Description

G. Owen (1975, Math. Programming 9, 358-370) assigned to each linear production process a cooperative game, a “linear production game”. Further, he introduced a method to find a subset of the core of linear production games that verifies certain properties, which is called the “Owen set.” This package computes the linear production games and allocation rules associated.

Details

Package: coopProductGame
Type: Package
Version: 2.0
Date: 2018-07-01
License: GPL-3

The most important function is coopProductGame. Other functions included in the package are auxiliary ones that can be used independently.

Author(s)

Daniel Prieto Rodríguez
Maintainer: Daniel Prieto Rodríguez <daniel.prieto.rodriguez89@gmail.com>

References

coalitions

Coalitions for a given numbers of players n.

Description
This functions gives all the coalitions, including the empty coalition, for a number of players n.

Usage
coalitions(n)

Arguments
n Number of players.

Value
A list with the following components:

Binary Matrix where each row is a binary representation of the coalition.
Usual Vector with the usual configurations of the coalitions.

Author(s)
D. Prieto

Examples
# Number of players:
n <- 3
# Associated coalitions:
coalitions(n)

# $Binary
# [1,] 0 0 0
# [2,] 1 0 0
# [3,] 0 1 0
# [4,] 0 0 1
# [5,] 1 1 0
# [6,] 1 0 1
# [7,] 0 1 1
# [8,] 1 1 1
#
# $Usual
# [1] 0 1 2 3 12 13 23 123
coopProductGame  Cooperative linear production games

Description

Given a linear production problem \( \mathbf{A}\mathbf{x} \leq \mathbf{b} \), the coopProductGame solves the problem by making use of lpSolveAPI where each agent provides his own resources.

Usage

go:: cooperate:: coopProductGame(c, A, B, plot = FALSE, show.data = FALSE)

Arguments

c  vector containing the benefits of the products.
A  production matrix.
B  matrix containing the amount of resources of the several players where each row is one player.
plot  logical value indicating if the function displays graphical solution (TRUE) or not (FALSE). Note that this option only makes sense when we have a two-dimension problem.
show.data  logical value indicating if the function displays the console output (TRUE) or not (FALSE). By default the value is TRUE.

Value

go:: cooperate:: coopProductGame returns a list with the solution of the problem, the objective value and a Owen allocation if it exists. If we have a two dimension dual problem, the function returns all the Owen allocations (if there are more than one we obtain the end points of the segment that contains all possible allocations.)

Author(s)

D. Prieto

Examples

# Vector of benefits
c <- c(68, 52)
# Production matrix
A <- matrix(c(4, 5, 6, 2), ncol = 2, byrow = TRUE)
# Matrix of resources. Each row is the vector of resources of each player
B <- matrix(c(4, 6, 60, 33, 39, 0), ncol = 3, byrow = TRUE)
# Solution of the associated linear production game
coopProductGame(c, A, B, show.data = TRUE)

# ____________________________________________________________
linearProductionGame  Cooperative linear production games

Description

Given a linear production problem, the linearProductionGame function solves the problem by making use of lpSolveAPI where each agent provides his own resources.

Usage

linearProductionGame(c, A, B, plot = FALSE, show.data = FALSE)

Arguments

c vector containing the benefits of the products.
A production matrix.
B matrix containing the amount of resources of the several players where each row is one player.
plot logical value indicating if the function displays graphical solution (TRUE) or not (FALSE). Note that this option only makes sense when we have a two-dimension problem.
show.data logical value indicating if the function displays the console output (TRUE) or not (FALSE). By default the value is TRUE.
Value

`linearProductionGame` returns a list with the solutions of the associated problem of each coalition and the objective value for coalition N.

Author(s)

D. Prieto

Examples

```r
# Vector of benefits
c <- c(68, 52)
# Production matrix
A <- matrix(c(4, 5, 6, 2), ncol = 2, byrow = TRUE)
# Matrix of resources. Each column is the vector of resources of each player
B <- matrix(c(4, 6, 60, 33, 39, 0), ncol = 3, byrow = TRUE)
# Solution of the associated linear production game
linearProductionGame(c, A, B, show.data = TRUE)

# --------------------------------------------------------------------------------------
# Optimal solution of the problem for each coalition:
# --------------------------------------------------------------------------------------
#
# S={1} 1.00 0.00
# S={2} 1.50 0.00
# S={3} 0.00 0.00
# S={1,2} 2.50 0.00
# S={1,3} 1.68 11.45
# S={2,3} 2.86 10.91
# S={1,2,3} 10.00 6.00
#
# --------------------------------------------------------------------------------------
# Cooperative production game:
# --------------------------------------------------------------------------------------
# S={0} S={1} S={2} S={3} S={1,2} S={1,3} S={2,3} S={1,2,3}
# Associated game 0 68 102 0 170 710 762 992
# --------------------------------------------------------------------------------------
```

Description

Given a linear production problem $Ax \leq b$, the `makeLP` function creates a new `lpSolve` linear program model object.
**nucleolus**

**Usage**

```r
makeLP(c, A, b)
```

**Arguments**

- `c`: vector of benefits.
- `A`: production matrix.
- `b`: vector of resources.

**Value**

`makeLP` returns a `lpSolve` linear program model object. Specifically an R external pointer with class `lpExtPtr`.

**Author(s)**

D. Prieto

**Examples**

```r
# Vector of benefits
c <- c(68,52)
# Production matrix
A <- matrix(c(4, 5, 6, 2), ncol = 2, byrow = TRUE)
# Vector of resources
b <- c(4,33)
# Make the associated linear production problem
prod <- makeLP(c, A, b)
```

---

**nucleolus**  

* **Nucleolus solution**

**Description**

This function computes the nucleolus solution of a game with a maximum of 4 agents.

