

# Package ‘CoopGame’

October 12, 2022

**Type** Package

**Title** Important Concepts of Cooperative Game Theory

**Version** 0.2.2

**Maintainer** Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**Description** The theory of cooperative games with transferable utility offers useful insights into the way parties can share gains from cooperation and secure sustainable agreements, see e.g. one of the books by Chakravarty, Mitra and Sarkar (2015, ISBN:978-1107058798) or by Driessen (1988, ISBN:978-9027727299) for more details. A comprehensive set of tools for cooperative game theory with transferable utility is provided. Users can create special families of cooperative games, like e.g. bankruptcy games, cost sharing games and weighted voting games. There are functions to check various game properties and to compute five different set-valued solution concepts for cooperative games. A large number of point-valued solution concepts is available reflecting the diverse application areas of cooperative game theory. Some of these point-valued solution concepts can be used to analyze weighted voting games and measure the influence of individual voters within a voting body. There are routines for visualizing both set-valued and point-valued solutions in the case of three or four players.

**License** GPL-2

**Encoding** UTF-8

**Depends** R (>= 2.10.0), utils, geometry (>= 0.3-6), rcmd (>= 1.1)

**Imports** gtools (>= 3.5.0), methods (>= 3.3.1)

**Suggests** testthat, knitr, rmarkdown, rgl

**RoxygenNote** 7.1.1

**VignetteBuilder** knitr, rmarkdown

**NeedsCompilation** no

**Author** Jochen Staudacher [aut, cre, cph],  
Johannes Anwander [aut, cph],  
Alexandra Tiukkel [aut, cph],  
Michael Maerz [aut, cph],

Franz Mueller [aut, cph],  
 Daniel Gebele [aut, cph],  
 Anna Merkle [aut, cph],  
 Fatma Tokay [aut, cph],  
 Kuebra Tokay [aut, cph],  
 Nicole Cyl [aut, cph]

**Repository** CRAN

**Date/Publication** 2021-08-23 18:40:09 UTC

## R topics documented:

absolutePublicGoodValue . . . . .	6
absolutePublicHelpChiValue . . . . .	7
absolutePublicHelpValue . . . . .	8
apexGame . . . . .	9
apexGameValue . . . . .	10
apexGameVector . . . . .	11
bankruptcyGame . . . . .	12
bankruptcyGameValue . . . . .	13
bankruptcyGameVector . . . . .	14
banzhafValue . . . . .	16
baruaChakravartySarkarIndex . . . . .	17
belongsToCore . . . . .	18
belongsToCoreCover . . . . .	19
belongsToImputationset . . . . .	20
belongsToReasonableSet . . . . .	21
belongsToWeberset . . . . .	22
cardinalityGame . . . . .	23
cardinalityGameValue . . . . .	24
cardinalityGameVector . . . . .	25
centroidCore . . . . .	25
centroidCoreCover . . . . .	26
centroidImputationSet . . . . .	27
centroidReasonableSet . . . . .	28
centroidWeberSet . . . . .	29
colemanCollectivityPowerIndex . . . . .	30
colemanInitiativePowerIndex . . . . .	31
colemanPreventivePowerIndex . . . . .	32
coreCoverVertices . . . . .	33
coreVertices . . . . .	34
costSharingGame . . . . .	35
costSharingGameValue . . . . .	36
costSharingGameVector . . . . .	37
createBitMatrix . . . . .	39
deeganPackelIndex . . . . .	40
dictatorGame . . . . .	41
dictatorGameValue . . . . .	42

dictatorGameVector . . . . .	43
disruptionNucleolus . . . . .	44
divideTheDollarGame . . . . .	45
divideTheDollarGameValue . . . . .	46
divideTheDollarGameVector . . . . .	47
drawCentroidCore . . . . .	48
drawCentroidCoreCover . . . . .	49
drawCentroidImputationSet . . . . .	50
drawCentroidReasonableSet . . . . .	51
drawCentroidWeberSet . . . . .	52
drawCore . . . . .	53
drawCoreCover . . . . .	54
drawDeeganPackelIndex . . . . .	55
drawDisruptionNucleolus . . . . .	56
drawGatelyValue . . . . .	57
drawImputationset . . . . .	58
drawJohnstonIndex . . . . .	59
drawModiclus . . . . .	60
drawNormalizedBanzhafIndex . . . . .	61
drawNormalizedBanzhafValue . . . . .	62
drawNucleolus . . . . .	63
drawPerCapitaNucleolus . . . . .	64
drawPrenucleolus . . . . .	66
drawProportionalNucleolus . . . . .	67
drawPublicGoodIndex . . . . .	68
drawPublicGoodValue . . . . .	69
drawPublicHelpChiIndex . . . . .	70
drawPublicHelpChiValue . . . . .	71
drawPublicHelpIndex . . . . .	72
drawPublicHelpValue . . . . .	73
drawReasonableSet . . . . .	74
drawShapleyShubikIndex . . . . .	75
drawShapleyValue . . . . .	76
drawSimplifiedModiclus . . . . .	77
drawTauValue . . . . .	78
drawWeberset . . . . .	79
equalPropensityToDisrupt . . . . .	80
gatelyValue . . . . .	81
getCriticalCoalitionsOfPlayer . . . . .	82
getDualGameVector . . . . .	84
getEmptyParamCheckResult . . . . .	85
getExcessCoefficients . . . . .	86
getGainingCoalitions . . . . .	87
getGapFunctionCoefficients . . . . .	88
getkCover . . . . .	88
getMarginalContributions . . . . .	89
getMinimalRights . . . . .	90
getMinimumWinningCoalitions . . . . .	91

getNumberOfPlayers . . . . .	92
getPerCapitaExcessCoefficients . . . . .	93
getPlayersFromBitVector . . . . .	94
getPlayersFromBMRow . . . . .	94
getRealGainingCoalitions . . . . .	95
getUnanimityCoefficients . . . . .	96
getUtopiaPayoff . . . . .	97
getVectorOfPropensitiesToDisrupt . . . . .	98
getWinningCoalitions . . . . .	99
getZeroNormalizedGameVector . . . . .	100
getZeroOneNormalizedGameVector . . . . .	101
gloveGame . . . . .	102
gloveGameValue . . . . .	103
gloveGameVector . . . . .	104
imputationsetVertices . . . . .	105
is1ConvexGame . . . . .	106
isAdditiveGame . . . . .	107
isBalancedGame . . . . .	108
isConstantSumGame . . . . .	109
isConvexGame . . . . .	110
isDegenerateGame . . . . .	111
isEssentialGame . . . . .	112
isConvexGame . . . . .	113
isMonotonicGame . . . . .	115
isNonnegativeGame . . . . .	116
isQuasiBalancedGame . . . . .	117
isSemiConvexGame . . . . .	118
isSimpleGame . . . . .	119
isSuperadditiveGame . . . . .	120
isSymmetricGame . . . . .	121
isWeaklyConstantSumGame . . . . .	122
isWeaklySuperadditiveGame . . . . .	123
johnstonIndex . . . . .	124
koenigBraeuningerIndex . . . . .	125
majoritySingleVetoGame . . . . .	126
majoritySingleVetoGameValue . . . . .	127
majoritySingleVetoGameVector . . . . .	128
modiclus . . . . .	129
nevisonIndex . . . . .	130
nonNormalizedBanzhafIndex . . . . .	131
normalizedBanzhafIndex . . . . .	132
normalizedBanzhafValue . . . . .	133
nucleolus . . . . .	134
perCapitaNucleolus . . . . .	135
PreNucleolus . . . . .	136
propensityToDisrupt . . . . .	138
proportionalNucleolus . . . . .	139
publicGoodIndex . . . . .	140

publicGoodValue . . . . .	141
publicHelpChiIndex . . . . .	142
publicHelpChiValue . . . . .	143
publicHelpIndex . . . . .	144
publicHelpValue . . . . .	145
raeIndex . . . . .	146
rawBanzhafIndex . . . . .	147
rawBanzhafValue . . . . .	148
reasonableSetVertices . . . . .	149
shapleyShubikIndex . . . . .	150
shapleyValue . . . . .	151
simplifiedModiclus . . . . .	152
stopOnInconsistentEstateAndClaimsVector . . . . .	153
stopOnInvalidAllocation . . . . .	154
stopOnInvalidBoolean . . . . .	155
stopOnInvalidClaimsVector . . . . .	156
stopOnInvalidCoalitionS . . . . .	157
stopOnInvalidDictator . . . . .	158
stopOnInvalidEstate . . . . .	159
stopOnInvalidGameVector . . . . .	160
stopOnInvalidGrandCoalitionN . . . . .	162
stopOnInvalidIndex . . . . .	163
stopOnInvalidLeftRightGloveGame . . . . .	164
stopOnInvalidNChooseB . . . . .	165
stopOnInvalidNumber . . . . .	166
stopOnInvalidNumberOfPlayers . . . . .	167
stopOnInvalidQuota . . . . .	168
stopOnInvalidVetoPlayer . . . . .	169
stopOnInvalidWeightVector . . . . .	170
stopOnParamCheckError . . . . .	171
tauValue . . . . .	172
unanimityGame . . . . .	173
unanimityGameValue . . . . .	174
unanimityGameVector . . . . .	175
webersetVertices . . . . .	176
weightedVotingGame . . . . .	177
weightedVotingGameValue . . . . .	178
weightedVotingGameVector . . . . .	179

---

`absolutePublicGoodValue`*Compute absolute Public Good value*

---

**Description**

`absolutePublicGoodValue` calculates the absolute Public Good value for a specified nonnegative TU game. Note that in general the absolute Public Good value is not an efficient vector, i.e. the sum of its entries is not always 1.

**Usage**

```
absolutePublicGoodValue(v)
```

**Arguments**

`v` Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with  $n$  players

**Value**

Absolute Public Good value for specified nonnegative game

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**References**

Holler M.J. and Li X. (1995) "From public good index to public value. An axiomatic approach and generalization", *Control and Cybernetics* 24, pp. 257–270

Bertini C. and Stach I. (2015) "On Public Values and Power Indices", *Decision Making in Manufacturing and Services* 9(1), pp. 9–25

**Examples**

```
library(CoopGame)
v <- c(1,2,3,4,0,0,0)
absolutePublicGoodValue(v)
```

```
v=c(0,0,0,0.7,11,0,15)
absolutePublicGoodValue(v)
#[1] 26.7 15.7 26.0
```

---

`absolutePublicHelpChiValue`*Compute absolute Public Help value Chi*

---

**Description**

Calculates the absolute Public Help value Chi for a specified nonnegative TU game. Note that in general the absolute Public Help value Chi is not an efficient vector, i.e. the sum of its entries is not always 1. Hence no drawing routine for the absolute Public Help value Chi is provided. Note that the greek letter Xi (instead of Chi) was used in the original paper by Bertini and Stach (2015).

**Usage**`absolutePublicHelpChiValue(v)`**Arguments**

`v` Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with  $n$  players

**Value**

Absolute Public Help value Chi for specified nonnegative game

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**References**

Bertini C. and Stach I. (2015) "On Public Values and Power Indices", Decision Making in Manufacturing and Services 9(1), pp. 9–25

**Examples**

```
library(CoopGame)
v=c(0,0,0,2,2,0,2)
absolutePublicHelpChiValue(v)
```

---

`absolutePublicHelpValue`*Compute absolute Public Help value Theta*

---

**Description**

`absolutePublicHelpValue` calculates the absolute Public Help value Theta for a specified nonnegative TU game. Note that in general the absolute Public Help value Theta is not an efficient vector, i.e. the sum of its entries is not always 1. Hence no drawing routine for the absolute Public Help Value is provided.

**Usage**`absolutePublicHelpValue(v)`**Arguments**

`v` Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with  $n$  players

**Value**

Absolute Public Help value Theta for specified simple game

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**References**

Bertini C. and Stach I. (2015) "On Public Values and Power Indices", *Decision Making in Manufacturing and Services* 9(1), pp. 9–25

**Examples**

```
library(CoopGame)
v=c(0,0,0,0.7,11,0,15)
absolutePublicHelpValue(v)
```



---

apexGame	<i>Construct an apex game</i>
----------	-------------------------------

---

### Description

**Create a list containing all information about a specified apex game:**

A coalition can only win (and hence obtain the value 1) if it

a) contains both the apex player and one additional player

or

b) contains all players except for the apex player.

Any non-winning coalitions obtain the value 0.

Note that apex games are always simple games.

### Usage

```
apexGame(n, apexPlayer)
```

### Arguments

n represents the number of players

apexPlayer specifies the number of the apex player

### Value

A list with three elements representing the apex game (n, apexPlayer, Game vector v)

### Related Functions

[apexGameValue](#), [apexGameVector](#)

### Author(s)

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

Johannes Anwander <anwander.johannes@gmail.com>

### References

Peters H. (2015) *Game Theory: A Multi-Leveled Approach*, 2nd Edition, Springer, pp. 164–165

### Examples

```
library(CoopGame)
apexGameVector(n=3, apexPlayer=2)
```

```
library(CoopGame)
#Example with four players, apex player is number 3
(vv<-apexGame(n=4, apexPlayer=3))
```

```

#$n
#[1] 4

#$apexPlayer
#[1] 4

#$v
# [1] 0 0 0 0 0 1 0 1 0 1 1 1 1 1 1

```

---

apexGameValue	<i>Compute value of a coalition for an apex game</i>
---------------	--

---

### Description

**Coalition value for an apex game:**  
 For further information see [apexGame](#)

### Usage

```
apexGameValue(S, n, apexPlayer)
```

### Arguments

S	numeric vector with coalition of players
n	represents the number of players
apexPlayer	specifies the number of the apex player

### Value

value of coalition S

### Author(s)

Alexandra Tiukkel  
 Jochen Staudacher <jochen.staudacher@hs-kempten.de>

### References

Peters H. (2015) *Game Theory: A Multi-Leveled Approach*, 2nd Edition, Springer, pp. 164–165

**Examples**

```
library(CoopGame)
apexGameValue(c(1,2),3,2)
```

```
library(CoopGame)
apexGameValue(c(1,2,3,4),4,3)
# Output:
# [1] 1
```

---

apexGameVector	<i>Compute game vector for an apex game</i>
----------------	---

---

**Description**

**Game vector for an apex game:**  
For further information see [apexGame](#)

**Usage**

```
apexGameVector(n, apexPlayer)
```

**Arguments**

n	represents the number of players
apexPlayer	specifies the number of the apex player

**Value**

Game vector for the apex game

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**References**

Peters H. (2015) *Game Theory: A Multi-Leveled Approach*, 2nd Edition, Springer, pp. 164–165

**Examples**

```
library(CoopGame)
apexGameVector(n=3, apexPlayer=2)
```

```
library(CoopGame)
(v <- apexGameVector(n=4, apexPlayer=3))
#[1] 0 0 0 0 0 1 0 1 0 1 1 1 1 1
```

---

 bankruptcyGame

*Construct a bankruptcy game*


---

**Description**

**Create a list containing all information about a specified bankruptcy game:**

The list contains the number of players, the claims vector, the estate and the bankruptcy game vector. Bankruptcy games are defined by a vector of debts  $d$  of  $n$  creditors (players) and an estate  $E$  less than the sum of the debt vector. The roots of bankruptcy games can be traced back to the Babylonian Talmud.

**Usage**

```
bankruptcyGame(n, d, E)
```

**Arguments**

$n$	represents the number of players
$d$	numeric vector which contains the claims of each player in a bankruptcy game
$E$	is the value of the estate in a bankruptcy game

**Value**

A list with four elements representing the specified bankruptcy game ( $n$ ,  $d$ ,  $E$ , Game vector  $v$ )

**Related Functions**

[bankruptcyGameValue](#), [bankruptcyGameVector](#)

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

## References

- O'Neill, B. (1982) "A problem of rights arbitration from the Talmud", *Mathematical Social Sciences* 4(2), pp. 345 – 371
- Aumann R.J. and Maschler M. (1985) "Game Theoretic Analysis of a Bankruptcy Problem from the Talmud", *Journal of Economic Theory* 36(1), pp. 195 – 213
- Aumann R.J. (2002) "Game Theory in the Talmud", *Research Bulletin Series on Jewish Law and Economics*, 12 pages.
- Gura E. and Maschler M. (2008) *Insights into Game Theory*, Cambridge University Press, pp. 166–204

## Examples

```
library(CoopGame)
bankruptcyGame(n=3, d=c(1,2,3), E=4)

#Estate division problem from Babylonian Talmud
#from paper by Aumann (2002) with E=300
library(CoopGame)
bankruptcyGame(n=3, d=c(100, 200, 300), E=300)
#Output
#$n
#[1] 3

#$d
#[1] 100 200 300

#$E
#[1] 300

#$v
#[1] 0 0 0 0 100 200 300
```

---

bankruptcyGameValue     *Compute value of a coalition for a bankruptcy game*

---

## Description

**Coalition value for a specified bankruptcy game:**  
 For further information see [bankruptcyGame](#)

## Usage

```
bankruptcyGameValue(S, d, E)
```

**Arguments**

S	numeric vector with coalition of players
d	numeric vector which contains the claims of each player in a bankruptcy game
E	is the value of the estate in a bankruptcy game

**Value**

A positive value if the sum of the claims outside of coalition S is less than E else 0

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**References**

O'Neill, B. (1982) "A problem of rights arbitration from the Talmud", *Mathematical Social Sciences* 4(2), pp. 345 – 371

Aumann R.J. and Maschler M. (1985) "Game Theoretic Analysis of a Bankruptcy Problem from the Talmud", *Journal of Economic Theory* 36(1), pp. 195 – 213

Aumann R.J. (2002) "Game Theory in the Talmud", *Research Bulletin Series on Jewish Law and Economics*, 12 pages.

Gura E. and Maschler M. (2008) *Insights into Game Theory*, Cambridge University Press, pp. 166–204

**Examples**

```
library(CoopGame)
bankruptcyGameValue(S=c(2,3),d=c(1,2,3),E=4)

#Estate division problem from Babylonian Talmud
#from paper by Aumann (2002) with E=300
library(CoopGame)
bankruptcyGameValue(S=c(2,3),d=c(100,200,300),E=300)
#Output
#[1] 200
```

---

bankruptcyGameVector    *Compute game vector for a bankruptcy game*

---

**Description**

**Game vector for a specified bankruptcy game:**

For further information see [bankruptcyGame](#)

**Usage**

```
bankruptcyGameVector(n, d, E)
```

**Arguments**

n	represents the number of players
d	numeric vector which contains the claims of each player in a bankruptcy game
E	is the value of the estate in a bankruptcy game

**Value**

Game Vector where each element contains a positive value if the sum of the claims outside of coalition 'S' is less than E else 0

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**References**

O'Neill, B. (1982) "A problem of rights arbitration from the Talmud", *Mathematical Social Sciences* 4(2), pp. 345 – 371

Aumann R.J. and Maschler M. (1985) "Game Theoretic Analysis of a Bankruptcy Problem from the Talmud", *Journal of Economic Theory* 36(1), pp. 195 – 213

Aumann R.J. (2002) "Game Theory in the Talmud", *Research Bulletin Series on Jewish Law and Economics*, 12 pages.

Gura E. and Maschler M. (2008) *Insights into Game Theory*, Cambridge University Press, pp. 166–204

**Examples**

```
library(CoopGame)
bankruptcyGameVector(n=3, d=c(1,2,3), E=4)

#Estate division problem from Babylonian Talmud
#from paper by Aumann (2002) with E=300
library(CoopGame)
bankruptcyGameVector(n=3,d=c(100,200,300),E=300)
#Output
#[1] 0 0 0 0 100 200 300
```

---

banzhafValue

*Compute Banzhaf value*


---

### Description

banzhafValue computes the Banzhaf value for a specified TU game. The Banzhaf value itself is an alternative to the Shapley value.

Conceptually, the Banzhaf value is very similar to the Shapley value. Its main difference from the Shapley value is that the Banzhaf value is coalition based rather than permutation based. Note that in general the Banzhaf vector is not efficient! In this sense this implementation of the Banzhaf value could also be referred to as the non-normalized Banzhaf value, see formula (20.6) in on p. 368 of the book by Hans Peters (2015).

### Usage

```
banzhafValue(v)
```

### Arguments

`v` Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with  $n$  players

### Value

The return value is a numeric vector which contains the Banzhaf value for each player.

### Author(s)

Johannes Anwander <anwander.johannes@gmail.com>

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

### References

Peters H. (2015) *Game Theory: A Multi-Leveled Approach*, 2nd Edition, Springer, pp. 367–370

Chakravarty S.R., Mitra M. and Sarkar P. (2015) *A Course on Cooperative Game Theory*, Cambridge University Press, pp. 118–119

Gambarelli G. (2011) "Banzhaf value", *Encyclopedia of Power*, SAGE Publications, pp. 53–54

### Examples

```
library(CoopGame)
v=c(0,0,0,1,2,1,4)
banzhafValue(v)
```

```
#Example from paper by Gambarelli (2011)
library(CoopGame)
```



```
v=c(0,0,0,1,2,1,3)
banzhafValue(v)
#[1] 1.25 0.75 1.25
```

---

baruaChakravartySarkarIndex

*Compute Barua Chakravarty Sarkar index*

---

### Description

Calculates the Barua Chakravarty Sarkar index for a specified simple TU game. Note that in general the Barua Chakravarty Sarkar index is not an efficient vector, i.e. the sum of its entries is not always 1. Hence no drawing routine for the Barua Chakravarty Sarkar index is provided.

### Usage

```
baruaChakravartySarkarIndex(v)
```

### Arguments

`v` Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with  $n$  players

### Value

Barua Chakravarty Sarkar index for specified simple game

### Author(s)

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

### References

Barua R., Chakravarty S.R. and Sarkar P. (2012) "Measuring p-power of voting", *Journal of Economic Theory and Social Development* 1(1), pp. 81–91

Chakravarty S.R., Mitra M. and Sarkar P. (2015) *A Course on Cooperative Game Theory*, Cambridge University Press, pp. 120–123

### Examples

```
library(CoopGame)
v=c(0,0,0,1,1,0,1)
baruaChakravartySarkarIndex(v)
```

---

belongsToCore                      *Check if point is core element*

---

### Description

belongsToCore checks if a given point is in the core

### Usage

```
belongsToCore(x, v)
```

### Arguments

x	numeric vector containing allocations for each player
v	Numeric vector of length $2^n - 1$ representing the values of the coalitions of a TU game with n players

### Value

TRUE for a point belonging to the core and FALSE otherwise

### Author(s)

Franz Mueller  
 Jochen Staudacher <jochen.staudacher@hs-kempten.de>

### References

Gillies D.B. (1953) *Some Theorems on n-person Games*, Ph.D. Thesis, Princeton University Press.  
 Aumann R.J. (1961) "The core of a cooperative game without side payments", Transactions of the American Mathematical Society 98(3), pp. 539–552  
 Peleg B. and Sudhoelter P. (2007) *Theory of cooperative games*, 2nd Edition, Springer, pp. 27–49  
 Maschler M., Solan E. and Zamir S. (2013) *Game Theory*, Cambridge University Press, pp. 686–747  
 Osborne M.J. and Rubinstein A. (1994) *A Course in Game Theory*, MIT Press, pp. 257–275

### Examples

```
library(CoopGame)
v = c(0,1,2,3,4,5,6)
belongsToCore(c(1,2,3),v)
```

---

belongsToCoreCover      *Check if point is core cover element*

---

### Description

belongsToCoreCover checks if the point is in the core cover

### Usage

```
belongsToCoreCover(x, v)
```

### Arguments

x	numeric vector containing allocations for each player
v	Numeric vector of length $2^n - 1$ representing the values of the coalitions of a TU game with n players

### Value

TRUE if point belongs to core cover, FALSE otherwise

### Author(s)

Johannes Anwander <anwander.johannes@gmail.com>

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

### References

Tijs S.H. and Lipperts F.A.S. (1982) "The hypercube and the core cover of n-person cooperative games", *Cahiers du Centre d' Etudes de Recherche Operationelle* 24, pp. 27–37

Branzei R., Dimitrov D. and Tijs S. (2006) *Models in cooperative game theory*, Springer, p. 21

Chakravarty S.R., Mitra M. and Sarkar P. (2015) *A Course on Cooperative Game Theory*, Cambridge University Press, pp. 45–46

### Examples

```
library(CoopGame)
belongsToCoreCover(x=c(1,1,1), v=c(0,0,0,1,1,1,3))
```

```
library(CoopGame)
v <- c(0,0,0,3,3,3,6)
belongsToCoreCover(x=c(2,2,2),v)
#[1] TRUE
belongsToCoreCover(x=c(1,2,4),v)
#[1] FALSE
```

---

belongsToImputationset

*Check if point is imputation*

---

### Description

belongsToImputationset checks if the point belongs to the imputation set

### Usage

```
belongsToImputationset(x, v)
```

### Arguments

x	numeric vector containing allocations for each player
v	Numeric vector of length $2^n - 1$ representing the values of the coalitions of a TU game with n players

### Value

TRUE for a point belonging to the imputation set and FALSE otherwise

### Author(s)

Johannes Anwander <anwander.johannes@gmail.com>

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

### References

Peleg B. and Sudhoelter P. (2007) *Theory of cooperative games*, 2nd Edition, Springer, p. 20  
Maschler M., Solan E. and Zamir S. (2013) *Game Theory*, Cambridge University Press, p. 674  
Osborne M.J. and Rubinstein A. (1994) *A Course in Game Theory*, MIT Press, p. 278  
Narahari Y. (2015) *Game Theory and Mechanism Design*, World Scientific Publishing, p. 407

### Examples

```
library(CoopGame)
belongsToImputationset(x=c(1,0.5,0.5), v=c(0,0,0,1,1,1,2))
```

```
library(CoopGame)
v=c(0,1,2,3,4,5,6)
```

```
#Point belongs to imputation set:
belongsToImputationset(x=c(1.5,1,3.5),v)
```

```
#Point does not belong to imputation set:
```

`belongsToImputationset(x=c(2.05,2,2),v)`

`belongsToReasonableSet`

*Check if point is element of reasonable set*

### **Description**

`belongsToReasonableSet` checks if the point is in the reasonable set

### **Usage**

`belongsToReasonableSet(x, v)`

### **Arguments**

- `x` numeric vector containing allocations for each player
- `v` Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with  $n$  players

### **Value**

TRUE if point belongs to reasonable set, FALSE otherwise

### **Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

### **References**

- Milnor J.W. (1953) *Reasonable Outcomes for N-person Games*, Rand Corporation, Research Memorandum RM 916.
- Branzei R., Dimitrov D. and Tijs S. (2006) *Models in cooperative game theory*, Springer, p. 21
- Chakravarty S.R., Mitra M. and Sarkar P. (2015) *A Course on Cooperative Game Theory*, Cambridge University Press, pp. 43–44
- Gerard-Varet L.A. and Zamir S. (1987) "Remarks on the reasonable set of outcomes in a general coalition function form game", *Int. Journal of Game Theory* 16(2), pp. 123–143

**Examples**

```
library(CoopGame)
belongsToReasonableSet(x=c(1,0.5,0.5), v=c(0,0,0,1,1,1,2))
```

```
library(CoopGame)
v <- c(0,0,0,3,3,3,6)
belongsToReasonableSet(x=c(2,2,2), v)
#[1] TRUE
belongsToReasonableSet(x=c(1,2,4), v)
#[1] FALSE
```

---

belongsToWeberset	<i>Check if point is element of Weber Set</i>
-------------------	---

---

**Description**

belongsToWeberset checks if the point is in the Weber Set

**Usage**

```
belongsToWeberset(x, v)
```

**Arguments**

x	numeric vector containing allocations for each player
v	Numeric vector of length $2^n - 1$ representing the values of the coalitions of a TU game with n players

**Value**

TRUE if point belongs to Weber Set and FALSE otherwise

**Author(s)**

Johannes Anwander <anwander.johannes@gmail.com>

**References**

Weber R.J. (1988) "Probabilistic values for games". In: Roth A.E. (Ed.), *The Shapley Value. Essays in honor of Lloyd S. Shapley*, Cambridge University Press, pp. 101–119

Peters H. (2015) *Game Theory: A Multi-Levelled Approach*, 2nd Edition, Springer, pp. 327–329

## Examples

```
library(CoopGame)
belongsToWeberset(x=c(1,0.5,0.5), v=c(0,0,0,1,1,1,2))
```

```
library(CoopGame)
v=c(0,1,2,3,4,5,6)
```

```
#Point belongs to Weber Set:
belongsToWeberset(x=c(1.5,1,3.5),v)
```

```
#Point does not belong to Weber Set:
belongsToWeberset(x=c(2.05,2,2),v)
```

---

cardinalityGame	<i>Construct a cardinality game</i>
-----------------	-------------------------------------

---

## Description

### **Create a list containing all information about a specified cardinality game:**

For a cardinality game the worth of each coalition is simply the number of the members of the coalition.

