Package 'obfuscatoR'

July 22, 2025

Type Package

Title Obfuscation Game Designs

Version 0.2.2

Maintainer Erlend Dancke Sandorf <erlend.dancke.sandorf@nmbu.no>

Description When people make decisions, they may do so using a wide variety of decision rules. The package allows users to easily create obfuscation games to test the obfuscation hypothesis. It provides an easy to use interface and multiple options designed to vary the difficulty of the game and tailor it to the user's needs. For more detail: Chorus et al., 2021, Obfuscation maximization-based decision-making: Theory, methodology and first empirical evidence, Mathematical Social Sciences, 109, 28-44, <doi:10.1016/j.mathsocsci.2020.10.002>.

License GPL-3

Encoding UTF-8

URL https://obfuscator.edsandorf.me,
https://github.com/edsandorf/obfuscatoR

Depends R (>= 3.1.0)

Imports Rfast, stats, matrixStats, stringr, readr, tibble, crayon

Suggests testthat, knitr, rmarkdown

RoxygenNote 7.2.3

VignetteBuilder knitr

NeedsCompilation no

Author Erlend Dancke Sandorf [aut, cre],
Caspar Chorus [aut],
Sander van Cranenburgh [aut]

Repository CRAN

Date/Publication 2023-09-07 13:10:02 UTC

Contents

`.onAttach` *Print package startup message*

Description

The function is called when the package is loaded through library or require.

Usage

```
.onAttach(libname, pkgname)
```

Arguments

`libname`	Library name
`pkgname`	Package name

Value

Nothing

`calculate_entropy` *Calculate the entropy of each action in the design*

Description

The function is a wrapper for `calc_entropy` and is meant for external use by the user. The goal for the decision maker is to choose an action such that the observer is left as clueless as possible as to which rule governs his actions, i.e. maximize entropy.

Usage

```
calculate_entropy(design, priors = NULL)
```

Arguments

`design`	A matrix with rows equal to the number of rules and columns equal to the number of actions or a list of such matrices.
`priors`	A vector of prior values. If the design is a list of matrices, priors can be a matrix with rows equal to the length of the design and columns equal to the number of rules.

Value

A list of of vectors of entropies for each possible action with the following attributes:

1. design
2. priors
3. pr_aj_rk
4. pr_rk_aj

Examples

```
  design <- matrix(c(-1, -1, -1, -1,  1,
                     -1,  0,  0, -1,  0,
                     -1,  0, -1,  0,  0,
                      0,  0, -1,  0, -1), nrow = 4L, byrow = TRUE)

calculate_entropy(design)
```

calculate_payouts *Calculate payouts*

Description

The function is a wrapper function for `calc_payout_obs` and `calc_payout_dm`, and exported to be used by the user. It calculates the expected payout to both observers and decision makers for each possible action undertaken by the decision maker, and the observers choice of whether or not to try and guess the rule.

Usage

```
calculate_payouts(
  entropy,
  pay_obs,
  pay_dm,
  pay_no_guess,
  deterministic = FALSE
)
```

Arguments

`entropy`	A list containing the entropy
`pay_obs`	A numeric with pay to the observer for guessing correctly
`pay_dm`	A numeric with pay to the decision maker if the observer does not guess
`pay_no_guess`	A numeric with pay to the observer for not guessing
`deterministic`	If TRUE a deterministic procedure is used to determine whether the observer tries to guess. Default is FALSE and the probability is calculated using a logit expression

Value

A list or list of lists where each list contains the payout to the observer and decision maker.

calc_entropy *Calculate Shannon's Entropy*

Description

The function calculates Shannon's Entropy. The function is meant for internal use only. To calculate the entropy for each action in the design, please use the wrapper function `calculate_entropy`

Usage

```
calc_entropy(design, priors = NULL)
```

Arguments

design	A matrix with rows equal to the number of rules and columns equal to the number of actions or a list of such matrices.
priors	A vector of prior values. If the design is a list of matrices, priors can be a matrix with rows equal to the length of the design and columns equal to the number of rules.

