Package 'GeneCycle'

January 20, 2025

Version 1.1.5

Date 2021-01-04

Title Identification of Periodically Expressed Genes

Author Miika Ahdesmaki, Konstantinos Fokianos, and Korbinian Strimmer.

Maintainer Miika Ahdesmaki <miika.ahdesmaki@gmail.com>

Depends R (>= 2.7.0), MASS, longitudinal (>= 1.1.3), fdrtool (>= 1.2.5)

Description The GeneCycle package implements the approaches of Wichert et al. (2004) <doi:10.1093/bioinformatics/btg364>, Ahdesmaki et al. (2005) <doi:10.1186/1471-2105-6-117> and Ahdesmaki et al. (2007) <DOI:10.1186/1471-2105-8-233> for detecting periodically expressed genes from gene expression time series data.

License GPL (>= 3)

Repository CRAN

Date/Publication 2021-01-05 17:20:03 UTC

NeedsCompilation no

Contents

avgp *Average Periodogram for Multiple (Genetic) Time Series*

Description

avgp calculates and plots the average periodogram as described in Wichert, Fokianos and Strimmer (2004).

Usage

```
avgp(x, title = deparse(substitute(x)), plot = TRUE, angular = FALSE, ...)
```

Arguments

x	multiple (genetic) time series data. Each **column** of this matrix corresponds to a separate variable/time series
title	name of the data set (default is the name of the data object)
plot	plot the average periodogram?
angular	convert frequencies to angular frequencies?
...	arguments passed to plot and to periodogram

Details

The average periodogram is simply the frequency-wise average of the spectral density (as estimated by the Fourier transform) over all times series. To calculate the average periodogram the function periodogram is used. See Wichert, Fokianos and Strimmer (2004) for more details.

Value

A list object with the following components:

freq	A vector with the discrete Fourier frequencies (see periodogram). If the option angular=TRUE then the output are angular frequencies (2*pi*f).
avg.spec	A vector with the average power spectral density at each frequency.
title	Name of the data set underlying the average periodogram.

The result is returned invisibly if plot is true.

Author(s)

Konstantinos Fokianos and Korbinian Strimmer (https://www.strimmerlab.org/).

References

Wichert, S., Fokianos, K., and Strimmer, K. (2004). Identifying periodically expressed transcripts in microarray time series data. *Bioinformatics* **20**:5-20.

See Also

periodogram, spectrum.

Examples

```
# load GeneCycle library
library("GeneCycle")

# load data set
data(caulobacter)

# how many samples and how many genes?
dim(caulobacter)

# average periodogram
avgp.caulobacter <- avgp(caulobacter, "Caulobacter")
avgp.caulobacter

# just compute and don't plot
avgp(caulobacter, "Caulobacter", plot=FALSE)
```

---

caulobacter *Microarray Time Series Data for 1444 Caulobacter Crescentus Genes*

Description

This data set describes the temporal expression of 1444 genes (open reading frames) in the cell cycle of the bacterium *Caulobacter crescentus*.

Usage

```
data(caulobacter)
```

Format

caulobacter is a longitudinal object containing the data from the Laub et al. (2000) experiment. Essentially, this is a matrix with with 1444 columns (=genes) and 11 rows (=time points)

Source

This data is described in Laub et al. (2000).

References

Laub, M.T., McAdams, H.H., Feldblyum, Fraser, C.M., and Shapiro, L. (2000) Global analysis of the genetic network controlling a bacterial cell cycle. *Science*, **290**, 2144–1248.

Examples

```
# load GeneCycle library
library("GeneCycle")

# load data set
data(caulobacter)
is.longitudinal(caulobacter)

# how many samples and how many genes?
dim(caulobacter)
summary(caulobacter)
get.time.repeats(caulobacter)

# plot first nine time series
plot(caulobacter, 1:9)
```

`dominant.freqs` *Dominant Frequencies in Multiple (Genetic) Time Series*

Description

`dominant.freqs` returns the m dominant frequencies (highest peaks) in each of the periodogram computed for the individual time series.