## Usage

```
cardinalityGame(n)
```

## Arguments

n represents the number of players

## Value

A list with two elements representing the cardinality game (n, Game vector v)

## Related Functions

[cardinalityGameValue](#), [cardinalityGameVector](#)

## Author(s)

Jochen Staudacher <jochen.staudacher@hs-kempten.de>  
Johannes Anwander <anwander.johannes@gmail.com>

**Examples**

```
library(CoopGame)
cardinalityGame(n=3)

library(CoopGame)
#Example: Cardinality function for four players
(vv<-cardinalityGame(n=4))
#$n
#[1] 4

#$v
#[1] 1 1 1 1 2 2 2 2 2 2 3 3 3 3 4
```

---

cardinalityGameValue *Compute value of a coalition for a cardinality game*

---

**Description**

**Coalition value for a cardinality game:**  
For further information see [cardinalityGame](#)

**Usage**

```
cardinalityGameValue(S)
```

**Arguments**

S                    numeric vector with coalition of players

**Value**

The return value is the cardinality, i.e. the number of elements, of coalition S

**Author(s)**

Johannes Anwander <anwander.johannes@gmail.com>  
Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**Examples**

```
library(CoopGame)
S=c(1,2,4,5)
cardinalityGameValue(S)
```



---

cardinalityGameVector *Compute game vector for a cardinality game*

---

**Description****Game vector for a cardinality game:**

For further information see [cardinalityGame](#)

**Usage**

```
cardinalityGameVector(n)
```

**Arguments**

n represents the number of players

**Value**

Game vector for the cardinality game

**Author(s)**

Johannes Anwander <anwander.johannes@gmail.com>

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**Examples**

```
library(CoopGame)
cardinalityGameVector(n=3)
```

```
library(CoopGame)
(v <- cardinalityGameVector(n=4))
#[1] 1 1 1 1 2 2 2 2 2 2 3 3 3 3 4
```

---

centroidCore *Compute centroid of core*

---

**Description**

Calculates the centroid of the core for specified game.

**Usage**

```
centroidCore(v)
```

**Arguments**

v                      Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with n players

**Value**

Calculates the centroid of the core for a game specified by a game vector v.

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**References**

- Gillies D.B. (1953) *Some Theorems on n-person Games*, Ph.D. Thesis, Princeton University Press.
- Aumann R.J. (1961) "The core of a cooperative game without side payments", *Transactions of the American Mathematical Society* 98(3), pp. 539–552
- Peleg B. and Sudhoelter P. (2007) *Theory of cooperative games*, 2nd Edition, Springer, pp. 27–49
- Maschler M., Solan E. and Zamir S. (2013) *Game Theory*, Cambridge University Press, pp. 686–747
- Osborne M.J. and Rubinstein A. (1994) *A Course in Game Theory*, MIT Press, pp. 257–275

**Examples**

```
library(CoopGame)
v=c(0,0,0,0,2,2,3,5)
centroidCore(v)
```

---

centroidCoreCover            *Compute centroid of the core cover*

---

**Description**

Calculates the centroid of the core cover for specified game.

**Usage**

```
centroidCoreCover(v)
```

**Arguments**

v                      Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with n players

**Value**

Centroid of the core cover for a quasi-balanced game specified by a game vector

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**References**

Tijs S.H. and Lipperts F.A.S. (1982) "The hypercube and the core cover of n-person cooperative games", Cahiers du Centre d' Etudes de Recherche Operationelle 24, pp. 27–37

Branzei R., Dimitrov D. and Tijs S. (2006) *Models in cooperative game theory*, Springer, p. 21

Chakravarty S.R., Mitra M. and Sarkar P. (2015) *A Course on Cooperative Game Theory*, Cambridge University Press, pp. 45–46

**Examples**

```
library(CoopGame)
v=c(0,0,0,2,2,3,5)
centroidCoreCover(v)
```

---

centroidImputationSet *Compute centroid of the imputation set*

---

**Description**

Calculates the centroid of the imputation set for specified game.

**Usage**

```
centroidImputationSet(v)
```

**Arguments**

**v** Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with  $n$  players

**Value**

Calculates the centroid of the imputation set for a game specified by a game vector.

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**References**

- Peleg B. and Sudhoelster P. (2007) *Theory of cooperative games*, 2nd Edition, Springer, p. 20
- Maschler M., Solan E. and Zamir S. (2013) *Game Theory*, Cambridge University Press, p. 674
- Osborne M.J. and Rubinstein A. (1994) *A Course in Game Theory*, MIT Press, p. 278
- Narahari Y. (2015) *Game Theory and Mechanism Design*, World Scientific Publishing, p. 407

**Examples**

```
library(CoopGame)
v=c(0,0,0,2,2,3,5)
centroidImputationSet(v)
```

---

centroidReasonableSet *Compute centroid of reasonable set*

---

**Description**

Calculates the centroid of the reasonable set for specified game.

**Usage**

```
centroidReasonableSet(v)
```

**Arguments**

v                      Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with n players

**Value**

Calculates the centroid of the reasonable set for a game specified by a game vector.

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**References**

- Milnor J.W. (1953) *Reasonable Outcomes for N-person Games*, Rand Corporation, Research Memorandum RM 916.
- Branzei R., Dimitrov D. and Tijs S. (2006) *Models in cooperative game theory*, Springer, p. 21
- Chakravarty S.R., Mitra M. and Sarkar P. (2015) *A Course on Cooperative Game Theory*, Cambridge University Press, pp. 43–44
- Gerard-Varet L.A. and Zamir S. (1987) "Remarks on the reasonable set of outcomes in a general coalition function form game", *Int. Journal of Game Theory* 16(2), pp. 123–143

**Examples**

```
library(CoopGame)
v=c(0,0,0,2,2,3,5)
centroidReasonableSet(v)
```

---

centroidWeberSet	<i>Compute centroid of Weber set</i>
------------------	--------------------------------------

---

**Description**

Calculates the centroid of the Weber set for specified game.

**Usage**

```
centroidWeberSet(v)
```

**Arguments**

**v** Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with  $n$  players

**Value**

Calculates the centroid of the Weber set for a game specified by a game vector.

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**References**

Weber R.J. (1988) "Probabilistic values for games". In: Roth A.E. (Ed.), *The Shapley Value. Essays in honor of Lloyd S. Shapley*, Cambridge University Press, pp. 101–119

Peters H. (2015) *Game Theory: A Multi-Leveled Approach*, 2nd Edition, Springer, pp. 327–329

**Examples**

```
library(CoopGame)
v=c(0,0,0,2,2,3,5)
centroidWeberSet(v)
```

---

`colemanCollectivityPowerIndex`*Compute Coleman Power index of a Collectivity to Act*

---

**Description**

Calculates the Coleman Power index of a Collectivity to Act for a specified simple TU game. Note that in general the Coleman Power index of a Collectivity to Act is not an efficient vector, i.e. the sum of its entries is not always 1. Note also that the the Coleman Power index of a Collectivity to Act is identical for each player, i.e. the result for each player is the number of winning coalitions divided by  $2^n$ . Hence no drawing routine for the Coleman Power index of a Collectivity to Act is provided.

**Usage**`colemanCollectivityPowerIndex(v)`**Arguments**

`v` Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with  $n$  players

**Value**

Coleman Power index of a Collectivity to Act for specified simple game

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**References**

Coleman J.S. (1971) "Control of collectivities and the power of a collectivity to act". In: Liberman B. (Ed.), *Social Choice*, Gordon and Breach, pp. 269–300

De Keijzer B. (2008) "A survey on the computation of power indices", Technical Report, Delft University of Technology, p. 18

Bertini C. and Stach I. (2011) "Coleman index", *Encyclopedia of Power*, SAGE Publications, p. 117–119

**Examples**

```
library(CoopGame)
v=c(0,0,0,0,1,1,0,1)
colemanCollectivityPowerIndex(v)
```

---

`colemanInitiativePowerIndex`*Compute Coleman Initiative Power index*

---

**Description**

Calculates the Coleman Initiative Power index for a specified simple TU game. Note that in general the Coleman Initiative Power index is not an efficient vector, i.e. the sum of its entries is not always 1. Hence no drawing routine for the Coleman Initiative Power index is provided.

**Usage**`colemanInitiativePowerIndex(v)`**Arguments**

`v` Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with  $n$  players

**Value**

Coleman Initiative Power index for specified simple game

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**References**

Coleman J.S. (1971) "Control of collectivities and the power of a collectivity to act". In: Liberman B. (Ed.), *Social Choice*, Gordon and Breach, pp. 269–300

Chakravarty S.R., Mitra M. and Sarkar P. (2015) *A Course on Cooperative Game Theory*, Cambridge University Press, pp. 120–123

De Keijzer B. (2008) "A survey on the computation of power indices", Technical Report, Delft University of Technology, p. 18

Bertini C. and Stach I. (2011) "Coleman index", *Encyclopedia of Power*, SAGE Publications, p. 117–119

**Examples**

```
library(CoopGame)
v=c(0,0,0,1,1,0,1)
colemanInitiativePowerIndex(v)
```

---

colemanPreventivePowerIndex

*Compute Coleman Preventive Power index*

---

### Description

Calculates the Coleman Preventive Power index for a specified simple TU game. Note that in general the Coleman Preventive Power index is not an efficient vector, i.e. the sum of its entries is not always 1. Hence no drawing routine for the Coleman Preventive Power index is provided.

### Usage

```
colemanPreventivePowerIndex(v)
```

### Arguments

v                      Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with n players

### Value

Coleman Preventive Power index for specified simple game

### Author(s)

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

### References

Coleman J.S. (1971) "Control of collectivities and the power of a collectivity to act". In: Liberman B. (Ed.), *Social Choice*, Gordon and Breach, pp. 269–300

Chakravarty S.R., Mitra M. and Sarkar P. (2015) *A Course on Cooperative Game Theory*, Cambridge University Press, pp. 120–123

De Keijzer B. (2008) "A survey on the computation of power indices", Technical Report, Delft University of Technology, p. 18

Bertini C. and Stach I. (2011) "Coleman index", *Encyclopedia of Power*, SAGE Publications, p. 117–119

### Examples

```
library(CoopGame)
v=c(0,0,0,1,1,0,1)
colemanPreventivePowerIndex(v)
```



---

coreCoverVertices      *Compute core cover vertices*

---

### Description

Calculates the core cover for a given game vector

### Usage

```
coreCoverVertices(v)
```

### Arguments

`v`                      Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with  $n$  players

### Value

rows of the matrix are the vertices of the core cover

### Author(s)

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

### References

Tijs S.H. and Lipperts F.A.S. (1982) "The hypercube and the core cover of  $n$ -person cooperative games", *Cahiers du Centre d' Etudes de Recherche Operationelle* 24, pp. 27–37

Branzei R., Dimitrov D. and Tijs S. (2006) *Models in cooperative game theory*, Springer, p. 21

Chakravarty S.R., Mitra M. and Sarkar P. (2015) *A Course on Cooperative Game Theory*, Cambridge University Press, pp. 45–46

### Examples

```
library(CoopGame)
coreCoverVertices(c(0,0,0,1,1,1,3))
```

```
library(CoopGame)
v <- c(0,0,0,3,3,3,6)
coreCoverVertices(v)
#      [,1] [,2] [,3]
# [1,]   3   0   3
# [2,]   0   3   3
# [3,]   3   3   0
```

---

coreVertices	<i>Compute core vertices</i>
--------------	------------------------------

---

**Description**

Calculates the core vertices for given game vector

**Usage**

```
coreVertices(v)
```

**Arguments**

v	Numeric vector of length $2^n - 1$ representing the values of the coalitions of a TU game with n players
---	--

**Value**

rows of the matrix are the vertices of the core

**Author(s)**

Franz Mueller  
 Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**References**

Gillies D.B. (1953) *Some Theorems on n-person Games*, Ph.D. Thesis, Princeton University Press.  
 Aumann R.J. (1961) "The core of a cooperative game without side payments", *Transactions of the American Mathematical Society* 98(3), pp. 539–552  
 Peleg B. and Sudhoelter P. (2007) *Theory of cooperative games*, 2nd Edition, Springer, pp. 27–49  
 Maschler M., Solan E. and Zamir S. (2013) *Game Theory*, Cambridge University Press, pp. 686–747  
 Osborne M.J. and Rubinstein A. (1994) *A Course in Game Theory*, MIT Press, pp. 257–275

**Examples**

```
library(CoopGame)
coreVertices(c(0,0,0,1,1,1,3))

#In the following case the core consists of a single point
v1 = c(0,1,2,3,4,5,6)
coreVertices(v1)
#      [,1] [,2] [,3]
#[1,]    1    2    3
```

```
#Users may also want to try the following commands:
coreVertices(c(0,0,0,60,80,100,135))
coreVertices(c(5,2,4,7,15,15,15,15,15,20,20,20,20,35))
coreVertices(c(0,0,0,0,0,5,5,5,5,5,5,5,5,5,15,15,15,15,15,15,15,15,15,15,15,30,30,30,30,30,60))
```

---

costSharingGame	<i>Construct a cost sharing game</i>
-----------------	--------------------------------------

---

### Description

**Create a list containing all information about a specified cost sharing game:**

The user may specify the cost function of a cost allocation problem. A corresponding savings game will be calculated. The savings game specified by the game vector  $v$  will work like an ordinary TU game.

### Usage

```
costSharingGame(n, Costs)
```

### Arguments

n	represents the number of players
Costs	A vector containing the costs each coalition has to pay

### Value

A list with three elements representing the specified cost sharing game (n, Costs, Game vector  $v$ )

### Related Functions

[costSharingGameValue](#), [costSharingGameVector](#)

### Author(s)

Jochen Staudacher <jochen.staudacher@hs-kempten.de>  
 Johannes Anwander <anwander.johannes@gmail.com>

### References

Peleg B. and Sudhoelter P. (2007) *Theory of cooperative games*, 2nd Edition, Springer, pp. 14–16  
 Maschler M., Solan E. and Zamir S. (2013) *Game Theory*, Cambridge University Press, pp. 667–668

**Examples**

```

library(CoopGame)
costSharingGame(n=3, Costs=c(2,2,2,3,3,3,4))

#Example with 3 students sharing an apartment:
#-----
#| costs      | A | B | C |
#- -----
#|single      | 300 | 270 | 280 |
#|apartment   |     |     |     |
#-----
#
#Apartment for 2 persons => costs: 410
#Apartment for 3 persons => costs: 550

#Savings for all combinations sharing apartments
library(CoopGame)
(vv <- costSharingGame(n=3, Costs=c(300,270,280,410,410,410,550)))
#Output:
#$n
#[1] 3

#$Costs
#[1] 300 270 280 410 410 410 550

#$v
#[1] 0 0 0 160 170 140 300

```

---

costSharingGameValue *Compute value of a coalition for a cost game*

---

**Description****Coalition value for a cost sharing game:**

For further information see [costSharingGame](#)

**Usage**

```
costSharingGameValue(S, Costs)
```

**Arguments**

S                    numeric vector with coalition of players  
Costs                A vector containing the costs each coalition has to pay

**Value**

Cost savings of coalition S as compared to singleton coalitions

**Author(s)**

Johannes Anwander <anwander.johannes@gmail.com>

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**References**

Peleg B. and Sudhoelter P. (2007) *Theory of cooperative games*, 2nd Edition, Springer, pp. 14–16

Maschler M., Solan E. and Zamir S. (2013) *Game Theory*, Cambridge University Press, pp. 667–668

**Examples**

```
library(CoopGame)
costSharingGameValue(S=c(1,2), Costs=c(2,2,2,3,3,3,4))

#Example with 3 students sharing an apartment:
#-----
#| costs   | A | B | C |
#- -----
#|single   | 300 | 270 | 280 |
#|apartment|    |    |    |
#-----
#
#Apartment for 2 persons => costs: 410
#Apartment for 3 persons => costs: 550

#Savings when A and B share apartment
library(CoopGame)
costSharingGameValue(S=c(1,2), Costs=c(300,270,280,410,410,410,550))
#Output:
#[1] 160
```

---

costSharingGameVector *Compute game vector for a cost sharing game*

---

**Description**

**Coalition vector for a cost sharing game:**

For further information see [costSharingGame](#)

**Usage**

```
costSharingGameVector(n, Costs)
```

**Arguments**

n represents the number of players  
 Costs A vector containing the costs each coalition has to pay

**Value**

Game vector with cost-savings of each coalition S as compared to singleton coalitions.

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>  
 Johannes Anwander <anwander.johannes@gmail.com>

**References**

Peleg B. and Sudhoelter P. (2007) *Theory of cooperative games*, 2nd Edition, Springer, pp. 14–16  
 Maschler M., Solan E. and Zamir S. (2013) *Game Theory*, Cambridge University Press, pp. 667–668

**Examples**

```
library(CoopGame)
costSharingGameVector(n=3, Costs=c(2,2,2,3,3,3,4))

#Example with 3 students sharing an apartment:
#-----
#| costs      | A | B | C |
#- -----
#|single     | 300 | 270 | 280 |
#|apartment  |     |     |     |
#-----
#
#Apartment for 2 persons => costs: 410
#Apartment for 3 persons => costs: 550

#Savings for all combinations sharing apartments
library(CoopGame)
(v=costSharingGameVector(n=3, Costs=c(300,270,280,410,410,410,550)))
#Output:
#[1] 0 0 0 160 170 140 300
```

---

createBitMatrix      *create bit matrix*

---

### Description

createBitMatrix creates a bit matrix with (numberOfPlayers+1) columns and (2<sup>numberOfPlayers</sup>-1) rows which contains all possible coalitions (apart from the null coalition) for the set of all players. Each player is represented by a column which describes if this player is either participating (value 1) or not participating (value 0). The last column (named cVal) contains the values of each coalition. Accordingly, each row of the bit matrix expresses a coalition as a subset of all players.

### Usage

```
createBitMatrix(n, A = NULL)
```

### Arguments

n	represents the number of players
A	Numeric vector of appropriate size

### Value

The return is a bit matrix containing all possible coalitions apart from the empty coalition

### Author(s)

Johannes Anwander <anwander.johannes@gmail.com>  
Jochen Staudacher <jochen.staudacher@hs-kempten.de>

### Examples

```
library(CoopGame)
createBitMatrix(3,c(0,0,0,60,60,60,72))
```

```
library(CoopGame)
A=weightedVotingGameVector(n=3,w=c(1,2,3),q=5)
bm=createBitMatrix(3,A)
bm
# Output:
#           cVal
# [1,] 1 0 0    0
# [2,] 0 1 0    0
# [3,] 0 0 1    0
# [4,] 1 1 0    0
# [5,] 1 0 1    0
# [6,] 0 1 1    1
# [7,] 1 1 1    1
```

---

deeganPackelIndex      *Compute Deegan-Packel index*

---

### Description

deeganPackelIndex calculates the Deegan-Packel index for simple games

### Usage

```
deeganPackelIndex(v)
```

### Arguments

v                      Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with n players

### Value

Deegan-Packel index for a specified simple game

### Author(s)

Johannes Anwander <anwander.johannes@gmail.com>

Michael Maerz

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

### References

Deegan J. and Packel E.W. (1978) "A new index of power for simple n-person games", Int. Journal of Game Theory 7(2), pp. 151–161

Holler M.J. and Illing G. (2006) "Einfuehrung in die Spieltheorie". 6th Edition (in German), Springer, pp. 323–324

### Examples

```
library(CoopGame)
deeganPackelIndex(c(0,0,0,0,1,0,1))

#Example from HOLLER & ILLING (2006), chapter 6.3.3
#Expected result: dpv=(18/60,9/60,11/60,11/60,11/60)
library(CoopGame)
v1=weightedVotingGameVector(n = 5, w=c(35,20,15,15,15), q=51)
deeganPackelIndex(v1)
#Output (as expected, see HOLLER & ILLING chapter 6.3.3) :
#[1] 0.3000000 0.1500000 0.1833333 0.1833333 0.1833333
```



---

dictatorGame	<i>Construct a dictator game</i>
--------------	----------------------------------

---

### Description

**Create a list containing all information about a specified dictator game:**

Any coalitions including the dictator receive coalition value 1. All the other coalitions, i.e. each and every coalition not containing the dictator, receives coalition value 0.

Note that dictator games are always simple games.

### Usage

```
dictatorGame(n, dictator)
```

### Arguments

n represents the number of players

dictator Number of the dictator

### Value

A list with three elements representing the dictator game (n, dictator, Game vector v)

### Related Functions

[dictatorGameValue](#), [dictatorGameVector](#)

### Author(s)

Johannes Anwander <anwander.johannes@gmail.com>

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

### References

Peters H. (2015) *Game Theory: A Multi-Leveled Approach*, 2nd Edition, Springer, p. 295

Maschler M., Solan E. and Zamir S. (2013) *Game Theory*, Cambridge University Press, p. 764

### Examples

```
library(CoopGame)
dictatorGame(n=3,dictator=2)
```

```
library(CoopGame)
dictatorGame(n=4,dictator=2)
#Output:
# $n
#[1] 4
```

```

#$dictator
#[1] 2

#$v
#[1] 0 1 0 0 1 0 0 1 1 0 1 1 0 1 1

```

---

dictatorGameValue      *Compute value of a coalition for a dictator game*

---

### Description

**Coalition value for a dictator game:**  
 For further information see [dictatorGame](#)

### Usage

```
dictatorGameValue(S, dictator)
```

### Arguments

S	numeric vector with coalition of players
dictator	Number of the dictator

### Value

1 if dictator is involved in coalition, 0 otherwise.

### Author(s)

Johannes Anwander <anwander.johannes@gmail.com>  
 Jochen Staudacher <jochen.staudacher@hs-kempten.de>

### References

Peters H. (2015) *Game Theory: A Multi-Leveled Approach*, 2nd Edition, Springer, p. 295  
 Maschler M., Solan E. and Zamir S. (2013) *Game Theory*, Cambridge University Press, p. 764

### Examples

```

library(CoopGame)
dictatorGameValue(S=c(1,2,3),dictator=2)

```

---

dictatorGameVector     *Compute game vector for a dictator game*

---

## Description

### Game vector for a dictator game:

For further information see [dictatorGame](#)

## Usage

```
dictatorGameVector(n, dictator)
```

## Arguments

n	represents the number of players
dictator	Number of the dictator

## Value

Game vector where each element contains 1 if dictator is involved, 0 otherwise.

## Author(s)

Johannes Anwander <anwander.johannes@gmail.com>

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

## References

Peters H. (2015) *Game Theory: A Multi-Leveled Approach*, 2nd Edition, Springer, p. 295

Maschler M., Solan E. and Zamir S. (2013) *Game Theory*, Cambridge University Press, p. 764

## Examples

```
library(CoopGame)
dictatorGameVector(n=3,dictator=2)
```

---

disruptionNucleolus    *Compute disruption nucleolus*

---

### Description

Computes the disruption nucleolus of a balanced TU game with  $n$  players. Note that the disruption nucleolus needs to be a member of the core.

### Usage

```
disruptionNucleolus(v)
```

### Arguments

$v$                       Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with  $n$  players

### Value

Numeric vector of length  $n$  representing the disruption nucleolus of the specified TU game

### Author(s)

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

Johannes Anwander <anwander.johannes@gmail.com>

### References

Littlechild S.C. and Vaidya K.G. (1976) "The propensity to disrupt and the disruption nucleolus of a characteristic function game", *Int. Journal of Game Theory* 5(2), pp. 151–161

### Examples

```
library(CoopGame)
v<-c(0, 0, 0, 1, 1, 0, 1)
disruptionNucleolus(v)
```

```
library(CoopGame)
exampleVector<-c(0,0,0,0,2,3,4,1,3,2,8,11,6.5,9.5,14)
disruptionNucleolus(exampleVector)
#[1] 3.193548 4.754839 2.129032 3.922581
```

---

divideTheDollarGame    *Construct a divide-the-dollar game*

---

### Description

**Create a list containing all information about a specified divide-the-dollar game:**

Returns a divide-the-dollar game with  $n$  players:

This sample game is taken from the book 'Social and Economic Networks' by Matthew O. Jackson (see p. 413 ff.). If coalition  $S$  has at least  $n/2$  members it generates a value of 1, otherwise 0.

Note that divide-the-dollar games are always simple games.

### Usage

```
divideTheDollarGame(n)
```

### Arguments

$n$                       represents the number of players

### Value

A list with two elements representing the divide-the-dollar game ( $n$ , Game vector  $v$ )

### Related Functions

[divideTheDollarGameValue](#), [divideTheDollarGameVector](#)

### Author(s)

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

Johannes Anwander <anwander.johannes@gmail.com>

### References

Jackson M.O. (2008) *Social and Economic Networks*, Princeton University Press, p. 413

### Examples

```
library(CoopGame)
divideTheDollarGame(n=3)

#Example with four players
library(CoopGame)
(vv<-divideTheDollarGame(n=4))
#$n
#[1] 4

#v
```

```
#[1] 0 0 0 0 1 1 1 1 1 1 1 1 1 1
```

---

divideTheDollarGameValue

*Compute value of a coalition for a divide-the-dollar game*

---

### Description

**Coalition value for a divide-the-dollar game:**

For further information see [divideTheDollarGame](#)

### Usage

```
divideTheDollarGameValue(S, n)
```

### Arguments

S                    numeric vector with coalition of players

n                    represents the number of players

### Value

value of coalition

### Author(s)

Michael Maerz

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

### References

Jackson M.O. (2008) *Social and Economic Networks*, Princeton University Press, p. 413

### Examples

```
library(CoopGame)
S <- c(1,2)
divideTheDollarGameValue(S, n = 3)
```

---

`divideTheDollarGameVector`*Compute game vector for a divide-the-dollar game*

---

**Description****Game vector for a divide-the-dollar game:**For further information see [divideTheDollarGame](#)**Usage**`divideTheDollarGameVector(n)`**Arguments**`n` represents the number of players**Value**

Game vector for the specified divide-the-dollar game

**Author(s)**

Jochen Staudacher &lt;jochen.staudacher@hs-kempten.de&gt;

Johannes Anwander &lt;anwander.johannes@gmail.com&gt;

**References**Jackson M.O. (2008) *Social and Economic Networks*, Princeton University Press, p. 413**Examples**

```
library(CoopGame)
divideTheDollarGameVector(n=3)

library(CoopGame)
(v <- divideTheDollarGameVector(n=4))
#Output:
# [1] 0 0 0 0 1 1 1 1 1 1 1 1 1 1
```

---

drawCentroidCore      *draw centroid of the core for 3 or 4 players*

---

### Description

drawCentroidCore draws the centroid of the core for 3 or 4 players.

### Usage

```
drawCentroidCore(
  v,
  holdOn = FALSE,
  colour = NA,
  label = TRUE,
  name = "centroid of core"
)
```

### Arguments

v	Numeric vector of length $2^n - 1$ representing the values of the coalitions of a TU game with n players
holdOn	draws in a existing plot
colour	draws the geometric object (i.e. point or convex polyhedron) with this colour, all colour names can be seen with "colors()"
label	activates the labels for the figure
name	set a name for the label

### Value

None.

### Author(s)

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

### References

Gillies D.B. (1953) *Some Theorems on n-person Games*, Ph.D. Thesis, Princeton University Press.

Aumann R.J. (1961) "The core of a cooperative game without side payments", *Transactions of the American Mathematical Society* 98(3), pp. 539–552

Peleg B. and Sudhoelter P. (2007) *Theory of cooperative games*, 2nd Edition, Springer, pp. 27–49

Maschler M., Solan E. and Zamir S. (2013) *Game Theory*, Cambridge University Press, pp. 686–747

Osborne M.J. and Rubinstein A. (1994) *A Course in Game Theory*, MIT Press, pp. 257–275



**Examples**

```
library(CoopGame)
v <-c(1,2,3,60,60,60,142)
drawCentroidCore(v,colour="green")
```

---

drawCentroidCoreCover *draw centroid of core cover for 3 or 4 players*

---

**Description**

drawCentroidCoreCover draws the centroid of the core cover for 3 or 4 players.

**Usage**

```
drawCentroidCoreCover(
  v,
  holdOn = FALSE,
  colour = NA,
  label = TRUE,
  name = "centroid of core cover"
)
```

**Arguments**

v	Numeric vector of length $2^n - 1$ representing the values of the coalitions of a TU game with n players
holdOn	draws in a existing plot
colour	draws the geometric object (i.e. point or convex polyhedron) with this colour, all colour names can be seen with "colors()"
label	activates the labels for the figure
name	set a name for the label

**Value**

None.

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**References**

Tijs S.H. and Lipperts F.A.S. (1982) "The hypercube and the core cover of n-person cooperative games", Cahiers du Centre d' Etudes de Recherche Operationelle 24, pp. 27–37

Branzei R., Dimitrov D. and Tijs S. (2006) *Models in cooperative game theory*, Springer, p. 21

Chakravarty S.R., Mitra M. and Sarkar P. (2015) *A Course on Cooperative Game Theory*, Cambridge University Press, pp. 45–46

**Examples**

```
library(CoopGame)
v <-c(1,2,3,60,60,60,142)
drawCentroidCoreCover(v,colour="black")
```

---

drawCentroidImputationSet

*draw centroid of imputation set for 3 or 4 players*

---

**Description**

drawCentroidImputationSet draws the centroid of the imputation set for 3 or 4 players.

