

# Package ‘DiscoRhythm’

December 30, 2024

**Title** Interactive Workflow for Discovering Rhythmicity in Biological Data

**Version** 1.22.0

**Description** Set of functions for estimation of cyclical characteristics, such as period, phase, amplitude, and statistical significance in large temporal datasets. Supporting functions are available for quality control, dimensionality reduction, spectral analysis, and analysis of experimental replicates. Contains a R Shiny web interface to execute all workflow steps.

**Depends** R (>= 3.6.0)

**License** GPL-3

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.1.1

**VignetteBuilder** knitr

**Imports** matrixTests, matrixStats, MetaCycle (>= 1.2.0), data.table, ggplot2, ggExtra, dplyr, broom, shiny, shinyBS, shinycssloaders, shinydashboard, shinyjs, BiocStyle, rmarkdown, knitr, kableExtra, magick, VennDiagram, UpSetR, heatmaply, viridis, plotly, DT, gridExtra, methods, stats, SummarizedExperiment, BiocGenerics, S4Vectors, zip, reshape2

**Suggests** testthat

**SystemRequirements** To generate html reports pandoc (<http://pandoc.org/installing.html>) is required.

**biocViews** Software, TimeCourse, QualityControl, Visualization, GUI, PrincipalComponent

**URL** <https://github.com/matthewcarlucci/DiscoRhythm>

**BugReports** <https://github.com/matthewcarlucci/DiscoRhythm/issues>

**git\_url** <https://git.bioconductor.org/packages/DiscoRhythm>

**git\_branch** RELEASE\_3\_20

**git\_last\_commit** c84efd0

**git\_last\_commit\_date** 2024-10-29

**Repository** Bioconductor 3.20

**Date/Publication** 2024-12-30

**Author** Matthew Carlucci [aut, cre],  
 Algimantas Kriščiūnas [aut],  
 Haohan Li [aut],  
 Povilas Gibas [aut],  
 Karolis Koncevičius [aut],  
 Art Petronis [aut],  
 Gabriel Oh [aut]

**Maintainer** Matthew Carlucci <Matthew.Carlucci@camh.ca>

## Contents

DiscoRhythm-package . . . . .	2
checkJTKperiod . . . . .	3
checkPeriod . . . . .	4
discoApp . . . . .	4
discoBatch . . . . .	5
discoCheckInput . . . . .	6
discoColors . . . . .	7
discoDesignSummary . . . . .	8
discoDFtoSE . . . . .	8
discoGetSimu . . . . .	9
discoODAexclusionMatrix . . . . .	10
discoODAid2name . . . . .	11
discoODAs . . . . .	11
discoParseMeta . . . . .	13
discoPCA . . . . .	14
discoPCAgetOutliers . . . . .	14
discoPeriodDetection . . . . .	15
discoQC . . . . .	15
discoShinyHandler . . . . .	17
fisherExact . . . . .	17
inferFilteredDesign . . . . .	18
lmCSmat . . . . .	18
lmCSmatNoNA . . . . .	19
PeriodDetection_range . . . . .	20
sincos . . . . .	21
theme_disco . . . . .	21

<b>Index</b>	<b>22</b>
--------------	-----------

---

DiscoRhythm-package	<i>DiscoRhythm: Interactive Workflow for Discovering Rhythmicity in Biological Data</i>
---------------------	---

---

## Description

Set of functions for estimation of cyclical characteristics, such as period, phase, amplitude, and statistical significance in large temporal datasets. Supporting functions are available for quality control, dimensionality reduction, spectral analysis, and analysis of experimental replicates. Contains a R Shiny web interface to execute all workflow steps.

**Details**

The main function to run DiscoRhythm in batch mode is [discoBatch()]. Or to access the DiscoRhythm web application use [discoApp()].

