Supplementary Material - package runibic

The runibic biclustering algorithm is a parallel version of the original UniBic method. The algorithm implements very similar steps as its sequential predecessor (Wang et al., 2016):

1. Discretize the input matrix as described in (Wang et al., 2016). Determine the number of partitions using significance parameter (default: $\alpha = 0.05$) of the bicluster that is going to be identified. Set the number of partitions. By default we set $k$ to 4, as in the original method, we are interested in finding biclusters that contain at least 5 rows.
2. (executed in parallel) Create a matrix of indices $Y = y_{ij}$. Sort each of the rows of the input matrix and calculate:
   $y_{ij} = j$-th smallest entry in row $i$.
3. (executed in parallel) Apply Longest Common Subsequence (LCS) algorithm in order to identify seeds for the future biclusters. LCS algorithm is executed for each pair of the rows and parallelized using OpenMP; each time one of the rows is picked as the seed. Sort seeds in decreasing order based on the length of the seed.
4. (executed partially in parallel) Extend biclusters by adding strict order-preserving biclusters to the previously identified seeds. Add greedily as many rows as possible to each of the seed until the bicluster has more rows than columns, while maintaining its monotonously increasing order.
5. (executed partially in parallel) Extend biclusters to approximate ones, first by adding columns which have an error rate $r \leq 0.3$. Allow the negative trends as well as approximate ones (i.e. those in which over $t \geq 85\%$ of the trend is in the order specified by a seed row). Both columns and rows are added one at a time. The extension is made by calculating LCS between the extended seed and each of the rows (for negative trend the sequence is reverted). Remove seed from the list of potential solutions. Go back to Step 4 until the required number of biclusters is found or the list becomes empty.
6. Return nbics biclusters (up to a 100) and wrap them into biclust::Biclust class.

Implementation details

In the provided implementation of UniBic algorithm, we migrated the original code from C to C++11 programming language and added OpenMP support. Code refactoring allowed us to take advantage of multiple aspects of modern-style language programming:

- safer and more modern memory management replaced difficult to maintain C style memory allocations and deallocations,
- fast and efficient containers from Standard Template Library (STL), such as vectors, sets and algorithms, were used for acceleration of common operations like iterate, sort, search, count, or copy,
- the original implementation in most cases allocated a large number of simple arrays and used loops with slow indexing for common operations,
- removing many redundant copying and memory allocations,
- fixing a couple of memory leaks, which caused segmentation fault for some datasets.

Porting the code improved interpretability of the code allowed to remove multiple redundancies present in the previous UniBic implementation. For example, we replaced the original four functions that calculated the Longest Common Subsequence with a single one with multiple options. Similar improvements were made in other code sections, for example: in discretization, in calculation of Longest Common Subsequence between each pair of rows, in clustering and bicluster expansion parts. In order to provide more insightful analysis into the modules of UniBic algorithm, we separated and exported the major steps of the original method. Thus, the algorithm may be run using either a single command, or executed step by step. This provides much better control over the code, improves clarity of the method, and allows its future customization. The algorithm provided in runibic package is divided into the following sections:

- set_runibic_params - a function that sets the default parameters for algorithm,
- runDiscretize - the original UniBic discretize approach, which take into account the number of ranks from ‘div’ parameter (default value: 15) and quantile value from ‘q’ parameter (by default: div/number_of_columns),
- unisort - a function that sorts the rows of a matrix and returns the indexes of sorted columns in each row (Step 1),
- calculateLCS - a function that calculates the Longest Common Subsequence (LCS) between each unique pair of rows in the matrix, returns a list of LCS lengths and row pairs (Step 2),
- cluster - the main biclustering method which builds biclusters based on the input data and calculateLCS results (Steps 3-5).

