Package 'TUvalues'

November 8, 2026

Type Package

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

Version 1.3.0

Description The main objective of cooperative Transferable-Utility games (TU-games) is to allocate a good among the agents involved. The package implements major solution concepts including the Shapley value, Banzhaf value, and egalitarian rules, alongside their extensions for structured games: the Owen value and Banzhaf-Owen value for games with a priori unions, and the Myerson value for communication games on networks. To address the inherent exponential computational complexity of exact evaluation, the package offers both exact algorithms and sampling-based approximation methods, enabling the analysis of large-scale games. The characteristic function can be supplied as a vector or as an R function, so that coalition values are evaluated only when required. Additionally, it supports core set-based solutions, allowing computation of the vertices and the centroid of the core and checking whether an allocation belongs to it.

License GPL (>= 2)

Encoding UTF-8

RoxygenNote 7.3.3

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

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

Imports utils, stats, ROI, ROI.plugin.glpk, rcdd

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>), Jesús Camacho [aut] (ORCID: <https://orcid.org/0000-0002-2527-7351>)

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

Repository CRAN

Date/Publication 2026-11-07 23:40:10 UTC

Contents

TUvalues-package . . . 2
banzhaf . . . 3
banzhaf_owen . . . 4
centroid . . . 5
check_core . . . 7
coalitions . . . 8
communication_game . . . 9
coreVertex . . . 10
egalitarian . . . 12
egalitarian_unions . . . 13
equal_surplus_division . . . 14
equal_surplus_division_unions . . . 15
myerson . . . 16
myerson_unions . . . 18
owen . . . 19
shapley . . . 20

Index **23**

`TUvalues-package` *TUvalues: Tools for Calculating Allocations in Game Theory using Exact and Approximated Methods*

Description

The main objective of cooperative Transferable-Utility games (TU-games) is to allocate a good among the agents involved. The package implements major solution concepts including the Shapley value, Banzhaf value, and egalitarian rules, alongside their extensions for structured games: the Owen value and Banzhaf-Owen value for games with a priori unions, and the Myerson value for communication games on networks. To address the inherent exponential computational complexity of exact evaluation, the package offers both exact algorithms and sampling-based approximation methods, enabling the analysis of large-scale games. The characteristic function can be supplied as a vector or as an R function, so that coalition values are evaluated only when required. Additionally, it supports core set-based solutions, allowing computation of the vertices and the centroid of the core and checking whether an allocation belongs to it.

Acknowledgements

This research has been partially supported by Grants PID2021-124030NB-C31, PID2025-172680NB-I00 and PID2025-170122NB-C32 from MICINN, Spain, and ERDF, "A way to make Europe", European Union. This research was also partially supported by the CIGE/2026/47 grant, funded by the Conselleria de Educación, Cultura, Universidades y Empleo, Generalitat Valenciana.

Author(s)

Maintainer: Maria D. Guillen <maria.guilleng@umh.es> (ORCID)

Authors:

- Juan Carlos Gonçalves <jgoncalves@umh.es> (ORCID)
- Jesús Camacho <j.camacho@umh.es> (ORCID)

See Also

Useful links:

- https://github.com/mariaguilleng/TUvalues
- Report bugs at https://github.com/mariaguilleng/TUvalues/issues

---

banzhaf *Banzhaf value*

Description

Calculate the Banzhaf value

Usage

```
banzhaf(
  characteristic_func,
  n_players = 0,
  method = "exact",
  n_rep = 1000,
  replace = TRUE,
  echo = TRUE
)
```

Arguments

characteristic_func	The characteristic function of the game. It can be a numeric vector with the value of every coalition, in the order given by coalitions: of length $2^n - 1$, without the empty coalition (whose value is taken as 0), or of length 2^n, with the empty coalition first. Vectors support up to 30 players. It can also be a function that receives a coalition (a vector of players) and returns its value; in that case there is no limit on the number of players.
n_players	Only used if characteristic_func is a function. The number of players in the game.
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
replace	Only used if method is appro. Should sampling be with replacement? Without replacement, n_rep is reduced, if necessary, to the number of different coalitions that can be sampled for each player.
echo	Only used if method is appro. Show progress of the approximated calculation.

