Package 'demodelr'

May 8, 2026

Type Package

Title Simulating Differential Equations with Data

Version 2.0.1

Depends R (>= 4.1.0)

Description Designed to support the visualization, numerical computation, qualitative analysis, model-data fusion, and stochastic simulation for autonomous systems of differential equations. Euler and Runge-Kutta methods are implemented, along with tools to visualize the two-dimensional phaseplane. Likelihood surfaces and a simple Markov Chain Monte Carlo parameter estimator can be used for model-data fusion of differential equations and empirical models. The Euler-Maruyama method is provided for simulation of stochastic differential equations. The package was originally written for internal use to support teaching by Zobitz, and refined to support the text ``Exploring modeling with data and differential equations using R'' by John Zobitz (2021) <https://jmzobitz.github.io/ModelingWithR/index.html>.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Imports ggplot2, purrr, tidyr, dplyr, formula.tools, GGally, rlang, utils, tibble, tidyselect

RoxygenNote 7.3.2

Suggests knitr, rmarkdown

URL https://github.com/jmzobitz/demodelr

BugReports https://github.com/jmzobitz/demodelr/issues

NeedsCompilation no

Author John Zobitz [aut, cre] (ORCID: <https://orcid.org/0000-0002-1830-143X>)

Maintainer John Zobitz <zobitz@augsburg.edu>

Repository CRAN

Date/Publication 2026-01-16 00:50:03 UTC

Contents

`compute_likelihood` *Likelihood plot of a two parameter model*

Description

`compute_likelihood` computes the likelihood for a model

Usage

```
compute_likelihood(model, data, parameters, logLikely = FALSE)
```

Arguments

`model`	a function or model of our situation, written with formula notation
`data`	Data frame of data First column is the independent variable, second column dependent variable. Must be a data.frame
`parameters`	The data frame matrix of values of the parameters we are using. This will be made using expand.grid or equivalent
`logLikely`	Do we compute the log likelihood function (default is FALSE). NOTE: what gets returned is - logLikely - meaning that this will be a positive number to work with.

Value

A list with two entries: (1) the likelihood values and (2) values of parameters that optimize the likelihood.

Examples

```
### Contour plot of a logistic model for two parameters K and b
### using data collected from growth of yeast population

# Define the solution to the differential equation with
# parameters K and b Gause model equation
gause_model <- volume ~ K / (1 + exp(log(K / 0.45 - 1) - b * time))
# Identify the ranges of the parameters that we wish to investigate
kParam <- seq(5, 20, length.out = 100)
bParam <- seq(0, 1, length.out = 100)
# Allow for all the possible combinations of parameters
gause_parameters <- expand.grid(K = kParam, b = bParam)
# Now compute the likelihood
gause_likelihood <- compute_likelihood( model = gause_model,
                                        data = yeast,
                                        parameters = gause_parameters,
                                        logLikely = FALSE
)
```

`eigenvalues` *Matrix eigenvalues and eigenvectors*

Description

`eigenvalues` visualizes the vector field for a one or two dimensional differential equation.

Usage

```
eigenvalues(matrix_entries, matrix_rows = 2)
```

Arguments

`matrix_entries`	entries of your matrix in row wise format. So the matrix # 4 3 # 2 1 # would be entered in c(4,3,2,1)
`matrix_rows`	the number of rows and columns in your SQUARE matrix.

Value

The result is a list with two elements (denoted by the "$"), values and vectors. result$values are the eigenvalues, stored as a vector. The leading eigenvalue is the first entry in the vector.

Examples

```
eigenvalues(c(1,2,3,4))

# Note: for the 3 x 3 case, we need to define the number of matrix rows:
eigenvalues(c(1,2,3,4,5,6,7,8,9),matrix_rows=3)
```

euler *Euler's method solution for a differential equation.*

Description

euler solves a multi-dimensional differential equation with Euler's method. The parameters listed as required are needed See the vignette for detailed examples of usage.

