Package ‘doremi’

October 13, 2022

Type Package

Title Dynamics of Return to Equilibrium During Multiple Inputs

Version 1.0.0

Description Provides models to fit the dynamics of a regulated system experiencing exogenous inputs.
The underlying models use differential equations and linear mixed-
effects regressions to estimate the
coefficients of the equation. With them, the functions can provide an estimated signal.
The package provides simulation and analysis functions and also print, summary, plot and predict methods,
adapted to the function outputs, for easy implementation and presentation of results.

License GPL-3

URL `https://github.com/dcourvoisier/doremi`

BugReports `https://github.com/dcourvoisier/doremi/issues`

Encoding UTF-8

LazyData true

Imports zoo, data.table, lme4, ggplot2, gridExtra, lmerTest, deSolve,
futile.logger

RoxygenNote 7.1.1

Suggests knitr, rmarkdown, devtools, roxygen2

VignetteBuilder knitr

NeedsCompilation no

Author Mongin Denis [aut, cre] (<`https://orcid.org/0000-0002-4801-8395`>),
Uribe Adriana [aut],
Courvoisier Delphine [aut] (<`https://orcid.org/0000-0002-1956-2607`>)

Maintainer Mongin Denis <`Denis.Mongin@unige.ch`>

Repository CRAN

Date/Publication 2021-01-29 13:30:02 UTC

R topics documented:

`analyze.1order` *DOREMI first order analysis function*

Description

`analyze.1order` estimates the coefficients of a first order differential equation of the form:

$$\frac{dy}{dt}(t) + \gamma(y(t) - yeq) = \gamma k * u(t)$$

using linear mixed-effect models. Where y(t) is the individual's signal,γ the decay rate (and $\tau = 1/\gamma thedecaytime$), $\frac{dy(t)}{dt}$ is the derivative and u(t) is an external excitation perturbing the dynamics.

Usage

```
analyze.1order(
  data,
  id = NULL,
  input = NULL,
  time = NULL,
  signal,
  dermethod = "gold",
  derparam = 3,
  order = 1,
  verbose = FALSE
)
```

Arguments

`data`	Is a data frame containing at least one column, that is the signal to be analyzed.
`id`	Is a CHARACTER containing the NAME of the column of data containing the identifier of the individual. If this parameter is not entered when calling the function, a single individual is assumed and a linear regression is done instead of the linear mixed-effects regression.
`input`	Is a CHARACTER or a VECTOR OF CHARACTERS containing the NAME(s) of data column(s) containing the EXCITATION vector(s). If this parameter is not entered when calling the function, the excitation is assumed to be unknown. In this case, the linear mixed-effect regression is still carried out but no coefficient is calculated for the excitation term. The function then uses the parameters estimated by the regression to carry out an exponential fit of the signal and build the estimated curve. The function will consider as an excitation each column of data having a name contained in the input vector. The function will return a coefficient for each one of the excitation variables included in the input vector.
`time`	Is a CHARACTER containing the NAME of the column of data containing the time vector. If this parameter is not entered when calling the function, it is assumed that time steps are of 1 unit and the time vector is generated internally in the function.
`signal`	Is a CHARACTER containing the NAME of the column of the data frame containing the SIGNAL to be studied.
`dermethod`	is the derivative estimation method. default is "gold". The values are "gold","glla" and "fda", corresponding to the use of `calculate.gold`, `calculate.glla` or `calculate.fda` to estimate the derivatives
`derparam`	If dermethod "glla" or "gold" are chosen, it is the embedding number, a positive integer containing the number of points to be used for the calculation of the derivatives. Default is 3. At least two points are needed for the calculation of the first derivative. If dermethod "fda" is chosen, this parameter is spar, the parameter related to the smoothing parameter lambda used in the penalization function of the function `smooth.spline` to estimate the derivative via splines (Functional Data Analysis). In this case, the value should be between 0 and 1, see `?smooth.spline`
`order`	is the maximum order of the derivative estimated when using `calculate.gold` or `calculate.glla`. Although only the first derivative is used here, using a higher order can enhance derivative estimation (see doi: 10.1080/00273171.2015.1123138Chow et al.(2016))
`verbose`	Is a boolean that displays status messages of the function when set to 1. Useful for debugging.

Details

The analysis performs the following linear mixed-effects regression:

$$\dot{y_{ij}} \sim (b_0 + b_{0j}) + (b_1 + b_{1j})y_{ij} + (b_2 + b_{2j})u_{ij} + e_{ij}$$

with i accounting for the time and j for the different individuals. e_{ij} are the residuals, $\dot{y_{ij}}$ is the derivative calculated on embedding points and y_{ij} and u_{ij} are the signal and the excitation averaged on embedding points. The fixed effect coefficients estimated from the regression are:

- gamma: b_1 (γ from the differential equation)
- kgamma: b_2 ($k\gamma$ from the differential equation)
- yeqgamma: b_0 (γy_{eq} from the differential equation)

The coefficients derived to characterize the signal are calculated as follows:

- the decay time, tau: $\tau = \frac{1}{b_1} = \frac{1}{\gamma}$
- the gain, k: $\gamma = \frac{b_2}{b_1}$. It is the proportionality between the stationary increase of the signal and the excitation increase that caused it.
- the equilibrium value, yeq: $yeq = \frac{b_0}{b_1}$. It is the stationary value reached in the absence of excitation.

