Package 'EBcoexpress'

June 6, 2023

Type Package

Title EBcoexpress for Differential Co-Expression Analysis

Version 1.44.0

Date 2012-03-21

Author John A. Dawson

Maintainer John A. Dawson <jadawson@wisc.edu>

Description An Empirical Bayesian Approach to Differential Co-Expression Analysis at the Gene-Pair Level

License GPL (>= 2)

LazyLoad yes

Depends EBarrays, mclust, minqa

Suggests graph, igraph, colorspace

biocViews Bayesian

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

git_branch RELEASE_3_17

git_last_commit 483e5f5

git_last_commit_date 2023-04-25

Date/Publication 2023-06-06

R topics documented:

ebCoexpressMeta *A function for DC meta-analysis that combines individual study analyses*

Description

This function performs a DC meta-analysis, using the hyperparameter estimates obtained from individual-study DC analyses via one of the ebCoexpress series.

Usage

```
ebCoexpressMeta(DList, conditionsList, pattern, hpEstsList, controlOptions = list())
```

Arguments

DList	A list of the individual D matrices from the studies under consideration. They should have the same dimensions, corresponding to the same gene-pairs and the same conditions
conditionsList	A list of the individual conditions arrays from the studies under consideration
pattern	The appropriate output from ebPatterns()
hpEstsList	A list of the individual hyperparameter estimate objects from the studies under consideration, as outputted from the ebCoexpress series (or initializeHP(), the format is the same)
controlOptions	A list with many options for controlling execution: applyTransform: Should Fisher's Z-transformation be applied? Defaults to TRUE verbose: Controls auto-commenting; set to 0 to turn off comments convtol: Convergence tolerance for the EM; default is 5e-04 enforceFloor: Should EC proportion never drop below 0.8? Default is TRUE

Details

Since the meta-analysis model assumes that each study has its own study-specific parameters, those parameters should be estimated using a single-study DC function (one of the other members of the ebCoexpress series); their outputs are used by the hpEstsList option. The EM is then run to determine the system-wide mixing proportions, which are used to compute meta posterior probabilities for all EC/DC classes

Value

The output is a list with two members, MODEL and POSTPROBS:

MODEL is a list containing an array MIX and a list HPS. MIX contains estimated mixing proportions for EC/DC classes. HPS is the inputed list of lists

POSTPROBS is a p-by-L matrix containing posterior probabilities of EC and DC over all L EC/DC classes. The EC posterior probabilities will always be in the first column (which should be fed into crit.fun() if using the soft threshold). Total posterior probabilities of DC for each gene pair are found by summing over the other L-1 columns (or taking 1 minus the first (EC) column)

Author(s)

John A. Dawson <jadawson@wisc.edu>

References

Dawson JA and Kendziorski C. An empirical Bayesian approach for identifying differential co-expression in high-throughput experiments. (2011) Biometrics. E-publication before print: http://onlinelibrary.wiley.com/doi 0420.2011.01688.x/abstract

Examples

```
data(fiftyGenes)
tinyCond <- c(rep(1,100),rep(2,25))
D <- makeMyD(fiftyGenes, tinyCond, useBWMC=TRUE)
set.seed(3)
initHP <- initializeHP(D, tinyCond)

D1 <- D
D2 <- D
DList <- list(D1, D2)
cond1 <- tinyCond
cond2 <- tinyCond
conditionsList <- list(cond1, cond2)
pattern <- ebPatterns(c("1,1","1,2"))
initHP1 <- initHP
initHP2 <- initHP
out1 <- ebCoexpressZeroStep(D1, cond1, pattern, initHP1)
out2 <- ebCoexpressZeroStep(D2, cond2, pattern, initHP2)
hpEstsList <- list(out1$MODEL$HPS, out2$MODEL$HPS)

metaResults <- ebCoexpressMeta(
    DList, conditionsList, pattern, hpEstsList)
```

---

ebCoexpressSeries *Functions to run single-study analyses*

Description

Core functions that run single-study DC analyses in EBcoexpress

Usage

```
ebCoexpressFullTCAECM(D, conditions, pattern, hpEsts, controlOptions = list())
ebCoexpressOneStep(D, conditions, pattern, hpEsts, controlOptions = list())
ebCoexpressZeroStep(D, conditions, pattern, hpEsts, controlOptions = list())
```

