Package 'twig'

January 15, 2025

Version 1.0.0.0

Title For Streamlining Decision and Economic Evaluation Models using Grammar of Modeling

URL https://www.dashlab.ca/, https://hjalal.github.io/twig/, https://www.dashlab.ca/projects/decision_twig/

BugReports https://github.com/hjalal/twig/issues

Maintainer Hawre Jalal <hjalal@uottawa.ca>

Description Provides tools for building decision and cost-effectiveness analysis models. It enables users to write these models concisely, simulate outcomes—including probabilistic analyses—efficiently using optimized vectorized processes and parallel computing, and produce results. The package employs a Grammar of Modeling approach, inspired by the Grammar of Graphics, to streamline model construction. For an interactive graphical user interface, see 'DecisionTwig' at <https://www.dashlab.ca/projects/decision_twig/>. Comprehensive tutorials and vignettes are available at <https://hjalal.github.io/twig/>.

License GPL (>= 3)

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Imports ggplot2, parallel, foreach, utils, stats, reshape2, abind, doParallel

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Hawre Jalal [aut, cre] (<https://orcid.org/0000-0002-8224-6834>)

Contents

`+.twig_class` *Define a method for the + operator for `twig` objects*

Description

Adds layers to the twig object. If a states layer is added, the twig object is treated as a Markov model.

Usage

```
## S3 method for class 'twig_class'
twig_obj + layer
```

Arguments

`twig_obj`	A twig object created by the `twig` function.
`layer`	A layer to be added to the twig object.

Details

This method allows layers to be added to a twig object using the + operator.

Value

The modified twig object with the new layer added.

Examples

```
twig_obj <- twig() +
  decisions(names = c(StandardOfCare, StrategyA, StrategyB, StrategyAB))
```

`calculate_icers` *Calculate Incremental Cost-Effectiveness Ratios (ICERs)*

Description

This function calculates the Incremental Cost-Effectiveness Ratios (ICERs) for a set of strategies based on their costs and effects.

Usage

```
calculate_icers(payoffs_summary, col_names = c("cost", "utility"))
```

Arguments

`payoffs_summary`	A matrix or data frame containing the summary statistics of the payoffs. It must have columns for cost and utility.
`col_names`	A character vector specifying the names of the columns for cost and utility. Default is c("cost", "utility").

Value

A data frame with the following columns:

- `decision`: The name of the strategy.
- `cost`: The cost of the strategy.
- `utility`: The effect (utility) of the strategy.
- `inc_cost`: The incremental cost compared to the next less effective strategy.
- `inc_utility`: The incremental effect compared to the next less effective strategy.
- `ICER`: The Incremental Cost-Effectiveness Ratio.
- `status`: The dominance status of the strategy (ND = non-dominated, D = dominated, ED = extendedly dominated).

Examples

```
payoffs_summary <- matrix(c(100, 200, 0.5, 0.7), ncol = 2,
                          dimnames = list(c("Strategy A", "Strategy B"),
                                          c("cost", "utility")))
calculate_icers(payoffs_summary)
```

`decisions` *Add decisions to a twig*

Description

Add decisions to a twig

Usage

```
decisions(names)
```

Arguments

`names`	decision names, a character vector of decision names. They don't need to be included in quotes.

Value

a twig layer with decision names

Examples

```
decisions(names = c(A, B, C))
decisions(names = c("A", "B", "C"))
```

event — *Add an event layer to a twig object*

Description

This function creates an event layer that can be added to a twig object. The event layer defines the possible outcomes of an event, their probabilities, and the transitions between states.

Usage

```
event(name, options, probs, transitions)
```

Arguments

`name`	A character string representing the name of the event. It doesn't need to be quoted.
`options`	A character vector of possible outcomes for the event. They don't need to be included in quotes. One of these options must be none.
`probs`	A character vector of probability function names for each outcome. They don't need to be included in quotes. One of these can be leftover for the remaining probability.
`transitions`	A character vector of state transitions corresponding to each outcome. They don't need to be included in quotes. These could be event names or states if a states layer defined. One of these can be stay for the Markov state to remain the same.

Value

A list representing the event layer.

Examples

```
#' # Adding the event layer to a twig object
twig_obj <- twig() + event(name = event_progress,
                           options = c(yes, none),
                           probs = c(pProgress, leftover),
                           transitions = c(Severe, stay))

event_layer <- event(name = "event_progress",
                     options = c("yes", "none"),
                     probs = c(pProgress, leftover),
                     transitions = c("Severe", "stay"))
```

`payoffs` *Add payoffs to a twig object*

Description

This function creates a payoffs layer that can be added to a twig object. The payoffs layer defines the payoffs and their associated discount rates.