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_owen *Banzhaf-Owen value*

Description

Calculate the Banzhaf-Owen value

Usage

```
banzhaf_owen(
  characteristic_func,
  union,
  n_players = 0,
  method = "exact",
  n_rep = 1000,
  replace = TRUE,
  echo = TRUE
)
```

Arguments

characteristic_func	The characteristic function of the game. It can be a numeric vector with the value of every coalition, in the order given by coalitions: of length $2^n - 1$, without the empty coalition (whose value is taken as 0), or of length 2^n, with the empty coalition first. Vectors support up to 30 players. It can also be a function that receives a coalition (a vector of players) and returns its value; in that case there is no limit on the number of players.
union	List of vectors indicating the a priori unions between the players
n_players	Only used if characteristic_func is a function. The number of players in the game.
method	Method used to calculate the Banzhaf-Owen value. Valid methods are: exact for the exact calculation or appro for approximated polynomial calculation based on sampling proposed by Saavedra-Nieves & Fiestras-Janeiro (2021).
n_rep	Only used if method is appro. The number of iterations to perform in the approximated calculation
replace	Only used if method is appro. Should sampling be with replacement? Without replacement, n_rep is reduced, if necessary, to the number of different coalitions that can be sampled for each player.
echo	Only used if method is appro. Show progress of the approximated calculation.

Value

The Banzhaf-Owen value for each player

References

Saavedra-Nieves, A., & Fiestras-Janeiro, M. G. (2021). Sampling methods to estimate the Banzhaf–Owen value. Annals of Operations Research, 301(1), 199-223.

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)
```

---

centroid *Centroid of the core of the game*

Description

Calculate the centroid of the core of the game if it exists.

Usage

```
centroid(
  characteristic_func,
  n_players = 0,
  method = "exact",
  n_rep = 500,
  echo = TRUE,
  arithmetic = c("double", "rational"),
  tol = 1e-10
)
```

Arguments

characteristic_func	The characteristic function of the game. It can be a numeric vector with the value of every coalition, in the order given by coalitions: of length $2^n - 1$, without the empty coalition (whose value is taken as 0), or of length 2^n, with the empty coalition first. Vectors support up to 30 players. It can also be a function that receives a coalition (a vector of players) and returns its value; in that case there is no limit on the number of players.
n_players	Only used if characteristic_func is a function. The number of players in the game.
method	Method used to calculate the core. Valid methods are: exact for the exact calculation or appro for approximated core based on Camacho et al. (2025).
n_rep	Only used if method is appro. The number of iterations to perform in the approximated calculation.
echo	Only used if method is appro. Show progress of the approximated calculation.
arithmetic	Only used if method is exact. Arithmetic used in the vertex enumeration: "double" (floating point) or "rational" (exact rational arithmetic).
tol	Only used if method is exact. Positive tolerance used in floating point comparisons.

Value

The centroid of the core if it exists.

References

Camacho, J., Gonçalves-Dosantos, J. C., & Sánchez-Soriano, J. (2025). A Linear Programming Approach to Estimate the Core in Cooperative Games. arXiv preprint arXiv:2510.01766.

Examples

```
v <- c(2,3,5,5,7,8,10)
centroid(v, method = "exact")
centroid(v, method = "appro", n_rep = 100)
```

```
n <- 3
v <- function(coalition) {
 size <- length(coalition)
 if (size <= 1) {
   return(0)
 } else if (size == 2) {
   return(10)
 } else if (size == 3) {
   return(24)
 } else {
   return(0)
 }
}
centroid(v, n, method = "exact")
centroid(v, n, method = "appro", n_rep = 200)
```

`check_core` *Core membership of an allocation*

Description

Check whether an allocation belongs to the core of the game, that is, whether it is efficient (the players share exactly $v(N)$) and coalitionally rational (every coalition S receives at least $v(S)$).

Usage