**Author(s)**

**Maintainer:** Matthew Carlucci <Matthew.Carlucci@camh.ca>

Authors:

- Algimantas Kriščiūnas <algimantas.krisciunas@gmc.vu.lt>
- Haohan Li <Xavier.li@edu.uwaterloo.ca>
- Povilas Gibas <povilas.gibas@bti.vu.lt>
- Karolis Koncevičius <karolis.koncevicius@gmc.vu.lt>
- Art Petronis <Art.Petronis@camh.ca>
- Gabriel Oh <Gabriel.Oh@camh.ca>

**See Also**

Useful links:

- <https://github.com/matthewcarlucci/DiscoRhythm>
- Report bugs at <https://github.com/matthewcarlucci/DiscoRhythm/issues>

---

checkJTKperiod

*Validate Detection Period for JTK Cycle*

---

**Description**

Validate Detection Period for JTK Cycle

**Usage**

```
checkJTKperiod(time, period)
```

**Value**

logical stating whether the period is appropriate for JTK Cycle for this dataset.

---

checkPeriod	<i>Validate Detection Period</i>
-------------	----------------------------------

---

**Description**

Validate Detection Period

**Usage**

```
checkPeriod(time, period, min_n_values = 3)
```

**Arguments**

time	numeric vector of sample collection times.
period	hypothesized period.
min_n_values	numeric value specifying minimal number of unique "time MODULO period" values.

**Value**

logical indicating whether the period is suitable for testing given the sampling times of the dataset.

---

discoApp	<i>Launch the DiscoRhythm Shiny Application</i>
----------	---

---

**Description**

This launches the web interface to DiscoRhythm containing all analysis tools. The vignette contains details on usage.

**Usage**

```
discoApp(ncores = 1, port = 3838, local = TRUE)
```

**Arguments**

ncores	numeric, number of cores to use for parallelized tasks. Currently, only used in oscillation detection function <code>discoODAs</code> .
port	numeric, port to run the shiny application on. Sets shiny.port option.
local	logical, set to FALSE for public server mode to reduce file size limits.

**Value**

Nothing is returned by this function.

**Examples**

```
## Not run:
discoApp()

## End(Not run)
```

## Description

Execute the DiscoRhythm workflow with one command to obtain the results of oscillation detection (discoODAs) and optionally generate an html report with data visualizations from an Rmarkdown template. See the DiscoRhythm vignette for more details on the analysis procedures.

## Usage

```
discoBatch(
  indata,
  report = NULL,
  outdata = TRUE,
  ncores = 1,
  timeType = "circular",
  main_per = 24,
  cor_threshold = 3,
  cor_method = "pearson",
  cor_threshType = "sd",
  pca_threshold = 3,
  pca_scale = TRUE,
  pca_pcToCut = paste0("PC", seq_len(4)),
  aov_method = "None",
  aov_pcut = 0.05,
  aov_Fcut = 0,
  avg_method = "Median",
  osc_method = NULL,
  osc_period = 24
)
```

## Arguments

indata	SummarizedExperiment or data.frame, see the vignette for the specific formats expected for each of these input types. discoParseMeta.
report	character, if !is.null(report) an html report with
outdata	logical, whether to return the final discoODAs (note if run with is.null(report) discoBatch will return nothing).
ncores	numeric, number of cores to use for parallelized tasks. Currently, only used in oscillation detection function discoODAs.
timeType	character, nature of the sample times provided (one of "circular" or "linear").
main_per	numeric, the length of the main hypothesized period (e.g. 24hr for circadian experiments). Used in discoPeriodDetection.
cor_threshold	numeric, threshold used in inter-sample correlation analysis for outlier detection. Either in units of correlation coefficient or standard deviations from the mean (see cor_threshType).
cor_method	character, which correlation method to use for outlier removal (see cor for more details).

