Package: GLMMcosinor (via r-universe)

July 31, 2026

Type Package

Title Fit a Cosinor Model Using a Generalized Mixed Modeling Framework

Version 0.2.1.9000

Description Allows users to fit a cosinor model using the 'glmmTMB' framework. This extends on existing cosinor modeling packages, including 'cosinor' and 'circacompare', by including a wide range of available link functions and the capability to fit mixed models. The cosinor model is described by Cornelissen (2014) <doi:10.1186/1742-4682-11-16>.

License GPL (>= 3)

URL https://github.com/ropensci/GLMMcosinor, https://docs.ropensci.org/GLMMcosinor/

BugReports https://github.com/ropensci/GLMMcosinor/issues

Imports assertthat, cowplot, ggforce, ggplot2, glmmTMB, lme4, rlang, scales, stats

Suggests cosinor, covr, DHARMa, dplyr, DT, flextable, ftExtra, knitr, rmarkdown, testthat (>= 3.0.0), vdiffr, withr

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE, roclets = c (` `namespace", ` `rd", ` `srr::srr_stats_roclet"))

RoxygenNote 7.3.3

Depends R (>= 4.1.0)

Language en-US

Config/pak/sysreqs cmake libfontconfig1-dev libfreetype6-dev make libicu-dev

Repository https://ropensci.r-universe.dev

Date/Publication 2025-12-04 01:40:03 UTC

RemoteUrl https://github.com/ropensci/GLMMcosinor

RemoteRef main

RemoteSha bc86b08db1cb47e615d9e091a85cebd2689073d1

Contents

`amp_acro` *Used to specify a cosinor component in the model formula.*

Description

Checks the validity of user inputs before creating an updated formula and associated modifications to the `data.frame`.

Usage

```
amp_acro(time_col, n_components = 1, group, period, ...)
```

Arguments

time_col	A numeric column within the data.frame() passed by via the data arg containing the time values.
n_components	The Number of cosinor components in the model.
group	A vector of the names for the group factors (column names within the data.frame() passed by via the data arg).
period	A numeric value or vector containing the period. The number of values should be equal to n_components.
...	Extra arguments for use within GLMMcosinor.

Value

A data.frame and formula appropriate for use by data_processor().

Examples

```
# Single component cosinor model
cglmm(
  vit_d ~ amp_acro(time_col = time, group = "X", period = 12),
  data = vitamind
)

# 2-component cosinor model with simulated data
sim_data <- simulate_cosinor(
  n = 500,
  mesor = 5,
  amp = c(2, 1),
  acro = c(1, 1.5),
  beta.mesor = 2,
  beta.amp = c(2, 1),
  beta.acro = c(1, 1.5),
  family = "gaussian",
  period = c(12, 6),
  n_components = 2,
  beta.group = TRUE,
)

cglmm(
  Y ~ group + amp_acro(times,
    n_components = 2,
    group = "group",
    period = c(12, 6)
  ),
  data = sim_data,
  family = gaussian
)
```

`autoplot.cglmm` *Plot a cosinor model*

Description

Given a cglmm model fit, generate a plot of the data with the fitted values. Optionally allows for plotting by covariates

Usage

```
## S3 method for class 'cglmm'
autoplot(
  object,
  ci_level = 0.95,
  x_str,
  type = "response",
  xlims,
  pred.length.out,
  points_per_min_cycle_length = 20,
  superimpose.data = FALSE,
  data_opacity = 0.3,
  predict.ribbon = TRUE,
  ranef_plot = NULL,
  cov_list = NULL,
  quietly = TRUE,
  ...
)
```

Arguments

`object`	A `cglmm` object.
`ci_level`	The level for calculated confidence intervals. Defaults to `0.95`.
`x_str`	A `character` vector naming variable(s) to be plotted. Default has no value and plots all groups.
`type`	A `character` that will be passed as an argument to `predict.cglmm()`, specifying the type of prediction (e.g, "response", or "link"). See `?glmmTMB::predict.glmmTMB` for full list of possible inputs.
`xlims`	A vector of length two containing the limits for the x-axis.
`pred.length.out`	An integer value that specifies the number of predicted data points. The larger the value, the more smooth the fitted line will appear. If missing, uses `points_per_min_cycle_length` to generate a sensible default value.
`points_per_min_cycle_length`	Used to determine the number of samples to create plot if `pred.length.out` is missing.

`superimpose.data`
: A `logical`. If `TRUE`, data from the original data used to fit the model (`object`) will be superimposed over the predicted fit.

`data_opacity`
: A number between 0 and 1 inclusive that controls the opacity of the superimposed data. (Used as the `alpha` when calling `ggplot2::geom_point()`).

`predict.ribbon`
: A `logical`. If `TRUE`, a prediction interval is plotted.

`ranef_plot`
: Specify the random effects variables that you wish to plot. If not specified, only the fixed effects will be visualized.

`cov_list`
: Specify the levels of the covariates that you wish to plot as a list. For example, if you have two covariates: var1, and var 2, you could fix the level to be plotted as such `cov_list = list(var1 = 'a', var2 = 1)`, where 'a' is a level in 'var1', and 1 is a level in 'var2'. See the examples for a demonstration. If not specified, the reference level of the covariate(s) will be used. `points_per_min_cycle_length` is the number of points plotted per the minimum cycle length (period) of all cosinor components in the model.