```
check_core(allocation, characteristic_func, details = FALSE)
```

Arguments

`allocation`	Numeric vector with the payoff of each player. Its length is the number of players of the game.
`characteristic_func`	The characteristic function of the game. It can be a numeric vector with the value of every coalition, in the order given by `coalitions`: of length $2^n - 1$, without the empty coalition (whose value is taken as 0), or of length 2^n, with the empty coalition first. Vectors support up to 30 players. It can also be a function that receives a coalition (a vector of players) and returns its value; in that case there is no limit on the number of players.
`details`	If `FALSE`, only `TRUE` or `FALSE` is returned and the check stops at the first violated constraint. If `TRUE`, every constraint is checked and reported.

Details

All comparisons are exact: integer allocations are summed exactly in double precision and any other allocation is handled with exact rational arithmetic (package `rcdd`). Values are interpreted as the binary numbers stored by R. The value of the empty coalition is taken as 0.

When `characteristic_func` is a function, there is no limit on the number of players, but up to 2^n coalitions must be evaluated.

Value

If `details` is `FALSE`, `TRUE` if the allocation belongs to the core and `FALSE` otherwise. If `details` is `TRUE`, a list with:

`in_core`	Whether the allocation belongs to the core.
`efficient`	Whether the allocation is efficient.
`coalitionally_rational`	Whether every coalition receives at least its value.
`violations`	The rows of `constraints` that are not satisfied.
`constraints`	A `data.frame` with one row per coalition (in the order of `coalitions`): the payoff assigned to the coalition, its value, the slack, the relation required and whether it is satisfied.

Examples

```
v <- c(0, 0, 0, 2, 2, 2, 3)
check_core(c(1, 1, 1), v)
check_core(c(0, 1, 2), v)
check_core(c(0, 1, 2), v, details = TRUE)

v <- function(coalition) length(coalition)^2
check_core(rep(4, 4), v)
```

`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

Examples

```
# Order in which the values of a game given as a vector must be supplied
coalitions(3)
```

`communication_game` *Communication game*

Description

Build the communication (graph-restricted) game of a communication situation, in which only connected players can cooperate (Myerson, 1977).

Usage

```
communication_game(characteristic_func, graph_edges, n_players = 0)
```

Arguments

`characteristic_func`
: The characteristic function of the game. It can be a numeric vector with the value of every coalition, in the order given by `coalitions`: of length $2^n - 1$, without the empty coalition (whose value is taken as 0), or of length 2^n, with the empty coalition first. Vectors support up to 30 players. It can also be a function that receives a coalition (a vector of players) and returns its value; in that case there is no limit on the number of players.

`graph_edges`
: Edges of the communication graph of the game: a `list` of pairs of players (for example `list(c(1, 2), c(2, 3))`), or a two-column matrix or data frame. Each edge must join two different players.

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

Details

The worth of a coalition S in the communication game is the sum of the worths of the connected components of the subgraph induced by S:

$$v^G(S) = \sum_{C \in C^G(S)} v(C).$$

The Myerson value is the Shapley value of this game, so `shapley(communication_game(v, graph_edges))` equals `myerson(v, graph_edges)`.

The value of the empty coalition is taken as 0.

Value

If `characteristic_func` is a vector, a vector with the value of every nonempty coalition of the communication game, in the order given by `coalitions`. If `characteristic_func` is a function, a function that receives a coalition (a vector of players) and returns its value in the communication game. It evaluates `characteristic_func` only on the connected components of the coalitions it receives, so there is no limit on the number of players, and it can be passed directly to the other functions of the package.

References

Myerson, R. B. (1977). Graphs and Cooperation in Games. Mathematics of Operations Research, 2(3), 225–229. doi:10.1287/moor.2.3.225

See Also

myerson, myerson_unions

Examples

```
v <- c(1, 2, 3, 10, 20, 30, 100)
communication_game(v, graph_edges = list(c(1, 3)))

v <- function(coalition) length(coalition)^2
graph_edges <- list(c(1, 2), c(2, 3))
vG <- communication_game(v, graph_edges, n_players = 4)
vG(c(1, 2, 4))
shapley(vG, n_players = 4)
myerson(v, graph_edges, n_players = 4)
```

---

coreVertex *Vertices of the core of the game*

Description

Calculate the vertices of the core of the game if it exists.

Usage