Usage

```
dominant.freqs(x, m=1, ...)
```

Arguments

`x`	multivariate (genetic) time series (each **column** of this matrix corresponds to a separate variable/time series), or a vector with a single time series
`m`	number of dominant frequences
`...`	arguments passed to `periodogram`

Value

A matrix (or vector, if only 1 time series is considered) with the dominant frequencies. In a matrix, each column corresponds to one time series.

Author(s)

Konstantinos Fokianos and Korbinian Strimmer (https://www.strimmerlab.org/).

See Also

`periodogram`, `spectrum`.

Examples

```
# load GeneCycle library
library("GeneCycle")

# load data set
data(caulobacter)

# how many samples and how many genes?
dim(caulobacter)

# first three dominant frequencies for each gene
dominant.freqs(caulobacter, 3)

# first four dominant frequencies for gene no. 1000
dominant.freqs(caulobacter[,1000], 4)
```

`fisher.g.test` *Fisher's Exact g Test for Multiple (Genetic) Time Series*

Description

`fisher.g.test` calculates the p-value(s) according to Fisher's exact g test for one or more time series. This test is useful to detect hidden periodicities of unknown frequency in a data set. For an application to microarray data see Wichert, Fokianos, and Strimmer (2004).

Usage

```
fisher.g.test(x, ...)
```

Arguments

`x`	vector or matrix with time series data (one time series per column).
`...`	arguments passed to `periodogram`

Details

Fisher (1929) devised an exact procedure to test the null hypothesis of Gaussian white noise against the alternative of an added deterministic periodic component of unspecified frequency. The basic idea behind the test is to reject the null hypothesis if the periodogram contains a value significantly larger than the average value (cf. Brockwell and Davis, 1991). This test is useful in the context of microarray genetic time series analysis as a gene selection method - see Wichert, Fokianos and Strimmer (2004) for more details. Note that in the special case of a constant time series the p-value returned by `fisher.g.test` is exactly 1 (i.e. the null hypothesis is not rejected).

Value

A vector of p-values (one for each time series). Multiple testing may then be done using the the false discover rate approach (function `fdrtool`).

Author(s)

Konstantinos Fokianos and Korbinian Strimmer (https://www.strimmerlab.org/).

References

Fisher, R.A. (1929). Tests of significance in harmonic analysis. *Proc. Roy. Soc. A*, **125**, 54–59.

Brockwell, P.J., and Davis, R.A. (1991). Time Series: Theory and Methods (2nd ed). Springer Verlag. (the g-test is discussed in section 10.2).

Wichert, S., Fokianos, K., and Strimmer, K. (2004). Identifying periodically expressed transcripts in microarray time series data. *Bioinformatics* **20**:5-20.

See Also

fdrtool.

Examples

```
# load GeneCycle library
library("GeneCycle")

# load data set
data(caulobacter)

# how many samples and and how many genes?
dim(caulobacter)

# p-values from Fisher's g test
pval.caulobacter <- fisher.g.test(caulobacter)
pval.caulobacter

# compute Fdr and fdr values
fdr.out <- fdrtool(pval.caulobacter, statistic="pvalue")

# how many significant?
sum(fdr.out$qval < 0.05) # tail area-based Fdr
sum(fdr.out$lfdr < 0.2)  # density-based local fdr
```

---

is.constant *Simple Check for Constant Time Series*

Description

is.constant is a utility function that checks whether a time series is constant.

Usage

```
is.constant(x)
```

Arguments

x	vector or matrix with time series data (one time series per column)

Value

A vector with a boolean statement (TRUE or FALSE) for each time series.

Author(s)

Korbinian Strimmer (https://www.strimmerlab.org/).

