Package ‘limorhyde2’

March 8, 2022

Type Package

Title Quantify Rhythmicity and Differential Rhythmicity in Genomic Data

Version 0.0.7

Description Fit linear models based on periodic splines, moderate model coefficients using multivariate adaptive shrinkage, then compute properties of the moderated curves.

URL https://limorhyde2.hugheylab.org,
https://github.com/hugheylab/limorhyde2

License GPL-2

Encoding UTF-8

LazyData true

RoxygenNote 7.1.2

Depends R (>= 3.6)

Imports abind (>= 1.4-5), ashr (>= 2.2-54), checkmate (>= 2.0.0), data.table (>= 1.12.8), DESeq2 (>= 1.30.0), foreach (>= 1.5.0), HDInterval (>= 0.2.2), iterators (>= 1.0.12), limma (>= 3.42.2), mashr (>= 0.2.50), pbs (>= 1.1), zeallot (>= 0.1.0)

Suggests cowplot (>= 1.1.1), knitr, doParallel (>= 1.0.15), ggplot2 (>= 3.3.5), glue (>= 1.6.1), qs (>= 0.24.1), rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation no

Author Jake Hughey [aut, cre], Dora Obodo [aut], Elliot Outland [aut]

Maintainer Jake Hughey <jakehughey@gmail.com>

Repository CRAN

Date/Publication 2022-03-08 20:20:02 UTC
getDiffRhythmStats

Description
This function computes differences in rhythmicity between fitted curves for a given pair of conditions.

Usage
getDiffRhythmStats(fit, rhyStats, conds = fit$conds, dopar = TRUE)

Arguments

fit A limorhyde2 object containing data from multiple conditions.
rhyStats A data.table of rhythmic statistics, as returned by getRhythmStats(), for fitted models in fit.
conds A character vector indicating the conditions to compare pairwise, by default all conditions in fit.
dopar Logical indicating whether to run calculations in parallel if a parallel backend is already set up, e.g., using doParallel::registerDoParallel(). Recommended to minimize runtime.

Value
A data.table containing the following differential rhythm statistics:

- mean_mesor
- mean_peak_trough_amp
- mean_rms_amp (only calculated if rms to getRhythmStats() was TRUE)
- diff_mesor
getExpectedMeas

- diff_peak_trough_amp
- diff_rms_amp (only calculated if rms to getRhythmStats() was TRUE)
- diff_peak_phase: circular difference between -fit$period/2 and fit$period/2
- diff_trough_phase: circular difference between -fit$period/2 and fit$period/2
- diff_rhy_dist: Euclidean distance between polar coordinates (peak_trough_amp, peak_phase)
- rms_diff_rhy: root mean square difference in mean-centered fitted curves (only calculated if rms to getRhythmStats() was TRUE)

The stats will be based on the value for cond2 minus the value for cond1. The rows of the data.table depend on the 'fitType' attribute of rhyStats:

- 'fitType' is 'posterior_mean' or 'raw': one row per feature per pair of conditions.
- 'fitType' is 'posterior_samples': one row per feature per posterior sample per pair of conditions.

See Also
getRhythmStats(), getStatsIntervals()

Examples

```r
library('data.table')

# rhythmicity in one condition
y = GSE54650$y
metadata = GSE54650$metadata

fit = getModelFit(y, metadata)
fit = getPosteriorFit(fit)
rhyStats = getRhythmStats(fit, features = c('13170', '13869'))

# rhythmicity and differential rhythmicity in multiple conditions
y = GSE34018$y
metadata = GSE34018$metadata

fit = getModelFit(y, metadata, nKnots = 3L, condColname = 'cond')
fit = getPosteriorFit(fit)
rhyStats = getRhythmStats(fit, features = c('13170', '12686'))
diffRhyStats = getDiffRhythmStats(fit, rhyStats)
```

getExpectedMeas  
Compute expected measurements from fitted models

Description

This function computes expected measurements (corresponding to the fitted curves) for the specified times and features in all combinations of conditions and covariates (if they exist).
getExpectedMeas