`quietly`
: A `logical`. If `TRUE`, shows warning messages when wrangling data and fitting model. Defaults to `TRUE`.

`...`
: Additional, ignored arguments.

Value

Returns a `ggplot` object.

Examples

```
# A simple model
model <- cglmm(
  vit_d ~ X + amp_acro(time, group = "X", period = 12),
  data = vitamind
)
autoplot(model, x_str = "X")


# Plotting a model with various covariates
test_data <- vitamind[vitamind$X == 1, ]
test_data$var1 <- sample(c("a", "b", "c"), size = nrow(test_data), replace = TRUE)
test_data$var2 <- rnorm(n = nrow(test_data))

object <- cglmm(
  vit_d ~ amp_acro(time, period = 12) + var1 + var2,
  data = test_data
)
autoplot(object,
  cov_list = list(
    var1 = "a",
    var2 = 1
  ),
  superimpose.data = TRUE
)
```

`autoplot_data_processor` *Process data for autoplot*

Description

Process data for autoplot

Usage

```
autoplot_data_processor(x, newdata, x_str, ranef_plot = NULL)
```

Arguments

`x`	cglmm object.
`newdata`	new dataset (if used).
`x_str`	A `character` vector naming variable(s) to be plotted. Default has no value and plots all groups.
`ranef_plot`	Specify the random effects variables that you wish to plot. If not specified, only the fixed effects will be visualized.

Value

a `data.frame`.

Examples

```
object <- cglmm(
  vit_d ~ amp_acro(time_col = time, group = "X", period = 12),
  data = vitamind
)
GLMMcosinor:::autoplot_data_processor(
  object,
  vitamind[ "time"],
  x_str = "X"
)
```

cglmm *Fit cosinor model with* {glmmTMB}

Description

Given an outcome and time variable, fit the cosinor model with optional covariate effects.

Usage

```
cglmm(
  formula,
  data,
  family = stats::gaussian(),
  quietly = TRUE,
  dispformula = ~1,
  ziformula = ~0,
  ...
)
```

Arguments

formula	A formula specifying the cosinor model to be fit. The cosinor portion of the formula is controlled by including amp_acro() on the right hand side of the formula. See amp_acro for more details.
data	A data.frame containing the variables used in the model.
family	A family function or a character string naming a family function. See ?family and ?glmmTMB::family_glmmTMB for options.
quietly	A logical. If TRUE, shows warning messages when wrangling data and fitting model. Defaults to TRUE.
dispformula	A one-sided (i.e., no response variable) formula for dispersion combining fixed and random effects, including cosinor components using amp_acro(). Defaults to ~1.
ziformula	A one-sided (i.e., no response variable) formula for zero-inflation combining fixed and random effects, including cosinor components using amp_acro(). Defaults to ~0.
...	Optional additional arguments passed to glmmTMB::glmmTMB().

Value

Returns a fitted cosinor model as a cglmm object.

References

Tong, YL. Parameter Estimation in Studying Circadian Rhythms, Biometrics (1976). 32(1):85–94.

Examples

```
# Single component cosinor model
cglmm(
  vit_d ~ amp_acro(time_col = time, group = "X", period = 12),
  data = vitamind
)

# 2-component cosinor model with simulated data
sim_data <- simulate_cosinor(
  n = 500,
  mesor = 5,
  amp = c(2, 1),
  acro = c(1, 1.5),
  beta.mesor = 2,
  beta.amp = c(2, 1),
  beta.acro = c(1, 1.5),
  family = "gaussian",
  period = c(12, 6),
  n_components = 2,
  beta.group = TRUE,
)

cglmm(
  Y ~ group + amp_acro(times,
    n_components = 2,
    group = "group",
    period = c(12, 6)
  ),
  data = sim_data,
  family = gaussian
)
```

`cosinor_mixed` *cosinor_mixed dataset for cosinor modeling examples.*

Description

Simulated data set to illustrate a mixed cosinor model. The `Y` column contains a simulated outcome variable that varies over the time variable (`times`). The `subject` column is a grouping variable that can be used as a random effect. The rhythm has a period of 24 hours. Data was simulated using `simulate_cosinor`.

Usage

```
cosinor_mixed
```

Format

A `data.frame` with 3 variables: `Y`, `times`, and `subject`.

`fit_model_and_process` *Fit the cosinor GLMM model using the output from* `update_formula_and_data()` *and a new formula*

Description

Fit the cosinor GLMM model using the output from `update_formula_and_data()` and a new formula

Usage

```
fit_model_and_process(obj, formula, ...)
```

Arguments

`obj`	Output from `update_formula_and_data()`.
`formula`	A (optionally) new formula to use when fitting the cosinor model (maybe with random effects) or other covariates found in the data.
`...`	Optional additional arguments passed to `glmmTMB::glmmTMB()`.

Value

Returns a fitted cosinor model as a `cglmm` object.

