Package 'cycle'

November 11, 2026

Version 1.66.0

Date 2016-02-18

Title Significance of periodic expression pattern in time-series data

Author Matthias Futschik <mfutschik@ualg.pt>

Maintainer Matthias Futschik <mfutschik@ualg.pt>

Depends R (>= 2.10.0), Mfuzz

Imports Biobase, stats

Description Package for assessing the statistical significance of periodic expression based on Fourier analysis and comparison with data generated by different background models

License GPL-2

URL http://cycle.sysbiolab.eu

biocViews Microarray, TimeCourse

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

git_branch RELEASE_3_23

git_last_commit 2e33d7b

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-11-11

Contents

ar1analysis 2
backgroundData 3
fdrfourier 4
fourierscore 6

Index **9**

ar1analysis *Performs AR1 fitting*

Description

Calculation of the autocorrelation coefficients genes and variance of corresponding random variables to fit gene expression time series by AR1 processes

Usage

```
ar1analysis(eset)
```

Arguments

eset	object of the class "ExpressionSet"

Value

List of fitted autocorrelation coefficients (alpha) for ExpressionSet features and variance (sigma2) of corresponding random variables obtained using the ar function of the *stats* package.

Note

Note that this function evaluates soley the exprs matrix and no information is used from the phenoData. In particular, the ordering of samples (arrays) is the same as the ordering of the columns in the exprs matrix. Also, replicated arrays in the exprs matrix are treated as independent i.e. they should be averagered prior to analysis or placed into different distinct "ExpressionSet" objects.

See Also

ar

Examples

```
data(yeast) # loading the reduced CDC28 yeast set (from the Mfuzz package)

# Data preprocessing
if (interactive()){
data(yeast)

yeast <- filter.NA(yeast)
# filters genes with more than 25% of the expression values missing

yeast  <- fill.NA(yeast)
# for illustration only; rather use knn method for replacing missing values

tmp <- ar1analysis(yeast)
# fits AR1 process autocorrelation coefficients
```

```
plot(density(tmp$alpha),main="Autocorrelation")

}
```

`backgroundData` *Generation of background expression set*

Description

The function generates background expression sets using different methods (permutation within rows, Gaussian distribution, auto-regressive models)

Usage

```
backgroundData(eset,model=c("rr", "gauss", "ar1"))
```

Arguments

`eset`	object of the class "ExpressionSet"
`model`	model for generation of background data: "rr"- random permutation, "gauss"- Gaussian and "ar1"- AR1 models

Details

Microarray data comprise the measurements of transcript levels for many thousands of genes. Due to the large number of genes, it can be expected that some genes show periodicity simply by chance. To assess therefore the significance of periodic signals, it is necessary first to define what distribution of signals can be expected if the studied process exhibits no true periodicity. In statistical terms this is equivalent with the definition of a null hypothesis of non-periodic expression.

The most simple model for non-periodic expression is based on randomization of the observed times series. A background distribution can then be constructed by (repeated) random permutation of the sequentially ordered measurements in the experiment. This background model is used here if `model="rr"` is chosen.

Alternatively, non-periodic expression can be derived using a statistical model. A conventional approach is based on the assumption of data normality and to use the normal distribution. This background model is chosen if `model="gauss"`.

However, these two approaches neglect the fact that time series data exhibit generally a considerable autocorrelation i.e. correlation between successive measurements. Therefore, neither the assumptions of data normality nor for randomizations may hold. As demonstrated for yeast cell cycle data (*Bioinformatics* 2008), this failure can substantially interfere with the significance testing, and that neglecting autocorrelation can potentially lead to a considerable overestimation of the number of periodically expressed genes.

A more suitable model is based on autoregressive processes of order one (AR(1)), for which the value of the time-dependent variable X depends on its previous value up to a normally distributed random variable Z. Such model is used here for the setting of `model="ar1"`. The autocorrelation of

X and variance of Z is estimated for each feature of the `ExpressionSet` object separately. Mathematical details can be found in the given reference.