Usage

```
payoffs(names, discount_rates = NULL)
```

Arguments

`names`	A character vector of payoff function names. They don't need to be included in quotes.
`discount_rates`	A numeric vector of discount rates for each payoff. If NULL, a discount rate of 0 is assumed for each payoff. discount_rates must have the same length as payoff names.

Value

A list representing the payoffs layer.

Examples

```
payoffs_layer <- payoffs(names = c(cost, effectiveness), discount_rates = c(0.03, 0.03))
```

`plot_ceac` *Plot Cost-Effectiveness Acceptability Curve (CEAC)*

Description

This function plots the Cost-Effectiveness Acceptability Curve (CEAC) for a set of strategies based on their costs and utilities.

Usage

```
plot_ceac(payoffs_sim, wtp_range, col_names = c("cost", "utility"))
```

Arguments

`payoffs_sim`	A 3D array containing the simulated payoffs. The dimensions should be decision, payoff (cost and utility), and simulation.
`wtp_range`	A numeric vector specifying the range of willingness to pay (WTP) thresholds.
`col_names`	A character vector specifying the names of the columns for cost and utility. Default is c("cost", "utility").

Value

A ggplot object representing the CEAC.

Examples

```
# Example payoffs simulation array
payoffs_sim <- array(
  data = c(1000, 2000, 1500, 0.8, 0.85, 0.82, 1000, 2000, 1500, 0.8, 0.85, 0.82),
  dim = c(3, 2, 2),
  dimnames = list(c("StrategyA", "StrategyB", "StrategyC"), c("cost", "utility"), NULL)
)

# Define WTP range
wtp_range <- seq(0, 100000, by = 1000)

# Plot CEAC
ceac_plot <- plot_ceac(payoffs_sim, wtp_range)
print(ceac_plot)
```

`prob2rate` *Convert Probability to Rate*

Description

This function converts a probability to a rate using the formula `-log(1 - prob)`.

Usage

```
prob2rate(prob)
```

Arguments

`prob`	A numeric value representing the probability.

Value

A numeric value representing the rate.

Examples

```
prob <- 0.1
rate <- prob2rate(prob)
print(rate)
```

`rate2prob` *Convert Rate to Probability*

Description

This function converts a rate to a probability using the formula `1 - exp(-rate)`.

Usage

```
rate2prob(rate)
```

Arguments

`rate`	A numeric value representing the rate.

Value

A numeric value representing the probability.

Examples

```
rate <- 0.1
prob <- rate2prob(rate)
print(prob)
```

`run_twig` *Run a twig model*

Description

This function runs a twig model, which currently can be either a decision tree or a Markov model.

Usage

```
run_twig(
  twig_obj,
  params,
  n_cycles = NULL,
  verbose = FALSE,
  parallel = FALSE,
  offset_trace_cycle = 1,
  ncore = NULL,
  progress_bar = TRUE
)
```

Arguments

`twig_obj`	A twig object created by the `twig` function.
`params`	A data frame or list of parameters to be used in the model.
`n_cycles`	An integer specifying the number of cycles for a Markov model. Default is NULL.
`verbose`	A logical value indicating whether to print detailed output. Default is FALSE.
`parallel`	A logical value indicating whether to run the model in parallel. Default is FALSE.
`offset_trace_cycle`	An integer specifying the offset trace cycle. Default is 1. This is used to adjust the cycle number in the trace output. If set to 0, the initial state distribution will be used as the first cycle. If set to 1, the initial state distribution will be ignored in the Markov trace. In both situations, the total number of cycles will be the same as the input n_cycles.
`ncore`	An integer specifying the number of cores to use for parallel processing. Default is total number of cores - 1.
`progress_bar`	A logical value indicating whether to display a progress bar. Default is TRUE.

Value

A list containing the results of the model run. The list includes the following elements:

- mean_ev A matrix of size decision x payoff containing the mean expected values (EV)s across simulations if params is a data.frame with more than 1 row.
- sim_ev An array of size decision x payoff x simulation containing the simulated expected values (EV) by simulation.