cor_threshType	character, one of "sd" or "value" indicating whether cor_threshold should be set by absolute correlation coefficient or by standard deviations from the mean of all samples.
pca_threshold	numeric, the number of standard deviations to set as the threshold for outlier detection in PCA outlier removal.
pca_scale	logical, whether to scale the data prior to PCA.
pca_pcToCut	character, names of which PCs to use for outlier detection (e.g. "PC1", "PC2" etc.).
aov_method	character, method to use for ANOVA. One of: "Equal Variance", "Welch", or "None".
aov_pcut	numeric, p-value cutoff used to select rows with statistically significant signal-to-noise.
aov_Fcut	numeric, F-statistic cutoff used to select rows with high signal-to-noise based on magnitude.
avg_method	character, method for averaging technical replicates. One of: "Median", "Mean", "Random", or "None".
osc_method	character, vector of oscillation detection algorithms to apply to the data. Methods that are detminded to be innappropraite for the experimental design (using the discoODAexclusionMatrix) will be ignored. If is.null(osc_method) all suitable methods will be executed.
osc_period	numeric, a fixed period to use for oscillation detection using all methods.

**Value**

returns the results of discoODAs

**See Also**

discoODAs, discoRepAnalysis, discoPeriodDetection, discoPCAoutliers, discoInterCorOurliers

**Examples**

```

indata <- discoGetSimu()

# Batch execute (on demo data) to generate a DiscoRhythm_report.html report.
# Returns the results of discoODAs
discoODAs <- discoBatch(indata,
report="DiscoRhythm_report.html",
osc_method="CS")

```

---

discoCheckInput

*Import Data for DiscoRhythm Analysis*


---

**Description**

Performs various checks and cleaning operations on the input data.

## Usage

```
discoCheckInput(se, n_min_sample = 3)
```

## Arguments

se	SummarizedExperiment, the main data object used by DiscoRhythm expected to contain se\$ID, se\$ReplicateID, se\$Time sample metadata and non-null row-names. See the vignette for more details.
n_min_sample	numeric value specifying minimal number of samples needed to perform analysis.

## Details

Rows containing NA's or all constant values are removed. If matrix values are character it will be attempted to convert them to numeric. If input is not a matrix it will be converted using `as.matrix()`. User will be warned if row IDs contain duplicate entries.

## Value

SummarizedExperiment checked for errors and modified as needed

## Examples

```
se <- discoGetSimu(TRUE)
se_clean <- discoCheckInput(se)
```

---

discoColors	<i>Color pallete used by DiscoRhythm for plotting. This palette is duplicated in inst/app/www/custom_styles.css for application to the shiny app.</i>
-------------	---

---

## Description

Color pallete used by DiscoRhythm for plotting. This palette is duplicated in `inst/app/www/custom_styles.css` for application to the shiny app.

## Usage

```
discoColors
```

## Format

An object of class `list` of length 14.

---

discoDesignSummary      *Summarize the experimental design*

---

### Description

Using sample times and biological sample Ids, constructs a summary table of the number of total samples at each timepoint and additionally summarizes the number of replicates for each biological sample.

### Usage

```
discoDesignSummary(Metadata)
```

### Arguments

Metadata      data.frame of sample data, usually generated by using discoParseMeta on the column names of the Maindata data.frame. If is.null(Metadata) and Maindata is provided as input, Metadata will be generated from Maindata.

### Value

A table where the first row summarizes the number of datapoints for each timepoint and other cells indicate the number of technical replicates for a given biological sample.

### See Also

discoParseMeta

### Examples

```
# import example data
Metadata <- SummarizedExperiment::colData(discoGetSimu(TRUE))
# Summarize the experiment design
discoDesignSummary(Metadata)
```

---

discoDFtoSE      *Data formatting for DiscoRhythm*

---

### Description

Functions to import a data.frame (from the format expected by the web application discoApp()) as a SummarizedExperiment object or to export a SummarizedExperiment for use with the web application.