It is important to note in this context, that AR(1) processes cannot capture periodic patterns except for alternations with period two. Since Z is a random variable, we can readily generate a collection of time series with the same autocorrelation as in the original data set. Therefore, although AR(1) processes constitute random processes, they allow us to construct a background distribution that captures the autocorrelation structure of original gene expression time series without fitting the potentially included periodic pattern.

Value

ExpressionSet object with expression data generated by the chosen background model

Note

Note that this function evaluates soley the `exprs` matrix and no information is used from the `phenoData`. In particular, the ordering of samples (arrays) is the same as the ordering of the columns in the `exprs` matrix. Also, replicated arrays in the `exprs` matrix are treated as independent i.e. they should be averagered prior to analysis or placed into different distinct "ExpressionSet" objects.

Author(s)

Matthias E. Futschik (http://www.cbme.ualg.pt/mfutschik_cbme.html)

References

Matthias E. Futschik and Hanspeter Herzel (2008) Are we overestimating the number of cell-cycling genes? The impact of background models on time series analysis, *Bioinformatics*, 24(8):1063-1069

`fdrfourier` *Calculation of the false discovery rates (FDR) for periodic expression*

Description

The function calculates the empirical FDR based on derived Fourier scores derived by `fourierscore` for the observed expression and the comparison with scores derived for different background model generated by `backgroundData`.

Usage

```
fdrfourier(eset,T,times,background.model="rr",N=100,progress=FALSE)
```

Arguments

eset	object of the class "ExpressionSet"
T	cycle period
times	time of measurements
background.model	model for generation of background data: "rr"- permutation within rows, "gauss"- Gaussian background, "ar1"- AR1 models
N	number of generated data sets for the background distribution
progress	if set to TRUE, a progress of calculations is reported

Details

To assess the significance of the Fourier score obtained for the original gene expression time series, the probability has to be calculated of how often such a score would be observed by chance based on the chosen background distribution. The statistical significance is given by the calculated false discovery rate. It is defined here as the expected proportion of false positives among all genes detected as periodically expressed. Mathematical details can be found in the given reference.

Value

List with FDR for the features of the eset object (fdr), and Fourier scores for ExpressionSet object (F) and the background data (F.b).

Note

This is the main function of the cycle package. Note that the calculation of FDR employing empirical background distributions can require considerable time (up to several days for large gene expression data sets).

Importantly, this function evaluates soley the exprs matrix and no information is used from the phenoData. In particular, the ordering of samples (arrays) is the same as the ordering of the columns in the exprs matrix. Also, replicated arrays in the exprs matrix are treated as independent i.e. they should be averagered prior to analysis or placed into different distinct "ExpressionSet" objects.

Author(s)

Matthias E. Futschik (http://www.cbme.ualg.pt/mfutschik_cbme.html)

References

Matthias E. Futschik and Hanspeter Herzel (2008) Are we overestimating the number of cell-cycling genes? The impact of background models on time series analysis, *Bioinformatics*, 24(8):1063-1069