**Usage**

```
drawCentroidImputationSet(
  v,
  holdOn = FALSE,
  colour = NA,
  label = TRUE,
  name = "centroid of imputation set"
)
```

**Arguments**

v	Numeric vector of length $2^n - 1$ representing the values of the coalitions of a TU game with n players
holdOn	draws in a existing plot
colour	draws the geometric object (i.e. point or convex polyhedron) with this colour, all colour names can be seen with "colors()"
label	activates the labels for the figure
name	set a name for the label

**Value**

None.

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**References**

- Peleg B. and Sudhoelter P. (2007) *Theory of cooperative games*, 2nd Edition, Springer, p. 20
- Maschler M., Solan E. and Zamir S. (2013) *Game Theory*, Cambridge University Press, p. 674
- Osborne M.J. and Rubinstein A. (1994) *A Course in Game Theory*, MIT Press, p. 278
- Narahari Y. (2015) *Game Theory and Mechanism Design*, World Scientific Publishing, p. 407

**Examples**

```
library(CoopGame)
v <-c(1,2,3,60,60,60,142)
drawCentroidImputationSet(v,colour="green")
```

---

```
drawCentroidReasonableSet
```

*draw centroid of reasonable set for 3 or 4 players*

---

**Description**

drawCentroidReasonableSet draws the centroid of the reasonable set for 3 or 4 players.

**Usage**

```
drawCentroidReasonableSet(
  v,
  holdOn = FALSE,
  colour = NA,
  label = TRUE,
  name = "centroid of reasonable set"
)
```

**Arguments**

v	Numeric vector of length $2^n - 1$ representing the values of the coalitions of a TU game with n players
holdOn	draws in a existing plot
colour	draws the geometric object (i.e. point or convex polyhedron) with this colour, all colour names can be seen with "colors()"
label	activates the labels for the figure
name	set a name for the label

**Value**

None.

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**References**

Milnor J.W. (1953) *Reasonable Outcomes for N-person Games*, Rand Corporation, Research Memorandum RM 916.

Branzei R., Dimitrov D. and Tijs S. (2006) *Models in cooperative game theory*, Springer, p. 21

Chakravarty S.R., Mitra M. and Sarkar P. (2015) *A Course on Cooperative Game Theory*, Cambridge University Press, pp. 43–44

Gerard-Varet L.A. and Zamir S. (1987) "Remarks on the reasonable set of outcomes in a general coalition function form game", *Int. Journal of Game Theory* 16(2), pp. 123–143

**Examples**

```
library(CoopGame)
v <-c(1,2,3,60,60,60,142)
drawCentroidReasonableSet(v,colour="green")
```

---

drawCentroidWeberSet *draw centroid of Weber set for 3 or 4 players*

---

**Description**

drawCentroidWeberSet draws the centroid of the Weber set for 3 or 4 players.

**Usage**

```
drawCentroidWeberSet(
  v,
  holdOn = FALSE,
  colour = NA,
  label = TRUE,
  name = "centroid of Weber set"
)
```

**Arguments**

v	Numeric vector of length $2^n - 1$ representing the values of the coalitions of a TU game with n players
holdOn	draws in a existing plot
colour	draws the geometric object (i.e. point or convex polyhedron) with this colour, all colour names can be seen with "colors()"
label	activates the labels for the figure
name	set a name for the label

**Value**

None.

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**References**

Weber R.J. (1988) "Probabilistic values for games". In: Roth A.E. (Ed.), The Shapley Value. Essays in honor of Lloyd S. Shapley, Cambridge University Press, pp. 101–119

Peters H. (2015) *Game Theory: A Multi-Leveled Approach*, 2nd Edition, Springer, pp. 327–329

**Examples**

```
library(CoopGame)
v <-c(1,2,3,60,60,60,142)
drawCentroidWeberSet(v,colour="blue")
```

---

drawCore

*Draw core for 3 or 4 players*

---

**Description**

drawCore draws the core for 3 or 4 players.

**Usage**

```
drawCore(v, holdOn = FALSE, colour = "red", label = FALSE, name = "Core")
```

**Arguments**

v	Numeric vector of length $2^n - 1$ representing the values of the coalitions of a TU game with n players
holdOn	draws in a existing plot
colour	draws the geometric object (i.e. point or convex polyhedron) with this colour, all colour names can be seen with "colors()"
label	activates the labels for the figure
name	set a name for the label

**Value**

None.

**Author(s)**

Johannes Anwander <anwander.johannes@gmail.com>  
 Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**References**

Gillies D.B. (1953) *Some Theorems on n-person Games*, Ph.D. Thesis, Princeton University Press.  
 Aumann R.J. (1961) "The core of a cooperative game without side payments", *Transactions of the American Mathematical Society* 98(3), pp. 539–552  
 Peleg B. and Sudhoelter P. (2007) *Theory of cooperative games*, 2nd Edition, Springer, pp. 27–49  
 Maschler M., Solan E. and Zamir S. (2013) *Game Theory*, Cambridge University Press, pp. 686–747  
 Osborne M.J. and Rubinstein A. (1994) *A Course in Game Theory*, MIT Press, pp. 257–275

**Examples**

```
library(CoopGame)
v <- c(0,0,0,3,3,3,6)
drawCore(v)
```

---

drawCoreCover

*Draw core cover for 3 or 4 players*

---

**Description**

drawCoreCover draws the core cover for 3 or 4 players.

**Usage**

```
drawCoreCover(
  v,
  holdOn = FALSE,
  colour = NA,
  label = FALSE,
  name = "Core Cover"
)
```

**Arguments**

v	Numeric vector of length $2^n - 1$ representing the values of the coalitions of a TU game with n players
holdOn	draws in a existing plot
colour	draws the geometric object (i.e. point or convex polyhedron) with this colour, all colour names can be seen with "colors()"
label	activates the labels for the figure
name	set a name for the label

**Value**

None.

**Author(s)**

Johannes Anwander <anwander.johannes@gmail.com>  
 Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**References**

Tijs S.H. and Lipperts F.A.S. (1982) "The hypercube and the core cover of n-person cooperative games", Cahiers du Centre d' Etudes de Recherche Operationelle 24, pp. 27–37  
 Branzei R., Dimitrov D. and Tijs S. (2006) *Models in cooperative game theory*, Springer, p. 21  
 Chakravarty S.R., Mitra M. and Sarkar P. (2015) *A Course on Cooperative Game Theory*, Cambridge University Press, pp. 45–46

**Examples**

```
library(CoopGame)
v <- c(0,0,0,3,3,3,6)
drawCoreCover(v)
```

---

drawDeeganPackelIndex *draw Deegan-Packel index for 3 or 4 players*

---

**Description**

drawDeeganPackelIndex draws the Deegan-Packel index for 3 or 4 players.

**Usage**

```
drawDeeganPackelIndex(
  v,
  holdOn = FALSE,
  colour = NA,
  label = TRUE,
  name = "Deegan Packel Index"
)
```

**Arguments**

v	Numeric vector of length $2^n - 1$ representing the values of the coalitions of a TU game with n players
holdOn	draws in a existing plot
colour	draws the geometric object (i.e. point or convex polyhedron) with this colour, all colour names can be seen with "colors()"
label	activates the labels for the figure
name	set a name for the label

**Value**

None.

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**References**

Deegan J. and Packel E.W. (1978) "A new index of power for simple n-person games", Int. Journal of Game Theory 7(2), pp. 151–161

Holler M.J. and Illing G. (2006) "Einfuehrung in die Spieltheorie". 6th Edition (in German), Springer, pp. 323–324

**Examples**

```
library(CoopGame)
v=c(0,0,0,1,1,0,1)
drawDeeganPackelIndex(v)
```

---

```
drawDisruptionNucleolus
```

*draw disruption nucleolus for 3 or 4 players*

---

**Description**

drawDisruptionNucleolus draws the disruption nucleolus for 3 or 4 players.

**Usage**

```
drawDisruptionNucleolus(
  v,
  holdOn = FALSE,
  colour = NA,
  label = TRUE,
  name = "Disruption Nucleolus"
)
```

**Arguments**

v	Numeric vector of length $2^n - 1$ representing the values of the coalitions of a TU game with n players
holdOn	draws in a existing plot
colour	draws the geometric object (i.e. point or convex polyhedron) with this colour, all colour names can be seen with "colors()"
label	activates the labels for the figure
name	set a name for the label



**Value**

None.

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**References**

Littlechild S.C. and Vaidya K.G. (1976) "The propensity to disrupt and the disruption nucleolus of a characteristic function game", Int. Journal of Game Theory 5(2), pp. 151–161

**Examples**

```
library(CoopGame)
v<-bankruptcyGameVector(n=3,d=c(100,200,300),E=200)
drawDisruptionNucleolus(v)
```

---

drawGatelyValue      *draw Gately point for 3 or 4 players*

---

**Description**

drawGatelyValue draws the Gately point for 3 or 4 players.

**Usage**

```
drawGatelyValue(
  v,
  holdOn = FALSE,
  colour = NA,
  label = TRUE,
  name = "Gately Value"
)
```

**Arguments**

v	Numeric vector of length $2^n - 1$ representing the values of the coalitions of a TU game with n players
holdOn	draws in a existing plot
colour	draws the geometric object (i.e. point or convex polyhedron) with this colour, all colour names can be seen with "colors()"
label	activates the labels for the figure
name	set a name for the label

**Value**

None.

**Author(s)**

Johannes Anwander <anwander.johannes@gmail.com>

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**References**

Gately D. (1974) "Sharing the Gains from Regional Cooperation. A Game Theoretic Application to Planning Investment in Electric Power", *International Economic Review* 15(1), pp. 195–208

Staudacher J. and Anwander J. (2019) "Conditions for the uniqueness of the Gately point for cooperative games", arXiv preprint, arXiv:1901.01485, 10 pages.

Littlechild S.C. and Vaidya K.G. (1976) "The propensity to disrupt and the disruption nucleolus of a characteristic function game", *Int. Journal of Game Theory* 5(2), pp. 151–161

Narahari Y. (2015) *Game Theory and Mechanism Design*, World Scientific Publishing, pp. 455–456

**Examples**

```
library(CoopGame)
drawGatelyValue(c(0,0,0,1,1,1,3.5))
```

```
#Example from original paper by Gately (1974):
library(CoopGame)
v=c(0,0,0,1170,770,210,1530)
drawGatelyValue(v)
```

---

drawImputationset      *Draw imputation set for 3 or 4 players*

---

**Description**

drawImputationset draws the imputation set for 3 or 4 players.

**Usage**

```
drawImputationset(v, label = TRUE)
```

**Arguments**

v	Numeric vector of length $2^n - 1$ representing the values of the coalitions of a TU game with n players
label	activates the labels for the figure

**Value**

None.

**Author(s)**

Johannes Anwander <anwander.johannes@gmail.com>  
 Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**References**

Peleg B. and Sudhoelter P. (2007) *Theory of cooperative games*, 2nd Edition, Springer, p. 20  
 Maschler M., Solan E. and Zamir S. (2013) *Game Theory*, Cambridge University Press, p. 674  
 Osborne M.J. and Rubinstein A. (1994) *A Course in Game Theory*, MIT Press, p. 278  
 Narahari Y. (2015) *Game Theory and Mechanism Design*, World Scientific Publishing, p. 407

**Examples**

```
library(CoopGame)
v=c(0,1,2,3,4,5,6)
drawImputationset(v)
```

---

drawJohnstonIndex      *Draw Johnston index for 3 or 4 players*

---

**Description**

drawJohnstonIndex draws the Johnston index for 3 or 4 players.

**Usage**

```
drawJohnstonIndex(
  v,
  holdOn = FALSE,
  colour = NA,
  label = TRUE,
  name = "Johnston index"
)
```

**Arguments**

v	Numeric vector of length $2^n - 1$ representing the values of the coalitions of a TU game with n players
holdOn	draws in a existing plot
colour	draws the geometric object (i.e. point or convex polyhedron) with this colour, all colour names can be seen with "colors()"
label	activates the labels for the figure
name	set a name for the label

**Value**

None.

**References**

Johnston R.J. (1978) "On the measurement of power: Some reactions to Laver", Environment and Planning A, pp. 907–914

Chakravarty S.R., Mitra M. and Sarkar P. (2015) *A Course on Cooperative Game Theory*, Cambridge University Press, p. 124

**Examples**

```
library(CoopGame)
v <- c(0,0,0,1,1,0,1)
drawJohnstonIndex(v)
```

---

drawModiclus	<i>Draw modiclus for 3 or 4 players</i>
--------------	---

---

**Description**

drawModiclus draws the modiclus for 3 or 4 players.

**Usage**

```
drawModiclus(v, holdOn = FALSE, colour = NA, label = TRUE, name = "Modiclus")
```

**Arguments**

v	Numeric vector of length $2^n - 1$ representing the values of the coalitions of a TU game with n players
holdOn	draws in a existing plot
colour	draws the geometric object (i.e. point or convex polyhedron) with this colour, all colour names can be seen with "colors()"
label	activates the labels for the figure
name	set a name for the label

**Value**

None.

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

## References

- Peleg B. and Sudhoelter P. (2007) *Theory of cooperative games*, 2nd Edition, Springer, pp. 124–132
- Sudhoelter P. (1997) "The Modified Nucleolus. Properties and Axiomatizations", *Int. Journal of Game Theory* 26(2), pp. 147–182
- Sudhoelter P. (1996) "The Modified Nucleolus as Canonical Representation of Weighted Majority Games", *Mathematics of Operations Research* 21(3), pp. 734–756

## Examples

```
library(CoopGame)
v=c(1, 1, 1, 2, 3, 4, 5)
drawModiclus(v)
```

---

```
drawNormalizedBanzhafIndex
```

*draw normalized Banzhaf Index for 3 or 4 players*

---

## Description

drawNormalizedBanzhafIndex draws the Banzhaf Value for 3 or 4 players.  
 Drawing any kind of Banzhaf values only makes sense from our point of view for the normalized Banzhaf index for simple games, because only in this case will the Banzhaf index be efficient.

## Usage

```
drawNormalizedBanzhafIndex(
  v,
  holdOn = FALSE,
  colour = NA,
  label = TRUE,
  name = "Normalized Banzhaf index"
)
```

## Arguments

v	Numeric vector of length $2^n - 1$ representing the values of the coalitions of a TU game with n players
holdOn	draws in a existing plot
colour	draws the geometric object (i.e. point or convex polyhedron) with this colour, all colour names can be seen with "colors()"
label	activates the labels for the figure
name	set a name for the label

## Value

None.

**Author(s)**

Johannes Anwander <anwander.johannes@gmail.com>  
 Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**References**

Peters H. (2015) *Game Theory: A Multi-Leveled Approach*, 2nd Edition, Springer, pp. 367–370  
 Chakravarty S.R., Mitra M. and Sarkar P. (2015) *A Course on Cooperative Game Theory*, Cambridge University Press, pp. 118–119  
 Bertini C. and Stach I. (2011) "Banzhaf voting power measure", *Encyclopedia of Power*, SAGE Publications, pp. 54–55

**Examples**

```
library(CoopGame)
v<-weightedVotingGameVector(n=3,w=c(50,30,20),q=c(67))
drawNormalizedBanzhafIndex(v)
```

---

```
drawNormalizedBanzhafValue
```

*draw normalized Banzhaf value for 3 or 4 players*

---

**Description**

drawNormalizedBanzhafValue draws the Banzhaf Value for 3 or 4 players.  
 Drawing any kind of Banzhaf values only makes sense from our point of view for the normalized Banzhaf value, because only in this case will the Banzhaf value be efficient.

**Usage**

```
drawNormalizedBanzhafValue(
  v,
  holdOn = FALSE,
  colour = NA,
  label = TRUE,
  name = "Normalized Banzhaf value"
)
```

**Arguments**

v	Numeric vector of length $2^n - 1$ representing the values of the coalitions of a TU game with n players
holdOn	draws in a existing plot
colour	draws the geometric object (i.e. point or convex polyhedron) with this colour, all colour names can be seen with "colors()"

label            activates the labels for the figure  
 name            set a name for the label

**Value**

None.

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**References**

Gambarelli G. (2011) "Banzhaf value", Encyclopedia of Power, SAGE Publications, pp. 53–54  
 Stach I. (2017) "Sub-Coalitional Approach to Values", In: Nguyen, N.T. and Kowalczyk, R. (Eds.): Transactions on Computational Collective Intelligence XXVI, Springer, pp. 74–86

**Examples**

```
library(CoopGame)
drawNormalizedBanzhafValue(c(0,0,0,1,2,3,6))
```

```
#Example from paper by Gambarelli (2011)
library(CoopGame)
v=c(0,0,0,1,2,1,3)
drawNormalizedBanzhafValue(v)
```

---

drawNucleolus            *Draw nucleolus for 3 or 4 players*

---

**Description**

drawNucleolus draws the nucleolus for 3 or 4 players.

**Usage**

```
drawNucleolus(v, holdOn = FALSE, colour = NA, label = TRUE, name = "Nucleolus")
```

**Arguments**

v                      Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with n players  
 holdOn                draws in a existing plot  
 colour                draws the geometric object (i.e. point or convex polyhedron) with this colour, all colour names can be seen with "colors()"

label            activates the labels for the figure  
 name            set a name for the label

**Value**

None.

**Author(s)**

Johannes Anwander <anwander.johannes@gmail.com>

**References**

- Schmeidler D. (1969) "The nucleolus of a characteristic function game", *SIAM Journal on applied mathematics* 17(6), pp. 1163–1170
- Kohlberg E. (1971) "On the nucleolus of a characteristic function game", *SIAM Journal on applied mathematics* 20(1), pp. 62–66
- Kopelowitz A. (1967) "Computation of the kernels of simple games and the nucleolus of n-person games", Technical Report, Department of Mathematics, The Hebrew University of Jerusalem, 45 pages.
- Megiddo N. (1974) "On the nonmonotonicity of the bargaining set, the kernel and the nucleolus of a game", *SIAM Journal on applied mathematics* 27(2), pp. 355–358
- Peleg B. and Sudhoelter P. (2007) *Theory of cooperative games*, 2nd Edition, Springer, pp. 82–86

**Examples**

```
library(CoopGame)
v=c(0,0,0,1,1,0,3)
drawNucleolus(v)
```

```
#Visualization for estate division problem from Babylonian Talmud with E=300,
#see e.g. seminal paper by Aumann & Maschler from 1985 on
#'Game Theoretic Analysis of a Bankruptcy Problem from the Talmud'
library(CoopGame)
v<-bankruptcyGameVector(n=3,d=c(100,200,300),E=300)
drawNucleolus(v)
```

---

```
drawPerCapitaNucleolus
```

*Draw per capita nucleolus for 3 or 4 players*

---

**Description**

drawPerCapitaNucleolus draws the per capita nucleolus for 3 or 4 players.



**Usage**

```
drawPerCapitaNucleolus(
  v,
  holdOn = FALSE,
  colour = NA,
  label = TRUE,
  name = "Per Capita Nucleolus"
)
```

**Arguments**

v	Numeric vector of length $2^n - 1$ representing the values of the coalitions of a TU game with n players
holdOn	draws in a existing plot
colour	draws the geometric object (i.e. point or convex polyhedron) with this colour, all colour names can be seen with "colors()"
label	activates the labels for the figure
name	set a name for the label

**Value**

None.

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**References**

Young H.P. (1985) "Monotonic Solutions of cooperative games", Int. Journal of Game Theory 14(2), pp. 65–72

**Examples**

```
library(CoopGame)
v=c(0,0,0,1,1,0,3)
drawPerCapitaNucleolus(v)
```

```
#Example from YOUNG 1985, p. 68
library(CoopGame)
v=c(0,0,0,0,9,10,12)
drawPerCapitaNucleolus(v)
```

---

drawPrenucleolus      *Draw prenucleolus for 3 or 4 players*

---

### Description

drawPrenucleolus draws the prenucleolus for 3 or 4 players.

### Usage

```
drawPrenucleolus(
  v,
  holdOn = FALSE,
  colour = NA,
  label = TRUE,
  name = "Prenucleolus"
)
```

### Arguments

v	Numeric vector of length $2^n - 1$ representing the values of the coalitions of a TU game with n players
holdOn	draws in a existing plot
colour	draws the geometric object (i.e. point or convex polyhedron) with this colour, all colour names can be seen with "colors()"
label	activates the labels for the figure
name	set a name for the label

### Value

None.

### Author(s)

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

### References

Peleg B. and Sudhoelter P. (2007) *Theory of cooperative games*, 2nd Edition, Springer, pp. 107–132

### Examples

```
library(CoopGame)
v=c(0,0,0,1,1,0,3)
drawPrenucleolus(v)
```

```
#Visualization for estate division problem from Babylonian Talmud with E=200,
```

```
#see e.g. seminal paper by Aumann & Maschler from 1985 on
#'Game Theoretic Analysis of a Bankruptcy Problem from the Talmud'
library(CoopGame)
v<-bankruptcyGameVector(n=3,d=c(100,200,300),E=200)
drawPrenucleolus(v)
```

---

```
drawProportionalNucleolus
```

*Draw proportional nucleolus for 3 or 4 players*

---

### Description

drawProportionalNucleolus draws the proportional nucleolus for 3 or 4 players.

### Usage

```
drawProportionalNucleolus(
  v,
  holdOn = FALSE,
  colour = NA,
  label = TRUE,
  name = "Proportional Nucleolus"
)
```

### Arguments

v	Numeric vector of length $2^n - 1$ representing the values of the coalitions of a TU game with n players
holdOn	draws in a existing plot
colour	draws the geometric object (i.e. point or convex polyhedron) with this colour, all colour names can be seen with "colors()"
label	activates the labels for the figure
name	set a name for the label

### Value

None.

### Author(s)

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

### References

Young H. P., Okada N. and Hashimoto, T. (1982) "Cost allocation in water resources development", Water resources research 18(3), pp. 463–475

**Examples**

```
library(CoopGame)
v<-c(0,0,0,48,60,72,140)
drawProportionalNucleolus(v)
```

---

drawPublicGoodIndex     *Draw Public Good index for 3 or 4 players*

---

**Description**

drawPublicGoodIndex draws the Public Good Index of a simple game for 3 or 4 players.

**Usage**

```
drawPublicGoodIndex(
  v,
  holdOn = FALSE,
  colour = NA,
  label = TRUE,
  name = "Public Good Index"
)
```

**Arguments**

v	Numeric vector of length $2^n - 1$ representing the values of the coalitions of a TU game with n players
holdOn	draws in a existing plot
colour	draws the geometric object (i.e. point or convex polyhedron) with this colour, all colour names can be seen with "colors()"
label	activates the labels for the figure
name	set a name for the label

**Value**

None.

**Author(s)**

Johannes Anwander <anwander.johannes@gmail.com>  
 Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**References**

Holler M.J. and Packel E.W. (1983) "Power, luck and the right index", Zeitschrift fuer Nationaloekonomie 43(1), pp. 21–29  
 Holler M. (2011) "Public Goods index", Encyclopedia of Power, SAGE Publications, pp. 541–542

**Examples**

```
library(CoopGame)
v=c(0,0,0,1,1,0,1)
drawPublicGoodIndex(v)
```

---

drawPublicGoodValue     *Draw Public Good value for 3 or 4 players*

---

**Description**

drawPublicGoodValue draws the (normalized) Public Good value for 3 or 4 players.

**Usage**

```
drawPublicGoodValue(
  v,
  holdOn = FALSE,
  colour = NA,
  label = TRUE,
  name = "Normalized Public Good Value"
)
```

**Arguments**

v	Numeric vector of length $2^n - 1$ representing the values of the coalitions of a TU game with n players
holdOn	draws in a existing plot
colour	draws the geometric object (i.e. point or convex polyhedron) with this colour, all colour names can be seen with "colors()"
label	activates the labels for the figure
name	set a name for the label

**Value**

None.

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**References**

Holler M.J. and Li X. (1995) "From public good index to public value. An axiomatic approach and generalization", Control and Cybernetics 24, pp. 257 – 270

Bertini C. and Stach I. (2015) "On Public Values and Power Indices", Decision Making in Manufacturing and Services 9(1), pp. 9 – 25

**Examples**

```
library(CoopGame)
v=c(0,0,0,1,1,0,1)
drawPublicGoodValue(v)
```

---

```
drawPublicHelpChiIndex
```

*Draw Public Help index Chi for 3 or 4 players*

---

**Description**

drawPublicHelpChiIndex draws the Public Help index Chi for a simple game with 3 or 4 players.

**Usage**

```
drawPublicHelpChiIndex(
  v,
  holdOn = FALSE,
  colour = NA,
  label = TRUE,
  name = "Public Help Chi Index"
)
```

**Arguments**

v	Numeric vector of length $2^n - 1$ representing the values of the coalitions of a TU game with n players
holdOn	draws in a existing plot
colour	draws the geometric object (i.e. point or convex polyhedron) with this colour, all colour names can be seen with "colors()"
label	activates the labels for the figure
name	set a name for the label

**Value**

None.

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**References**

Bertini C. and Stach I. (2015) "On Public Values and Power Indices", Decision Making in Manufacturing and Services 9(1), pp. 9–25

Stach I. (2016) "Power Measures and Public Goods", In: Nguyen, N.T. and Kowalczyk, R. (Eds.): Transactions on Computational Collective Intelligence XXIII, Springer, pp. 99–110

**Examples**

```
library(CoopGame)
v=c(0,0,0,1,1,0,1)
drawPublicHelpChiIndex(v)
```

---

```
drawPublicHelpChiValue
```

*Draw Public Help value Chi for 3 or 4 players*

---

**Description**

drawPublicHelpChiValue draws the (normalized) Public Help value Chi for 3 or 4 players.

**Usage**

```
drawPublicHelpChiValue(
  v,
  holdOn = FALSE,
  colour = NA,
  label = TRUE,
  name = "Normalized Public Help Value Chi"
)
```

**Arguments**

v	Numeric vector of length $2^n - 1$ representing the values of the coalitions of a TU game with n players
holdOn	draws in a existing plot
colour	draws the geometric object (i.e. point or convex polyhedron) with this colour, all colour names can be seen with "colors()"
label	activates the labels for the figure
name	set a name for the label

**Value**

None.

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**References**

Bertini C. and Stach I. (2015) "On Public Values and Power Indices", Decision Making in Manufacturing and Services 9(1), pp. 9–25

**Examples**

```
library(CoopGame)
v=c(0,0,0,2,2,0,3)
drawPublicHelpChiValue(v)
```

---

drawPublicHelpIndex     *Draw Public Help index Theta for 3 or 4 players*

---

**Description**

drawPublicHelpIndex draws the Public Help index Theta for a simple game with 3 or 4 players.

**Usage**

```
drawPublicHelpIndex(
  v,
  holdOn = FALSE,
  colour = NA,
  label = TRUE,
  name = "Public Help Index"
)
```

**Arguments**

v	Numeric vector of length $2^n - 1$ representing the values of the coalitions of a TU game with n players
holdOn	draws in a existing plot
colour	draws the geometric object (i.e. point or convex polyhedron) with this colour, all colour names can be seen with "colors()"
label	activates the labels for the figure
name	set a name for the label

**Value**

None.

**Author(s)**

Johannes Anwander <anwander.johannes@gmail.com>

Jochen Staudacher <jochen.staudacher@hs-kempten.de>



## References

- Bertini C., Gambarelli G. and Stach I. (2008) "A public help index", In: Braham, M. and Steffen, F. (Eds): Power, freedom, and voting: Essays in Honour of Manfred J. Holler, pp. 83–98
- Bertini C. and Stach I. (2015) "On Public Values and Power Indices", Decision Making in Manufacturing and Services 9(1), pp. 9–25
- Stach I. (2016) "Power Measures and Public Goods", In: Nguyen, N.T. and Kowalczyk, R. (Eds.): Transactions on Computational Collective Intelligence XXIII, Springer, pp. 99–110

## Examples

```
library(CoopGame)
v=c(0,0,0,1,1,0,1)
drawPublicHelpIndex(v)
```

---

drawPublicHelpValue     *Draw Public Help value Theta for 3 or 4 players*

---

## Description

drawPublicHelpValue draws the (normalized) Public Help value Theta for 3 or 4 players.

## Usage

```
drawPublicHelpValue(
  v,
  holdOn = FALSE,
  colour = NA,
  label = TRUE,
  name = "Normalized Public Help Value"
)
```

## Arguments

v	Numeric vector of length $2^n - 1$ representing the values of the coalitions of a TU game with n players
holdOn	draws in a existing plot
colour	draws the geometric object (i.e. point or convex polyhedron) with this colour, all colour names can be seen with "colors()"
label	activates the labels for the figure
name	set a name for the label

## Value

None.

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**References**

Bertini C. and Stach I. (2015) "On Public Values and Power Indices", Decision Making in Manufacturing and Services 9(1), pp. 9–25

**Examples**

```
library(CoopGame)
v=c(0,0,0,1,1,0,1)
drawPublicHelpValue(v)
```

---

drawReasonableSet	<i>Draw reasonable set for 3 or 4 players</i>
-------------------	---

---

**Description**

drawReasonableSet draws the reasonable set for 3 or 4 players.

**Usage**

```
drawReasonableSet(
  v,
  holdOn = FALSE,
  colour = NA,
  label = FALSE,
  name = "Reasonable Set"
)
```

**Arguments**

v	Numeric vector of length $2^n - 1$ representing the values of the coalitions of a TU game with n players
holdOn	draws in a existing plot
colour	draws the geometric object (i.e. point or convex polyhedron) with this colour, all colour names can be seen with "colors()"
label	activates the labels for the figure
name	set a name for the label

**Value**

None.

