# Package: AirportProblems (via r-universe)

May 18, 2026

**Type** Package

**Title** Analysis of Cost Allocation for Airport Problems

**Version** 0.1.0

**Date** 2025-04-29

**Maintainer** Alejandro Bernárdez Ferradás <alejandro.bernardez@uvigo.es>

**Description** Airport problems, introduced by Littlechild and Owen (1973) <https://www.jstor.org/stable/2629727>, are cost allocation problems where agents share the cost of a facility (or service) based on their ordered needs. Valid allocations must satisfy no-subsidy constraints, meaning that no group of agents contributes more than the highest cost of its members (i.e., no agent is allowed to subsidize another). A rule is a mechanism that selects an allocation vector for a given problem. This package computes several rules proposed in the literature, including both standard rules and their variants, such as weighted versions, rules for clones, and rules based on the agents' hierarchy order. These rules can be applied to various problems of interest, including the allocation of liabilities and the maintenance of irrigation systems, among others. Moreover, the package provides functions for graphical representation, enabling users to visually compare the outcomes produced by each rule, or to display the no-subsidy set. In addition, it includes four datasets illustrating different applications and examples of airport problems. For a more detailed explanation of all concepts, see Thomson (2024) <doi:10.1016/j.mathsocsci.2024.03.007>.

**License** GPL-3

**Depends** R (>= 3.5)

**Imports** graphics, grDevices, magrittr, plotly, stats, utils

**Encoding** UTF-8

**LazyData** True

**NeedsCompilation** no

**RoxygenNote** 7.3.2

**URL** https://github.com/alexbernardez/AirportProblems

**BugReports** https://github.com/alexbernardez/AirportProblems/issues

**Config/pak/sysreqs** cmake make libicu-dev libuv1-dev libssl-dev

**Repository** https://alexbernardez.r-universe.dev

**Date/Publication** 2025-08-19 18:02:20 UTC

**RemoteUrl** https://github.com/alexbernardez/airportproblems

**RemoteRef** HEAD

**RemoteSha** 204ae9a980f014ce972d76e038b63c88a3b85179

# Contents

`airportgame` *Coalitional game associated with an airport problem*

## Description

`airportgame` computes the coalitional game for cost-sharing in an airport problem.

## Usage

```
airportgame(c, lex = TRUE)
```

## Arguments

| | |
|---|---|
| `c` | A numeric cost vector. |
| `lex` | A logical value indicating the output order of the game. By default, `lex = TRUE` returns the game in lexicographic order. However, if `lex = FALSE`, the game is returned in binary order. |

## Details

Let $N = \{1, \ldots, n\}$ denote the set of agents, and let $c \in \mathbb{R}^N_+$ be the cost vector such that $c \geq 0$. The value $c_i$ should be interpreted as the associated cost for each agent $i$ in the context of the problem, i.e., every component represents the cost of the facility required by an agent. Segmental costs are defined as the difference between a given cost and the first immediately lower cost: $c_i - c_{i-1}$ for $i \in N \backslash \{1\}$. Therefore, $C^N$ represents the domain of all problems.

Given an airport problem $c \in \mathbb{R}^N_+$, the corresponding coalitional game is defined, for each $S \subseteq N$, as:

$$v(S) = \max\{c_j : j \in S\}.$$

It is easy to check that this class of games associated with airport problems is always concave, since for any pair of coalitions $S \subseteq T \subseteq N$, it is verified that:

$$v(S \cup \{i\}) - v(S) \geq v(T \cup \{i\}) - v(T) \quad \text{for all} \quad i \notin T$$

An efficient way to represent a nonempty coalition $S \in 2^N$ is by identifying it with the binary sequence $a_n, a_{n-1}, \ldots a_1$, where $a_i = 1$ if $i \in S$ and $a_i = 0$ otherwise. Consequently, each coalition $S$ is represented by the number associated with its binary representation: $\sum_{i \in S} 2^{i-1}$. Then coalitions can be ordered by their associated numbers.

Alternatively, coalitions can be ordered lexicographically, meaning they are first sorted by increasing size, and then by lexicographic order among coalitions of the same size.

## Value

A numeric vector representing the associated coalitional game.

## References

Littlechild, S. C. and Owen, G. (1973). A simple expression for the Shapley value in a special case. *Management Science*, 23, 370-372.

Thomson, W. (2024). Cost allocation and airport problems. *Mathematical Social Sciences*, 31(C), 17–31.

## See Also

airportvector

## Examples

```
# 4 agents
(c <- c(1, 3, 7, 10)) # Vector of costs
airportgame(c, lex = TRUE)  # Game in lexicographic order
airportgame(c, lex = FALSE) # Game in binary order
```

---

# airportvector *Cost vector associated with an airport game*

---

## Description

airportvector computes the cost vector corresponding to an airport problem.

## Usage

```
airportvector(v, lex = TRUE)
```

## Arguments

| | |
|---|---|
| v | A numeric vector that represents the characteristic function of the airport game. |
| lex | A logical value indicating the input order of the game. By default, if lex = TRUE, the game has been introduced in lexicographic order. However, if lex = FALSE, the game has been established in binary order. |

## Details

A cooperative game $(N, v)$ is considered an airport game provided that its characteristic function $v$ satisfies:

$$v(S) = \max\{c_j : j \in S\}, \quad \forall S \subseteq N, S \neq \emptyset$$

where $c_j$ represents the individual cost associated with each agent $j$.

Evidently, this property implies that the cost assigned to a coalition is determined by the most expensive cost for its members. It is for this reason that this class of games is always concave.

The airport game can be given in lexicographic order or binary order. For instance, if $n = 3$, the characteristic function of the associated airport game in lexicographic order is:

$$v = [v(\{1\}), v(\{2\}), v(\{3\}), v(\{1,2\}), v(\{1,3\}), v(\{2,3\}), v(\{1,2,3\})]$$

On the other hand, in binary order, it would be:

$$v = [v(\{1\}), v(\{2\}), v(\{1,2\}), v(\{3\}), v(\{1,3\}), v(\{2,3\}), v(\{1,2,3\})]$$

Anyway, in both cases, we have that $v(\{2\}) = v(\{1,2\})$ and $v(\{3\}) = v(\{1,3\}) = v(\{2,3\}) = v(\{1,2,3\})$.

Given an airport game $(N, v)$, it is possible to extract the corresponding cost vector $c$ by setting:

$$c_j = \min\{v(S) : j \in S\}, \quad \text{for all} \quad j \in N.$$

## Value

A numeric vector representing the cost for each agent of an airport game.

## References

Littlechild, S. C. and Owen, G. (1973). A simple expression for the Shapley value in a special case. *Management Science*, 23, 370-372.

Thomson, W. (2024). Cost allocation and airport problems. *Mathematical Social Sciences*, 31(C), 17–31.

## See Also

airportgame

## Examples

```
# 4 agents in lexicographic order
v <- c(1, 3, 7, 10, 3, 7, 10, 7, 10, 10, 7, 10, 10, 10, 10)
airportvector(v, lex = TRUE)

# 4 agents in binary order
u <- c(1, 3, 3, 7, 7, 7, 7, 10, 10, 10, 10, 10, 10, 10, 10)
airportvector(u, lex = FALSE)
```

---

# basicrule *Allocation rule*

---

## Description

`basicrule` calculates the contribution vector resulting from the payment allocation among the different agents using one of the various predefined rules.

## Usage

```
basicrule(c, rule, a = NULL, order = NULL)
```

## Arguments

| | |
|---|---|
| `c` | A numeric cost vector. |
| `rule` | A character string specifying the rule to apply.The rules that can be selected are: `"SFC"`, `"SEC"`, `"CEC"`, `"CP"`, `"CEB"`, `"SM"`, `"CC"`, `"SIGMA"` and `"PRIOR"`. |
| `a` | A numeric value in the range [0,1], controlling the parameterization of the rule. It can only be defined when `rule = "SIGMA"`. By default, `a = 0.5`. |
| `order` | A numeric vector indicating the priority order of agents when making contributions. It can only be defined when `rule = "PRIOR"`. By default, agents follow their original indexing and contribute accordingly. |

## Details

A rule is a mapping $\mathcal{R} : C^N \to \mathbb{R}^N$ which associates with each problem $c \in C^N$ a cost allocation vector $\mathcal{R}(c)$ such that $0 \leq \mathcal{R}(c) \leq c$. In other words, a rule is a mechanism that selects for each airport problem an allocation vector.

The various proposed rules, despite their differences, share a key characteristic: for any given problem, each rule selects an allocation vector that belongs to its no-subsidy set. Although these rules have been individually characterized in different functions, the one in question encompasses all of them.

## Value

A numeric contribution vector, where each element represents the payment of the different agents.

## Note

When `rule = "CC"`, the execution time of the function may significantly increase if the number of agents exceeds 150.

## References

Bernárdez Ferradás, A., Mirás Calvo, M. Á., Quinteiro Sandomingo, C., and Sánchez-Rodríguez, E. (2025). Airport problems with cloned agents. [Preprint manuscript].

Potters, J. and Sudhölter, P. (1999). Airport problems and consistent allocation rules. *Mathematical Social Sciences*, 38, 83–102.

Thomson, W. (2024). Cost allocation and airport problems. *Mathematical Social Sciences*, 31(C), 17–31.

## See Also

`NSset`, `weightedrule`, `clonesrule`, `hierarchicalrule`, `multibasicrules`

## Examples

```
c <- c(1, 3, 7, 10) # Cost vector

# SEC rule
basicrule(c, rule = "SEC")

# PRIOR rule
order <- c(1, 3, 2, 4)
basicrule(c, "PRIOR", order = order)
```

---

`Birmingham` *Birmingham Airport data*

---

## Description

The data includes several variables describing 13,572 aircraft operations — 6,876 take-offs and 6,876 landings — involving 11 different aircraft types at Birmingham Airport during the period 1968–1969.

## Usage

```
Birmingham
```

## Format

A data frame with 11 rows and 9 columns (variables):

**type** Aircraft type

**i** Aircraft index (subscript)

**eta** Number of movements

**p** Average fee per movement (in pounds)

**d** Estimated diversion cost per movement (in pounds)

**b** Average benefit per movement (in pounds)

**l** Runway user index

**c** Maintenance cost per movement (in pounds)

**g** Assumed capital cost (in pounds)

## Source

Littlechild, S. C. and Thompson, G. F. (1977). Aircraft landing fees: a game theory approach. *The Bell Journal of Economics*, 8, 186-204.