By designing a modular structure of the package we intended to simplify flexible modifications of the original algorithm. Such methods may use different preprocessing or ranking techniques, or expand bicusters using different rows as seeds. An example includes different method of sorting results from calculateLCS. The proposed method, which is based on a stable STL sort, could be used as an alternative to the old C style pointer and sorting based on Fibonacci Heap. In our opinion the proposed method is more robust and better reflects the original intention. The choice of the method may implicate the outcome of the algorithm, as different LCSes of the same length may be chosen as seeds.

In order to improve the algorithm execution time the most crucial and computationally intensive parts of the code were parallelized using OpenMP standard. One of the most time consuming steps of UniBic is calculating Longest Common Subsequence (LCS) between unique pairs of rows. We rearranged the code and achieved parallelization where each core of the CPU calculates unique LCS between unique pair of rows simultaneously. Similarly, we also paralleled the data preprocessing required by the method, so as expansions of each of the biclusters, which required calculations of LCS between each row and the seed. All mentioned changes allowed us to obtain biclustering results in several minutes on the modern computer with modern processor.
Example: workflow with SummarizedExperiment

In this example we present how to use runibic package with SummarizedExperiment class that contains lists of assays. We start by loading the required packages:

```r
library(runibic)
library(SummarizedExperiment)
library(gplots)
```

We create a matrix with 1000 rows and 100 columns and implant inside it 4 upregulated biclusters of size 20x20. We intentionally sort each the values in each row of the biclusters in monotonously increasing order. We preliminary inspect the matrix using heatmap.2 function from gplots library. The results are presented in Fig. 2.

```r
set.seed(42)
m <- matrix( rnorm(1000*100,mean=0,sd=1), 1000, 100)
rownames(m) <- paste("row",1:1000, sep="_")
colnames(m) <- paste("col",1:100, sep="_")
for (k in 0:4) {
  for (i in 1:20) {
    bicl <- rnorm(20,mean=0,sd=1)+4
    m[i+k*20,1:20+k*20]=sort(bicl)
  }
}
heatmap.2(m, Colv= FALSE, Rowv = FALSE, dendrogram="none" , density.info="none" , trace="none" , margins=c(11,11), labRow = FALSE, labCol = FALSE)
```

![Fig. 1. A heatmap for the randomly initialized martix with 5 implanted biclusters.](image)

In the next step we build a SummarizedExperiment from an exemplary matrix. We assume that it has a single assay only in which matrix m is contained.

```r
se <- SummarizedExperiment(assays=list(m))
```

Our next step is to run runibic biclustering method. For the matrices that have less than 2000 rows, runibic algorithm in Bioconductor 3.6 requires to set two additional parameters q and div. Starting from Bioconductor 3.7, setting those parameters is obsolete. We also specify nbic, which is the number of required biclusters to be returned by the runibic method.

```r
```
# Run runibic on the toy dataset

# For Bioconductor 3.6
res1 <- runibic(m, q = 15/ncol(m), div = 15, nbic = 5)
res2 <- runibic(se, q = 15/ncol(m), div = 15, nbic = 5)

# Starting from Bioconductor 3.7:
res1 <- runibic(m, nbic = 5)
res2 <- runibic(se, nbic = 5)

# Inspect the results
res1

# An object of class Biclust

# Call:
# NULL

# Number of Clusters found: 5

# First 5 Cluster sizes:
# BC 1 BC 2 BC 3 BC 4 BC 5
# Number of Rows: 19 19 13 12 14
# Number of Columns: 13 13 17 17 14

We inspect the results by drawing a histogram for the first bicluster. Two calls are compared: either using an input matrix, or a `SummarizedExperiment` class.