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**References**

Milnor J.W. (1953) *Reasonable Outcomes for N-person Games*, Rand Corporation, Research Memorandum RM 916.

Branzei R., Dimitrov D. and Tijs S. (2006) *Models in cooperative game theory*, Springer, p. 21

Chakravarty S.R., Mitra M. and Sarkar P. (2015) *A Course on Cooperative Game Theory*, Cambridge University Press, pp. 43–44

Gerard-Varet L.A. and Zamir S. (1987) "Remarks on the reasonable set of outcomes in a general coalition function form game", *Int. Journal of Game Theory* 16(2), pp. 123–143

**Examples**

```
library(CoopGame)
v <- c(0,0,0,3,3,3,6)
drawReasonableSet(v)
```

---

```
drawShapleyShubikIndex
```

*Draw Shapley-Shubik index for 3 or 4 players*

---

**Description**

drawShapleyShubik draws the Shapley-Shubik index simple game with 3 or 4 players.

**Usage**

```
drawShapleyShubikIndex(
  v,
  holdOn = FALSE,
  colour = NA,
  label = TRUE,
  name = "Shapley-Shubik index"
)
```

**Arguments**

v	Numeric vector of length $2^n - 1$ representing the values of the coalitions of a TU game with n players
holdOn	draws in a existing plot
colour	draws the geometric object (i.e. point or convex polyhedron) with this colour, all colour names can be seen with "colors()"
label	activates the labels for the figure
name	set a name for the label

**Value**

None.

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**References**

- Shapley L.S. and Shubik M. (1954) "A method for evaluating the distribution of power in a committee system". *American political science review* 48(3), pp. 787–792
- Shapley L.S. (1953) "A value for n-person games". In: Kuhn, H., Tucker, A.W. (Eds.), *Contributions to the Theory of Games II*, Princeton University Press, pp. 307–317
- Peters H. (2015) *Game Theory: A Multi-Leveled Approach*, 2nd Edition, Springer, pp. 156–159
- Maschler M., Solan E. and Zamir S. (2013) *Game Theory*, Cambridge University Press, pp. 748–781
- Stach I. (2011) "Shapley-Shubik index", *Encyclopedia of Power*, SAGE Publications, pp. 603–606

**Examples**

```
library(CoopGame)
v=c(0,0,0,1,1,0,1)
drawShapleyShubikIndex(v)
```

---

drawShapleyValue

*Draw Shapley value for 3 or 4 players*

---

**Description**

drawShapleyValue draws the Shapley value for 3 or 4 players.

**Usage**

```
drawShapleyValue(
  v,
  holdOn = FALSE,
  colour = NA,
  label = TRUE,
  name = "Shapley value"
)
```

**Arguments**

v	Numeric vector of length $2^n - 1$ representing the values of the coalitions of a TU game with n players
holdOn	draws in a existing plot
colour	draws the geometric object (i.e. point or convex polyhedron) with this colour, all colour names can be seen with "colors()"
label	activates the labels for the figure
name	set a name for the label

**Value**

None.

**Author(s)**

Alexandra Tiukkel

**References**

- Shapley L.S. (1953) "A value for n-person games". In: Kuhn, H., Tucker, A.W. (Eds.), Contributions to the Theory of Games II, Princeton University Press, pp. 307–317
- Aumann R.J. (2010) "Some non-superadditive games, and their Shapley values, in the Talmud", Int. Journal of Game Theory 39(1), pp. 3–10
- Peters H. (2015) *Game Theory: A Multi-Leveled Approach*, 2nd Edition, Springer, pp. 156–159
- Maschler M., Solan E. and Zamir S. (2013) *Game Theory*, Cambridge University Press, pp. 748–781

**Examples**

```
library(CoopGame)
v=c(0,0,0,1,1,0,1)
drawShapleyValue(v)
```

---

```
drawSimplifiedModiclus
```

*Draw simplified modiclus for 3 or 4 players*

---

**Description**

drawSimplifiedModiclus draws the simplified modiclus for 3 or 4 players.

**Usage**

```
drawSimplifiedModiclus(  
  v,  
  holdOn = FALSE,  
  colour = NA,  
  label = TRUE,  
  name = "Simplified Modiclus"  
)
```

**Arguments**

v	Numeric vector of length $2^n - 1$ representing the values of the coalitions of a TU game with n players
holdOn	draws in a existing plot
colour	draws the geometric object (i.e. point or convex polyhedron) with this colour, all colour names can be seen with "colors()"
label	activates the labels for the figure
name	set a name for the label

**Value**

None.

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**References**

Tarashnina S. (2011) "The simplified modified nucleolus of a cooperative TU-game", TOP 19(1), pp. 150–166

**Examples**

```
library(CoopGame)  
v=c(0, 0, 0, 1, 1, 0, 1)  
drawSimplifiedModiclus(v)
```

---

drawTauValue

*Draw tau-value for 3 or 4 players*

---

**Description**

drawTauValue draws the tau-value for 3 or 4 players.

**Usage**

```
drawTauValue(v, holdOn = FALSE, colour = NA, label = TRUE, name = "Tau value")
```

**Arguments**

v	Numeric vector of length $2^n - 1$ representing the values of the coalitions of a TU game with n players
holdOn	draws in a existing plot
colour	draws the geometric object (i.e. point or convex polyhedron) with this colour, all colour names can be seen with "colors()"
label	activates the labels for the figure
name	set a name for the label

**Value**

None.

**Author(s)**

Johannes Anwander <anwander.johannes@gmail.com>  
 Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**References**

Branzei R., Dimitrov D. and Tijs S. (2006) *Models in cooperative game theory*, Springer, p. 32  
 Tijs S. (1981) "Bounds for the core of a game and the t-value", In: Moeschlin, O. and Pallaschke, D. (Eds.): *Game Theory and Mathematical Economics*, North-Holland, pp. 123–132  
 Stach I. (2011) "Tijs value", *Encyclopedia of Power*, SAGE Publications, pp. 667–670

**Examples**

```
library(CoopGame)
v <-c(1,2,3,60,60,60,142)
drawTauValue(v, colour="green")
```

---

drawWeberset	<i>Draw Weber Set for 3 or 4 players</i>
--------------	--

---

**Description**

drawWeberset draws the Weber Set for 3 or 4 players.

**Usage**

```
drawWeberset(v, holdOn = FALSE, colour = NA, label = FALSE, name = "Weber Set")
```

**Arguments**

<code>v</code>	Numeric vector of length $2^n - 1$ representing the values of the coalitions of a TU game with $n$ players
<code>holdOn</code>	draws in a existing plot
<code>colour</code>	draws the geometric object (i.e. point or convex polyhedron) with this colour, all colour names can be seen with "colors()"
<code>label</code>	activates the labels for the figure
<code>name</code>	set a name for the label

**Value**

None.

**Author(s)**

Johannes Anwander <anwander.johannes@gmail.com>

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**References**

Weber R.J. (1988) "Probabilistic values for games". In: Roth A.E. (Ed.), The Shapley Value. Essays in honor of Lloyd S. Shapley, Cambridge University Press, pp. 101–119

Peters H. (2015) *Game Theory: A Multi-Leveled Approach*, 2nd Edition, Springer, pp. 327–329

**Examples**

```
library(CoopGame)
v = c(0,1,2,3,4,5,6)
drawWeberset(v, colour ="yellow")
```

---

```
equalPropensityToDisrupt
```

*Compute equal propensity to disrupt*

---

**Description**

`equalPropensityToDisrupt` calculates the equal propensity to disrupt for a TU game with  $n$  players and a specified coalition size  $k$ . See the original paper by Littlechild & Vaidya (1976) for the formula with general  $k$  and the paper by Staudacher & Anwander (2019) for the specific expression for  $k=1$  and interpretations of the equal propensity to disrupt.

**Usage**

```
equalPropensityToDisrupt(v, k = 1)
```



**Arguments**

- v Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with n players
- k is the fixed coalition size to be considered when calculating the equal propensity to disrupt

**Value**

the value for the equal propensity to disrupt

**Author(s)**

Johannes Anwander <anwander.johannes@gmail.com>  
 Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**References**

- Littlechild S.C. and Vaidya K.G. (1976) "The propensity to disrupt and the disruption nucleolus of a characteristic function game", Int. Journal of Game Theory 5(2), pp. 151–161
- Staudacher J. and Anwander J. (2019) "Conditions for the uniqueness of the Gately point for cooperative games", arXiv preprint, arXiv:1901.01485, 10 pages.

**Examples**

```
library(CoopGame)
v=c(0,0,0,4,0,3,6)
equalPropensityToDisrupt(v, k=1)
```

---

gatelyValue	<i>Compute Gately point</i>
-------------	-----------------------------

---

**Description**

gatelyValue calculates the Gately point for a given TU game

**Usage**

```
gatelyValue(v)
```

**Arguments**

- v Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with n players

**Value**

Gately point of the TU game or NULL in case the Gately point is not defined

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**References**

- Gately D. (1974) "Sharing the Gains from Regional Cooperation. A Game Theoretic Application to Planning Investment in Electric Power", *International Economic Review* 15(1), pp. 195–208
- Staudacher J. and Anwander J. (2019) "Conditions for the uniqueness of the Gately point for cooperative games", arXiv preprint, arXiv:1901.01485, 10 pages.
- Littlechild S.C. and Vaidya K.G. (1976) "The propensity to disrupt and the disruption nucleolus of a characteristic function game", *Int. Journal of Game Theory* 5(2), pp. 151–161
- Narahari Y. (2015) *Game Theory and Mechanism Design*, World Scientific Publishing, pp. 455–456

**Examples**

```
library(CoopGame)
gatelyValue(c(0,0,0,1,1,1,3.5))
```

```
library(CoopGame)
v=c(0,0,0,4,0,3,6)
gatelyValue(v)
```

```
#Output (18/11,36/11,12/11):
#1.636364 3.272727 1.090909
```

```
#Example from original paper by Gately (1974)
library(CoopGame)
v=c(0,0,0,1170,770,210,1530)
gatelyValue(v)
```

```
#Output:
#827.7049 476.5574 225.7377
```

---

```
getCriticalCoalitionsOfPlayer
```

*Compute critical coalitions of a player for simple games*

---

**Description**

getCriticalCoalitionsOfPlayer identifies all coalitions for one player in which that player is critical (within a simple game). These coalitions are characterized by the circumstance that without this player the other players generate no value (then also called a losing coalition) - therefore this player is also described as a critical player.

**Usage**

```
getCriticalCoalitionsOfPlayer(player, v)
```

**Arguments**

player	represents the observed player
v	Numeric vector of length $2^n - 1$ representing the values of the coalitions of a TU game with n players

**Value**

A data frame containing all minimal winning coalitions for one special player

**Author(s)**

Johannes Anwander <anwander.johannes@gmail.com>

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**References**

Deegan J. and Packel E.W. (1978) "A new index of power for simple n-person games", Int. Journal of Game Theory 7(2), pp. 151–161

**Examples**

```
library(CoopGame)
getCriticalCoalitionsOfPlayer(2,v=c(0,0,0,0,0,1,1))
```

```
library(CoopGame)
v=c(0,1,0,1,0,1,1)
```

```
#Get coalitions where player 2 is critical:
getCriticalCoalitionsOfPlayer(2,v)
#Output are all coalitions where player 2 is involved.
#Observe that player 2 is dictator in this game.
#
#   V1 V2 V3 cVal bmRow
# 2  0  1  0    1     2
# 4  1  1  0    1     4
# 6  0  1  1    1     6
# 7  1  1  1    1     7
```

---

getDualGameVector	<i>Compute dual game vector</i>
-------------------	---------------------------------

---

**Description**

Computes the dual game for a given TU game with  $n$  players specified by a game vector.

**Usage**

```
getDualGameVector(v)
```

**Arguments**

v	Numeric vector of length $2^n - 1$ representing the values of the coalitions of a TU game with $n$ players
---	--

**Value**

Numeric vector of length  $(2^n)-1$  representing the dual game.

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

Johannes Anwander <anwander.johannes@gmail.com>

**References**

Peleg B. and Sudhoelter P. (2007) *Theory of cooperative games*, 2nd Edition, Springer, p. 125

Branzei R., Dimitrov D. and Tijs S. (2006) *Models in cooperative game theory*, Springer, p. 7

Maschler M., Solan E. and Zamir S. (2013) *Game Theory*, Cambridge University Press, p. 737

**Examples**

```
library(CoopGame)
v<-unanimityGameVector(4,c(1,2))
getDualGameVector(v)
```

---

```
getEmptyParamCheckResult
```

*getEmptyParamCheckResult for generating structure according to parameter check results*

---

### Description

Returns a defined data structure which is intended to store an error code and a message after the check of function parameters was executed. In case parameter check was successful the error code has the value '0' and the message is 'NULL'.

### Usage

```
getEmptyParamCheckResult()
```

### Value

list with 2 elements named `errCode` which contains an integer representing the error code ('0' if no error) and `errMessage` for the error message ('NULL' if no error)

### Author(s)

Johannes Anwander <anwander.johannes@gmail.com>

### See Also

Other ParameterChecks\_CoopGame: [stopOnInconsistentEstateAndClaimsVector\(\)](#), [stopOnInvalidAllocation\(\)](#), [stopOnInvalidBoolean\(\)](#), [stopOnInvalidClaimsVector\(\)](#), [stopOnInvalidCoalitionS\(\)](#), [stopOnInvalidDictator\(\)](#), [stopOnInvalidEstate\(\)](#), [stopOnInvalidGameVector\(\)](#), [stopOnInvalidGrandCoalitionN\(\)](#), [stopOnInvalidIndex\(\)](#), [stopOnInvalidLeftRightGloveGame\(\)](#), [stopOnInvalidNChooseB\(\)](#), [stopOnInvalidNumberOfPlayers\(\)](#), [stopOnInvalidNumber\(\)](#), [stopOnInvalidQuota\(\)](#), [stopOnInvalidVetoPlayer\(\)](#), [stopOnInvalidWeightVector\(\)](#), [stopOnParamCheckError\(\)](#)

### Examples

```
library(CoopGame)

initParamCheck_example=function(numberOfPlayers){
  paramCheckResult=getEmptyParamCheckResult()
  if(numberOfPlayers!=3){
    paramCheckResult$errMessage="The number of players is not 3 as expected"
    paramCheckResult$errCode=1
  }
  return(paramCheckResult)
}

initParamCheck_example(3)
#Output:
#$errCode
```

```
#[1] 0  
#$errMessage  
#NULL
```

---

getExcessCoefficients *Compute excess coefficients*

---

### Description

getExcessCoefficients computes the excess coefficients for a specified TU game and an allocation  $x$

### Usage

```
getExcessCoefficients(v, x)
```

### Arguments

v	Numeric vector of length $2^n - 1$ representing the values of the coalitions of a TU game with $n$ players
x	numeric vector containing allocations for each player

### Value

numeric vector containing the excess coefficients for every coalition

### Author(s)

Johannes Anwander <anwander.johannes@gmail.com>  
Jochen Staudacher <jochen.staudacher@hs-kempten.de>

### References

Peleg B. and Sudhoelter P. (2007) *Theory of cooperative games*, 2nd Edition, Springer, p. 58  
Driessen T. (1998) *Cooperative Games, Solutions and Applications*, Springer, p. 12

### Examples

```
library(CoopGame)  
getExcessCoefficients(c(0,0,0,60,48,30,72), c(24,24,24))
```

---

getGainingCoalitions *Compute gaining coalitions of a TU game*

---

**Description**

The function getGainingCoalitions identifies all gaining coalitions. Coalition  $S$  is a gaining coalition if there holds:  $v(S) > 0$

**Usage**

```
getGainingCoalitions(v)
```

**Arguments**

$v$  Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with  $n$  players

**Value**

A data frame containing all gaining coalitions.

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**References**

Bertini C. and Stach I. (2015) "On Public Values and Power Indices", Decision Making in Manufacturing and Services 9(1), pp. 9–25

**Examples**

```
library(CoopGame)
getGainingCoalitions(v=c(0,0,0,2,0,2,3))
```

```
library(CoopGame)
v <- c(1,2,3,4,0,0,11)
getGainingCoalitions(v)
# Output:
#   V1 V2 V3 cVal
# 1  1  0  0    1
# 2  0  1  0    2
# 3  0  0  1    3
# 4  1  1  0    4
# 7  1  1  1   11
```

---

getGapFunctionCoefficients  
*Compute gap function coefficients*

---

**Description**

getGapFunctionCoefficients computes the gap function coefficients for a specified TU game

**Usage**

```
getGapFunctionCoefficients(v)
```

**Arguments**

v                    Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with n players

**Value**

numeric vector containing the gap function coefficients for every coalition

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**References**

Driessen T. (1998) *Cooperative Games, Solutions and Applications*, Springer, p. 57

**Examples**

```
library(CoopGame)
getGapFunctionCoefficients(c(0,0,0,60,48,30,72))
```

---

getkCover                    *Compute k-cover of game*

---

**Description**

getkCover returns the k-cover for a given TU game according to the formula on p. 173 in the book by Driessen. Note that the k-cover does not exist if condition (7.2) on p. 173 in the book by Driessen is not satisfied.

**Usage**

```
getkCover(v, k)
```



**Arguments**

- v                    Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with n players
- k                    An integer specifying k in the k-cover

**Value**

numeric vector containing the k-cover of the given game if the k-cover exists, NULL otherwise

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**References**

Driessen T. (1998) *Cooperative Games, Solutions and Applications*, Springer, p. 173

**Examples**

```
library(CoopGame)
getkCover(c(0,0,0,9,9,12,18),k=1)
```

```
library(CoopGame)
#Example from textbook by Driessen, p. 175, with alpha = 0.6 and k = 2
alpha = 0.6
getkCover(c(0,0,0,alpha,alpha,0,1), k=2)
#[1] 0.0 0.0 0.0 0.6 0.6 0.0 1.0
```

---

```
getMarginalContributions
```

*Compute marginal contributions*

---

**Description**

Calculates the marginal contributions for all permutations of the players

**Usage**

```
getMarginalContributions(v)
```

**Arguments**

- v                    Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with n players

**Value**

a list with given game vector, a matrix of combinations used and a matrix with the marginal contributions

**Author(s)**

Alexandra Tiukkel

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**References**

Peters H. (2015) *Game Theory: A Multi-Leveled Approach*, 2nd Edition, Springer, pp. 156–159

Branzei R., Dimitrov D. and Tijs S. (2006) *Models in cooperative game theory*, Springer, p. 6

**Examples**

```
library(CoopGame)
v=c(0,0,0,1,1,0,1)
getMarginalContributions(v)
```

---

<code>getMinimalRights</code>	<i>Compute minimal rights vector</i>
-------------------------------	--------------------------------------

---

**Description**

Calculates the minimal rights vector.

**Usage**

```
getMinimalRights(v)
```

**Arguments**

`v` Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with  $n$  players

**Value**

Vector of minimal rights of each player

**Author(s)**

Johannes Anwander <anwander.johannes@gmail.com>

Michael Maerz

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

## References

Branzei R., Dimitrov D. and Tijs S. (2006) *Models in cooperative game theory*, Springer, pp. 20–21

## Examples

```
library(CoopGame)
getMinimalRights(c(0,0,0,1,0,1,1))
```

```
library(CoopGame)
v1 <- c(0,0,0,60,60,60,72)
getMinimalRights(v1)
#[1] 48 48 48
```

```
library(CoopGame)
v2 <- c(2,4,5,18,14,9,24)
getMinimalRights(v2)
#[1] 8 4 5
```

---

```
getMinimumWinningCoalitions
```

*Compute minimal winning coalitions in a simple game*

---

## Description

The function `getMinimumWinningCoalitions` identifies all minimal winning coalitions of a specified simple game. These coalitions are characterized by the circumstance that if any player breaks away from them, then the coalition generates no value (then also called a losing coalition) - all players in the coalition can therefore be described as critical players.

## Usage

```
getMinimumWinningCoalitions(v)
```

## Arguments

`v` Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with  $n$  players

## Value

A data frame containing all minimum winning coalitions for a simple game.

## Author(s)

Johannes Anwander <anwander.johannes@gmail.com>

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**References**

- Deegan J. and Packel E.W. (1978) "A new index of power for simple n-person games", Int. Journal of Game Theory 7(2), pp. 151–161
- Peters H. (2015) *Game Theory: A Multi-Leveled Approach*, 2nd Edition, Springer, p. 295
- Bertini C. (2011) "Minimal winning coalition", Encyclopedia of Power, SAGE Publications, pp. 422–423

**Examples**

```
library(CoopGame)
getMinimumWinningCoalitions(v=c(0,0,0,0,0,0,1))
```

```
library(CoopGame)
v=weightedVotingGameVector(n=3,w=c(1,2,3),q=5)
getMinimumWinningCoalitions(v)
# Output:
#   V1 V2 V3 cVal
# 6  0  1  1   1
# => the coalition containing player 2 and 3 is a minimal winning coalition
```

---

```
getNumberOfPlayers    Get number of players
```

---

**Description**

Gets the number of players from a game vector

**Usage**

```
getNumberOfPlayers(v)
```

**Arguments**

*v* Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with *n* players

**Value**

Number of players in the game (specified by game vector *v*)

**Author(s)**

Michael Maerz  
 Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**Examples**

```
library(CoopGame)
maschlerGame=c(0,0,0,60,60,60,72)
getNumberOfPlayers(maschlerGame)
```

---

getPerCapitaExcessCoefficients

*Compute per capita excess coefficients*

---

**Description**

getPerCapitaExcessCoefficients computes the per capita excess coefficients for a specified TU game and an allocation  $x$

**Usage**

```
getPerCapitaExcessCoefficients(v, x)
```

**Arguments**

$v$	Numeric vector of length $2^n - 1$ representing the values of the coalitions of a TU game with $n$ players
$x$	numeric vector containing allocations for each player

**Value**

numeric vector containing the per capita excess coefficients for every coalition

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**Examples**

```
library(CoopGame)
getPerCapitaExcessCoefficients(c(0,0,0,60,48,30,72), c(24,24,24))
```

getPlayersFromBitVector

*Extract players from bit vector*

---

### Description

getPlayersFromBitVector determines players involved in a coalition from a binary vector.

### Usage

```
getPlayersFromBitVector(bitVector)
```

### Arguments

bitVector        represents the binary vector

### Value

playerVector contains the numbers of the players involved in the coalition

### Author(s)

Johannes Anwander <anwander.johannes@gmail.com>

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

### Examples

```
library(CoopGame)
myBitVector <-c(1,0,1,0)
(players<-getPlayersFromBitVector(myBitVector))
```

---

getPlayersFromBMRow    *Extract players from bit matrix row*

---

### Description

getPlayersFromBMRow determines players involved in a coalition from the row of a bit matrix

### Usage

```
getPlayersFromBMRow(bmRow)
```

### Arguments

bmRow            represents the bit matrix row

**Value**

playerVector contains involved players (e.g. c(1,3), see example below for bitIndex=5 and n=3)

**Author(s)**

Johannes Anwander <anwander.johannes@gmail.com>

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**Examples**

```
library(CoopGame)
bm=createBitMatrix(n=3,A=c(0,0,0,1,1,1,2))
getPlayersFromBMRow(bmRow=bm[4,])
```

```
library(CoopGame)
bm=createBitMatrix(n=3,A=c(1:7))
#Corresponding bit matrix:
#      cVal
#[1,] 1 0 0 1
#[2,] 0 1 0 2
#[3,] 0 0 1 3
#[4,] 1 1 0 4
#[5,] 1 0 1 5 <=Specified bit index
#[6,] 0 1 1 6
#[7,] 1 1 1 7

#Determine players from bit matrix row by index 5
players=getPlayersFromBMRow(bmRow=bm[5,])
#Result:
players
#[1] 1 3
```

---

```
getRealGainingCoalitions
```

*Compute real gaining coalitions of game*

---

**Description**

The function getRealGainingCoalitions identifies all real gaining coalitions. Coalition S is a real gaining coalition if for any true subset T of S there holds:  $v(S) > v(T)$

**Usage**

```
getRealGainingCoalitions(v)
```

**Arguments**

`v` Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with  $n$  players

**Value**

A data frame containing all real gaining coalitions.

**Author(s)**

Johannes Anwander <anwander.johannes@gmail.com>

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**References**

Holler M.J. and Li X. (1995) "From public good index to public value. An axiomatic approach and generalization", *Control and Cybernetics* 24, pp. 257–270

Bertini C. and Stach I. (2015) "On Public Values and Power Indices", *Decision Making in Manufacturing and Services* 9(1), pp. 9–25

**Examples**

```
library(CoopGame)
getRealGainingCoalitions(v=c(0,0,0,0,0,0,2))
```

```
library(CoopGame)
v <- c(1,2,3,4,0,0,0)
getRealGainingCoalitions(v)
# Output:
#   V1 V2 V3 cVal
# 1  1  0  0    1
# 2  0  1  0    2
# 3  0  0  1    3
# 4  1  1  0    4
```

---

```
getUnanimityCoefficients
```

*Compute unanimity coefficients of game*

---

**Description**

`getUnanimityCoefficients` calculates to unanimity coefficients of a specified TU game. Note that the unanimity coefficients are also frequently referred to as Harsanyi dividends in the literature.



**Usage**

```
getUnanimityCoefficients(v)
```

**Arguments**

v                      Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with n players

**Value**

numeric vector containing the unanimity coefficients

**Author(s)**

Johannes Anwander <anwander.johannes@gmail.com>

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**References**

Peleg B. and Sudhoelter P. (2007) *Theory of cooperative games*, 2nd Edition, Springer, p. 153

Gilles R. P. (2015) *The Cooperative Game Theory of Networks and Hierarchies*, Springer, pp. 15–17

Shapley L.S. (1953) "A value for n-person games". In: Kuhn, H., Tucker, A.W. (Eds.), *Contributions to the Theory of Games II*, Princeton University Press, pp. 307–317

**Examples**

```
library(CoopGame)
getUnanimityCoefficients(c(0,0,0,60,48,30,72))
```

---

getUtopiaPayoff                      *Compute utopia payoff vector of game*

---

**Description**

getUtopiaPayoff calculates the utopia payoff vector for each player in a TU game. The utopia payoff of player i is the marginal contribution of player i to the grand coalition.

**Usage**

```
getUtopiaPayoff(v)
```

**Arguments**

v                      Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with n players

**Value**

utopia payoffs for each player

**Author(s)**

Johannes Anwander <anwander.johannes@gmail.com>

Michael Maerz

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**References**

Branzei R., Dimitrov D. and Tijs S. (2006) *Models in cooperative game theory*, Springer, p. 31

**Examples**

```
library(CoopGame)
maschlerGame <- c(0,0,0,60,60,60,72)
getUtopiaPayoff(maschlerGame)
```

---

```
getVectorOfPropensitiesToDisrupt
      Compute vector of propensities to disrupt
```

---

**Description**

getVectorOfPropensitiesToDisrupt computes a vector of propensities to disrupt for game vector  $v$  and an allocation  $x$

**Usage**

```
getVectorOfPropensitiesToDisrupt(v, x)
```

**Arguments**

$v$	Numeric vector of length $2^n - 1$ representing the values of the coalitions of a TU game with $n$ players
$x$	numeric vector containing allocations for each player

**Value**

a numerical vector of propensities to disrupt at a given allocation  $x$

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

## References

Littlechild S.C. and Vaidya K.G. (1976) "The propensity to disrupt and the disruption nucleolus of a characteristic function game", *Int. Journal of Game Theory* 5(2), pp. 151–161

Staudacher J. and Anwander J. (2019) "Conditions for the uniqueness of the Gately point for cooperative games", arXiv preprint, arXiv:1901.01485, 10 pages.

## Examples

```
library(CoopGame)
v=c(0,0,0,4,0,3,6)
x=c(2,3,1)
getVectorOfPropensitiesToDisrupt(v,x)
```

---

getWinningCoalitions    *Compute winning coalitions in a simple game*

---

## Description

The function getWinningCoalitions identifies all winning coalitions of a specified simple game.

## Usage

```
getWinningCoalitions(v)
```

## Arguments

v                      Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with n players

## Value

A data frame containing all winning coalitions for a simple game.

## Author(s)

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

## References

Bertini C., Gambarelli G. and Stach I. (2008) "A public help index", In: Braham, M. and Steffen, F. (Eds): *Power, freedom, and voting: Essays in Honour of Manfred J. Holler*, pp. 83–98

Bertini C. and Stach I. (2015) "On Public Values and Power Indices", *Decision Making in Manufacturing and Services* 9(1), pp. 9–25

Stach I. (2016) "Power Measures and Public Goods", In: Nguyen, N.T. and Kowalczyk, R. (Eds.): *Transactions on Computational Collective Intelligence XXIII*, Springer, pp. 99–110

**Examples**

```
library(CoopGame)
getWinningCoalitions(v=c(0,0,0,1,0,1,1))

library(CoopGame)
v=weightedVotingGameVector(n=3,w=c(1,2,3),q=5)
getWinningCoalitions(v)
# Output:
#   V1 V2 V3 cVal
# 6  0  1  1    1
# 7  1  1  1    1
# => the coalition containing player 2 and 3 and
#     the grand coalition are winning coalitions
```

---

```
getZeroNormalizedGameVector
```

*Compute 0-normalized game vector*

---

**Description**

Computes the zero-normalized game for a given game specified by a game vector.