The estimation is performed using the function lmer if there are several individuals or lm if there is only one. With the above estimated parameters, the estimated signal can be reconstructed for each individual by using the generate.1order function of this package (based on deSolve's ode function). The function returns five objects:

1. data- A data.frame including the input data, the intermediate calculations used to prepare the variables for the fit and the estimated trajectories for each individual.
   (a) signal_rollmean - calculation of the moving average of the signal over embedding points.
   (b) signal_derivate1 - calculation of the first derivative of the signal with the gold, glla or fda methods in embedding points.
   (c) time_derivate - calculation of the moving average of the time vector over embedding points.
   (d) input_rollmean - calculation of the moving average of the excitation vector over embedding points.
   (e) estimated - the estimated signal calculated using generate.1order with the excitation provided as input and the estimated decay time, gain and equilibrium value obtained from the regression. The initial condition y0 and t0 are the first value of the moving averaged signal (signal_rollmean) and time (time_derivate)
2. resultmean - A data.frame including the fixed effects of the coefficients estimated from the regression gamma, yeqgamma and the kgamma for each excitation considered, with their associated standard error gamma_stde, yeqgamma_stde and kgamma_stde, together with the derived coefficient tau (the decay time), yeq (the equilibrium value) and k (the gain)
3. resultid - A data.frame including, for each individual listed by id number, gamma, yeqgamma and the kgamma, together with the decay time tau, the gain k and the equilibrium value yeq

4. regression - A list containing the summary of the linear mixed-effects regression.

   As seen in the Description section, the print method by default prints only the resultmean element. Each one of the other objects can be accessed by indicating $ and their name after the result, for instance, for a DOREMI object called "result", it is possible to access the regression summary by typing result$regression.
5. embedding - contains the embedding number used to generate the results (same as function input argument)
6. spar - contains the smoothing parameter used for the estimation of the derivatives using splines (method "fda")

SImulation presenting the statistical propoerties of the ethod can be found in doi: 10.1080/00273171.2020.1754155(Mongin et al. (2020)) Example of application of this function can be found in:

- doi: 10.1088/13616579/abbb6e
- doi: 10.1109/ESGCO49734.2020.9158156
- doi: 10.1080/00273171.2020.1754155
- doi: 10.1038/s41598020692181

Value

Returns a summary of the fixed effect coefficients estimated by the linear regression

See Also

`calculate.gold`, `calculate.glla`, `calculate.fda` to compute the derivatives, for details on embedding/spar. and `generate.1order` the function to generate the solution of the first order differential equation

Examples

```
myresult <- analyze.1order(data = cardio,
                  id = "id",
                  input = "load",
                  time = "time",
                  signal = "hr",
                  dermethod ="gold",
                  derparam = 5)
```

`analyze.2order` *DOREMI second order analysis function*

Description

`analyze.2order` estimates the coefficients of a second order differential equation of the form:

$$\frac{d^2y}{dt}(t) + 2\xi\omega_n\frac{dy}{dt}(t) + \omega_n^2(y - y_{eq}) = \omega_n^2 k * u(t)$$

Where y(t) is the individual's signal, $\frac{dy}{dt}(t)$ is its first derivative,$\frac{d^2y}{dt}(t)$ its second derivative and u(t) is the excitation. The function estimates the coefficients $2\xi\omega_n, \omega_n^2 k$ and $\omega_n^2 y_{eq}$, from which the oscillation period T, the damping ratio ξ, the equilibrium y_{eq} value and the gain k can be derived. Th estimation is based on a two step procedure: the first step consists in estimating the derivatives to then estimate in a second step the differential equation coefficients through a linear mixed-effect model. Three different method to estimate the derivative during the first step are proposed.

Usage

```
analyze.2order(
  data,
  id = NULL,
  input = NULL,
  time = NULL,
  signal,
  dermethod = "gold",
  derparam = 3,
  order = 2,
  verbose = FALSE
)
```

Arguments

`data`	Is a data frame containing at least one column, that is the signal to be analyzed.
`id`	Is a CHARACTER containing the NAME of the column of data containing the identifier of the individual. If this parameter is not entered when calling the function, a single individual is assumed and a linear regression is done instead of the linear mixed-effects regression.
`input`	Is a CHARACTER or a VECTOR OF CHARACTERS containing the NAME(s) of data column(s) containing the EXCITATION vector(s). If this parameter is not entered when calling the function, the excitation is assumed to be unknown. In this case, the linear mixed-effect regression is still carried out but no coefficient is calculated for the excitation term. If no excitation term is supplied, one of the initial conditions is different from 0 (signal or derivative) and xi<1 the function will estimate a damped linear oscillator (DLO)
`time`	Is a CHARACTER containing the NAME of the column of data containing the time vector. If this parameter is not entered when calling the function, it is assumed that time steps are of 1 unit and the time vector is generated internally in the function.
`signal`	Is a CHARACTER containing the NAME of the column of the data frame containing the SIGNAL to be studied.

`dermethod`	is the derivative estimation method. The following methods are available: "gold","glla" and "fda"
`derparam`	If dermethod "glla" or "gold" are chosen, it is the embedding number, a positive integer containing the number of points to be used for the calculation of the derivatives. Its value by default is 3 as at least three points are needed for the calculation of the second derivative. If dermethod "fda" is chosen, this parameter is spar, the parameter related to the smoothing parameter lambda used in the penalization function of the function `smooth.spline` to estimate the derivative via splines (Functional Data Analysis)
`order`	is the maximum order of the derivative to estimate. Using an order higher than that of the maximum derivative to estimate (1 in first order differential equations and 2 in second order differential equations), for instance, order=4 might enhance derivative estimation (see doi: 10.1080/00273171.2015.1123138Chow et al.(2016))
`verbose`	Is a boolean that displays status messages of the function when set to 1.

