Package ‘deFit’

June 29, 2023

Type Package

Title Fitting Differential Equations to Time Series Data

Version 0.2.1

Description
Use numerical optimization to fit ordinary differential equations (ODEs) to time series data to examine the dynamic relationships between variables or the characteristics of a dynamical system. It can now be used to estimate the parameters of ODEs up to second order, and can also apply to multilevel systems. See <`https://github.com/yueqinhu/defit`> for details.

License GPL (>= 3)

URL `https://github.com/yueqinhu/defit`

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Imports deSolve, ggplot2, stats, R6

Depends R (>= 3.50)

NeedsCompilation no

Author Yueqin Hu [aut, cre],
Qingshan Liu [aut]

Maintainer Yueqin Hu <`yueqinhu@bnu.edu.cn`>

Repository CRAN

Date/Publication 2023-06-29 14:40:08 UTC

R topics documented:

`AdjustModel_func` *Adjust the model function*

Description

Adjust the model function

Usage

```
AdjustModel_func(model)
```

Arguments

`model`	The original string of users' defined.

Value

dataframe data.frame(model)

`CalcDe_func` *Choose the core code by the model.*

Description

Choose the core code by the model.

Usage

```
CalcDe_func(
  data,
  model,
  guess = c(0, 0, 0, 0, 0, 0),
  method,
  chooseModel = NULL,
  guess2,
  method2
)
```

Arguments

`data`	Users' data
`model`	The original string users defined.
`guess`	The guess values users input.
`method`	The method users selected.
`chooseModel`	c(2,1): Bivariate first-order differential equation; c(1,2): Univariable second-order differential equation.
`guess2`	Use for Multilevel
`method2`	"Nelder-Mead", "BFGS", "CG", "L-BFGS-B", "SANN" and "Brent"

Value

The result of optimization,SE,RMSE,r-squared,users' data,predictor data and output table.

`defit` *Fitting Differential Equations to Time Series Data*

Description

Use numerical optimization to fit ordinary differential equations (ODEs) to time series data to examine the dynamic relationships between variables or the characteristics of a dynamical system. It can now be used to estimate the parameters of ODEs up to second order, and can also apply to multilevel systems.

Usage

```
defit(data, model, guess = NULL, method = NULL, plot = FALSE)
```

Arguments

`data`	a data frame containing all model variables. The "time" column must be included.
`model`	a string specifying the model to be used. The "=~" operator is used to define variables, with the name of the variable user defined on the left and the name of the variable in the data on the right. The '~' operator specifies a differential equation, with the dependent variable on the left and the independent variables on the right. See also 'Details'.
`guess`	an optional vector that allows the user to give starting values for the model parameters, including the model coefficients and variable initial states.
`method`	an optional string indicating which optimizer to use. The default method is subject to the specific model. The available options are 'Nelder-Mead','L-BFGS-B','SANN' and 'BFGS'.
`plot`	an optional TRUE or FALSE that TRUE will draw the plot of the raw data and the predicted values.

Details

We suggest choosing the method by default. The guess values contain the coefficient of the model and initial values (the values of t0). Different models have different number of values.