**Usage**

```
getZeroNormalizedGameVector(v)
```

**Arguments**

**v** Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with  $n$  players

**Value**

Numeric vector of length  $(2^n)-1$  representing the zero-normalized game.

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>  
 Johannes Anwander <anwander.johannes@gmail.com>

**References**

Branzei R., Dimitrov D. and Tijs S. (2006) *Models in cooperative game theory*, Springer, p. 9  
 Peleg B. and Sudhoelter P. (2007) *Theory of cooperative games*, 2nd Edition, Springer, p. 11

**Examples**

```
library(CoopGame)
v<-c(1:7)
getZeroNormalizedGameVector(v)
```

---

```
getZeroOneNormalizedGameVector
      Compute 0-1-normalized game vector
```

---

**Description**

Computes the zero-one-normalized game for a given game specified by a game vector.

**Usage**

```
getZeroOneNormalizedGameVector(v)
```

**Arguments**

**v** Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with  $n$  players

**Value**

Numeric vector of length  $(2^n)-1$  representing the zero-one-normalized game.

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>  
Johannes Anwander <anwander.johannes@gmail.com>

**References**

Gilles R. P. (2015) *The Cooperative Game Theory of Networks and Hierarchies*, Springer, p. 18  
Maschler M., Solan E. and Zamir S. (2013) *Game Theory*, Cambridge University Press, p. 670

**Examples**

```
library(CoopGame)
v<-c(1:7)
getZeroOneNormalizedGameVector(v)
```

gloveGame

*Construct a glove game***Description****Create a list containing all information about a specified glove game:**

We have a set of players L with left-hand gloves and a set of players R with right-hand gloves. The worth of a coalition S equals the number of pairs of gloves the members of S can make. Note that the sets L and R have to be disjoint.

**Usage**

```
gloveGame(n, L, R)
```

**Arguments**

n	represents the number of players
L	numeric vector of players owning one left-hand glove each
R	numeric vector of players owning one right-hand glove each

**Value**

A list with four elements representing the glove game (n, L, R, Game vector v)

**Related Functions**

[gloveGameValue](#), [gloveGameVector](#)

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**References**

Peters H. (2015) *Game Theory: A Multi-Leveled Approach*, 2nd Edition, Springer, pp. 155–156

**Examples**

```
library(CoopGame)
gloveGame(n=3,L=c(1,2), R=c(3))

#Example with four players:
#players 1, 2 and 4 hold a left-hand glove each,
#player 3 holds a right-hand glove.
library(CoopGame)
(vv<-gloveGame(n=4,L=c(1,2,4), R=c(3)))
# $n
```

```

#[1] 3

#$L
#[1] 1 2 4
#
#$R
#[1] 3
#
#$v
#[1] 0 0 0 0 0 1 0 1 0 1 1 0 1 1 1

```

---

gloveGameValue      *Compute value of a coalition for a glove game*

---

### Description

**Coalition value for a specified glove game:**

For further information see [gloveGame](#)

### Usage

```
gloveGameValue(S, L, R)
```

### Arguments

S	numeric vector with coalition of players
L	numeric vector of players owning one left-hand glove each
R	numeric vector of players owning one right-hand glove each

### Value

Number of matched pairs of gloves for given coalition S

### Author(s)

Alexandra Tiukkel  
 Johannes Anwander <anwander.johannes@gmail.com>  
 Jochen Staudacher <jochen.staudacher@hs-kempten.de>

### References

Peters H. (2015) *Game Theory: A Multi-Leveled Approach*, 2nd Edition, Springer, pp. 155–156

### Examples

```

library(CoopGame)
gloveGameValue(S=c(1,2), L=c(1,2), R=c(3))

```

---

gloveGameVector      *Compute game vector for glove game*

---

### Description

**Game vector for glove game:**

For further information see [gloveGame](#)

### Usage

```
gloveGameVector(n, L, R)
```

### Arguments

n                    represents the number of players  
L                    numeric vector of players owning one left-hand glove each  
R                    numeric vector of players owning one right-hand glove each

### Value

Game vector of the specified glove game

### Author(s)

Johannes Anwander <[anwander.johannes@gmail.com](mailto:anwander.johannes@gmail.com)>

Jochen Staudacher <[jochen.staudacher@hs-kempten.de](mailto:jochen.staudacher@hs-kempten.de)>

### References

Peters H. (2015) *Game Theory: A Multi-Leveled Approach*, 2nd Edition, Springer, pp. 155–156

### Examples

```
library(CoopGame)  
gloveGameVector(3, L=c(1,2), R=c(3))
```



---

imputationsetVertices *Compute vertices of imputation set*

---

### Description

imputationsetVertices calculates the imputation set vertices for given game vector.

### Usage

```
imputationsetVertices(v)
```

### Arguments

v                    Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with n players

### Value

rows of the matrix are the vertices of the imputation set

### Author(s)

Michael Maerz

Franz Mueller

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

### References

- Peleg B. and Sudhoelter P. (2007) *Theory of cooperative games*, 2nd Edition, Springer, p. 20  
 Maschler M., Solan E. and Zamir S. (2013) *Game Theory*, Cambridge University Press, p. 674  
 Osborne M.J. and Rubinstein A. (1994) *A Course in Game Theory*, MIT Press, p. 278  
 Narahari Y. (2015) *Game Theory and Mechanism Design*, World Scientific Publishing, p. 407

### Examples

```
library(CoopGame)
imputationsetVertices(c(0,0,0,1,1,1,2))
```

```
library(CoopGame)
v = c(2, 4, 5, 18, 24, 9, 24)
```

```
imputationsetVertices(v)
```

```
#      [,1] [,2] [,3]
#[1,]  15   4   5
#[2,]   2  17   5
```

```
#[3,] 2 4 18
```

---

```
is1ConvexGame      Check if game is 1-Convex
```

---

### Description

is1ConvexGame checks if a TU game is 1-convex. A TU game is 1-convex if and only if the following condition holds true: Let  $S$  be a nonempty coalition. Whenever all players outside  $S$  receive their payoffs according to the utopia payoff of the game, then the remaining part of the total savings is at least  $v(S)$ .

### Usage

```
is1ConvexGame(v)
```

### Arguments

$v$  Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with  $n$  players

### Value

TRUE if the game is 1-convex, else FALSE

### Author(s)

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

### References

Driessen T. (1998) *Cooperative Games, Solutions and Applications*, Springer, p. 73

### Examples

```
library(CoopGame)
is1ConvexGame(c(0,0,0,9,9,12,18))

#1-convex game (taken from book by T. Driessen, p. 75)
library(CoopGame)
v=c(0,0,0,9,9,15,18)
is1ConvexGame(v)

#Example of a game which is not 1-convex
library(CoopGame)
v=c(1:7)
```

```
is1ConvexGame(v)
```

---

```
isAdditiveGame      Check if game is additive
```

---

### Description

Checks if a TU game with  $n$  players is additive.

In an additive game for any two disjoint coalitions  $S$  and  $T$  the value of the union of  $S$  and  $T$  equals the sum of the values of  $S$  and  $T$ . In other words, additive games are constant-sum and the imputation set of an additive game consists of exactly one point.

### Usage

```
isAdditiveGame(v)
```

### Arguments

$v$  Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with  $n$  players

### Value

TRUE if the game is additive, else FALSE

### Author(s)

Alexandra Tiukkel

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

### References

Peleg B. and Sudhoelter P. (2007) *Theory of cooperative games*, 2nd Edition, Springer, p. 11

Peters H. (2015) *Game Theory: A Multi-Leveled Approach*, 2nd Edition, Springer, p. 292

Osborne M.J. and Rubinstein A. (1994) *A Course in Game Theory*, MIT Press, p. 261

### Examples

```
library(CoopGame)
isAdditiveGame(c(1,1,1,2,2,2,3))
```

```
#The following game is not additive
library(CoopGame)
v=c(0,0,0,40,50,20,100)
isAdditiveGame(v)
```

```
#The following game is additive
library(CoopGame)
v=c(1,1,1,1, 2,2,2,2,2,2, 3,3,3,3, 4)
isAdditiveGame(v)
```

---

isBalancedGame      *Check if game is balanced*

---

### Description

Checks if a game is balanced. A game is balanced if the core is a nonempty set.

### Usage

```
isBalancedGame(v)
```

### Arguments

v                      Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with n players

### Value

TRUE if the game is balanced, else FALSE

### Author(s)

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

### References

- Bondareva O.N. (1963) "Some applications of linear programming methods to the theory of cooperative games". *Problemy kibernetiki* 10, pp. 119–139
- Shapley L.S. (1967) "On Balanced Sets and Cores". *Naval Research Logistics Quarterly* 14, pp. 453–460
- Peleg B. and Sudhoelter P. (2007) *Theory of cooperative games*, 2nd Edition, Springer, pp. 27–32
- Maschler M., Solan E. and Zamir S. (2013) *Game Theory*, Cambridge University Press, pp. 691–702
- Slikker M. and van den Nouweland A. (2001) *Social and Economic Networks in Cooperative Game Theory*, Springer, pp. 6–7
- Osborne M.J. and Rubinstein A. (1994) *A Course in Game Theory*, MIT Press, pp. 262–263

**Examples**

```
library(CoopGame)
v=c(0,0,0,40,50,20,100)
isBalancedGame(v)

#Example of an unbalanced game with 3 players
library(CoopGame)
v=c(1,1,1,2,3,4,3)
isBalancedGame(v)

#Example of an unbalanced game with 4 players
library(CoopGame)
v=c(0,0,0,0,1,0,0,0,0,3,3,3,3,3,4)
isBalancedGame(v)

#Example of a balanced game with 4 players
library(CoopGame)
v= c(0,0,0,0,1,0,0,0,0,2,2,2,2,2,4)
isBalancedGame(v)
```

---

isConstantSumGame      *Check if game is constant-sum*

---

**Description**

Checks if a TU game with  $n$  players is constant-sum.

In a constant-sum game for any coalition  $S$  the sums of the values of the coalition  $S$  and its complement equal the value of the grand coalition  $N$ .

**Usage**

```
isConstantSumGame(v)
```

**Arguments**

$v$                       Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with  $n$  players

**Value**

TRUE if the game is constant-sum, else FALSE.

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**References**

Peleg B. and Sudhoelster P. (2007) *Theory of cooperative games*, 2nd Edition, Springer, p. 11

**Examples**

```
library(CoopGame)
v=c(0,0,0,2,2,2,2)
isConstantSumGame(v)

#Example of a game that is not constant-sum
library(CoopGame)
v=c(0,0,0,40,30,130,100)
isConstantSumGame(v)

#Another example of a constant-sum game
library(CoopGame)
v=c(1,1,1,2, 2,2,2,2,2,2, 2,3,3,3, 4)
isConstantSumGame(v)
```

---

isConvexGame	<i>Check if game is convex</i>
--------------	--------------------------------

---

**Description**

isConvexGame checks if a TU game is convex. A TU game is convex if and only if each player's marginal contribution to any coalition is monotone nondecreasing with respect to set-theoretic inclusion.

**Usage**

```
isConvexGame(v)
```

**Arguments**

v	Numeric vector of length $2^n - 1$ representing the values of the coalitions of a TU game with n players
---	--

**Value**

TRUE if the game is convex, else FALSE

**Author(s)**

Johannes Anwander <anwander.johannes@gmail.com>  
 Jochen Staudacher <jochen.staudacher@hs-kempten.de>

## References

- Peleg B. and Sudhoelter P. (2007) *Theory of cooperative games*, 2nd Edition, Springer, p. 10
- Peters H. (2015) *Game Theory: A Multi-Leveled Approach*, 2nd Edition, Springer, p. 329
- Maschler M., Solan E. and Zamir S. (2013) *Game Theory*, Cambridge University Press, pp. 717–718
- Osborne M.J. and Rubinstein A. (1994) *A Course in Game Theory*, MIT Press, pp. 260–261

## Examples

```
library(CoopGame)
isConvexGame(c(0,0,0,1,1,1,5))

#Example of a convex game with three players
library(CoopGame)
v=c(0,0,0,1,2,1,4)
isConvexGame(v)

#Example of a nonconvex game
library(CoopGame)
v=c(1:7)
isConvexGame(v)
```

---

<code>isDegenerateGame</code>	<i>Check if game is degenerate</i>
-------------------------------	------------------------------------

---

## Description

Checks if a TU game is degenerate. We call a game essential if the value of the grand coalition is greater than the sum of the values of the singleton coalitions. We call a game degenerate (or inessential), if

$$v(N) = \sum v(i)$$

## Usage

```
isDegenerateGame(v)
```

## Arguments

`v` Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with  $n$  players

**Value**

TRUE if the game is degenerate, else FALSE

**Author(s)**

Michael Maerz

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**Examples**

```
library(CoopGame)
isDegenerateGame(c(1,2,3,4,4,4,6))

#The following game, i.e. the Maschler game, is not degenerate
library(CoopGame)
v1 <- c(0,0,0,60,60,60,72)
isDegenerateGame(v1)

#The following game is also not degenerate
library(CoopGame)
v2 <- c(30,30,15,60,60,60,72)
isDegenerateGame(v2)

#The following game is degenerate
library(CoopGame)
v3 <- c(20,20,32,60,60,60,72)
isDegenerateGame(v3)
```

---

isEssentialGame	<i>Check if game is essential</i>
-----------------	-----------------------------------

---

**Description**

Checks if a TU game with  $n$  players is essential. We call a game essential, if the value of the grand coalition is greater than the sum of the values of the singleton coalitions. A game is essential, if

$$v(N) > \sum v(i)$$

For an essential game the imputation set is nonempty and consists of more than one point.

**Usage**

```
isEssentialGame(v)
```



**Arguments**

`v` Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with  $n$  players

**Value**

TRUE if the game is essential, else FALSE.

**Author(s)**

Michael Maerz

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**References**

Chakravarty S.R., Mitra M. and Sarkar P. (2015) *A Course on Cooperative Game Theory*, Cambridge University Press, p. 23

Gilles R. P. (2015) *The Cooperative Game Theory of Networks and Hierarchies*, Springer, p. 18

**Examples**

```
library(CoopGame)
isEssentialGame(c(1,2,3,4,4,4,7))

# Example of an essential game
library(CoopGame)
v1 <- c(0,0,0,60,60,60,72)
isEssentialGame(v1)

# Example of a game that is not essential
library(CoopGame)
v2 <- c(30,30,15,60,60,60,72)
isEssentialGame(v2)

# Example of a game that is not essential
library(CoopGame)
v3 <- c(20,20,32,60,60,60,72)
isEssentialGame(v3)
```

---

iskConvexGame

*Check if game is k-Convex*


---

**Description**

iskConvexGame checks if a TU game is  $k$ -convex. A TU game is  $k$ -convex if and only if its  $k$ -cover exists and is convex. See section 7.1 of the book by Driessen for more details

**Usage**

```
iskConvexGame(v, k)
```

**Arguments**

v                    Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with n players

k                    An integer specifying k

**Value**

TRUE if the game is k-convex, else FALSE

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**References**

Driessen T. (1998) *Cooperative Games, Solutions and Applications*, Springer, p. 171–178

**Examples**

```
library(CoopGame)
iskConvexGame(v=c(0,0,0,9,9,12,18), k=1)

# Two examples motivated by the book by T. Driessen, p. 175:
#
# The following game is 2-convex
library(CoopGame)
alpha = 0.4
v=c(0,0,0,alpha,alpha,0,1)
iskConvexGame(v,2)

# The following game is not 2-convex
library(CoopGame)
alpha = 0.7
v=c(0,0,0,alpha,alpha,0,1)
iskConvexGame(v,2)
```

---

isMonotonicGame	<i>Check if game is monotonic</i>
-----------------	-----------------------------------

---

**Description**

Checks if a TU game with  $n$  players is monotonic.

For a monotonic game a coalition  $S$  can never obtain a larger value than another coalition  $T$  if  $S$  is contained in  $T$ .

**Usage**

```
isMonotonicGame(v)
```

**Arguments**

$v$  Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with  $n$  players

**Value**

TRUE if the game is monotonic, else FALSE

**Author(s)**

Johannes Anwander <anwander.johannes@gmail.com>

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**References**

Peleg B. and Sudhoelter P. (2007) *Theory of cooperative games*, 2nd Edition, Springer, p. 12

Narahari Y. (2015) *Game Theory and Mechanism Design*, World Scientific Publishing, p. 408

**Examples**

```
library(CoopGame)
isMonotonicGame(c(0,0,0,1,0,1,1))
```

```
#Example of a non-monotonic game
library(CoopGame)
v1=c(4,2,5,2,3,6,10)
isMonotonicGame(v1)
```

```
#Example of a monotonic game
library(CoopGame)
v2=c(2,5,7,10, 9, 13,20)
isMonotonicGame(v2)
```

---

isNonnegativeGame      *Check if game is nonnegative*

---

### Description

isNonnegativeGame checks if a TU game is a nonnegative game. A TU game is a nonnegative game if the game vector does not contain any negative entries.

### Usage

```
isNonnegativeGame(v)
```

### Arguments

v                      Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with n players

### Value

TRUE if the game is nonnegative, else FALSE.

### Author(s)

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

### Examples

```
library(CoopGame)
isNonnegativeGame(c(0,0,0,0.5,0.1,0.4,1))

#Nonnegative game
library(CoopGame)
v1<-c(0,0,0,0,1,1,1)
isNonnegativeGame(v1)

#Example for game which is not nonnegative
library(CoopGame)
v2<-c(0,0,0,0,-1.1,1,2)
isNonnegativeGame(v2)
```

---

isQuasiBalancedGame    *Check if game is quasi-balanced*

---

### Description

Checks if a TU game is quasi-balanced.

A TU game is quasi-balanced if

a) the components of its minimal rights vector are less or equal than the components of its utopia payoff vector

and

b) the sum of the components of its minimal rights vector is less or equal the value of the grand coalition which in turn is less or equal than the sum of the components of its utopia payoff vector.

Note that any balanced game is also quasi-balanced, but not vice versa.

Note that the quasi-balanced games are those games with a non-empty core cover. Note also that quasi-balancedness is sometimes in the literature also referred to as compromise-admissibility.

### Usage

```
isQuasiBalancedGame(v)
```

### Arguments

v                    Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with n players

### Value

TRUE if the game is quasi-balanced, else FALSE.

### Author(s)

Johannes Anwander <anwander.johannes@gmail.com>

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

### References

Branzei R., Dimitrov D. and Tijs S. (2006) *Models in cooperative game theory*, Springer, p. 31

### Examples

```
library(CoopGame)
isQuasiBalancedGame(c(0,0,0,1,1,1,4))
```

```
#Example of a quasi-balanced game:
library(CoopGame)
v1=c(1,1,2,6,8,14,16)
isQuasiBalancedGame(v1)
```

```
#Example of a game which is not quasi-balanced:
library(CoopGame)
v2=c(1:7)
isQuasiBalancedGame(v2)
```

---

```
isSemiConvexGame      Check if game is semiconvex
```

---

### Description

isSemiConvexGame checks if a TU game is semiconvex. A TU game is semiconvex if and only if the following conditions hold true: The gap function of any single player  $i$  is minimal among the gap function values of coalitions  $S$  containing player  $i$ . Also, the gap function itself is required to be nonnegative.

### Usage

```
isSemiConvexGame(v)
```

### Arguments

$v$  Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with  $n$  players

### Value

TRUE if the game is semiconvex, else FALSE.

### Author(s)

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

### References

Driessen T. and Tijs S. (1985) "The tau-value, the core and semiconvex games", Int. Journal of Game Theory 14(4), pp. 229–247  
 Driessen T. (1998) *Cooperative Games, Solutions and Applications*, Springer, p. 76

### Examples

```
library(CoopGame)
isSemiConvexGame(c(0,0,0,1,1,1,4))
```

```
#Example of a semiconvex game
library(CoopGame)
```

```

v1<-c(3,4,5,9,10,11,18)
isSemiConvexGame(v1)

#Example of a game which not semiconvex
library(CoopGame)
v2=c(1:7)
isSemiConvexGame(v2)

```

---

isSimpleGame	<i>Check if game is simple</i>
--------------	--------------------------------

---

### Description

isSimpleGame checks if a TU game is a simple game. A TU game is a simple game in the sense of the book by Peleg and Sudhoelter (2007), p. 16, if and only if the game is monotonic and the values of all coalitions are either 0 or 1.

### Usage

```
isSimpleGame(v)
```

### Arguments

v	Numeric vector of length $2^n - 1$ representing the values of the coalitions of a TU game with n players
---	--

### Value

TRUE if the game is essential, else FALSE.

### Author(s)

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

### References

Peleg B. and Sudhoelter P. (2007) *Theory of cooperative games*, 2nd Edition, Springer, p. 16

### Examples

```

library(CoopGame)
isSimpleGame(c(0,0,0,1,0,1,1))

#Example of a simple game
library(CoopGame)
v1<-c(0,0,0,0,1,1,1)

```

```

isSimpleGame(v1)

#Example of a game which not simple
library(CoopGame)
v2<-c(0,0,0,0,1,1,2)
isSimpleGame(v2)

#Another example of a game which not simple
#according to our definition
library(CoopGame)
v3<-c(1,0,0,0,1,1,1)
isSimpleGame(v3)

```

---

```
isSuperadditiveGame
```

*Check if game is superadditive*

---

### Description

Checks if a TU game with  $n$  players is superadditive.

In a superadditive game for any two disjoint coalitions  $S$  and  $T$  the value of the union of  $S$  and  $T$  is always greater or equal the sum of the values of  $S$  and  $T$ . In other words, the members of any two disjoint coalitions  $S$  and  $T$  will never be discouraged from collaborating.

### Usage

```
isSuperadditiveGame(v)
```

### Arguments

$v$  Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with  $n$  players

### Value

TRUE if the game is superadditive, else FALSE.

### Author(s)

Alexandra Tiukkel  
 Jochen Staudacher <jochen.staudacher@hs-kempten.de>

### References

Peleg B. and Sudhoelter P. (2007) *Theory of cooperative games*, 2nd Edition, Springer, p. 10  
 Peters H. (2015) *Game Theory: A Multi-Leveled Approach*, 2nd Edition, Springer, p. 295  
 Maschler M., Solan E. and Zamir S. (2013) *Game Theory*, Cambridge University Press, p. 671  
 Narahari Y. (2015) *Game Theory and Mechanism Design*, World Scientific Publishing, p. 408



**Examples**

```

library(CoopGame)
isSuperadditiveGame(c(0,0,0,1,1,1,2))

#Example of a superadditive game
library(CoopGame)
v1=c(0,0,0,40,50,20,100)
isSuperadditiveGame(v1)

#Example of a game that is not superadditive
library(CoopGame)
v2=c(0,0,0,40,30,130,100)
isSuperadditiveGame(v2)

#Another example of a superadditive game
library(CoopGame)
v3=c(1,1,1,1, 2,2,2,2,2,2, 3,3,3,3, 4)
isSuperadditiveGame(v3)

```

---

isSymmetricGame	<i>Check if game is symmetric</i>
-----------------	-----------------------------------

---

**Description**

isSymmetricGame checks if a TU game is symmetric. A TU game is symmetric if and only if the values of all coalitions containing the same number of players are identical.

**Usage**

```
isSymmetricGame(v)
```

**Arguments**

v	Numeric vector of length $2^n - 1$ representing the values of the coalitions of a TU game with n players
---	--

**Value**

TRUE if the game is symmetric, else FALSE.

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

## References

- Peleg B. and Sudhoelster P. (2007) *Theory of cooperative games*, 2nd Edition, Springer, p. 12
- Chakravarty S.R., Mitra M. and Sarkar P. (2015) *A Course on Cooperative Game Theory*, Cambridge University Press, p. 26

## Examples

```
library(CoopGame)
isSymmetricGame(c(0,0,0,1,1,1,2))
```

```
#Example of a symmetric game
library(CoopGame)
v1<-c(3,3,3,10,10,10,17)
isSymmetricGame(v1)
```

```
#Example of a game which is not symmetric
library(CoopGame)
v2=c(1:7)
isSymmetricGame(v2)
```

---

```
isWeaklyConstantSumGame
```

*Check if game is weakly constant-sum*

---

## Description

Checks if a TU game with  $n$  players is weakly constant-sum.

In a weakly constant-sum game for any singleton coalition the sums of the values of that singleton coalition and its complement equal the value of the grand coalition  $N$ .

## Usage

```
isWeaklyConstantSumGame(v)
```

## Arguments

$v$  Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with  $n$  players

## Value

TRUE if the game is weakly constant-sum, else FALSE.

## Author(s)

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

## References

Staudacher J. and Anwander J. (2019) "Conditions for the uniqueness of the Gately point for cooperative games", arXiv preprint, arXiv:1901.01485, 10 pages.

## Examples

```
library(CoopGame)
v1=c(0,0,0,2,2,2,2)
isWeaklyConstantSumGame(v1)

#Example of a game that is not weakly constant-sum
library(CoopGame)
v2=c(0,0,0,40,30,130,100)
isWeaklyConstantSumGame(v2)

#Another example of a weakly constant-sum game
library(CoopGame)
v3=c(1,1,1,2, 7,7,7,7,7, 2,3,3,3, 4)
isWeaklyConstantSumGame(v3)
```

---

```
isWeaklySuperadditiveGame
```

*Check if game is weakly superadditive*

---

## Description

Checks if a TU game with  $n$  players is weakly superadditive.

Let  $S$  be a coalition and  $i$  a player not contained in  $S$ . Then the TU game is weakly superadditive if for any  $S$  and any  $i$  the value of the union of  $S$  and  $i$  is greater or equal the sum of the values of  $S$  and  $i$ .

Note that weak superadditivity is equivalent to zero-monotonicity.

## Usage

```
isWeaklySuperadditiveGame(v)
```

## Arguments

$v$  Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with  $n$  players

## Value

TRUE if the game is weakly superadditive, else FALSE.

**Author(s)**

Johannes Anwander <anwander.johannes@gmail.com>

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**References**

Peleg B. and Sudhoelter P. (2007) *Theory of cooperative games*, 2nd Edition, Springer, p. 10

**Examples**

```
library(CoopGame)
isWeaklySuperadditiveGame(c(0,0,0,1,1,1,1))

#Example of a weakly superadditive game
library(CoopGame)
v1=c(1:15)
isWeaklySuperadditiveGame(v1)

#Example of a game which is not weakly superadditive
library(CoopGame)
v2=c(1:5,7,7)
isWeaklySuperadditiveGame(v2)
```

---

johnstonIndex

*Compute Johnston index*

---

**Description**

johnstonIndex calculates the Johnston index for a simple game.

**Usage**

```
johnstonIndex(v)
```

**Arguments**

v                      Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with n players

**Value**

Johnston index for a specified simple game

**Author(s)**

Johannes Anwander <anwander.johannes@gmail.com>

Michael Maerz

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**References**

Johnston R.J. (1978) "On the measurement of power: Some reactions to Laver", Environment and Planning A, pp. 907–914

Chakravarty S.R., Mitra M. and Sarkar P. (2015) *A Course on Cooperative Game Theory*, Cambridge University Press, p. 124

**Examples**

```
library(CoopGame)
johnstonIndex(c(0,0,0,1,0,0,1))

#player 1 has 3 votes
#player 2 has 2 votes
#player 3 has 1 vote
#majority for the decision is 4 (quota)

library(CoopGame)
#function call generating the game vector:
v <- weightedVotingGameVector(n = 3, w = c(3,2,1), q = 4)

johnstonIndex(v)
#[1] 0.6666667 0.1666667 0.1666667
```

---

koenigBraeuningerIndex

*Compute Koenig-Braeuninger index*

---

**Description**

Calculates the Koenig-Braeuninger index for a specified simple TU game. Note that in general the Koenig-Braeuninger index is not an efficient vector, i.e. the sum of its entries is not always 1. Hence no drawing routine for the Koenig-Braeuninger index is provided.

**Usage**

```
koenigBraeuningerIndex(v)
```

**Arguments**

`v` Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with  $n$  players

**Value**

Koenig-Braeuninger index for specified simple game

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**References**

Koenig T. and Braeuninger T. (1998) "The inclusiveness of European decision rules", *Journal of Theoretical Politics* 10(1), pp. 125–142

Nevison C.H., Zicht, B. and Schoepke S. (1978) "A naive approach to the Banzhaf index of power", *Behavioral Science* 23(2), pp. 130–131

Bertini C. and Stach I. (2015) "On Public Values and Power Indices", *Decision Making in Manufacturing and Services* 9(1), pp. 9–25

**Examples**

```
library(CoopGame)
v=c(0,0,0,0,1,1,0,1)
koenigBraeuningerIndex(v)
```

---

```
majoritySingleVetoGame
```

*Construct a weighted majority game with a single veto player*

---

**Description**

**Create a list containing all information about a specified weighted majority game with a single veto player:**

If coalition  $S$  has at least 2 members and if the veto player is part of the coalition it generates a value of 1, otherwise 0.