## Examples

```
# Based on capital cost (g)
G <- multiclonesrules(Birmingham$g, Birmingham$eta, c("SEC", "CEC", "SM"),
group_contribution = FALSE, agents_names = Birmingham$i, labels = FALSE)
(cost_allocation <- round(G, 2))

# Based on capital cost (g) + add maintenance cost per movement (c)
sweep(cost_allocation, MARGIN = 2, Birmingham$c, FUN = "+")
```

---

`CCrule` *Core-center rule*

---

## Description

`CCrule` calculates the contribution vector selected by the CC rule.

## Usage

```
CCrule(c)
```

## Arguments

| | |
|---|---|
| `c` | A numeric cost vector. |

## Details

The core-center rule, CC, assigns to each $c \in C^N$ the contribution vector given by the mean value of $U(c) \sim U(\text{NS}(c))$, that is,

$$\text{CC}(c) = \mathbb{E}[U(\text{NS}(c))].$$

Therefore, this rule is the center of gravity of the set of allocations satisfying the no-subsidy constraints. It coincides with the core-center of the cooperative game $v$ associated with $c \in C^N$.

## Value

A numeric contribution vector, where each element represents the payment of the different agents.

## Note

The execution time of the function may significantly increase if the number of agents exceeds 150.

## References

González-Díaz, J. and Sánchez-Rodríguez, E. (2007). A natural selection from the core of a TU game: the core-center. *International Journal of Game Theory*, 36, 27-26.

González-Díaz, J., Mirás Calvo, M. Á., Quinteiro Sandomingo, C., and Sánchez-Rodríguez (2015). Monotonicity of the core-center of the airport game. *TOP*, 23, 773-798.

González-Díaz, J., Mirás Calvo, M. Á., Quinteiro Sandomingo, C., and Sánchez-Rodríguez (2016). Airport games: the core and its center. *Mathematical Social Sciences*, 82, 105-115.

## See Also

NSset, basicrule, clonesrule hierarchicalrule

## Examples

```
c <- c(1, 3, 7, 10) # Cost vector
CCrule(c)
```

---

`CEBrule` *Constrained equal benefits rule*

---

## Description

`CEBrule` calculates the contribution vector selected by the CEB rule.

## Usage

```
CEBrule(c)
```

## Arguments

| | |
|---|---|
| `c` | A numeric cost vector. |

## Details

For each $c \in C^N$ and each $i \in N$, the constrained equal benefits rule is defined by

$$\text{CEB}_i(c) = \max\{c_i - \beta,\ 0\}$$

where $\beta > 0$ is chosen so that $\sum_{i=1}^{n} \text{CEB}_i(c) = c_n$.

This rule focuses on the benefits each agent receives from not having to fully cover their own needs, aiming to distribute them as equitably as possible, without any agent subsidizing another.

The contribution selected by the CEB rule for a problem $c \in C^N$ coincides with the payoff vector assigned by the modified nucleolus.

## Value

A numeric contribution vector, where each element represents the payment of the different agents.

## References

Hu, C.-C., Tsay, M.-H., and Yeh, C.-H. (2012). Axiomatic and strategic justifications for the constrained equal benefits rule in the airport problem. *Games and Economic Behavior*, 75, 185-197.

Potters, J. and Sudhölter, P. (1999). Airport problems and consistent allocation rules. *Mathematical Social Sciences*, 38, 83–102.

Sudhölter, P. (1997). The modified nucleolus: Properties and axiomatizations. *International Journal of Game Theory*, 26, 146-182.

Thomson, W. (2024). Cost allocation and airport problems. *Mathematical Social Sciences*, 31(C), 17–31.

## See Also

basicrule, weightedrule, clonesrule, hierarchicalrule

## Examples

```
c <- c(1, 3, 7, 10) # Cost vector
CEBrule(c)
```

---

# CECrule *Constrained equal contributions rule*

---

## Description

CECrule calculates the contribution vector selected by the CEC rule.

## Usage

```
CECrule(c)
```

## Arguments

| | |
|---|---|
| c | A numeric cost vector. |

## Details

Let $N_-^i \{j \in N : c_j < c_i\}$. For each $c \in C^N$ and each $i \in N$, the constrained equal contributions rule is defined by

$$\text{CEC}_i(c) = \min \left\{ \frac{1}{r-i+1} \left( c_r - \sum_{j \in N_-^i} \text{CEC}_j(c) \right) : r = 1, \ldots, n \right\}.$$

This rule offers a different approach to achieving equality. Contributions are distributed as evenly as possible while ensuring compliance with the no-subsidy constraints.

The contribution selected by the CEC rule for a problem $c \in C^N$ coincides with the payoff vector assigned by the Dutta-Ray solution (denoted by EA) to the associated airport game $v \in G^N$, that is, $\text{CEC}(c) = \text{EA}(v)$.

## Value

A numeric contribution vector, where each element represents the payment of the different agents.

## References

Aadland, D. and Kolpin, V. (1998). Shared irrigation costs: an empirical and axiomatic analysis. *Mathematical Social Sciences*, 35, 203-218.

Thomson, W. (2024). Cost allocation and airport problems. *Mathematical Social Sciences*, 31(C), 17–31.

## See Also

SIGMArule, basicrule, weightedrule, clonesrule, hierarchicalrule

## Examples

```
c <- c(1, 3, 7, 10) # Cost vector
CECrule(c)
```

---

# clonesgroups *Cloned agents in an airport problem*

---

## Description

clonesgroups determines a new cost vector that excludes cloned agents, and calculates the size of each clone group that shares the same cost.

## Usage

```
clonesgroups(c)
```

## Arguments

| | |
|---|---|
| c | A numeric cost vector. |

## Details

In an airport problem, agents $i, j \in N$ are clones if both have the same cost parameter, that is, if $c_i = c_j$.

If a problem has cloned agents, then the agent set $N$ can be divided into several non-overlapping groups such that any pair of agents that belong to the same group are clones, but any two agents from two different groups have different cost parameters.

For each $t \in N$, let $T = \{1, \ldots, t\}$ and let $\mathcal{A}_t^N$ be the set of pairs $(\eta, c) \in \mathbb{N}^t \times \mathbb{R}^t$ such that:

$$\eta = (\eta_1, \ldots, \eta_t) \in \mathbb{N}^t \text{ with } \eta_1 + \cdots + \eta_t = n$$

$$c = (c_1, \ldots, c_t) \in C^T \text{ with } c_1 < \cdots < c_t$$

Given $t \in N$ and $(\eta, c) \in \mathcal{A}_t^N$ we define the cost problem $\eta * c \in C^N$ as:

$$\eta * c = (\eta_1 * c_1, \ldots, \eta_t * c_t) = (c_1, \overset{\eta_1}{\ldots}, c_1, \ldots, c_t, \overset{\eta_t}{\ldots}, c_t) \in C^N$$

Given a problem $d \in C^N$ there are unique $t \in N$ and $(\eta, c) \in \mathcal{A}_t^N$ such that $d = n * c$. We refer to the problem $c \in C^T$, formed by the different cost parameters of problem $d \in C^N$, as the reduced problem without clones associated with $d \in C^N$. Clearly, $d \in C^N$ is obtained from $c \in C^T$ by adding, for each $i \in T$, $\eta_i$, clones of agent $i$.

Let $t \in N$ and $(\eta, c) \in \mathcal{A}_t^{\mathcal{N}}$. For each $s \in T = \{1, \ldots, t\}$, let $M_s^\eta = \eta_1 + \cdots + \eta_s$ and, to simplify the notation, write $N_s^\eta = N_s^{\eta * c} = \{j \in N : (\eta * c)_j = c_s\}$. Therefore, $N_1^\eta = \{1, \ldots, M_1^\eta\}$ and $N_s^\eta = \{M_{s-1}^\eta + 1, \ldots, M_s^\eta\}$ if $s \in T \backslash \{1\}$. Obviously, the family $\{N_1^\eta, \ldots, N_t^\eta\}$ is a partition of $N$ and $|N_s^\eta| = \eta_s$ for all $s \in T$. Moreover, all the agents that belong to $N_s^\eta$ have the same cost parameter $c_s$, i.e., $(\eta * c)_j = c_s$ for all $s \in T$ and $j \in N_s^\eta$. So, each agent in the reduced problem $c \in C^T$ can be seen as a representative agent of the corresponding group of clones in the original problem $d = \eta * c \in C^N$.

## Value

A list containing the following items:

| | |
|---|---|
| `cw` | A numeric cost vector, with the same length as `eta`. |
| `eta` | A numeric vector representing the size of each group of cloned agents. |

## References

Bernárdez Ferradás, A., Mirás Calvo, M. Á., Quinteiro Sandomingo, C., and Sánchez-Rodríguez, E. (2025). Airport problems with cloned agents. [Preprint manuscript].

Littlechild, S. C. and Thompson, G. F. (1977). Aircraft landing fees: a game theory approach. *The Bell Journal of Economics*, 8, 186-204.

## See Also

`clonesproblem`

## Examples

```
# 9 different agents
c <- c(2, 2, 2, 5, 5, 7, 7, 7, 12) # Cost vector
clonesgroups(c) # 4 groups of cloned agents
```

`clonesproblem` *Cost vector with cloned agents*

## Description

`clonesproblem` determines the new cost vector after disaggregating the original groups of clones.

## Usage

```
clonesproblem(cw, eta)
```

## Arguments

| | |
|---|---|
| `cw` | A numeric cost vector, with the same length as `eta`. |
| `eta` | A numeric vector representing the size of each group of cloned agents. All its elements must be positive integers. |

## Details

In an airport problem, agents $i, j \in N$ are clones if both have the same cost parameter, that is, if $c_i = c_j$.

If a problem has cloned agents, then the agent set $N$ can be divided into several non-overlapping groups such that any pair of agents that belong to the same group are clones, but any two agents from two different groups have different cost parameters.

For each $t \in N$, let $T = \{1, \dots, t\}$ and let $\mathcal{A}_t^N$ be the set of pairs $(\eta, c) \in \mathbb{N}^t \times \mathbb{R}^t$ such that:

$$\eta = (\eta_1, \dots, \eta_t) \in \mathbb{N}^t \text{ with } \eta_1 + \cdots + \eta_t = n$$

$$c = (c_1, \dots, c_t) \in C^T \text{ with } c_1 < \cdots < c_t$$

Given $t \in N$ and $(\eta, c) \in \mathcal{A}_t^N$ we define the cost problem $\eta * c \in C^N$ as:

$$\eta * c = (\eta_1 * c_1, \dots, \eta_t * c_t) = (c_1, \overset{\eta_1}{\dots}, c_1, \dots, c_t, \overset{\eta_t}{\dots}, c_t) \in C^N$$

Given a problem $d \in C^N$ there are unique $t \in N$ and $(\eta, c) \in \mathcal{A}_t^N$ such that $d = n * c$. We refer to the problem $c \in C^T$, formed by the different cost parameters of problem $d \in C^N$, as the reduced problem without clones associated with $d \in C^N$. Clearly, $d \in C^N$ is obtained from $c \in C^T$ by adding, for each $i \in T$, $\eta_i$, clones of agent $i$.