Time(param) sequence for which output is wanted; the first value of times must be the initial time.

```
# eg1. An example of the univariate second-order differential equation (damped oscillator model)
data('example1')
model1 <- '
   X =~ myX
   time =~ myTime
   X(2) ~ X(1) + X
  '
result1 <- defit(data = example1, model = model1)
#> Program will fit the data with a univariate second-order differential equation.
#> The differential equation is:
#> x(2) = beta1 * x + beta2 * x(1)
#> Optimizing...
#> Finishing optimization...
#> Estimating R_squared
#> Estimating RMSE
#> Estimating Hessian
# result1$table get the result
# names(result1) get all names of object

#--------------
# eg3. An example of the multilevel univariate second-order differential equation
data('example3')
model3 <- '
   X =~ current
   time =~ myTime
   X(2) ~ X(1) + X + (1 + X(1) + X | year)
   '
example3_use <- example3[(example3["year"] >= 2015)&(example3["year"] <= 2018),] # Note: select a subse
example3_c <- scale_within(example3_use, model3) # note: centering X variable by year
result3 <- defit(data=example3_c,model = model3,plot=FALSE)
#> subject checking
#> Random effects and fixed effects have been defined
#> Estimating population parameter
#> Waiting...
#> next...subject...2015
#> next...subject...2016
#> next...subject...2017
#> next...subject...2018
#> Estimating R_squared
#> Estimating RMSE
#> Estimating Hessian

#--------------
```

```
# eg4. An example of the multilevel bivariate first-order differential equations
data('example3')
model4 <- '
   X =~ current
   Y =~ expected
   time =~ myTime
   X(1) ~ X + Y + (1 + X + Y | year)
   Y(1) ~ X + Y + (1 + X + Y | year)
   '
example4_use <- example3[(example3["year"] >= 2015)&(example3["year"] <= 2018),] # Note: select a subse
example4_c <- scale_within(example4_use, model4) # centering X and Y variable by year
result4 <- defit(data=example4_c,model = model4,plot=FALSE)
#> subject checking
#> Random effects and fixed effects have been defined
#> Estimating parameters
#> Waiting...
#> next...subject...2015
#> next...subject...2016
#> next...subject...2017
#> next...subject...2018
#> Estimating R_squared
#> Estimating RMSE
#> Estimating Hessian
```

Value

object: directly type the defit object will print all results. The function summary is used to print the summary of all results, and the exact values of each result can be extracted by the "$" operator.

userdata: the data that contains a sequence 'seq' starting from 1, the original time variable 'time', and all other variables user defined.

parameter: the best set of parameters found, including parameter values, gradient, convergence, message and hessian matrix.

predict: a dataframe of model predicted variable states at each time point.

r_squared: r_squared is the square of the correlation between the observed values and the predicted values, representing the proportion of variance explained by the model.

RMSE: RMSE (Root Mean Squared Error) is the standard deviation of the residuals.

SE: a symmetric matrix giving standard error of the model parameters.

equation: a string prints the estimated differential equations and initial states.

table: a summary table of parameter estimates and their corresponding SEs.

convergence: a message returns the result of the optimization convergence check.

Examples

```
#eg2. An example of bivariate first-order differential equation
data('example2')
model2 <- '
```

```
        # define variable
        X =~ myX
        Y =~ myY

        # define time
        time =~ myTime

        # define differential equation
        X(1) ~ X + Y
        Y(1) ~ Y + X
      '
    result2 <- defit(data = example2, model = model2)
    result2
    # extract details and values
    # result2$summary()
    # result2$userdata
    # result2$parameter$par
    # result2$equation
    # result2$table
    # result2$plot()
```

example1 — *Univariate second-order differential equation*

Description

A dataset containing the myX and time of almost 100 example. The variables are as follows:

Usage

```
example1
```

Format

A data frame with 100 rows and 3 variables:

seq sequence of observations

myTime timestamp of observations; the first value of the time variable must be the initial time.

myX the observed scores

`example2` *Bivariate first-order differential equation*

Description

A dataset containing the myX, myY and time of almost 30 example. The variables are as follows:

Usage

```
example2
```

Format

A data frame with 30 rows and 4 variables:

myTime timestamp of observations; the first value of the time variable must be the initial time.

myX the observed scores of variable X

myY the observed scores of variable Y

`example3` *University of Michigan consumer sentiment index*

Description

The Surveys of Consumers are conducted by the Survey Research Center at the University of Michigan. Founded in 1946 by George Katona, the surveys have long stressed the important influence of consumer spending and saving decisions in determining the course of the national economy.

Usage

```
example3
```

Format

A data frame with 540 rows and 6 variables:

seq sequence of observations

month months of data

year from 1978 to 2022

current the Index of Current Economic Conditions (ICC)

expected the Index of Consumer Expectations (ICE)

myTime months converted to time series

Source

University of Michigan, Survey Research Center, Surveys of Consumers. https://data.sca.isr.umich.edu/

`JudgeModel_func` *Judge the model belongs of N-variable and N-order differential equation.*

Description

Judge the model belongs of N-variable and N-order differential equation.