Note that weighted majority games with a single veto player are always simple games.

**Usage**

```
majoritySingleVetoGame(n, vetoPlayer)
```

**Arguments**

`n` represents the number of players

`vetoPlayer` represents the veto player

**Value**

A list with three elements representing the specified weighted majority game with a single veto player (n, vetoPlayer, Game vector v)

**Related Functions**

[majoritySingleVetoGameValue](#), [majoritySingleVetoGameVector](#)

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**References**

Jackson M.O. (2008) *Social and Economic Networks*, Princeton University Press, p. 415

**Examples**

```
library(CoopGame)
majoritySingleVetoGame(n=3, vetoPlayer=1)
```

---

majoritySingleVetoGameValue

*Compute value of a coalition for a weighted majority game with a single veto player*

---

**Description**

**Coalition value for a weighted majority game with a single veto player:**  
For further information see [majoritySingleVetoGame](#)

**Usage**

```
majoritySingleVetoGameValue(S, vetoPlayer)
```

**Arguments**

S                    numeric vector with coalition of players  
vetoPlayer        represents the veto player

**Value**

1 if vetoPlayer is included in S and S is not a singleton coalition, 0 otherwise

**Author(s)**

Michael Maerz

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**References**

Jackson M.O. (2008) *Social and Economic Networks*, Princeton University Press, p. 415

**Examples**

```
library(CoopGame)
majoritySingleVetoGameValue(S=c(1,2), vetoPlayer=1)
```

---

majoritySingleVetoGameVector

*Compute game vector for a weighted majority game with a single veto player*

---

**Description**

**Game vector for a weighted majority game with a single veto player:**

For further information see [majoritySingleVetoGame](#)

**Usage**

```
majoritySingleVetoGameVector(n, vetoPlayer)
```

**Arguments**

n represents the number of players

vetoPlayer represents the veto player

**Value**

Game Vector where each elements contains 1 if vetoPlayer is included in S and S is not a singleton coalition, 0 otherwise

**Author(s)**

Michael Maerz

**References**

Jackson M.O. (2008) *Social and Economic Networks*, Princeton University Press, p. 415



**Examples**

```
library(CoopGame)
majoritySingleVetoGameVector(n=3, vetoPlayer=1)
```

---

modiclus

*Compute modiclus*

---

**Description**

Calculates the modiclus of a TU game with a non-empty imputation set and  $n$  players. Note that the modiclus is also known as the modified nucleolus in the literature. Due to complexity of modiclus computation we recommend to use this function for at most  $n=11$  players. Note that the modiclus is a member of the set of preimputations.

**Usage**

```
modiclus(v)
```

**Arguments**

$v$  Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with  $n$  players

**Value**

Numeric vector of length  $n$  representing the modiclus (aka modified nucleolus) of the specified TU game.

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

Johannes Anwander <anwander.johannes@gmail.com>

**References**

Peleg B. and Sudhoelter P. (2007) *Theory of cooperative games*, 2nd Edition, Springer, pp. 124–132

Sudhoelter P. (1997) "The Modified Nucleolus. Properties and Axiomatizations", *Int. Journal of Game Theory* 26(2), pp. 147–182

Sudhoelter P. (1996) "The Modified Nucleolus as Canonical Representation of Weighted Majority Games", *Mathematics of Operations Research* 21(3), pp. 734–756

**Examples**

```
library(CoopGame)
modiclus(c(1, 1, 1, 2, 3, 4, 5))
```

```
library(CoopGame)
modiclus(c(0, 0, 0, 0, 5, 5, 8, 9, 10, 8, 13, 15, 16, 17, 21))
#[1] 4.25 5.25 5.75 5.75
```

---

nevisonIndex

---

*Compute Nevison index*


---

**Description**

Calculates the Nevison index for a specified simple TU game. Note that in general the Nevison index is not an efficient vector, i.e. the sum of its entries is not always 1. Hence no drawing routine for the Nevison index is provided.

**Usage**

```
nevisonIndex(v)
```

**Arguments**

**v** Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with  $n$  players

**Value**

Nevison index for a specified simple game

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**References**

Nevison, H. (1979) "Structural power and satisfaction in simple games", In: Applied Game Theory, Springer, pp. 39–57

**Examples**

```
library(CoopGame)
v=c(0,0,0,0,1,1,0,1)
nevisonIndex(v)
```

---

 nonNormalizedBanzhafIndex

*Compute non-normalized Banzhaf index*


---

### Description

non-normalized Banzhaf index for a specified simple game, see formula (7.5) on p. 119 of the book by Chakravarty, Mitra and Sarkar

### Usage

```
nonNormalizedBanzhafIndex(v)
```

### Arguments

`v` Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with `n` players

### Value

The return value is a vector which contains the non-normalized Banzhaf index for each player.

### Author(s)

Johannes Anwander <anwander.johannes@gmail.com>

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

### References

Chakravarty S.R., Mitra M. and Sarkar P. (2015) *A Course on Cooperative Game Theory*, Cambridge University Press, pp. 118–119

Bertini C. and Stach I. (2011) "Banzhaf voting power measure", *Encyclopedia of Power*, SAGE Publications, pp. 54–55

### Examples

```
library(CoopGame)
nonNormalizedBanzhafIndex(dictatorGameVector(n=3, dictator=1))
```

```
library(CoopGame)
v<-weightedVotingGameVector(n=4,w=c(8,6,4,2),q=c(12))
nonNormalizedBanzhafIndex(v)
#[1] 0.625 0.375 0.375 0.125
```

```
library(CoopGame)
v<- apexGameVector(n = 4,apexPlayer=3)
nonNormalizedBanzhafIndex(v)
```

```
#[1] 0.25 0.25 0.75 0.25

library(CoopGame)
#N=c(1,2,3), w=(50,49,1), q=51
v=weightedVotingGameVector(n=3, w=c(50,49,1),q=51)
nonNormalizedBanzhafIndex(v)
#[1] 0.75 0.25 0.25

library(CoopGame)
v<-weightedVotingGameVector(n=3,w=c(50,30,20),q=c(67))
nonNormalizedBanzhafIndex(v)
#[1] 0.75 0.25 0.25
```

---

normalizedBanzhafIndex

*Compute normalized Banzhaf index*

---

### Description

Normalized Banzhaf index for a specified simple game, see formula (7.6) on p. 119 of the book by Chakravarty, Mitra and Sarkar

### Usage

```
normalizedBanzhafIndex(v)
```

### Arguments

**v** Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with  $n$  players

### Value

The return value is a numeric vector which contains the normalized Banzhaf index for each player.

### Author(s)

Johannes Anwander <anwander.johannes@gmail.com>  
 Jochen Staudacher <jochen.staudacher@hs-kempten.de>

### References

Peters H. (2015) *Game Theory: A Multi-Leveled Approach*, 2nd Edition, Springer, pp. 367–370  
 Chakravarty S.R., Mitra M. and Sarkar P. (2015) *A Course on Cooperative Game Theory*, Cambridge University Press, pp. 118–119  
 Bertini C. and Stach I. (2011) "Banzhaf voting power measure", *Encyclopedia of Power*, SAGE Publications, pp. 54–55

**Examples**

```
library(CoopGame)
normalizedBanzhafIndex(dictatorGameVector(n=3, dictator=1))
```

```
library(CoopGame)
v<-weightedVotingGameVector(n=4,w=c(8,6,4,2),q=c(12))
normalizedBanzhafIndex(v)
#[1] 0.41666667 0.25000000 0.25000000 0.08333333
```

```
library(CoopGame)
v<- apexGameVector(n = 4,apexPlayer=3)
normalizedBanzhafIndex(v)
#[1] 0.1666667 0.1666667 0.5000000 0.1666667
```

```
library(CoopGame)
#N=c(1,2,3), w=(50,49,1), q=51
v=weightedVotingGameVector(n=3, w=c(50,49,1),q=51)
normalizedBanzhafIndex(v)
#[1] 0.6 0.2 0.2
```

```
library(CoopGame)
v<-weightedVotingGameVector(n=3,w=c(50,30,20),q=c(67))
normalizedBanzhafIndex(v)
#[1] 0.6 0.2 0.2
```

---

normalizedBanzhafValue

*Compute normalized Banzhaf value*

---

**Description**

normalizedBanzhafValue computes the normalized Banzhaf value for a specified TU game. The corresponding formula can e.g. be found in the article by Stach (2017), p. 77.

**Usage**

```
normalizedBanzhafValue(v)
```

**Arguments**

v                    Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with n players

**Value**

The return value is a numeric vector which contains the normalized Banzhaf value for each player.

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**References**

- Gambarelli G. (2011) "Banzhaf value", Encyclopedia of Power, SAGE Publications, pp. 53–54
- Stach I. (2017) "Sub-Coalitional Approach to Values", In: Nguyen, N.T. and Kowalczyk, R. (Eds.): Transactions on Computational Collective Intelligence XXVI, Springer, pp. 74–86

**Examples**

```
library(CoopGame)
normalizedBanzhafValue(c(0,0,0,1,2,3,6))

#Example from paper by Gambarelli (2011)
library(CoopGame)
v=c(0,0,0,1,2,1,3)
normalizedBanzhafValue(v)
#[1] 1.1538462 0.6923077 1.1538462
#Expected Result: 15/13 9/13 15/13
```

---

nucleolus

*Compute nucleolus*

---

**Description**

Computes the nucleolus of a TU game with a non-empty imputation set and  $n$  players. Note that the nucleolus is a member of the imputation set.

**Usage**

```
nucleolus(v)
```

**Arguments**

$v$  Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with  $n$  players

**Value**

Numeric vector of length  $n$  representing the nucleolus.

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

Johannes Anwander <anwander.johannes@gmail.com>

Daniel Gebele <daniel.a.gebele@stud.hs-kempten.de>

**References**

Schmeidler D. (1969) "The nucleolus of a characteristic function game", SIAM Journal on applied mathematics 17(6), pp. 1163–1170

Kohlberg E. (1971) "On the nucleolus of a characteristic function game", SIAM Journal on applied mathematics 20(1), pp. 62–66

Kopelowitz A. (1967) "Computation of the kernels of simple games and the nucleolus of n-person games", Technical Report, Department of Mathematics, The Hebrew University of Jerusalem, 45 pages.

Megiddo N. (1974) "On the nonmonotonicity of the bargaining set, the kernel and the nucleolus of a game", SIAM Journal on applied mathematics 27(2), pp. 355–358

Peleg B. and Sudhoelter P. (2007) *Theory of cooperative games*, 2nd Edition, Springer, pp. 82–86

**Examples**

```
library(CoopGame)
nucleolus(c(1, 1, 1, 2, 3, 4, 5))
```

```
library(CoopGame)
nucleolus(c(0, 0, 0, 0, 5, 5, 8, 9, 10, 8, 13, 15, 16, 17, 21))
#[1] 3.5 4.5 5.5 7.5
```

```
#Final example:
#Estate division problem from Babylonian Talmud with E=300,
#see e.g. seminal paper by Aumann & Maschler from 1985 on
#'Game Theoretic Analysis of a Bankruptcy Problem from the Talmud'
library(CoopGame)
v<-bankruptcyGameVector(n=3,d=c(100,200,300),E=300)
nucleolus(v)
#[1] 50 100 150
```

---

perCapitaNucleolus      *Compute per capita nucleolus*

---

**Description**

perCapitaNucleolus calculates the per capita nucleolus for a TU game with a non-empty imputation set specified by a game vector. Note that the per capita nucleolus is a member of the imputation set.

**Usage**

```
perCapitaNucleolus(v)
```

**Arguments**

`v` Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with  $n$  players

**Value**

per capita nucleolus for a specified TU game with  $n$  players

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>  
 Johannes Anwander <anwander.johannes@gmail.com>

**References**

Young H.P. (1985) "Monotonic Solutions of cooperative games", Int. Journal of Game Theory 14(2), pp. 65–72

**Examples**

```
library(CoopGame)
perCapitaNucleolus(c(1, 1, 1, 2, 3, 4, 5))

#Example from YOUNG 1985, p. 68
v<-costSharingGameVector(n=3,C=c(15,20,55,35,61,65,78))
perCapitaNucleolus(v)
#[1] 0.6666667 1.1666667 10.1666667
```

---

Prenucleolus

*Compute prenucleolus*

---

**Description**

Computes the prenucleolus of a TU game with  $n$  players. Note that the prenucleolus is a member of the set of preimputations.

**Usage**

```
prenucleolus(v)
```



**Arguments**

`v` Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with  $n$  players

**Value**

Numeric vector of length  $n$  representing the prenucleolus.

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

Johannes Anwander <anwander.johannes@gmail.com>

Daniel Gebele <daniel.a.gebele@stud.hs-kempten.de>

**References**

Peleg B. and Sudhoelter P. (2007) *Theory of cooperative games*, 2nd Edition, Springer, pp. 107–132

**Examples**

```
library(CoopGame)
prenucleolus(c(1, 1, 1, 2, 3, 4, 5))
```

```
#Example 5.5.12 from Peleg/Sudhoelter, p. 96
library(CoopGame)
prenucleolus(c(0,0,0,10,0,0,2))
#Output
#[1] 3 3 -4
#In the above example nucleolus and prenucleolus do not coincide!
```

```
library(CoopGame)
prenucleolus(c(0, 0, 0, 0, 5, 5, 8, 9, 10, 8, 13, 15, 16, 17, 21))
# [1] 3.5 4.5 5.5 7.5
```

```
#Final example:
#Estate division problem from Babylonian Talmud with E=200,
#see e.g. seminal paper by Aumann & Maschler from 1985 on
# 'Game Theoretic Analysis of a Bankruptcy Problem from the Talmud'
library(CoopGame)
v<-bankruptcyGameVector(n=3,d=c(100,200,300),E=200)
prenucleolus(v)
#[1] 50 75 75
#Note that nucleolus and prenucleolus need to coincide for the above game
```

---

propensityToDisrupt    *Compute propensity to disrupt*

---

### Description

propensityToDisrupt for calculating the propensity of disrupt for game vector  $v$ , an allocation  $x$  and a specified coalition  $S$

### Usage

```
propensityToDisrupt(v, x, S)
```

### Arguments

$v$	Numeric vector of length $2^n - 1$ representing the values of the coalitions of a TU game with $n$ players
$x$	numeric vector containing allocations for each player
$S$	numeric vector with coalition of players

### Value

propensity to disrupt as numerical value

### Author(s)

Johannes Anwander <anwander.johannes@gmail.com>  
Jochen Staudacher <jochen.staudacher@hs-kempten.de>

### References

Littlechild S.C. and Vaidya K.G. (1976) "The propensity to disrupt and the disruption nucleolus of a characteristic function game", Int. Journal of Game Theory 5(2), pp. 151–161

Staudacher J. and Anwander J. (2019) "Conditions for the uniqueness of the Gately point for cooperative games", arXiv preprint, arXiv:1901.01485, 10 pages.

### Examples

```
library(CoopGame)
v=c(0,0,0,4,0,3,6)
x=c(2,3,1)
propensityToDisrupt(v,x,S=c(1))
```

---

proportionalNucleolus *Compute proportional nucleolus*

---

### Description

proportionalNucleolus calculates the proportional nucleolus for a TU game with a non-empty imputation set and  $n$  players specified by game vector. Note that the proportional nucleolus is a member of the imputation set.

### Usage

```
proportionalNucleolus(v)
```

### Arguments

$v$  Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with  $n$  players

### Value

proportional nucleolus for specified TU game with  $n$  players

### Author(s)

Jochen Staudacher <jochen.staudacher@hs-kempten.de>  
Johannes Anwander <anwander.johannes@gmail.com>

### References

Young H. P., Okada N. and Hashimoto, T. (1982) "Cost allocation in water resources development", Water resources research 18(3), pp. 463–475

### Examples

```
library(CoopGame)
v<-c(0,0,0,48,60,72,140)
proportionalNucleolus(v)
```

---

publicGoodIndex      *Compute Public Good index*

---

### Description

Calculates the Public Good index (aka Holler index) for a specified simple game.

### Usage

```
publicGoodIndex(v)
```

### Arguments

`v`                      Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with  $n$  players

### Value

The return value is a vector containing the Public Good index

### Author(s)

Johannes Anwander <anwander.johannes@gmail.com>

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

### References

Holler M.J. and Packel E.W. (1983) "Power, luck and the right index", *Zeitschrift fuer Nationaloekonomie* 43(1), pp. 21–29

Holler M.J. (1982) "Forming coalitions and measuring voting power", *Political Studies* 30(2), pp. 262–271

Holler M. (2011) "Public Goods index", *Encyclopedia of Power*, SAGE Publications, pp. 541–542

### Examples

```
library(CoopGame)
publicGoodIndex(v=c(0,0,0,1,1,0,1))
```

```
#Example from Holler (2011) illustrating paradox of weighted voting
library(CoopGame)
v=weightedVotingGameVector(n=5,w=c(35,20,15,15,15), q=51)
publicGoodIndex(v)
#[1] 0.2666667 0.1333333 0.2000000 0.2000000 0.2000000
```

---

publicGoodValue      *Compute (normalized) Public Good value*

---

### Description

Calculates the (normalized) Public Good value for a specified nonnegative TU game. Note that the normalized Public Good value is sometimes also referred to as Holler value in the literature. Our function implements the formula from Definition 5.4, p. 19, in the paper by Bertini and Stach from 2015.

### Usage

```
publicGoodValue(v)
```

### Arguments

v                      Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with n players

### Value

Public Good value for specified nonnegative TU game

### Author(s)

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

### References

Holler M.J. and Li X. (1995) "From public good index to public value. An axiomatic approach and generalization", Control and Cybernetics 24, pp. 257–270

Bertini C. and Stach I. (2015) "On Public Values and Power Indices", Decision Making in Manufacturing and Services 9(1), pp. 9–25

### Examples

```
library(CoopGame)
v=c(0,0,0,0,0.7,11,0,15)
publicGoodValue(v)
```

---

publicHelpChiIndex      *Compute Public Help index Chi*

---

### Description

Calculates the Public Help index Chi for a specified simple TU game. Note that the greek letter Xi (instead of Chi) was used in the original paper by Bertini and Stach (2015).

### Usage

```
publicHelpChiIndex(v)
```

### Arguments

v                      Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with n players

### Value

Public Help index Chi for specified simple game

### Author(s)

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

### References

Bertini C. and Stach I. (2015) "On Public Values and Power Indices", Decision Making in Manufacturing and Services 9(1), pp. 9–25

Stach I. (2016) "Power Measures and Public Goods", In: Nguyen, N.T. and Kowalczyk, R. (Eds.): Transactions on Computational Collective Intelligence XXIII, Springer, pp. 99–110

### Examples

```
library(CoopGame)
publicHelpChiIndex(v=c(0,0,0,0,1,0,1))
```

```
#Example from original paper by Stach (2016), p. 105:
```

```
library(CoopGame)
v=c(0,0,0,0,1,1,0,1)
publicHelpChiIndex(v)
#result: 0.4583333 0.2708333 0.2708333
```

```
#Second example from original paper by Stach (2016), p. 105:
```

```
library(CoopGame)
v=c(0,0,0,0,0,1,1,0,0,0,0,0,1,1,1,0,1)
publicHelpChiIndex(v)
#result: 0.3981481 0.2376543 0.2376543 0.1265432
```

---

publicHelpChiValue      *Compute (normalized) Public Help value Chi*

---

**Description**

Calculates the (normalized) Public Help value Chi by Bertini & Stach (2015) for a nonnegative TU game. Note that the greek letter Xi (instead of Chi) was used in the original paper by Bertini and Stach (2015).

**Usage**

```
publicHelpChiValue(v)
```

**Arguments**

v                      Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with n players

**Value**

Public Help value Chi for specified nonnegative TU game

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**References**

Bertini C. and Stach I. (2015) "On Public Values and Power Indices", Decision Making in Manufacturing and Services 9(1), pp. 9–25

**Examples**

```
library(CoopGame)
v=c(0,0,0,2,2,0,2)
publicHelpChiValue(v)
```

---

publicHelpIndex      *Compute Public Help index Theta*

---

### Description

Calculates the Public Help index Theta for a specified simple TU game. Note that the Public Help index Theta goes back to the paper by Bertini, Gambarelli and Stach (2008) and is frequently simply referred to as Public Help index in the literature.

### Usage

```
publicHelpIndex(v)
```

### Arguments

**v**                      Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with  $n$  players

### Value

Public Help index Theta for specified simple game

### Author(s)

Johannes Anwander <anwander.johannes@gmail.com>  
 Jochen Staudacher <jochen.staudacher@hs-kempten.de>

### References

Bertini C., Gambarelli G. and Stach I. (2008) "A public help index", In: Braham, M. and Steffen, F. (Eds): Power, freedom, and voting: Essays in Honour of Manfred J. Holler, pp. 83–98

Bertini C. and Stach I. (2015) "On Public Values and Power Indices", Decision Making in Manufacturing and Services 9(1), pp. 9–25

Stach I. (2016) "Power Measures and Public Goods", In: Nguyen, N.T. and Kowalczyk, R. (Eds.): Transactions on Computational Collective Intelligence XXIII, Springer, pp. 99–110

### Examples

```
library(CoopGame)
publicHelpIndex(v=c(0,0,0,0,1,0,1))

#Example from paper by Stach (2016), p. 105:
library(CoopGame)
v=c(0,0,0,1,1,0,1)
publicHelpIndex(v)
#result: 0.4285714 0.2857143 0.2857143
```



```
#Second example from paper by Stach (2016), p. 105:
library(CoopGame)
v=c(0,0,0,0,1,1,0,0,0,0,1,1,1,0,1)
publicHelpIndex(v)
#result: 0.3529412 0.2352941 0.2352941 0.1764706
```

---

publicHelpValue      *Compute Public Help value Theta*

---

### Description

publicHelpValue calculates the (normalized) Public Help value Theta for a specified nonnegative TU game. Our function implements the formula from Definition 5.7, p. 20, in the paper by Bertini and Stach from 2015.

### Usage

```
publicHelpValue(v)
```

### Arguments

v                      Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with n players

### Value

Public Help value Theta for specified nonnegative TU game

### Author(s)

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

### References

Bertini C. and Stach I. (2015) "On Public Values and Power Indices", Decision Making in Manufacturing and Services 9(1), pp. 9–25

### Examples

```
library(CoopGame)
v=c(0,0,0,0.7,11,0,15)
publicHelpValue(v)
```

---

 raeIndex

 Compute Rae index
 

---

### Description

raeIndex calculates the Rae index for a specified simple TU game. Note that in general the Rae index is not an efficient vector, i.e. the sum of its entries is not always 1. Hence no drawing routine for the Rae index is provided.

### Usage

```
raeIndex(v)
```

### Arguments

v                    Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with n players

### Value

Rae index for specified simple game

### Author(s)

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

### References

Rae, D.W. (1969) "Decision-rules and individual values in constitutional choice", American Political Science Review 63(1), pp. 40–56

Chakravarty S.R., Mitra M. and Sarkar P. (2015) *A Course on Cooperative Game Theory*, Cambridge University Press, pp. 119–120

### Examples

```
library(CoopGame)
v=c(0,0,0,1,1,0,1)
raeIndex(v)
```

```
library(CoopGame)
v=c(0,0,0,0,1,1,0,0,0,0,1,1,1,0,1)
raeIndex(v)
#result: [1] 0.875 0.625 0.625 0.500
```

---

rawBanzhafIndex	<i>Compute raw Banzhaf Index</i>
-----------------	----------------------------------

---

**Description**

Raw Banzhaf Index for a specified simple game, see formula (7.4) on p. 118 of the book by Chakravarty, Mitra and Sarkar

**Usage**

```
rawBanzhafIndex(v)
```

**Arguments**

`v` Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with `n` players

**Value**

The return value is a numeric vector which contains the raw Banzhaf index for each player.

**Author(s)**

Johannes Anwander <anwander.johannes@gmail.com>

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**References**

Chakravarty S.R., Mitra M. and Sarkar P. (2015) *A Course on Cooperative Game Theory*, Cambridge University Press, pp. 118–119

**Examples**

```
library(CoopGame)
rawBanzhafIndex(apexGameVector(n=3, apexPlayer=1))
```

```
v<- apexGameVector(n = 4, apexPlayer=3)
rawBanzhafIndex(v)
#[1] 2 2 6 2
```

```
#N=c(1,2,3), w=(50,49,1), q=51
v=weightedVotingGameVector(n=3, w=c(50,49,1),q=51)
rawBanzhafIndex(v)
#[1] 3 1 1
```

```
v<-weightedVotingGameVector(n=3,w=c(50,30,20),q=c(67))
rawBanzhafIndex(v)
#[1] 3 1 1
```

---

rawBanzhafValue      *Compute raw Banzhaf Value*

---

**Description**

raw Banzhaf Value, i.e. the Banzhaf Value without the division by the scaling factor  $2^{(n-1)}$

**Usage**

```
rawBanzhafValue(v)
```

**Arguments**

v                      Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with n players

**Value**

The return value is a numeric vector which contains the raw Banzhaf value for each player.

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**References**

Chakravarty S.R., Mitra M. and Sarkar P. (2015) *A Course on Cooperative Game Theory*, Cambridge University Press, pp. 118–119

**Examples**

```
library(CoopGame)
v = c(0,0,0,1,1,2,5)
rawBanzhafValue(v)
```

```
library(CoopGame)
v = c(0,0,0,2,2,3,5)
rawBanzhafValue(v)
#[1] 6 8 8
```

---

reasonableSetVertices *Compute vertices of reasonable set*

---

### Description

Calculates the vertices of the reasonable set for given game vector.

### Usage

```
reasonableSetVertices(v)
```

### Arguments

v                    Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with n players

### Value

rows of the matrix are the vertices of the reasonable set

### Author(s)

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

### References

- Milnor J.W. (1953) *Reasonable Outcomes for N-person Games*, Rand Corporation, Research Memorandum RM 916.
- Branzei R., Dimitrov D. and Tijs S. (2006) *Models in cooperative game theory*, Springer, p. 21
- Chakravarty S.R., Mitra M. and Sarkar P. (2015) *A Course on Cooperative Game Theory*, Cambridge University Press, pp. 43–44
- Gerard-Varet L.A. and Zamir S. (1987) "Remarks on the reasonable set of outcomes in a general coalition function form game", *Int. Journal of Game Theory* 16(2), pp. 123–143

### Examples

```
library(CoopGame)
reasonableSetVertices(c(0,0,0,1,1,1,2))
```

```
library(CoopGame)
v <- c(0,0,0,3,3,3,6)
reasonableSetVertices(v)
#      [,1] [,2] [,3]
# [1,]   3   0   3
# [2,]   0   3   3
# [3,]   3   3   0
```

---

shapleyShubikIndex      *Compute Shapley-Shubik index*

---

### Description

Calculates the Shapley-Shubik index for a specified simple game with  $n$  players. Note that no separate drawing routine for the Shapley-Shubik index is provide as users can always resort to [drawShapleyValue](#)

### Usage

```
shapleyShubikIndex(v)
```

### Arguments

$v$                       Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with  $n$  players

### Value

Shapley-Shubik index for given simple game

### Author(s)

Alexandra Tiukkel  
 Jochen Staudacher <jochen.staudacher@hs-kempten.de>

### References

Shapley L.S. and Shubik M. (1954) "A method for evaluating the distribution of power in a committee system". *American political science review* 48(3), pp. 787–792

Shapley L.S. (1953) "A value for  $n$ -person games". In: Kuhn, H., Tucker, A.W. (Eds.), *Contributions to the Theory of Games II*, Princeton University Press, pp. 307–317

Peters H. (2015) *Game Theory: A Multi-Leveled Approach*, 2nd Edition, Springer, pp. 156–159

Maschler M., Solan E. and Zamir S. (2013) *Game Theory*, Cambridge University Press, pp. 748–781

Stach I. (2011) "Shapley-Shubik index", *Encyclopedia of Power*, SAGE Publications, pp. 603–606

### Examples

```
library(CoopGame)
shapleyShubikIndex(v=c(0,0,0,0,1,0,1))

#Example from Stach (2011):
library(CoopGame)
v=weightedVotingGameVector(n=4,q=50,w=c(10,10,20,30))
```

```
shapleyShubikIndex(v)
#[1] 0.08333333 0.08333333 0.25000000 0.58333333
```

---

shapleyValue	<i>Compute Shapley value</i>
--------------	------------------------------

---

### Description

Calculates the Shapley value for  $n$  players with formula from Lloyd Shapley.

