```
i <-1

#Uncomment the next line to validate all EBIC results (it takes time!):
#for (i in 1:res@Number) {

#We extract information from the i-th bicluster
  probes<-rownames(bicluster(data,res,i)[[1]])
  #We use missMethyl package to extract Entrez Gene IDs associated with the probes
  mappedEz <- getMappedEntrezIDs(probes,allcpgs,array.type="450K")
  #Only significant genes are interesting
  genes <- mappedEz$sig.eg

  #The next step is performing a hypergeometric test using the found genes and universe to find biological processes associated with genes
  params <- new("GOHyperGParams", geneIds = genes, universeGeneIds = universe, ontology = "BP", annotation = "org.Hs.eg.db")
  hgOver <- hyperGTest(params)
```

```
  #Correction for multiple testing using Benjamini–Hochberg procedure
  Pval<–p.adjust(pvalues(hgOver), method="BH",n = length(pvalues(hgOver)))

  #count number of GO terms that remained after application of a threshold
  enrichedNum<–length(which(Pval<0.05))

  #format the columns and present the results
  enr<–data.frame(summary(hgOver)[1:enrichedNum,c('GOBPID','Pvalue')],
                  Pval[1:enrichedNum], summary(hgOver)[1:enrichedNum,c('OddsRatio','ExpCount','Count','Size','Term')])
  colnames(enr)[3]="adj–Pvalue"
  print(enr)
  print("–––––––––")
# Uncomment the next line to validate all EBIC results:
#}
```

The results of validation of the first bicluster are presented below:

#	GOBPID	Pvalue	adj–Pvalue	OddsRatio	ExpCount	Count	Size	Term
#1	GO:0048856	9.574462e–15	5.474678e–11	2.047535	157.0756827	238	5447	anatomical structure development
#2	GO:0032502	4.078241e–14	1.165969e–10	2.003276	169.1007533	249	5864	developmental process
#3	GO:0044767	2.015559e–13	3.841656e–10	1.966351	166.5919256	244	5777	single–organism developmental process
#4	GO:0044707	8.625468e–13	1.233011e–09	1.926015	175.1565443	251	6074	single–multicellular organism process
#5	GO:0048731	2.526397e–11	2.617723e–08	1.894819	127.8925377	194	4435	system development
#6	GO:0007275	2.746823e–11	2.617723e–08	1.867349	144.0989878	212	4997	multicellular organism development
#7	GO:0007156	1.442068e–10	1.177964e–07	6.015916	4.5274247	23	157	homophilic cell adhesion via plasma membrane adhesion molecules
#8	GO:0007267	1.678146e–10	1.199455e–07	2.286643	44.6686676	89	1549	cell–cell signaling
#9	GO:0007399	2.706450e–09	1.678020e–06	1.999407	63.3551083	110	2197	nervous system development
#10	GO:0098742	2.934628e–09	1.678020e–06	4.476716	6.6325330	26	230	cell–cell adhesion via plasma–membrane adhesion molecules
#11	GO:0023052	8.527828e–09	4.432920e–06	1.690553	180.2030367	241	6249	signaling
#12	GO:0051239	9.815001e–09	4.676848e–06	1.882434	76.1587806	124	2641	regulation of multicellular organismal process
#13	GO:0044700	1.298385e–08	5.710899e–06	1.679440	180.0011770	240	6242	single organism signaling
#14	GO:0022610	2.503423e–08	1.022470e–05	2.158327	38.9300847	75	1350	biological adhesion
#15	GO:0045595	3.132664e–08	1.179066e–05	2.085759	43.6016949	81	1512	regulation of cell differentiation

We have also observed that multiple biclusters found by EBIC are significantly enriched in this methylation dataset.

Scripts for downloading datasets

The code presented below downloads from GEO datasets, which were used in this paper for presenting speedup of EBIC.

```
#GDS1490
library(GEOquery)
library(pcaMethods)
gdsname<-"GDS1490"
impname<-"svdImpute"
gds <- getGEO(gdsname)
eset <- GDS2eSet(gds)
data<-eset[rowSums(is.na(exprs(eset)))!=dim(eset)[2],]
pc <- pca(data, nPcs=5, method=impname)
dataImp<-asExprSet(pc,data)
write.table(exprs(dataImp), 'gds1490.txt', quote=F)

#GSE65194
library(GEOquery)
library(affy)
gse <- getGEO(gsename,destdir=".")
gsename<-"GSE65194"
getGEOSuppFiles(gsename)
untar(paste(gsename,"/",gsename,"_RAW.tar", sep=""), exdir=paste(gsename,"/","data", sep=""))
raf <- ReadAffy(celfile.path=paste(gsename,"/","data", sep=""))
rmaEset <- rma(raf)
write.table(exprs(rmaEset),file="gse65194.txt",append=FALSE,quote=FALSE)
```

Environment

```
> sessionInfo()
R version 3.5.0 (2018-04-23)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: CentOS Linux 7 (Core)

Matrix products: default
BLAS: /misc/appl/R-3.5.0/lib64/R/lib/libRblas.so
LAPACK: /misc/appl/R-3.5.0/lib64/R/lib/libRlapack.so

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
 [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

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

other attached packages:
 [1] rqubic_1.26.0         QUBIC_1.8.0           BiocInstaller_1.30.0
 [4] biclust_2.0.1         lattice_0.20-35       colorspace_1.3-2
 [7] MASS_7.3-50           hgu133plus2cdf_2.18.0 affy_1.58.0
[10] bindrcpp_0.2.2        pcaMethods_1.72.0     GEOquery_2.48.0
[13] Biobase_2.40.0        BiocGenerics_0.26.0

loaded via a namespace (and not attached):
 [1] modeltools_0.2-21     tidyselect_0.2.4      purrr_0.2.5
 [4] flexclust_1.3-5       stats4_3.5.0          blob_1.1.1
 [7] rlang_0.2.1           pillar_1.2.3          glue_1.2.0
[10] DBI_1.0.0             bit64_0.9-7           affyio_1.50.0
```

```
[13] bindr_0.1.1           plyr_1.8.4            zlibbioc_1.26.0
[16] munsell_0.5.0         gtable_0.2.0          memoise_1.1.0
[19] IRanges_2.14.10       curl_3.2              AnnotationDbi_1.42.1
[22] preprocessCore_1.42.0 Rcpp_0.12.17          readr_1.1.1
[25] scales_0.5.0          limma_3.36.2          S4Vectors_0.18.3
[28] additivityTests_1.1-4 bit_1.1-14            ggplot2_2.2.1
[31] hms_0.4.2             digest_0.6.15         stringi_1.2.3
[34] dplyr_0.7.5           tools_3.5.0           magrittr_1.5
[37] lazyeval_0.2.1        tibble_1.4.2          RSQLite_2.1.1
[40] tidyr_0.8.1           pkgconfig_2.0.1       xml2_1.2.0
[43] assertthat_0.2.0      R6_2.2.2              compiler_3.5.0
```