Let $t \in N$ and $(\eta, c) \in \mathcal{A}_t^{\mathcal{N}}$. For each $s \in T = \{1, \dots, t\}$, let $M_s^\eta = \eta_1 + \cdots + \eta_s$ and, to simplify the notation, write $N_s^\eta = N_s^{\eta * c} = \{j \in N : (\eta * c)_j = c_s\}$. Therefore, $N_1^\eta = \{1, \dots, M_1^\eta\}$ and $N_s^\eta = \{M_{s-1}^\eta + 1, \dots, M_s^\eta\}$ if $s \in T \backslash \{1\}$. Obviously, the family $\{N_1^\eta, \dots, N_t^\eta\}$ is a partition of $N$ and $|N_s^\eta| = \eta_s$ for all $s \in T$. Moreover, all the agents that belong to $N_s^\eta$ have the same cost parameter $c_s$, i.e., $(\eta * c)_j = c_s$ for all $s \in T$ and $j \in N_s^\eta$. So, each agent in the reduced problem $c \in C^T$ can be seen as a representative agent of the corresponding group of clones in the original problem $d = \eta * c \in C^N$.

## Value

A numeric cost vector, where each element corresponds to a different agent's cost.

## References

Bernárdez Ferradás, A., Mirás Calvo, M. Á., Quinteiro Sandomingo, C., and Sánchez-Rodríguez, E. (2025). Airport problems with cloned agents. [Preprint manuscript].

Littlechild, S. C. and Thompson, G. F. (1977). Aircraft landing fees: a game theory approach. *The Bell Journal of Economics*, 8, 186-204.

## See Also

clonesgroups

## Examples

```
# 4 groups of cloned agents
cw <- c(2, 5, 7, 12) # Different costs
eta <- c(3, 2, 3, 1) # Size of each group of clones
clonesproblem(cw, eta) # General cost vector
```

---

# clonesrule *Allocation rule with cloned agents*

---

## Description

`clonesrule` calculates the contribution vector resulting from the payment allocation among the different agents using one of the versions for clones of the various predefined rules.

## Usage

```
clonesrule(cw, eta, rule, group_contribution = TRUE, a = NULL, order = NULL)
```

## Arguments

| | |
|---|---|
| `cw` | A numeric cost vector, with the same length as `eta`. |
| `eta` | A numeric vector representing the size of each group of cloned agents. All its elements must be positive integers. |
| `rule` | A character string specifying the rule to apply. The rules that can be selected are: `"SFC"`, `"SEC"`, `"CEC"`, `"CP"`, `"CEB"`, `"SM"`, `"CC"`, `"SIGMA"` and `"PRIOR"`. |
| `group_contribution` | A logical value. By default, if `group_contribution = TRUE`, the cost allocation vector stores the aggregated contribution for each group of clones. However, if `group_contribution = FALSE`, the cost allocation vector represents the individual contribution of one of the agents in the group of clones. |

`a` A numeric value in the range [0,1], controlling the parameterization of the rule. It can only be defined when `rule = "SIGMA"`. By default, `a = 0.5`.

`order` A numeric vector indicating the priority order of agents when making contributions. It can only be defined when `rule = "PRIOR"`. By default, agents follow their original indexing and contribute accordingly.

## Details

Let $\mathcal{R}$ be a rule, $t \in N$, and $(\eta, c) \in \mathcal{A}_t^N$. For each $i \in T = \{1, \ldots, t\}$, the sum of the contributions requested by $\mathcal{R}$ from the group of clones $N_i^\eta$ is $\mathcal{R}(\eta * c, N_i^\eta) = \sum_{j \in N_i^\eta} \mathcal{R}_j(n * c)$.

The computation of the cost allocation selected by a rule for a given problem can be substantially simplified where there are cloned agents. Through this function, a direct method is proposed to obtain either the aggregate contribution or the individual contribution of each group of cloned agents, based on the associated reduced problem and the number of clones in each group.

The version for clones of the SFC, SEC, CEC and CP rules is equal to their respective weighted version, so the formulation of these rules will be the same for the weighted version. Only the CEB rule has a weighted version and a clone version that are different.

If a rule $\mathcal{R}$ satisfies equal treatment of equals (cloned agents pay equal amounts), then $\mathcal{R}(\eta * c, N_i^\eta) = \eta_i \mathcal{R}_s(n * c)$ for any $s \in N_i^\eta$. All the rules listed, with the exception of the PRIOR rule, satisfy equal treatment of equals. Therefore, the contribution demanded by these rules from a group of clones is divided equally among them.

Finally, we define a $k$-replica of an airport problem as the problem in which every agent is replaced by $k$ clones of itself. If $k$ increases, the number of groups of cloned agents does not change, but the number of agents is large. Thus, we say that a rule satisfies replication invariance if for each $c \in C^N$, each $i \in N$, and each $k \in \mathbb{N}$, we have $\mathcal{R}(k * c, N_i^k) = \mathcal{R}_i(c)$; i.e., if in any $k$-replica of a problem each group of cloned agents contributes an amount independent of $k$. The SFC, SEC, CEC, CP, and PRIOR rules verify this property, while the others do not.

## Value

A numeric contribution vector. By default, if `group_contribution = TRUE`, each element represents the payment made by each group of cloned agents. However, if `group_contribution = FALSE`, each element reflects the individual payment made by a representative agent from each group.

## Note

When `rule = "CC"`, the execution time of the function may significantly increase if the number of individual agents exceeds 150.

## References

Bernárdez Ferradás, A., Mirás Calvo, M. Á., Quinteiro Sandomingo, C., and Sánchez-Rodríguez, E. (2025). Airport problems with cloned agents. [Preprint manuscript].

Littlechild, S. C. and Thompson, G. F. (1977). Aircraft landing fees: a game theory approach. *The Bell Journal of Economics*, 8, 186-204.

## See Also

clonesgroups, clonesproblem, multiclonesrules, basicrule, weightedrule

## Examples

```
# Clones SEC rule
cw <- c(1, 3, 7, 10) # Different costs
eta <- c(3, 4, 1, 2)  # Size of each groups of clones
clonesrule(cw, eta, "SEC")

# CEC rule satisfies replication invariance
cw <- c(1, 5, 12) # Different costs
eta <- rep(8, 3) # Size of each groups of clones
all.equal(clonesrule(cw, eta, "CEC"), CECrule(cw))
```

---

# comparisonallocations *Graphical evaluation of the contribution vectors*

---

## Description

comparisonallocations generates a graphical representation in which, for each agent or group of clones, both the maximum cost they can bear and their corresponding marginal contribution are displayed.

## Usage

```
comparisonallocations(
  c,
  contributions,
  col = NULL,
  colors = NULL,
  agents_names = NULL,
  labels = TRUE,
  legend = NULL,
  tol = 1e-06
)
```

## Arguments

| | |
|---|---|
| c | A numeric cost vector. |
| contributions | A list containing the different cost allocation vectors to be compared. It is required that the sum of the coordinates of each vector equals the total cost to be allocated. |
| col | A character string reflecting the color of the NS constraint for each agent. By default, the color "dodgerblue" is used. |

| | |
|---|---|
| `colors` | A vector that indicates the colors used to represent each contribution vector. By default, a color palette of different shades is used. |
| `agents_names` | A vector defining the name assigned to each agent. By default, the names follow a sequence of natural numbers, starting from 1. |
| `labels` | A logical value indicating whether the labels and the title of the plot should be displayed. By default, `labels = TRUE`. |
| `legend` | A vector or list where each of its elements represents a different contribution vector. By default, the coordinates of each contribution vector are displayed with two decimal places. |
| `tol` | Tolerance level for evaluating compliance with the NS constraint. |

## Details

For each $c \in C^N$ let $H(c) = \{x \in \mathbb{R} : x(N) = c_n\}$ be the hyperplane of $\mathbb{R}^N$ given by all the vectors whose coordinates add up to $c_n$. A cost allocation for $c \in C^N$ is a vector $x \in H(c)$ such that $0 \leq x \leq c$. The component $x_i$ is the contribution requested from agent $i$. Let $X(c)$ be the set of cost allocations for $c \in C^N$.

A basic requirement is that at an allocation $x \in X(c)$ on group $N' \subset N$ of agents would subsidize the other agents by contributing more than what the group would have to pay on its own. The no-subsidy constraint for the group $N' \subset N$ is $x(N') \geq \max\{c_j : j \in N'\}$. The set of cost allocations for $c \in C^N$ that satisfy the no-subsidy constraints, the no-subsidy set for short, is given by:

$$NS(c) = \{x \in X(c) : x(N') \leq \max\{c_j : j \in N'\}, \text{ for all } N' \subset N\}$$

$$= \{x \in \mathbb{R}^N : x \geq 0,\ x(N) = c_n,\ x_1 + \cdots + x_i \leq c_i, \text{ for all } i \in N \backslash \{n\}\}$$

Thus, the no-subsidy correspondence NS assigns to each $c \in C^N$ the set $NS(c)$.

A rule is a mapping $\mathcal{R} : C^N \to \mathbb{R}^N$ which associates with each problem $c \in C^N$ a contribution vector $\mathcal{R}(c) \in X(c)$. In other words, a rule is a mechanism that, for each airport problem, selects an allocation vector belonging to its no-subsidy set.

## Value

A vertical line plot in which each line represents the maximum amount an agent can pay without violating the NS constraint, while the points along the lines indicate the contributions made by the agent.

## References

Bernárdez Ferradás, A., Mirás Calvo, M. Á., Quinteiro Sandomingo, C., and Sánchez-Rodríguez, E. (2025). Airport problems with cloned agents. [Preprint manuscript].

Thomson, W. (2024). Cost allocation and airport problems. *Mathematical Social Sciences*, 31(C), 17–31.