Arguments

D	The correlation matrix output of makeMyD()
conditions	The conditions array
pattern	An appropriate output from ebPatterns()
hpEsts	The initial hyperparameter estimates from initializeHP() or some other method
controlOptions	A list with many options for controlling execution: ** These two are common to all members of the series: applyTransform: Should Fisher's Z-transformation be applied? Defaults to TRUE verbose: Controls auto-commenting; set to 0 to turn off comments ** These are used only by the OneStep and FullTCAECM versions: convtol: Convergence tolerance for the EM; default is 5e-04 enforceFloor: Should EC proportion never drop below 0.8? Default is TRUE subsize: If non-NULL, a value less than the 1st dimension of D (p). The EM will use subsize randomly chosen pairs in its computations instead of all p pairs; by default, all pairs are used. We suggest use of this option when the number of pairs is very large m2MaxIter: Upper limit of the number of itertions the optimizer from minqa will use in the M2 step; defaults to 100. For unrestricted iterations, set this to NULL

Details

These three functions represent different flavors of the TCAECM. The FullTCAECM version will run a full TCAECM. The OneStep version will perform a single iteration of the TCAECM and return the results. The ZeroStep version does not perform any EM calculations and instead uses the initial estimates of the hyperparameters to generate posterior probabilities of DC.

We recommend using the OneStep version in most cases.

Value

The output is a list with two members, MODEL and POSTPROBS:

MODEL is a list containing an array MIX and a list HPS. MIX contains estimated mixing proportions for EC/DC classes. HPS contains model specifications:

G	The number of mixture components (1, 2 or 3)
MUS	Means across mixture components
TAUS	Standard deviations across mixture components
WEIGHTS	Weights across mixture components (these sum to 1)

The list of lists required by ebCoexpressMeta() is obtained by listing the separate analyses' HPS lists

POSTPROBS is a p-by-L matrix containing posterior probabilities of EC and DC over all L EC/DC classes. The EC posterior probabilities will always be in the first column (which should be fed into crit.fun() if using the soft threshold). Total posterior probabilities of DC for each gene pair are found by summing over the other L-1 columns (or taking 1 minus the first (EC) column)

Author(s)

John A. Dawson <jadawson@wisc.edu>

References

Dawson JA and Kendziorski C. An empirical Bayesian approach for identifying differential co-expression in high-throughput experiments. (2011) Biometrics. E-publication before print: http://onlinelibrary.wiley.com/doi 0420.2011.01688.x/abstract

Examples

```
data(fiftyGenes)
tinyCond <- c(rep(1,100),rep(2,25))
tinyPat <- ebPatterns(c("1,1","1,2"))
D <- makeMyD(fiftyGenes, tinyCond, useBWMC=TRUE)
set.seed(3)
initHP <- initializeHP(D, tinyCond)

zout <- ebCoexpressZeroStep(D, tinyCond, tinyPat, initHP)
## Not run: oout <- ebCoexpressOneStep(D, tinyCond, tinyPat, initHP)
## Not run: fout <- ebCoexpressFullTCAECM(D, tinyCond, tinyPat, initHP)

softThresh <- crit.fun(zout$POSTPROB[,1], 0.05)
```

`fiftyGenes` *The fiftyGenes expression matrix*

Description

A simulated expression matrix for fifty genes in two conditions, with one hundred chips and twenty-five chips in the two coniditions. Most gene pairs are uncorrelated, but all gene pairs involving only genes X1 to X25, or only X26 to X50, are DC. So for instance, X1~X2 is DC but X1~X30 is not

Usage

```
data(fiftyGenes)
```

Format

num [1:50, 1:125] 0.655 0.0188 1.0786 1.6856 0.4814 ... - attr(*, "dimnames")=List of 2 ..$: chr [1:50] "X1" "X2" "X3" "X4" ... ..$: chr [1:125] "C1x1" "C1x2" "C1x3" "C1x4" ...