Usage

getExpectedMeas(
  fit,
  times,
  fitType = c("posterior_mean", "posterior_samples", "raw"),
  features = NULL,
  dopar = TRUE
)

Arguments

  fit          A 'limorhyde2' object.
  times        Numeric vector of times, in units of fit$metadata[[fit$timeColname]].
  fitType      String indicating which fitted models to use to compute the expected measurements. A typical analysis using limorhyde2 will be based on 'posterior_mean', the default.
  features     Vector of names, row numbers, or logical values for subsetting the features. NULL indicates all features.
  dopar        Logical indicating whether to run calculations in parallel if a parallel backend is already set up, e.g., using doParallel::registerDoParallel(). Recommended to minimize runtime.

Value

A data.table.

See Also

getModelFit(), getPosteriorFit(), getPosteriorSamples(), getExpectedMeasIntervals()

Examples

library('data.table')

y = GSE34018$y
metadata = GSE34018$metadata

fit = getModelFit(y, metadata)
fit = getPosteriorFit(fit)

measObs = mergeMeasMeta(y, metadata, features = c('13170', '12686'))
measFitMean = getExpectedMeas(
  fit, times = seq(0, 24, 0.5), features = c('13170', '12686'))
getExpectedMeasIntervals

Compute credible intervals for expected measurements

Description
This function uses posterior samples to quantify uncertainty in the expected measurements from fitted models.

Usage
getExpectedMeasIntervals(expectedMeas, mass = 0.9, method = c("eti", "hdi"))

Arguments
- `expectedMeas`: A `data.table` of expected measurements for posterior samples, as returned by `getExpectedMeas()`.
- `mass`: Number between 0 and 1 indicating the probability mass for which to calculate the intervals.
- `method`: String indicating the type of interval: 'eti' for equal-tailed using `stats::quantile()`, or 'hdi' for highest density using `HDInterval::hdi()`.

Value
A `data.table` containing lower and upper bounds of the expected measurement for each combination of feature, time, and possibly condition and covariate.

See Also
- `getExpectedMeas()`, `getStatsIntervals()`

Examples
library('data.table')
y = GSE34018$y
metadata = GSE34018$metadata

fit = getModelFit(y, metadata)
fit = getPosteriorFit(fit)
fit = getPosteriorSamples(fit, nPosteriorSamples = 10L)

measFitSamps = getExpectedMeas(
  fit, times = seq(0, 24, 0.5), fitType = 'posterior_samples',
  features = c('13170', '12686'))
measFitInts = getExpectedMeasIntervals(measFitSamps)
getModelFit

Fit linear models for rhythmicity in one or more conditions

Description

This is the first step in an analysis using limorhyde2, the second is to moderate the fits using getPosteriorFit().

Usage

getModelFit(
  y,
  metadata,
  period = 24,
  nKnots = 4L,
  timeColname = "time",
  condColname = NULL,
  covarColnames = NULL,
  sampleColname = "sample",
  nShifts = 3L,
  method = c("trend", "voom", "deseq2"),
  lmFitArgs = list(),
  eBayesArgs = if (method == "trend") list(trend = TRUE) else list(),
  DESeqArgs = list(),
  keepLmFits = FALSE
)

Arguments

y
Matrix-like object of measurements, with rows corresponding to features and columns to samples.

metadata
data.frame containing experimental design information for each sample. Rows of metadata must correspond to columns of y. Row names are ignored.

period
Number specifying the period for the time variable, in the same units as the values in the timeColname column.

nKnots
Number of internal knots for the periodic spline for the time variable. Use NULL to fit a cosinor-based model instead of a spline-based model.

timeColname
String indicating the column in metadata containing the time at which each sample was acquired.

condColname
String indicating the column in metadata containing the condition in which each sample was acquired. NULL indicates all samples came from the same condition. If not NULL, the model will include main effects and interactions with the terms for time.

