Package 'golden'

March 16, 2026

Type Package

Title Framework for Patient-Level Microsimulation of Risk Factor Trajectories & Hazard-Based Events

Version 0.0.1

Date 2026-03-04

Description Fast, flexible, patient-level microsimulation. Time-stepped simulation with a 'C++' backend from user-supplied initial population, trajectories, hazards, and corresponding event transitions. User-defined aggregate time series histories are returned together with the final population. Designed for simulation of chronic diseases with continuous and evolving risk factors, but could easily be applied more generally.

License MIT + file LICENSE

Imports Rcpp (>= 1.1.0), data.table

LinkingTo Rcpp

Suggests testthat (>= 3.0.0), SciViews, knitr, rmarkdown, ggplot2

Config/testthat/edition 3

Depends R (>= 3.5)

LazyData true

RoxygenNote 7.3.3

Encoding UTF-8

VignetteBuilder knitr

NeedsCompilation yes

Author Pete Dodd [aut, cre] (ORCID: <https://orcid.org/0000-0001-5825-9347>),
Robert Chisholm [aut] (ORCID: <https://orcid.org/0000-0002-6983-2759>),
University of Sheffield [cph],
Horizon Europe [fnd]

Maintainer Pete Dodd <p.j.dodd@sheffield.ac.uk>

Repository CRAN

Date/Publication 2026-03-16 19:50:02 UTC

Contents

`bmi_fits` *Example BMI distribution*

Description

details TODO

Usage

```
bmi_fits
```

Format

TODO

foo bar

Source

TODO

---

check_column — *Validate an history column object If validation fails, an exception will be raised.*

Description

Validate an history column object If validation fails, an exception will be raised.

Usage

```
check_column(column, initPop = NULL)
```

Arguments

column	An S3 object of class "golden_history_column"
initPop	(Optional) data.table to check columns required by functions exist

Value

No return value, called for side effects.

Examples

```
library(data.table)
dt <- data.table(a = rep(0, 100))
# Create an S3 golden_history_column
col <- new_column("sum_a", sum, c("a"))
# check_column() will not throw an exception
# as col is a valid S3 golden_history_column
# and dt contains column "a"
check_column(col, dt)
```

---

check_hazard — *Validate an hazard object If validation fails, an exception will be raised.*

Description

Validate an hazard object If validation fails, an exception will be raised.

Usage

```
check_hazard(hazard, initPop = NULL)
```

Arguments

`hazard`	An S3 object of class "golden_hazard"
`initPop`	(Optional) data.table to check columns required by functions exist

Value

No return value, called for side effects.

Examples

```
library(data.table)
N <- 100
dt <- data.table(a = runif(N, 0, 1), b = rep(0, N))
# Define a hazard function, which returns a vector of equal length uncertainties
test_hazard <- function(a) {
    ret <- (a < 0.5)
}
# Define a transition function, which sets all "b" columns affected by the hazard to 100
test_transition <- function() {
    return (100)
}
# Create an S3 golden_hazard
haz <- new_hazard(
              test_hazard,
              c("a"),
              new_transition(test_transition, c(), "b")
            )
# check_hazard() will not throw an exception
# as haz is a valid S3 golden_hazard
# and dt contains column "a"
check_hazard(haz, dt)
```

`check_history` *Validate an history object If validation fails, an exception will be raised.*

Description

Validate an history object If validation fails, an exception will be raised.

Usage

```
check_history(history, initPop = NULL)
```

Arguments

`history`	An S3 object of class "golden_history"
`initPop`	(Optional) data.table to check columns required by functions exist

Value

No return value, called for side effects.

Examples

```
library(data.table)
dt <- data.table(a = rep(0, 100))
# Create an S3 golden_history, containing 1 golden_history_column
hist <- new_history(new_column("sum_a", sum, c("a")))
# check_history() will not throw an exception
# as hist is a valid S3 golden_history
# and dt contains column "a" used by the column
check_history(hist, dt)
```

`check_parameters` *Validate the configuration passed to run_simulation() If validation fails, an exception will be raised.*

Description

Validate the configuration passed to run_simulation() If validation fails, an exception will be raised.

Usage

```
check_parameters(parameters, initPop = NULL)
```

Arguments

`parameters`	An golden_parameters S3 object to be validated
`initPop`	data.frame which contains the columns required by parameters

Value

No return value, called for side effects.