## See Also

`NScheck`, `basicrule`, `plotallocations`

## Examples

```
# CEB rule vs weighted CEB rule vs clones CEB rule
c <- c(1, 3, 7, 10) # Cost vector
w <- c(1, 4, 8, 2) # Weight vector
eta <- w # Size of each groups of clones
CEB <- basicrule(c, "CEB")
wCEB <- weightedrule(c, w, "CEB")
clCEB <- clonesrule(c, w, "CEB")
comparisonallocations(c, list(CEB, wCEB, clCEB))

# SEC rule vs CEC rule vs SM rule
c <- c(5, 10, 20) # Cost vector
comparisonallocations(c, list(SECrule(c), CECrule(c), SMrule(c)),
col = "green", agents_names = c("Alex", "Estela", "Carmen"), labels = FALSE,
legend = list("SEC", "CEC", "SM"))
```

---

`CPrule` *Constrained proportional rule*

---

## Description

`CPrule` calculates the contribution vector selected by the CP rule.

## Usage

```
CPrule(c)
```

## Arguments

| | |
|---|---|
| `c` | A numeric cost vector. |

## Details

For each $c \in C^N$, let $c_0 = 0$ and $(q_0, q_1, \ldots, q_s) \in \mathbb{N}^{s+1}$, with $0 = q_0 < q_1 < \cdots < q_s = n$, defined recursively, for $j \geq 0$, by

$$q_{j+1} = \max\left\{ q \in N_+^{q_j} : \frac{c_q - c_{q_j}}{c_{q_j+1} + \cdots + c_q} = \min\left\{ \frac{c_r - c_{q_j}}{c_{q_j+1} + \cdots + c_r} : r \in N_+^{q_j} \right\} \right\}.$$

Then, for each $j \in \{0, \ldots, s-1\}$ and each $i \in Q_j = \{q_j + 1, \ldots, q_{j+1}\}$,

$$\mathrm{CP}_i(c) = \frac{c_i}{c(Q_j)}(c_{q_{j+1}} - c_{q_j}).$$

With this rule, calculating each agent's contribution is not always straightforward. When a coalition of agents violates the NS constraint, it becomes necessary to proceed in two or more steps.

The core idea of this rule is proportionality, aiming to ensure that agents' contributions are as close as possible to being proportional to the cost parameters, while respecting these constraints.

## Value

A numeric contribution vector, where each element represents the payment of the different agents.

## References

Bernárdez Ferradás, A., Mirás Calvo, M. Á., Quinteiro Sandomingo, C., and Sánchez-Rodríguez, E. (2025). Airport problems with cloned agents. [Preprint manuscript].

Thomson, W. (2024). Cost allocation and airport problems. *Mathematical Social Sciences*, 31(C), 17–31.

## See Also

basicrule, weightedrule, clonesrule, hierarchicalrule

## Examples

```
c <- c(1, 3, 7, 10) # Cost vector
CPrule(c)
```

---

# elevator *Elevator data*

---

## Description

The data includes 4 variables related to the 14 apartments in a building, within the context of distributing the costs associated with installing an elevator in that building.

## Usage

```
elevator
```

## Format

A data frame with 14 rows and 4 columns (variables):

**id** Unit identifier

**floor** Floor number

**c** Cumulative elevator installation cost for each dwelling (in euros).

**area** Area of each dwelling (in square meters).

## Source

Own elaboration.

## Examples

```
# By number of floors (floor)
cw <- clonesgroups(elevator$c)$cw
eta <- clonesgroups(elevator$c)$eta
rules <- c("SFC", "SEC", "CEC", "CP", "CC")
by_floor <- list()
for (rule in rules){
  by_floor[[rule]] <- clonesrule(cw, eta, rule)
}
comparisonallocations(cw, by_floor, agents_names = unique(elevator$floor),
legend = rules, labels = FALSE)

# Based on the area of each dwelling (area), by unit identifier (id)
c <- elevator$c
w <- elevator$area
by_id <- multiweightedrules(c, w, rules = c("SFC", "SEC", "CSEC", "CEC", "CP"),
         draw = TRUE, agents_names = elevator$id, labels = FALSE)
round(by_id, 2)
```

---

# hierarchicalrule *Allocation rule according to the agents' hierarchical order*

---

## Description

`hierarchicalrule` calculates the contribution vector resulting from the payment allocation among the different agents using one of the various predefined rules in relation to the agents' hierarchical order.

## Usage

```
hierarchicalrule(c, P, rule, a = NULL)
```

## Arguments

| | |
|---|---|
| `c` | A numeric cost vector. |
| `P` | A list showing the agents involved in the different distribution stages. |
| `rule` | A character string specifying the rule to apply.The rules that can be selected are: `"SFC"`, `"SEC"`, `"CEC"`, `"CP"`, `"CEB"`, `"SM"`, `"CC"` and `"SIGMA"`. |
| `a` | A numeric value in the range [0,1], controlling the parameterization of the rule. It can only be defined when `rule = "SIGMA"`. By default, `a = 0.5`. |

## Details

Let $N = \{1, \ldots, n\}$ be a finite set of agents and let $P = \{P_1, \ldots, P_{m+1}\}$ be a partition of $N$, with $m \leq n - 1$. So, the hierarchical structure, $P_> = \{P_1 > P_2 > \cdots > P_m > P_{m+1}\}$, implies that agents in $P_1$ have priority over agents in $N \backslash P_1$, agents in $P_2$ have priority over agents in $N \backslash P_1 \cup P_2$, and so on. Let $\mathcal{P}(N)$ denote the family of all hierarchical structures over $N$.

A hierarchical rule is a mapping $\mathcal{R}_P : C^N \to \mathbb{R}^N$ that, based on a hierarchical structure $P_>$, associates with each problem $c \in C^N$ a cost allocation vector $\mathcal{R}_P(c)$ such that $0 \leq \mathcal{R}_P(c) \leq c$. In other words, the allocation proceeds by assigning costs to agents in higher-hierarchy coalitions before those in lower-hierarchy ones, through the successive application of the rule to the no-subsidy faces of coalitions obtained from $P_>$. These rules are on the boundary of the no-subsidy set.

In each stage, the agents share the accumulated costs up to that point. However, the remaining costs are allocated in subsequent stages.

## Value

A numeric contribution vector, where each element represents the payment of the different agents.

## References

Bernárdez Ferradás, A., Mirás Calvo, M. Á., Quinteiro Sandomingo, C., and Sánchez-Rodríguez, E. (2025). Airport problems with cloned agents. [Preprint manuscript].

Faigle, U. and Kern, W. (1992). The Shapley value for cooperation games under precedence constraints. *International Journal of Game Theory*, 21, 249-266.

Fiestras-Janeiro, M. G., Sánchez-Rodríguez, E., and Schuster, M. (2016). A precedence constraint value revisited. *TOP*, 24, 156-179.

Mirás Calvo, M. Á., Quinteiro Sandomingo, C., and Sánchez-Rodríguez, E. (2020). The boundary of the core of a balanced game: faces games. *International Journal of Game Theory*, 49(2), 579-599.

## See Also

NSfaces, PRIORrule, basicrule

## Examples

```
# Two stages
c <- c(1, 3, 7, 10) # Cost vector
P <- list(c(1, 2), c(3, 4)) # Agents' hierarchical order
hierarchicalrule(c, P, rule = "SEC") # SEC rule

# Three stages
c <- c(1, 1, 3, 3, 7, 10) # Cost vector
P <- list(2, c(1, 3), c(4, 5, 6)) # Agents' hierarchical order
hierarchicalrule(c, P, "CEC") # CEC rule
```

`multibasicrules` *Overview of the allocation rules*

## Description

`multibasicrules` calculates the contribution vectors resulting from the allocation of payments among different agents, applying various predefined rules. It also generates a graphical representation of the allocations based on the implemented rules.

## Usage

```
multibasicrules(
  c,
  rules = c("SFC", "SEC", "CEC", "CEB", "CP", "SM", "CC", "SIGMA", "PRIOR"),
  draw = TRUE,
  col = NULL,
  a = NULL,
  order = NULL,
  agents_names = NULL,
  labels = TRUE
)
```

## Arguments

| | |
|---|---|
| `c` | A numeric cost vector. |
| `rules` | A character vector specifying the allocation rules. The available rules are: `"SFC"`, `"SEC"`, `"CEC"`, `"CP"`, `"CEB"`, `"SM"`, `"CC"`, `"SIGMA"` and `"PRIOR"`. By default, all the rules are selected. |
| `draw` | A logical value indicating whether or not the plot should be generated. By default, `draw = TRUE`. |
| `col` | A vector that indicates the colors used to represent each agent in the graphical representation. By default, the colors are selected by the function `rainbow()`. |
| `a` | A numeric value in the range [0,1], controlling the parameterization of the rule. It can only be defined when `"SIGMA"` is included in `rules`. By default, `a = 0.5`. |
| `order` | A numeric vector indicating the priority order of agents when making contributions. It can only be defined when `"PRIOR"` is included in `rules`. By default, agents follow their original indexing and contribute accordingly. |
| `agents_names` | A vector defining the name assigned to each agent. By default, the names follow a sequence of natural numbers, starting from 1. |
| `labels` | A logical value indicating whether the labels and the title of the plot should be displayed. By default, `labels = TRUE`. |

## Details

Let $X(c)$ be the set of cost allocations for $c \in C^N$. A rule is a mapping $\mathcal{R} : C^N \to \mathbb{R}^N$ that associates with each problem a contribution vector $\mathcal{R}_P(c)$ such that $0 \leq \mathcal{R}_P(c) \leq c$. In other words, a rule is a mechanism that selects an allocation vector for each airport problem.

The various proposed rules, despite their differences, share a key characteristic: for any given problem, each rule selects an allocation vector that belongs to its no-subsidy set. Although these rules have been individually characterized in different functions, the one in question allows for the calculation of all of them at once.

## Value

A data frame containing the contribution vectors determined by the selected allocation rules. Additionally, if `draw = TRUE`, a mosaic plot displaying the allocations obtained for the different rules.

## Note

When `"CC"` is included in the `rules` argument, the execution time of the function may significantly increase if the number of agents exceeds 150.

## References

Bernárdez Ferradás, A., Mirás Calvo, M. Á., Quinteiro Sandomingo, C., and Sánchez-Rodríguez, E. (2025). Airport problems with cloned agents. [Preprint manuscript].

Potters, J. and Sudhölter, P. (1999). Airport problems and consistent allocation rules. *Mathematical Social Sciences*, 38, 83–102.

Thomson, W. (2024). Cost allocation and airport problems. *Mathematical Social Sciences*, 31(C), 17–31.