### Usage

```
discoDFtoSE(Maindata, Metadata = NULL, shinySession = NULL)
```

```
discoSEtoDF(se)
```



**Arguments**

Maingroup	data.frame with the first column containing row IDs and all subsequent columns containing experimental values. Columns should follow the expected naming format described in the vignette.
Metadata	data.frame of sample data, usually generated by using <code>discoParseMeta</code> on the column names of the Maingroup data.frame. If <code>is.null(Metadata)</code> and Maingroup is provided as input, Metadata will be generated from Maingroup.
shinySession	shiny session object for use only by the DiscoRhythm shiny app <code>discoApp()</code> to update the axis labels using the time value prefix.
se	SummarizedExperiment, the main data object used by DiscoRhythm expected to contain <code>se\$ID</code> , <code>se\$ReplicateID</code> , <code>se\$Time</code> sample metadata and non-null row-names. See the vignette for more details.

**Value**

`discoDFtoSE` returns a SummarizedExperiment object with `colData` containing sample metadata.  
`discoSEtoDF` returns a DiscoRhythm format data.frame.

**Examples**

```
df <- discoGetSimu()
se <- discoDFtoSE(df)

df <- discoSEtoDF(se)
```

---

`discoGetSimu`

*Read in the DiscoRhythm Simulated dataset*

---

**Description**

A convenience function to get the simulated circadian transcriptomic system data file used in DiscoRhythm for various demonstrations and tests.

**Usage**

```
discoGetSimu(as_se = FALSE)
```

**Arguments**

as_se	logical, indicates if example data should be returned as a SummarizedExperiment or data.frame.
-------	--

**Value**

The simulated demo dataset used in the DiscoRhythm web application as a data.frame or SummarizedExperiment.

**Examples**

```
indata <- discoGetSimu()
```

---

`discoODAexclusionMatrix`*Algorithm Exclusion Matrix*

---

**Description**

A small matrix indicating which algorithms should be excluded given certain experimental designs and data types.

**Usage**`discoODAexclusionMatrix`**Format**

An object of class `matrix` (inherits from `array`) with 4 rows and 7 columns.

**Examples**

```
# Code used to generate discoODAexclusionMatrix

itemNames <- c(
  "missing_value",
  "with_bio_replicate",
  "non_integer_interval",
  "uneven_interval",
  "circular_t",
  "invalidPeriod",
  "invalidJTKperiod"
)

# Creating requirements matrix, first assuming all methods are valid
# Then applying exclusion criteria of MetaCycle plus CS criteria
mat <- matrix(TRUE, nrow = 4, ncol = length(itemNames))
rownames(mat) <- c("CS", "JTK", "LS", "ARS")
colnames(mat) <- itemNames

# Exclusion criteria from MetaCycle v1.1, i.e. can algorithm handle XXX
mat[c("ARS", "JTK"), c("non_integer_interval", "uneven_interval")] <- FALSE
mat["ARS", "with_bio_replicate"] <- FALSE
mat["ARS", "missing_value"] <- FALSE
mat["JTK", "invalidJTKperiod"] <- FALSE

# Additional exclusion criteria
mat["ARS", "circular_t"] <- FALSE
mat[c("CS", "JTK", "ARS", "LS"), "invalidPeriod"] <- FALSE

discoODAexclusionMatrix <- mat
```

---

discoODAid2name      *Mapping Identifiers to Full Names*

---

### Description

A small named vector mapping oscillation detection algorithm names to a convenient identifier.

### Usage

```
discoODAid2name
```

### Format

A named vector, length 4

**names(discoODAid2name)** Identifier

**as.vector(discoODAid2name)** Full names

---

discoODAs      *Execute Oscillation Detection Using DiscoRhythm*

---

### Description

Runs specified oscillation detection algorithms (ODAs) sequentially to obtain oscillation characteristics for each row of the input data.

### Usage

```
discoGetODAs(se, method = NULL, period, circular_t = FALSE)
```

```
discoODAs(
  se,
  period = 24,
  method = c("CS", "JTK", "LS", "ARS"),
  circular_t = FALSE,
  ncores = 1
)
```

### Arguments

se	SummarizedExperiment, the main data object used by DiscoRhythm expected to contain se\$ID, se\$ReplicateID, se\$Time sample metadata and non-null row-names. See the vignette for more details.
method	character, short names of ODAs to use. If length>1 all input method names will be evaluated.
period	numeric, the hypothesized period to test for.
circular_t	logical, is time circular on some base-cycle (ex. time of day). See the DiscoRhythm vignette for details.
ncores	numeric, number of cores to parallelize with (applicable to JTK, ARSER and LS only). If 1, will execute in serial.

