Package 'nlstimedist'

August 27, 2020

Type Package

Title Non-Linear Model Fitting of Time Distribution of Biological Phenomena

Version 2.0.0

Description Fit biologically meaningful distribution functions to time-sequence data (phenology), estimate parameters to draw the cumulative distribution function and probability density function and calculate standard statistical moments and percentiles. These methods are described in Steer et al. (2019) <doi:10.1111/2041-210X.13293>.

URL https://github.com/nathaneastwood/nlstimedist

BugReports https://github.com/nathaneastwood/nlstimedist/issues

Depends R (>= 3.5.0)

Imports ggplot2 (>= 2.1.0), minpack.lm (>= 1.2-0), nlstools (>= 1.0-2), poorman (>= 0.2.1), stats

Suggests knitr, rmarkdown, tinytest

License GPL-2

LazyData true

Encoding UTF-8

RoxygenNote 7.1.0

VignetteBuilder knitr

NeedsCompilation no

Author Nathan Eastwood [aut, cre, prg, trl, ctr], Nicola Steer [ctb] (<https://orcid.org/0000-0002-6446-8601>), Miguel Franco [ctb] (<https://orcid.org/0000-0002-7249-4981>), Paul Ramsay [ctb] (<https://orcid.org/0000-0001-7631-6480>)

Maintainer Nathan Eastwood <nathan.eastwood@icloud.com>

Repository CRAN

Date/Publication 2020-08-27 08:20:03 UTC

R topics documented:

`glance` *Take a glance at a model*

Description

Construct a single row of model summary statistics.

Usage

```
glance(x)
```

Arguments

`x`	An object of class `timedist`.

Value

`glance()` returns a one row `data.frame` with the columns

`sigma`	the square root of the estimated residual variance
`isConv`	whether the fit successfully converged
`finTol`	the achieved convergence tolerance
`logLik`	the data's log-likelihood under the model
`AIC`	the Akaike Information Criterion
`BIC`	the Bayesian Information Criterion
`deviance`	deviance
`df.residual`	residual degrees of freedom
`RSS`	corrected residual sum of squares
`nobs`	the number of observations from the model fit

Examples

```
tdTilia <- tdData(tilia, x = "Day", y = "Trees")
model <- timedist(data = tdTilia, x = "Day", y = "propMax", r = 0.1, c = 0.5, t = 120)
glance(model)
```

`lobelia` *Lobelia urens seeds data*

Description

This data describes the number of germinating lobelia urens seeds at different temperatures.

Usage

```
lobelia
```

Format

A data frame with 231 rows and 3 variables:

Day The day number

Temperature The temperature

Germination The number which germinated

Details

The total numbers which failed to germinate are 59, 52, 35, 22, 10, 7 and 12 for temperatures 9.8, 12.5, 16.7, 20.2, 24.3, 28.5 and 32.0, respectively.

Examples

```
lobelia
```

`nlstimedist` *Fit the time-course of biological phenomena*

Description

nlstimedist fits a biologically meaningful distribution function to time-sequence data (phenology), estimates parameters to draw the cumulative distribution function and probability density function and calculates standard statistical moments and percentiles.

pupae *Emergence of butterflies data*

Description

This data describes the emergence of butterflies from their pupae from four different cohorts.

Usage

```
pupae
```

Format

A data frame with 64 rows and 3 variables:

Day The day number

Cohort The cohort number

Emergence The number of butterflies to emerge

Examples

```
pupae
```

tdCdfPlot *Plot the timedist PDF or CDF*

Description

Given a model (or models) of class `timedist`, produce a cumulative distribution plot for each of them.

Usage

```
tdCdfPlot(..., S = NULL, xVals = NULL)

tdPdfPlot(..., S = NULL, xVals = NULL)
```

Arguments

`...`	`timedist` model(s).
`S`	`numeric(1)`. Scaling factor for the PDF.
`xVals`	`numeric(n)`. A sequence of values between the x limits (x1, x2) of the plot.

Examples

```
tdTilia <- tdData(tilia, x = "Day", y = "Trees")
model <- timedist(data = tdTilia, x = "Day", y = "propMax", r = 0.1, c = 0.5, t = 120)
tdCdfPlot(model)

tdPdfPlot(model)
```

---

tdData *Prepare* nlstimedist *data*

Description

The data for nlstimedist needs to be in a particular format. This function prepares the data for the model.

Usage

```
tdData(data, x, y, group = NULL)
```

Arguments

data	A data.frame. The raw data to be cleaned.
x	character(1). The time variable.
y	character(1). The number of events.
group	character(1). The run numbers. This is NULL by default if you are only using the function for one run.

Value

A data.frame of the cleaned data to be supplied to the timedist() function.