Examples

```
library(data.table)
N <- 100
dt <- data.table(a = runif(N, 0, 1), b = rep(0, N))
# Define a hazard function, which returns a vector of equal length uncertainties
test_hazard <- function(a) {
    ret <- (a < 0.5)
}
# Define a transition function, which sets all "b" columns affected by the hazard to 100
test_transition <- function() {
    return (100)
}
# Create an S3 golden_hazard
```

```
haz <- new_hazard(
              test_hazard,
              c("a"),
              new_transition(test_transition, c(), "b")
            )
# Define a trajectory function, which adds 2 to all members of the input vector
test_trajectory <- function(a) {
    return (a + 2)
}
# Create an S3 golden_trajectory
trj <- new_trajectory(test_trajectory, c("b"), "b")
# Create an S3 golden_history, containing 1 golden_history_column
hist <- new_history(new_column("sum_a", sum, c("a")))
# Create an S3 golden_parameters
params <- new_parameters(
  hazards = haz,
  trajectories = trj,
  steps = 10,
  debug = FALSE,
  history = hist
)
# check_parameters() will not throw an exception
# as params is a valid S3 golden_parameters
# and dt contains columns "a" and "b"
check_parameters(params, dt)
```

check_trajectory — *Validate an trajectory object If validation fails, an exception will be raised.*

Description

Validate an trajectory object If validation fails, an exception will be raised.

Usage

```
check_trajectory(trajectory, initPop = NULL)
```

Arguments

`trajectory`	An S3 object of class "golden_trajectory"
`initPop`	(Optional) data.table to check columns required by functions exist

Value

No return value, called for side effects.

Examples

```
library(data.table)
dt <- data.table(b = rep(0, 100))
# Define a trajectory function, which adds 2 to all members of the input vector
test_trajectory <- function(a) {
    return (a + 2)
}
# Create an S3 golden_trajectory
trj <- new_trajectory(test_trajectory, c("b"), "b")
# check_trajectory() will not throw an exception
# as trj is a valid S3 golden_trajectory
# and dt contains column "b"
check_trajectory(trj, dt)
```

`check_transition` *Validate an transition object If validation fails, an exception will be raised.*

Description

Validate an transition object If validation fails, an exception will be raised.

Usage

```
check_transition(transition, initPop = NULL)
```

Arguments

`transition`	An S3 object of class "golden_transition"
`initPop`	(Optional) data.table to check columns required by functions exist

Value

No return value, called for side effects.

Examples

```
library(data.table)
dt <- data.table(b = rep(0, 100))
# Define a transition function, which sets all columns affected by the hazard to 100
test_transition <- function() {
    return (100)
}
# Define an S3 golden_transition
trn <- new_transition(test_transition, c(), "b")
# check_transition() will not throw an exception
# as trn is a valid S3 golden_transition
# and dt contains column "b"
check_transition(trn, dt)
```

`create_cohort` *Create a new cohort*

Description

Temporary testing method, probably replaced in future with R's simdata package or similar

Usage

```
create_cohort(demog, N)
```

Arguments

`demog`	Demographic information containing columns AgeGrp/PopMale/PopFemale/PopTotal
`N`	Size of the population to generate

Value

A sample population data.table, with columns male/age/bmi/death

Examples

```
library(data.table)
demog <- data.table(
  AgeGrp = c(0, 1, 2, 3),
  PopMale = c(1000, 1100, 1050, 980),
  PopFemale = c(950, 1020, 1005, 970)
)
demog[, PopTotal := PopMale + PopFemale]
cohort <- create_cohort(demog, 100)
```

`globorisk_coefs` *Example globorisk coefficients*

Description

details TODO

Usage

```
globorisk_coefs
```

Format

TODO

foo bar

Source

TODO

`globorisk_cvdr` *Example globorisk baseline hazard*

Description

details TODO

Usage

```
globorisk_cvdr
```

Format

TODO

foo bar

Source

TODO

`globorisk_rf` *Example globorisk reference values*

Description

details TODO

Usage

```
globorisk_rf
```

Format

TODO

foo bar

Source

TODO

`golden` *Golden: Framework for Patient-Level Microsimulation of Risk Factor Trajectories & Hazard-Based Events*

Description

Fast, flexible, patient-level microsimulation. Time-stepped simulation with a 'C++' back-end from user-supplied initial population, trajectories, hazards, and corresponding event transitions. User-defined aggregate time series histories are returned together with the final population. Designed for simulation of chronic diseases with continuous and evolving risk factors, but could easily be applied more generally.