Examples

```
# Use vitamind data but add a "patient" identifier used as a random effect
vitamind2 <- vitamind
vitamind2$patient <- sample(
  LETTERS[1:5],
  size = nrow(vitamind2), replace = TRUE
)

# Use update_formula_and_data() to perform wrangling steps of cglmm()
# without yet fitting the model
data_and_formula <- update_formula_and_data(
  data = vitamind2,
  formula = vit_d ~ X + amp_acro(time,
    group = "X",
    period = 12
  )
)

# print formula from above
data_and_formula$newformula

# fit model while adding random effect to cosinor model formula.
mod <- fit_model_and_process(
  obj = data_and_formula,
```

```
  formula = update.formula(
    data_and_formula$newformula, . ~ . + (1 | patient)
  )
)

mod
mod$fit # printing the `glmmTMB` model within shows Std.Dev. of random effect
```

`get_background_grid` *Create the background for a polar plot.*

Description

Create the background for a polar plot.

Usage

```
get_background_grid(
  n_breaks,
  max_plot_radius,
  circle_linetype,
  dial_pos_full_x,
  dial_pos_full_y,
  time_labels,
  text_size,
  text_opacity,
  contour_labels,
  grid_angle_segments
)
```

Arguments

`n_breaks`	The number of concentric circles that will be plotted using the `scales::breaks_pretty()` function. By default, 5 breaks will be used. The number of breaks may be adjusted to result in an even interval. For example, if n_breaks is 3, but the maximum plot radius is 8, instead of plotting circles in intervals in 1.6, this interval will be rounded to 2 to result in the sequence: 0, 2, 4, 6, 8. See `?scales::breaks_pretty` for more details.
`max_plot_radius`	The maximum radius of the background
`circle_linetype`	The linetype of the concentric rings.
`dial_pos_full_x`	Dimensions of the background.
`dial_pos_full_y`	Dimensions of the background.
`time_labels`	Labels for the tick marks (time).

text_size	Size of label size.
text_opacity	Opactiy of labels.
contour_labels	Labels to use for the concentric rings (amplitude).
grid_angle_segments	How many segments to split the plot into.

Value

a ggplot2 object.

Examples

```
GLMMcosinor:::get_background_grid(
  n_breaks = 5,
  max_plot_radius = 10,
  circle_linetype = "dotted",
  dial_pos_full_x = c(10, 7.08, 0.008, -7.1, -10, -7.05, -0.02, 7.09, 10),
  dial_pos_full_y = c(0, 7.07, 10, 7.04, 0.016, -7.09, -10, -7.06, -0.03),
  time_labels = c(0, 1.5, 3, 4.5, 6, 7.5, 9, 10.5, 12),
  text_size = 3.5,
  text_opacity = 1,
  contour_labels = c(0, 2, 4, 6, 8, 10),
  grid_angle_segments = 8
)
```

---

get_point_estimate_plot

Add a ellipses layer to a polar plot.

Description

Generates the point estimates and confidence ellipses. If you wish to add multiple estimates and ellipses on the same plot, run this function with the appropriate inputs, with 'plot_background' being set to the plot you wish to layer upon.

Usage

```
get_point_estimate_plot(
  est_rrr,
  est_sss,
  a_trans,
  b_trans,
  offset,
  direction,
  est_acr,
  group_level,
  group_level_colour_index,
```

```
  group_check,
  ellipse_opacity,
  plot_background
)
```

Arguments

est_rrr	The rrr estimate.
est_sss	The sss estimate.
a_trans	Ellipse long dimension.
b_trans	Ellipse short dimension.
offset	Determines where angle starts.
direction	-1 for clockwise, 1 for anti.
est_acr	Acrophase estimate.
group_level	A vector of group levels.
group_level_colour_index	index for colours.
group_check	Whether or not group is used in amp_acro().
ellipse_opacity	0 to 1, alpha value.
plot_background	The plot to layer upon.

Value

a ggplot2 object.

Examples

```
plot_bground <- GLMMcosinor:::get_background_grid(
  n_breaks = 5,
  max_plot_radius = 10,
  circle_linetype = "dotted",
  dial_pos_full_x = c(10, 7.07, 0.01, -7.09, -10, -7.05, -0.02, 7.09, 10),
  dial_pos_full_y = c(0, 7.07, 10, 7.04, 0.015, -7.09, -10, -7.06, -0.03),
  time_labels = c(0, 1.5, 3, 4.5, 6, 7.5, 9, 10.5, 12),
  text_size = 3.5,
  text_opacity = 1,
  contour_labels = c(0, 2, 4, 6, 8, 10),
  grid_angle_segments = 8
)

GLMMcosinor:::get_point_estimate_plot(
  est_rrr = c(0.86, 6.47),
  est_sss = c(6.24, 4.66),
  a_trans = c(1.33, 1.77),
  b_trans = c(1.26, 1.88),
  offset = 0,
```

```
  direction = 1,
  est_acr = c(1.43, 0.62),
  group_level = structure(c("0", "1"), dim = 2L),
  group_level_colour_index = 2L,
  group_check = TRUE,
  ellipse_opacity = 0.3,
  plot_background = plot_bground
)
```