Details

The analysis performs the following linear mixed-effects regression:

$$\ddot{y_{ij}} \sim (b_0 + b_{0j}) + (b_1 + b_{1j})\dot{y_{ij}} + (b_2 + b_{2j})y_{ij} + (b_3 + b_{3j})u_{ij} + e_{ij}$$

with i accounting for the time and j for the different individuals. e_{ij} are the residuals, $\dot{y_{ij}}$ is the first derivative and $\ddot{y_{ij}}$ the second derivative calculated on embedding points, and y_{ij} and u_{ij} are the signal and the excitation averaged on embedding points. The fixed effect coefficients estimated from the regression are:

- xi2omega: b_1 ($2\xi\omega^2$ from the differential equation)
- omega2: b_2 (ω^2 from the differential equation)
- komega2: b_3 ($k\omega^2$ from the differential equation)
- yeqomega2: b_0 ($\omega^2 y_{eq}$ from the differential equation)

The coefficients derived to characterize the signal are calculated as follows:

- the oscillation period $T = \sqrt{\frac{2\pi}{b_2}}$
- the damping factor xi: $xi = \frac{b_1}{2*\sqrt{b_2}}$
- the gain, k: $k = \frac{b_3}{b_1}$. It is the proportionality between the stationary increase of the signal and the excitation increase that caused it.
- the equilibrium value, yeq: $yeq = \frac{b_0}{b_2}$. It is the stationary value reached in the absence of excitation.

The estimation is performed using the function lmer if there are several individuals or lm if there is only one. With the above estimated parameters, the estimated signal is reconstructed for each individual by using the generate.2order function of this package (based on deSolve's ode function). The function returns five objects:

1. data- A data.frame including the input data, the intermediate calculations used to prepare the variables for the fit and the estimated trajectories for each individual.

   (a) signal_rollmean - calculation of the o order derivative of the signal over embedding points.
   (b) signal_derivate1 - calculation of the first derivative of the signal with the chosen method in embedding points/with smoothing parameter spar
   (c) signal_derivate2 - calculation of the second derivative of the signal with the chosen method in embedding points/with smoothing parameter spar
   (d) time_derivate - calculation of the moving average of the time vector over embedding points.
   (e) input_rollmean - calculation of the moving average of the excitation vector over embedding points.
   (f) estimated - the estimated signal calculated using deSolve's ode function with a second order model, the excitation provided as input and the coefficients obtained from the fit.

2. resultid- A data.frame including for each individual, listed by id number, the coefficients calculated (thus fixed + random component)
3. resultmean- A data.frame including the fixed effects of the coefficients mentioned above with their standard error, the coefficients characterizing the signal shape (i.e. the period, the damping factor, the gain and the equilibrium value), and the R2 resulting from the estimation
4. regression- A list containing the summary of the linear mixed-effects regression.

   As seen in the Description section, the print method by default prints only the resultmean element. Each one of the other objects can be accessed by indicating $ and their name after the result, for instance, for a DOREMI object called "result", it is possible to access the regression summary by typing result$regression.
5. derparam - contains the embedding number used to generate the results (if the derivative estimation method chosen is "glla" or "gold") or the smoothing parameter spar if the chosen method is fda

Value

Returns a summary of the fixed effect coefficients (see details)

See Also

`calculate.gold`, `calculate.glla`, `calculate.fda` to compute the derivatives, for details on embedding/spar See `generate.2order` for the generation of the solution of the second order differential equation.

Examples

```
# generate a panel of oscillating signals
test   <- generate.panel.2order(time = 0:100,
                              excitation = as.numeric(0:100>50),
                              period = 15,
                              xi = 0.3,
                              k = 2,
                              internoise = 0.2,
                              intranoise = 0.3,
                              nind = 10)
```

```
# plot the signal to analyze
plot(test)


# analyze them
res <- analyze.2order(data = test,
                      id = "id",
                      input = "excitation",
                      time =  "time",
                      signal = "signal",
                      derparam = 13)
res
plot(res)
```

`calculate.fda` *Calculation of derivatives using the Functional Data Analysis (FDA) method.*

Description

`calculate.fda` estimates the derivatives of a variable using the FDA method described in several sources, such as in doi: 10.1007/b98888Ramsay et al. (2009) and doi: 10.1080/00273171.2015.1123138Chow et al. (2016). This method estimates a spline function that fits all the data points and then derivates this function to estimate derivatives at those points. In order for the derivatives to exist, the function must be smooth. A roughness penalty function controlled by a smoothing parameter is then used. The estimations are done by using the R's base smooth.spline function.

Usage

```
calculate.fda(signal, time, spar = NULL, order = NULL)
```

Arguments

`signal`	is a vector containing the data from which the derivative is estimated.
`time`	is a vector containing the time values corresponding to the signal. Arguments signal and time must have the same length.
`spar`	is the smoothing parameter used by the roughness penalty function in the smooth.spline R function.
`order`	parameter not used, for consistency with calculate.glla and calculate.gold

Value

Returns a list containing two elements:

dtime- contains the initial time values provided.

dsignal- is a data.frame containing three columns and the same number of rows as the signal. The first column is the signal data points, the second is the first derivative evaluated at those points, and the third is the second derivative evaluated at those points.

Examples

```
#In the following example the derivatives for the function y(t) = t^2 are calculated.
#The expected results are:
#y'(t) = 2t and y''(t) = 2
time <- c(1:500)/100
signal <- time^2
result <- calculate.fda(signal = signal, time = time)
```

`calculate.glla` *Calculation of derivatives using the GLLA method*

Description

`calculate.glla` estimates the derivatives of a variable using the Generalized Local Linear Approximation (GLLA) method described in doi: 10.4324/9780203864746Boker et al.(2010). This method estimates the derivatives over a number of measurement points called the embedding number assuming an equally spaced time series.

