Package ‘TUvalues’

May 22, 2025

Type Package

Title Tools for Calculating Allocations in Game Theory using Exact and Approximated Methods

Version 1.0.0

Description The main objective of cooperative games is to allocate a good among the agents involved. This package includes the most well-known allocation rules, i.e., the Shapley value, the Banzhaf value, the egalitarian rule, and the equal surplus division value. In addition, it considers the point of view of a priori unions (situations in which agents can form coalitions). For this purpose, the package includes the Owen value, the Banzhaf-Owen value, and the corresponding extensions of the egalitarian rules. All these values can be calculated exactly or estimated by sampling.

License AGPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.2

URL https://github.com/mariaguilleng/TUvalues

BugReports https://github.com/mariaguilleng/TUvalues/issues

Imports utils, gtools

NeedsCompilation no

Author Maria D. Guillen [cre, aut] (ORCID: <https://orcid.org/0000-0002-2445-5654>), Juan Carlos Gonçalves [aut] (ORCID: <https://orcid.org/0000-0002-0867-0004>)

Maintainer Maria D. Guillen <maria.guilleng@umh.es>

Repository CRAN

Date/Publication 2025-05-22 17:10:02 UTC

Contents

banzhaf *Banzhaf value*

Description

Calculate the Banzhaf value

Usage

```
banzhaf(
  characteristic_func,
  method = "exact",
  n_rep = 10000,
  n_players = 0,
  replace = FALSE
)
```

Arguments

`characteristic_func`
The valued function defined on the subsets of the number of players.

`method`
Method used to calculate the Banzhaf value. Valid methods are: `exact` for the exact calculation or `appro` for approximated polynomial calculation based on sampling.

`n_rep`
Only used if `method` is `appro`. The number of iterations to perform in the approximated calculation

`n_players`
Only used if `characteristic_func` is a `function`. The number of players in the game.

`replace`
should sampling be with replacement?

Value

The Banzhaf value for each player

Examples

```
n <- 8
v <- function(coalition) {
if (length(coalition) > n/2) {
   return(1)
 } else {
   return(0)
 }
}
banzhaf(v, method = "exact", n_players = n)
banzhaf(v, method = "appro", n_rep = 2000, n_players = n, replace = TRUE)

v<-c(0,0,0,1,2,1,3)
banzhaf(v, method = "exact")
banzhaf(v, method = "appro", n_rep = 2000, replace = TRUE)
```

`banzhaf_appro` *Banzhaf Index (approximated)*

Description

Calculate the approximated Banzhaf Index based on sampling

Usage

```
banzhaf_appro(characteristic_func, n_players, n_rep, replace = TRUE)
```

Arguments

`characteristic_func`
The valued function defined on the subsets of the number of players

`n_players` The number of players

`n_rep` The number of iterations to perform in the approximated calculation

`replace` should sampling be with replacement?

Value

The Shapley value for each player

banzhaf_exact *Banzhaf Index (exact)*

Description

Calculate the approximated Banzhaf Index

Usage

```
banzhaf_exact(characteristic_func, n_players)
```

Arguments

characteristic_func	The valued function defined on the subsets of the number of players
n_players	The number of players in the game.

Value

The Banzhaf Index for each player

banzhaf_owen *Banzhaf-Owen value*

Description

Calculate the Banzhaf-Owen value

Usage

```
banzhaf_owen(
  characteristic_func,
  union,
  method = "exact",
  n_rep = 10000,
  n_players = 0,
  replace = TRUE
)
```

Arguments

`characteristic_func`	The valued function defined on the subsets of the number of players
`union`	List of vectors indicating the a priori unions between the players
`method`	Method used to calculate the Owen value. Valid methods are: `exact` for the exact calculation or `appro` for approximated polynomial calculation based on sampling.
`n_rep`	Only used if `method` is `appro`. The number of iterations to perform in the approximated calculation
`n_players`	Only used if `characteristic_func` is a `function`. The number of players in the game.
`replace`	should sampling be with replacement?

Value

The Banzhaf-Owen value for each player

Examples

```
characteristic_func <- c(0,0,0,0,30,30,40,40,50,50,60,70,80,90,100)
union <- list(c(1,3),c(2),c(4))
banzhaf_owen(characteristic_func, union)
banzhaf_owen(characteristic_func, union, method = "appro", n_rep = 4000)
```

`banzhaf_owen_appro` *Banzhaf-Owen Value*

Description

Calculate the approximated Banzhaf-Owen value

Usage

```
banzhaf_owen_appro(characteristic_func, union, n_players, n_rep, replace)
```

Arguments

`characteristic_func`	The valued function defined on the subsets of the number of players
`union`	List of vectors indicating the a priori unions between the players
`n_players`	The number of players
`n_rep`	Only used if `method` is `appro`. The number of iterations to perform in the approximated calculation.
`replace`	should sampling be with replacement?

Value

The Banzhaf-Owen Index for each player

---

banzhaf_owen_exact *Banzhaf-Owen Value*

Description

Calculate the approximated Banzhaf-Owen value

Usage

```
banzhaf_owen_exact(characteristic_func, union, n_players)
```

Arguments

`characteristic_func`	The valued function defined on the subsets of the number of players
`union`	List of vectors indicating the a priori unions between the players
`n_players`	The number of players in the game.