Examples

```
data(fiftyGenes)
```

`initializeHP` *Initializing Hyperparameters for EM*

Description

A function for initializing the EM hyperparameters. While the user is free to do this in any manner s/he deems fit, we use the excellent Mclust approach of package R/mclust

Usage

```
initializeHP(D, conditions, seed = NULL, plottingOn = FALSE, colx = "red",
applyTransform = TRUE, verbose = 0, subsize = NULL, ...)
```

Arguments

`D`	The correlation matrix output of makeMyD()
`conditions`	The conditions array
`seed`	A seed for making this procedure deterministic
`plottingOn`	Should the weighted vs. unweighted comparison plots be shown to the user? Default is FALSE (no plotting)
`colx`	The color of the fitted empirical density curve, if plottingOn is TRUE
`applyTransform`	Should Fisher's Z-transformation be applied to the correlations?
`verbose`	An option to control comments as initialization proceeds. Set to 1 to see comments, default is 0
`subsize`	A value less than the 1st dimension of D, if it is desired that only some of the pairs be used in the estimation process (for computational reasons)
`...`	Other options to be passed to plot()

Details

initializeHP() initializes the hyperparameters by asking Mclust to find the 1-, 2- or 3- component Normal mixture model that best fits the (transformed) correlations as a whole. Mclust directly returns estimates for G and the MUS; TAUS are estimated using Mclust's estimates and sample sizes, per our model. WEIGHTS are estimated using the mixture component classifications Mclust provides; however, it is unclear whether those classifications should we weighted by how confident Mclust is in their accuracy. We have tried both approaches and neither is superior to the other in all cases, so we compute both, compare the model fits and return the WEIGHTS that best empirically fit the data. This process is shown visually if plottingOn is TRUE and the comments (if enabled) will describe the comparison. In the event that the TAUS are estimated to be less than 0, half of the (TAUS+SS_VARIANCE) estimate provided by Mclust is used for TAUS; we especially do not recommend the use of ebCoexpressZeroStep when this occurs (as opposed to generally favoring the one-step version over the zero-step)

Value

A list with four components, describing the hyperparameters:

G	The number of mixture components (1, 2 or 3)
MUS	Means across mixture components
TAUS	Standard deviations across mixture components
WEIGHTS	Weights across mixture components (these sum to 1)

Author(s)

John A. Dawson <jadawson@wisc.edu>

References

Dawson JA and Kendziorski C. An empirical Bayesian approach for identifying differential co-expression in high-throughput experiments. (2011) Biometrics. E-publication before print: http://onlinelibrary.wiley.com/doi
0420.2011.01688.x/abstract

Examples

```
data(fiftyGenes)
tinyCond <- c(rep(1,100),rep(2,25))
tinyPat <- ebPatterns(c("1,1","1,2"))
D <- makeMyD(fiftyGenes, tinyCond, useBWMC=TRUE)
set.seed(3)
initHP <- initializeHP(D, tinyCond)
```

---

makeMyD — *A function to convert the X expression matrix into the D correlation matrix*

Description

A function to convert the X expression matrix into the D correlation matrix; uses either Pearson's correlation coefficient or biweight midcorrelation

Usage

```
makeMyD(X, conditions, useBWMC = FALSE, gpsep = "~")
```

Arguments

X	An m-by-n expression matrix, where rows are genes and columns are chips (subjects); include all chips in X, indicate condition in the conditions array
conditions	The conditions array
useBWMC	Should biweight midcorrelation be used instead of Pearson's correlation coefficient?
gpsep	A separator that indicates a gene-pair, such as P53~MAPK1. The separator should not appear in any of the gene names

Value

A p-by-K matrix of observed correlations for all p gene-pairs, where p is choose(m,2), m is the 1st dimension of X and K is the number of conditions specified by the conditions array

Author(s)

John A. Dawson <jadawson@wisc.edu>

References

Dawson JA and Kendziorski C. An empirical Bayesian approach for identifying differential co-expression in high-throughput experiments. (2011) Biometrics. E-publication before print: http://onlinelibrary.wiley.com/doi 0420.2011.01688.x/abstract