Author(s)

Maintainer: Pete Dodd `<p.j.dodd@sheffield.ac.uk>` (ORCID)

Authors:

- Robert Chisholm `<robert.chisholm@sheffield.ac.uk>` (ORCID)

Other contributors:

- University of Sheffield [copyright holder]
- Horizon Europe [funder]

Examples

```
## Not run:
# A full example can be found in the vignettes

## End(Not run)
```

`lifetable_data` *Example life table data*

Description

details TODO

Usage

```
lifetable_data
```

Format

TODO

foo bar

Source

TODO

`new_column` *Create a new golden_history_column*

Description

Create a new golden_history_column

Usage

```
new_column(name, fn, args, filter_fn = NULL, filter_args = NULL)
```

Arguments

`name`	Name of the column in the output data-table
`fn`	Reduction function, which converts the input columns to a single value
`args`	Names of columns and special variables to be passed to fn
`filter_fn`	(Optional) Filter function, which returns a bool vector denoting which rows should be reduced
`filter_args`	(Optional) Names of columns and special variables to be passed to filter_fn. Required if filter_fn is

Value

An object of class "golden_history_column"

Examples

```
# Create an S3 golden_history_column named "sum_a", using sum(). with column "a"
col <- new_column("sum_a", sum, c("a"))
```

new_hazard *Create a new hazard object*

Description

Create a new hazard object

Usage

```
new_hazard(
  fn,
  args,
  transitions,
  freq = 1,
  first = 1,
  last = 2147483647,
  name = NULL
)
```

Arguments

`fn`	Function which calculates the hazard likelihood
`args`	Character vector of parameter names expected by fn
`transitions`	Transition object(s) to be applied where the hazard is successful
`freq`	(Optional) The frequency of hazard execution, hazards always execute on first step
`first`	(Optional) First step the hazard should be enabled (initial step is index 1)
`last`	(Optional) Last step the hazard should be enabled (initial step is index 1)
`name`	(Optional) Name used in error messages and similar. Defaults to an automatic name

Value

An object of class "golden_hazard"

Examples

```
# Define a hazard function, which returns a vector of equal length uncertainties
test_hazard <- function(a) {
    ret <- (a < 0.5)
}
# Define a transition function, which sets all "b" columns affected by the hazard to 100
test_transition <- function() {
    return (100)
}
# Create an S3 golden_hazard
```

```
haz <- new_hazard(
              test_hazard,
              c("a"),
              new_transition(test_transition, c(), "b")
            )
```

---

new_history *Create a new golden_history*

Description

Create a new golden_history

Usage

```
new_history(columns, frequency = 1)
```

Arguments

`columns`	golden_history_column S3 object(s)
`frequency`	The number of simulation steps per history collection.

Value

An object of class "golden_history"

Examples

```
# Create an S3 golden_history, containing 1 golden_history_column
hist <- new_history(new_column("sum_a", sum, c("a")))
```

---

new_parameters *Create a new golden_parameters*

Description

Create a new golden_parameters

Usage

```
new_parameters(
  hazards = list(),
  trajectories = list(),
  steps,
  random_seed = 0,
  debug = TRUE,
  print_timing = TRUE,
  history = NULL
)
```

Arguments

`hazards`	golden_hazard S3 object(s)
`trajectories`	golden_trajectory S3 object(s)
`steps`	Number of steps to run
`random_seed`	Seed to be used for random generation. If set 0, current R random state will be used.
`debug`	(TRUE/FALSE) flag indicating whether validation checks are enabled. These catch NaN, but reduce performance
`print_timing`	(TRUE/FALSE) flag indicating whether a per-function timing report should be printed after the simulation, this will always be suppressed for fast (<= 1 second) simulations.
`history`	golden_history S3 object representing the columns of data to be aggregated during simulation

Value

An object of class "golden_parameters"

Examples

```
# Define a hazard function, which returns a vector of equal length uncertainties
test_hazard <- function(a) {
    ret <- (a < 0.5)
}
# Define a transition function, which sets all "b" columns affected by the hazard to 100
test_transition <- function() {
    return (100)
}
# Create an S3 golden_hazard
haz <- new_hazard(
              test_hazard,
              c("a"),
              new_transition(test_transition, c(), "b")
            )
# Define a trajectory function, which adds 2 to all members of the input vector
test_trajectory <- function(a) {
    return (a + 2)
}
# Create an S3 golden_trajectory
trj <- new_trajectory(test_trajectory, c("b"), "b")
# Create an S3 golden_history, containing 1 golden_history_column
hist <- new_history(new_column("sum_a", sum, c("a")))
# Create an S3 golden_parameters
params <- new_parameters(
  hazards = haz,
  trajectories = trj,
  steps = 10,
  debug = FALSE,
  history = hist
)
```