`polar_plot` *Generates a polar plot with elliptical confidence intervals*

Description

Generates a polar plot with elliptical confidence intervals

Usage

```
polar_plot(
  x,
  ci_level = 0.95,
  n_breaks = 5,
  component_index = NULL,
  grid_angle_segments = 8,
  radial_units = c("radians", "degrees", "period"),
  clockwise = FALSE,
  text_size = 3,
  text_opacity = 0.5,
  fill_colors,
  ellipse_opacity = 0.3,
  circle_linetype = "dotted",
  start = c("right", "left", "top", "bottom"),
  view = c("full", "zoom", "zoom_origin"),
  overlay_parameter_info = FALSE,
  quietly = TRUE,
  show_component_labels = TRUE,
  xlims,
  ylims,
  ...
)
```

Arguments

`x`	An object of class `cglmm`
`ci_level`	The level for calculated confidence ellipses. Defaults to 0.95.

n_breaks The number of concentric circles that will be plotted using the `scales::breaks_pretty()` function. By default, 5 breaks will be used. The number of breaks may be adjusted to result in an even interval. For example, if n_breaks is 3, but the maximum plot radius is 8, instead of plotting circles in intervals in 1.6, this interval will be rounded to 2 to result in the sequence: 0, 2, 4, 6, 8. See `?scales::breaks_pretty` for more details.

component_index A number that corresponds to a particular component from the `cglmm()` object that will be used to create polar plot. If missing (default), then plots for all components will be arranged in the returned plot. If a single or multiple values are provided, then these components will be returned. (for example `component_index = 1`, `component_index = c(1, 3)`).

grid_angle_segments An `integer`. Determines the total number of segments in the background of the polar plot. For example, a value of 4 will create quadrants around the origin. Defaults to 8.

radial_units A `character` specifying the angular units of the plot. Possible values are one of `c('radians', 'degrees', 'period')`. These units relate to the period of the component being visualized.

`'radians'`: $[0, 2\pi]$

`'degrees'`: $[0, 360]$

`'period'`: $[0, period]$

clockwise A `logical`. If `TRUE`, the angles increase in a clockwise fashion. If `FALSE`, anti-clockwise. Defaults to `FALSE`.

text_size A number controlling the font size of the text labels. Defaults to 3.

text_opacity A `numeric` between 0 and 1 inclusive that controls the opacity of the text labels.

fill_colors A `character` vector containing colors that will be mapped to levels within a group. If the model has components with different number of levels per factor, the length of this input should match the greatest number of levels. If not, or if the number of levels exceeds the length of the default argument (8), colors are generated using `rainbow()`.

ellipse_opacity A `numeric` between 0 and 1 inclusive that controls the opacity of the confidence ellipses. Defaults to 0.3.

circle_linetype A `character` or `numeric` that determines the `linetype` of the radial circles in background of the polar plot. See `?linetype` for more details.

start A `character`, within `c("right", "left", "top", "bottom")` that determines where angle 0 is located. If `start = "top"`, and `clockwise = TRUE`, the angle will rotate clockwise, starting at the '12 o-clock' position on a clock.

view A `character`, within `c("full", "zoom", "zoom_origin")` that controls the view of the plots.

`'full'`**: maintains a full view of the polar plot, including the background radial circles.**

'zoom': finds the minimum view window which contains all confidence ellipses.

'zoom_origin': zooms into the confidence ellipses (like "zoom"), but also keeps the origin within f

overlay_parameter_info
: A logical argument. If TRUE, more information about the acrophase and amplitude are displayed on the polar plots.

quietly
: Analogous to verbose, this logical argument controls whether messages are displayed in the console.

show_component_labels
: Logical argument, TRUE by default. When TRUE, the polar plots have labels corresponding to their components.

xlims
: A vector of length two containing the limits for the x-axis.

ylims
: A vector of length two containing the limits for the y-axis.

...
: Additional, ignored arguments.

Value

Returns a ggplot object.

Examples

```
data(vitamind)
model <- cglmm(
  vit_d ~ X + amp_acro(time, group = "X", period = 12),
  data = vitamind
)
polar_plot(model, radial_units = "period")
```

---

polar_plot.cglmm *Generates a polar plot with elliptical confidence intervals*

Description

Generates a polar plot with elliptical confidence intervals

Usage

```
## S3 method for class 'cglmm'
polar_plot(
  x,
  ci_level = 0.95,
  n_breaks = 5,
  component_index = NULL,
  grid_angle_segments = 8,
  radial_units = c("radians", "degrees", "period"),
```

```
  clockwise = FALSE,
  text_size = 3.5,
  text_opacity = 1,
  fill_colors,
  ellipse_opacity = 0.3,
  circle_linetype = "dotted",
  start = c("right", "left", "top", "bottom"),
  view = c("full", "zoom", "zoom_origin"),
  overlay_parameter_info = FALSE,
  quietly = TRUE,
  show_component_labels = TRUE,
  xlims,
  ylims,
  ...
)
```