### Usage

```
shapleyValue(v)
```

### Arguments

$v$	Numeric vector of length $2^n - 1$ representing the values of the coalitions of a TU game with $n$ players
-----	--

### Value

Shapley value for given game vector with  $n$  players

### Author(s)

Alexandra Tiukkel  
Jochen Staudacher <jochen.staudacher@hs-kempten.de>

### References

- Shapley L.S. (1953) "A value for  $n$ -person games". In: Kuhn, H., Tucker, A.W. (Eds.), *Contributions to the Theory of Games II*, Princeton University Press, pp. 307–317
- Aumann R.J. (2010) "Some non-superadditive games, and their Shapley values, in the Talmud", *Int. Journal of Game Theory* 39(1), pp. 3–10
- Peters H. (2015) *Game Theory: A Multi-Leveled Approach*, 2nd Edition, Springer, pp. 156–159
- Maschler M., Solan E. and Zamir S. (2013) *Game Theory*, Cambridge University Press, pp. 748–781
- Bertini C. (2011) "Shapley value", *Encyclopedia of Power*, SAGE Publications, p. 600–603

**Examples**

```

library(CoopGame)
shapleyValue(v=c(0,0,0,1,2,3,7.5))

#Example of a non-superadditive game,
#i.e. the inheritance problem due to Ibn Ezra (1146),
#from paper by Robert Aumann from 2010 on
#'Some non-superadditive games, and their Shapley values, in the Talmud'
library(CoopGame)
Aumann2010Example<-c(120,60,40,30,120,120,120,60,60,40,120,120,120,60,120)
shapleyValue(Aumann2010Example)
#[1] 80.83333 20.83333 10.83333 7.50000

```

---

simplifiedModiclus      *Compute simplified modiclus*

---

**Description**

Computes the simplified modiclus of a TU game with a non-empty imputation set and  $n$  players. Note that the simplified modiclus is a member of the set of preimputations.

**Usage**

```
simplifiedModiclus(v)
```

**Arguments**

$v$                       Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with  $n$  players

**Value**

Numeric vector of length  $n$  representing the simplified modiclus of the specified TU game.

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>  
 Johannes Anwander <anwander.johannes@gmail.com>

**References**

Tarashnina S. (2011) "The simplified modified nucleolus of a cooperative TU-game", TOP 19(1), pp. 150–166



**Examples**

```

library(CoopGame)
simplifiedModiclus(c(0, 0, 0, 1, 1, 0, 1))

#Second example:
#Estate division problem from Babylonian Talmud with E=100,
#see e.g. seminal paper by Aumann & Maschler from 1985 on
#'Game Theoretic Analysis of a Bankruptcy Problem from the Talmud'
library(CoopGame)
v<-bankruptcyGameVector(n=3,d=c(100,200,300),E=100)
simplifiedModiclus(v)
#[1] 33.33333 33.33333 33.33333

```

---

stopOnInconsistentEstateAndClaimsVector

*Parameter Function stopOnInconsistentEstateAndClaimsVector*


---

**Description**

stopOnInconsistentEstateAndClaimsVector checks if sum of claims is greater or equal estate (in bankruptcy games). Calculation stops with an error message if claims vector and estate are inconsistent.

**Usage**

```
stopOnInconsistentEstateAndClaimsVector(paramCheckResult, E, d)
```

**Arguments**

paramCheckResult  
list object for check result with list element 'errCode' for the error code and 'errMessage' for the error message.

E  
is the value of the estate in a bankruptcy game

d  
numeric vector which contains the claims of each player in a bankruptcy game

**Error Code Ranges**

Error codes and messages shown to user if error on parameter check occurs

<b>Error Code</b>	<b>Message</b>
1170	Estate E must be less or equal the sum of claims!

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**See Also**

Other ParameterChecks\_CoopGame: [getEmptyParamCheckResult\(\)](#), [stopOnInvalidAllocation\(\)](#), [stopOnInvalidBoolean\(\)](#), [stopOnInvalidClaimsVector\(\)](#), [stopOnInvalidCoalitionS\(\)](#), [stopOnInvalidDictator\(\)](#), [stopOnInvalidEstate\(\)](#), [stopOnInvalidGameVector\(\)](#), [stopOnInvalidGrandCoalitionN\(\)](#), [stopOnInvalidIndex\(\)](#), [stopOnInvalidLeftRightGloveGame\(\)](#), [stopOnInvalidNChooseB\(\)](#), [stopOnInvalidNumberOfPlayers\(\)](#), [stopOnInvalidNumber\(\)](#), [stopOnInvalidQuota\(\)](#), [stopOnInvalidVetoPlayer\(\)](#), [stopOnInvalidWeightVector\(\)](#), [stopOnParamCheckError\(\)](#)

**Examples**

```
library(CoopGame)
paramCheckResult=getEmptyParamCheckResult()
consistentClaims= c(26,27,55,57)
consistentE = 110
stopOnInconsistentEstateAndClaimsVector(paramCheckResult, d=consistentClaims, E=consistentE)
```

---

stopOnInvalidAllocation

*Parameter Function stopOnInvalidAllocation*

---

**Description**

stopOnInvalidAllocation checks if allocation is specified correctly. Validation result gets stored to object paramCheckResult in case an error occurred and causes calculation to stop.

**Usage**

```
stopOnInvalidAllocation(paramCheckResult, x, n = NULL, v = NULL)
```

**Arguments**

paramCheckResult	list object for check result with list element 'errCode' for the error code and 'errMessage' for the error message.
x	numeric vector containing allocations for each player
n	represents the number of players
v	Numeric vector of length $2^n - 1$ representing the values of the coalitions of a TU game with n players

**Error Code Ranges**

Error codes and messages shown to user if error on parameter check occurs

Error Code	Message
1100	Allocation 'x' is NULL
1101	Allocation 'x' is not of type numeric.
1102	Allocation 'x' has wrong number of elements as compared to number of players.
1103	Allocation is inconsistent with game vector.

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>  
 Johannes Anwander <anwander.johannes@gmail.com>

**See Also**

Other ParameterChecks\_CoopGame: [getEmptyParamCheckResult\(\)](#), [stopOnInconsistentEstateAndClaimsVector\(\)](#), [stopOnInvalidBoolean\(\)](#), [stopOnInvalidClaimsVector\(\)](#), [stopOnInvalidCoalitionS\(\)](#), [stopOnInvalidDictator\(\)](#), [stopOnInvalidEstate\(\)](#), [stopOnInvalidGameVector\(\)](#), [stopOnInvalidGrandCoalitionN\(\)](#), [stopOnInvalidIndex\(\)](#), [stopOnInvalidLeftRightGloveGame\(\)](#), [stopOnInvalidNChooseB\(\)](#), [stopOnInvalidNumberOfPlayers\(\)](#), [stopOnInvalidNumber\(\)](#), [stopOnInvalidQuota\(\)](#), [stopOnInvalidVetoPlayer\(\)](#), [stopOnInvalidWeightVector\(\)](#), [stopOnParamCheckError\(\)](#)

**Examples**

```
library(CoopGame)
paramCheckResult=getEmptyParamCheckResult()
validAllocation=c(1,2,3)
stopOnInvalidAllocation(paramCheckResult,x=validAllocation,n=3)
```

---

stopOnInvalidBoolean    *Parameter Function stopOnInvalidBoolean*

---

**Description**

stopOnInvalidBoolean checks definition is the parameter a boolean

**Usage**

```
stopOnInvalidBoolean(paramCheckResult, boolean)
```

**Arguments**

paramCheckResult    list object for check result with list element 'errCode' for the error code and 'errMessage' for the error message.

boolean    parameter which is checked if it is a valid boolean.

**Error Code Ranges**

Error codes and messages shown to user if error on parameter check occurs

Error Code	Message
1120	Parameter is not a boolean value
1121	Parameter is not of length 1

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>  
 Franz Mueller

**See Also**

Other ParameterChecks\_CoopGame: [getEmptyParamCheckResult\(\)](#), [stopOnInconsistentEstateAndClaimsVector\(\)](#), [stopOnInvalidAllocation\(\)](#), [stopOnInvalidClaimsVector\(\)](#), [stopOnInvalidCoalitionS\(\)](#), [stopOnInvalidDictator\(\)](#), [stopOnInvalidEstate\(\)](#), [stopOnInvalidGameVector\(\)](#), [stopOnInvalidGrandCoalition\(\)](#), [stopOnInvalidIndex\(\)](#), [stopOnInvalidLeftRightGloveGame\(\)](#), [stopOnInvalidNChooseB\(\)](#), [stopOnInvalidNumberOfPlayers\(\)](#), [stopOnInvalidNumber\(\)](#), [stopOnInvalidQuota\(\)](#), [stopOnInvalidVetoPlayer\(\)](#), [stopOnInvalidWeightVector\(\)](#), [stopOnParamCheckError\(\)](#)

**Examples**

```
library(CoopGame)
paramCheckResult=getEmptyParamCheckResult()
validBoolean = TRUE
stopOnInvalidBoolean(paramCheckResult, validBoolean)
```

---

stopOnInvalidClaimsVector

*Parameter Function stopOnInvalidClaimsVector*

---

**Description**

stopOnInvalidClaimsVector checks if claims vector in a bankruptcy game is specified correctly. Validation result gets stored to object paramCheckResult in case an error occurred and causes stop otherwise.

**Usage**

```
stopOnInvalidClaimsVector(paramCheckResult, n, d)
```

**Arguments**

paramCheckResult  
 list object for check result with list element 'errCode' for the error code and 'errMessage' for the error message.

n  
 represents the number of players

d  
 numeric vector which contains the claims of each player in a bankruptcy game

**Error Code Ranges**

Error codes and messages shown to user if error on parameter check occurs

<b>Error Code</b>	<b>Message</b>
1160	Number of claims must equal the number of players in the bankruptcy game!
1161	Invalid claims vector as d must be numeric

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**See Also**

Other ParameterChecks\_CoopGame: [getEmptyParamCheckResult\(\)](#), [stopOnInconsistentEstateAndClaimsVector\(\)](#), [stopOnInvalidAllocation\(\)](#), [stopOnInvalidBoolean\(\)](#), [stopOnInvalidCoalitionS\(\)](#), [stopOnInvalidDictator\(\)](#), [stopOnInvalidEstate\(\)](#), [stopOnInvalidGameVector\(\)](#), [stopOnInvalidGrandCoalitionN\(\)](#), [stopOnInvalidIndex\(\)](#), [stopOnInvalidLeftRightGloveGame\(\)](#), [stopOnInvalidNChooseB\(\)](#), [stopOnInvalidNumberOfPlayers\(\)](#), [stopOnInvalidNumber\(\)](#), [stopOnInvalidQuota\(\)](#), [stopOnInvalidVetoPlayer\(\)](#), [stopOnInvalidWeightVector\(\)](#), [stopOnParamCheckError\(\)](#)

**Examples**

```
library(CoopGame)
paramCheckResult=getEmptyParamCheckResult()
validClaimsVector = c(100,150,200)
stopOnInvalidClaimsVector(paramCheckResult, n=3, d=validClaimsVector)
```

---

stopOnInvalidCoalitionS

*Parameter Function stopOnInvalidCoalitionS*

---

**Description**

stopOnInvalidCoalitionS checks if coalition S as subset of grand coalition N is specified correctly and causes calculation to stop otherwise.

**Usage**

```
stopOnInvalidCoalitionS(paramCheckResult, S, N = NULL, n = NULL, v = NULL)
```

**Arguments**

paramCheckResult	list object for check result with list element 'errCode' for the error code and 'errMessage' for the error message.
S	numeric vector with coalition of players
N	represents the grand coalition.
n	represents the number of players
v	Numeric vector of length $2^n - 1$ representing the values of the coalitions of a TU game with n players

**Error Code Ranges**

Error codes and messages shown to user if error on parameter check occurs

<b>Error Code</b>	<b>Message</b>
1020	Coalition vector S is invalid as 'NULL'
1021	Coalition vector S is invalid as not numeric
1022	Coalition vector S no subset of grand coalition N
1023	The number of players in S cannot be greater than the number of players in N
1024	Specified coalition is inconsistent with game vector

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

Johannes Anwander <anwander.johannes@gmail.com>

**See Also**

Other ParameterChecks\_CoopGame: [getEmptyParamCheckResult\(\)](#), [stopOnInconsistentEstateAndClaimsVector\(\)](#), [stopOnInvalidAllocation\(\)](#), [stopOnInvalidBoolean\(\)](#), [stopOnInvalidClaimsVector\(\)](#), [stopOnInvalidDictator\(\)](#), [stopOnInvalidEstate\(\)](#), [stopOnInvalidGameVector\(\)](#), [stopOnInvalidGrandCoalitionN\(\)](#), [stopOnInvalidIndex\(\)](#), [stopOnInvalidLeftRightGloveGame\(\)](#), [stopOnInvalidNChooseB\(\)](#), [stopOnInvalidNumberOfPlayers\(\)](#), [stopOnInvalidNumber\(\)](#), [stopOnInvalidQuota\(\)](#), [stopOnInvalidVetoPlayer\(\)](#), [stopOnInvalidWeightVector\(\)](#), [stopOnParamCheckError\(\)](#)

**Examples**

```
library(CoopGame)
paramCheckResult=getEmptyParamCheckResult()
validCoalition = c(1,2,3)
stopOnInvalidCoalitionS(paramCheckResult, S=validCoalition, N=c(1,2,3,4,5))
```

---

*stopOnInvalidDictator* *Parameter Function stopOnInvalidDictator*

---

**Description**

stopOnInvalidDictator checks if dictator is specified correctly in a dictator game. Validation result gets stored to object paramCheckResult in case an error occurred and causes calculation to stop.

**Usage**

```
stopOnInvalidDictator(paramCheckResult, dictator, n = NULL)
```

**Arguments**

paramCheckResult	list object for check result with list element 'errCode' for the error code and 'errMessage' for the error message.
dictator	Number of the dictator
n	represents the number of players

**Error Code Ranges**

Error codes and messages shown to user if error on parameter check occurs

Error Code	Message
1090	'dictator' does not contain only one single element
1091	Representation of 'dictator' is not 'numeric'
1092	'dictator' is not element of grand coalition
1093	'dictator' is 'NULL'

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

Johannes Anwander <anwander.johannes@gmail.com>

**See Also**

Other ParameterChecks\_CoopGame: [getEmptyParamCheckResult\(\)](#), [stopOnInconsistentEstateAndClaimsVector\(\)](#), [stopOnInvalidAllocation\(\)](#), [stopOnInvalidBoolean\(\)](#), [stopOnInvalidClaimsVector\(\)](#), [stopOnInvalidCoalitions\(\)](#), [stopOnInvalidEstate\(\)](#), [stopOnInvalidGameVector\(\)](#), [stopOnInvalidGrandCoalitionN\(\)](#), [stopOnInvalidIndex\(\)](#), [stopOnInvalidLeftRightGloveGame\(\)](#), [stopOnInvalidNChooseB\(\)](#), [stopOnInvalidNumberOfPlayers\(\)](#), [stopOnInvalidNumber\(\)](#), [stopOnInvalidQuota\(\)](#), [stopOnInvalidVetoPlayer\(\)](#), [stopOnInvalidWeightVector\(\)](#), [stopOnParamCheckError\(\)](#)

**Examples**

```
library(CoopGame)
paramCheckResult=getEmptyParamCheckResult()
validDictator = 3
stopOnInvalidDictator(paramCheckResult,dictator=validDictator,n=3)
```

---

stopOnInvalidEstate     *Parameter Function stopOnInvalidEstate*

---

**Description**

stopOnInvalidBankruptcy checks if estate is specified correctly (as parameter in a bankruptcy game). Validation result gets stored to object paramCheckResult in case an error occurred and causes stop otherwise.

**Usage**

```
stopOnInvalidEstate(paramCheckResult, E)
```

**Arguments**

paramCheckResult      list object for check result with list element 'errCode' for the error code and 'errMessage' for the error message.

E                        is the value of the estate in a bankruptcy game

**Error Code Ranges**

Error codes and messages shown to user if error on parameter check occurs

<b>Error Code</b>	<b>Message</b>
1150	Estate must be nonnegative!
1151	Estate must be numeric!
1152	Invalid estate as E is NULL

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**See Also**

Other ParameterChecks\_CoopGame: [getEmptyParamCheckResult\(\)](#), [stopOnInconsistentEstateAndClaimsVector\(\)](#), [stopOnInvalidAllocation\(\)](#), [stopOnInvalidBoolean\(\)](#), [stopOnInvalidClaimsVector\(\)](#), [stopOnInvalidCoalitions\(\)](#), [stopOnInvalidDictator\(\)](#), [stopOnInvalidGameVector\(\)](#), [stopOnInvalidGrandCoalitionN\(\)](#), [stopOnInvalidIndex\(\)](#), [stopOnInvalidLeftRightGloveGame\(\)](#), [stopOnInvalidNChooseB\(\)](#), [stopOnInvalidNumberOfPlayers\(\)](#), [stopOnInvalidNumber\(\)](#), [stopOnInvalidQuota\(\)](#), [stopOnInvalidVetoPlayer\(\)](#), [stopOnInvalidWeightVector\(\)](#), [stopOnParamCheckError\(\)](#)

**Examples**

```
library(CoopGame)
paramCheckResult=getEmptyParamCheckResult()
validEstate = 55
stopOnInvalidEstate(paramCheckResult, E=validEstate)
```

---

stopOnInvalidGameVector

*Parameter Function stopOnInvalidGameVector*

---

**Description**

stopOnInvalidGameVector checks if game vector v is specified correctly. Validation result gets stored to object paramCheckResult in case an error occurred and causes calculation to stop.



**Usage**

```
stopOnInvalidGameVector(paramCheckResult, v, n = NULL)
```

**Arguments**

`paramCheckResult` list object for check result with list element 'errCode' for the error code and 'errMessage' for the error message.

`v` Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with `n` players

`n` represents the number of players

**Error Code Ranges**

Error codes and messages shown to user if error on parameter check occurs

Error Code	Message
1000	Game vector is invalid as 'NULL'
1001	Number of elements in game vector is invalid
1002	Type of game vector is not numeric
1003	Game vector has different number of players than <code>n</code>
1004	Null game specified, value for every player is 0

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

Johannes Anwander <anwander.johannes@gmail.com>

**See Also**

Other ParameterChecks\_CoopGame: [getEmptyParamCheckResult\(\)](#), [stopOnInconsistentEstateAndClaimsVector\(\)](#), [stopOnInvalidAllocation\(\)](#), [stopOnInvalidBoolean\(\)](#), [stopOnInvalidClaimsVector\(\)](#), [stopOnInvalidCoalitions\(\)](#), [stopOnInvalidDictator\(\)](#), [stopOnInvalidEstate\(\)](#), [stopOnInvalidGrandCoalitionN\(\)](#), [stopOnInvalidIndex\(\)](#), [stopOnInvalidLeftRightGloveGame\(\)](#), [stopOnInvalidNChooseB\(\)](#), [stopOnInvalidNumberOfPlayers\(\)](#), [stopOnInvalidNumber\(\)](#), [stopOnInvalidQuota\(\)](#), [stopOnInvalidVetoPlayer\(\)](#), [stopOnInvalidWeightVector\(\)](#), [stopOnParamCheckError\(\)](#)

**Examples**

```
library(CoopGame)
validGameVector=c(0,0,0,60,60,60,72)
stopOnInvalidGameVector(paramCheckResult,validGameVector)
```

---

stopOnInvalidGrandCoalitionN

*Parameter Function stopOnInvalidGrandCoalitionN*

---

### Description

stopOnInvalidGrandCoalitionN checks if grand coalition N is specified correctly and causes calculation to stop otherwise.

### Usage

stopOnInvalidGrandCoalitionN(paramCheckResult, N)

### Arguments

paramCheckResult

list object for check result with list element 'errCode' for the error code and 'errMessage' for the error message.

N

represents the grand coalition.

### Error Code Ranges

Error codes and messages shown to user if error on parameter check occurs

Error Code	Message
1010	Grand coalition vector N is invalid as 'NULL'
1011	Grand coalition vector N is invalid as not numeric

### Author(s)

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

Johannes Anwander <anwander.johannes@gmail.com>

### See Also

Other ParameterChecks\_CoopGame: [getEmptyParamCheckResult\(\)](#), [stopOnInconsistentEstateAndClaimsVector\(\)](#), [stopOnInvalidAllocation\(\)](#), [stopOnInvalidBoolean\(\)](#), [stopOnInvalidClaimsVector\(\)](#), [stopOnInvalidCoalitions\(\)](#), [stopOnInvalidDictator\(\)](#), [stopOnInvalidEstate\(\)](#), [stopOnInvalidGameVector\(\)](#), [stopOnInvalidIndex\(\)](#), [stopOnInvalidLeftRightGloveGame\(\)](#), [stopOnInvalidNChooseB\(\)](#), [stopOnInvalidNumberOfPlayers\(\)](#), [stopOnInvalidNumber\(\)](#), [stopOnInvalidQuota\(\)](#), [stopOnInvalidVetoPlayer\(\)](#), [stopOnInvalidWeightVector\(\)](#), [stopOnParamCheckError\(\)](#)

### Examples

```
library(CoopGame)
paramCheckResult=getEmptyParamCheckResult()
validGrandCoalition = c(1,2,3,4,5)
```

```
stopOnInvalidGrandCoalitionN(paramCheckResult, N=validGrandCoalition)
```

---

```
stopOnInvalidIndex      Parameter Function stopOnInvalidIndex
```

---

### Description

stopOnInvalidIndex checks if coalition function (in the form of either v or A) is specified correctly and causes calculation to stop otherwise.

### Usage

```
stopOnInvalidIndex(paramCheckResult, index, n = NULL)
```

### Arguments

paramCheckResult	list object for check result with list element 'errCode' for the error code and 'errMessage' for the error message.
index	index which is checked to be a valid index
n	represents the number of players

### Error Code Ranges

Error codes and messages shown to user if error on parameter check occurs

Error Code	Message
1070	Index is 'NULL'.
1071	Index is 'not numeric'.
1072	Index is within the wrong range according to number of players n.

### Author(s)

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

Johannes Anwander <anwander.johannes@gmail.com>

### See Also

Other ParameterChecks\_CoopGame: [getEmptyParamCheckResult\(\)](#), [stopOnInconsistentEstateAndClaimsVector\(\)](#), [stopOnInvalidAllocation\(\)](#), [stopOnInvalidBoolean\(\)](#), [stopOnInvalidClaimsVector\(\)](#), [stopOnInvalidCoalitions](#), [stopOnInvalidDictator\(\)](#), [stopOnInvalidEstate\(\)](#), [stopOnInvalidGameVector\(\)](#), [stopOnInvalidGrandCoalition](#), [stopOnInvalidLeftRightGloveGame\(\)](#), [stopOnInvalidNChooseB\(\)](#), [stopOnInvalidNumberOfPlayers\(\)](#), [stopOnInvalidNumber\(\)](#), [stopOnInvalidQuota\(\)](#), [stopOnInvalidVetoPlayer\(\)](#), [stopOnInvalidWeightVector\(\)](#), [stopOnParamCheckError\(\)](#)

**Examples**

```
library(CoopGame)
v=c(1:7)
paramCheckResult=getEmptyParamCheckResult()
validIndex = 5
stopOnInvalidIndex(paramCheckResult, index=validIndex, n=3)
```

---

stopOnInvalidLeftRightGloveGame

*Parameter Function stopOnInvalidLeftRightGloveGame*

---

**Description**

stopOnInvalidLeftRightGloveGame checks if L (left gloves) and R (right gloves) are specified as parameter correctly (also regarding grand coalition). Validation result gets stored to object paramCheckResult in case an error occurred and causes calculation to stop.

**Usage**

```
stopOnInvalidLeftRightGloveGame(paramCheckResult, L, R, N)
```

**Arguments**

paramCheckResult	list object for check result with list element 'errCode' for the error code and 'errMessage' for the error message.
L	numeric vector of players owning one left-hand glove each
R	numeric vector of players owning one right-hand glove each
N	represents the grand coalition.

**Error Code Ranges**

Error codes and messages shown to user if error on parameter check occurs

<b>Error Code</b>	<b>Message</b>
1140	Not all players in L and R included.
1141	L must have size > 0.
1142	R must have size > 0.
1143	L and R have to be disjoint sets.

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

Johannes Anwander <anwander.johannes@gmail.com>

**See Also**

Other ParameterChecks\_CoopGame: [getEmptyParamCheckResult\(\)](#), [stopOnInconsistentEstateAndClaimsVector\(\)](#), [stopOnInvalidAllocation\(\)](#), [stopOnInvalidBoolean\(\)](#), [stopOnInvalidClaimsVector\(\)](#), [stopOnInvalidCoalitions\(\)](#), [stopOnInvalidDictator\(\)](#), [stopOnInvalidEstate\(\)](#), [stopOnInvalidGameVector\(\)](#), [stopOnInvalidGrandCoalitions\(\)](#), [stopOnInvalidIndex\(\)](#), [stopOnInvalidNChooseB\(\)](#), [stopOnInvalidNumberOfPlayers\(\)](#), [stopOnInvalidNumber\(\)](#), [stopOnInvalidQuota\(\)](#), [stopOnInvalidVetoPlayer\(\)](#), [stopOnInvalidWeightVector\(\)](#), [stopOnParamCheckError\(\)](#)

**Examples**

```
library(CoopGame)
paramCheckResult=getEmptyParamCheckResult()
validL=c(1,3)
validR=c(2)
stopOnInvalidLeftRightGloveGame(paramCheckResult, L=validL,R=validR,N=c(1,2,3))
```

---

stopOnInvalidNChooseB *Parameter Function stopOnInvalidNChooseB*

---

**Description**

stopOnInvalidNChooseB checks if definition of n choose b is specified correctly and causes stop otherwise.

**Usage**

```
stopOnInvalidNChooseB(paramCheckResult, n, b)
```

**Arguments**

paramCheckResult  
list object for check result with list element 'errCode' for the error code and 'errMessage' for the error message.

n  
represents the number of players

b  
number of players in subset

**Error Code Ranges**

Error codes and messages shown to user if error on parameter check occurs

Error Code	Message
1080	Number of players 'n' is 'NULL'
1081	Number of involved players 'b' is 'NULL'
1082	Number of players 'n' is not 'numeric'
1083	Number of involved players 'b' is not 'numeric'
1084	Number of involved players 'b' is greater than of players 'n'

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>  
 Johannes Anwander <anwander.johannes@gmail.com>

**See Also**

Other ParameterChecks\_CoopGame: [getEmptyParamCheckResult\(\)](#), [stopOnInconsistentEstateAndClaimsVector\(\)](#), [stopOnInvalidAllocation\(\)](#), [stopOnInvalidBoolean\(\)](#), [stopOnInvalidClaimsVector\(\)](#), [stopOnInvalidCoalitions\(\)](#), [stopOnInvalidDictator\(\)](#), [stopOnInvalidEstate\(\)](#), [stopOnInvalidGameVector\(\)](#), [stopOnInvalidGrandCoalitions\(\)](#), [stopOnInvalidIndex\(\)](#), [stopOnInvalidLeftRightGloveGame\(\)](#), [stopOnInvalidNumberOfPlayers\(\)](#), [stopOnInvalidNumber\(\)](#), [stopOnInvalidQuota\(\)](#), [stopOnInvalidVetoPlayer\(\)](#), [stopOnInvalidWeightVector\(\)](#), [stopOnParamCheckError\(\)](#)

**Examples**

```
library(CoopGame)
paramCheckResult=getEmptyParamCheckResult()
validN = 3
validAndConsistentB = 2
stopOnInvalidNChooseB(paramCheckResult, n=validN, b=validAndConsistentB)
```

---

stopOnInvalidNumber     *Parameter Function stopOnInvalidNumber*

---

**Description**

stopOnInvalidNumber checks definition is the parameter a number

**Usage**

```
stopOnInvalidNumber(paramCheckResult, number)
```

**Arguments**

paramCheckResult     list object for check result with list element 'errCode' for the error code and 'errMessage' for the error message.

number     input which is checked to be valid number

**Error Code Ranges**

Error codes and messages shown to user if error on parameter check occurs

<b>Error Code</b>	<b>Message</b>
1130	Parameter is not a number
1131	Parameter is not of length 1

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>  
 Franz Mueller

**See Also**

Other ParameterChecks\_CoopGame: [getEmptyParamCheckResult\(\)](#), [stopOnInconsistentEstateAndClaimsVector\(\)](#), [stopOnInvalidAllocation\(\)](#), [stopOnInvalidBoolean\(\)](#), [stopOnInvalidClaimsVector\(\)](#), [stopOnInvalidCoalitions\(\)](#), [stopOnInvalidDictator\(\)](#), [stopOnInvalidEstate\(\)](#), [stopOnInvalidGameVector\(\)](#), [stopOnInvalidGrandCoalitions\(\)](#), [stopOnInvalidIndex\(\)](#), [stopOnInvalidLeftRightGloveGame\(\)](#), [stopOnInvalidNChooseB\(\)](#), [stopOnInvalidNumberOfPlayers\(\)](#), [stopOnInvalidQuota\(\)](#), [stopOnInvalidVetoPlayer\(\)](#), [stopOnInvalidWeightVector\(\)](#), [stopOnParamCheckError\(\)](#)

**Examples**

```
library(CoopGame)
paramCheckResult=getEmptyParamCheckResult()
validNumber = 5
stopOnInvalidNumber(paramCheckResult, validNumber)
```

---

stopOnInvalidNumberOfPlayers

*Parameter Function stopOnInvalidNumberOfPlayers*

---

**Description**

stopOnInvalidNumberOfPlayers checks if number of players is specified correctly and causes calculation to stop otherwise.