# Extract the first bicluster either from matrix m or from SummarizedExperiment class:
heatmap.2(bicluster(m, res1, 1)[[1]], Colv = FALSE, Rowv = FALSE, dendrogram = "none", density.info = "none", trace = "none", margins = c(5, 5))
heatmap.2(bicluster(assays(se)[[1]], res2[[1]], 1)[[1]], Colv = FALSE, Rowv = FALSE, dendrogram = "none", density.info = "none", trace = "none", margins = c(5, 5))

---

**Fig. 2.** A heatmap for the examined bicluster.
Now we inspect the second bicluster from the `SummarizedExperiment`. As this class may contain multiple assays, it is important to refer to the requested assay. In our toy example the list contains a single assay, therefore we will use `[[1]]` to extract biclusters for this particular assay.

```
#extract the second bicluster from the SummarizedExperiment class
bicluster(assays(se)[[1]], res2[[1]], 2)
```

```
#Bicluster2
# col
61 col
63 col
64 col
66 col
68 col
69 col
71 col
72 col
74 col
76 col
78 col
79 col
80

#row
68 0.9035566 2.648049 2.280202 2.638208 3.094568 3.202966 3.483871 3.513828 3.882798 3.994085 4.583249 5.217786 5.238829
```

#notice that the same result could be obtained with the following command:  
```
bicluster(m, res1, 2)
```
Example: workflow with ExpressionSet

In the second example we show how to use and visualize the results of runibic on the real dataset GDS589, which could be downloaded using GEOquery package.

```r
library(runibic)
library(GEOquery)
library(pcaMethods)
library(QUBIC)
library(qgraph)
```

In the first lines we load all required libraries, i.e. our package runibic, GEOquery (in order to download the dataset from Gene Expression Omnibus), pcaMethods (for missing values imputation), QUBIC (in order to create gene interaction network), and qgraph to visualize the network in form of a graph.

We are ready to download the dataset from NCBI Gene Expression Omnibus (GEO) repository. We use GDS2eSet function in order to transform dataset to ExpressionSet. The dataset contains 8799 features and 122 samples. In order to prepare the matrix for runibic we apply llsImpute methods from pcaMethods package in order to impute missing values. Once the data is ready for the algorithm, we apply runibic procedure and analyze the results.

```r
gdsname = "GDS589"
arr <- getGEO(gdsname, destdir = "./")
eset <- GDS2eSet(arr, do.log2 = F)
eset

#ExpressionSet (storageMode: lockedEnvironment)
#

# element names: exprs
#
#protocolData: none
#
#sampleData:
#
# sampleNames: GSM15231 GSM15232 ... GSM15188 (122 total)
#
# varLabels: sample strain tissue description
#
# varMetadata: labelDescription
#
#featureData:
#
# featureNames: A01157cds_s ... Z96106_at (8799 total)
#
# fvarLabels: ID Gene title ... GO:Component ID (21 total)
#
# featureMetadata: Column labelDescription
#
#experimentData: use 'experimentData(object)'
#
# pubMedIds: 15990018
#
#Annotation:

#Extract expression data
array <- exprs(eset)

#Apply missing value imputation to the data
result <- pcaMethods::llsImpute(data.matrix(array[rowSums(is.na(array)) != ncol(array),],k=5))
array <- completeObs(result)
```

Please notice the usage of useLegacy = TRUE flag, which was introduced in the runibic package for Bioconductor 3.7. Without this flag, the approximate trends are added to each of the seeds according to the equation that originally appeared in UniBic implementation (1):

$$r = \lfloor cols(B) \ast t - 1 \rfloor$$

where $B$ is a given Bicluster, $cols$ is the number of its columns in the consistent order with the seed $t=0.85$ by default. Starting from Bioconductor 3.7, the following threshold is used for approximate trends (2):

$$r = \lfloor cols(B) \ast t \rfloor$$

In the previous version of the software, if the number of columns of the bicluster is equal to 4, all rows with at least 2 columns will be added to some narrow biclusters, what would unnecessarily inflate their sizes. From Bioconductor 3.7, only trends that have at least 3 out of 4 columns in the consistent order with the seed will be included.