Arguments

x	An object of class cglmm
ci_level	The level for calculated confidence ellipses. Defaults to 0.95.
n_breaks	The number of concentric circles that will be plotted using the scales::breaks_pretty() function. By default, 5 breaks will be used. The number of breaks may be adjusted to result in an even interval. For example, if n_breaks is 3, but the maximum plot radius is 8, instead of plotting circles in intervals in 1.6, this interval will be rounded to 2 to result in the sequence: 0, 2, 4, 6, 8. See ?scales::breaks_pretty for more details.
component_index	A number that corresponds to a particular component from the cglmm() object that will be used to create polar plot. If missing (default), then plots for all components will be arranged in the returned plot. If a single or multiple values are provided, then these components will be returned. (for example component_index = 1, component_index = c(1, 3)).
grid_angle_segments	An integer. Determines the total number of segments in the background of the polar plot. For example, a value of 4 will create quadrants around the origin. Defaults to 8.
radial_units	A character specifying the angular units of the plot. Possible values are one of c('radians', 'degrees', 'period'). These units relate to the period of the component being visualized. 'radians': $[0, 2\pi]$ 'degrees': $[0, 360]$ 'period': $[0, period]$
clockwise	A logical. If TRUE, the angles increase in a clockwise fashion. If FALSE, anti-clockwise. Defaults to FALSE.
text_size	A number controlling the font size of the text labels. Defaults to 3.
text_opacity	A numeric between 0 and 1 inclusive that controls the opacity of the text labels.

`fill_colors` A `character` vector containing colors that will be mapped to levels within a group. If the model has components with different number of levels per factor, the length of this input should match the greatest number of levels. If not, or if the number of levels exceeds the length of the default argument (8), colors are generated using `rainbow()`.

`ellipse_opacity`
A `numeric` between 0 and 1 inclusive that controls the opacity of the confidence ellipses. Defaults to 0.3.

`circle_linetype`
A `character` or `numeric` that determines the `linetype` of the radial circles in background of the polar plot. See `?linetype` for more details.

`start` A `character`, within `c("right", "left", "top", "bottom")` that determines where angle 0 is located. If `start = "top"`, and `clockwise = TRUE`, the angle will rotate clockwise, starting at the '12 o-clock' position on a clock.

`view` A `character`, within `c("full", "zoom", "zoom_origin")` that controls the view of the plots.

`'full'`**: maintains a full view of the polar plot, including the background radial circles.**

`'zoom'`**: finds the minimum view window which contains all confidence ellipses.**

`'zoom_origin'`**: zooms into the confidence ellipses (like "zoom"), but also keeps the origin within f**

`overlay_parameter_info`
A `logical` argument. If `TRUE`, more information about the acrophase and amplitude are displayed on the polar plots.

`quietly` Analogous to verbose, this `logical` argument controls whether messages are displayed in the console.

`show_component_labels`
Logical argument, TRUE by default. When TRUE, the polar plots have labels corresponding to their components.

`xlims` A vector of length two containing the limits for the x-axis.

`ylims` A vector of length two containing the limits for the y-axis.

`...` Additional, ignored arguments.

Value

Returns a `ggplot` object.

Examples

```
model <- cglmm(
  vit_d ~ X + amp_acro(time, group = "X", period = 12),
  data = vitamind
)
polar_plot(model, radial_units = "period")
```

predict.cglmm *Predict from a cosinor model*

Description

Given a time variable and optional covariates, generate predicted values from a cosinor fit. Default prediction is the mean value, optionally can predict at a given month

Usage

```
## S3 method for class 'cglmm'
predict(object, newdata, ...)
```

Arguments

object	An object of class cglmm.
newdata	Optional new data.
...	other arguments passed to glmmTMB:::predict.glmmTMB.

Value

Returns predicted values from the cosinor model.

Examples

```
fit <- cglmm(vit_d ~ X + amp_acro(time,
  group = "X",
  n_components = 1,
  period = 12
), data = vitamind)
predict(fit)
```

print.cglmm *Print a brief summary of the* cglmm *model.*

Description

Print a brief summary of the cglmm model.

Usage

```
## S3 method for class 'cglmm'
print(x, digits = getOption("digits"), ...)
```

Arguments

x	A cglmm object.
digits	Controls the number of digits displayed in the summary output.
...	Additional, ignored arguments.

Value

print(x) returns x invisibly.

Examples

```
# Single component cosinor model
cglmm(
  vit_d ~ amp_acro(time_col = time, group = "X", period = 12),
  data = vitamind
)
```

---

print.cglmmSubTest *Print test of model*

Description

Print test of model

Usage

```
## S3 method for class 'cglmmSubTest'
print(x, ...)
```

Arguments

x	A sub_test_cosinor object.
...	Additional, ignored arguments.

Value

print(x) returns x invisibly.