Usage

```
euler(
  system_eq,
  initial_condition,
  parameters = NULL,
  t_start = 0,
  deltaT = 1,
  n_steps = 1
)
```

Arguments

system_eq	(REQUIRED) The 1 or multi dimensional system of equations, written in formula notation as a vector (i.e. c(dx ~ f(x,y), dy ~ g(x,y)))
initial_condition	(REQUIRED) Listing of initial conditions, as a vector
parameters	The values of the parameters we are using (optional)
t_start	The starting time point (defaults to t = 0)
deltaT	The timestep length (defaults to 1)
n_steps	The number of timesteps to compute solution (defaults to n_steps = 1)

Value

A tidy of data frame for the calculated solutions and the time

See Also

rk4

Examples

```
# Define the rate equation:
lynx_hare_eq <- c(
 dHdt ~ r * H - b * H * L,
 dLdt ~ e * b * H * L - d * L
)

# Define the parameters (as a named vector):
```

```
lynx_hare_params <- c(r = 2, b = 0.5, e = 0.1, d = 1)

# Define the initial condition (as a named vector):
lynx_hare_init <- c(H = 1, L = 3)

# Define deltaT and the number of time steps:
deltaT <- 0.05
n_steps <- 200

# Compute the solution via Euler's method:
out_solution <- euler(system_eq = lynx_hare_eq,
                      parameters = lynx_hare_params,
                      initial_condition = lynx_hare_init,
                      deltaT = deltaT,
                      n_steps = n_steps
)
```

`euler_stochastic` *Euler-Maruyama method solution for a stochastic differential equation.*

Description

`euler_stochastic` solves a multi-dimensional differential equation with the Euler-Maruyama method with stochastic elements.

Usage

```
euler_stochastic(
  deterministic_rate,
  stochastic_rate,
  initial_condition,
  parameters = NULL,
  t_start = 0,
  deltaT = 1,
  n_steps = 1,
  D = 1
)
```

Arguments

`deterministic_rate`
: The 1 or multi dimensional system of equations for the deterministic part of the differential equation, written in formula notation as a vector (i.e. c(dx ~ f(x,y), dy ~ g(x,y)))

`stochastic_rate`
: The 1 or multi dimensional system of equations for the stochastic part of the differential equation, written in formula notation as a vector (i.e. c(dx ~ f(x,y), dy ~ g(x,y)))

initial_condition	(REQUIRED) Listing of initial conditions, as a vector
parameters	The values of the parameters we are using
t_start	The starting time point (defaults to t = 0)
deltaT	The timestep length (defaults to 1)
n_steps	The number of timesteps to compute solution (defaults to n_steps = 1)
D	diffusion coefficient for the stochastic part of the SDE

Value

A tidy of data frame the solutions

Examples

```
### Simulate the stochastic differential equation dx = r*x*(1-x/K) dt + dW(t)
# Identify the deterministic and stochastic parts of the DE:
deterministic_logistic <- c(dx ~ r*x*(1-x/K))
stochastic_logistic <- c(dx ~ 1)

# Identify the initial condition and any parameters
init_logistic <- c(x=3)

logistic_parameters <- c(r=0.8, K=100) # parameters: a named vector

# Identify how long we run the simulation
deltaT_logistic <- .05 # timestep length
timesteps_logistic <- 200 # must be a number greater than 1

# Identify the standard deviation of the stochastic noise
D_logistic <- 1

# Do one simulation of this differential equation
logistic_out <- euler_stochastic(
deterministic_rate = deterministic_logistic,
stochastic_rate = stochastic_logistic,
initial_condition = init_logistic,
parameters = logistic_parameters,
deltaT = deltaT_logistic,
n_steps = timesteps_logistic, D = D_logistic
)
### Simulate a stochastic process for the tourism model presented in
### Sinay, Laura, and Leon Sinay. 2006. “A Simple Mathematical
### Model for the Effects of the Growth of Tourism on Environment.”
### In International Tourism Conference. Alanya, Turkey.
### where we have the following SDE:
### dr = r*(1-r)-a*v dt, dv = b*v*(r-v) dt + v*(r-v) dW(t)

# Identify the deterministic and stochastic parts of the DE:
deterministic_tourism<- c(dr ~ r*(1-r)-a*v, dv ~ b*v*(r-v))
stochastic_tourism <-  c(dr ~ 0, dv ~ v*(r-v))
```

```
# Identify the initial condition and any parameters
init_tourism <- c(r = 0.995, v = 0.00167)
tourism_parameters <- c(a = 0.15, b = 0.3316)   #

deltaT_tourism <- .5 # timestep length
timeSteps_tourism <- 200 # must be a number greater than 1

# Identify the diffusion coefficient
D_tourism <- .05

# Do one simulation of this differential equation
tourism_out <- euler_stochastic(
  deterministic_rate = deterministic_tourism,
 stochastic_rate = stochastic_tourism,
 initial_condition = init_tourism,
 parameters = tourism_parameters,
 deltaT = deltaT_tourism,
 n_steps = timeSteps_tourism,
 D = D_tourism
)
```

`global_temperature` *Measured average global temperature anomaly by year*

Description

A dataset containing average global temperature anomaly for each year since 1880. The variables are as follows:

Usage

```
data(global_temperature)
```

Format

A data frame with 142 rows and 2 variables

Details

- year_since_1880. The year since 1880 (year)
- temperature_anomaly. Average global temperature anomaly (degrees Celsius, relative to 1951-1980)

Source

The data were collected from NOAA. https://science.nasa.gov/earth/explore/earth-indicators/global-temperature/, download 2022-06-08

mcmc_analyze *Markov Chain parameter estimates*

Description

mcmc_analyze Computes summary histograms and model-data comparisons from and Markov Chain Monte Carlo parameter estimate for a given model

Usage

```
mcmc_analyze(
  model,
  data,
  mcmc_out,
  mode = c("emp", "de"),
  initial_condition = NULL,
  deltaT = NULL,
  n_steps = NULL,
  verbose = TRUE
)
```

Arguments

model	the model equations that we use to compute the result.
data	the data used to assess the model
mcmc_out	A dataframe: the first column is the accept flag of the mcmc run (TRUE/FALSE), the log likelihood, and the parameter values
mode	two choices: emp –> empirical (default) or de –> differential equations. The estimator works differently depending on which is used.
initial_condition	The initial condition for the differential equation (DE mode only)
deltaT	The length between timesteps (DE mode only)
n_steps	The number of time steps we run the model (DE mode only)
verbose	TRUE / FALSE indicate if parameter estimates should be printed to console (option, defaults to TRUE)

Value

Two plots: (1) fitted model results compared to data, and (2) pairwise parameter histograms and scatterplots to test model equifinality.

See Also

mcmc_estimate

Examples

```
## Example with an empirical model:
## Step 1: Define the model and parameters
phos_model <- daphnia ~ c * algae^(1 / theta)

phos_param <- tibble::tibble( name = c("c", "theta"),
lower_bound = c(0, 1),
upper_bound = c(2, 20))

## Step 2: Determine MCMC settings
# Define the number of iterations
phos_iter <- 1000

## Step 3: Compute MCMC estimate
phos_mcmc <- mcmc_estimate(model = phos_model,
data = phosphorous,
parameters = phos_param,
iterations = phos_iter)

## Step 4: Analyze results:
mcmc_analyze(model = phos_model,
data = phosphorous,
mcmc_out = phos_mcmc)

## Example with a differential equation:
## Step 1: Define the model, parameters, and data
## Define the tourism model
tourism_model <- c(dRdt ~ resources * (1 - resources) - a * visitors,
dVdt ~ b * visitors * (resources - visitors))

# Define the parameters that you will use with their bounds
tourism_param <- tibble::tibble( name = c("a", "b"),
lower_bound = c(10, 0),
upper_bound = c(30, 5))

## Step 2: Determine MCMC settings
# Define the initial conditions
tourism_init <- c(resources = 0.995, visitors = 0.00167)
deltaT <- .1 # timestep length
n_steps <- 15 # must be a number greater than 1
# Define the number of iterations
tourism_iter <- 1000

## Step 3: Compute MCMC estimate
tourism_out <- mcmc_estimate(
 model = tourism_model,
 data = parks,
 parameters = tourism_param,
 mode = "de",
 initial_condition = tourism_init, deltaT = deltaT,
 n_steps = n_steps,
 iterations = tourism_iter)
```

```
## Step 4: Analyze results
mcmc_analyze(
 model = tourism_model,
 data = parks,
 mcmc_out = tourism_out,
 mode = "de",
 initial_condition = tourism_init, deltaT = deltaT,
 n_steps = n_steps
)
```

`mcmc_estimate` *Markov Chain parameter estimates*

Description

`mcmc_estimate` Computes and Markov Chain Monte Carlo parameter estimate for a given model

Usage

```
mcmc_estimate(
  model,
  data,
  parameters,
  iterations = 1,
  knob_flag = FALSE,
  mode = c("emp", "de"),
  initial_condition = NULL,
  deltaT = NULL,
  n_steps = NULL
)
```

Arguments

`model`	the model equations that we use to compute the result.
`data`	the data used to assess the model
`parameters`	a data frame that lists the names of the parameters along with upper and lower bounds
`iterations`	the number of iterations we wish to run the MCMC for.
`knob_flag`	determines if we tune the range that can be search (annealing)
`mode`	two choices: emp –> empirical (default) or de –> differential equations. The estimator works differently depending on which is used.
`initial_condition`	The initial condition for the differential equation (DE mode only)
`deltaT`	The length between timesteps (DE mode only)
`n_steps`	The number of time steps we run the model (DE mode only)

Value

A dataframe: the first column is the accept flag of the mcmc run (TRUE/FALSE), the log likelihood, and the parameter values