Value

Returns a vector of entropies for each possible action with the following attributes:

1. design
2. priors
3. pr_aj_rk
4. pr_rk_aj

---

calc_payout_dm *Calculate expected payout to the decision maker*

Description

The function calculates the expected payout to the decision maker. The payout to the decision maker depends on whether or not the observer tries to guess the rule, and the monetary payout for choosing an action that leaves the observer clueless enough about the rule to refrain from guessing. The function is meant for internal use only. To calculate the payout to the decision maker, use the wrapper function calculate_payouts.

Usage

```
calc_payout_dm(pr_guess, pay_dm)
```

Arguments

pr_guess	A vector of probabilities that the observer will guess.
pay_dm	The pay to the decision maker if the observer does not guess.

Value

A vector of expected payouts for each possible guess made by the observer

calc_payout_obs *Calculate expected payout to the observer*

Description

The function calculates the expected payout to the observer. The payout to the observer depends on the posterior probabilities, i.e. the probability of a rule conditional on observing an action, and the monetary payout for guessing correctly. The function is meant for internal use only. To calculate the payout to the observer, use the wrapper function `calculate_payouts`.

Usage

```
calc_payout_obs(pr_rk_aj, pay_obs)
```

Arguments

pr_rk_aj	A matrix of posterior probabilities
pay_obs	The pay to the observer for guessing correctly.

Value

A vector of expected pays for each possible guess

calc_pr_aj_rk *Calculate Pr(a_j|r_k)*

Description

The function calculates the probability of an action conditional on a given rule and is part of calculating the entropy of an action. The function is meant for internal use only.

Usage

```
calc_pr_aj_rk(design)
```

Arguments

design	A matrix with rows equal to the number of rules and columns equal to the number of actions or a list of such matrices.

Value

An r x a matrix of probabilities

calc_pr_guess *Calculate the probability that the observer will try to guess the rule*

Description

The function calculates the probability that an observer will try to guess which rule governs the decision maker's actions. The function is meant for internal use only. It can be printed alongside the payouts calculated using `print_payout` if `print_all = TRUE`.

Usage

```
calc_pr_guess(expected_payout_obs, payout_obs_no_guess, deterministic)
```

Arguments

`expected_payout_obs`	Vector of expected payout to the observer from guessing
`payout_obs_no_guess`	The payout to the observer from not guessing
`deterministic`	A boolean equal to TRUE if we treat the decision to guess as deterministic. Defaults to TRUE.

Value

A vector with the probabilities that an observer will guess

calc_pr_rk_aj *Calculate Pr(r_k|a_j)*

Description

The function calculates the probability of a rule conditional on observing a given action and is part of calculating the entropy of an action. This probability is also referred to as the posterior probability. The function is meant for internal use only.

Usage

```
calc_pr_rk_aj(pr_aj_rk, priors)
```

Arguments

`pr_aj_rk`	A matrix with the probabilities of actions conditional on a given rule.
`priors`	A vector of prior values. If the design is a list of matrices, priors can be a matrix with rows equal to the length of the design and columns equal to the number of rules.

Value

An r x a matrix of probabilities

`check_design_opt` *Check design options*

Description

The function checks the list of design options specified by the user and sets sensible defaults where no option is specified. The function is meant for internal use only and is not exported to be used by the users. All options can be overridden by the the user by appropriately specifying `design_opt_input`.

Below is a list defining each of the options available to be specified in `design_opt_input`.

Usage

```
check_design_opt(design_opt_input)
```

Arguments

`design_opt_input`
: A list of user supplied design options.

Details

rules Number of rules (i.e. rows)

actions Number of actions (i.e. columns)

min Minimum number of actions available for the considered rule

max Maximum number of actions available for the considered rule

min_fit Minimum number of rules fitting each permitted action conditional on the rule

obligatory Number of rules with obligatory actions

sd_entropy Specifies the standard deviation of the entropy values

designs Number of designs to generate

max_iter Maximum number of iterations before stopping search for designs

seed A seed for the random number generator. Useful for replicability

Value

Returns a list of design options with the missing from input replaced by default values

construct_design *Function to create a rule-action matrix*

Description

The function creates a rule-action matrix (i.e. an obfuscation design) subject to a list of pre-programmed restrictions. These restrictions are in place to ensure that no invalid designs are created. Some of these restrictions can be changed by the user by appropriately specifying the `design_opt_input`. Each matrix is a design for one period of the the obfuscation game. This function is for internal use only. To create an obfuscation design, the user should use `generate_designs`.