Examples

```
data_2_component <- simulate_cosinor(
  n = 10000,
  mesor = 5,
  amp = c(2, 5),
  acro = c(0, pi),
  beta.mesor = 4,
  beta.amp = c(3, 4),
  beta.acro = c(0, pi / 2),
```

```
  family = "gaussian",
  n_components = 2,
  period = c(10, 12),
  beta.group = TRUE
)
mod_2_component <- cglmm(
  Y ~ group + amp_acro(times,
    n_components = 2, group = "group",
    period = c(10, 12)
  ),
  data = data_2_component
)
test_output <- test_cosinor_levels(
  mod_2_component,
  param = "amp",
  x_str = "group"
)
print(test_output$global.test)
```

`print.cglmmSummary` *Print the summary of a cosinor model*

Description

Print the summary of a cosinor model

Usage

```
## S3 method for class 'cglmmSummary'
print(x, digits = getOption("digits"), ...)
```

Arguments

`x`	An object of class `cglmmSummary`
`digits`	Controls the number of digits displayed in the summary output
`...`	Currently unused

Value

`print` returns `x` invisibly.

Examples

```
fit <- cglmm(vit_d ~ X + amp_acro(time,
  group = "X",
  n_components = 1,
  period = 12
), data = vitamind)
summary(fit)
```

print.cglmmTest *Print results of test of cosinor model*

Description

Print results of test of cosinor model

Usage

```
## S3 method for class 'cglmmTest'
print(x, ...)
```

Arguments

x	A test_cosinor object.
...	Arguments passed to print

Value

print(x) returns x invisibly.

Examples

```
data_2_component <- simulate_cosinor(
  n = 10000,
  mesor = 5,
  amp = c(2, 5),
  acro = c(0, pi),
  beta.mesor = 4,
  beta.amp = c(3, 4),
  beta.acro = c(0, pi / 2),
  family = "gaussian",
  n_components = 2,
  period = c(10, 12),
  beta.group = TRUE
)
mod_2_component <- cglmm(
  Y ~ group + amp_acro(times,
    n_components = 2, group = "group",
    period = c(10, 12)
  ),
  data = data_2_component
)
test_cosinor_levels(
  mod_2_component,
  param = "amp",
  x_str = "group"
)
```

`sigma.cglmm` *Extract residual standard deviation or dispersion parameter*

Description

see `?glmmTMB::sigma` for more details.

Usage

```
## S3 method for class 'cglmm'
sigma(object, ...)
```

Arguments

`object`	An object of class `cglmm`.
`...`	(ignored; for method compatibility)

Value

a `numeric`.

Examples

```
testdata_poisson <- simulate_cosinor(100,
  n_period = 2,
  mesor = 7,
  amp = c(0.1, 0.5),
  acro = c(1, 1),
  beta.mesor = 4.4,
  beta.amp = c(0.1, 0.46),
  beta.acro = c(0.5, -1.5),
  family = "poisson",
  period = c(12, 6),
  n_components = 2,
  beta.group = TRUE
)

mod <- cosinor_model <- cglmm(
  Y ~ group + amp_acro(times,
    period = c(12, 6),
    n_components = 2,
    group = "group"
  ),
  data = testdata_poisson,
  family = glmmTMB::nbinom1()
)
sigma(mod)
```

`simulate_cosinor` *Simulate data from a cosinor model*

Description

This function simulates data from a cosinor model with a single covariate, where the time scale is month, and optionally allows for single covariate effects on the mean, amplitude, and acrophase.

Usage

```
simulate_cosinor(
  n,
  mesor,
  amp,
  acro,
  period = 24,
  n_components,
  beta.group = FALSE,
  beta.mesor,
  beta.amp,
  beta.acro,
  n_period = 1,
  family = c("gaussian", "poisson", "binomial", "gamma"),
  ...
)
```

Arguments

`n`	The sample size. An `integer` greater than 0.
`mesor`	A `numeric`. The MESOR (midline estimating statistic of rhythm) for `group = 0`. The MESOR is independent of the cosinor components, so only one value is allowed even if there are multiple components in the data being simulated.
`amp`	A `numeric`. The amplitude value (for `group = 0` if grouped data are being simulated (`beta.group = TRUE`)). If simulating data with multiple components, specify a vector with values for each component. E.g: `amp = c(5, 10)`.
`acro`	A `numeric`. The acrophase value in radians (for `group = 0` if grouped data are being simulated (`beta.group = TRUE`)). If simulating data with multiple components, specify a vector with values for each component. E.g: `acr = c(0, pi)` for two components.
`period`	The period of the rhythm data (for `group = 0` if grouped data are being simulated (`beta.group = TRUE`)). If simulating data with multiple components, specify a vector with values for each component. E.g: `period = c(12, 6)` for two components.
`n_components`	The number of components in the model. This must match the length of the inputs for `amp` and `acro`.

beta.group	A logical. If TRUE a second group of data will be simulated and included in the returned data set. If FALSE, beta.acro, beta.mesor, and beta.amp arguments will be ignored.
beta.mesor	A numeric. The MESOR value term for group = 1
beta.amp	A numeric. The amplitude value for group = 1. If simulating data with multiple components, specify a vector with values for each component. E.g: amp = c(2, 8).
beta.acro	A numeric. The acrophase value in radians (for group = 1. If simulating data with multiple components, specify a vector with values for each component. E.g: acr = c(2, 5) for two components.
n_period	A numeric. The number of cycles of the rhythm to be simulated.
family	A character. The family (see ?family) of the simulated dataset. Can handle values in c("poisson", "binomial","gamma", "gaussian").
...	Extra arguments, including alpha that controls the shape argument when sampling from a gamma distribution (when family = "gamma"; default is 1), and sd (standard deviation) which is used when sampling from a normal distribution (when family = "gaussian"; default is 1). To specify these parameters for the beta (treatment) group, use beta.alpha and beta.sd

Value

Returns simulated data in a data.frame.

