Package: deFit (via r-universe)

May 24, 2026

Type Package

Title Fitting Differential Equations to Time Series Data

Version 0.3.0

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)

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], Minglan Li [aut]

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

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

Date/Publication 2024-10-18 11:10:13 UTC

RemoteUrl https://github.com/cran/deFit

RemoteRef HEAD

RemoteSha f2594abcde9d55e08a7adbf83b1c38975ee20fdb

Contents

`calcDerivatives` *Calculating the Derivatives*

Description

Calculating the Derivatives

Usage

```
calcDerivatives(data, column, groupby, time = NA, order = 2, window = 5)
```

Arguments

`data`	a data frame.
`column`	names of variables in the long format that correspond to multiple variables in the wide format.
`groupby`	Character vector. Only used if the data is in a data.frame.
`time`	A variable name in the data frame containing sampling time information.
`order`	integer scalar.
`window`	integer scalar.Must be an odd number

Details

examples

```
#eg1.
derivatives1 <- calcDerivatives(data=example3,column='expected',groupby='year',order=2,window=5,inte
#eg2.
derivatives2 <- calcDerivatives(data=example3,column=c('expected','current'),groupby='year',time='my
#eg3.
derivativese3 <- calcDerivatives(data=example3,column=c('expected','current'),groupby='year',order=2
```

Value

a data frame contains derivatives.

`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,
  fixed_first = TRUE
)
```

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.
fixed_first	an optional TRUE or FALSE that TRUE will estimate the multilevel model parameters using a two-step approach.

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 + X(1)
  '
result1 <- defit(data = example1, model = model1)
# 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 + X(1) + (1 + X + X(1) | 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)

#--------------
# 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)
```

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
model2 <- '
    # define variable
    X =~ myX
    Y =~ myY

    # define time
    time =~ myTime

    # define differential equation
    X(1) ~ X + Y
    Y(1) ~ X + Y
  '
result2 <- defit(data = example2, model = model2,method='Nelder-Mead')
# Note: the method argument will override the default "L-BFGS-B" method
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 15 example. The variables are as follows:

Usage

```
example2
```

Format

A data frame with 15 rows and 3 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/

example4 *Bivariate first-order differential equation*

Description

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

Usage

```
example4
```

Format

A data frame with 15 rows and 3 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

Info_func *Calculate R-squared RMSE SE and so on.*

Description

Calculate R-squared RMSE SE and so on.

Usage

```
Info_func(solve_data, userdata, predict, var_model, table)
```

Arguments

solve_data	a list data, that is a solve of differential equations.
userdata	a data frame containing all model variables. The "time" column must be included.
predict	predict data.
var_model	a dataframe containing equations.
table	outtable.

Value

a list

Init_func *Initialize model Judgement variables and so on.*

Description

Initialize model Judgement variables and so on.

Usage

```
Init_func(userdata, model, guess, method, plot)
```

Arguments

userdata	a data frame containing all model variables. The "time" column must be included.
model	a string of model.
guess	a list or a string. Guess the coefficients or initial values.
method	a list or a string. The available options are 'Nelder-Mead','L-BFGS-B','SANN' and 'BFGS'.
plot	TRUE or FALSE.

Value

a list

`Plot_func` *Draw the diagram of differential equation*

Description

Draw the diagram of differential equation

Usage

```
Plot_func(
  userdata,
  predict,
  modelDF,
  var_model,
  field_model,
  order_model,
  multi_model
)
```

Arguments

`userdata`	a data frame containing all model variables. The "time" column must be included.
`predict`	predict data.
`modelDF`	a dataframe of full model.
`var_model`	a dataframe containing equations.
`field_model`	the user's data columns.
`order_model`	N-order differential equation.
`multi_model`	TRUE or FALSE

Value

plot

Scale_within *Center the data according to model*

Description

Center the data according to model

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)
```

Solver_BiFirst_func *Solver of bivariate first-order differential equation*

Description

Solver of bivariate first-order differential equation

Usage

```
Solver_BiFirst_func(userdata, var_model, guess, method)
```

Arguments

userdata	a data frame containing all model variables. The "time" column must be included.
var_model	a dataframe containing equations.
guess	a list or a string. Guess the coefficients or initial values.
method	a list or a string. The available options are 'Nelder-Mead','L-BFGS-B','SANN' and 'BFGS'.

Value

a list

Solver_MultiBiFirst_func *Solver of Multilevel bivariate first-order differential equation*

Description

Solver of Multilevel bivariate first-order differential equation

Usage

```
Solver_MultiBiFirst_func(init_list)
```

Arguments

init_list	a list of init_func

Value

a list

Solver_MultiUniSecond_func *Solver of Multilevel univariate second-order differential equation*

Description

Solver of Multilevel univariate second-order differential equation

Usage

```
Solver_MultiUniSecond_func(init_list)
```

Arguments

init_list	a list of init_func

Value

a list

Solver_UniSecond_func *Solver of univariate second-order differential equation*

Description

Solver of univariate second-order differential equation

Usage

```
Solver_UniSecond_func(userdata, var_model, guess, method)
```

Arguments

userdata	a data frame containing all model variables. The "time" column must be included.
var_model	a dataframe containing equations
guess	a list or a string. Guess the coefficients or initial values.
method	a list or a string. The available options are 'Nelder-Mead','L-BFGS-B','SANN' and 'BFGS'.

Value

a list

Index