Usage

```
construct_design(design_opt)
```

Arguments

`design_opt`	List of design options

Value

A rules-action matrix

extract_attr *Extract attributes*

Description

Extracts the attributes of objects nested in a list

Usage

```
extract_attr(x, str_attr)
```

Arguments

`x`	A list of objects with attributes or an object with an attribute
`str_attr`	A non-empty character string specifying which attribute is to be extracted

Value

Returns a list the length of x containing the specified attribute. If the attribute does not exist, returns NULL

Examples

```
design_opt_input <- list(rules = 4, actions = 5)
design <- generate_designs(design_opt_input)
extract_attr(design, "design_conditions")

design_opt_input <- list(rules = 4, actions = 5, designs = 2)
design <- generate_designs(design_opt_input)
extract_attr(design, "design_conditions")
```

---

generate_designs *Generate obfuscation designs*

Description

The function takes the list of design options `design_opt_input` and generates one or more obfuscation designs subject to the specified restrictions. A full specification of all the options available can be found in the manual along with detailed examples of different designs. At a minimum the user must supply the number of rules and actions, i.e. the dimensions of the design problem.

Usage

```
generate_designs(design_opt_input = list())
```

Arguments

`design_opt_input`
: A list of user supplied design options.

Value

A list of matrices with rules and actions

Examples

```
design_opt_input <- list(rules = 4,
                         actions = 5)

generate_designs(design_opt_input)
```

last *Get the last element of a vector*

Description

`last` extracts the last element of a vector

Usage

```
last(x)
```

Arguments

`x`	A vector

Examples

```
x <- 1:4
last(x)

x <- c("hello", "my", "name", "is", "buttons")
last(x)
```

print_design *Prints the design*

Description

Takes a design or list of designs and prints them to the console. To store a design, please see `save_design`. Depending on the print options, additional text is provided with information on the considered rule and/or the design generation process.

Usage

```
print_design(design, print_all = FALSE)
```

Arguments

`design`	A matrix with rows equal to the number of rules and columns equal to the number of actions
`print_all`	If TRUE prints information on the number of iterations and and whether all design conditions were met. Default is FALSE

Examples

```
design_opt_input <- list(rules = 4,
                         actions = 5)

design <- generate_designs(design_opt_input)

print_design(design)
print_design(design, TRUE)
```

`print_entropy` *Prints the entropy of the different actions*

Description

The function prints the vector of entropies for each possible action. Depending on printing options, additional information about the probability calculations can be provided.

Usage

```
print_entropy(entropy, digits = 3, print_all = FALSE)
```

Arguments

`entropy`	The entropy measure from `calculate_entropy`
`digits`	The number of digits to round to. Default 3.
`print_all`	If TRUE will print all information on intermediary calculations

Examples

```
design <- matrix(c(-1, -1, -1, -1,  1,
                   -1,  0,  0, -1,  0,
                   -1,  0, -1,  0,  0,
                    0,  0, -1,  0, -1), nrow = 4, byrow = TRUE)

entropy <- calculate_entropy(design)

print_entropy(entropy)
print_entropy(entropy, digits = 4)
print_entropy(entropy, print_all = TRUE)
```

print_payout *Print the payouts*

Description

The function formats and prints the payout to the observer and decision maker.

Usage

```
print_payout(payout, digits = 3, print_all = FALSE)
```

Arguments

payout	A list of calculated payouts
digits	The number of digits to round to. Default 3.
print_all	If TRUE will print the probabilities of guessing

save_design *Save obfuscation designs*

Description

The function takes a design or a list of designs and stores them in .csv files in the specified folder.

Usage

```
save_design(x, x_name, path = getwd())
```

Arguments

x	A design or list of designs
x_name	A character string with the name of the file
path	A string giving the path to where the designs are stored. The default is the current working directory

Value

Nothing is returned

Index