We are ready to execute runibic procedure on the gene expression array.

```r
#Apply runibic to the expression data
#For Bioconductor 3.6:
res <- runibic(array)
#For Bioconductor 3.7:
res <- runibic(array, useLegacy=TRUE)
```
# Inspect the result

res

# An object of class Biclust

call:

# NULL

# Number of Clusters found: 100

# First 5 Cluster sizes:

#     BC 1 BC 2 BC 3 BC 4 BC 5

# Number of Rows:  2344 1011 867 696 525

# Number of Columns:  5  6  6  6  7

Runibic returned 100 biclusters. Some of them are very large and difficult to visualize, because of their sizes.

In our next step we visualize one of the results of biclustering with *runibic* using *qunetwork* function from *QUBIC* package. The function creates a networks of co-expressed genes based on the identified bicluster. The network is later visualized using *qgraph* function. The results are presented in Fig. 3. For the visualization purposes, we have chosen 31st bicluster, which has limited number of rows (i.e. less than 50). The vertices of the graph represent genes, and width and color of the edges represent the strength of correlations between the genes. Negative correlations are plotted in red color, positive in green.

```r
# Create a network of connections for the given bicluster
net1 <- qunetwork(array, res, number = 31, group = 31, method= "spearman")

# Extract edge coloring:
g <- qgraph(chbind(1:6,1:6,c(-0.9, -0.6, -0.3, 0.3, 0.6,0.9)), DoNotPlot=FALSE)

# 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 = chbind(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")
```

![Fig. 3. Graph representing the network of correlations for the 31st bicluster.](image-url)
You may notice how strongly correlated are some genes within the bicluster. Also gene with some weaker correlations are included.

We create a plot with parallel coordinates for the given bicluster in order to inspect the pattern of the bicluster. The results are presented in Fig. 4.

```r
#Show parallel coordinates plot of the same bicluster.
parallelCoordinates(array, res, 31)
```

![Parallel coordinates plot presenting gene expressions of 31st bicluster.](image)

We are interested to find if the results found in the previous step are bio-meaningful. Thus, we perform biological validation of the findings using the annotation of the dataset (rgu34a) as well as GOSTats package in order to perform enrichment analysis using hypergeometric test.

```r
#load required annotation
library(rgu34a.db)
library(GOstats)

#We extract the name of the genes that belong to a given bicluster.
#First we extract probe identifiers from biclust and map them into probe names
biclusterID=1
probeids <- row(matrix(res@RowxNumber[, biclusterID]))[res@RowxNumber[, biclusterID]==T]
probes <- rownames(array)[probeids]

#Next, we map genes to unique ENTREZ identifiers, keeping only the first name of the gene.
genes <- unique(mapIds(rgu34a.db, keys=as.character(probes), c("ENTREZID"), keytype="PROBEID", multiVals="first"))

#Similarly we map names of the probes in an array.
universe <- unique(mapIds(rgu34a.db, keys=as.character(rownames(array)), c("ENTREZID"), keytype="PROBEID", multiVals="first"))

#We prepare the required parameters for hypergeometric test and run the analysis
params <- new("GOHyperGParams", geneIds = genes, universeGeneIds = universe, ontology = "BP", annotation = "rgu34a.db")
hgOver <- hyperGTest(params)
```
During the procedure the following warning will appear, as the relation between probes and genes was not unique.

```r
# select () returned 1:many mapping between keys and columns
```

We would like to take a closer look at the findings of the algorithm, thus we inspect the result of the test.

```r
print (summary(hgOver))
```