covarColnames
Character vector indicating the columns in metadata containing covariates to include in the model. NULL indicates no covariates.
getModelFit

sampleColname  String indicating the column in metadata containing the name of each sample, which must correspond to the column names of y.
nShifts  Number of shifted models to fit. Only used for periodic splines, not for cosinor. Do not change from the default unless you know what you’re doing.
method  String indicating method to estimate model coefficients. For microarray data, use ‘trend’. For RNA-seq count data, use ‘voom’ or ‘deseq2’.
lmFitArgs  List of arguments passed to limma::lmFit().
eBayesArgs  List of arguments passed to limma::eBayes().
DESeqArgs  List of arguments passed to DESeq2::DESeq().
keepLmFits  Logical indicating whether to keep the complete fit objects from limma or DESeq2. Not needed by any functions in limorhyde2.

Value

A limorhyde2 object with elements:

- metadata: As supplied above, converted to a data.table.
- timeColname: As supplied above.
- condColname: As supplied above.
- covarColnames: As supplied above.
- coefficients: Matrix with rows corresponding to features and columns to model terms, including all shifted models.
- shifts: Numeric vector indicating amount by which timepoints were shifted for each shifted model.
- period: As supplied above.
- conds: If condColname is not NULL, a vector of unique values of the condition variable.
- nKnots: Number of knots, where 2 indicates a cosinor-based model.
- nConds: Number of conditions.
- nCovs: Number of covariates.
- lmFits: If keepLmFits is TRUE, a list of objects from limma or DESeq2, with length equal to length of the shifts element.

See Also

ggetPosteriorFit()

Examples

library('data.table')

# rhythmicity in one condition
y = GSE54650$y
metadata = GSE54650$metadata

fit = getModelFit(y, metadata)
fit = getPosteriorFit(fit)

rhyStats = getRhythmStats(fit, features = c('13170', '13869'))

# rhythmicity and differential rhythmicity in multiple conditions
y = GSE34018$y
metadata = GSE34018$metadata

fit = getModelFit(y, metadata, nKnots = 3L, condColname = 'cond')
fit = getPosteriorFit(fit)

rhyStats = getRhythmStats(fit, features = c('13170', '12686'))
diffRhyStats = getDiffRhythmStats(fit, rhyStats)

---

**getPosteriorFit**

Compute posterior fit for linear models for rhythmicity

**Description**

This is the second step in an analysis using limorhyde2, the first is to fit linear models using `getModelFit()`. This function obtains posterior estimates of coefficients using multivariate adaptive shrinkage (mash), which learns patterns in the data and accounts for noise in the original fits. The defaults for arguments should work well in most cases, so only change them if you know what you’re doing.

**Usage**

```r
getPosteriorFit(
  fit,
  covMethod = c("data-driven", "canonical", "both"),
  getSigResArgs = list(),
  npc = fit$nKnots,
  covEdArgs = list(),
  overwrite = FALSE,
  ...
)
```

**Arguments**

- **fit**: A limorhyde2 object.
- **covMethod**: String indicating the type(s) of covariance matrices to use for the mash fit.
- **getSigResArgs**: List of arguments passed to `mashr::get_significant_results()`. Only used if covMethod is 'data-driven' or 'both'.
- **npc**: Number of principal components passed to `mashr::cov_pca()`. Only used if covMethod is 'data-driven' or 'both'.
- **covEdArgs**: List of arguments passed to `mashr::cov_ed()`. Only used if covMethod is 'data-driven' or 'both'.
- **overwrite**: Logical for whether to recompute the mash fit if it already exists.
- **...**: Additional arguments passed to `mashr::mash()`. 
getPosteriorSamples

Value

A limorhyde2 object containing everything in fit with added or updated elements:

- mashData: list of mash data objects
- mashFits: list of mash fit objects
- mashCoefficients: Matrix of posterior mean coefficients, with rows corresponding to features and columns to model terms.
- mashIdx: Vector indicating which model terms were included in the mash fit.