Value

The Banzhaf Index for each player

---

coalitions *coalitions*

Description

Create all the possible coalitions given the number of players

Usage

```
coalitions(n_players)
```

Arguments

`n_players`	Number of players

Value

A list containing a `data.frame` of the binary representation of the coalitions and a `vector` of the classical representation (as sets) of the coalitions

egalitarian *Egalitarian value*

Description

Calculate the egalitarian value

Usage

```
egalitarian(characteristic_func, n_players = 0)
```

Arguments

`characteristic_func`
The valued function defined on the subsets of the number of players

`n_players` Only used if `characteristic_func` is a `function`. The number of players in the game.

Value

The egalitarian value for each player

Examples

```
n <- 10
v <- function(coalition) {
  if (length(coalition) > n/2) {
    return(1)
  } else {
    return(0)
  }
}
egalitarian(v,n)
```

equal_surplus_division *Equal Surplus Division value*

Description

Calculate the equal surplus division value

Usage

```
equal_surplus_division(characteristic_func, n_players = 0)
```

Arguments

characteristic_func
: The valued function defined on the subsets of the number of players

n_players
: Only used if characteristic_func is a function. The number of players in the game.

Value

The equal surplus division value for each player

Examples

```
n <- 10
v <- function(coalition) {
  if (length(coalition) > n/2) {
    return(1)
  } else {
    return(0)
  }
}
equal_surplus_division(v,n)
```

owen *Owen value*

Description

Calculate the Owen value

Usage

```
owen(
  characteristic_func,
  union,
  method = "exact",
  n_rep = 10000,
  n_players = 0
)
```

Arguments

characteristic_func
: The valued function defined on the subsets of the number of players.

union
: List of vectors indicating the a priori unions between the players.

method
: Method used to calculate the Owen value. Valid methods are: exact for the exact calculation or appro for approximated polynomial calculation based on sampling.

`n_rep` Only used if `method` is `appro`. The number of iterations to perform in the approximated calculation.

`n_players` The number of players in the game.

Value

The Owen value for each player.

Examples

```
n <- 10
v <- function(coalition) {
  if (length(coalition) > n/2) {
    return(1)
  } else {
    return(0)
  }
}
u <- lapply(1:(n/2), function(i) c(2*i - 1, 2*i))
owen(v, union = u, method = "appro", n_rep = 4000, n_players = n)

characteristic_func <- c(1,1,2,1,2,2,2)
union <- list(c(1,2),c(3))
owen(characteristic_func, union)
owen(characteristic_func, union, method = "appro", n_rep = 4000)
```

`owen_appro` *Owen value (approximation)*

Description

Calculate the approximated Owen value based on sampling

Usage

```
owen_appro(characteristic_func, union, n_players, n_rep)
```

Arguments

`characteristic_func` The valued function defined on the subsets of the number of players

`union` List of vectors indicating the a priori unions between the players

`n_players` The number of players

`n_rep` The number of iterations to perform in the approximated calculation

Value

The Owen value for each player

`owen_exact` *Owen value (exact)*

Description

Calculate the exact Owen

Usage

```
owen_exact(characteristic_func, union, n_players = NULL)
```

Arguments

`characteristic_func`	The valued function defined on the subsets of the number of players
`union`	List of vectors indicating the a priori unions between the players
`n_players`	The number of players

Value

The Owen value for each player

`predecessor` *Predecessor*

Description

Given a permutation 0 of players and a player i, calculate the set of predecessors of the player i in the order 0

Usage

```
predecessor(permutation, player, include_player = FALSE)
```

Arguments

`permutation`	A permutation of the players
`player`	Number of the player i
`include_player`	Whether the player i is included as predecessor of itself or not

Value

The set of predecessors of the player i in the order 0 `shapley` *Shapley value*

Description

Calculate the Shapley value

Usage

```
shapley(characteristic_func, method = "exact", n_rep = 10000, n_players = 0)
```

Arguments

`characteristic_func`
: The valued function defined on the subsets of the number of players.

`method`
: Method used to calculate the Shapley value. Valid methods are: `exact` for the exact calculation or `appro` for approximated polynomial calculation based on sampling.

`n_rep`
: Only used if `method` is `appro`. The number of iterations to perform in the approximated calculation.

`n_players`
: Only used if `characteristic_func` is a `function`. The number of players in the game.

Value

The Shapley value for each player.

Examples

```
n <- 10
v <- function(coalition) {
if (length(coalition) > n/2) {
   return(1)
 } else {
   return(0)
 }
}
shapley(v, method = "appro", n_rep = 4000, n_players = n)

n <- 3
v <- c(1,1,2,1,2,2,2)
shapley(v, method = "exact")
shapley(v, method = "appro", n_rep = 4000)
```

`shapley_appro` *Shapley value (approximation)*

Description

Calculate the approximated Shapley value based on sampling

Usage

```
shapley_appro(characteristic_func, n_players, n_rep)
```

Arguments

`characteristic_func`	The valued function defined on the subsets of the number of players
`n_players`	The number of players
`n_rep`	The number of iterations to perform in the approximated calculation

Value

The Shapley value for each player

`shapley_exact` *Shapley value (exact)*

Description

Calculate the exact Shapley value

Usage

```
shapley_exact(characteristic_func, n_players)
```

Arguments

`characteristic_func`	The valued function defined on the subsets of the number of players
`n_players`	The number of players

Value

The Shapley value for each player

Index