## Details

There are currently 4 available algorithms for rhythm detection:

- CS = Cosinor (Cornelissen,G. 2014): a.k.a “Harmonic Regression” fits a sinusoid with a free phase parameter.
- LS = Lomb-Scargle (Glynn, 2006): an approach using spectral power density.
- ARS = ARSER (Yang, 2010): removes linear trends and performs the Cosinor test.
- JTK = JTK Cycle (Hughes, 2010): non-parametric test of rhythmicity robust to outliers.

LS, ARS, and JTK results come directly from MetaCycle `meta2d()` output using the specified fixed period. ARS`mle` is set to "nomle" and no method integration is used (see `meta2d` documentation for details).

CS is implemented directly in DiscoRhythm’s `lmCSmat()` as the single-component cosinor described in Cornelissen,G. (2014).

All q-values are calculated by performing `p.adjust()` on the resulting p-values with `method="fdr"`.

Technical replicates are expected to be merged (likely by `discoRepAnalysis`) prior to usage of `discoODAs`.

The `discoGetODAs` function is called by `discoODAs` to determine if the selected methods may be used. If any methods are not valid, a warning will be thrown and only valid methods will be computed. `discoGetODAs` is not typically used directly, however, it may be called by the user to determine if the provided `SummarizedExperiment` is suitable for use with the specified methods.

## Value

A named list of results where each element is a `data.frame` for the corresponding method with rownames corresponding to the feature identifiers and columns containing estimates for:

- acrophase
- amplitude
- p-value
- q-value

Additional columns relevant to each method will be present.

## References

Yang R. and Su Z. (2010). Analyzing circadian expression data by harmonic regression based on autoregressive spectral estimation. *Bioinformatics*, **26(12)**, i168–i174.

Hughes M. E., Hogenesch J. B. and Kornacker K. (2010). JTK\_CYCLE: an efficient nonparametric algorithm for detecting rhythmic components in genome-scale data sets. *Journal of Biological Rhythms*, **25(5)**, 372–380.

Glynn E. F., Chen J. and Mushegian A. R. (2006). Detecting periodic patterns in unevenly spaced gene expression time series using Lomb-Scargle periodograms. *Bioinformatics*, **22(3)**, 310–316.

Cornelissen,G. (2014) Cosinor-based rhythmometry. *Theor. Biol. Med. Model.*, **11**, 16.

## See Also

[lmCSmat meta2d](#)

**Examples**

```

# Return valid ODAs for example dataset
discoGetODAs(discoGetSimu(as_se=TRUE),period=24)

# Import the simulated example dataset
se <- discoCheckInput(discoGetSimu(TRUE))

# Use discoRepAnalysis to average technical replicates
se_merged <- discoRepAnalysis(se,aov_pcut=1)$se

# Execute the Cosinor and JTK methods with a 24hr period
discoODAs <- discoODAs(se_merged,method=c("CS","JTK"))

# Get the index of rhythmic features detected by both methods at qvalue<0.05
idx <- which(discoODAs$CS$qvalue<0.05 & discoODAs$JTK$qvalue<0.05)

# Get the identifiers for common rhythmic features
rownames(se_merged)[idx]

```

---

discoParseMeta

*Generate Experiment Metadata*


---

**Description**

Parses the sample metadata from a vector of sample names (often column names of a Maindata format data.frame).

**Usage**

```
discoParseMeta(samplenames, shinySession = NULL)
```

**Arguments**

samplenames	character, a list of sample names following the DiscoRhythm naming convention (<prefix><Time>_<UniqueID>_<ReplicateID>).
shinySession	shiny session object for use only by the DiscoRhythm shiny app discoApp() to update the axis labels using the time value prefix.