<table>
<thead>
<tr>
<th>#</th>
<th>G O B P I D</th>
<th>Pvalue</th>
<th>OddsRatio</th>
<th>ExpCount</th>
<th>Count</th>
<th>Size</th>
<th>Term</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>GO:0002376 1.658316e−07</td>
<td>1.517021</td>
<td>265.215965</td>
<td>328</td>
<td>789</td>
<td>immune system process</td>
<td></td>
</tr>
<tr>
<td>#2</td>
<td>GO:0002684 3.660337−07</td>
<td>1.788777</td>
<td>111.599113</td>
<td>154</td>
<td>332</td>
<td>positive regulation of immune system process</td>
<td></td>
</tr>
<tr>
<td>#3</td>
<td>GO:0006669 4.675320e−07</td>
<td>1.926398</td>
<td>84.035477</td>
<td>121</td>
<td>250</td>
<td>lymphocyte activation</td>
<td></td>
</tr>
<tr>
<td>#4</td>
<td>GO:0008771 3.825497e−07</td>
<td>1.767646</td>
<td>112.271397</td>
<td>154</td>
<td>334</td>
<td>gland development</td>
<td></td>
</tr>
<tr>
<td>#5</td>
<td>GO:0008283 3.085573e−06</td>
<td>1.460346</td>
<td>284.039911</td>
<td>344</td>
<td>845</td>
<td>cell proliferation</td>
<td></td>
</tr>
<tr>
<td>#6</td>
<td>GO:0004292 1.741112e−06</td>
<td>1.646302</td>
<td>26.190687</td>
<td>319</td>
<td>780</td>
<td>homeostatic process</td>
<td></td>
</tr>
<tr>
<td>#7</td>
<td>GO:0002062 1.898996e−06</td>
<td>1.604551</td>
<td>155.297561</td>
<td>201</td>
<td>462</td>
<td>regulation of immune system process</td>
<td></td>
</tr>
<tr>
<td>#8</td>
<td>GO:0070482 2.506913e−06</td>
<td>1.756590</td>
<td>99.834146</td>
<td>157</td>
<td>378</td>
<td>response to oxygen levels</td>
<td></td>
</tr>
<tr>
<td>#9</td>
<td>GO:0001775 3.132966−06</td>
<td>1.652226</td>
<td>127.061641</td>
<td>168</td>
<td>378</td>
<td>cell activation</td>
<td></td>
</tr>
<tr>
<td>#10</td>
<td>GO:0068513 5.364295e−06</td>
<td>1.344049</td>
<td>492.784035</td>
<td>539</td>
<td>1268</td>
<td>animal organ development</td>
<td></td>
</tr>
<tr>
<td>#11</td>
<td>GO:0009605 5.999994e−06</td>
<td>1.380084</td>
<td>358.999557</td>
<td>419</td>
<td>1068</td>
<td>response to external stimulus</td>
<td></td>
</tr>
<tr>
<td>#12</td>
<td>GO:002694 6.874545e−06</td>
<td>1.916519</td>
<td>66.550698</td>
<td>96</td>
<td>238</td>
<td>regulation of leukocyte activation</td>
<td></td>
</tr>
<tr>
<td>#13</td>
<td>GO:0036293 7.897443e−06</td>
<td>1.741327</td>
<td>92.102882</td>
<td>126</td>
<td>274</td>
<td>response to decreased oxygen levels</td>
<td></td>
</tr>
<tr>
<td>#14</td>
<td>GO:0070371 8.163009e−06</td>
<td>2.056243</td>
<td>52.774279</td>
<td>79</td>
<td>187</td>
<td>ERK1 and ERK2 cascade</td>
<td></td>
</tr>
<tr>
<td>#15</td>
<td>GO:0006429 8.283770−06</td>
<td>1.544665</td>
<td>158.995122</td>
<td>202</td>
<td>473</td>
<td>epithelium development</td>
<td></td>
</tr>
<tr>
<td>#16</td>
<td>GO:0002682 1.898996e−06</td>
<td>1.604551</td>
<td>155.297561</td>
<td>201</td>
<td>462</td>
<td>regulation of immune system process</td>
<td></td>
</tr>
<tr>
<td>#17</td>
<td>GO:002694 6.874545e−06</td>
<td>1.916519</td>
<td>66.550698</td>
<td>96</td>
<td>238</td>
<td>response to oxygen levels</td>
<td></td>
</tr>
<tr>
<td>#18</td>
<td>GO:0009605 5.999994e−06</td>
<td>1.380084</td>
<td>358.999557</td>
<td>419</td>
<td>1068</td>
<td>response to external stimulus</td>
<td></td>
</tr>
<tr>
<td>#19</td>
<td>GO:0002062 1.898996e−06</td>
<td>1.604551</td>
<td>155.297561</td>
<td>201</td>
<td>462</td>
<td>regulation of immune system process</td>
<td></td>
</tr>
</tbody>
</table>

