Package 'QUBIC'

December 31, 2024

Type Package

Title An R package for qualitative biclustering in support of gene co-expression analyses

Description The core function of this R package is to provide the implementation of the well-cited and well-reviewed QUBIC algorithm, aiming to deliver an effective and efficient biclustering capability. This package also includes the following related functions: (i) a qualitative representation of the input gene expression data, through a well-designed discretization way considering the underlying data property, which can be directly used in other biclustering programs; (ii) visualization of identified biclusters using heatmap in support of overall expression pattern analysis; (iii) bicluster-based co-expression network elucidation and visualization, where different correlation coefficient scores between a pair of genes are provided; and (iv) a generalize output format of biclusters and corresponding network can be freely downloaded so that a user can easily do following comprehensive functional enrichment analysis (e.g. DAVID) and advanced network visualization (e.g. Cytoscape).

VignetteBuilder knitr

biocViews StatisticalMethod, Microarray, DifferentialExpression, MultipleComparison, Clustering, Visualization, GeneExpression, Network

Version 1.34.0

License CC BY-NC-ND 4.0 + file LICENSE

Depends R (>= 3.1), biclust

Imports Rcpp (>= 0.11.0), methods, Matrix

LinkingTo Rcpp, RcppArmadillo

Suggests QUBICdata, qgraph, fields, knitr, rmarkdown

SystemRequirements C++11, Rtools (>= 3.1)

Enhances RColorBrewer

URL http://github.com/zy26/QUBIC

BugReports http://github.com/zy26/QUBIC/issues

git_url https://git.bioconductor.org/packages/QUBIC

git_branch RELEASE_3_20

git_last_commit 4c4d83e

git_last_commit_date 2024-10-29

Repository Bioconductor 3.20

Date/Publication 2024-12-30

Author Yu Zhang [aut, cre],
Qin Ma [aut]

Maintainer Yu Zhang <zy26@jlu.edu.cn>

Contents

`BCQU-class` *Class BCQU.*

Description

Class `BCQU` define a QUalitative BIClustering calcuator.

See Also

`BCQU` `qudiscretize` `qunetwork` `qunet2xml` `biclust`

`QUBIC` *QUBIC: A Qualitative Biclustering Algorithm for Analyses of Gene Expression Data*

Description

`QUBIC` is a biclustering package, with source code upgrading from C code to C++ code. The updated source code can avoid memory allocation error and is much efficient than the original one. Based on our preliminary analysis, it can save 40% running time on a plant microarray data. Whenever using this package, please cite as Yu Zhang, Juan Xie, Jinyu Yang, Anne Fennell, Chi Zhang, Qin Ma; QUBIC: a bioconductor package for qualitative biclustering analysis of gene co-expression data. *Bioinformatics*, 2017; 33 (3): 450-452. doi: 10.1093/bioinformatics/btw635

`BCQUD` performs a QUalitative BIClustering for a discret matrix.

Usage

```
## S4 method for signature 'matrix,BCQU'
biclust(x, method = BCQU(),
        r = 1, q = 0.06,
        c = 0.95, o = 100, f = 1,
        k = max(ncol(x) %/% 20, 2),
        type = 'default', P = FALSE, C = FALSE, verbose = TRUE,
        weight = NULL, seedbicluster = NULL)

## S4 method for signature 'matrix,BCQUD'
biclust(x, method = BCQUD(),
        c = 0.95, o = 100, f = 1,
        k = max(ncol(x) %/% 20, 2),
        type = 'default', P = FALSE, C = FALSE, verbose = TRUE,
        weight = NULL, seedbicluster = NULL)

qubiclust_d(x, c = 0.95, o = 100, f = 1,
        k = max(ncol(x) %/% 20, 2),
        type = 'default', P = FALSE, C = FALSE, verbose = TRUE,
        weight = NULL, seedbicluster = NULL)

qubiclust(x, r = 1L, q = 0.06, c = 0.95, o = 100, f = 1,
        k = max(ncol(x) %/% 20, 2),
        type = 'default', P = FALSE, C = FALSE, verbose = TRUE,
        weight = NULL, seedbicluster = NULL)
```