```
coreVertex(
  characteristic_func,
  n_players = 0,
  method = "exact",
  n_rep = 500,
  echo = TRUE,
  arithmetic = c("double", "rational"),
  tol = 1e-10
)
```

Arguments

characteristic_func

The characteristic function of the game. It can be a numeric vector with the value of every coalition, in the order given by coalitions: of length $2^n - 1$, without the empty coalition (whose value is taken as 0), or of length 2^n, with the empty coalition first. Vectors support up to 30 players. It can also be a function that receives a coalition (a vector of players) and returns its value; in that case there is no limit on the number of players.

n_players	Only used if characteristic_func is a function. The number of players in the game.
method	Method used to calculate the core. Valid methods are: exact for the exact calculation or appro for approximated core based on Camacho et al. (2025).
n_rep	Only used if method is appro. The number of iterations to perform in the approximated calculation.
echo	Only used if method is appro. Show progress of the approximated calculation.
arithmetic	Only used if method is exact. Arithmetic used in the vertex enumeration: "double" (floating point) or "rational" (exact rational arithmetic).
tol	Only used if method is exact. Positive tolerance used in floating point comparisons.

Value

The vertices of the core if it exists.

References

Camacho, J., Gonçalves-Dosantos, J. C., & Sánchez-Soriano, J. (2025). A Linear Programming Approach to Estimate the Core in Cooperative Games. arXiv preprint arXiv:2510.01766.

Examples

```
v <- c(2,3,5,5,7,8,10)
coreVertex(v, method = "exact")
coreVertex(v, method = "appro", n_rep = 100)

n <- 3
v <- function(coalition) {
 size <- length(coalition)
 if (size <= 1) {
   return(0)
 } else if (size == 2) {
   return(10)
 } else if (size == 3) {
   return(24)
 } else {
   return(0)
 }
}
coreVertex(v, n, method = "exact")
coreVertex(v, n, method = "appro", n_rep = 200)
```

`egalitarian` *Egalitarian rule*

Description

Calculate the egalitarian rule

Usage

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

Arguments

`characteristic_func`	The characteristic function of the game. It can be a numeric vector with the value of every coalition, in the order given by `coalitions`: of length $2^n - 1$, without the empty coalition (whose value is taken as 0), or of length 2^n, with the empty coalition first. Vectors support up to 30 players. It can also be a function that receives a coalition (a vector of players) and returns its value; in that case there is no limit on the number of players.
`n_players`	Only used if `characteristic_func` is a `function`. The number of players in the game.

Value

The egalitarian rule for each player

Examples

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

v <- c(1,1,2,1,2,2,2)
egalitarian(v)
```

egalitarian_unions *Egalitarian rule with a priori unions*

Description

Calculate the egalitarian rule in games with a priori unions

Usage

```
egalitarian_unions(characteristic_func, union, n_players = 0)
```

Arguments

characteristic_func	The characteristic function of the game. It can be a numeric vector with the value of every coalition, in the order given by coalitions: of length $2^n - 1$, without the empty coalition (whose value is taken as 0), or of length 2^n, with the empty coalition first. Vectors support up to 30 players. It can also be a function that receives a coalition (a vector of players) and returns its value; in that case there is no limit on the number of players.
union	List of vectors indicating the a priori unions between the players.
n_players	Only used if characteristic_func is a function. The number of players in the game.

Value

The egalitarian rule for each player

Examples

```
n <- 10
v <- function(coalition) {
  if (length(coalition) > n/2) {
    return(1)
  } else {
    return(0)
  }
}
union <- list(1:4,5:n)
egalitarian_unions(v,union,n)

v <- c(1,1,2,1,2,2,2)
union <- list(c(1,2),c(3))
egalitarian_unions(v, union)
```

`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 characteristic function of the game. It can be a numeric vector with the value of every coalition, in the order given by `coalitions`: of length $2^n - 1$, without the empty coalition (whose value is taken as 0), or of length 2^n, with the empty coalition first. Vectors support up to 30 players. It can also be a function that receives a coalition (a vector of players) and returns its value; in that case there is no limit on 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)

v <- c(1,1,2,1,2,2,2)
equal_surplus_division(v)
```

`equal_surplus_division_unions` *Equal Surplus Division value with a priori unions*

Description

Calculate the equal surplus division value in games with a priori unions

Usage

```
equal_surplus_division_unions(
  characteristic_func,
  union,
  n_players = 0,
  type = 1
)
```

Arguments

`characteristic_func`	The characteristic function of the game. It can be a numeric vector with the value of every coalition, in the order given by `coalitions`: of length $2^n - 1$, without the empty coalition (whose value is taken as 0), or of length 2^n, with the empty coalition first. Vectors support up to 30 players. It can also be a function that receives a coalition (a vector of players) and returns its value; in that case there is no limit on the number of players.
`union`	List of vectors indicating the a priori unions between the players.
`n_players`	Only used if `characteristic_func` is a `function`. The number of players in the game.
`type`	Number indicating the type of equal surplus division value to compute following Alonso-Meijide et al. (2020). Values 1, 2 and 3 are implemented.

Value

The equal surplus division value for each player

References

Alonso-Meijide, J. M., Costa, J., García-Jurado, I., & Gonçalves-Dosantos, J. C. (2020). On egalitarian values for cooperative games with a priori unions. Top, 28(3), 672-688.

Examples