## See Also

`basicrule`, `comparisonallocations`, `plotallocations`

## Examples

```
# All rules with graphical representation
c <- c(1, 3, 7, 10) # Cost vector
multibasicrules(c)

# SEC, CEC and SIGMA rule without plot
c <- c(5, 10, 20) # Cost vector
multibasicrules(c, rules = c("SEC", "CEC", "SIGMA"), draw = FALSE, a = 0.75,
agents_names = c("Alex", "Estela", "Carmen"))
```

`multiclonesrules` *Overview of the allocation rules with cloned agents*

## Description

`multiclonesrules` calculates the contribution vectors resulting from the allocation of payments among different agents, applying the versions for clones of various predefined rules. It also generates a graphical representation of the allocations based on the implemented rules.

## Usage

```
multiclonesrules(
  cw,
  eta,
  rules = c("SFC", "SEC", "CEC", "CEB", "CP", "SM", "CC", "SIGMA", "PRIOR"),
  group_contribution = TRUE,
  draw = TRUE,
  col = NULL,
  a = NULL,
  order = NULL,
  agents_names = NULL,
  labels = TRUE
)
```

## Arguments

| | |
|---|---|
| `cw` | A numeric cost vector, with the same length as `eta`. |
| `eta` | A numeric vector representing the size of each group of cloned agents. All its elements must be positive integers. |
| `rules` | A character vector specifying the allocation rules. The available rules are: `"SFC"`, `"SEC"`, `"CEC"`, `"CP"`, `"CEB"`, `"SM"`, `"CC"`, `"SIGMA"` and `"PRIOR"`. By default, all the rules are selected. |
| `group_contribution` | A logical value. By default, if `group_contribution = TRUE`, the cost allocation vector stores the aggregated contribution for each group of clones. However, if `group_contribution = FALSE`, the cost allocation vector represents the individual contribution of one of the agents in the group of clones. |
| `draw` | A logical value indicating whether the plot should be generated. By default, `draw = TRUE`. |
| `col` | A vector that indicates the colors used to represent each agent in the graphical representation. By default, the colors are selected by the function `rainbow()`. |
| `a` | A numeric value in the range [0,1], controlling the parameterization of the rule. It can only be defined when `"SIGMA"` is included in `rules`. By default, `a = 0.5`. |
| `order` | A numeric vector indicating the priority order of agents when making contributions. It can only be defined when `"PRIOR"` is included in `rules`. By default, agents follow their original indexing and contribute accordingly. |

`agents_names` A vector defining the name assigned to each group of clones. By default, the names follow a sequence of natural numbers, starting from 1.

`labels` A logical value indicating whether the labels and the title of the plot should be displayed. By default, `labels = TRUE`.

## Details

Let $\mathcal{R}$ be a rule, $t \in N$, and $(\eta, c) \in \mathcal{A}_t^N$. For each $i \in T = \{1, \ldots, t\}$, the sum of the contributions requested by $\mathcal{R}$ from the group of clones $N_i^\eta$ is $\mathcal{R}(\eta * c, N_i^\eta) = \sum_{j \in N_i^\eta} \mathcal{R}_j(n * c)$.

The computation of the cost allocation selected by a rule for a given problem can be substantially simplified where there are cloned agents. Through this function, a direct method is proposed to obtain either the aggregate contribution or the individual contribution of each group of cloned agents, based on the associated reduced problem and the number of clones in each group.

The version for clones of the SFC, SEC, CEC and CP rules is equal to their respective weighted version, so the formulation of these rules will be the same for the weighted version. Only the CEB rule has a weighted version and a clone version that are different.

If a rule $\mathcal{R}$ satisfies equal treatment of equals (cloned agents pay equal amounts), then $\mathcal{R}(\eta * c, N_i^\eta) = \eta_i \mathcal{R}_s(n * c)$ for any $s \in N_i^\eta$. All the rules listed, with the exception of the PRIOR rule, satisfy equal treatment of equals. Therefore, the contribution demanded by these rules from a group of clones is divided equally among them.

Finally, we define a $k$-replica of an airport problem as the problem in which every agent is replaced by $k$ clones of itself. If $k$ increases, the number of groups of cloned agents does not change, but the number of agents is large. Thus, we say that a rule satisfies replication invariance if for each $c \in C^N$, each $i \in N$, and each $k \in \mathbb{N}$, we have $\mathcal{R}(k * c, N_i^k) = \mathcal{R}_i(c)$; i.e., if in any $k$-replica of a problem each group of cloned agents contributes an amount independent of $k$. The SFC, SEC, CEC, CP, and PRIOR rules verify this property, while the others do not.

## Value

A data frame containing the contribution vectors determined by the selected allocation rules. Additionally, if `draw = TRUE`, a mosaic plot displaying the allocations obtained for the different rules.

## Note

When `"CC"` is included in the `rules` argument, the execution time of the function may significantly increase if the number of individual agents exceeds 150.

## References

Bernárdez Ferradás, A., Mirás Calvo, M. Á., Quinteiro Sandomingo, C., and Sánchez-Rodríguez, E. (2025). Airport problems with cloned agents. [Preprint manuscript].

Littlechild, S. C. and Thompson, G. F. (1977). Aircraft landing fees: a game theory approach. *The Bell Journal of Economics*, 8, 186-204.

## See Also

`clonesgroups`, `clonesproblem`, `clonesrule`, `comparisonallocations`, `plotallocations`

## Examples

```
# All rules for clones with graphical representation
cw <- c(1, 3, 7, 10) # Different costs
eta <- c(3, 4, 1, 2)  # Size of each groups of clones
multiclonesrules(cw, eta)

# SEC, CEC and CP rule for clones without plot
cw <- c(5, 10, 20) # Different costs
eta <- c(8, 2, 4) # Size of each groups of clones
multiclonesrules(cw, eta, rules = c("SEC", "CEC", "CP"),
group_contribution = FALSE, draw = FALSE,
agents_names = c("Suppliers", "Wholesalers", "Retailers"))
```

---

# multihierarchicalrules

*Overview of the allocation rules according to the agents' hierarchical order*

---

## Description

`multihierarchicalrules` calculates the contribution vectors resulting from the allocation of payments among different agents, applying various predefined rules in relation to the agents' hierarchical order. It also generates a graphical representation of the allocations based on the implemented rules.

## Usage

```
multihierarchicalrules(
  c,
  P,
  rules = c("SFC", "SEC", "CEC", "CEB", "CP", "SM", "CC", "SIGMA"),
  a = NULL,
  draw = TRUE,
  col = NULL,
  agents_names = NULL,
  labels = TRUE
)
```

## Arguments

| | |
|---|---|
| `c` | A numeric cost vector. |
| `P` | A list showing the agents involved in the different distribution stages. |
| `rules` | A character vector specifying the allocation rules. The available rules are: `"SFC"`, `"SEC"`, `"CEC"`, `"CP"`, `"CEB"`, `"SM"`, `"CC"` and `"SIGMA"`. By default, all the rules are selected. |

| | |
|---|---|
| a | A numeric value in the range [0,1], controlling the parameterization of the rule. It can only be defined when "SIGMA" is included in rules. By default, a = 0.5. |
| draw | A logical value indicating whether the plot should be generated. By default, draw = TRUE. |
| col | A vector that indicates the colors used to represent each agent in the graphical representation. By default, the colors are selected by the function rainbow(). |
| agents_names | A vector defining the name assigned to each agent. By default, the names follow a sequence of natural numbers, starting from 1. |
| labels | A logical value indicating whether the labels and the title of the plot should be displayed. By default, labels = TRUE. |

## Details

Let $N = \{1, \ldots, n\}$ be a finite set of agents and let $P = \{P_1, \ldots, P_{m+1}\}$ be a partition of $N$, with $m \leq n - 1$. So, the hierarchical structure, $P_> = \{P_1 > P_2 > \cdots > P_m > P_{m+1}\}$, implies that agents in $P_1$ have priority over agents in $N \backslash P_1$, agents in $P_2$ have priority over agents in $N \backslash P_1 \cup P_2$, and so on. Let $\mathcal{P}(N)$ denote the family of all hierarchical structures over $N$.

A hierarchical rule is a mapping $\mathcal{R}_P : C^N \to \mathbb{R}^N$ that, based on a hierarchical structure $P_>$, associates with each problem $c \in C^N$ a cost allocation vector $\mathcal{R}_P(c)$ such that $0 \leq \mathcal{R}_P(c) \leq c$. In other words, the allocation proceeds by assigning costs to agents in higher-hierarchy coalitions before those in lower-hierarchy ones, through the successive application of the rule to the no-subsidy faces of coalitions obtained from $P_>$. These rules are on the boundary of the no-subsidy set.

In each stage, the agents share the accumulated costs up to that point. However, the remaining costs are allocated in subsequent stages.

## Value

A data frame containing the contribution vectors determined by the selected allocation rules. Additionally, if draw = TRUE, a mosaic plot displaying the allocations obtained for the different rules.

## References

Bernárdez Ferradás, A., Mirás Calvo, M. Á., Quinteiro Sandomingo, C., and Sánchez-Rodríguez, E. (2025). Airport problems with cloned agents. [Preprint manuscript].

Faigle, U. and Kern, W. (1992). The Shapley value for cooperation games under precedence constraints. *International Journal of Game Theory*, 21, 249-266.

Fiestras-Janeiro, M. G., Sánchez-Rodríguez, E., and Schuster, M. (2016). A precedence constraint value revisited. *TOP*, 24, 156-179.

Mirás Calvo, M. Á., Quinteiro Sandomingo, C., and Sánchez-Rodríguez, E. (2020). The boundary of the core of a balanced game: faces games. *International Journal of Game Theory*, 49(2), 579-599.

## See Also

NSfaces, PRIORrule, basicrule, comparisonallocations, plotallocations

## Examples

```
# All rules in two stages with graphical representation
c <- c(1, 3, 3, 7, 10) # Cost vector
P <- list(c(1, 2, 3), c(4, 5)) # Agents' hierarchical order
multihierarchicalrules(c, P)

# SEC, CEC and SM rule in three stages without plot
c <- c(5, 10, 20, 20, 30, 50) # Cost vector
P <- list(c(1, 2), c(3, 4), c(5, 6)) # Agents' hierarchical order
multihierarchicalrules(c, P, rules = c("SEC", "CEC", "SM"), draw = FALSE,
agents_names = c("Alex", "Estela", "Carmen", "Miguel", "Gloria", "Brais"))
```

---

`multiweightedrules` *Overview of the weighted allocation rules*

---

## Description

`multiweightedrules` calculates the contribution vectors resulting from the allocation of payments among different agents, applying various predefined weighted rules. It also generates a graphical representation of the allocations based on the implemented weighted rules.