Arguments

x	the input data matrix, which could be the normalized gene expression matrix or its qualitative representation from Qdiscretization or other discretization ways. (for example: a qualitative representation of gene expression data) For `BCQU()`, the data matrix should be real For `BCQUD()`, the data matrix should be discretized as integer. Zeros in the matrix will be treated as non-relevant value.
r	Affect the granularity of the biclusters. The range of possible ranks. A user can start with a small value of `r` (the default value is `1` so the corresponding data matrix consists of values '`1`', '`-1`' and '`0`'), evaluate the results, and then use larger values (should not be larger than half of the number of the columns) to look for fine structures within the identified biclusters.
q	Affect the granularity of the biclusters. The percentage of the regulating conditions for each gene. The choice of `q`'s value depends on the specific application goals; that is if the goal is to find genes that are responsive to local regulators, we should use a relatively small q-value; otherwise we may want to consider larger q-values. The default value of `q` is `0.06` in QUBIC (this value is selected based on the optimal biclustering results on simulated data).
c	The required consistency level of a bicluster. The default value of `c` is `0.95`
o	The number of output biclusters. `o`'s default value is `100`.
f	Control parameter, to control the level of overlaps between to-be-identified biclusters. The filter cut-off for data post-processing. For overlaps among to-be-identified biclusters. Its default value is set to `1` to ensure that no two reported biclusters overlap more than `f`.

k	The minimum column width of the block, minimum `max(ncol(x) %/% 20, 2)` columns.
type	The constrain type. If `type` is omitted or `type='default'`, the original objective function in QUBIC will be used, which is to maximize the minimal value of numbers of rows and columns. If `type='area'`, the program tries to identify the bicluster with the maximal value of number of rows multiplied by number of columns. Other types are reserved for future use.
P	The flag to enlarge current bicluster using a *p*-value contrain, which is defined based on its significance of expression consistency comparing to some simulated submatrix. Default: `FALSE`.
C	The flag to set the lower bound of the condition number in a bicluster as 5% of the total condition number in the input data. Only suggested to use when the input data has a few conditions (e.g. less than `20`). Default: `FALSE`.
verbose	If 'TRUE', prints extra information on progress.
weight	Alternative weight matrix provided by user, will append to default weight. `o`, `f`, `k`, `P`, `type`, `C` will be ignored if using this parameter.
seedbicluster	Seed provided by user, normally should be a result of function `biclust`.
method	`BCQU()` or `BCQUD()`, to perform QUBIC algorithm

Details

For a given representing matrix of a microarray data set, we construct a weighted graph G with genes represented as vertices, edges connecting every pair of genes, and the weight of each edge being the similarity level between the two corresponding (entire) rows. Clearly, the higher a weight, the more similar two corresponding rows are. Intuitively, genes in a bicluster should induce a heavier subgraph of G because under a subset of the conditions, these genes have highly similar expression patterns that should make the weight of each involved edge heavier, comparing to the edges in the background. But it should be noted that some heavy subgraph may not necessarily correspond to a bicluster, i.e. genes from a heavy subgraph may not necessarily have similar expression patterns because different edges in a subgraph may have heavier weights under completely different subsets of conditions. It should also be noted that recognizing all heavy subgraphs in a weighted graph itself is computationally intractable because identification of maximum cliques in a graph is a special case of this, and the maximum clique problem is a well known intractable problem (NP-hard). So in our solution, we do not directly solve the problem of finding heavy subgraphs in a graph. Instead, we built our biclustering algorithm based on this graph representation of a microarray gene expression data, and tackle the biclustering problem as follows. We find all feasible biclusters (I,J) in the given data set such that min{|I|, |J|} is as large as possible, where I and J are subsets of genes and conditions, respectively.

Value

Returns an Biclust object, which contains bicluster candidates

Functions

- `BCQU`: Performs a QUalitative BIClustering.
- `BCQUD`: Performs a QUalitative BIClustering for a discret matrix.
- `qubiclust_d`: Performs a QUalitative BIClustering for a discret matrix.
- `qubiclust`: Performs a QUalitative BIClustering.