Usage

```
calculate.glla(signal, time, embedding = 3, n = 2)
```

Arguments

`signal`	is the input vector containing the data from which the derivatives are estimated.
`time`	is a vector containing the time values corresponding to the signal. Arguments signal and time must have the same length.
`embedding`	is an integer indicating the embedding dimension, that is the number of points to consider for derivative calculation. Embedding must be at least #' 2 for the calculation of the first derivative (first order models) and at least 3 for the calculation of the second derivative (second order models).
`n`	is the maximum order of the derivative to calculate

Value

Returns a list containing three columns:

dtime- contains the time values in which the derivative was calculated. That is, the moving average of the input time over embedding points.

dsignal- is a data.frame containing n+1 columns and the same number of rows as the signal. The column k is the k-1 order derivative of the signal over embedding points.

embedding- number of points used for the derivative calculation.

n - the maximum derivative order calculated n

Examples

```
#In the following example the derivatives for the function y(t) = t^2 are calculated.
#The expected results are:
#y'(t) = 2t and y''(t) = 2
time <- c(1:500)/100
signal <- time^2
result <- calculate.glla(signal = signal, time = time, embedding = 5)
```

---

calculate.gold *Calculation of derivatives using the GOLD method*

Description

`calculate.gold` estimates the derivatives of a variable using the Generalized Orthogonal Local Derivative (GOLD) method described in doi: 10.1080/00273171.2010.498294Deboeck (2010). The code available on this paper was extracted and adapted for non constant time steps. This method allows calculating over a number of measurement points (called the embedding number) derivatives with uncorrelated errors.

Usage

```
calculate.gold(signal, time, embedding = 3, n = 2)
```

Arguments

`signal`	is a vector containing the data from which the derivative is estimated.
`time`	is a vector containing the time values corresponding to the signal. Arguments signal and time must have the same length.
`embedding`	is an integer indicating the number of points to consider for derivative calculation. Embedding must be greater than 1 because at least two points are needed for the calculation of the first derivative and at least 3 for the calculation of the second derivative.
`n`	is the maximum order of derivative to estimate.

Value

Returns a list containing the following elements:

dtime- contains the time values in which the derivative was calculated. That is, the moving average of the input time over embedding points.

dsignal- is a data.frame containing n+1 columns and the same number of rows as the signal. The column k is the k-1 order derivative of the signal over embedding points.

embedding- number of points used for the derivative calculation.

n - the maximum derivative order calculated n

Examples

```
#In the following example the derivatives for the function y(t) = t^2 are calculated.
#The expected results are:
#y'(t) = 2t and y''(t) = 2
time <- c(1:500)/100
signal <- time^2
result <- calculate.gold(signal = signal, time = time, embedding = 5)
```

---

cardio *Measurements of cardiac frequency in 21 patients during effort tests*

Description

Data containing time, cardiac frequency and load of a resistive bicycle run by patients during effort tests

Usage

```
data(cardio)
```

Format

A data frame with 1686 rows and 4 variables

id positive integer, arbitrary identifier of the patient

time positive real number, time since the beginning of the test, in seconds (s)

load positive real number, load of the resistive bicycle, effort that the patient needs to do, in watts (W)

hr positive real number, patient's cardiac rhythm, in heart beats per minute (1/min)

Source

Mongin et al. 2018, under review (future DOI will be inserted here) `errorcheck` *Displays error messages for the analysis function according to the nature of the error* `errorcheck` *displays error messages and/or warnings concerning the validity of input arguments provided to the analysis function*

Description

Displays error messages for the analysis function according to the nature of the error `errorcheck` displays error messages and/or warnings concerning the validity of input arguments provided to the analysis function

Usage

```
errorcheck(data, col_var)
```

Arguments

`data`	data.frame or data.table containing the data to be analyzed. Same object that is passed as input argument to the analysis function.
`col_var`	column variable. Contains a string that indicates the name of the column to analyze ("id","input",etc.)

Value

Doesn't return a value. Either displays directly the error message/warning or changes data type in the data.frame/data.table provided

`generate.1order` *Generation of the first order differential equation solution with deSolve, for given fixed coefficients and initial condition*

Description

`generate.1order` returns a data frame containing the time (supplied as input) and the solution to a first order differential equation. The coefficients are provided as inputs, as well as the initial condition

$$\frac{dy(t)}{dt} + \frac{(y(t) - yeq)}{\tau} = \frac{k}{\tau}u(t)$$

Where y(t) is the signal, dy(t) its derivative, τ is the decay time, k the gain and yeq the equilibrium value. u(t) is the source term of the equation, that is an external excitation perturbing the dynamics. The latter is provided as input or is set to null. The numerical solution is generated with deSolve.

Usage

```
generate.1order(
  time = 0:100,
  excitation = NULL,
  y0 = 0,
  t0 = NULL,
  tau = 10,
  k = 1,
  yeq = 0
)
```

Arguments

`time`	Is a vector containing the time values corresponding to the excitation signal.
`excitation`	Is a vector containing the values of the excitation signal (u(t) in the equation). If NULL, it is considered to be 0.
`y0`	Signal initial value y(t=t0). Default is 0
`t0`	Time corresponding to the signal initial value y(t=t0). Default is the minimum value of the time vector. Must be a value between minimum and maximum value of the time vector
`tau`	Signal decay time. It represents the characteristic response time of the solution of the differential equation. A negative value will produce divergence from equilibrium.
`k`	Signal gain. Default is 1. It represents the proportionality between the stationary increase of signal and the excitation increase that caused it. Only relevant if the excitation is non null.
`yeq`	Signal equilibrium value. Stationary value when the excitation term is 0.