`new_trajectory` *Create a new trajectory object*

Description

Create a new trajectory object

Usage

```
new_trajectory(fn, args, property, name = NULL)
```

Arguments

`fn`	Function defining the trajectory function
`args`	Character vector of parameter names expected by fn
`property`	Name(s) of the column(s) where the result(s) of the trajectory function is to be stored
`name`	(Optional) Name used in error messages and similar. Defaults to an automatic name

Value

An object of class "golden_trajectory"

Note

If a list if passed to property, fn must return a list of equal length

Examples

```
# Define a trajectory function, which adds 2 to all members of the input vector
test_trajectory <- function(a) {
    return (a + 2)
}
# Create an S3 golden_trajectory
trj <- new_trajectory(test_trajectory, c("b"), "b")
```

`new_transition` *Create a new transition object*

Description

Create a new transition object

Usage

```
new_transition(fn, args, state, name = NULL)
```

Arguments

`fn`	Function defining the transition functions
`args`	Character vector of parameter names expected by fn
`state`	Name(s) of the column(s) where the result of the transition function is to be stored
`name`	(Optional) Name used in error messages and similar. Defaults to an automatic name

Value

An object of class "golden_transition"

Examples

```
# Define a transition function, which sets all columns affected by the hazard to 100
test_transition <- function() {
    return (100)
}
# Define an S3 golden_transition
trn <- new_transition(test_transition, c(), "b")
```

`pop_snapshot` *Example population structure*

Description

details TODO

Usage

```
pop_snapshot
```

Format

TODO

foo bar

Source

TODO

`print.golden_hazard` *Print the contents of a golden_hazard type S3 object*

Description

Print the contents of a golden_hazard type S3 object

Usage

```
## S3 method for class 'golden_hazard'
print(x, ..., indent = 0L)
```

Arguments

`x`	The object to be printed
`...`	Not used. Included for S3 method compatibility.
`indent`	(Optional) The level the printing is indented, useful if nested within another S3 object

Value

No return value, called for side effects.

Examples

```
# Define a hazard function, which returns a vector of equal length uncertainties
test_hazard <- function(a) {
    ret <- (a < 0.5)
}
# Define a transition function, which sets all "b" columns affected by the hazard to 100
test_transition <- function() {
    return (100)
}
# Create an S3 golden_hazard
haz <- new_hazard(
              test_hazard,
              c("a"),
              new_transition(test_transition, c(), "b")
            )
print(haz)
```

print.golden_history *Print the contents of a golden_history type S3 object*

Description

Print the contents of a golden_history type S3 object

Usage

```
## S3 method for class 'golden_history'
print(x, ..., indent = 0L)
```

Arguments

x	The object to be printed
...	Not used. Included for S3 method compatibility.
indent	(Optional) The level the printing is indented, useful if nested within another S3 object

Value

No return value, called for side effects.

Examples

```
# Create an S3 golden_history
hist <- new_history(new_column("sum_a", sum, c("a")))
print(hist)
```

print.golden_history_column *Print the contents of a golden_history_column type S3 object*

Description

Print the contents of a golden_history_column type S3 object

Usage

```
## S3 method for class 'golden_history_column'
print(x, ..., indent = 0L)
```

Arguments

`x`	The object to be printed
`...`	Not used. Included for S3 method compatibility.
`indent`	(Optional) The level the printing is indented, useful if nested within another S3 object

Value

No return value, called for side effects.

Examples

```
# Create an S3 golden_history_column
col <- new_column("sum_a", sum, c("a"))
print(col)
```

print.golden_parameters

Print the contents of a golden_parameters type S3 object

Description

Print the contents of a golden_parameters type S3 object

Usage

```
## S3 method for class 'golden_parameters'
print(x, ..., indent = 0)
```

Arguments

`x`	The object to be printed
`...`	Not used. Included for S3 method compatibility.
`indent`	(Optional) The level the printing is indented, useful if nested within another S3 object

Value

No return value, called for side effects.

Examples

```
# Define a hazard function, which returns a vector of equal length uncertainties
test_hazard <- function(a) {
    ret <- (a < 0.5)
}
# Define a transition function, which sets all "b" columns affected by the hazard to 100
test_transition <- function() {
    return (100)
}
# Create an S3 golden_hazard
haz <- new_hazard(
              test_hazard,
              c("a"),
              new_transition(test_transition, c(), "b")
            )
# Define a trajectory function, which adds 2 to all members of the input vector
test_trajectory <- function(a) {
    return (a + 2)
}
# Create an S3 golden_trajectory
trj <- new_trajectory(test_trajectory, c("b"), "b")
# Create an S3 golden_history, containing 1 golden_history_column
hist <- new_history(new_column("sum_a", sum, c("a")))
# Create an S3 golden_parameters
params <- new_parameters(
  hazards = haz,
  trajectories = trj,
  steps = 10,
  debug = FALSE,
  history = hist
)
print(params)
```

---

print.golden_timing *Print the contents of a golden_timing type S3 object*

Description

Print the contents of a golden_timing type S3 object

Usage

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

Arguments

x	The object to be printed
...	Not used. Included for S3 method compatibility.

