Package: voteSim (via r-universe)

September 12, 2024

Type Package

Title Generate Simulated Data for Voting Rules using Evaluations

Version 0.1.1

Maintainer Antoine Rolland <antoine.rolland@univ-lyon2.fr>

Description Provide functions to generate random simulated evaluations on candidates by voters for evaluation-based elections. Functions are based on several models for continuous or discrete evaluations.

License GPL-3

Encoding UTF-8

URL https://eric.univ-lyon2.fr/arolland/

Imports truncnorm, extraDistr, GenOrd

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

RoxygenNote 7.2.3

NeedsCompilation no

Author Antoine Rolland [aut, cre], Nathan Grimault [aut]

Date/Publication 2023-12-19 14:40:03 UTC

Repository https://antoinerollandlyon.r-universe.dev

RemoteUrl https://github.com/cran/voteSim

RemoteRef HEAD

RemoteSha c91d3943feb98e21e8ffd78cded67d593e94ad71

Contents

`distance` *Distance formula*

Description

Distance formula

Usage

```
distance(votant, candidats)
```

Arguments

`votant`	array
`candidats`	array

Value

distance

distance_to_pref *Distance formula*

Description

Distance formula

Usage

```
distance_to_pref(distance_matrix)
```

Arguments

distance_matrix	distance_matrix

Value

mat_inverse

DistToScores *Distance to score*

Description

Distance to score

Usage

```
DistToScores(dist, dim = 2, method = "linear", lambda = 5)
```

Arguments

dist	int
dim	dimension int
method	method string
lambda	lambdad int

Value

score

`generate_beta` *Generates a simulation of voting according to a beta law, returns voters preferences*

Description

Generates a simulation of voting according to a beta law, returns voters preferences

Usage

```
generate_beta(
  n_voters,
  n_candidates,
  beta_a = 0.5,
  beta_b = 0.5,
  lambda = 0,
  min = 0,
  max = 1
)
```

Arguments

`n_voters`	integer, represents the number of voters in the election
`n_candidates`	integer, represents the number of candidates in the election
`beta_a`	double, parameter of the Beta law (by default 0.5)
`beta_b`	double, parameter of the Beta law (by default 0.5)
`lambda`	double, alternative parameter of the Beta law
`min`	int, the minimum value of the range of possible scores (by default 0)
`max`	int, the maximum value of the range of possible scores (by default 1)

Value

scores

Examples

```
voting_situation<- generate_beta(n_voters=10, n_candidates=3, beta_a=1, beta_b=5)
```

generate_beta_binomial *Generate beta-binomial scores*

Description

This function generates discrete scores following a beta-binomial distribution on a given scale

Usage

```
generate_beta_binomial(
  n_voters,
  n_candidates,
  min = 0,
  max = 10,
  alpha = 0.5,
  beta = 0.5
)
```

Arguments

n_voters	integer, the number of voters to generate scores for.
n_candidates	integer, The number of candidates to generate scores for.
min	The minimum value of the distribution, by default 0
max	The maximum value of the distribution, by default 10
alpha	The first parameter of the beta-binomial distribution, by default 0.5
beta	The second parameter of the beta-binomial distribution, by default 0.5

Value

A matrix of scores with 'n_candidates' rows and 'n_voters' columns.

Examples

```
voting_situation <- generate_beta_binomial(n_voters=10, n_candidates=3,  max=7)
```

generate_binomial *Generate binomial scores*

Description

This function generates discrete scores following a binomial distribution on a given scale

Usage

```
generate_binomial(n_voters, n_candidates, min = 0, max = 10, mean = 5)
```

Arguments

n_voters	integer, the number of voters to generate scores for.
n_candidates	integer, The number of candidates to generate scores for.
min	The minimum value of the distribution, by default 0
max	The maximum value of the distribution, by default 10
mean	The mean value of the distribution, by default 5

Value

A matrix of scores with 'n_candidates' rows and 'n_voters' columns.

Examples

```
voting_situation <- generate_binomial(n_voters=10, n_candidates=3, min=0, max=7, mean=5)
```

generate_dirichlet *Generate multinomial scores*

Description

This function generates scores following a Dirichlet distribution

Usage

```
generate_dirichlet(n_voters, n_candidates, probs = 0)
```

Arguments

n_voters	integer, the number of voters to generate scores for.
n_candidates	integer, The number of candidates to generate scores for.
probs	A vector of size n_candidates corresponding to the parameters of the Dirichlet distribution. By default all values are equal to 1.

Value

A matrix of scores with 'n_candidates' rows and 'n_voters' columns.

Examples

```
voting_situation <- generate_dirichlet(n_voters=10, n_candidates=3, probs=c(0.5, 0.3, 0.2))
```

`generate_discrete_copula_based`

Discrete Copula based scores

Description

This function generates discrete scores following marginals distributions linked by a copula #'

Usage

```
generate_discrete_copula_based(
  n_voters,
  n_candidates,
  min = 0,
  max = 10,
  margins = list("default"),
  cor_mat = 0
)
```

Arguments

`n_voters`	integer, the number of voters to generate scores for.
`n_candidates`	integer, The number of candidates to generate scores for.
`min`	The minimum value of the distribution, by default 0
`max`	The maximum value of the distribution, by default 10
`margins`	A list of n_candidates cumulative distribution vectors of length (max-min-1) : the last value of the cumulative distribution, 1, should be omitted. By default margin distribution are uniform distributions.
`cor_mat`	A matrix of correlation coefficients between the n_candidates distributions. By default all correlation coefficients are set up alternatively to 0.5 or -0.5.

Value

A matrix of scores with 'n_candidates' rows and 'n_voters' columns.