Examples

```
data(fiftyGenes)
tinyCond <- c(rep(1,100),rep(2,25))
tinyPat <- ebPatterns(c("1,1","1,2"))
D <- makeMyD(fiftyGenes, tinyCond, useBWMC=TRUE)
```

`priorDiagnostic` *Visual diagnostic for the EBcoexpress prior*

Description

A visual diagnostic used to check the prior estimated by an ebCoexpressSeries function against the data using the prior predictive distribution specified by the EBcoexpress model. The function compares the empirical prior predictive distribution of the (transformed) correlations in one condition against the theoretical prior predictive distribution

Usage

```
priorDiagnostic(D, conditions, ebOutObj, focusCond, seed = NULL, colx = "red", applyTransform = TRUE, su
```

Arguments

`D`	The correlation matrix output of makeMyD()
`conditions`	The conditions array
`ebOutObj`	The structured list output from an ebCoexpressSeries function
`focusCond`	A condition whose correlations will be used in the diagnostic. We suggest running the diagnostic for each condition, one at a time
`seed`	A seed for making the subsize subselection deterministic; has no effect if subsize= is left NULL
`colx`	A color for the fitted marginal distribution. Defaults to red
`applyTransform`	Should Fisher's Z-transformation be applied? Defaults to TRUE

subsize	If non-NULL, a value less than the 1st dimension of D (p). The diagnostic will use subsize randomly chosen correlations from the condition in its computation of the empirical density instead of all p pairs; by default, all pairs are used. We suggest use of this option when the number of pairs is very large
...	Other parameters to be passed to plot()

Details

This function is a diagnostic tool for checking the prior distribution selected by the EM during an ebCoexpressSeries function's computations using the prior predictive distribution. The better the prior fits the observed data, the more confidence we should have in the posterior probabilities generated by the EM.

When run, the user specifies a condition. All of the (transformed) correlations from that condition (or just some of them if the subsize= option is non-NULL) will be used to estimate the empirical prior predictive distribution of the data in that condition; this will be plotted in black. The diagnostic then calculates the the theoretical prior predictive distribution and plots it using a dashed, colored line (set by colx=). If the two densities are similar, this indicates the selected prior fits the data in this condition well. The process can and should be repeated for all other conditions

Value

Returns invisible(NULL)

Author(s)

John A. Dawson <jadawson@wisc.edu>

References

Dawson JA and Kendziorski C. An empirical Bayesian approach for identifying differential co-expression in high-throughput experiments. (2011) Biometrics. E-publication before print: http://onlinelibrary.wiley.com/doi
0420.2011.01688.x/abstract

Examples

```
data(fiftyGenes)
tinyCond <- c(rep(1,100),rep(2,25))
tinyPat <- ebPatterns(c("1,1","1,2"))
D <- makeMyD(fiftyGenes, tinyCond, useBWMC=TRUE)
set.seed(3)
initHP <- initializeHP(D, tinyCond)

zout <- ebCoexpressZeroStep(D, tinyCond, tinyPat, initHP)
par(mfrow=c(2,1))
priorDiagnostic(D, tinyCond, zout, 1)
priorDiagnostic(D, tinyCond, zout, 2)
par(mfrow=c(1,1))
```

`rankMyGenes` *A function to rank the genes by the number of DC pairs in which they appear*

Description

This function uses a threshold to determine the names of the DC pairs. It then splits those pairs into their constituent genes and tables them. A sorted version of that table is then returned. This information may be useful for those investigating 'differential hubbing' – see the Hudson et al. reference for more information

Usage

```
rankMyGenes(emOut, thresh = 0.95, sep = "~")
```

Arguments

`emOut`	The output of an ebCoexpressSeries function call
`thresh`	A threshold for determining whether a pair is DC. This may be set as a hard threshold (default is hard 5 threshold, as returned by crit.fun
`sep`	The separator used in the pair names

Value

A sorted, named array of gene counts

Author(s)

John A. Dawson <jadawson@wisc.edu>

References

Dawson JA and Kendziorski C. An empirical Bayesian approach for identifying differential co-expression in high-throughput experiments. (2011) Biometrics. E-publication before print: http://onlinelibrary.wiley.com/doi 0420.2011.01688.x/abstract