Examples

```
if (interactive()){

set.seed(1)
data(yeast) # loading the reduced CDC28 yeast set (from the Mfuzz package)
```

```
# Data preprocessing
yeast <- filter.NA(yeast) # filters genes with more than 25% of the expression values missing
yeast  <- fill.NA(yeast) # for illustration only; rather use knn method for
yeast <- standardise(yeast)
#
T.yeast <- 85   # cell cycle period (t=85min)
times.yeast <-  pData(yeast)$time  # time of measurements
#
yeast.test <- yeast[1:600,] # To speed up the example
#

NN <- 50 # number of generated background models
         # Here, a small number was chosen for demonstration purpose.
  # For the actual analysis, rather set N = 1000

# Calculation of FDRs
# i) based on random permutation as background model
fdr.rr <- fdrfourier(eset=yeast.test,T=T.yeast,
                    times=times.yeast,background.model="rr",N=NN,progress=TRUE)
# ii) based on Gaussian distribution
fdr.g <- fdrfourier(eset=yeast.test,T=T.yeast,
                   times=times.yeast,background.model="gauss",N=NN,progress=TRUE)
# iii) based on AR(1) models as background
fdr.ar1 <- fdrfourier(eset=yeast.test,T=T.yeast,
                     times=times.yeast,background.model="ar1",N=NN,progress=TRUE)


# Number of significant genes based on diff. background models
sum(fdr.rr$fdr < 0.1)
sum(fdr.g$fdr < 0.1)
sum(fdr.ar1$fdr < 0.1)

# Plot top scoring gene
plot(times.yeast,exprs(yeast.test)[order(fdr.ar1$fdr)[1],],type="o",
     xlab="Time",ylab="Expression",
     main=paste(featureNames(yeast.test)[order(fdr.ar1$fdr)[1]],"-- FDR:",
       fdr.ar1$fdr[order(fdr.ar1$fdr)[1]]))

# List significant genes
fdr.ar1$fdr[which(fdr.ar1$fdr < 0.1)]
}
```

`fourierscore` *Calculation of the Fourier score*

Description

The function calculates fourier score for a chosen periodicity.

Usage

```
fourierscore(eset,T,times=seq_len(ncol(eset)))
```

Arguments

eset	object of the class "ExpressionSet"
T	length of chosen period
times	measurement times (with the index of the column as default)

Details

The Fourier score can be used to detect periodic signals. The closer a gene's expression follows a (possibly shifted) cosine curve of period T, the larger is the Fourier score. Mathematical details can be found in the given reference. The function fourierscore calculates the Fourier scores for all features of an ExpressionSet object.

Value

Fourier scores for all features (genes) of eset

Note

Note that this function evaluates soley the exprs matrix and no information is used from the phenoData. In particular, the ordering of samples (arrays) is the same as the ordering of the columns in the exprs matrix. Also, replicated arrays in the exprs matrix are treated as independent i.e. they should be averagered prior to analysis or placed into different distinct "ExpressionSet" objects.

Author(s)

Matthias E. Futschik (http://www.cbme.ualg.pt/mfutschik_cbme.html)

References

Matthias E. Futschik and Hanspeter Herzel (2008) Are we overestimating the number of cell-cycling genes? The impact of background models on time series analysis, *Bioinformatics*, 24(8):1063-1069

Examples

```
# Data preprocessing
if (interactive()){
set.seed(1)
data(yeast)  # loading the reduced CDC28 yeast set (from the Mfuzz package)
yeast <- yeast[1:600,]  # for illustration
yeast <- filter.NA(yeast) # filters genes with more than 25% of the expression values missing
yeast  <- fill.NA(yeast) # for illustration only; rather use knn method for
yeast <- standardise(yeast)

T.yeast <- 85   # cell cycle period (t=85min)
times.yeast <-  pData(yeast)$time  # time of measurements
```

```
F <- fourierscore(yeast,T = T.yeast, times= times.yeast)
# calculates the Fourier scores for yeast genes

# Plot highest scoring gene
plot(times.yeast,exprs(yeast)[order(-F)[1],],type="o",
     xlab="Time",ylab="Expression",
     main=featureNames(yeast)[order(-F)[1]])

# Plot lowest scoring gene
plot(times.yeast,exprs(yeast)[order(F)[1],],type="o",
     xlab="Time",ylab="Expression",
     main=featureNames(yeast)[order(F)[1]])
}
```

Index

∗ **datagen**
backgroundData, 3
∗ **distribution**
backgroundData, 3
∗ **htest**
fdrfourier, 4
∗ **ts**
ar1analysis, 2
fdrfourier, 4
fourierscore, 6

ar, 2
ar1analysis, 2

backgroundData, 3

fdrfourier, 4
fourierscore, 6