Value

Returns a data.table containing three elements:

- y is a vector containing the values calculated with deSolve so that y is a solution to a first order differential equation with the constant coefficients provided as input.
- t is a vector containing the corresponding time values
- exc

Examples

```
generate.1order(t0 = 2.5,y0 = 2)
test <- generate.1order(time = 0:49, excitation = c(rep(0,10),rep(1,40)))
plot(test$t,test$y)
lines(test$t,test$exc,col = 2)

### see the influence of tau

different_tau <- data.table::rbindlist(lapply(1:5*4,function(x){
```

```
tmp <- generate.1order(t0 = 0,
                       y0 = 2,
                       tau = x)
tmp[,tau := as.factor(x)][]
}))

ggplot2::ggplot(data = different_tau,
                ggplot2::aes(t,y,color = tau))+
  ggplot2::geom_line()

### effect of the gain

different_gain <- data.table::rbindlist(lapply(1:5,function(x){
tmp <- generate.1order(
  time = 1:100,
  excitation = as.numeric(1:100 > 50),
  y0 = 0,
  tau = 10,
  k = x)
tmp[,k := as.factor(x)][]
}) )

  ggplot2::ggplot(different_gain)+
  ggplot2::geom_line(ggplot2::aes(t,y,color = k))+
  ggplot2::geom_line(ggplot2::aes(t,exc,color = "excitation"))
```

`generate.2order` *Generation of the second order differential equation solution with deSolve*

Description

`generate.2order` returns a data frame containing the time (supplied as input) and a simulated signal generated as a solution to a second order differential equation with constant coefficients that are provided as inputs:

$$\frac{d^2y}{dt} + 2\xi\omega_n\frac{dy}{dt} + \omega_n^2 y = \omega_n^2 k * u(t)$$

Where: y(t) is the signal, $\frac{dy}{dt}$ its derivative and $\frac{d^2y}{dt}$ its second derivative

- $\omega_n = \frac{2\pi}{T}$ -where T is the period of the oscillation- is the system's natural frequency, the frequency with which the system would vibrate if there were no damping. The term ω_n^2 represents thus the ratio between the attraction to the equilibrium and the inertia. If we considered the example of a mass attached to a spring, this term would represent the ratio of the spring constant and the object's mass.
- ξ is the damping ratio. It represents the friction that damps the oscillation of the system (slows the rate of change of the variable). The term $2\xi\omega_n$ thus represents the respective contribution of the inertia, the friction and the attraction to the equilibrium. The value of ξ determines the shape of the system time response, which can be: $\xi < 0$ Unstable, oscillations of increasing

magnitude $\xi = 0$ Undamped, oscillating $0 < \xi < 1$ Underdamped or simply "damped": the oscillations are damped by an exponential of damping rate $\xi\omega_n$ $\xi = 1$ Critically damped $\xi > 1$ Over-damped, no oscillations in the return to equilibrium

- k is the gain
- u(t) is an external excitation perturbing the dynamics

The excitation is also provided as input and it can be null (then the solution will be a damped linear oscillator when the initial condition is different from 0)

Usage

```
generate.2order(
  time = 0:100,
  excitation = NULL,
  y0 = 0,
  v0 = 0,
  t0 = NULL,
  xi = 0.1,
  period = 10,
  k = 1,
  yeq = 0
)
```

Arguments

`time`	is a vector containing the time values corresponding to the excitation signal.
`excitation`	Is a vector containing the values of the excitation signal.
`y0`	is the initial condition for the variable y(t=t0), (0, by default), it is a scalar.
`v0`	is the initial condition for the derivative dy(t=t0), (0, by default), it is a scalar.
`t0`	is the time corresponding to the initial condition y0 and v0. Default is the minimum value of the time vector.
`xi`	is the damping factor. A negative value will produce divergence from equilibrium.
`period`	is the period T of the oscillation, $T = \frac{2*\pi}{\omega_n}$ as mentioned
`k`	Default is 1. It represents the proportionality between the stationary increase of signal and the excitation increase that caused it. Only relevant if the excitation is non null.
`yeq`	is the signal equilibrium value, i.e. the stationary value reached when the excitation term is 0.

Value

Returns a data.table containing four elements:

- t is a vector containing the corresponding time values

- y is a vector containing the values calculated with deSolve so that y is a solution to a second order differential equation with constant coefficients (provided as input) evaluated at the time points given by t
- dy is a vector containing the values of the derivative calculated at the same time points
- exc is the excitation vector

Examples

```
generate.2order(time=0:249,excitation=c(rep(0,10),rep(1,240)),period=10)
generate.2order(y0=10)
```

`generate.excitation` *Excitation signal generation*

Description

`generate.excitation` generates a vector of randomly located square pulses with a given amplitude, duration and spacing between the pulses. A pulse is where the excitation passes from value 0 to value amplitude for a given duration and then returns back to 0, thus producing a square shape.

Usage

```
generate.excitation(
  amplitude = 1,
  nexc = 1,
  duration = 2,
  deltatf = 0.1,
  tmax = 10,
  minspacing = 1
)
```

Arguments

`amplitude`	is a vector of values different from 0 indicating the amplitude of the excitation. It should contain as many values as the number of pulses (nexc). If the elements are less than the number of pulses, the amplitude vector will be "recycled" and the elements from it will be repeated until all the pulses are covered (for instance, if the number of excitations nexc is 6 and the amplitude vector has two elements, pulses 1,3 and 5 will have the same amplitude as the first element of the amplitude vector and pulses 2,4 and 6 that of the second element).
`nexc`	is an integer greater than 0 indicating the number of pulses to generate.
`duration`	is a vector of values greater or equal to 0 indicating the duration of each pulse in time units. It should have as many elements as the number of pulses (nexc). If the elements are less than the number of pulses, the amplitude vector will be "recycled" and the elements from it will be repeated until all the pulses are covered.

`deltatf`	is a value greater than 0 indicating the time step between two consecutive data points.
`tmax`	is a value greater than 0 indicating the maximum time range of the excitation vector in time units. The time vector generated will go from 0 to tmax.
`minspacing`	as pulses are generated randomly, minspacing is a value greater than or equal to 0 that indicates minimum spacing between pulses, thus avoiding overlapping of the pulses in time. A value of 0 indicates that pulses can follow one another.

Details

Used for simulations in the context of the package. Beware that the following condition should apply:

$$tmax >= (duration + minspacing) * nexc$$

so that the pulses "fit" in the time lapse defined. Compared to `pulsew` from the `seewave` package, this function can generate pulses of different duration and amplitude.