The following will also be returned if verbose is TRUE for Markov models:

- sim: The simulation ID. If params is a dataset, only the first simulation will be used (sim = 1).
- evaluated_funs: A list of dataframes of evaluated functions. Each function returns a data frame enumerating the dependencies of the function along with the value returned by that function for each combination of values.
- evaluated_prob_funs_combined: A data frame containing the evaluated probability function values for each state, cycle, decision, and event that are merged into a single dataframe.
- path_event_options: A data frame of the event options along each path. Rows = paths, columns = event_options.
- path_probs: A data frame containing the path probabilities for each state, cycle, and decision.
- event_probs: A data frame containing the event options along each path and the destination.
- markov_trans_probs: An array containing the transition probabilities for each origin state, destination state, cycle, and decision.
- markov_trace: An array containing the Markov trace for each cycle, state, and decision.
- cycle_payoffs: An array containing the payoffs for each cycle, state, decision, and payoff.
- cycle_ev: An array containing the Expected Value (EV) for each cycle, state, decision and payoff. cycle_ev = markov_trace * cycle_payoffs.
- sim_ev: A matrix of total expected values (EV) of size decision x payoffs.
- mean_ev: A matrix of mean expected values (EV) across simulations. Since this is for a single simulation, mean_ev = sim_ev.

The following will also be returned if verbose is TRUE for decision trees:

- sim: The simulation ID. If params is a dataset, only the first simulation will be used (sim = 1).
- evaluated_funs: A list of dataframes of evaluated functions. Each function returns a data frame enumerating the dependencies of the function along with the value returned by that function for each combination of values.
- evaluated_prob_funs_combined: A data frame of the probability function values evaluated by decision and events harmonized to the same combinations of decisions and events across all probability functions.
- event_probs: A data frame containing the probability of event options by decision.
- outcome_probs: A matrix of outcome probabilities. Outcomes are the terminal event transitions of size decision x outcomes.
- path_event_options: A data frame of the event options along each path. Rows = paths, columns = events.
- path_probs: A matrix containing the path probabilities of size decision x paths.
- path_payoffs: An array containing the path payoffs of size decision x paths x payoffs. Paths are indexed by their final outcomes in the twig and a key to event options is provided in path_event_options.
- path_ev: An array containing the path expected values (EV) = path_probs x path_payoffs. This is also of size decision x paths x payoffs.
- sim_ev: A matrix of total expected values (EV) of size decision x payoffs.
- mean_ev: A matrix of mean expected values (EV) across simulations. Since this is for a single simulation, mean_ev = sim_ev.

See Also

Getting started guide and vignettes

Examples

```
library(twig)

# define a Markov model twig
mytwig <- twig() +
  decisions(names = c(A,B)) +
  states(names = c(H,D),
         init_probs = c(1,0)) +
  event(name = death_event,
        options = c(yes, none),
        probs = c(pDie, leftover),
        transitions = c(D, stay)) +
  payoffs(names = c(utility))

# define the parameters
params <- list(prob_die = 0.1, rrA = 0.9)

# define vectorized functions
pDie <- function(decision, state, prob_die, rrA){
  # prob death is 0.1 if healthy and 0 otherwise
  prob_die * (state=="H") *
    # multiplied by a relative risk of 0.9 if the decision is A, and 1 otherwise
    rrA ^ (decision=="A")
}

utility <- function(state){
  1 * (state=="H") # utility is 1 if healthy and 0 otherwise
}

# run the model for 10 cycles
run_twig(mytwig, params = params, n_cycles = 10)

# see the vignettes for more examples
```

states *Add Markov states to a twig*

Description

Add Markov states to a twig

Usage

```
states(names, init_probs, max_cycles = NULL)
```

Arguments

`names`	... a character vector of Markov state names. They don't need to be included in quotes.
`init_probs`	... a vector of initial probs, these could be numeric or function names. The functions can depend on the decision and variables in the params list of dataframe. One of these can be leftover for the remaining probability in that event. init_probs must have the same length as state names.
`max_cycles`	... optional max tunnel lenghts (tunnel length). This defines the duration allowable in each state. If ignored a length of 1 is assumed. #' max_cycles if provided must have the same length as state names.

Value

a twig layer with Markov state names

Examples

```
states(names = c(H,S,D),
                 init_probs = c(0.5, prob_fun, leftover),
                 max_cycles = c(1, 2, 1))
```

`twig` *Create a new twig object*

Description

This function initializes a new twig object, which can be used to build Markov models and decision trees.

Usage

```
twig()
```

Value

A new twig object of class `decision_twig` and `twig_class` by default.

Examples

```
twig_obj <- twig()
# see vignettes for more
```

`%out%` *Negation of %in% operator*

Description

This function checks if elements of a vector are not in another vector.

Usage

```
x %out% table
```

Arguments

`x`	A vector of values to be checked.
`table`	A vector of values to be compared against.

Value

A logical vector indicating if the elements of `x` are not in `table`.

Examples

```
x <- c("A", "B", "C")
table <- c("B", "C", "D")
x %out% table
```