We have discovered that multiple GO terms are enriched within the result. We still need to adjust for multiple hypothesis testing. We perform Benjamini-Hochberg procedure for false discovery rate, as some of the previously detected p-values might have appeared by chance. We apply a cutoff of 0.05.

```r
# adjust p-values 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( 'G O B P I D', 'Pvalue' )] , Pval[1:enrichedNum], summary(hgOver)[1:enrichedNum, c( 'OddsRatio', 'ExpCount', 'Count', 'Size', 'Term' )])

# adj-Pvalue
```

<table>
<thead>
<tr>
<th>#</th>
<th>G O B P I D</th>
<th>Pvalue</th>
<th>OddsRatio</th>
<th>ExpCount</th>
<th>Count</th>
<th>Size</th>
<th>Term</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>GO:0002376 1.658316e−07</td>
<td>1.517021</td>
<td>265.215965</td>
<td>328</td>
<td>789</td>
<td>immune system process</td>
<td></td>
</tr>
<tr>
<td>#2</td>
<td>GO:0002684 3.660337−07</td>
<td>1.788777</td>
<td>111.599113</td>
<td>154</td>
<td>332</td>
<td>positive regulation of immune system process</td>
<td></td>
</tr>
<tr>
<td>#3</td>
<td>GO:0006669 4.675320e−07</td>
<td>1.926398</td>
<td>84.035477</td>
<td>121</td>
<td>250</td>
<td>lymphocyte activation</td>
<td></td>
</tr>
<tr>
<td>#4</td>
<td>GO:0008771 3.825497e−07</td>
<td>1.767646</td>
<td>112.271397</td>
<td>154</td>
<td>334</td>
<td>gland development</td>
<td></td>
</tr>
<tr>
<td>#5</td>
<td>GO:0008283 3.085573e−06</td>
<td>1.460346</td>
<td>284.039911</td>
<td>344</td>
<td>845</td>
<td>cell proliferation</td>
<td></td>
</tr>
<tr>
<td>#6</td>
<td>GO:0004292 1.741112e−06</td>
<td>1.646302</td>
<td>26.190687</td>
<td>319</td>
<td>780</td>
<td>homeostatic process</td>
<td></td>
</tr>
<tr>
<td>#7</td>
<td>GO:0002062 1.898996e−06</td>
<td>1.604551</td>
<td>155.297561</td>
<td>201</td>
<td>462</td>
<td>regulation of immune system process</td>
<td></td>
</tr>
<tr>
<td>#8</td>
<td>GO:0070482 2.506913e−06</td>
<td>1.756590</td>
<td>99.834146</td>
<td>157</td>
<td>378</td>
<td>response to oxygen levels</td>
<td></td>
</tr>
<tr>
<td>#9</td>
<td>GO:0001775 3.132966−06</td>
<td>1.652226</td>
<td>127.061641</td>
<td>168</td>
<td>378</td>
<td>cell activation</td>
<td></td>
</tr>
<tr>
<td>#10</td>
<td>GO:0068513 5.364295e−06</td>
<td>1.344049</td>
<td>492.784035</td>
<td>539</td>
<td>1268</td>
<td>animal organ development</td>
<td></td>
</tr>
<tr>
<td>#11</td>
<td>GO:0009605 5.999994e−06</td>
<td>1.380084</td>
<td>358.999557</td>
<td>419</td>
<td>1068</td>
<td>response to external stimulus</td>
<td></td>
</tr>
<tr>
<td>#12</td>
<td>GO:002694 6.874545e−06</td>
<td>1.916519</td>
<td>66.550698</td>
<td>96</td>
<td>238</td>
<td>regulation of leukocyte activation</td>
<td></td>
</tr>
<tr>
<td>#13</td>
<td>GO:0036293 7.897443e−06</td>
<td>1.741327</td>
<td>92.102882</td>
<td>126</td>
<td>274</td>
<td>response to decreased oxygen levels</td>
<td></td>
</tr>
<tr>
<td>#14</td>
<td>GO:0070371 8.163009e−06</td>
<td>2.056243</td>
<td>52.774279</td>
<td>79</td>
<td>187</td>
<td>ERK1 and ERK2 cascade</td>
<td></td>
</tr>
<tr>
<td>#15</td>
<td>GO:0006429 8.283770−06</td>
<td>1.544665</td>
<td>158.995122</td>
<td>202</td>
<td>473</td>
<td>epithelium development</td>
<td></td>
</tr>
<tr>
<td>#16</td>
<td>GO:0002062 1.898996e−06</td>
<td>1.604551</td>
<td>155.297561</td>
<td>201</td>
<td>462</td>
<td>regulation of immune system process</td>
<td></td>
</tr>
<tr>
<td>#17</td>
<td>GO:0009605 5.999994e−06</td>
<td>1.380084</td>
<td>358.999557</td>
<td>419</td>
<td>1068</td>
<td>response to external stimulus</td>
<td></td>
</tr>
<tr>
<td>#18</td>
<td>GO:0002062 1.898996e−06</td>
<td>1.604551</td>
<td>155.297561</td>
<td>201</td>
<td>462</td>
<td>regulation of immune system process</td>
<td></td>
</tr>
</tbody>
</table>