**Details**

The regular expression used to obtain metadata is "`^([[:alpha:]]*)(\-[0-9]+[.]?[0-9]*)\_\_? ([[:alnum:]]*\.)*\_\_?([[:alnum:]]*\.)*$`"

Where each () will be used to construct the final metadata data.frame

**Value**

a data.frame containing 3 columns of metadata. ID = unique sample identity. Time = sample collection time. ReplicateID = Identifier where Time + ReplicateID indicates a biological sample ID.

**Examples**

```
discoParseMeta(c("CT24_AD_1", "CT24_AS_1", "CT24_AE_2", "CT24_AW_2",
"CT26_AB_1", "CT26_AC_1", "CT26_BB_2", "CT26_BC_2"))
```

---

discoPCA

*Perform PCA*


---

**Description**

Calculates PCA results from `prcomp` with error handling and outputs suitable for the DiscoRhythm workflow.

**Usage**

```
discoPCA(se, scale = TRUE, npcs = 10)
```

**Arguments**

<code>se</code>	SummarizedExperiment, the main data object used by DiscoRhythm expected to contain <code>se\$ID</code> , <code>se\$ReplicateID</code> , <code>se\$Time</code> sample metadata and non-null row-names. See the vignette for more details.
<code>scale</code>	logical, whether or not to scale the data prior to PCA, see <a href="#">prcomp</a> for more details.
<code>npcs</code>	numeric, maximum number of principal components to return.

**Value**

output from [prcomp](#) with an added table summary

**Examples**

```
se <- discoGetSimu(TRUE)
pca <- discoPCA(se)
```

---

discoPCAgetOutliers

*Internal function for applying SD cutoff to PCA results Returns a logical indicating which samples are not outliers*


---

**Description**

Internal function for applying SD cutoff to PCA results Returns a logical indicating which samples are not outliers

**Usage**

```
discoPCAgetOutliers(x, SDfactor = 3, pcToCut = seq_len(4))
```

**Value**

logical indicating which samples are outliers in PCA

---

discoPeriodDetection *Detect dataset-wide fits to multiple periodicities*

---

### Description

Detect dataset-wide fits to multiple periodicities

### Usage

```
discoPeriodDetection(
  se,
  timeType = c("linear", "circular"),
  main_per = 24,
  test_periods = NULL
)
```

### Arguments

se	SummarizedExperiment, the main data object used by DiscoRhythm expected to contain se\$ID, se\$ReplicateID, se\$Time sample metadata and non-null row-names. See the vignette for more details.
timeType	character, time is either reported as "linear" or "circular" on some base-cycle (ex. time of day). This determines the periods that will be tested for.
main_per	numeric, if timeType=="circular" main_per indicates the period of the base-cycle where sampling times are derived.
test_periods	numeric, a vector of the periods to test. if timeType=="linear" and length(test_periods)==2 it will be assumed to be a range of periods to test over.

### Value

A data.frame of Rsquared values for each period, for each row of Maindata.

### Examples

```
se <- discoGetSimu(TRUE)

# Detect periods
rsqs <- discoPeriodDetection(se)
```

---

discoQC

*Quality Control for DiscoRhythm*

---

### Description

Functions for executing outlier detection and row filtering procedures prior to rhythmicity analysis.

**Usage**

```
discoPCAoutliers(se, threshold = 3, scale = TRUE, pcToCut = seq_len(4))
```

```
discoInterCorOutliers(
  se,
  cor_method = c("pearson", "kendall", "spearman"),
  threshold = 3,
  thresh_type = c("sd", "value")
)
```

```
discoRepAnalysis(
  se,
  aov_method = c("Equal Variance", "Welch", "None"),
  aov_pcut = 0.05,
  aov_Fcut = 0,
  avg_method = c("Median", "Mean", "Random", "None")
)
```