Value

No return value, called for side effects.

Examples

```
library(data.table)
N <- 100
dt <- data.table(a = runif(N, 0, 1), b = rep(0, N))
# Define a hazard function, which returns a vector of equal length uncertainties
test_hazard <- function(a) {
    ret <- (a < 0.5)
}
# Define a transition function, which sets all "b" columns affected by the hazard to 100
test_transition <- function() {
    return (100)
}
# Create an S3 golden_hazard
haz <- new_hazard(
              test_hazard,
              c("a"),
              new_transition(test_transition, c(), "b")
            )
# Define a trajectory function, which adds 2 to all members of the input vector
test_trajectory <- function(a) {
    return (a + 2)
}
# Define an S3 golden_trajectory
trj <- new_trajectory(test_trajectory, c("b"), "b")
# Create an S3 golden_history, containing 1 golden_history_column
hist <- new_history(new_column("sum_a", sum, c("a")))
# Define an S3 golden_parameters
params <- new_parameters(
  hazards = haz,
  trajectories = trj,
  steps = 10,
  debug = FALSE,
  history = hist
)
# Run the simulation to collect results
results <- run_simulation(dt, params)
print(results$timing)
```

`print.golden_trajectory` *Print the contents of a golden_trajectory type S3 object*

Description

Print the contents of a golden_trajectory type S3 object

Usage

```
## S3 method for class 'golden_trajectory'
print(x, ..., indent = 0L)
```

Arguments

`x`	The object to be printed
`...`	Not used. Included for S3 method compatibility.
`indent`	(Optional) The level the printing is indented, useful if nested within another S3 object

Value

No return value, called for side effects.

Examples

```
# Define a trajectory function, which adds 2 to all members of the input vector
test_trajectory <- function(a) {
    return (a + 2)
}
# Create an S3 golden_trajectory
trj <- new_trajectory(test_trajectory, c("b"), "b")
print(trj)
```

`print.golden_transition` *Print the contents of a golden_transition type S3 object*

Description

Print the contents of a golden_transition type S3 object

Usage

```
## S3 method for class 'golden_transition'
print(x, ..., indent = 0L)
```

Arguments

`x`	The object to be printed
`...`	Not used. Included for S3 method compatibility.
`indent`	(Optional) The level the printing is indented, useful if nested within another S3 object

Value

No return value, called for side effects.

Examples

```
# Define a transition function, which sets all columns affected by the hazard to 100
test_transition <- function() {
    return (100)
}
# Define an S3 golden_transition
trn <- new_transition(test_transition, c(), "b")
print(trn)
```

`run_simulation` *Execute a patient trajectory simulation*

Description

Execute a patient trajectory simulation

Usage

```
run_simulation(initPop, parameters)
```

Arguments

`initPop`	data.table containing initial population for simulation
`parameters`	Simulation configuration

Value

An list containing final population, history and timing data.tables

Examples

```
library(data.table)
N <- 100
dt <- data.table(a = runif(N, 0, 1), b = rep(0, N))
# Define a hazard function, which returns a vector of equal length uncertainties
test_hazard <- function(a) {
    ret <- (a < 0.5)
}
# Define a transition function, which sets all "b" columns affected by the hazard to 100
test_transition <- function() {
    return (100)
}
# Create an S3 golden_hazard
haz <- new_hazard(
              test_hazard,
              c("a"),
              new_transition(test_transition, c(), "b")
            )
# Define a trajectory function, which adds 2 to all members of the input vector
test_trajectory <- function(a) {
    return (a + 2)
}
# Define an S3 golden_trajectory
trj <- new_trajectory(test_trajectory, c("b"), "b")
# Create an S3 golden_history, containing 1 golden_history_column
hist <- new_history(new_column("sum_a", sum, c("a")))
# Define an S3 golden_parameters
params <- new_parameters(
  hazards = haz,
  trajectories = trj,
  steps = 10,
  debug = FALSE,
  history = hist
)
# Run the simulation to collect results
results <- run_simulation(dt, params)
```

Index