Value

Returns two vectors:

E- vector containing the values of the excitation generated.

t- vector containing the values of time generated.

Examples

```
generate.excitation (amplitude = 3,
                     nexc = 6,
                     duration = 2,
                     deltatf = 1,
                     tmax = 200,
                     minspacing = 2)
#Vector of length 201 (deltatf x tmax + 1 as it includes 0 as initial time value)
generate.excitation (amplitude = c(1,10,20),
                     nexc = 3,
                     duration = c(1,2,4),
                     deltatf = 0.5,
                     tmax = 100,
                     minspacing = 10)
```

`generate.panel.1order` *Generation of first order differential equation solutions for several individuals with intra-individual and inter-individual noise*

Description

`generate.panel.1order` Generation of first order differential equation solutions for several individuals with intra-individual and inter-individual noise. For a panel of nind individual, the function generates nind solutions of a first order differential equation with constant coefficients distributed along a normal distribution. Measurement noise is added to each individual signal according to the value of the intranoise parameter.

Usage

```
generate.panel.1order(
  time,
  excitation = NULL,
  y0 = 0,
  t0 = NULL,
  tau = 10,
  k = 1,
  yeq = 0,
  nind = 1,
  internoise = 0,
  intranoise = 0
)
```

Arguments

`time`	Is a vector containing the time values corresponding to the excitation signal.
`excitation`	Is a vector containing the values of the excitation signal (u(t) in the equation). If NULL, it is considered to be 0.
`y0`	Signal initial value y(t=t0). Default is 0
`t0`	Time corresponding to the signal initial value y(t=t0). Default is the minimum value of the time vector. Must be a value between minimum and maximum value of the time vector
`tau`	Signal decay time. It represents the characteristic response time of the solution of the differential equation. A negative value will produce divergence from equilibrium.
`k`	Signal gain. Default is 1. It represents the proportionality between the stationary increase of signal and the excitation increase that caused it. Only relevant if the excitation is non null.
`yeq`	Signal equilibrium value. Stationary value when the excitation term is 0.
`nind`	number of individuals.
`internoise`	Is the inter-individual noise added. The tau across individuals follows a normal distribution centered on the input parameter tau with a standard deviation of internoise*tau, except if any decay time is negative (see Details section). The same applies to the other coefficients of the differential equation (k and yeq)
`intranoise`	Is the noise to signal ratio: dynamic noise added to each signal defined as the ratio between the variance of the noise and the variance of the signal

Details

Used for simulations in the context of the package. The function currently simulates only positive decay times corresponding to a regulated system. When the decay time is low and the inter individual noise is high, some individuals' decay time could be negative. In that case, the decay time distribution is truncated at 0.1*deltat and values below are set to this limit. High values are symmetrically set at the upper percentile value similar to a Winsorized mean. A warning provides the initial inter individual noise set as input argument and the inter individual noise obtained after truncation.

Value

Returns a data frame containing the following columns:

- id - individual identifier (from 1 to nind).
- excitation - excitation signal
- time - time values
- signalraw - signal with no noise (internoise provided added for each individual)
- dampedsignal - signal with intranoise added

See Also

`generate.1order` for calculation of the numerical solution to the differential equation and `generate.excitation` for excitation signal generation

Examples

```
generate.panel.1order(time = generate.excitation(3, 6, 2, 1, 200, 2)$t,
                      excitation = generate.excitation(3, 6, 2, 1, 200, 2)$exc,
                      y0 = 0,
                      tau = 10,
                      k = 1,
                      yeq = 0,
                      nind = 5,
                      internoise = 0.2,
                      intranoise = 1)
```

`generate.panel.2order` *Generation of second order differential equation solutions for several individuals with intra-individual and inter-individual noise*

Description

`generate.panel.2order` Generation of second order differential equation solutions for several individuals with intra-individual and inter-individual noise. The function generates the equation coefficients following a normal distribution based on the parameter internoise and the coefficients provided as input. It then calls the function `generate.2order` to generate a solution of a second order differential equation with these parameters for the nind individuals. Finally it adds measurement noise to each signal according to the value of the parameter intranoise.

Usage

```
generate.panel.2order(
  time,
  excitation = NULL,
  y0 = 1,
  v0 = 0,
  t0 = NULL,
  xi = 0.1,
  period = 10,
  k = 1,
  yeq = 0,
  nind = 1,
  internoise = 0,
  intranoise = 0
)
```

Arguments

`time`	is a vector containing the time values corresponding to the excitation signal.
`excitation`	Is a vector containing the values of the excitation signal.
`y0`	is the initial condition for the variable y(t=t0), (0, by default), it is a scalar.
`v0`	is the initial condition for the derivative dy(t=t0), (0, by default), it is a scalar.
`t0`	is the time corresponding to the initial condition y0 and v0. Default is the minimum value of the time vector.
`xi`	is the damping factor. A negative value will produce divergence from equilibrium.
`period`	is the period T of the oscillation, $T = \frac{2*\pi}{\omega_n}$ as mentioned
`k`	Default is 1. It represents the proportionality between the stationary increase of signal and the excitation increase that caused it. Only relevant if the excitation is non null.
`yeq`	is the signal equilibrium value, i.e. the stationary value reached when the excitation term is 0.
`nind`	number of individuals.
`internoise`	Is the inter-individual noise added. The damping factor across individuals follows a normal distribution centered on the input parameter xi with a standard deviation of internoise*xi. The same applies to the other coefficients of the differential equation (T,k and yeq) and to the initial conditions (y0 and v0)
`intranoise`	Is the noise to signal ratio: dynamic noise added to each signal defined as the ratio between the variance of the noise and the variance of the signal

Details

Used for simulations in the context of the package.