Examples

```
# load GeneCycle library
library("GeneCycle")

# load data set
data(caulobacter)

# any constant genes?
sum(is.constant(caulobacter))

# but here:
series.1 <- rep(1, 10)
series.2 <- seq(1, 10)
is.constant( cbind(series.1, series.2) )
```

---

periodogram *Periodogram Power Spectral Density*

Description

periodogram is a wrapper function for spectrum with some special options set. It returns the power spectral density, i.e. the squared modulus of the Fourier coefficient divided by the length of the series, for multiple time series as well as the corresponding Fourier frequencies. The frequencies range between 0 and the Nyquist critical frequency fc = frequency(x)/2.

periodogram is used by the functions avgp and fisher.g.test. For general periodogram functions please refer to spectrum.

Usage

```
periodogram(x, method = "builtin")
```

Arguments

x	vector or matrix containing the time series data (one time series per column)
method	a string that specifies which method should be used to compute the spectral density: "builtin" employs the function spectrum with the options taper=0, plot=FALSE, fast=FALSE, detrend=FALSE, and demean=TRUE; "clone" employs directly the Fourier transform function fft (with sames results as "builtin"); and "smooth" uses the function spectrum with options as above plus span=3.

Value

A list object with the following components:

spec	A vector or matrix with the estimated power spectral densities (one column per time series).
freq	A vector with frequencies f ranging from 0 to fc (if the sampling rate frequency(x)) equals 1 then fc = 0.5). Angular frequencies may be obtained by multiplication with 2*pi (i.e. omega = 2*pi*f).

Author(s)

Konstantinos Fokianos and Korbinian Strimmer (https://www.strimmerlab.org/).

See Also

spectrum, avgp, fisher.g.test.

Examples

```
# load GeneCycle library
library("GeneCycle")

# load data set
data(caulobacter)

# how many genes and how many samples?
dim(caulobacter)

# periodograms of the first 10 genes
periodogram(caulobacter[,1:10])
```

---

robust.g.test *Robust g Test for Multiple (Genetic) Time Series*

Description

robust.g.test calculates the p-value(s) for a robust nonparametric version of Fisher's g-test (1929). Details of this approach are described in Ahdesmaki et al. (2005), along with an extensive discussion of its application to gene expression data. From GeneCycle 1.1.0 on the robust regression based method published in Ahdesmaki et al. (2007) is also implemented (using Tukey's biweight based M-estimation/regression.)

robust.spectrum computes a robust rank-based estimate of the periodogram/correlogram - see Ahdesmaki et al. (2005) for details. Alternatively it can also be used (since GeneCycle 1.1.0) for evaluating the robust regression based spectral estimates, suitable for processing non-uniformly sampled data (unknown periodicity time: return spectral estimates, known periodicity time: return p-values).

Usage

```
robust.g.test(y, index, perm = FALSE, x, noOfPermutations = 300,
algorithm=c("rank", "regression"), t)
robust.spectrum(x, algorithm = c("rank", "regression"), t,
periodicity.time = FALSE, noOfPermutations = 300)
```

Arguments

y	the matrix consisting of the spectral estimates as column vectors
index	an index to the spectral estimates (RANK BASED APPROACH ONLY; for specifying a periodicity time in the regression approach, see the parameter periodicity.time) that is to be used in the testing for periodicity. If index is missing for the rank based approach, the maximum component of the spectral estimate is used in testing (regardless of the frequency of this maximum)
periodicity.time	time (same units as in vector t) of period where periodicity will be detected (ROBUST REGRESSION BASED APPROACH ONLY) that is to be used in the search for periodicity. If periodicity.time is not given for the regression based approach, the whole spectrum is evaluated (more time consuming) and the maximum periodogram ordinate will be investigated
perm	if perm is FALSE, a simulated distribution for the g-statistic is used (applies to the rank based approach only). If per perm is TRUE, permutation tests are used to find the distribution of the g-statistic for each time series separately. With the regression based approach (Ahdesmaki et al. 2007) permutation tests will always be used
x	a matrix consisting of the time series as column vectors. In robust.g.test only needed if permutation tests are used
noOfPermutations	number of permutations that are used for each time series (default = 300)
algorithm	rank corresponds to the rank based approach (Ahdesmaki et al. 2005) and regression for the regression based approach (Ahdesmaki et al. 2007), which is more suitable for time series with non-uniform sampling (default = rank)
t	sampling time vector (only for the regression based approach)

Details

Application of `robust.g.test` can be very computer intensive, especially the production of the distribution of the test statistics may take a lot of time. Therefore, this distribution (dependening on the length of the time series) is stored in an external file to avoid recomputation (see example below). When applying permutation tests no external file is used but the computation time will always be high.