See Also

mcmc_analyze

Examples

```
## Example with an empirical model:
## Step 1: Define the model and parameters
phos_model <- daphnia ~ c * algae^(1 / theta)

phos_param <- tibble::tibble( name = c("c", "theta"),
lower_bound = c(0, 1),
upper_bound = c(2, 20))

## Step 2: Determine MCMC settings
# Define the number of iterations
phos_iter <- 1000

## Step 3: Compute MCMC estimate
phos_mcmc <- mcmc_estimate(model = phos_model,
data = phosphorous,
parameters = phos_param,
iterations = phos_iter)

## Example with a differential equation:
## Step 1: Define the model, parameters, and data
## Define the tourism model
tourism_model <- c(dRdt ~ resources * (1 - resources) - a * visitors,
dVdt ~ b * visitors * (resources - visitors))

# Define the parameters that you will use with their bounds
tourism_param <- tibble::tibble( name = c("a", "b"),
lower_bound = c(10, 0),
upper_bound = c(30, 5))

## Step 2: Determine MCMC settings
# Define the initial conditions
tourism_init <- c(resources = 0.995, visitors = 0.00167)
deltaT <- .1 # timestep length
n_steps <- 15 # must be a number greater than 1
# Define the number of iterations
tourism_iter <- 1000

## Step 3: Compute MCMC estimate
tourism_out <- mcmc_estimate(
 model = tourism_model,
 data = parks,
```

```
 parameters = tourism_param,
mode = "de",
 initial_condition = tourism_init, deltaT = deltaT,
n_steps = n_steps,
 iterations = tourism_iter)
```

`parks` *Visitor and resource usage to a national park*

Description

A dataset containing scaled visitor usage to a national park. The variables are as follows:

Usage

```
data(parks)
```

Format

A data frame with 8 rows and 3 variables

Details

- time. (days)
- visitors. number of visitors to a national parked, scaled by the equilibrium value.
- resources. scaled area of the reserve not deforested.

Source

Sinay, Laura, and Leon Sinay. 2006. "A Simple Mathematical Model for the Effects of the Growth of Tourism on Environment." In International Tourism Conference. Alanya, Turkey.

`phaseplane` *Phase plane of differential equation.*

Description

`phaseplane` visualizes the vector field for a one or two dimensional differential equation.

Usage

```
phaseplane(
  system_eq,
  x_var,
  y_var,
  parameters = NULL,
  x_window = c(-4, 4),
  y_window = c(-4, 4),
  plot_points = 10,
  eq_soln = FALSE,
  precision = 2
)
```

Arguments

system_eq	(required) The 1 or 2 dimensional system of equations, written in formula notation as a vector (i.e. c(dx ~ f(x,y), dy ~ g(x,y)))
x_var	(required) x axis variable (used to create the plot and label axes)
y_var	(required) y axis variable (used to create the plot and label axes)
parameters	(optional) any parameters in the system of equations
x_window	(optional) x axis limits. Must be of the form c(minVal,maxVal). Defaults to -4 to 4.
y_window	(optional) y axis limits. Must be of the form c(minVal,maxVal). Defaults to -4 to 4.
plot_points	(optional) number of points we evaluate on the grid in both directions. Defaults to 10.
eq_soln	(optional) TRUE / FALSE - lets you know if you want the code to estimate if there are any equilibrium solutions in the provided window. This will print out the equilibrium solutions to the console.
precision	(optional) number of digits to report the equilibrium solution. Increasing the number of digits may report a larger number of possible equilibrium solutions to test.

Value

A phase plane diagram of system of differential equations

Examples

```
# For a two variable system of differential equations we use the
# formula notation for dx/dt and the dy/dt separately:
system_eq <- c(dx ~ cos(y),
               dy ~ sin(x))
phaseplane(system_eq,x_var='x',y_var='y')

# For a one dimensional system: dy/dt = f(t,y).  In this case the
# xWindow represents time.
```

```
# However, the code is structured a little differently.
# Consider dy/dt = -y*(1-y):

system_eq <- c(dt ~ 1,
               dy ~ -y*(1-y))

 phaseplane(system_eq,x_var="t",y_var="y")

# Here is an example to find equilibrium solutions.

 system_eq <- c(dx ~ y+x,
               dy ~ x-y)

 phaseplane(system_eq,x_var='x',y_var='y',eq_soln=TRUE)

# We would expect an equilibrium at the origin,
# but no equilibrium solution was found, but if we narrow the search range:

 phaseplane(system_eq,x_var='x',y_var='y',x_window = c(-0.1,0.1),y_window=c(-0.1,0.1),eq_soln=TRUE)

# Confirm any equilibrium solutions through direct evaluation of the differential equation.
```