Usage

```
JudgeModel_func(model)
```

Arguments

`model`	The original string users defined.

Value

c(2,1): Bivariate first-order DE; c(1,2): Univariable second-order DE.

`PlotDe_func` *Plot all data*

Description

Plot all data

Usage

```
PlotDe_func(userdata, predictor, model, chooseModel)
```

Arguments

`userdata`	User's data.
`predictor`	The data of differential equation model
`model`	The model of standardization
`chooseModel`	c(2,1) or c(2,1)

Value

The plot of all data.

scale_within *Title*

Description

Title

Usage

```
scale_within(userdata, model = NA, center = FALSE, scale = FALSE)
```

Arguments

`userdata`	users' data
`model`	a string specifying the model to be used. The "=~" operator is used to define variables, with the name of the variable user defined on the left and the name of the variable in the data on the right. The '~' operator specifies a differential equation, with the dependent variable on the left and the independent variables on the right. See also 'Details'.
`center`	TRUE or FALSE
`scale`	TRUE or FALSE

Value

dataframe

Examples

```
#eg1.
data('example3')
multi_model <- '
  X =~ current
  time =~ myTime
  X(2) ~ X(1) + X + (1 + X(1) + X | year)
  '
scale_mydata <- scale_within(example3[(example3["year"] >= 2015)&(example3["year"] <= 2018),]
,multi_model
,center=TRUE)
```

`Slover_UniSec_func` *Core code of univariable second-order differential equation*

Description

Core code of univariable second-order differential equation

Usage

```
Slover_UniSec_func(data, model, guess, method)
```

Arguments

`data`	User's data
`model`	model's class is dataframe.
`guess`	Guess values that contain coefficient and initial values.
`method`	"Nelder-Mead", "BFGS", "CG", "L-BFGS-B", "SANN" and "Brent"

Value

The result of optimization, SE, RMSE, r-squared, users' data, predictor data and output table.

`Solver_BinFirst_func` *Core code of Binary first-order differential equational*

Description

Core code of Binary first-order differential equational

Usage

```
Solver_BinFirst_func(data, model, guess, method)
```

Arguments

`data`	User's data
`model`	Model's class is dataframe.
`guess`	Guess values that contain coefficient and initial values.
`method`	"Nelder-Mead", "BFGS", "CG", "L-BFGS-B", "SANN" and "Brent"

Value

The result of optimization,SE,RMSE,r-squared,users's data,predictor data and output table.

Solver_MultiBiFirst_func *Title*

Description

Title

Usage

```
Solver_MultiBiFirst_func(data, model, guess, method, guess2, method2)
```

Arguments

`data`	User's data
`model`	Model's class is dataframe.
`guess`	Guess values that contain coefficient and initial values.
`method`	"Nelder-Mead", "BFGS", "CG", "L-BFGS-B", "SANN" and "Brent"
`guess2`	Guess values of multilevel that contain coefficient and initial values.
`method2`	"Nelder-Mead", "BFGS", "CG", "L-BFGS-B", "SANN" and "Brent"

Value

The result of optimization,SE,RMSE,r-squared,users's data,predictor data and output table.

Solver_MultiUniSec_func *Title*

Description

Title

Usage

```
Solver_MultiUniSec_func(data, model, guess, method, guess2, method2)
```

Arguments

`data`	User's data
`model`	Model's class is dataframe.
`guess`	Guess values that contain coefficient and initial values.
`method`	"Nelder-Mead", "BFGS", "CG", "L-BFGS-B", "SANN" and "Brent"
`guess2`	Guess values of multilevel that contain coefficient and initial values.
`method2`	"Nelder-Mead", "BFGS", "CG", "L-BFGS-B", "SANN" and "Brent"

Value

The result of optimization,SE,RMSE,r-squared,users's data,predictor data and output table. init02', init03', init04'

Index