**Arguments**

se	SummarizedExperiment, the main data object used by DiscoRhythm expected to contain se\$ID, se\$ReplicateID, se\$Time sample metadata and non-null row-names. See the vignette for more details.
threshold	numeric, a threshold determining which samples are outliers (for discoInterCorOutliers, in units of thresh_type, for discoPCAoutliers in units of standard deviations).
scale	logical, whether or not to scale the data prior to PCA, see <a href="#">prcomp</a> for more details.
pcToCut	numeric, which PCs to use for outlier detection. It is recommended to select the first X PCs based on which PCs explain a significant amount of variance in the data.
cor_method	character, method of pairwise correlation (see <a href="#">cor</a> 's "method" argument for all options).
thresh_type	character indicating threshold type (either standard deviations below the mean, or an absolute correlation value). One of: "sd" or "value".
aov_method	character, method to use for ANOVA. One of: "Equal Variance", "Welch", or "None".
aov_pcut	numeric, p-value cutoff used to select rows with statistically significant signal-to-noise.
aov_Fcut	numeric, F-statistic cutoff used to select rows with high signal-to-noise based on magnitude.
avg_method	character, method for averaging technical replicates. One of: "Median", "Mean", "Random", or "None".

**Value**

list containing PCA results and the detected outliers

A list of 3 objects: 1) outliers - named logical indicating if the sample is an outlier 2) meanCor - mean of all pairwise correlations for a given sample 3) corMat - Matrix of all pairwise correlation values



**Examples**

```
se <- discoGetSimu(TRUE)
PCARes <- discoPCAoutliers(se)

CorRes <- discoInterCorOutliers(se)

ANOVARes <- discoRepAnalysis(se)
```

---

discoShinyHandler	<i>Handle Error/Warning messages appropriately with shiny notifications for warnings and pop-ups for errors</i>
-------------------	---

---

**Description**

Handle Error/Warning messages appropriately with shiny notifications for warnings and pop-ups for errors

**Usage**

```
discoShinyHandler(expr, section = "Execution", shinySession = NULL)
```

**Value**

output from expr

---

fisherExact	<i>Extract key values from stats::fisher.test results</i>
-------------	---

---

**Description**

Set p-values of 0 to  $< 2.2e-16$  and reformat odds ratio using formatC

**Usage**

```
fisherExact(var1, var2)
```

**Value**

modified output of [fisher.test](#)

---

`inferFilteredDesign`     *DiscoRhythm Experimental Design*

---

### Description

Infers the experimental design from various input data

### Usage

```
inferFilteredDesign(se)
```

### Arguments

`se`                      SummarizedExperiment, the main data object used by DiscoRhythm expected to contain `se$ID`, `se$ReplicateID`, `se$Time` sample metadata and non-null row-names. See the vignette for more details.

### Details

Characteristics of the experiment sampling are gathered to determine which oscillation detection algorithms are suitable.

### Value

list with inferred experimental design features needed to perform replicate analysis and merging in `discoRepAnalysis`.

---

`ImCSmat`                      *Cosinor*

---

### Description

Fixed period cosinor ("harmonic regression") on each row of a matrix

### Usage

```
ImCSmat(x, zts, per = 24)
```

### Arguments

`x`                            - numeric data matrix  
`zts`                         - numeric vector of length `ncol(data)` representing time points for each data column  
`per`                         - period of oscillations (default=24)

### Details

Fits a cosinor model to each row of a matrix.

**Value**

data frame with the following estimated statistics:

- acrophase - acrophases
- amplitude - amplitudes
- Rsq - r-squared values
- pvalue - p-values
- mesor - intercept coefficient
- sincoef - sine coefficient
- coscoef - cosine coefficient

**Author(s)**

Karolis Koncevičius

**Examples**

```
## Not run:
tmpData <- matrix(rnorm(24 * 1000), ncol = 24)
tmpData[sample(length(tmpData), nrow(tmpData))] <- NA
lmCSmat(tmpData, 1:24, 24)

## End(Not run)
```

---

lmCSmatNoNA

*Cosinor Without NA Values*


---

**Description**

Fixed period cosinor on each row of a matrix with no missing values.