## Usage

```
multiweightedrules(
  c,
  w,
  rules = c("SFC", "SEC", "CSEC", "CEC", "CEB", "CP"),
  draw = TRUE,
  col = NULL,
  agents_names = NULL,
  labels = TRUE
)
```

## Arguments

| | |
|---|---|
| `c` | A numeric cost vector. |
| `w` | A numeric weight vector. |
| `rules` | A character vector specifying the allocation rules. The available rules are: `"SFC"`, `"SEC"`, `"CSEC"`, `"CEC"`, `"CP"` and `"CEB"`. By default, all the rules are selected. |
| `draw` | A logical value indicating whether the plot should be generated. By default, `draw = TRUE`. |
| `col` | A vector that indicates the colors used to represent each agent in the graphical representation. By default, the colors are selected by the function `rainbow()`. |
| `agents_names` | A vector defining the name assigned to each agent. By default, the names follow a sequence of natural numbers, starting from 1. |
| `labels` | A logical value indicating whether the labels and the title of the plot should be displayed. By default, `labels = TRUE`. |

## Details

Let $w = (w_i)_{i \in N} \in \mathbb{R}^n$ be a positive weight vector, satisfying $w_i > 0$ for all $i \in N$ and $w(N) = 1$. Consider the $(n-1)$-standard simplex, defined as $\Delta_n = \{x \in \mathbb{R}^n : x \geq 0,\ x_1 + \cdots + x_n = 1\}$. Then, the set of all positive weight vectors corresponds to $\text{Int}(\Delta_n)$, the interior of the $(n-1)$-standard simplex.

A weighted rule is a mapping $\mathcal{R} : C^N \times \text{Int}(\Delta_N) \to \mathbb{R}^N$ which associates with a problem $c \in C^N$ and a positive weight vector $w \in \text{Int}(\Delta_n)$ a contribution vector $\mathcal{R}(c, w) \in X(c)$.

It is possible to define weighted versions of the rules: SFC, SEC, CEC, CP and CEB. In fact, two different rules emerge from the standard SEC rule: the weighted SEC rule and the coalition-weighted SEC rule. If $w_i = w_j$ for all $i, j \in N$, then the solution of weighted SEC$(c)$ and weighted CSEC$(c)$ coincides.

In all the rules, the higher the weight $w_i$, the more the corresponding agent will have to pay, except for the weighted CEB rule (the construction of this rule is based on the concept of allocating 'benefits', so it is logical that it is set up this way). Furthermore, as previously stated, all the rules, except for the weighted CSEC rule, require the weights to be positive. However, to standardize the criterion, we establish that the CSEC rule also demands positive weights.

The weighted version of the SFC, SEC, CEC and CP rules is equal to their respective versions for clones, so the formulation of these rules will be the same for the version with clones. Only the CEB rule has a weighted version and a clone version that are different.

## Value

A data frame containing the contribution vectors determined by the selected allocation rules. Additionally, if `draw = TRUE`, a mosaic plot displaying the allocations obtained for the different weighted rules.

## References

Bernárdez Ferradás, A., Mirás Calvo, M. Á., Quinteiro Sandomingo, C., and Sánchez-Rodríguez, E. (2025a). Airport problems with cloned agents. [Preprint manuscript].

Bernárdez Ferradás, A., Mirás Calvo, M. Á., Quinteiro Sandomingo, C., and Sánchez-Rodríguez, E. (2025b). A characterization of the CSEC rule for airport problems. [Prepint manuscript].

Sánchez-Rodríguez, E., Mirás Calvo, M. Á., Quinteiro Sandomingo, C., and Núñez Lugilde, I. (2024). Coalition-weighted Shapley values. *International Journal of Game Theory*, 53, 547-577.

Thomson, W. (2024). Cost allocation and airport problems. *Mathematical Social Sciences*, 31(C), 17–31.

## See Also

`weightedrule`, `comparisonallocations`, `plotallocations`

## Examples

```
# All weighted rules with graphical representation
c <- c(1, 3, 3, 7, 10) # Cost vector
w <- c(1, 4, 1, 2, 8) # Weight vector
multiweightedrules(c, w)
```

```
# Weighted SEC, CEC and CP rule without plot
c <- c(5, 10, 20)
w <- c(3, 2, 1)
multiweightedrules(c, w, rules = c("SEC", "CEC", "CP"), draw = FALSE,
agents_names = c("Alex", "Estela", "Carmen"))
```

---

NScheck *Verification of compliance with the no-subsidy constraints*

---

## Description

NScheck evaluates whether or not the no-subsidy constraint is satisfied and, if not, it can also determine one of the coalitions of agents that violates it, as long as the user requests it.

## Usage

```
NScheck(
  c,
  x,
  eta = rep(1, length(x)),
  group_contribution = TRUE,
  coalition = FALSE,
  tol = 1e-06
)
```

## Arguments

| | |
|---|---|
| c | A numeric cost vector. |
| x | A numeric cost allocation vector. |
| eta | A numeric vector representing the size of each group of cloned agents. All its elements must be positive integers. By default, all components of eta are set to 1. |
| group_contribution | A logical value. By default, if group_contribution = TRUE, x stores the aggregated contribution for each group of clones. However, if group_contribution = FALSE, x represents the individual contribution of one of the agents in the group of clones. |
| coalition | A logical value. By default, if coalition = FALSE, the function only returns whether the NS constraint is satisfied or not. However, if coalition = TRUE, the function also returns the coalition of agents that breach the NS constraint, if it is violated. |
| tol | Tolerance level for evaluating compliance with the NS constraint. |

## Details

For each $c \in C^N$ let $H(c) = \{x \in \mathbb{R} : x(N) = c_n\}$ be the hyperplane of $\mathbb{R}^N$ given by all the vectors whose coordinates add up to $c_n$. A cost allocation for $c \in C^N$ is a vector $x \in H(c)$ such that $0 \leq x \leq c$. The component $x_i$ is the contribution requested from agent $i$. Let $X(c)$ be the set of cost allocations for $c \in C^N$. Given $x \in X(c)$, the difference $c_i - x_i$ is the benefit of agent $i$ at $x$.

A basic requirement is that at an allocation $x \in X(c)$ on group $N' \subset N$ of agents would subsidize the other agents by contributing more than what the group would have to pay on its own. The no-subsidy constraint for the group $N' \subset N$ is $x(N') \geq \max\{c_j : j \in N'\}$. The set of cost allocations for $c \in C^N$ that satisfy the no-subsidy constraints, the no-subsidy set for short, is given by:

$$NS(c) = \{x \in X(c) : x(N') \leq \max\{c_j : j \in N'\}, \text{ for all } N' \subset N\}$$

$$= \{x \in \mathbb{R}^N : x \geq 0,\ x(N) = c_n,\ x_1 + \cdots + x_i \leq c_i, \text{ for all } i \in N \backslash \{n\}\}$$

Thus, the no-subsidy correspondence NS assigns to each $c \in C^N$ the set $NS(c)$.

Nevertheless, when a problem has group of cloned agents, the structure of its no-subsidy set is simpler than when all the cost parameters are different. Let $t \in N$, $\mathcal{A}_t^N$ be the set of pairs $(\eta, c) \in \mathbb{N}^t \times \mathbb{R}^t$ and $N_s^\eta = N_s^{\eta * c} = \{j \in N : (n * c)_j = c_s\}$. Then the no subsidy set for $\eta * c \in C^N$ is:

$$NS(\eta * c) = \{x \in \mathbb{R} : x \geq 0,\ x(N) = c_t,\ x(N_1^\eta) + \cdots + x(N_s^\eta) \leq c_s, \text{ for all } s < t\}.$$

It is worth noting that all allocation rules proposed in this package satisfy this property.

## Value

If `coalition = TRUE`, a logical value (TRUE or FALSE) indicating compliance with the NS constraint.

Otherwise, if `coalition = FALSE`, a list containing the following items:

| | |
|---|---|
| `flag` | A logical value (TRUE or FALSE) indicating compliance with the NS constraint. |
| `eta` | If the NS constraint is violated, the coalition of agents that breach it will be returned. |

## References

Bernárdez Ferradás, A., Mirás Calvo, M. Á., Quinteiro Sandomingo, C., and Sánchez-Rodríguez, E. (2025). Airport problems with cloned agents. [Preprint manuscript].

Thomson, W. (2024). Cost allocation and airport problems. *Mathematical Social Sciences*, 31(C), 17–31.

## See Also

`NSfaces`, `NSstructure`, `NSset`, `comparisonallocations`

## Examples

```
# Compliance with the NS constraint
c <- c(2, 5, 9)
x <- SECrule(c)
NScheck(c, x)
# Non-compliance with the NS constraint
c <- c(2, 3, 7, 10)
x <- c(1, 2, 5, 2)
NScheck(c, x, coalition = TRUE)
```

---

NSfaces *Face games associated with an airport problem*

---

## Description

NSfaces determines the airport problems defining the two components of the decomposition of one specific face game.

## Usage

```
NSfaces(c, R)
```

## Arguments

| | |
|---|---|
| c | A numeric cost vector. |
| R | A numeric vector representing the agents forming the coalition |

## Details

Let $c \in C^N$ be an airport problem and $v \in G^N$ its associated cost game, for each non-empty proper coalition $R \in 2^N \backslash \{\emptyset, N\}$ define the $N\backslash R$-face of Core$(v)$ as the set

$$F_{N\backslash R}(c) = \text{Core}(v) \cap \{x \in \mathbb{R}^N : x(R) = v(R)\}.$$

Also, the $N\backslash R$-face game $v_{F_{N\backslash R}} \in G^N$ is given by $v_{F_{N\backslash R}}(S) = v(S \cup R) - v(R) + v(S \cap R),\ S \in 2^N$. It turns out that $F_{N\backslash R}(c) = \text{Core}(v_{F_{N\backslash R}})$, the $N\backslash R$-face of the core of the associated cost game is the core of the $N\backslash R$-face game. Let $r \in N$ such that $c_r = v(R) = \max\{c_i : i \in R\}$ and denote $R_+ = \{k \in N : k > r\}$. Consider the airport problems

$$c_{|R} = (0_{N\backslash R}, c_R) \in C^N \text{ and } c_{|R_+} = (0_{N\backslash R_+}, c_{r+1} - c_r, \ldots, c_n - c_r) \in C^N$$

with associated games $v^{|R} \in G^N$ and $v^{|R_+} \in G^N$, respectively. It is easy to see that the $N\backslash R$-face game $v_{F_{N\backslash R}} \in G^N$ is decomposable with respect to the partition $R, N\backslash R$ and its components are $v^{|R}$ and $v^{|R_+}$. Moreover,

$$F_{N\backslash R}(c) = NS(c_R) \times NS(c_{r+1} - c_r, \ldots, c_n - c_r) \times 0_{N\backslash(R\cup R_+)}.$$

Therefore, the components of the face games of the associated cost game are associated cost games themselves. In the $N\backslash R$-face game, the players of $R$ play the game associated with the problem where the agents of $R$ keep their cost parameters while the cost parameter of the agents in $N\backslash R$ is null. On the other hand, since the players of $R$ already share $c_r = v(R)$ among themselves, the players of $N\backslash R$ with an initial cost lower than $c_r$ now have their cost parameter equal to zero while the others see their cost reduced by $c_r$.

