

Package ‘denim’

July 22, 2025

Type Package

Title Generate and Simulate Deterministic Discrete-Time Compartmental Models

Version 1.2.1

Date 2025-06-02

Description R package to build and simulate deterministic discrete-time compartmental models that can be non-Markov. Length of stay in each compartment can be defined to follow a parametric distribution (`d_exponential()`, `d_gamma()`, `d_weibull()`, `d_lognormal()`) or a non-parametric distribution (`nonparametric()`). Other supported types of transition from one compartment to another includes fixed transition (`constant()`), multinomial (`multinomial()`), fixed transition probability (`transprob()`).

License MIT + file LICENSE

URL <https://drthinhong.com/denim/>, <https://github.com/thinhong/denim>

BugReports <https://github.com/thinhong/denim/issues>

Depends R (>= 4.1.0)

Imports Rcpp (>= 1.0.6), colorspace, rlang, glue

Suggests covr, knitr, rmarkdown, testthat (>= 3.0.0), waldo, xml2, deSolve, tidyverse, DiagrammeR

LinkingTo Rcpp, testthat

Encoding UTF-8

RoxygenNote 7.3.2

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation yes

Author Thinh Ong [aut, cph] (ORCID: <<https://orcid.org/0000-0001-6772-9291>>),
Anh Phan [aut, cre] (ORCID: <<https://orcid.org/0009-0000-2129-435X>>),
Marc Choisy [aut] (ORCID: <<https://orcid.org/0000-0002-5187-6390>>),
Niels Lohman [ctb],
Bjoern Hoehrmann [ctb],
Florian Loitsch [ctb],
Ingo Berg [ctb]

Maintainer Anh Phan <anhptq@oucru.org>
Repository CRAN
Date/Publication 2025-06-02 13:30:07 UTC

Contents

denim-package	2
denim_dsl	3
d_exponential	4
d_gamma	4
d_lognormal	5
d_weibull	6
mathexpr	6
nonparametric	7
plot.denim	8
sim	8
Index	10

denim-package	<i>denim</i>
---------------	--------------

Description

Simulate deterministic discrete time model

Details

Imports

Author(s)

- Maintainer:** Anh Phan <anhptq@oucru.org> (ORCID)
- Authors:
- Thinh Ong <thinhop@oucru.org> (ORCID) [copyright holder]
 - Marc Choisy <mchoisy@oucru.org> (ORCID)
- Other contributors:
- Niels Lohman [contributor]
 - Bjoern Hoehrmann <bjoern@hoehrmann.de> [contributor]
 - Florian Loitsch [contributor]
 - Ingo Berg [contributor]

See Also

Useful links:

- <https://drthinhong.com/denim/>
- <https://github.com/thinhong/denim>
- Report bugs at <https://github.com/thinhong/denim/issues>

denim_dsl

Define transitions using denim's domain-specific language (DSL)

Description

This function parses model transitions defined in denim's DSL syntax

Usage

```
denim_dsl(x)
```

Arguments

- | | |
|---|--|
| x | • an expression written in denim's DSL syntax. Each line should be a transition written in the format <code>compartment -> out_compartment = expression</code> where expression can be either a math expression or one of denim's built-in dwell time distribution function |
|---|--|

Value

denim_transition object

Examples

```
transitions <- denim_dsl({  
  S -> I = beta * (I/N) * S * timeStep  
  I -> R = d_gamma(rate = 1/4, shape = 3)  
})
```

d_exponential	<i>Discrete exponential distribution</i>
---------------	--

Description

Discrete exponential distribution

Usage

```
d_exponential(rate, dist_init = FALSE)
```

Arguments

rate	rate parameter of an exponential distribution
dist_init	whether to distribute initial value across subcompartments following this distribution. (default to FALSE, meaning init value is always in the first compartment)

Value

a Distribution object for simulator

Examples

```
transitions <- list("I -> D" = d_exponential(0.3))
transitions <- denim_dsl({I -> D = d_exponential(0.3)})
```

d_gamma	<i>Discrete gamma distribution</i>
---------	------------------------------------

Description

Discrete gamma distribution

Usage

```
d_gamma(rate, shape, dist_init = FALSE)
```

Arguments

rate	rate parameter of a gamma distribution
shape	shape parameter of a gamma distribution
dist_init	whether to distribute initial value across subcompartments following this distribution.