References

Yu Zhang, Juan Xie, Jinyu Yang, Anne Fennell, Chi Zhang, Qin Ma; QUBIC: a bioconductor package for qualitative biclustering analysis of gene co-expression data. *Bioinformatics*, 2017; 33 (3): 450-452.

See Also

BCQU-class qudiscretize qunetwork qunet2xml biclust

Examples

```
# Random matrix with one embedded bicluster
test <- matrix(rnorm(5000), 100, 50)
test[11:20, 11:20] <- rnorm(100, 3, 0.3)
res <- biclust::biclust(test, method = BCQU())
summary(res)
show(res)
names(attributes(res))

## Not run:
# Load microarray matrix
data(BicatYeast)

# Display number of column and row of BicatYeast
ncol(BicatYeast)
nrow(BicatYeast)
# Bicluster on microarray matrix
system.time(res <- biclust::biclust(BicatYeast, method = BCQU()))

# Show bicluster info
res
# Show the first bicluster
biclust::bicluster(BicatYeast, res, 1)
# Get the 4th bicluster
bic4 <- biclust::bicluster(BicatYeast, res, 4)[[1]]

# or
bic4 <- biclust::bicluster(BicatYeast, res)[[4]]
# Show rownames of the 4th bicluster
rownames(bic4)
# Show colnames of the 4th bicluster
colnames(bic4)


## End(Not run)
## Not run:
# Bicluster on selected of genes
data(EisenYeast)
genes <- c("YHR051W", "YKL181W", "YHR124W", "YHL020C", "YGR072W", "YGR145W",
    "YGR218W", "YGL041C", "YOR202W", "YCR005C")
# same result as res <- biclust::biclust(EisenYeast[1:10,], method=BCQU())
res <- biclust::biclust(EisenYeast[genes, ], method = BCQU())
res


## End(Not run)
```

```
## Not run:
# Get bicluster by row name = 249364_at
biclust::bicluster(BicatYeast, res, which(res@RowxNumber[which(rownames(BicatYeast) ==
    "249364_at"), ]))


## End(Not run)
## Not run:
# Get bicluster by col name = cold_roots_6h
biclust::bicluster(BicatYeast, res, which(res@NumberxCol[, which(colnames(BicatYeast) ==
    "cold_roots_6h")]))


## End(Not run)
## Not run:
# Draw a single bicluster using drawHeatmap {bicust}
data(BicatYeast)
res <- biclust::biclust(BicatYeast, BCQU(), verbose = FALSE)
# Draw heatmap of the first cluster
biclust::drawHeatmap(BicatYeast, res, 1)


## End(Not run)
## Not run:
# Draw a single bicluster using heatmap {stats}
data(BicatYeast)
res <- biclust::biclust(BicatYeast, BCQU(), verbose = FALSE)
bic10 <- biclust::bicluster(BicatYeast, res, 10)[[1]]

# Draw heatmap of the 10th cluster using heatmap {stats}
heatmap(as.matrix(t(bic10)), Rowv = NA, Colv = NA, scale = 'none')

# Draw heatmap of the 10th cluster using plot_heatmap {phyloseq}
if (requireNamespace('phyloseq'))
    phyloseq::plot_heatmap(otu_table(bic10, taxa_are_rows = TRUE))


## End(Not run)
## Not run:
# Draw a single bicluster with original data background and color options
data(BicatYeast)
res <- biclust::biclust(BicatYeast, BCQU(), verbose = FALSE)
palette <- colorRampPalette(c('red', 'yellow', 'green'))(n = 100)
# Draw heatmap of the first cluster with color
biclust::drawHeatmap(BicatYeast, res, 1, FALSE, beamercolor = TRUE, paleta = palette)


## End(Not run)
## Not run:
# Draw some overlapped biclusters
data(BicatYeast)
res <- biclust::biclust(BicatYeast, BCQU(), verbose = FALSE)
biclusternumber(res, 1)
biclusternumber(res, 3)
# Draw overlapping heatmap
biclust::heatmapBC(x = BicatYeast, bicResult = res, number = c(1, 3), local = TRUE)
```

```
## End(Not run)
## Not run:
# Draw all the biclusters
data(BicatYeast)
res <- biclust::biclust(BicatYeast, BCQU(), verbose = FALSE)
# Draw the first bicluster on heatmap
biclust::heatmapBC(x = BicatYeast, bicResult = res, number = 1)
# Draw all the biclusters, not working well.
# Overlap plotting only works for two neighbor bicluster defined by the order in the number slot.
biclust::heatmapBC(x = BicatYeast, bicResult = res, number = 0)


## End(Not run)
# Biclustering of discretized yeast microarray data
data(BicatYeast)
disc<-qudiscretize(BicatYeast[1:10,1:10])
biclust::biclust(disc, method=BCQUD())
```