See Also

getModelFit(), getRhythmStats(), getExpectedMeas()

Examples

```r
library('data.table')

# rhythmicity in one condition
y = GSE54650$y
metadata = GSE54650$metadata

fit = getModelFit(y, metadata)
fit = getPosteriorFit(fit)
rhyStats = getRhythmStats(fit, features = c('13170', '13869'))

# rhythmicity and differential rhythmicity in multiple conditions
y = GSE34018$y
metadata = GSE34018$metadata

fit = getModelFit(y, metadata, nKnots = 3L, condColname = 'cond')
fit = getPosteriorFit(fit)
rhyStats = getRhythmStats(fit, features = c('13170', '12686'))
diffRhyStats = getDiffRhythmStats(fit, rhyStats)
```

getPosteriorSamples

**Draw samples from posterior distributions of fitted models**

Description

This is an optional step in an analysis using limorhyde2, and is useful for quantifying uncertainty in posterior estimates of fitted curves and rhythmic statistics. The function calls `mashr::mash_compute_posterior_matrices`.

Usage

getPosteriorSamples(fit, nPosteriorSamples = 200L, overwrite = FALSE)
getRhythmStats

Compute rhythm statistics from fitted models

Description

This function uses stats::optim() to compute various properties of fitted curves with respect to time, potentially in each condition and for each posterior sample, and adjusting for any covariates.

Usage

getRhythmStats(
  fit,
  fitType = c("posterior_mean", "posterior_samples", "raw"),
  features = NULL,
  dopar = TRUE,
  rms = FALSE
)
getRhythmStats

Arguments

- **fit**
  A `limorhyde2` object.

- **fitType**
  String indicating which fitted models to use to compute the rhythmic statistics. A typical analysis using `limorhyde2` will be based on `posterior_mean`, the default.

- **features**
  Vector of names, row numbers, or logical values for subsetting the features. `NULL` indicates all features.

- **dopar**
  Logical indicating whether to run calculations in parallel if a parallel backend is already set up, e.g., using `doParallel::registerDoParallel()`. Recommended to minimize runtime.

- **rms**
  Logical indicating whether to calculate `rms_amp`.

Value

A `data.table` containing the following rhythm statistics:

- `peak_phase`: time between 0 and `fit$period` at which the peak or maximum value occurs
- `peak_value`
- `trough_phase`: time between 0 and `fit$period` at which the trough or minimum value occurs
- `trough_value`
- `peak_trough_amp`: `peak_value` - `trough_value`
- `rms_amp`: root mean square difference between fitted curve and mean value between time 0 and `fit$period` (only calculated if `rms` is TRUE)
- `mesor`: mean value between time 0 and `fit$period`

The rows of the `data.table` depend on the `fit` object and `fitType`:

- `fit` contains data from one condition and `fitType` is `posterior_mean` or `raw`: one row per feature.
- `fit` contains data from one condition and `fitType` is `posterior_samples`: one row per feature per posterior sample.
- `fit` contains data from multiple conditions and `fitType` is `posterior_mean` or `raw`: one row per feature per condition.
- `fit` contains data from multiple conditions and `fitType` is `posterior_samples`: one row per feature per condition per posterior sample.

See Also

`getModelFit()`, `getPosteriorFit()`, `getPosteriorSamples()`, `getDiffRhythmStats()`, `getStatsIntervals()`
getStatsIntervals

Compute credible intervals for rhythm or differential rhythm statistics

Description
This function uses posterior samples to quantify uncertainty in the properties of fitted curves.

Usage
getStatsIntervals(posteriorStats, mass = 0.9, method = c("eti", "hdi"))

Arguments

- posteriorStats: A data.table of statistics for posterior samples, as returned by `getRhythmStats()` or `getDiffRhythmStats()`.
- mass: Number between 0 and 1 indicating the probability mass for which to calculate the intervals.
- method: String indicating the type of interval: 'eti' for equal-tailed using `stats::quantile()`, or 'hdi' for highest density using `HDInterval::hdi()`.

Value
A data.table containing lower and upper bounds of various statistics for each feature or each feature-condition pair. For `peak_trough_amp` and `rms_amp`, a negative lower bound indicates a rhythm of the opposite phase.