Value

a Distribution object for simulator

Examples

```
transitions <- list("S -> I" = d_gamma(rate = 1, shape = 5))
transitions_dsl <- denim_dsl({S -> I = d_gamma(rate = 1, shape = 5)})
# define model parameters as distributional parameters
transitions_dsl <- denim_dsl({S -> I = d_gamma(rate = i_rate, shape = i_shape)})
```

d_lognormal	<i>Discrete log-normal distribution</i>
-------------	---

Description

Discrete log-normal distribution

Usage

```
d_lognormal(mu, sigma, dist_init = FALSE)
```

Arguments

mu	location parameter or the ln mean
sigma	scale parameter or ln standard deviation
dist_init	whether to distribute initial value across subcompartments following this distribution. (default to FALSE, meaning init value is always in the first compartment)

Value

a Distribution object for simulator

Examples

```
transitions <- list("I -> D" = d_lognormal(3, 0.6))
transitions <- denim_dsl({I -> D = d_lognormal(3, 0.6)})
```

d_weibull	<i>Discrete Weibull distribution</i>
-----------	--------------------------------------

Description

Discrete Weibull distribution

Usage

```
d_weibull(scale, shape, dist_init = FALSE)
```

Arguments

scale	scale parameter of a Weibull distribution
shape	shape parameter of a Weibull distribution
dist_init	whether to distribute initial value across subcompartments following this distribution. (default to FALSE, meaning init value is always in the first compartment)

Value

a Distribution object for simulator

Examples

```
transitions <- list("I -> D" = d_weibull(0.6, 2))
transitions <- denim_dsl({ I -> D = d_weibull(0.6, 2) })
```

mathexpr	<i>Mathematical expression</i>
----------	--------------------------------

Description

Mathematical expression

Usage

```
mathexpr(expr)
```

Arguments

expr	User defined mathematical expression. The expression will be processed by muparser library which offers a wide variety of operators. Visit muparser website (https://beltoforion.de/en/muparser/features.php) to see full list of available operators.
------	--

Value

a Distribution object for simulator

Examples

```
transitions <- list("S->I"="beta*S/N")
transitions <- denim_dsl({S->I=beta*S/N})
# definition for parameters in the expression required
params <- c(N = 1000, beta = 0.3)
```

nonparametric	<i>Nonparametric distribution</i>
---------------	-----------------------------------

Description

Convert a vector of frequencies, percentages... into a distribution

Usage

```
nonparametric(x, dist_init = FALSE)
```

Arguments

x	a vector of values
dist_init	whether to distribute initial value across subcompartments following this distribution. (default to FALSE, meaning init value is always in the first compartment)

Value

a Distribution object for simulator

Examples

```
transitions <- list("S->I"=nonparametric( c(0.1, 0.2, 0.5, 0.2) ))
transitions <- denim_dsl({S->I=nonparametric( c(0.1, 0.2, 0.5, 0.2) )})
# you can also define a model parameter for the distribution
transitions <- denim_dsl({S->I=nonparametric( dwelltime_dist )})
```

plot.denim	<i>Overloaded plot function for denim object</i>
------------	--

Description

Overloaded plot function for denim object

Usage

```
## S3 method for class 'denim'
plot(x, ..., color_palette = NULL)
```

Arguments

x	• output of denim::sim function
...	• additional parameter for plot() function
color_palette	• a palette name from the colorspace package. You can view available palettes with colorspace::hcl_palettes("qualitative", plot = TRUE).

sim	<i>Simulator for deterministic discrete time model with memory</i>
-----	--

Description

Simulation function that call the C++ simulator

Usage

```
sim(
  transitions,
  initialValues,
  parameters = NULL,
  simulationDuration,
  timeStep = 1,
  errorTolerance = 0.001
)
```

Arguments

transitions	a list of transitions follows this format "transition" = distribution()
initialValues	a vector contains the initial values of all compartments defined in the transitions , follows this format compartment_name = initial_value
parameters	a vector contains values of any parameters that are not compartments, usually parameters used in mathexp() functions

simulationDuration duration of time to be simulate

timeStep set the output time interval. For example, if `simulationDuration = 10` means 10 days and `timeStep = 0.1`, the output will display results for each 0.1 daily interval

errorTolerance set the threshold so that a cumulative distribution function can be rounded to 1. For example, if we want a cumulative probability of 0.999 to be rounded as 1, we set `errorTolerance = 0.001` ($1 - 0.999 = 0.001$). Default is 0.001

Value

a data.frame with class `denim` that can be plotted with a `plot()` method

Examples

```
transitions <- list(
  "S -> I" = "beta * S * I / N",
  "I -> R" = d_gamma(1/3, 2)
)

initialValues <- c(
  S = 999,
  I = 1,
  R = 0
)

parameters <- c(
  beta = 0.012,
  N = 1000
)

simulationDuration <- 30
timeStep <- 0.01

mod <- sim(transitions = transitions,
           initialValues = initialValues,
           parameters = parameters,
           simulationDuration = simulationDuration,
           timeStep = timeStep)
```

Index

d_exponential, [4](#)
d_gamma, [4](#)
d_lognormal, [5](#)
d_weibull, [6](#)
denim (denim-package), [2](#)
denim-package, [2](#)
denim_dsl, [3](#)

mathexpr, [6](#)

nonparametric, [7](#)

plot.denim, [8](#)

sim, [8](#)