## Value

A numeric matrix with two rows representing the decomposition of the $R$-face game:

| | |
|---|---|
| `[1, ]` | The first row is obtained by setting the cost of a specific coalition to zero while retaining the cost parameters of the complementary coalition. |
| `[2, ]` | The second row is derived by subtracting the highest cost in the complementary coalition from each agent's cost, or setting it to zero if the result is negative. |

## References

Bernárdez Ferradás, A., Mirás Calvo, M. Á., Quinteiro Sandomingo, C., and Sánchez-Rodríguez, E. (2025). Airport problems with cloned agents. [Preprint manuscript].

González-Díaz, J. and Sánchez-Rodríguez, E. (2008). Cores of convex and strictly convex games. *Games and Economic Behavior*, 62, 100-105.

Mirás Calvo, M. Á., Quinteiro Sandomingo, C., and Sánchez-Rodríguez, E. (2020). The boundary of the core of a balanced game: faces games. *International Journal of Game Theory*, 49(2), 579-599.

## See Also

`NScheck`, `NSstructure`, `NSset`, `hierarchicalrule`

## Examples

```
c <- c(1, 3, 7, 10) # Cost vector
R <- c(3, 4) # Coalition of agents
NSfaces(c, R) # Components of the face game
```

---

`NSset` *Vertices and visualization of the NS set*

---

## Description

`NSset` calculates the coordinates of the vertices that make up the NS set. It also enables the generation of a graphical representation of the no-subsidy set in 1D, 2D, and 3D (available only when there are 2, 3, or 4 agents).

## Usage

```
NSset(
  c,
  draw = FALSE,
  dimension = NULL,
  representation = "projection",
  col = NULL,
  agents_names = NULL,
  labels = TRUE
)
```

## Arguments

| | |
|---|---|
| c | A numeric cost vector. |
| draw | A logical value indicating whether the plot should be generated. By default, draw = FALSE. |
| dimension | A character string that specifies the dimension of the graphic. Possible values are "1D", "2D", and "3D". By default, the dimension is chosen based on the number of agents: "1D" for 2 agents, "2D" for 3 agents, and "3D" for 4 agents. |
| representation | A character string indicating which NS set is displayed. Possible values are "real", "projection", and "both". By default, representation = "projection". |
| col | A character string reflecting the color tone of the NS set. By default, the color tone "dodgerblue" is used. |
| agents_names | A vector defining the name assigned to each agent. By default, the names follow a sequence of natural numbers, starting from 1. |
| labels | A logical value indicating whether the coordinates of the points and the plot title should be displayed. By default, labels = TRUE. |

## Details

For each $c \in C^N$ let $H(c) = \{x \in \mathbb{R} : x(N) = c_n\}$ be the hyperplane of $\mathbb{R}^N$ given by all the vectors whose coordinates add up to $c_n$. A cost allocation for $c \in C^N$ is a vector $x \in H(c)$ such that $0 \leq x \leq c$. The component $x_i$ is the contribution requested from agent $i$. Let $X(c)$ be the set of cost allocations for $c \in C^N$. Given $x \in X(c)$, the difference $c_i - x_I$ is the benefit of agent $i$ at $x$.

A basic requirement is that at an allocation $x \in X(c)$ on group $N' \subset N$ of agents would subsidize the other agents by contributing more than what the group would have to pay on its own. The no-subsidy constraint for the group $N' \subset N$ is $x(N') \geq \max\{c_j : j \in N'\}$. The set of cost allocations for $c \in C^N$ that satisfy the no-subsidy constraints, the no-subsidy set for short, is given by:

$$NS(c) = \{x \in X(c) : x(N') \leq \max\{c_j : j \in N'\}, \text{ for all } N' \subset N\}$$

$$= \{x \in \mathbb{R}^N : x \geq 0,\ x(N) = c_n,\ x_1 + \cdots + x_i \leq c_i, \text{ for all } i \in N \backslash \{n\}\}$$

Thus, the no-subsidy correspondence NS assigns to each $c \in C^N$ the set $NS(c)$.

Nevertheless, when a problem has group of cloned agents, the structure of its no-subsidy set is simpler than when all the cost parameters are different. Let $t \in N$, $\mathcal{A}_t^N$ be the set of pairs $(\eta, c) \in \mathbb{N}^t \times \mathbb{R}^t$ and $N_s^\eta = N_s^{\eta * c} = \{j \in N : (n * c)_j = c_s\}$. Then the no subsidy set for $\eta * c \in C^N$ is:

$$NS(\eta * c) = \{x \in \mathbb{R} : x \geq 0,\ x(N) = c_t,\ x(N_1^\eta) + \cdots + x(N_s^\eta) \leq c_s,\ \text{for all } s < t\}.$$

For any cost vector $c$, if there are $n$ agents with different cost parameters, the number of faces is $2n-2$. However, the number of full-dimensional faces is indeed affected by the presence of clones. Let $t \in N$, $(\eta, c) \in \mathcal{A}_t^N$, and $\eta * c \in C^N$, NS$(\eta * c)$ has $n + t - 2$ full-dimensional faces if $\eta_t = 1$ and $n + t - 1$ full-dimensional faces otherwise. On the other hand, the number of different extreme points of the set NS$(\eta * c)$ is: $\eta_t \prod_{i \in T \setminus \{t\}} (\eta_i + 1)$ (so, when there are no clones, the NS$(c)$ has $2^{n-1}$ extreme points).

## Value

A numeric matrix containing the vertices that determine the NS set. Additionally, if `draw = TRUE` and the number of agents is 2, 3, or 4, a plot displaying the faces and extreme points of the NS set will be generated.

## References

Bernárdez Ferradás, A., Mirás Calvo, M. Á., Quinteiro Sandomingo, C., and Sánchez-Rodríguez, E. (2025). Airport problems with cloned agents. [Preprint manuscript].

González-Díaz, J., Mirás Calvo, M. Á., Quinteiro Sandomingo, C., and Sánchez-Rodríguez, E. (2016). Airport games: the core and its center. *Mathematical Social Sciences*, 82, 105–115.

Mirás Calvo, M. Á., Quinteiro Sandomingo, C., and Sánchez-Rodríguez, E. (2020). The boundary of the core of a balanced game: faces games. *International Journal of Game Theory*, 49(2), 579-599.

## See Also

`plotallocations`, `NScheck`, `NSfaces`, `NSstructure`

## Examples

```
# Projected NS set for 3 agents
c <- c(5, 10, 20) # Cost vector
NSset(c, draw = TRUE)

# Real and projected NS set for 3 agents
c <- c(1, 2, 3) # Cost vector
NSset(c, TRUE, "3D", "both")

# Projected NS set for 4 agents
c <- c(3, 3, 3, 10) # Cost vector
NSset(c, TRUE, "3D", "projection", "aquamarine",
c("Alex", "Estela", "Carmen", "Miguel"))
```

`NSstructure` *Composition of the no-subsidy set*

## Description

`NSstructure` quantifies the key elements of the structure of the no-subsidy set.

## Usage

```
NSstructure(cw, eta = rep(1, length(cw)))
```

## Arguments

| | |
|---|---|
| `cw` | A numeric cost vector, with the same length as `eta`. |
| `eta` | A numeric vector representing the size of each group of cloned agents. All its elements must be positive integers. By default, `eta = rep(1, length(cw))`, i.e., all groups have size 1. |

## Details

For any cost vector $c$, if there are $n$ agents with different cost parameters, the number of faces of the NS set is $2n-2$. However, the number of full-dimensional faces is indeed affected by the presence of clones. Let $t \in N$, $(\eta, c) \in \mathcal{A}_t^N$, and $\eta * c \in C^N$, NS$(\eta * c)$ has $n+t-2$ full-dimensional faces if $\eta_t = 1$ and $n+t-1$ full-dimensional faces otherwise. On the other hand, the number of different extreme points of the set NS$(\eta * c)$ is: $\eta_t \prod_{i \in T \setminus \{t\}} (\eta_i + 1)$ (so, when there are no clones, the NS$(c)$ has $2^{n-1}$ extreme points).

Let $k \in \mathbb{N}$ and denote by $\lambda_k$ the $k$-dimensional Lebesgue measure. If $X = (X_1, \ldots, X_k)$ is a random vector with joint density function $f$ and $\Omega$ is a Borel set, then $P(X \in \Omega) = \int_\Omega f(x) d\lambda_k$ and the expected value of $X$ is $\mathbb{E}[X] = \int_{\mathbb{R}^k} x f(x) d\lambda_k$. Given a Borel set $\Omega \subset \mathbb{R}^k$ of positive measure, $\lambda_k(\Omega) > 0$, we say that a random vector $U = (U_1, \ldots, U_k)$ has a uniform distribution on $\Omega$, and we write $U \sim U(\Omega)$, if $U$ has a probability density function $f(x_1, \ldots, x_k) = \frac{1}{\lambda_k(\Omega)}$ if $(x_1, \ldots, x_k) \in \Omega$ and $f(x_1, \ldots, x_k) = 0$ otherwise. If $a = (a_1, \ldots, a_k) \in \mathbb{R}^k$ with $0 < a_1 \leq \cdots \leq a_k$, denote

$$V_k(a) = \int_0^{a_1} \ldots \int_0^{a_k - \sum_{j=1}^{k-1} x_j} dx_k \ldots dx_1.$$

Therefore, for each $c \in C^N$, the value $V_{n-1}(c_{-n})$ is the $(n-1)$-Lebesgue measure of $NS_n(c)$, so $\lambda_{n-1}(\text{NS}(c)) = \sqrt{n}\lambda_{n-1}(\text{NS}_n(c)) = \sqrt{n}V_{n-1}(c_{-n})$.