Examples

```
simulate_cosinor(
  n = 100,
  mesor = 1,
  amp = 1,
  acro = 1,
  period = 24,
  family = "gaussian"
)
```

---

sub_ggplot.cglmm.polar

create a (sub) polar plot (iterated over for each component)

Description

create a (sub) polar plot (iterated over for each component)

Usage

```
sub_ggplot.cglmm.polar(
  comp,
  x,
  sum,
  direction,
  offset,
  radial_units,
  grid_angle_segments,
  n_breaks,
  zoom,
  circle_linetype,
  text_size,
  text_opacity,
  ellipse_opacity,
  overlay_parameter_info,
  fill_colors_check,
  xlims_check,
  xlims,
  ylims_check,
  ylims,
  quietly,
  overlay_start,
  fill_colors,
  zoom_origin
)
```

Arguments

comp	Component.
x	cglmm object.
sum	Model summary.
direction	ifelse(clockwise, -1, 1).
offset	Applied to start position of phase angle.
radial_units	Units for phase.
grid_angle_segments	Number of sugments.
n_breaks	Number of breaks for amplitude.
zoom	Whether or not it is zoomed in.
circle_linetype	linetype for concentric circles.
text_size	Text size for axes.
text_opacity	Text opacity for axes.
ellipse_opacity	Opacity of confidence ellipses.

`overlay_parameter_info`
: Whether or not to show amplitude/phase estimates.

`fill_colors_check`
: `!missing(fill_colors)`.

`xlims_check`
: `TRUE` if limits are unspecified.

`xlims`
: User-specified limits.

`ylims_check`
: `TRUE` if limits are unspecified.

`ylims`
: User-specified limits.

`quietly`
: Send possibly unnecessary messages to console.

`overlay_start`
: background axes positioning.

`fill_colors`
: Colours for ellipses.

`zoom_origin`
: Zoom position if used.

Value

A `ggplot2` object.

Examples

```
model <- cglmm(
  vit_d ~ amp_acro(time_col = time, group = "X", period = 12),
  data = vitamind
)
GLMMcosinor:::sub_ggplot.cglmm.polar(
  comp = 1,
  x = model,
  sum = summary(model),
  direction = 1,
  offset = 0,
  radial_units = "period",
  grid_angle_segments = 8,
  n_breaks = 5,
  zoom = FALSE,
  circle_linetype = "dotted",
  text_size = 3.5,
  text_opacity = 1,
  ellipse_opacity = 0.3,
  overlay_parameter_info = FALSE,
  fill_colors_check = FALSE,
  xlims_check = FALSE,
  ylims_check = FALSE,
  quietly = TRUE,
  overlay_start = 0,
  zoom_origin = FALSE
)
```

summary.cglmm *Summarize a cosinor model Given a time variable and optional covariates, generate inference a cosinor fit. Gives estimates, confidence intervals, and tests for the raw parameters, and for the mean, amplitude, and acrophase parameters. If the model includes covariates, the function returns the estimates of the mean, amplitude, and acrophase for the group with covariates equal to 1 and equal to 0. This may not be the desired result for continuous covariates.*

Description

Summarize a cosinor model Given a time variable and optional covariates, generate inference a cosinor fit. Gives estimates, confidence intervals, and tests for the raw parameters, and for the mean, amplitude, and acrophase parameters. If the model includes covariates, the function returns the estimates of the mean, amplitude, and acrophase for the group with covariates equal to 1 and equal to 0. This may not be the desired result for continuous covariates.

Usage

```
## S3 method for class 'cglmm'
summary(object, ci_level = 0.95, ...)
```

Arguments

object	An object of class cglmm
ci_level	The level for calculated confidence intervals. Defaults to 0.95.
...	Currently unused

Value

Returns a summary of the cglmm model as a cglmmSummary object.

Examples

```
fit <- cglmm(vit_d ~ X + amp_acro(time,
  group = "X",
  n_components = 1,
  period = 12
), data = vitamind)
summary(fit)
```

`test_cosinor_components` *Test for differences in a cosinor model between components.*

Description

Given a time variable and optional covariates, generate inference a cosinor fit. For the covariate named (or vector of covariates), this function performs a Wald test comparing the group with covariates equal to 1 to the group with covariates equal to 0. This may not be the desired result for continuous covariates.

Usage

```
test_cosinor_components(
  x,
  x_str = NULL,
  param = "amp",
  comparison_A = 1,
  comparison_B = 2,
  level_index = 0,
  ci_level = 0.95
)
```

Arguments

`x`	A `cglmm` object.
`x_str`	A `character`. The name of the grouping variable within which differences in the selected cosinor characteristic (amplitude or acrophase) will be tested. If there is no grouping variable in the model, then this can be left as NULL (default).
`param`	A `character`. Either `"amp"` or `"acr"` for testing differences in amplitude or acrophase, respectively.
`comparison_A`	An `integer`. Refers to the component number that is to act as the reference group. for the comparison.
`comparison_B`	An `integer`. Refers to the component number that is to act as the comparator group
`level_index`	An `integer`. If `comparison_type = "components"`, `level_index` indicates which level of the grouping variable is being used for the comparison between components.
`ci_level`	The level for calculated confidence intervals. Defaults to `0.95`.