```
n <- 3
v <- function(coalition) {
 size <- length(coalition)
 if (size <= 1) {
```

```
    return(0)
  } else if (size == 2) {
    return(10)
  } else if (size == 3) {
    return(24)
  } else {
    return(0)
  }
}
union <- list(1:2,3)
equal_surplus_division_unions(v,union,n,type = 1)

v <- c(1,1,2,1,2,2,2)
union <- list(c(1,2),c(3))
equal_surplus_division_unions(v, union, type = 2)
```

`myerson` *Myerson value*

Description

Calculate the Myerson value in a communication game.

Usage

```
myerson(
  characteristic_func,
  graph_edges,
  n_players = 0,
  method = "exact",
  n_rep = 1000,
  echo = TRUE
)
```

Arguments

`characteristic_func`
: The characteristic function of the game. It can be a numeric vector with the value of every coalition, in the order given by `coalitions`: of length $2^n - 1$, without the empty coalition (whose value is taken as 0), or of length 2^n, with the empty coalition first. Vectors support up to 30 players. It can also be a function that receives a coalition (a vector of players) and returns its value; in that case there is no limit on the number of players.

`graph_edges`
: Edges of the communication graph of the game: a `list` of pairs of players (for example `list(c(1, 2), c(2, 3))`), or a two-column matrix or data frame. Each edge must join two different players.

n_players	Only used if characteristic_func is a function. The number of players in the game.
method	Method used to calculate the Myerson value. Valid methods are: exact for the exact calculation or appro for approximated polynomial calculation, applying to the communication game the sampling procedure of the Shapley value proposed by Castro et al. (2009).
n_rep	Only used if method is appro. The number of iterations to perform in the approximated calculation.
echo	Only used if method is appro. Show progress of the approximated calculation.

Value

The Myerson value for each player.

References

Myerson, R. B. (1977). Graphs and Cooperation in Games. Mathematics of Operations Research, 2(3), 225–229. doi:10.1287/moor.2.3.225

Castro, J., Gómez, D., & Tejada, J. (2009). Polynomial calculation of the Shapley value based on sampling. Computers & operations research, 36(5), 1726-1730.

Examples

```
characteristic_func <- c(
1, 2, 0, 3,
3, 1, 4, 2, 5, 3,
3, 6, 4, 5,
15
)
graph_edges <- list(c(1, 2), c(2, 4))
myerson(characteristic_func, graph_edges, method = "exact")
myerson(characteristic_func, graph_edges, method = "appro", n_rep = 1000)

v <- function(S) {
  if (length(S) == 2) {
    return(1)
   }
   return(0)
}
n <- 3
graph_edges <- list(c(1, 2))
myerson(v, graph_edges, n_players = n, method = "exact")
myerson(v, graph_edges, n_players = n, method = "appro", n_rep = 2000)
```

`myerson_unions` *Myerson value with a priori unions*

Description

Calculate the Myerson value in a communication game with a priori unions.

Usage