Value

Returns a data frame with signal and time values for the time and excitation vectors provided. It contains the following columns:

- id - individual identifier (from 1 to nind).
- excitation - excitation signal provided as input
- time - time values provided as input
- signalraw - signal with no noise (inter noise added for each individual)
- signal - signal with intra noise added

See Also

`generate.2order` for calculation of the numerical solution to the second order differential equation and `generate.excitation` for excitation signal generation

Examples

```
generate.panel.2order(time = generate.excitation(3, 6, 2, 1, 200, 2)$t,
                      excitation = generate.excitation(3, 6, 2, 1, 200, 2)$exc,
                      xi = 0.1,
                      period = 0.5,
                      k = 1,
                      nind = 5,
                      internoise = 0.2,
                      intranoise = 0.1)
```

---

optimum_param — *Function to find the optimum parameter for derivative estimation (embedding or spar according to derivative estimation method chosen)*

Description

`optimum_param` calculates the optimum parameter for derivative estimation by varying the latter in a range introduced as input and keeping the parameter and coefficients having the $R^2$ closest to 1.

Usage

```
optimum_param(
  data,
  id = NULL,
  input = NULL,
  time,
  signal,
  dermethod = "gold",
  model = "1order",
  order = 2,
  pmin = 3,
  pmax = 21,
  pstep = 2,
  verbose = FALSE
)
```

Arguments

`data`	Is a data frame containing at least one column, that is the signal to be analyzed.

`id`	Is a CHARACTER containing the NAME of the column of data containing the identifier of the individual. If this parameter is not entered when calling the function, a single individual is assumed and a linear regression is done instead of the linear mixed-effects regression.
`input`	Is a CHARACTER or a VECTOR OF CHARACTERS containing the NAME(s) of data column(s) containing the EXCITATION vector(s). If this parameter is not entered when calling the function, the excitation is assumed to be unknown. In this case, the linear mixed-effect regression is still carried out but no coefficient is calculated for the excitation term. The function then uses the parameters estimated by the regression to carry out an exponential fit of the signal and build the estimated curve. The function will consider as an excitation each column of data having a name contained in the input vector. The function will return a coefficient for each one of the excitation variables included in the input vector.
`time`	Is a CHARACTER containing the NAME of the column of data containing the time vector. If this parameter is not entered when calling the function, it is assumed that time steps are of 1 unit and the time vector is generated internally in the function.
`signal`	Is a CHARACTER containing the NAME of the column of the data frame containing the SIGNAL to be studied.
`dermethod`	is the derivative estimation method. The methods currently available are: "gold","glla" and "fda" (see their respective function for more details)
`model`	is the model to be used for analysis of the signal. The models available are "1order" and "2order"
`order`	is the maximum order of the derivative estimated when using `calculate.gold` or `calculate.glla`. Using a higher order can enhance derivative estimation (see doi: 10.1080/00273171.2015.1123138Chow et al.(2016))
`pmin`	is the minimum of the interval in which to vary the parameter (embedding number or spar according to derivative method chosen)
`pmax`	is the maximum of the interval in which to vary the parameter (embedding number or spar according to derivative method chosen)
`pstep`	is the step that will be considered when varying the parameter. For instance pmin=3, pmax=7 and pstep=2 and dermethod="gold" will make the embedding number take the values 3,5 and 7.
`verbose`	Is a boolean that displays status messages of the function (and functions it calls) when set to TRUE.

Value

Returns a list of three objects:

- analysis is a data.frame containing the resultmean object of the analysis made (result of the analyze.1order or analyze.2order function according to model chosen) with the different values of embedding/spar and the resulting $R^2$.
- summary_opt is a data.frame containing the analysis that had the best $R^2$ from the analysis data.frame previously mentioned
- d contains the optimum value of the embedding/spar

See Also

analyze.1order and analyze.2order for the estimation of equation coefficients in signals following a first and second order differential equation respectively

Examples

```
s2 <- generate.panel.2order(time = 0:100,
                            excitation = c(rep(0,25),rep(1,76)),
                            y0 = 0,
                            v0= 0,
                            xi = 0.05,
                            period=10,
                            k=1,
                            yeq=0,
                            nind=4,
                            internoise = 0.2,
                            intranoise = 8)
resgold <- optimum_param (data=s2,
                          id="id",
                          input="excitation",
                          time="time",
                          signal="signal",
                          model = "2order",
                          dermethod = "gold",
                          pmin = 3,
                          pmax = 13,
                          pstep = 2,
                          verbose = TRUE)
```

---

plot.doremi *S3 method to plot DOREMI objects*

Description

plot.doremi generates a plot with the observed values of the signal, the excitation values and the fitted signal over time for each individual.

Usage

```
## S3 method for class 'doremi'
plot(x, ..., id = NULL)
```

Arguments

x,	DOREMI object resulting from analyze.1order or analyze.2order analysis
...	includes the additional arguments inherited from the generic plot method
id	Identifiers of the individuals to be represented in the plot. By default, it will print the first six individuals.

Value

Returns a plot with axis labels, legend and title. The axis labels and legend include the names of the variables set as input arguments. The title includes the name of the DOREMI object result of the analysis. The function uses `ggplot` to generate the graphs and so it is possible to override the values of axis labels, legend and title through ggplot commands.

Examples

```
mydata <- generate.panel.1order(time= 0:100,
                                excitation = sin(0:100),
                                y0 = 0,
                                t0 = 0,
                                tau = 2,
                                k = 1,
                                yeq = 0,
                                nind = 2,
                                internoise = 0.1,
                                intranoise = 8)
myresult <- analyze.1order(data = mydata,
                           id = "id",
                           input = "excitation",
                           time = "time",
                           signal = "signal")
plot(myresult)
```

`plot.doremidata` *S3 method to plot DOREMIDATA objects*

Description

`plot.doremidata` generates a plot of the simulated signals resulting from the `generate.panel.1order` and `generate.panel.2order` functions

Usage

```
## S3 method for class 'doremidata'
plot(x, ...)
```

Arguments

`x`	DOREMIDATA object resulting from the aforementioned functions
`...`	includes the additional arguments inherited from the generic plot method

Value

Returns a plot with axis labels, legend and title. The title includes the name of the DOREMIDATA object result of the analysis. The function uses `ggplot` to generate the graphs and thus it is possible to override the values of axis labels, legend and title through ggplot commands.