We have discovered that multiple GO terms are enriched within the result. We still need to adjust for multiple hypothesis testing. We perform Benjamini-Hochberg procedure for false discovery rate, as some of the previously detected p-values might have appeared by chance. We apply a cutoff of 0.05.
Example: comparison with multiple other methods implemented in R

In this example we present the comparison of our method with multiple other biclustering algorithms available at Bioconductor that support `Biclust` class interface. We start with loading all required libraries.

```r
library(runibic)
library(QUBIC)
library(isa2)
library(BiBitR)
library(biclust)
data(BicatYeast)
```

Now, we perform biclustering using multiple aforementioned biclustering method and extract Biclust object:

```r
resCC <- biclust::biclust(BicatYeast, method = R(BC))
resBimax <- biclust::biclust(BicatYeast, method = R(BCBimax))
resQuest <- biclust::biclust(BicatYeast, method = R(Q))
resBiBit <- BiBitR::BiBitWorkflow(matrix = binarize(BicatYeast), minr = 50, minc = 5, noise = 0.2)$Biclust
```

Some charts will pop up during the previous analysis, as they are default for BiBit package workflow. Finally, we inspect the summary of the biclustering results using `showinfo` command from `QUBIC` package.

```r
QUBIC::showinfo(BicatYeast, c(resCC, resBimax, resQuest, resBiBit, resPlaid, resXmotifs, resUnibic))
```
Biclustering packages in Bioconductor (3.6)

Bioconductor in version 3.6 provides implementation of the following biclustering algorithms:

- ISA (Bergmann et al., 2003) - implemented in eisa and isa2 Bioconductor packages (Csardi et al., 2010),
- CC (Cheng and Church, 2000), Plaid methods (Lazzeroni and Owen, 2002), Bimax (Prelić et al., 2006), xMotifs (Murali and Kasif, 2003), Quest (Kaiser, 2011), Spectral Cluger et al. (2003) - all available in biclust package (Kaiser et al., 2015),
- FABIA, FABIAS, FABIAP - available in Bioconductor package fabia (Hochreiter et al., 2010),
- HapFABIA - implemented in package hapFabia (Hochreiter, 2013)
- QUBIC (Li et al., 2009) - implemented in more modern package QUBIC (Zhang et al., 2017) and older package qubic (Zhang, 2015),
- MChicest - available in Bioconductor package MChicest (Bentham, 2017),
- SVD (Lee et al., 2010) and S4VD (Sill et al., 2011) - available in Bioconductor package S4VD (Sill and Kaiser, 2015),
- Iterative Binary Biclustering of Gene sets - available in Bioconductor package iBBiG (Gusenleitner et al., 2012),
- Biclustering Analysis and Results Exploration - available in package BicARE (Gestraud, 2008),
- Biclustering Algorithm for extracting bit-patterns from binary data-sets, available in package BiBitR ()

Session Info

Below is information about the session under which the previous code was generated.

```r
> sessionInfo()

R version 3.4.4 (2018-03-15)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 16.04.4 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=en_US.UTF-8
[2] LC_NUMERIC=C
[3] LC_TIME=en_US.UTF-8
[4] LC_COLLATE=en_US.UTF-8
[5] LC_MONETARY=en_US.UTF-8
[6] LC_MESSAGES=en_US.UTF-8
[7] LC_PAPER=en_US.UTF-8
[8] LC_NAME=C
[9] LC_ADDRESS=C
[10] LC_TELEPHONE=C
[12] LC_IDENTIFICATION=C

attached base packages:
[1] parallel stats4 grid stats graphics grDevices utils datasets methods
[10] base

other attached packages:
[1] fabia_2.24.0 BiBitR_0.3.1 isa2_0.3.5
[4] QUBIC_1.6.0 runlibc_1.0.2 SummarizedExperiment_1.8.1
[7] DelayedArray_0.4.1 matrixStats_0.52.2 Biobase_2.38.0
[10] GenomicRanges_1.30.3 GenomeInfoDb_1.14.0 IRanges_2.12.0
[13] S4Vectors_0.16.0 Bioconductor_3.24.0 biclust_2.0.0
[16] lattice_0.20-35 colorspace_1.3-2 MASS_7.3-47

loaded via a namespace (and not attached):
[1] mclust_5.4 Rcpp_0.12.17 flexclust_1.3-5 mvtnorm_1.0-7
[5] tidyverse_0.8.1 class_7.3-14 VGAM_1.1.5 R6_2.2.2
[9] plyr_1.8.4 ggplot2_2.2.1 pillar_1.2.2 zlibbioc_1.24.0
[13] rlang_0.2.0 lazyeval_0.2.1 dplyr_0.7.5-7 curl_3.0
[17] whisker_0.3-2 kernlab_0.9-25 Matrix_1.2-11 randomcolor_1.1.0
[21] additivityTests_1.1-4 Rsne_0.13 stringr_1.2.0 foreign_0.8-69
[25] RCurl_1.95-4.8 munsell_0.4.3 compiler_3.4.4 nnet_7.3-12
[29] tibble_1.4.2 gridExtra_2.3 GenomeInfoDBData_0.99.1 dendscendt_1.7.0
[33] viridisLite_0.2.0 bitops_1.0-6 jsonlite_1.5 gtable_0.2.0
[37] magrittr_1.5 scales_0.5.0 stringi_1.2.2 XVector_0.18.0
[41] viridis_0.4.0 flexmix_2.3-14 testthat_1.0.0 robustbase_0.92-8
[45] tools_3.4.4 fpc_2.1-11 glue_1.2.0 trimcluster_0.1-2
[49] Biobase_2.38.0 purrr_0.2.4 cluster_2.0.6 prabclus_2.2-6
[53] modeltools_0.2-21
```
References