---

phosphorous *Measured phosphorous of Daphnia and algae*

Description

A dataset containing phosphorous content in Daphnia and algae. The variables are as follows:

Usage

```
data(phosphorous)
```

Format

A data frame with 6 rows and 2 variables

Details

- algae. Phosphorous content in algal food (%)
- daphnia. Phosphorous content in *Daphnia* (%)

Source

The data were digitized from Sterner and Elser *Ecological Stoichiometry*, page 22, Figure 1.9A. The original study was DeMott et. al (1998) *Limnol. Oceanogr.* 44:1557.

precipitation *Measured precipitation from a rainfall event*

Description

A dataset containing measured precipitation data from the Minneapolis St. Paul Area:

Usage

```
data(precipitation)
```

Format

A data frame with 56 rows and 5 variables

Details

- date. Calendar day of year of measurement
- time. Time measurement is made
- station_id Shorthand name for station in CoCoRaHS network
- station_name Name of station in CoCoRaHS network
- precip. Observed precipitation (inches)

Source

The data were collected from Community Collaborative Rain Hail and Snow Network (CoCoRaHS). https://www.cocorahs.org/ViewData/ListDailyPrecipReports.aspx

rk4 *Runge Kutta method solution for a differential equation.*

Description

rk4 solves a multi-dimensional differential equation with Runge-Kutta 4th order method. The parameters listed as required are needed See the vignette for detailed examples of usage.

Usage

```
rk4(
  system_eq,
  initial_condition,
  parameters = NULL,
  t_start = 0,
  deltaT = 1,
  n_steps = 1
)
```

Arguments

system_eq	(REQUIRED) The 1 or 2 dimensional system of equations, written in formula notation as a vector (i.e. c(dx ~ f(x,y), dy ~ g(x,y)))
initial_condition	(REQUIRED) Listing of initial conditions, as a vector
parameters	The values of the parameters we are using (optional)
t_start	The starting time point (defaults to t = 0)
deltaT	The timestep length (defaults to 1)
n_steps	The number of timesteps to compute solution (defaults to n_steps = 1)

Value

A tidy of data frame for the calculated solutions and the time

See Also

See Runge Kutta methods for more explanation of Runge-Kutta methods, as well as the code euler

Examples

```
# Define the rate equation:
quarantine_eq <- c(
 dSdt ~ -k * S * I,
 dIdt ~ k * S * I - beta * I
)
# Define the parameters (as a named vector):
quarantine_parameters <- c(k = .05, beta = .2)
# Define the initial condition (as a named vector):
quarantine_init <- c(S = 300, I = 1)
# Define deltaT and the number of time steps:
deltaT <- .1 # timestep length
n_steps <- 10 # must be a number greater than 1
# Compute the solution via Euler's method:
out_solution <- rk4(system_eq = quarantine_eq,
                    parameters = quarantine_parameters,
                    initial_condition = quarantine_init, deltaT = deltaT,
                    n_steps = n_steps
)
```

---

snowfall *Measured snowfall from a blizzard in April 2018*

Description

A dataset containing measured snowfall data from the Minneapolis St. Paul Area:

Usage

```
data(snowfall)
```

Format

A data frame with 16 rows and 5 variables

Details

- date. Calendar day of year of measurement
- time. Time measurement is made
- station_id Shorthand name for station in CoCoRaHS network
- station_name Name of station in CoCoRaHS network
- snowfall total snowfall (inches)

Source

The data were collected from Community Collaborative Rain Hail and Snow Network (CoCoRaHS). https://www.cocorahs.org/ViewData/ListDailyPrecipReports.aspx

---

wilson *Measured Weight of a dog over time*

Description

A dataset containing the mass of a growing dog.

Usage

```
data(wilson)
```

Format

A data frame with 19 rows and 2 variables

Details

- days since birth
- weight. (pounds)

Source

From https://bscheng.com/2014/05/07/modeling-logistic-growth-data-in-r/

`yeast` *Measured Sacchromyces data (yeast) from Gause 1932 "Experimental studies on the struggle for coexistence"*

Description

A dataset containing measurements of growth of yeast in a culture. The variables are as follows:

Usage

```
data(yeast)
```

Format

A data frame with 7 rows and 2 variables

Details

- time. (hours)
- volume. Sacchromyces volume in container (cubic centimeters)

Source

Table 1 from Gause, G. F. 1932. "Experimental Studies on the Struggle for Existence: I. Mixed Population of Two Species of Yeast." Journal of Experimental Biology 9 (4): 389–402.

Index