Examples

```
mydata <- generate.panel.1order(time=0:100,
                                excitation = c(rep(0,50),rep(1,51)),
                                nind = 6,
                                internoise = 0.2,
                                intranoise = 100)
plot(mydata)
```

`plot.doremiparam` *S3 method to plot DOREMIPARAM objects*

Description

`plot.doremiparam` generates a plot of the parameters resulting from the `optimum_param` function

Usage

```
## S3 method for class 'doremiparam'
plot(x, ...)
```

Arguments

`x`	DOREMIPARAM object resulting from the aforementioned function
`...`	includes the additional arguments inherited from the generic plot method

Value

Returns a plot showing the evolution of the first/second order differential equation coefficients and R2 with the values taken by the embedding number/smoothing parameter (see details of `optimum_param` function). The function uses `ggplot` to generate the graphs and thus it is possible to override the values of axis labels, legend and title through ggplot commands.

Examples

```
mydata <- generate.panel.1order(time = 0:130,
                          excitation = c(rep(0,30),rep(1,50),rep(0,51)),
                          nind = 5,
                          internoise = 0.2,
                          intranoise = 100)
myres<- optimum_param (data = mydata,
                         id = "id",
                         input ="excitation",
                         time = "time",
                         signal = "signal",
                         model = "1order",
                         dermethod = "gold",
                         pmin = 3,
                         pmax = 11,
                         pstep = 2)
plot(myres)
```

predict.doremi — *S3 method to predict signal values in a DOREMI object when entering a new excitation*

Description

`predict.doremi` predicts signal values with a DOREMI object when providing a new excitation vector(s).

Usage

```
## S3 method for class 'doremi'
predict(object, ..., newdata, verbose = FALSE)
```

Arguments

`object`	DOREMI object result of an analysis with the function remi
`...`	Additional arguments inherited from generic predict method.
`newdata`	includes a data frame containing three columns or more: id (optional), indicating the individual identifier time, containing the time values excitation, being one or several columns containing the different excitations used to estimate a new signal. As in the other methods for the predict function, the columns of newdata must have the same names as those of the original object.
`verbose`	Is a boolean that displays status messages of the function when set to 1.

Value

Returns a list containing the values of time, the values of the excitation and the predicted values of the signal for the new excitation(s).

Examples

```
myresult <- analyze.1order(data = cardio[id==1],
                id="id",
                input = "load",
                time = "time",
                signal = "hr")
#Copying cardio into a new data frame and modifying the excitation column
new_exc <- cardio[id==1]
et <- generate.excitation(amplitude = 100,
                         nexc = 6,
                         duration = 2,
                         deltatf = 1,
                         tmax = 49,
                         minspacing = 2)
new_exc$load <- et$exc
```

```
new_exc$time <- et$t
predresult <- predict(myresult, newdata = new_exc)
```

print.doremi *S3 method to print DOREMI objects*

Description

`print.doremi` prints the most important results of a DOREMI object

Usage

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

Arguments

`x`	DOREMI object
`...`	includes the additional arguments inherited from the generic print method

Value

Returns the coefficients of the differential equation estimated (fixed coefficients, table $resultmean of the DOREMI object)

Examples

```
myresult <- analyze.1order(data = cardio,
                 id = "id",
                 input = "load",
                 time = "time",
                 signal = "hr")
myresult
```

print.doremidata *S3 method to print DOREMIDATA objects*

Description

`print.doremidata` prints the a DOREMIDATA object

Usage

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

Arguments

`x`	DOREMIDATA object
`...`	includes the additional arguments inherited from the generic print method

Value

Returns the DOREMIDATA object (datatable))

Examples

```
time <- 0:100
data <- generate.panel.2order(time = time,
                              y0 = 10,
                              v0 = 0,
                              xi = 0.1,
                              period = 30,
                              k = 1,
                              yeq = 2,
                              nind = 6,
                              internoise = 0.3,
                              intranoise = 5)
data
```

`rotation` *Measurements of response time of 17 individuals when carrying out mental rotation tasks*

Description

Data containing reaction time to a mental rotation task over a 60 day period for 17 individuals doi: 10.1016/j.yhbeh.2012.12.007(Courvoisier et al., 2013).

Usage

```
data(rotation)
```

Format

A data frame with 619 rows and 5 variables

id positive integer, arbitrary identifier of the individual

sex character, sex of the individual, as the study highlighted the difference in response time according to sex

days positive integer,day since the beginning of the experiment

meanRT positive integer, mean response time of the individual to execute the mental rotation task, in milliseconds (ms)

logmeanRT natural logarithm of the mean response time

Source

Delphine S. Courvoisier, Olivier Renaud, Christian Geiser, Kerstin Paschke, Kevin Gaudy, Kirsten Jordan, Sex hormones and mental rotation: An intensive longitudinal investigation,

Hormones and Behavior,

Volume 63, Issue 2,

2013,

Pages 345-351,

doi: 10.1016/j.yhbeh.2012.12.007

`summary.doremi` *S3 method for DOREMI object summary*

Description

`summary.doremi` provides a summary containing the five lists of the DOREMI object

Usage

```
## S3 method for class 'doremi'
summary(object, ...)
```

Arguments

`object,`	DOREMI object (contains several lists)
`...`	includes the additional arguments inherited from the generic summary method

Value

Returns a summary containing the five lists of the DOREMI object

Examples

```
myresult <- analyze.1order(data = cardio,
                 id = "id",
                 input = "load",
                 time = "time",
                 signal = "hr")
summary(myresult)
```

Index