Hudson NJ, Reverter A, Dalrymple BP (2009) A Differential Wiring Analysis of Expression Data Correctly Identifies the Gene Containing the Causal Mutation. PLoS Comput Biol 5(5): e1000382. doi:10.1371/journal.pcbi.1000382

See Also

ebCoexpressSeries, crit.fun

Examples

```
data(fiftyGenes)
tinyCond <- c(rep(1,100),rep(2,25))
tinyPat <- ebPatterns(c("1,1","1,2"))
D <- makeMyD(fiftyGenes, tinyCond, useBWMC=TRUE)
set.seed(3)
initHP <- initializeHP(D, tinyCond)

zout <- ebCoexpressZeroStep(D, tinyCond, tinyPat, initHP)
rankMyGenes(zout)
```

---

showNetwork *A function for looking at the co-expression among a small group of genes*

Description

This function draws a network for a selected group of genes using igraph. The edges are colored in accordance with the correlation strength indicated by the inputted D matrix, ranging from red (strong negative correlation) to blue (strong positive correlation)

Usage

```
showNetwork(geneSet, D, condFocus, gsep = "~", layout = "kamada.kawai", seed = NULL, hidingThreshold=NU
```

Arguments

geneSet	An array of genes of interest; should not be larger than a dozen or so
D	The correlation matrix output of makeMyD()
condFocus	The condition of interest for this network. Should be one of the integers in the conditions array
gsep	A separator that indicates a gene-pair, such as P53~MAPK1. The separator should not appear in any of the gene names
layout	A layout to be parsed and used by igraph. Examples include circle (the default) and kamada.kawai; see the documentation for igraph for more information. At this time it is not possible to specify parameters specific to particular layouts
seed	A seed to be set before invoking igraph's layout generation. This is useful for layouts such as random, where node postion is not deterministic
hidingThreshold	A threshold which we will shorthand by 'h'. If this value is non-NULL, all correlations in [-h, h] will not be plotted in the network. This is useful for removing clutter in busy networks will relatively high (say, 20+) numbers of genes

... Other options to be passed to plot.igraph(). Networks generated by igraph require quite a bit of formatting, and it is up to the user to do so by specifying appropriate options from the following:

vertex.shape=, vertex.label.cex=, vertex.color=, vertex.frame.color=, vertex.size=, vertex.label.color=, vertex.label.family=, and edge.width=

The following options are hard-coded and may not be overwritten:

vertex.label=geneSet, edge.arrow.mode=0, edge.color=[red/blue colors]

where [red/blue colors] is determined by the correlation information contained in D, possibly overwritten in some cases if hidingThreshold is non-NULL

Value

Returns invisible(NULL)

Author(s)

John A. Dawson <jadawson@wisc.edu>

References

Dawson JA and Kendziorski C. An empirical Bayesian approach for identifying differential co-expression in high-throughput experiments. (2011) Biometrics. E-publication before print: http://onlinelibrary.wiley.com/doi 0420.2011.01688.x/abstract

See Also

igraph, igraph.layout

Examples

```
data(fiftyGenes)
tinyCond <- c(rep(1,100),rep(2,25))
tinyPat <- ebPatterns(c("1,1","1,2"))
D <- makeMyD(fiftyGenes, tinyCond, useBWMC=TRUE)
twentyGeneNames <- dimnames(fiftyGenes)[[1]][c(1:10,26:35)]

showNetwork(twentyGeneNames, D, condFocus = 1, gsep = "~",
  layout = "kamada.kawai", seed = 5, vertex.shape="circle",
  vertex.label.cex=1, vertex.color="white", edge.width=2,
  vertex.frame.color="black", vertex.size=20,
  vertex.label.color="black", vertex.label.family="sans",
  hidingThreshold=0.3)
#
showNetwork(twentyGeneNames, D, condFocus = 2, gsep = "~",
  layout = "kamada.kawai", seed = 5, vertex.shape="circle",
  vertex.label.cex=1, vertex.color="white", edge.width=2,
  vertex.frame.color="black", vertex.size=20,
  vertex.label.color="black", vertex.label.family="sans",
  hidingThreshold=0.3)
#
```

showPair *A function for looking at a pair's differential co-expression*

Description

This function plots the expression data for a given pair, coloring by condition. A regression line may be added; this line may be made robust so that it only uses those data points used by biweight midcorrelation