**Usage**

```
stopOnInvalidNumberOfPlayers(paramCheckResult, n)
```

**Arguments**

paramCheckResult  
 list object for check result with list element 'errCode' for the error code and 'errMessage' for the error message.

n  
 represents the number of players

**Error Code Ranges**

Error codes and messages shown to user if error on parameter check occurs

Error Code	Message
1050	Number of players is invalid as below 2

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>  
 Johannes Anwander <anwander.johannes@gmail.com>

**See Also**

Other ParameterChecks\_CoopGame: [getEmptyParamCheckResult\(\)](#), [stopOnInconsistentEstateAndClaimsVector\(\)](#), [stopOnInvalidAllocation\(\)](#), [stopOnInvalidBoolean\(\)](#), [stopOnInvalidClaimsVector\(\)](#), [stopOnInvalidCoalitions\(\)](#), [stopOnInvalidDictator\(\)](#), [stopOnInvalidEstate\(\)](#), [stopOnInvalidGameVector\(\)](#), [stopOnInvalidGrandCoalitions\(\)](#), [stopOnInvalidIndex\(\)](#), [stopOnInvalidLeftRightGloveGame\(\)](#), [stopOnInvalidNChooseB\(\)](#), [stopOnInvalidNumber\(\)](#), [stopOnInvalidQuota\(\)](#), [stopOnInvalidVetoPlayer\(\)](#), [stopOnInvalidWeightVector\(\)](#), [stopOnParamCheckError\(\)](#)

**Examples**

```
library(CoopGame)
paramCheckResult=getEmptyParamCheckResult()
validNumberOfPlayers = 10
stopOnInvalidNumberOfPlayers(paramCheckResult, n=validNumberOfPlayers)
```

---

stopOnInvalidQuota      *Parameter Function stopOnInvalidQuota*

---

**Description**

stopOnInvalidQuota checks if quotoa in a weighted voting game is specified correctly. Validation result gets stored to object paramCheckResult in case an error occurred and causes calculation to stop.

**Usage**

```
stopOnInvalidQuota(paramCheckResult, q)
```

**Arguments**

paramCheckResult      list object for check result with list element 'errCode' for the error code and 'errMessage' for the error message.  
 q                      is the quota

**Error Code Ranges**

Error codes and messages shown to user if error on parameter check occurs

Error Code	Message
1030	Invalid quota as q is NULL
1031	Quota must be greater than zero!
1032	Quota must be numeric!



**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>  
 Johannes Anwander <anwander.johannes@gmail.com>

**See Also**

Other ParameterChecks\_CoopGame: [getEmptyParamCheckResult\(\)](#), [stopOnInconsistentEstateAndClaimsVector\(\)](#), [stopOnInvalidAllocation\(\)](#), [stopOnInvalidBoolean\(\)](#), [stopOnInvalidClaimsVector\(\)](#), [stopOnInvalidCoalitions\(\)](#), [stopOnInvalidDictator\(\)](#), [stopOnInvalidEstate\(\)](#), [stopOnInvalidGameVector\(\)](#), [stopOnInvalidGrandCoalitions\(\)](#), [stopOnInvalidIndex\(\)](#), [stopOnInvalidLeftRightGloveGame\(\)](#), [stopOnInvalidNChooseB\(\)](#), [stopOnInvalidNumberOfPlayers\(\)](#), [stopOnInvalidNumber\(\)](#), [stopOnInvalidVetoPlayer\(\)](#), [stopOnInvalidWeightVector\(\)](#), [stopOnParamCheckError\(\)](#)

**Examples**

```
library(CoopGame)
paramCheckResult=getEmptyParamCheckResult()
validQuota = 3
stopOnInvalidQuota(paramCheckResult, q=validQuota)
```

---

stopOnInvalidVetoPlayer

*Parameter Function stopOnInvalidVetoPlayer*

---

**Description**

stopOnInvalidVetoPlayer checks if vetoPlayer is specified correctly. Validation result gets stored to object paramCheckResult in case an error occurred and causes calculation to stop.

**Usage**

```
stopOnInvalidVetoPlayer(paramCheckResult, vetoPlayer)
```

**Arguments**

paramCheckResult  
 list object for check result with list element 'errCode' for the error code and 'errMessage' for the error message.

vetoPlayer  
 represents the veto player

**Error Code Ranges**

Error codes and messages shown to user if error on parameter check occurs

Error Code	Message
1190	At least one veto player has to be specified
1191	Only a single veto player is allowed for this game

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>  
 Johannes Anwander <anwander.johannes@gmail.com>

**See Also**

Other ParameterChecks\_CoopGame: [getEmptyParamCheckResult\(\)](#), [stopOnInconsistentEstateAndClaimsVector\(\)](#), [stopOnInvalidAllocation\(\)](#), [stopOnInvalidBoolean\(\)](#), [stopOnInvalidClaimsVector\(\)](#), [stopOnInvalidCoalitions\(\)](#), [stopOnInvalidDictator\(\)](#), [stopOnInvalidEstate\(\)](#), [stopOnInvalidGameVector\(\)](#), [stopOnInvalidGrandCoalitions\(\)](#), [stopOnInvalidIndex\(\)](#), [stopOnInvalidLeftRightGloveGame\(\)](#), [stopOnInvalidNChooseB\(\)](#), [stopOnInvalidNumberOfPlayers\(\)](#), [stopOnInvalidNumber\(\)](#), [stopOnInvalidQuota\(\)](#), [stopOnInvalidWeightVector\(\)](#), [stopOnParamCheckError\(\)](#)

**Examples**

```
library(CoopGame)
paramCheckResult=getEmptyParamCheckResult()
validVetoPlayer = 3
stopOnInvalidVetoPlayer(paramCheckResult, vetoPlayer=validVetoPlayer)
```

---

stopOnInvalidWeightVector

*Parameter Function stopOnInvalidWeightVector*

---

**Description**

stopOnInvalidWeightVector checks if weight vector in a weighted voting game is specified correctly. Validation result gets stored to object paramCheckResult in case an error occurred and causes stop otherwise.

**Usage**

```
stopOnInvalidWeightVector(paramCheckResult, n, w)
```

**Arguments**

paramCheckResult	list object for check result with list element 'errCode' for the error code and 'errMessage' for the error message.
n	represents the number of players
w	numeric vector which contains the weight of each player

**Error Code Ranges**

Error codes and messages shown to user if error on parameter check occurs

<b>Error Code</b>	<b>Message</b>
1110	Number of weights must be equal or greater than number of players in coalition!
1111	Invalid weight vector as w is not numeric

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

Johannes Anwander <anwander.johannes@gmail.com>

**See Also**

Other ParameterChecks\_CoopGame: [getEmptyParamCheckResult\(\)](#), [stopOnInconsistentEstateAndClaimsVector\(\)](#), [stopOnInvalidAllocation\(\)](#), [stopOnInvalidBoolean\(\)](#), [stopOnInvalidClaimsVector\(\)](#), [stopOnInvalidCoalition\(\)](#), [stopOnInvalidDictator\(\)](#), [stopOnInvalidEstate\(\)](#), [stopOnInvalidGameVector\(\)](#), [stopOnInvalidGrandCoalition\(\)](#), [stopOnInvalidIndex\(\)](#), [stopOnInvalidLeftRightGloveGame\(\)](#), [stopOnInvalidNChooseB\(\)](#), [stopOnInvalidNumberOfPlayers\(\)](#), [stopOnInvalidNumber\(\)](#), [stopOnInvalidQuota\(\)](#), [stopOnInvalidVetoPlayer\(\)](#), [stopOnParamCheckError\(\)](#)

**Examples**

```
library(CoopGame)
paramCheckResult=getEmptyParamCheckResult()
validWeightVector = c(1,2,3)
stopOnInvalidWeightVector(paramCheckResult, n=3, w=validWeightVector)
```

---

`stopOnParamCheckError` *stopOnParamCheckError - stop and create error message on error*

---

**Description**

`stopOnParamCheckError` causes and creates error message on base of `paramCheckResult` parameter where `'errCode' <> '0'` in case error occurred.

**Usage**

```
stopOnParamCheckError(paramCheckResult)
```

**Arguments**

`paramCheckResult`

list object for check result with list element `'errCode'` for the error code and `'errMessage'` for the error message.

**Author(s)**

Johannes Anwander <anwander.johannes@gmail.com>

**See Also**

Other ParameterChecks\_CoopGame: [getEmptyParamCheckResult\(\)](#), [stopOnInconsistentEstateAndClaimsVector\(\)](#), [stopOnInvalidAllocation\(\)](#), [stopOnInvalidBoolean\(\)](#), [stopOnInvalidClaimsVector\(\)](#), [stopOnInvalidCoalitions\(\)](#), [stopOnInvalidDictator\(\)](#), [stopOnInvalidEstate\(\)](#), [stopOnInvalidGameVector\(\)](#), [stopOnInvalidGrandCoalitions\(\)](#), [stopOnInvalidIndex\(\)](#), [stopOnInvalidLeftRightGloveGame\(\)](#), [stopOnInvalidNChooseB\(\)](#), [stopOnInvalidNumberOfPlayers\(\)](#), [stopOnInvalidNumber\(\)](#), [stopOnInvalidQuota\(\)](#), [stopOnInvalidVetoPlayer\(\)](#), [stopOnInvalidWeightVector\(\)](#)

**Examples**

```
library(CoopGame)
paramCheckResult=getEmptyParamCheckResult()
stopOnParamCheckError(paramCheckResult)
```

---

tauValue

*Compute tau-value*


---

**Description**

Calculates the tau-value for a quasi-balanced TU game with n players.

**Usage**

```
tauValue(v)
```

**Arguments**

v                    Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with n players

**Value**

tau-value for a quasi-balanced TU game with n players

**Author(s)**

Johannes Anwander <[anwander.johannes@gmail.com](mailto:anwander.johannes@gmail.com)>

Jochen Staudacher <[jochen.staudacher@hs-kempten.de](mailto:jochen.staudacher@hs-kempten.de)>

**References**

- Branzei R., Dimitrov D. and Tijs S. (2006) *Models in cooperative game theory*, Springer, p. 32
- Tijs S. (1981) "Bounds for the core of a game and the t-value", In: Moeschlin, O. and Pallaschke, D. (Eds.): *Game Theory and Mathematical Economics*, North-Holland, pp. 123–132
- Stach I. (2011) "Tijs value", *Encyclopedia of Power*, SAGE Publications, pp. 667–670

**Examples**

```
library(CoopGame)
tauValue(v=c(0,0,0,0,1,0,1))

#Example from article by Stach (2011)
library(CoopGame)
v=c(0,0,0,1,2,1,3)
tauValue(v)
#[1] 1.2 0.6 1.2
```

---

unanimityGame	<i>Construct a unanimity game</i>
---------------	-----------------------------------

---

**Description**

**Create a list containing all information about a specified unanimity game:**

The player in coalition T are the productive players. If all players from T are included, the coalition generates value 1, otherwise 0.

Note that unanimity games are always simple games.

**Usage**

```
unanimityGame(n, T)
```

**Arguments**

n	represents the number of players
T	represents coalition which is subset of grand coalition and necessary for generating value

**Value**

A list with three elements representing the unanimity game (n, T, Game vector v)

**Related Functions**

[unanimityGameValue](#), [unanimityGameVector](#)

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**References**

Peleg B. and Sudhoelter P. (2007) *Theory of cooperative games*, 2nd Edition, Springer, p. 152  
 Maschler M., Solan E. and Zamir S. (2013) *Game Theory*, Cambridge University Press, p. 764

**Examples**

```

library(CoopGame)
unanimityGame(n=3,T=c(1,2))

library(CoopGame)
unanimityGame(n=4,T=c(1,2))
#Output
#n
#[1] 4
#
#$T
#[1] 1 2

#$v
#[1] 0 0 0 0 1 0 0 0 0 0 1 1 0 0 1

```

---

unanimityGameValue      *Compute value of a coalition for a unanimity game*

---

**Description**

**Coalition value for a specified unanimity game:**

For further information see [unanimityGame](#)

**Usage**

```
unanimityGameValue(S, T)
```

**Arguments**

S                    numeric vector with coalition of players

T                    represents coalition which is subset of grand coalition N and necessary for generating value

**Value**

1 if all players of coalition T are included in S, else 0

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

Johannes Anwander <anwander.johannes@gmail.com>

## References

- Peleg B. and Sudhoelter P. (2007) *Theory of cooperative games*, 2nd Edition, Springer, p. 152  
Maschler M., Solan E. and Zamir S. (2013) *Game Theory*, Cambridge University Press, p. 764

## Examples

```
library(CoopGame)
unanimityGameValue(S=c(1,2,3),T=c(2))
```

---

unanimityGameVector     *Compute game vector for a unanimity game*

---

## Description

### Game Vector for a specified unanimity game:

For further information see [unanimityGame](#)

## Usage

```
unanimityGameVector(n, T)
```

## Arguments

n	represents the number of players
T	represents coalition which is subset of grand coalition N and necessary for generating value

## Value

Game Vector where each element contains 1 if all players of coalition 'T' are included in 'S' else 0

## Author(s)

Johannes Anwander <anwander.johannes@gmail.com>  
Jochen Staudacher <jochen.staudacher@hs-kempten.de>

## References

- Peleg B. and Sudhoelter P. (2007) *Theory of cooperative games*, 2nd Edition, Springer, p. 152  
Maschler M., Solan E. and Zamir S. (2013) *Game Theory*, Cambridge University Press, p. 764

## Examples

```
library(CoopGame)
unanimityGameVector(n=3,T=c(2))
```

---

webersetVertices      *Compute vertices of Weber Set*

---

**Description**

Calculates the Weber Set for given game vector with n players.

**Usage**

```
webersetVertices(v)
```

**Arguments**

v                      Numeric vector of length  $2^n - 1$  representing the values of the coalitions of a TU game with n players

**Value**

rows of the matrix are the vertices of the Weber Set

**Author(s)**

Anna Merkle  
 Franz Mueller  
 Jochen Staudacher <jochen.staudacher@hs-kempten.de>

**References**

Weber R.J. (1988) "Probabilistic values for games". In: Roth A.E. (Ed.), *The Shapley Value. Essays in honor of Lloyd S. Shapley*, Cambridge University Press, pp. 101–119  
 Peters H. (2015) *Game Theory: A Multi-Leveled Approach*, 2nd Edition, Springer, pp. 327–329

**Examples**

```
library(CoopGame)
webersetVertices(c(0,0,0,1,1,1,2))

#Example of a 3-player TU game (with a Weber Set with 6 vertices)
library(CoopGame)
v = c(0,1,2,3,4,5,6)
webersetVertices(v)

#Example of a 4-player TU game (with a Weber Set with 14 vertices)
library(CoopGame)
v = c(5,2,4,7,15,15,15,15,15,15,20,20,20,20,35)
webersetVertices(v)
```



---

weightedVotingGame     *Construct a weighted voting game*

---

### Description

**Create a list containing all information about a specified weighted voting game:**

For a weighted voting game we receive a game vector where each element contains 1 if the sum of the weights of coalition  $S$  is greater or equal than quota  $q$ , else  $0$ .

Note that weighted voting games are always simple games.

### Usage

```
weightedVotingGame(n, w, q)
```

### Arguments

n	represents the number of players
w	numeric vector which contains the weight of each player
q	is the quota

### Value

A list with four elements representing the weighted voting game (n, w, q, Game vector v)

### Related Functions

[weightedVotingGameValue](#), [weightedVotingGameVector](#)

### Author(s)

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

### References

Peleg B. (2002) "Game-theoretic analysis of voting in committees". in: Handbook of Social Choice and Welfare 1, pp. 195–201

Peleg B. and Sudhoelter P. (2007) *Theory of cooperative games*, 2nd Edition, Springer, p. 17

Maschler M., Solan E. and Zamir S. (2013) *Game Theory*, Cambridge University Press, pp. 825–831

**Examples**

```

library(CoopGame)
weightedVotingGame(n=3,w=c(1,2,3),q=4)

library(CoopGame)
weightedVotingGame(n=4,w=c(1,2,3,4),q=5)

#Output:
#$n
#[1] 4

#$w
#[1] 1 2 3 4
#
#$q
#[1] 5
#
#$v
#[1] 0 0 0 0 0 0 1 1 1 1 1 1 1 1

```

---

```
weightedVotingGameValue
```

*Compute value of a coalition for a weighted voting game*

---

**Description**

**Coalition value for a specified weighted voting game:**

For further information see [weightedVotingGame](#)

**Usage**

```
weightedVotingGameValue(S, w, q)
```

**Arguments**

S	numeric vector with coalition of players
w	numeric vector which contains the weight of each player
q	is the quota

**Value**

1 if the sum of the weights of coalition S is greater or equal than quota q else 0

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

Johannes Anwander <anwander.johannes@gmail.com>

Michael Maerz

**References**

Peleg B. (2002) "Game-theoretic analysis of voting in committees". in: Handbook of Social Choice and Welfare 1, pp. 195–201

Peleg B. and Sudhoelter P. (2007) *Theory of cooperative games*, 2nd Edition, Springer, p. 17

Maschler M., Solan E. and Zamir S. (2013) *Game Theory*, Cambridge University Press, pp. 825–831

**Examples**

```
library(CoopGame)
weightedVotingGameValue(S=c(1,2,3),w=c(1,2,3),q=4)
```

---

weightedVotingGameVector

*Compute game vector for a weighted voting game (aka quota game)*

---

**Description**

**Game vector for a specified weighted voting game:**

For further information see [weightedVotingGame](#)

**Usage**

```
weightedVotingGameVector(n, w, q)
```

**Arguments**

n	represents the number of players
w	numeric vector which contains the weight of each player
q	is the quota

**Value**

Game Vector where each element contains 1 if the sum of the weights of coalition S is greater or equal than quota q, else 0

**Author(s)**

Jochen Staudacher <jochen.staudacher@hs-kempten.de>

Johannes Anwander <anwander.johannes@gmail.com>

Michael Maerz

**References**

Peleg B. (2002) "Game-theoretic analysis of voting in committees". in: Handbook of Social Choice and Welfare 1, pp. 195–201

Peleg B. and Sudhoelter P. (2007) *Theory of cooperative games*, 2nd Edition, Springer, p. 17

Maschler M., Solan E. and Zamir S. (2013) *Game Theory*, Cambridge University Press, pp. 825–831

**Examples**

```
library(CoopGame)
```

```
weightedVotingGameVector(n=3,w=c(1,2,3),q=4)
```

# Index

- \* **ParameterChecks\_CoopGame**
  - getEmptyParamCheckResult, 85
  - stopOnInconsistentEstateAndClaimsVector, 153
  - stopOnInvalidAllocation, 154
  - stopOnInvalidBoolean, 155
  - stopOnInvalidClaimsVector, 156
  - stopOnInvalidCoalitionS, 157
  - stopOnInvalidDictator, 158
  - stopOnInvalidEstate, 159
  - stopOnInvalidGameVector, 160
  - stopOnInvalidGrandCoalitionN, 162
  - stopOnInvalidIndex, 163
  - stopOnInvalidLeftRightGloveGame, 164
  - stopOnInvalidNChooseB, 165
  - stopOnInvalidNumber, 166
  - stopOnInvalidNumberOfPlayers, 167
  - stopOnInvalidQuota, 168
  - stopOnInvalidVetoPlayer, 169
  - stopOnInvalidWeightVector, 170
  - stopOnParamCheckError, 171
- absoluteHollerValue
  - (absolutePublicGoodValue), 6
- absolutePublicGoodValue, 6
- absolutePublicHelpChiValue, 7
- absolutePublicHelpThetaValue
  - (absolutePublicHelpValue), 8
- absolutePublicHelpValue, 8
- absolutePublicHelpValueChi
  - (absolutePublicHelpChiValue), 7
- absolutePublicHelpValueTheta
  - (absolutePublicHelpValue), 8
- absolutePublicHelpValueXi
  - (absolutePublicHelpChiValue), 7
- absolutePublicHelpXiValue
  - (absolutePublicHelpChiValue), 7
- apexGame, 9, 10, 11
- apexGameValue, 9, 10
- apexGameVector, 9, 11
- bankruptcyGame, 12, 13, 14
- bankruptcyGameValue, 12, 13
- bankruptcyGameVector, 12, 14
- banzhafValue, 16
- baruaChakravartySarkarIndex, 17
- belongsToCore, 18
- belongsToCoreCover, 19
- belongsToImputationset, 20
- belongsToReasonableSet, 21
- belongsToWeberset, 22
- cardinalityGame, 23, 24, 25
- cardinalityGameValue, 23, 24
- cardinalityGameVector, 23, 25
- centroidCore, 25
- centroidCoreCover, 26
- centroidImputationSet, 27
- centroidReasonableSet, 28
- centroidWeberSet, 29
- colemanCollectivityPowerIndex, 30
- colemanInitiativePowerIndex, 31
- colemanPreventivePowerIndex, 32
- coreCoverVertices, 33
- coreVertices, 34
- costSharingGame, 35, 36, 37
- costSharingGameValue, 35, 36
- costSharingGameVector, 35, 37
- createBitMatrix, 39
- deeganPackelIndex, 40
- dictatorGame, 41, 42, 43
- dictatorGameValue, 41, 42
- dictatorGameVector, 41, 43
- disruptionNucleolus, 44
- divideTheDollarGame, 45, 46, 47
- divideTheDollarGameValue, 45, 46
- divideTheDollarGameVector, 45, 47
- drawCentroidCore, 48

- drawCentroidCoreCover, 49
- drawCentroidImputationSet, 50
- drawCentroidReasonableSet, 51
- drawCentroidWeberSet, 52
- drawCore, 53
- drawCoreCover, 54
- drawDeeganPackelIndex, 55
- drawDisruptionNucleolus, 56
- drawGatelyValue, 57
- drawImputationSet (drawImputationset), 58
- drawImputationset, 58
- drawJohnstonIndex, 59
- drawModiclus, 60
- drawNormalizedBanzhafIndex, 61
- drawNormalizedBanzhafValue, 62
- drawNucleolus, 63
- drawPerCapitaNucleolus, 64
- drawPreNucleolus, 66
- drawProportionalNucleolus, 67
- drawPublicGoodIndex, 68
- drawPublicGoodValue, 69
- drawPublicHelpChiIndex, 70
- drawPublicHelpChiValue, 71
- drawPublicHelpIndex, 72
- drawPublicHelpValue, 73
- drawReasonableSet, 74
- drawShapleyShubikIndex, 75
- drawShapleyValue, 76, 150
- drawSimplifiedModiclus, 77
- drawTauValue, 78
- drawTijisValue (drawTauValue), 78
- drawWeberset, 79
  
- equalPropensityToDisrupt, 80
  
- gatelyPoint (gatelyValue), 81
- gatelyValue, 81
- getCriticalCoalitionsOfPlayer, 82
- getDualGameVector, 84
- getEmptyParamCheckResult, 85, 154–163, 165–172
- getExcessCoefficients, 86
- getGainingCoalitions, 87
- getGapFunctionCoefficients, 88
- getHarsanyiDividends (getUnanimityCoefficients), 96
- getkCover, 88
- getMarginalContributions, 89
  
- getMinimalRights, 90
- getMinimalRightsVector (getMinimalRights), 90
- getMinimalWinningCoalitions (getMinimumWinningCoalitions), 91
- getMinimumWinningCoalitions, 91
- getNumberOfPlayers, 92
- getPerCapitaExcessCoefficients, 93
- getPlayersFromBitVector, 94
- getPlayersFromBMRow, 94
- getRealGainingCoalitions, 95
- getUnanimityCoefficients, 96
- getUtopiaPayoff, 97
- getUtopiaPayoffVector (getUtopiaPayoff), 97
- getVectorOfPropensitiesToDisrupt, 98
- getWinningCoalitions, 99
- getZeroNormalizedGameVector, 100
- getZeroOneNormalizedGameVector, 101
- gloveGame, 102, 103, 104
- gloveGameValue, 102, 103
- gloveGameVector, 102, 104
  
- hollerIndex (publicGoodIndex), 140
- hollerValue (publicGoodValue), 141
  
- imputationSetVertices (imputationsetVertices), 105
- imputationsetVertices, 105
- is1ConvexGame, 106
- isAdditiveGame, 107
- isBalancedGame, 108
- isConstantSumGame, 109
- isConstantsumGame (isConstantSumGame), 109
- isConvexGame, 110
- isDegenerateGame, 111
- isEssentialGame, 112
- isInEssentialGame (isDegenerateGame), 111
- isInessentialGame (isDegenerateGame), 111
- iskConvexGame, 113
- isMonotonicGame, 115
- isNonnegativeGame, 116
- isQuasiBalancedGame, 117
- isSemiConvexGame, 118
- isSimpleGame, 119

- isSuperAdditiveGame  
(isSuperadditiveGame), 120
- isSuperadditiveGame, 120
- isSymmetricGame, 121
- isWeaklyConstantSumGame, 122
- isWeaklyConstantsumGame  
(isWeaklyConstantSumGame), 122
- isWeaklySuperAdditiveGame  
(isWeaklySuperadditiveGame),  
123
- isWeaklySuperadditiveGame, 123
- isZeroMonotonicGame  
(isWeaklySuperadditiveGame),  
123
  
- johnstonIndex, 124
  
- koenigBraeuningerIndex, 125
  
- majoritySingleVetoGame, 126, 127, 128
- majoritySingleVetoGameValue, 127, 127
- majoritySingleVetoGameVector, 127, 128
- modiclus, 129
  
- nevisonIndex, 130
- nonNormalizedBanzhafIndex, 131
- nonNormalizedBanzhafValue  
(banzhafValue), 16
- normalizedBanzhafIndex, 132
- normalizedBanzhafValue, 133
- nucleolus, 134
  
- perCapitaNucleolus, 135
- Preucleolus, 136
- preucleolus (Preucleolus), 136
- propensityToDisrupt, 138
- proportionalNucleolus, 139
- publicGoodIndex, 140
- publicGoodValue, 141
- publicHelpChiIndex, 142
- publicHelpChiValue, 143
- publicHelpIndex, 144
- publicHelpIndexChi  
(publicHelpChiIndex), 142
- publicHelpIndexTheta (publicHelpIndex),  
144
- publicHelpIndexXi (publicHelpChiIndex),  
142
- publicHelpThetaIndex (publicHelpIndex),  
144
  
- publicHelpThetaValue (publicHelpValue),  
145
- publicHelpValue, 145
- publicHelpValueChi  
(publicHelpChiValue), 143
- publicHelpValueTheta (publicHelpValue),  
145
- publicHelpValueXi (publicHelpChiValue),  
143
- publicHelpXiIndex (publicHelpChiIndex),  
142
- publicHelpXiValue (publicHelpChiValue),  
143
  
- quotaGame (weightedVotingGame), 177
- quotaGameValue  
(weightedVotingGameValue), 178
- quotaGameVector  
(weightedVotingGameVector), 179
  
- raeIndex, 146
- rawBanzhafIndex, 147
- rawBanzhafValue, 148
- reasonableSetVertices, 149
  
- shapleyShubikIndex, 150
- shapleyValue, 151
- simplifiedModiclus, 152
- stopOnInconsistentEstateAndClaimsVector,  
85, 153, 155–163, 165–172
- stopOnInvalidAllocation, 85, 154, 154,  
156–163, 165–172
- stopOnInvalidBoolean, 85, 154, 155, 155,  
157–163, 165–172
- stopOnInvalidClaimsVector, 85, 154–156,  
156, 158–163, 165–172
- stopOnInvalidCoalitionS, 85, 154–157,  
157, 159–163, 165–172
- stopOnInvalidDictator, 85, 154–158, 158,  
160–163, 165–172
- stopOnInvalidEstate, 85, 154–159, 159,  
161–163, 165–172
- stopOnInvalidGameVector, 85, 154–160,  
160, 162, 163, 165–172
- stopOnInvalidGrandCoalitionN, 85,  
154–161, 162, 163, 165–172
- stopOnInvalidIndex, 85, 154–162, 163,  
165–172

stopOnInvalidLeftRightGloveGame, [85](#),  
[154–163](#), [164](#), [166–172](#)

stopOnInvalidNChooseB, [85](#), [154–163](#), [165](#),  
[165](#), [167–172](#)

stopOnInvalidNumber, [85](#), [154–163](#), [165](#),  
[166](#), [166](#), [168–172](#)

stopOnInvalidNumberOfPlayers, [85](#),  
[154–163](#), [165–167](#), [167](#), [169–172](#)

stopOnInvalidQuota, [85](#), [154–163](#), [165–168](#),  
[168](#), [170–172](#)

stopOnInvalidVetoPlayer, [85](#), [154–163](#),  
[165–169](#), [169](#), [171](#), [172](#)

stopOnInvalidWeightVector, [85](#), [154–163](#),  
[165–170](#), [170](#), [172](#)

stopOnParamCheckError, [85](#), [154–163](#),  
[165–171](#), [171](#)

tauValue, [172](#)

tijsValue (tauValue), [172](#)

unanimityGame, [173](#), [174](#), [175](#)

unanimityGameValue, [173](#), [174](#)

unanimityGameVector, [173](#), [175](#)

webersetVertices, [176](#)

weightedVotingGame, [177](#), [178](#), [179](#)

weightedVotingGameValue, [177](#), [178](#)

weightedVotingGameVector, [177](#), [179](#)