```
myerson_unions(
  characteristic_func,
  n_players = 0,
  union,
  graph_edges,
  method = "exact",
  n_rep = 1000,
  echo = TRUE
)
```

Arguments

`characteristic_func`	The characteristic function of the game. It can be a numeric vector with the value of every coalition, in the order given by `coalitions`: of length $2^n - 1$, without the empty coalition (whose value is taken as 0), or of length 2^n, with the empty coalition first. Vectors support up to 30 players. It can also be a function that receives a coalition (a vector of players) and returns its value; in that case there is no limit on the number of players.
`n_players`	Only used if `characteristic_func` is a `function`. The number of players in the game.
`union`	List of vectors indicating the a priori unions between the players.
`graph_edges`	Edges of the communication graph of the game: a `list` of pairs of players (for example `list(c(1, 2), c(2, 3))`), or a two-column matrix or data frame. Each edge must join two different players.
`method`	Method used to calculate the Myerson value. Valid methods are: `exact` for the exact calculation or `appro` for approximated polynomial calculation, applying to the communication game the sampling procedure of the Owen value proposed by Saavedra-Nieves et al. (2018).
`n_rep`	Only used if `method` is `appro`. The number of iterations to perform in the approximated calculation.
`echo`	Only used if `method` is `appro`. Show progress of the approximated calculation.

Value

The Myerson value for each player.

References

Myerson, R. B. (1977). Graphs and Cooperation in Games. Mathematics of Operations Research, 2(3), 225–229. doi:10.1287/moor.2.3.225

Saavedra-Nieves, A., García-Jurado, I., & Fiestras-Janeiro, M. G. (2018). Estimation of the Owen value based on sampling. In The mathematics of the uncertain: A tribute to Pedro Gil (pp. 347-356). Cham: Springer International Publishing.

Examples

```
v <- c(
  0, 0, 0, 0,
  1, 1, 1, 0, 0, 0,
  1, 1, 1, 1,
  1
)
graph_edges <- list(c(2,3),c(3,1),c(1,4))
unions <- list(c(2,3),c(1),c(4))
myerson_unions(v, union = unions, graph_edges = graph_edges, method = "exact")
myerson_unions(v, union = unions, graph_edges = graph_edges, method = "appro", n_rep = 2000)
```

---

owen *Owen value*

Description

Calculate the Owen value

Usage

```
owen(
  characteristic_func,
  union,
  n_players = 0,
  method = "exact",
  n_rep = 1000,
  echo = TRUE
)
```

Arguments

`characteristic_func`

The characteristic function of the game. It can be a numeric vector with the value of every coalition, in the order given by `coalitions`: of length $2^n - 1$, without the empty coalition (whose value is taken as 0), or of length 2^n, with the empty coalition first. Vectors support up to 30 players. It can also be a function that receives a coalition (a vector of players) and returns its value; in that case there is no limit on the number of players.

union	List of vectors indicating the a priori unions between the players.
n_players	Only used if characteristic_func is a function. The number of players in the game.
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 proposed by Saavedra-Nieves et al. (2018).
n_rep	Only used if method is appro. The number of iterations to perform in the approximated calculation.
echo	Only used if method is appro. Show progress of the approximated calculation.

Value

The Owen value for each player.

References

Saavedra-Nieves, A., García-Jurado, I., & Fiestras-Janeiro, M. G. (2018). Estimation of the Owen value based on sampling. In The mathematics of the uncertain: A tribute to Pedro Gil (pp. 347-356). Cham: Springer International Publishing.

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)
```

---

shapley *Shapley value*

Description

Calculate the Shapley value

Usage

```
shapley(
  characteristic_func,
  n_players = 0,
  method = "exact",
  n_rep = 1000,
  echo = TRUE
)
```

Arguments

characteristic_func	The characteristic function of the game. It can be a numeric vector with the value of every coalition, in the order given by `coalitions`: of length $2^n - 1$, without the empty coalition (whose value is taken as 0), or of length 2^n, with the empty coalition first. Vectors support up to 30 players. It can also be a function that receives a coalition (a vector of players) and returns its value; in that case there is no limit on the number of players.
n_players	Only used if `characteristic_func` is a `function`. The number of players in the game.
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 proposed by Castro et al. (2009).
n_rep	Only used if `method` is `appro`. The number of iterations to perform in the approximated calculation.
echo	Only used if `method` is `appro`. Show progress of the approximated calculation.

Value

The Shapley value for each player.

References

Castro, J., Gómez, D., & Tejada, J. (2009). Polynomial calculation of the Shapley value based on sampling. Computers & operations research, 36(5), 1726-1730.

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)
```

Index

banzhaf, 3
banzhaf_owen, 4

centroid, 5
check_core, 7
coalitions, *3*, *5–8*, 8, *9, 10*, *12–16*, *18, 19*, *21*
communication_game, 9
coreVertex, 10

egalitarian, 12
egalitarian_unions, 13
equal_surplus_division, 14
equal_surplus_division_unions, 15

myerson, *10*, 16
myerson_unions, *10*, 18

owen, 19

shapley, 20

TUvalues *(*TUvalues-package*)*, 2
TUvalues-package, 2