Examples

```
# Example for 3 candidates, binomial distributions
 min=0
 max=7
n_candidates<-3
distribution<-dbinom(x=(min:max), size=max, prob=0.7)
distribution_cumul<-cumsum(distribution)
distribution_cumul<-distribution_cumul[-length(distribution_cumul)]
margins <- matrix(rep(distribution_cumul, n_candidates), ncol=n_candidates)
margins <-as.list(as.data.frame(margins))
cor_mat<-matrix(c(1,0.8,0,0.8,1,0, 0,0,1), ncol=n_candidates)
voting_situation <- generate_discrete_copula_based(10, 3, max=max, margins=margins, cor_mat=cor_mat)
```

`generate_multinom` *Generate multinomial scores*

Description

This function generates discrete scores following a multinomial distribution on a given scale

Usage

```
generate_multinom(n_voters, n_candidates, max = 10, probs = 0)
```

Arguments

`n_voters`	integer, the number of voters to generate scores for.
`n_candidates`	integer, The number of candidates to generate scores for.
`max`	The maximum value of the distribution, by default 10. It also corresponds to the sum of scores on all the candidates
`probs`	A vector of size n_candidates corresponding to the parameters of the multinomial distribution. By default all values are equal to 1/n_candidates

Value

A matrix of scores with 'n_candidates' rows and 'n_voters' columns.

Examples

```
voting_situation <- generate_multinom(n_voters=10, n_candidates=3,  max=100, probs=c(0.5, 0.3, 0.2))
```

generate_norm *Generate truncated normal scores*

Description

This function generates truncated normal scores using the 'rtruncnorm' function from the 'truncnorm' package.

Usage

```
generate_norm(n_voters, n_candidates, min = 0, max = 1, mean = 0.5, sd = 0.25)
```

Arguments

n_voters	The number of voters to generate scores for.
n_candidates	The number of candidates to generate scores for.
min	The minimum value of the truncated normal distribution.
max	The maximum value of the truncated normal distribution.
mean	The mean of the truncated normal distribution.
sd	The standard deviation of the truncated normal distribution.

Value

A matrix of scores with 'n_candidates' rows and 'n_voters' columns.

Examples

```
voting_situation<- generate_norm(n_voters=10, n_candidates=3, min=0, max=10, mean=0.7)
```

generate_spatial *Generate spatial simulation*

Description

This function generates spatial data consisting of `n_voters` voters and `n_candidates` candidates. The spatial model is created by placing the candidates on a 2-dimensional plane according to the `placement` parameter, and then computing a distance matrix between voters and candidates. The distances are then transformed into scores using the `score_method` parameter. Finally, a plot of the candidates and voters is produced.

Usage

```
generate_spatial(
  n_voters,
  n_candidates,
  placement = "uniform",
  score_method = "linear",
  dim = 2
)
```

Arguments

n_voters	The number of voters.
n_candidates	The number of candidates.
placement	The method used to place the candidates on the 2-dimensional plane. Must be either "uniform" or "beta". Default is "uniform".
score_method	The method used to transform distances into scores. Must be either "linear" or "sigmoide". Default is "linear".
dim	The dimension of the latent space (by default dim =2)

Value

A matrix of scores.

Examples

```
generate_spatial(n_candidates = 5,n_voters = 100, placement = "uniform", score_method = "linear")
```

---

generate_unif_continuous

Generates a simulation of voting according to a uniform law, returns voters preferences

---

Description

Generates a simulation of voting according to a uniform law, returns voters preferences

Usage

```
generate_unif_continuous(n_voters, n_candidates, min = 0, max = 1)
```

Arguments

n_voters	integer, represents the number of voters in the election
n_candidates	integer, represents the number of candidates in the election
min	int, the minimum value of the range of possible scores (by default 0)
max	int, the maximum value of the range of possible scores (by default 1)

Value

scores

Examples

```
voting_situation<- generate_unif_continuous(n_voters=10, n_candidates=3, min=0, max=10)
```

`generate_unif_disc` *Generate uniform discrete scores*

Description

This function generates uniform discrete scores on a given scale

Usage

```
generate_unif_disc(n_voters, n_candidates, min = 0, max = 10)
```

Arguments

`n_voters`	integer, the number of voters to generate scores for.
`n_candidates`	integer, The number of candidates to generate scores for.
`min`	The minimum value of the distribution, by default 0
`max`	The maximum value of the distribution, by default 10

Value

A matrix of scores with 'n_candidates' rows and 'n_voters' columns.

Examples

```
voting_situation <- generate_unif_disc(n_voters=10, n_candidates=3, min=0, max=5)
```

icdf *Generalized inverse of the empirical cumulative function.*

Description

Generalized inverse of the empirical cumulative function.

Usage

```
icdf(u, x, n)
```

Arguments

u	a numeric vector of quantiles to be transformed.
x	a numeric vector of data values.
n	a positive integer specifying the length of the output vector.

Details

Computes the generalized inverse of the empirical cumulative function, which transforms quantiles u to the corresponding values of x based on the frequency distribution of x.

Value

a numeric vector of transformed quantiles.

preferences_to_ranks *Preferences_to_ranks*

Description

Preferences_to_ranks

Usage

```
preferences_to_ranks(preferences)
```

Arguments

preferences	voters preferences

Value

ranks

rename_rows *Rename_rows*

Description

Rename_rows

Usage

```
rename_rows(preferences)
```

Arguments

preferences	voters preferences

Value

preferences

ScoresToDist *Score to distance*

Description

Score to distance

Usage

```
ScoresToDist(x, dim = 2, method = "linear")
```

Arguments

x	score
dim	dimension int
method	method string

Value

distance

Index