See Also
`getRhythmStats()`, `getDiffRhythmStats()`, `getExpectedMeasIntervals()`
Examples

```r
library('data.table')

y = GSE54650$y
metadata = GSE54650$metadata

fit = getModelFit(y, metadata)
fit = getPosteriorFit(fit)
fit = getPosteriorSamples(fit, nPosteriorSamples = 10L)

rhyStatsSamps = getRhythmStats(fit, features = c('13170', '13869'), fitType = 'posterior_samples')
rhyStatsInts = getStatsIntervals(rhyStatsSamps)
```

Description

Data are based on total RNA, measured by microarray, obtained from livers of wild-type and liver-specific Reverb alpha/beta double knockout mice at various times in a 12h:12h light:dark cycle. To save space and time, the data include only a subset of genes, and so are mainly useful for examples of how to use limorhyde2.

Usage

GSE34018

Format

A list with two elements:

- **y**: Matrix of normalized, log-transformed expression values. Rows correspond to genes (row-names are Entrez Gene IDs) and columns to samples.
- **metadata**: data.table with one row per sample. time is in hours.

Source


See Also

GSE54650, getModelFit()
mergeMeasMeta

GSE54650  Gene expression data for GSE54650

Description
Data are based on total RNA, measured by microarray, obtained from livers of wild-type mice at various times after transfer to constant darkness. To save space and time, the data include only a subset of genes, and so are mainly useful for examples of how to use limorhyde2.

Usage
GSE54650

Format
A list with two elements:

- \( y \): Matrix of normalized, log-transformed expression values. Rows correspond to genes (row-names are Entrez Gene IDs) and columns to samples.
- \( \text{metadata} \): data.table with one row per sample. time is in hours.

Source

See Also
GSE34018, getModelFit()

mergeMeasMeta  Merge measurements and metadata

Description
This function is useful for plotting time-courses for individual features.

Usage
mergeMeasMeta(y, metadata, features = NULL, sampleColname = "sample")
Arguments

- **y**: Matrix-like object of measurements, with rows corresponding to features and columns to samples.
- **metadata**: data.frame containing experimental design information for each sample. Rows of metadata must correspond to columns of y. Row names are ignored.
- **features**: Vector of names, row numbers, or logical values for subsetting the features. NULL indicates all features.
- **sampleColname**: String indicating the column in metadata containing the name of each sample, which must correspond to the column names of y.

Value

- A `data.table` with one row for each sample-feature pair.

See Also

- `getExpectedMeas()`

Examples

```r
library('data.table')

y = GSE34018$y
metadata = GSE34018$metadata

fit = getModelFit(y, metadata)
fit = getPosteriorFit(fit)

measObs = mergeMeasMeta(y, metadata, features = c('13170', '12686'))
measFitMean = getExpectedMeas(
  fit, times = seq(0, 24, 0.5), features = c('13170', '12686'))
```
Index

* datasets
  - GSE34018, 13
  - GSE54650, 14

DESeq2::DESeq(), 7

doParallel::registerDoParallel(), 2, 4, 11

getDiffRhythmStats, 2
getDiffRhythmStats(), 11, 12
getExpectedMeas, 3
getExpectedMeas(), 5, 9, 10, 15
getExpectedMeasIntervals, 5
getExpectedMeasIntervals(), 4, 12
getModelFit, 6
getModelFit(), 4, 8, 9, 11, 13, 14
getPosteriorFit, 8
getPosteriorFit(), 4, 6, 7, 10, 11
getPosteriorSamples, 9
getPosteriorSamples(), 4, 11
getRhythmStats, 10
getRhythmStats(), 2, 3, 9, 10, 12
getStatsIntervals, 12
getStatsIntervals(), 3, 5, 10, 11
GSE34018, 13, 14
GSE54650, 13, 14

HDInterval::hdi(), 5, 12

limma::eBayes(), 7
limma::lmFit(), 7

mashr::cov_ed(), 8
mashr::cov_pca(), 8
mashr::get_significant_results(), 8
mashr::mash(), 8
mashr::mash_compute_posterior_matrices(), 9
mergeMeasMeta, 14

stats::optim(), 10
stats::quantile(), 5, 12