`qudiscretize` *Create a qualitative discrete matrix for a given gene expression matrix*

Description

`qudiscretize` delivers a discrete matrix. It is useful if we just want to get a discretized matrix.

Usage

```
qudiscretize(x, r = 1L, q = 0.06)
```

Arguments

`x`	the input data matrix, which could be the normalized gene expression matrix or its qualitative representation from Qdiscretization or other discretization ways. (for example: a qualitative representation of gene expression data) For `BCQU()`, the data matrix should be real For `BCQUD()`, the data matrix should be discretized as integer. Zeros in the matrix will be treated as non-relevant value.
`r`	Affect the granularity of the biclusters. The range of possible ranks. A user can start with a small value of `r` (the default value is `1` so the corresponding data matrix consists of values '`1`', '`-1`' and '`0`'), evaluate the results, and then use larger values (should not be larger than half of the number of the columns) to look for fine structures within the identified biclusters.
`q`	Affect the granularity of the biclusters. The percentage of the regulating conditions for each gene. The choice of `q`'s value depends on the specific application goals; that is if the goal is to find genes that are responsive to local regulators, we should use a relatively small q-value; otherwise we may want to consider larger q-values. The default value of `q` is `0.06` in QUBIC (this value is selected based on the optimal biclustering results on simulated data).

Details

qudiscretize convert a given gene expression matrix to a discrete matrix. It's implimented in C++, providing a increase in speed over the C equivalent.

Value

A qualitative discrete matrix

See Also

QUBIC discretize

Examples

```
# Qualitative discretize yeast microarray data
data(BicatYeast)
qudiscretize(BicatYeast[1:7, 1:5])
```

---

quheatmap *Visualization of identified biclusters*

Description

This function can visualize the identifed biclusters using heatmap in support of overall expression pattern analysis,either for a single bicluster or two biclusters.

Usage

```
quheatmap(x, bicResult, number = 1, showlabel = FALSE, col = c("#313695",
  "#4575B4", "#74ADD1", "#ABD9E9", "#E0F3F8", "#FFFFBF", "#FEE090", "#FDAE61",
  "#F46D43", "#D73027", "#A50026"), ...)
```

Arguments

x	The data matrix
bicResult	biclust::BiclustResult object
number	which bicluster to be plotted
showlabel	If TRUE, show the xlabel and ylabel
col	default: c("#313695", "#4575B4", "#74ADD1", "#ABD9E9", "#E0F3F8", "#FFFFBF", "#FEE090", "#FDAE61", "#F46D43", "#D73027", "#A50026")
...	Additional options in fields::image.plot

See Also

qunet2xml QUBIC heatmapBC

Examples

```
# Load microarray matrix
data(BicatYeast)
res <- biclust::biclust(BicatYeast, method=BCQU(), verbose = FALSE)
# Draw heatmap for the 2th identified bicluster
par(mar = c(5, 4, 3, 5) + 0.1, mgp = c(0, 1, 0), cex.lab = 1.1, cex.axis = 0.5, cex.main = 1.1)
quheatmap(x = BicatYeast, res, number = 2, showlabel = TRUE)
# Draw heatmap for the 2th and 3th identified biclusters.
par(mar = c(5, 5, 5, 5), cex.lab = 1.1, cex.axis = 0.5, cex.main = 1.1)
quheatmap(x = BicatYeast, res, number = c(2, 3), showlabel = TRUE)
```

---

qunet2xml *Convert newwork to XGMML*

Description

This function can convert the constructed co-expression networks into XGMML format, which can be used to do further network analysis in Cytoscape, Biomax and JNets.