For the general idea behind the Fisher's g test also see `fisher.g.test` which implements an analytic approach for g-testing. This is faster but not robust and also assumes Gaussian noise.

Note that when using the regression based approach there will regularly be warnings about the non-convergence of the regression (iteration limit default at 20 cycles in rlm).

Value

`robust.g.test` returns a list of p-values. `robust.spectrum` returns a matrix where the column vectors correspond to the spectra corresponding to each time series. As an exception, if the robust regression based approach (Ahdesmaki et al. 2007) is used with a known periodicity time, the function robust.spectrum returns p-values (computation will take a lot of time depending on how many permutations are used per time series and time series length).

Author(s)

Miika Ahdesmaki (<miika.ahdesmaki@gmail.com>).

References

Fisher, R.A. (1929). Tests of significance in harmonic analysis. *Proc. Roy. Soc. A*, **125**, 54–59.

Ahdesmaki, M., Lahdesmaki, H., Pearson, R., Huttunen, H., and Yli-Harja O. (2005). *BMC Bioinformatics* **6**:117. https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-6-117

Ahdesmaki, M., Lahdesmaki, H., Gracey, A., Shmulevich, I., and Yli-Harja O. (2007). *BMC Bioinformatics* **8**:233. https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-8-233

See Also

fdrtool, fisher.g.test.

Examples

```
## Not run:

# load GeneCycle library
library("GeneCycle")

# load data set
data(caulobacter)

# how many samples and and how many genes?
dim(caulobacter)
```

```
# robust, rank-based spectral estimator applied to first 5 genes
spe5 = robust.spectrum(caulobacter[,1:5])

# g statistics can be computed from the spectrum (internal use mostly
# but can be checked here)
## g.statistic(spe5)

# robust p-values, use Monte Carlo simulation (not permutation tests)
# to estimate the null hypothesis distribution
pval = robust.g.test(spe5)  # generates a file with the name "g_pop_length_11.txt"
pval = robust.g.test(spe5)  # second call: much faster..

pval

# robust p-values, now look at index 4 (index can be anything from 1
# (DC-level) to N (length of the time series and highest frequency))
pval = robust.g.test(spe5, 4)  # generates a file
pval = robust.g.test(spe5, 4)  # second call: much faster..


pval

# delete the external files
unlink("g_pop_length_11.txt")
unlink("g_pop_length_11indexed.txt")

#
# Next let us see how the robust regression based approach can be
# applied (Ahdesmaki et al. 2007)
# First: Unknown frequencies
t=c(0,15,30,45,60,75,90,105,120,135,150)
y = robust.spectrum(x=caulobacter[,1:5],algorithm="regression", t=t)
pvals = robust.g.test(y = y, perm=TRUE, x=caulobacter[,1:5],
noOfPermutations = 50, algorithm = "regression", t=t)

pvals

#
# The following example illustrates how to use the regression based
# method if we have prior knowledge about the frequency/period time
# of periodicity
t = 0:9 # time indices
t = t + runif(10)-0.5 # make time indices non-uniform
A = 0.5 * matrix(rnorm(50),10,5)       # create random time series (no outliers)
A[,5]=A[,5]+matrix(sin(0.5*pi*t),10,1) # superimpose a sinusoidal
periodicity.time=4                     # where to look for periodicity
# note that now the function robust.spectrum returns the p-values (in
# all other cases it will return spectral estimates):
pvals=robust.spectrum(x=A,algorithm="regression",
t=t,periodicity.time=periodicity.time, noOfPermutations=50)
pvals  # 5th p-value is smallish, as expected
```

```
## End(Not run)
```

Index