**Usage**

```r
nucleolus(game, show.data = FALSE)
```

**Arguments**

- `game`: a vector that represents the cooperative game.
- `show.data`: logical value indicating if the function displays the console output (TRUE) or not (FALSE). By default the value is FALSE.
Value

nucleolus returns and prints the Nucleolus Solution of associated cooperative game.

Author(s)

D. Prieto

Examples

```r
# Cooperative game
game <- c(68, 102, 0, 170, 710, 762, 992)
# Nucleolus solution
nucleolus(game, show.data = TRUE)
```

```
# ------------------------
# Nucleolus Solution
# ------------------------
# [1] "(149, 192, 651)"
```

---

### Owen Set

**Owen Set**

**Description**

This function computes the Owen Set of a linear production game

**Usage**

```r
owenSet(c, A, B, show.data = FALSE)
```

**Arguments**

- `c` vector containing the benefits of the products.
- `A` production matrix.
- `B` matrix containing the amount of resources of the several players where each row is one player.
- `show.data` logical value indicating if the function displays the console output (TRUE) or not (FALSE). By default the value is FALSE.

**Value**

`owenSet` returns and prints the Owen Set of associated linear production problem.

**Author(s)**

D. Prieto
Examples

# Vector of benefits
c <- c(68, 52)
# Production matrix
A <- matrix(c(4, 5, 6, 2), ncol=2, byrow = TRUE)
# Matrix of resources. Each row is the vector of resources of each player
B <- matrix(c(4, 6, 60, 33, 39, 0), ncol = 3, byrow = TRUE)
# Solution of the associated linear production game
owenSet(c, A, B, show.data = TRUE)

# The linear production problem has a unique Owen's allocation:
# [1] "(230, 282, 480)"

plotCoreSet

Plot Core Set for cooperative production linear games.

Description

Given a linear production game, the plotCoreSet function plots the imputation Set, Core Set and the most common solutions (Nucleolus, Shapley Value and allocations of the Owen Set).

Usage

plotCoreSet(c, A, B)

Arguments

- `c`: vector containing the benefits of the products.
- `A`: production matrix.
- `B`: matrix containing the amount of resources of the several players where each row is one player.

Details

In most cases the Owen Set consists of a single allocation, but in some cases there are infinities. In the case that there are infinite allocations, if the problem has two dimensions, they will be given by a line, which we will represent graphically. If the problem has more than two dimensions, an allocation of all possible ones will be represented.

Value

plotCoreSet returns a ggplot object with the imputation set of the game, the core and the most common solutions.
plotlm

**Description**

This function plots the graphical solution of simple linear production programming problems with two decision variables. The decision variables must be real, nonnegative and cannot have a finite upper bound. Only inequality constraints are supported.

**Usage**

```r
plotlm(prod, A, b, c, title = NULL)
```

**Arguments**

- `prod` a linear production programming problem of class `lpExtPtr`.
- `A` production matrix.
- `b` vector of resources.
- `c` vector of benefits.
- `title` title of the plot. By default is `NULL`, so it returns a plot without title.

**Value**

Returns and plot a `ggplot` object with graphical solution of the problem.

**Author(s)**

D. Prieto

---

**Author(s)**

D. Prieto

**See Also**

`coopProductGame`

**Examples**

```r
# Vector of benefits
c <- c(68, 52)
# Production matrix
A <- matrix(c(4, 5, 6, 2), ncol = 2, byrow = TRUE)
# Matrix of resources. Each row is the vector of resources of each player
B <- matrix(c(4, 6, 60, 33, 39, 0), ncol = 3, byrow = TRUE)
# Solution of the associated linear production game
plotCoreSet(c, A, B)
```
See Also

makeLP.

Examples

# Vector of benefits
c <- c(68, 52)
# Matrix of coefficients
A <- matrix(c(4, 5, 6, 2), ncol = 2, byrow = TRUE)
# Vector of resources
b <- c(4, 33)
# Make the associated linear program
prod <- makeLP(c, A, b)
plotLM(prod, A, b, c)

productLinearProblem  Linear production programming problems

Description

Given a linear production programming problem $A \times x \leq b$, the productLinearProblem solves the problem by making use of lpSolveAPI.

Usage

productLinearProblem(c, A, b, plot = FALSE, show.data = FALSE)

Arguments

c  vector of benefits.
A  production matrix.
b  vector of resources.
plot logical value indicating if the function displays graphical solution (TRUE) or not (FALSE). Note that this option only makes sense when we have a two-dimension problem.
show.data  logical value indicating if the function displays the console output (TRUE) or not (FALSE). By default the value is TRUE.

Value

productLinearProblem returns and prints a list with the following components:

ObjectiveValue  Value of the objective function from a successfully solved linear production programming problem.

OptimalSolution  Values of the variables from a successfully solved linear production programming problem.
Description

Calculates the Shapley Value for a N-agent cooperative game.

Usage

shapleyValue(game, show.data = FALSE)

Arguments

game a vector that represents the cooperative game.

show.data logical value indicating if the function displays the console output (TRUE) or not (FALSE). By default the value is FALSE.

Value

shapleyValue returns and prints the Shapley Value of associated cooperative game.
Author(s)

D. Prieto

Examples

# Cooperative game
game <- c(68, 102, 0, 170, 710, 762, 992)
# Shapley Value
shapleyValue(game, show.data = TRUE)

# -------------------------------
# Shapley Value Solution:
# -------------------------------
# [1] "(229, 272, 491)"
Index

coalitions, 3
coopProducGame-package, 2
coopProductGame, 2, 4, 10
linearProductionGame, 5
makeLP, 6, 11
nucleolus, 7
owenSet, 8
plotCoreSet, 9
plotLm, 10
productLinearProblem, 11
shapleyValue, 12