Usage

```
qunet2xml(net, minimum = 0.6,
  color = cbind(grDevices::rainbow(length(net[[2]]) - 1), "gray"))
```

Arguments

net	Result of qunetwork
minimum	cutoff, default: 0.6
color	default: cbind(grDevices::rainbow(length(net[[2]]) - 1), 'gray')

Value

Text of XGMML

See Also

qunetwork QUBIC

Examples

```
# Load microarray matrix
data(BicatYeast)
res <- biclust::biclust(BicatYeast[1:50, ], method=BCQU(), verbose = FALSE)
# Get all biclusters
net <- qunetwork(BicatYeast[1:50, ], res, group = c(4, 13), method = 'spearman')
# Save the network to a XGMML file
sink('tempnetworkresult.gr')
qunet2xml(net, minimum = 0.6, color = cbind(grDevices::rainbow(length(net[[2]]) - 1), 'gray'))
sink()
# You can use Cytoscape, Biomax or JNets open file named tempnetworkresult.gr
```

qunetwork *Construction and visualization of co-expression network*

Description

This function can automatically create co-expression networks along with their visualization based on identified biclusters in QUBIC. Three correlation methods, Pearson, Kendall and Spearman, are available for a user, facilitating different preferences in practical usage.

Usage

```
qunetwork(x, BicRes, number = 1:BicRes@Number, groups = c(number[[1]]),
  method = c("pearson", "kendall", "spearman"))
```

Arguments

x	The data matrix
BicRes	biclust::BiclustResult object
number	Which bicluster to be plotted
groups	An object that indicates which nodes belong together.
method	A character string indicating which correlation coefficient (or covariance) is to be computed. One of 'pearson' (default), 'kendall', or 'spearman', can be abbreviated.

Value

a list contains a weights matrix and groupinfo

See Also

qunet2xml QUBIC cor

Examples

```
# Load microarray matrix
data(BicatYeast)
res <- biclust::biclust(BicatYeast[1:50, ], method=BCQU(), verbose = FALSE)
# Constructing the networks for the 4th and 13th identified biclusters.
net <- qunetwork(BicatYeast[1:50, ], res, number = c(4, 13), group = c(4, 13), method = 'spearman')
## Not run:
if (requireNamespace('qgraph'))
    qgraph::qgraph(net[[1]], groups = net[[2]], layout = 'spring', minimum = 0.6,
       color = cbind(rainbow(length(net[[2]]) - 1),'gray'), edge.label = FALSE)


## End(Not run)
## Not run:
#Load microarray matrix
data(BicatYeast)
res <- biclust::biclust(BicatYeast[1:50, ], method=BCQU(), verbose = FALSE)
# Constructing the networks for the 4th and 13th identified biclusters,
#   using the whole network as a background.
```

```
net <- qunetwork(BicatYeast[1:50, ], res, group = c(4, 13), method = 'spearman')
if (requireNamespace('qgraph'))
    qgraph::qgraph(net[[1]], groups = net[[2]], layout = 'spring', minimum = 0.6,
       color = cbind(rainbow(length(net[[2]]) - 1),'gray'), edge.label = FALSE)

## End(Not run)
```

---

showinfo *Show report of biclusters*

Description

This function can make a report for biclusters.

Usage

```
showinfo(matrix, bic)
```

Arguments

matrix	microarray matrix
bic	array of biclusters

Value

Text of report

See Also

QUBIC

Examples

```
# Load microarray matrix
data(BicatYeast)
matrix <- BicatYeast[1:50, ];
res1 <- biclust::biclust(matrix, method=BCQU(), verbose = FALSE)
res2 <- biclust::biclust(matrix, method=BCCC())
res3 <- biclust::biclust(matrix, method=BCBimax())
# Show the report
showinfo(matrix, c(res1, res2, res3))
```

Index