Usage

```
showPair(pair, X, conditions, gsep = "~", regLine = TRUE, useBWMC = TRUE, colors = NULL, ...)
```

Arguments

pair	A pair name, such as ABC~XYZ, where ~ is the separator given by gsep
X	An m-by-n expression matrix, where rows are genes and columns are chips (subjects); include all chips in X, indicate condition in the conditions array
conditions	The conditions array
gsep	A separator that indicates a gene-pair, such as P53~MAPK1. The separator should not appear in any of the gene names
regLine	Should a regression line be drawn for each condition?
useBWMC	Should the regression line be robust, a la biweight midcorrelation?
colors	Colors for the different conditions. Defaults to palette()
...	Other options to be passed to plot(), with three exceptions. The lty= and lwd= options will be passed to abline() and will have an effect on the plot when regLine=TRUE; col= is overwritten by the colors= array and may not be specified. All other ... options will be passed to the main plot.

Value

Returns invisble(NULL)

Author(s)

John A. Dawson <jadawson@wisc.edu>

References

Dawson JA and Kendziorski C. An empirical Bayesian approach for identifying differential co-expression in high-throughput experiments. (2011) Biometrics. E-publication before print: http://onlinelibrary.wiley.com/doi
0420.2011.01688.x/abstract

Examples

```
data(fiftyGenes)
tinyCond <- c(rep(1,100),rep(2,25))
showPair("X1~X2",fiftyGenes, conditions=tinyCond, pch=20,
  xlim=c(-4,4), ylim=c(-4,4))
#
showPair("X26~X35",fiftyGenes, conditions=tinyCond, pch=20,
  xlim=c(-4,4), ylim=c(-4,4))
#
showPair("X1~X35",fiftyGenes, conditions=tinyCond, pch=20,
  xlim=c(-4,4), ylim=c(-4,4))
```

`utilities` *Basic utilities for the EBcoexpress package*

Description

At present there are two utilties: crit.fun() and bwmc(). The former is used to compute soft thresholds for FDR control, the latter is like cor() but uses biwieght midcorrelation instead of the usual Pearson's correlation coefficient.

Usage

```
crit.fun(ecPostProbs, targetFDR)
bwmc(X)
```

Arguments

`ecPostProbs`	An array of posterior probabilities of equivalent coexpression for all pairs
`targetFDR`	A target FDR rate
`X`	An expression matrix in one condition where the rows correspond to genes

Details

crit.fun() returns a soft threshold for FDR control. It is similar to the function of the same name in the package EBarrays. bwmc() computes the biweight midcorrelation for an expression matrix; it is used internally to generate the D correlations matrix by makeMyD() when useBWMC is TRUE. It is also a handy little function so we made it visible at the top level. The guts of this function are in C for speed

Value

crit.fun returns a single value; under a soft thresholding approach, any pair with total posterior probability of differential co-expression (i.e., 1 - posterior probability of equivalent co-expression) greater than this value is deemed to be DC

If X has 1st dimension m, bwmc(t(X)) returns an m-by-m matrix of pairwise biweight midcorrelations as a matrix, in a manner similar to cor().

Author(s)

John A. Dawson <jadawson@wisc.edu>

References

Dawson JA and Kendziorski C. An empirical Bayesian approach for identifying differential co-expression in high-throughput experiments. (2011) Biometrics. E-publication before print: http://onlinelibrary.wiley.com/doi 0420.2011.01688.x/abstract

Examples

```
set.seed(1)
ecs <- c(runif(950),runif(50,0,0.01))
thresh <- crit.fun(ecs, 0.05)

set.seed(1)
X <- matrix(runif(10*100),10,100)
print(cor(t(X)))
print(bwmc(t(X)))
```

Index