Examples

```
tdData(tilia, x = "Day", y = "Trees")
tdData(lobelia, x = "Day", y = "Germination", group = "Temperature")
```

tdMoments *Calculate moments for the fitted* timedist *model*

Description

Calculate individual model summary statistics or use the wrapper, tdMoments(), to calculate all model summary statistics.

Usage

```
tdMoments(r, c, t, ...)

tdMean(r, c, t, upper = t * 10, ...)

tdVariance(r, c, t, upper = t * 10, ...)

tdSkew(r, c, t, upper = t * 10, ...)

tdKurtosis(r, c, t, upper = t * 10, alternative = FALSE, ...)

tdEntropy(r, c, t, upper = t * 10, ...)
```

Arguments

r, c, t	numeric(1). Parameters of the Franco distribution.
...	Additional arguments to be passed to stats::integrate().
upper	numeric(1). The upper limit of integration. Defaults to t * 10. Can be infinite for all moment functions except for entropy.
alternative	logical(1). Whether to use the alternative calculation method (TRUE) or not (default: FALSE).

Value

For the individual summary statistic functions, a single numeric; for tdMoments(), a single row data.frame of numerics containing all of the summary statistics as individual columns.

Examples

```
tdMoments(r = 0.1, c = 0.5, t = 120)

tdMean(r = 0.1, c = 0.5, t = 120)
tdVariance(r = 0.1, c = 0.5, t = 120)
tdSkew(r = 0.1, c = 0.5, t = 120)
tdKurtosis(r = 0.1, c = 0.5, t = 120)
tdKurtosis(r = 0.1, c = 0.5, t = 120, alternative = TRUE)
tdEntropy(r = 0.1, c = 0.5, t = 120)
```

tdPDF *Calculate the PDF and CDF*

Description

Calculate values of the probability density function (PDF) and the cumulative distribution function (CDF).

Usage

```
tdPDF(x, r, c, t, S = 1)

tdCDF(x, r, c, t, S = 1)
```

Arguments

x	numeric(n). Points at which to calculate the the PDF.
r, c, t	numeric(1). Parameter values within the model.
S	numeric(1). Scaling factor for the PDF.

Value

A vector of values from the PDF or CDF.

See Also

tdPdfPlot(), tdCdfPlot()

tdPercentiles *Calculate percentiles*

Description

Calculate the percentiles for a given model.

Usage

```
tdPercentiles(model, n, upper = model$m$getPars()["t"] * 10, ...)
```

Arguments

model	An object of class timedist.
n	numeric(n). A vector of percentiles to be calculated.
upper	numeric(1). The upper end point of the interval to search.
...	Additional parameters to be passed to stats::uniroot().

Examples

```
tdTilia <- tdData(tilia, x = "Day", y = "Trees")
model <- timedist(data = tdTilia, x = "Day", y = "propMax", r = 0.1, c = 0.5, t = 120)
tdPercentiles(model, n = 0.5)
tdPercentiles(model, n = seq(0, 0.9, 0.1))
```

---

tdRSS *Calculate the corrected residual sum of squares*

Description

Calculate the corrected residual sum of squares for a `timedist` model.

Usage

```
tdRSS(model)
```

Arguments

`model`	An object of class `timedist`.

Value

`numeric(1)`.

Examples

```
tdTilia <- tdData(tilia, x = "Day", y = "Trees")
model <- timedist(data = tdTilia, x = "Day", y = "propMax", r = 0.1, c = 0.5, t = 120)
model
tdRSS(model)
```

---

tilia *Leafing phenology of tilia cordata*

Description

This data describes the leafing phenology of lime trees (tilia cordata).

Usage

```
tilia
```

Format

A data frame with 34 rows and 2 variables:

Day The day number

Trees The number of trees

Examples

```
tilia
```

`timedist` *Fit the Franco model*

Description

Fit the time-course of biological phenomena.

Usage

```
timedist(data, x, y, r, c, t, ...)
```

Arguments

`data`	A `data.frame`. The data to be included in the model.
`x, y`	`character(1)`. The x and y values in the data, where the y values are the proportions.
`r, c, t`	`numeric(1)`. The starting parameters for the model.
`...`	Additional parameters to be passed to `minpack.lm::nlsLM()`.

Details

The `minpack.lm::nlsLM()` function is used instead of the `stats::nls()` function in order to use the Levenberg-Marquardt algorithm which is an extremely robust method of curve-fitting as it is able to switch between Gauss-Newton and gradient descent. This allows it to cope with far-off-optimal starting values. The standard `nls` function does not use Levenberg-Marquardt; it instead uses the Gauss-Newton type, the PORT routines and a partial linear fit.

See Also

`tdPDF()`, `tdCDF()`, `tdRSS()`, `glance()`, `tdMoments()`, `tdPercentiles()`

Examples

```
tdTilia <- tdData(tilia, x = "Day", y = "Trees")
model <- timedist(data = tdTilia, x = "Day", y = "propMax", r = 0.1, c = 0.5, t = 120)
model
```

Index