Value

Returns a `test_cosinor` object.

Examples

```
data_2_component <- simulate_cosinor(
  n = 10000,
  mesor = 5,
  amp = c(2, 5),
  acro = c(0, pi),
  beta.mesor = 4,
  beta.amp = c(3, 4),
  beta.acro = c(0, pi / 2),
  family = "gaussian",
  n_components = 2,
  period = c(10, 12),
  beta.group = TRUE
)
mod_2_component <- cglmm(
  Y ~ group + amp_acro(times,
    n_components = 2, group = "group",
    period = c(10, 12)
  ),
  data = data_2_component
)
test_cosinor_components(mod_2_component, param = "amp", x_str = "group")
```

---

test_cosinor_levels *Test for differences in a cosinor model between levels of the grouping variable.*

Description

Given a time variable and optional covariates, generate inference a cosinor fit. For the covariate named (or vector of covariates), this function performs a Wald test comparing the group with covariates equal to 1 to the group with covariates equal to 0. This may not be the desired result for continuous covariates.

Usage

```
test_cosinor_levels(
  x,
  x_str,
  param = "amp",
  comparison_A,
  comparison_B,
  component_index = 1,
  ci_level = 0.95
)
```

Arguments

x	A cglmm object.
x_str	A character. The name of the grouping variable within which differences in the selected cosinor characteristic (amplitude or acrophase) will be tested.
param	A character. Either "amp" or "acr" for testing differences in amplitude or acrophase, respectively.
comparison_A	An integer, or string. Refers to the first level within the grouping variable x_str that is to act as the reference group in the comparison. Ensure that it corresponds to the name of the level in the original dataset.
comparison_B	An integer, or string. Refers to the second level within the grouping variable x_str that is to act as the comparator group in the comparison. Ensure that it corresponds to the name of the level in the original dataset.
component_index	An integer. If comparison_type = "levels", component_index indicates which component is being compared between the levels of the grouping variable.
ci_level	The level for calculated confidence intervals. Defaults to 0.95.

Value

Returns a test_cosinor object.

Examples

```
data_2_component <- simulate_cosinor(
  n = 10000,
  mesor = 5,
  amp = c(2, 5),
  acro = c(0, pi),
  beta.mesor = 4,
  beta.amp = c(3, 4),
  beta.acro = c(0, pi / 2),
  family = "gaussian",
  n_components = 2,
  period = c(10, 12),
  beta.group = TRUE
)
mod_2_component <- cglmm(
  Y ~ group + amp_acro(times,
    n_components = 2, group = "group",
    period = c(10, 12)
  ),
  data = data_2_component
)
test_cosinor_levels(mod_2_component, param = "amp", x_str = "group")
```

update_formula_and_data *Update data and formula for fitting cglmm model*

Description

Update data and formula for fitting cglmm model

Usage

```
update_formula_and_data(
  data,
  formula,
  family = "gaussian",
  quietly = TRUE,
  dispformula = ~1,
  ziformula = ~0
)
```

Arguments

data	input data for fitting cglmm model.
formula	model formula, specified by user including amp_acro().
family	the model family.
quietly	controls whether messages from amp_acro are displayed. TRUE by default
dispformula	The formula specifying the dispersion model
ziformula	The formula specifying the zero-inflation model

Value

Returns a list.

Examples

```
# Use vitamind data but add a "patient" identifier used as a random effect
vitamind2 <- vitamind
vitamind2$patient <- sample(
  LETTERS[1:5],
  size = nrow(vitamind2), replace = TRUE
)

# Use update_formula_and_data() to perform wrangling steps of cglmm()
# without yet fitting the model
data_and_formula <- update_formula_and_data(
  data = vitamind2,
  formula = vit_d ~ X + amp_acro(time,
    group = "X",
```

```
    period = 12
  )
)

# print formula from above
data_and_formula$newformula

# fit model while adding random effect to cosinor model formula.
mod <- fit_model_and_process(
  obj = data_and_formula,
  formula = update.formula(
    data_and_formula$newformula, . ~ . + (1 | patient)
  )
)

mod
mod$fit # printing the `glmmTMB` model within shows Std.Dev. of random effect
```

`vitamind` *Vitamin D dataset for cosinor modeling examples.*

Description

Simulated data set to illustrate the cosinor model. The `vit_d` column contains the blood vitamin D levels which vary over time (`time`). The rhythm of the vitamind D fluctuations follows a cosine function and can be modeled with a cosinor model. The `X` column is a binary covariate representing two groups of patients and is associated with the characteristics of the rhythm. The rhythm has a period of about 12 hours.

Usage

```
vitamind
```

Format

A `data.frame` with 3 variables: `vit_d`, `time`, and `X`.

Index