**Usage**

```
lmCSmatNoNA(x, zts, per = 24)
```

**Arguments**

- |     |   |
|-----|---|
| x   | - numeric data matrix   |
| zts | - numeric vector of length ncol(data) representing time points for each data column |
| per | - period of oscillations (default=24)   |

**Details**

Fits a cosinor model to each row of a matrix that has no NA values.

**Value**

data frame with the following estimated statistics:

- acrophase - acrophases
- amplitude - amplitudes
- Rsq - r-squared values
- pvalue - p-values
- mesor - intercept coefficient
- sincoef - sine coefficient
- coscoef - cosine coefficient

**Author(s)**

Karolis Koncevičius

**Examples**

```
## Not run:  
tmpData <- matrix(rnorm(24 * 1000), ncol = 24)  
lmCSmat(tmpData, 1:24, 24)  
  
## End(Not run)
```

---

PeriodDetection\_range *Helper for discoPeriodDetection*

---

**Description**

Helper for discoPeriodDetection

**Usage**

```
PeriodDetection_range(times, circular_t, main_per, test_periods)
```

**Value**

a set of periods to use for discoPeriodDetection

---

sincos	<i>sin/cos cosinor operations</i>
--------	-----------------------------------

---

**Description**

Converts sine and cosine coefficients to acrophase/amplitude

**Usage**

```
sincos2acr(sin, cos, per = 24)
```

```
sincos2amp(sin, cos)
```

**Arguments**

sin	- sine coefficient returned from cosinor model.
cos	- cosine coefficient returned from cosinor model.
per	- period of oscillation (default = 24).

**Value**

acrophase

**Author(s)**

Matthew Carlucci

**Examples**

```
## Not run: # don't run since internal
sincos2acr(0.5, 0.5, per = 24)
sincos2amp(0.5, 0.5)

## End(Not run)
```

---

theme_disco	<i>Common theme elements in DiscoRhythm plots</i>
-------------	---

---

**Description**

Common theme elements in DiscoRhythm plots

**Usage**

```
theme_disco()
```

# Index

## \* datasets

discoODAexclusionMatrix, 10  
discoODAid2name, 11

## \* internal

checkJTKperiod, 3  
checkPeriod, 4  
discoColors, 7  
discoPCAgetOutliers, 14  
DiscoRhythm-package, 2  
discoShinyHandler, 17  
fisherExact, 17  
inferFilteredDesign, 18  
lmCSmat, 18  
lmCSmatNoNA, 19  
PeriodDetection\_range, 20  
sincos, 21  
theme\_disco, 21

checkJTKperiod, 3  
checkPeriod, 4  
cor, 5, 16

discoApp, 4  
discoBatch, 5  
discoCheckInput, 6  
discoColors, 7  
discoDesignSummary, 8  
discoDFtoSE, 8  
discoGetODAs (discoODAs), 11  
discoGetSimu, 9  
discoInterCorOutliers (discoQC), 15  
discoODAexclusionMatrix, 10  
discoODAid2name, 11  
discoODAs, 11  
discoParseMeta, 13  
discoPCA, 14  
discoPCAgetOutliers, 14  
discoPCAoutliers (discoQC), 15  
discoPeriodDetection, 15  
discoQC, 15  
discoRepAnalysis (discoQC), 15  
DiscoRhythm (DiscoRhythm-package), 2  
DiscoRhythm-package, 2  
discoSEtoDF (discoDFtoSE), 8

discoShinyHandler, 17

fisher.test, 17  
fisherExact, 17

inferFilteredDesign, 18

lmCSmat, 12, 18  
lmCSmatNoNA, 19

meta2d, 12

PeriodDetection\_range, 20  
prcomp, 14, 16

sincos, 21  
sincos2acr (sincos), 21  
sincos2amp (sincos), 21

theme\_disco, 21