## Value

A list containing the following items:

| | |
|---|---|
| `n.faces` | A positive integer representing the number of faces that form the NS set. |
| `n.full.dim.faces` | A positive integer indicating the number of full-dimensional faces forming the NS set. |

`n.extreme.points`
: A positive integer counting the number of extreme points of the NS set.

`actual.volume`
: A positive number representing the volume of the NS set.

`projected.volume`
: A positive number reflecting the projected volume of the NS set.

## References

Bernárdez Ferradás, A., Mirás Calvo, M. Á., Quinteiro Sandomingo, C., and Sánchez-Rodríguez, E. (2025). Airport problems with cloned agents. [Preprint manuscript].

González-Díaz, J., Mirás Calvo, M. Á., Quinteiro Sandomingo, C., and Sánchez-Rodríguez, E. (2016). Airport games: the core and its center. *Mathematical Social Sciences*, 82, 105–115.

## See Also

`NScheck`, `NSfaces`, `NSset`, `CCrule`

## Examples

```
# Without cloned agents
c <- c(1, 2, 3, 4)
NSstructure(c)

# With cloned agents
c <- c(1, 2)
eta <- c(3, 1)
NSstructure(c, eta)
```

---

# `Peinador` *Peinador airport data*

---

## Description

The data includes several variables describing 833 aircraft operations — 418 take-offs and 415 landings — involving 27 different aircraft types at Peinador Airport (Vigo, Spain) during January 2025.

## Usage

```
Peinador
```

## Format

A data frame with 27 rows and 6 columns (variables):

**type** Aircraft type

**i** Aircraft index (subscript)

**TO** Number of takeoffs

**LDG** Number of landings

**MTOW** Maximum takeoff weight (in tons)

**MLW** Maximum landing weight (in tons)

## Source

AENA. (2025, April 23). *Estadísticas de tráfico aéreo: Consultas personalizadas*. https://www.aena.es/es/estadisticas/consultas-personalizadas.html.

## Examples

```
# Allocation by MTOW (in t)
multiclonesrules(Peinador$MTOW, Peinador$TO, c("SFC", "SEC", "CEC", "CP"),
labels = FALSE)

# Allocation by MLW (in t)
positives <- Peinador$LDG > 0
MLW <- Peinador$MLW[positives]
LDG <- Peinador$LDG[positives]
multiclonesrules(MLW, LDG, c("SFC", "SEC", "CEC", "CP"), labels = FALSE)
```

---

# plotallocations *Graphical representation of the contribution vectors within the NS set*

---

## Description

`plotallocations` generates a graphical representation of the contribution vectors inside the NS set in 1D, 2D, and 3D (available only when there are 2, 3, or 4 agents).

## Usage

```
plotallocations(
  c,
  contributions,
  dimension = NULL,
  representation = "projection",
  col = NULL,
  colors = NULL,
  agents_names = NULL,
  labels = TRUE,
  contributions_names = NULL,
  tol = 1e-06
)
```

## Arguments

| | |
|---|---|
| `c` | A numeric cost vector. |
| `contributions` | A list containing different cost allocation vectors. |
| `dimension` | A character string that specifies the dimension of the graphic. Possible values are `"1D"`, `"2D"`, and `"3D"`. By default, the dimension is chosen based on the number of agents: `"1D"` for 2 agents, `"2D"` for 3 agents, and `"3D"` for 4 agents. |
| `representation` | A character string indicating which NS set and allocations are displayed. Possible values are `"real"`, `"projection"`, and `"both"`. By default, `representation = "projection"`. |
| `col` | A character string reflecting the color tone of the NS set. By default, the color tone `"dodgerblue"` is used. |
| `colors` | A vector that indicates the colors used to represent each contribution vector. By default, a color palette of different shades is used. |
| `agents_names` | A vector defining the name assigned to each agent. By default, the names follow a sequence of natural numbers, starting from 1. |
| `labels` | A logical value indicating whether the coordinates of the points and the plot title should be displayed. By default, `labels = TRUE`. |
| `contributions_names` | A vector defining the name assigned to each cost allocation vector. By default, and whenever `labels = TRUE`, the Cartesian coordinates of each point are displayed. |
| `tol` | Tolerance level for evaluating compliance with the NS constraint. |

## Details

For each $c \in C^N$ let $H(c) = \{x \in \mathbb{R} : x(N) = c_n\}$ be the hyperplane of $\mathbb{R}^N$ given by all the vectors whose coordinates add up to $c_n$. A cost allocation for $c \in C^N$ is a vector $x \in H(c)$ such that $0 \leq x \leq c$. The component $x_i$ is the contribution requested from agent $i$. Let $X(c)$ be the set of cost allocations for $c \in C^N$.

A basic requirement is that at an allocation $x \in X(c)$ on group $N' \subset N$ of agents would subsidize the other agents by contributing more than what the group would have to pay on its own. The no-subsidy constraint for the group $N' \subset N$ is $x(N') \geq \max\{c_j : j \in N'\}$. The set of cost allocations for $c \in C^N$ that satisfy the no-subsidy constraints, the no-subsidy set for short, is given by:

$$NS(c) = \{x \in X(c) : x(N') \leq \max\{c_j : j \in N'\}, \text{ for all } N' \subset N\}$$

$$= \{x \in \mathbb{R}^N : x \geq 0,\ x(N) = c_n,\ x_1 + \cdots + x_i \leq c_i, \text{ for all } i \in N \backslash \{n\}\}$$

Thus, the no-subsidy correspondence NS assigns to each $c \in C^N$ the set $NS(c)$.

A rule is a mapping $\mathcal{R} : C^N \to \mathbb{R}^N$ which associates with each problem $c \in C^N$ a contribution vector $\mathcal{R}(c) \in X(c)$. In other words, a rule is a mechanism that, for each airport problem, selects an allocation vector belonging to its no-subsidy set.

## Value

Only if the number of agents is 2, 3, or 4 will a plot be generated displaying the NS set and all cost allocation vectors.

## References

Bernárdez Ferradás, A., Mirás Calvo, M. Á., Quinteiro Sandomingo, C., and Sánchez-Rodríguez, E. (2025). Airport problems with cloned agents. [Preprint manuscript].

González-Díaz, J., Mirás Calvo, M. Á., Quinteiro Sandomingo, C., and Sánchez-Rodríguez, E. (2016). Airport games: the core and its center. *Mathematical Social Sciences*, 82, 105–115.

Mirás Calvo, M. Á., Quinteiro Sandomingo, C., and Sánchez-Rodríguez, E. (2020). The boundary of the core of a balanced game: faces games. *International Journal of Game Theory*, 49(2), 579-599.

## See Also

NSset, NScheck, comparisonallocations

## Examples

```
# Projected SEC rule, CEC rule and SM rule for 3 agents
c <- c(5, 10, 20) # Cost vector
plotallocations(c, list(SECrule(c), CECrule(c), SMrule(c)), "2D",
"projection", contributions_names = c("SEC", "CEC", "SM"))

# Real an projected SM rule and PRIOR rule for 3 agentes
c <- c(1, 2, 3) # Cost vector
SM <- SMrule(c)
PRIOR <- PRIORrule(c, order = c(2, 3, 1))
plotallocations(c, list(SM, PRIOR), "3D", "both")

# Projected CEB rule and weighted CEB rule for 4 agents
c <- c(3, 3, 3, 10) # Cost vector
w <- c(1, 4, 8, 2) # Weight vector
CEB <- basicrule(c, "CEB")
wCEB <- weightedrule(c, w, "CEB")
plotallocations(c, list(CEB, wCEB), contributions_names = c("CEB", "wCEB"))
```

---

# PRIORrule *Priority family of rules*

---

## Description

PRIORrule calculates the contribution vector selected by a priority rule.

## Usage

```
PRIORrule(c, order = NULL)
```

## Arguments

| | |
|---|---|
| c | A numeric cost vector. |
| order | A numeric vector indicating the priority order of agents when making contributions. By default, agents follow their original indexing and contribute accordingly. |

## Details

For each $c \in C^N$ and each $i \in N$, a priority rule relative to $\pi \in \Pi^N$ is defined by

$$\mathrm{P}_i^{\pi}(c) = \max\Big\{0,\ c_i - \max\{c_j : \pi(j) < \pi(i)\}\Big\}.$$

In this rule, each agent contributes at a different step, so that each one pays all that is necessary until reaching the no-subsidy constraint. Consequently, the agents who arrive first cover the cost of all subsequent agents whose cost is lower.

## Value

A numeric contribution vector, where each element represents the payment of the different agents.

## References

Bernárdez Ferradás, A., Mirás Calvo, M. Á., Quinteiro Sandomingo, C., and Sánchez-Rodríguez, E. (2025). Airport problems with cloned agents. [Preprint manuscript].

Thomson, W. (2024). Cost allocation and airport problems. *Mathematical Social Sciences*, 31(C), 17–31.

## See Also

NSfaces, basicrule, hierarchicalrule

## Examples

```
# Ascending order
c <- c(1, 3, 7, 10) # Cost vector
PRIORrule(c)

# Fluctuating order
c <- c(1, 3, 7, 10) # Cost vector
order <- c(2, 1, 4, 3) # Priority order
PRIORrule(c, order)
```

SECrule *Sequential equal contributions rule*

## Description

SECrule calculates the contribution vector selected by the SEC rule.

## Usage

```
SECrule(c)
```

## Arguments

| | |
|---|---|
| c | A numeric cost vector. |

## Details

For each $c \in C^N$ and each $i \in N$, the sequential equal contributions rule is defined by

$$\text{SEC}_i = \frac{c_1}{n} + \frac{c_2 - c_1}{n-1} + \cdots + \frac{c_i - c_{i-1}}{n-i+1}$$

This rule is based on applying an equal division to each segment separately, so that all agents using a given segment contribute equally to its cost. Each agent's contribution is then obtained as a sum of terms, one for each of the segments they use.

The contribution selected by the SEC rule for a problem $c \in C^N$ coincides with the payoff vector assigned by the Shapley value to the associated cost game $v \in G^N$, that is, $\text{SEC}(c) = \text{Sh}(v)$.

## Value

A numeric contribution vector, where each element represents the payment of the different agents.

## References

Chun, Y., Hu, C.-C., and Yeh, C. (2012). Characterizations of the sequential equal contributions rule for the airport problem. *International Journal of Economic Theory*, 8, 77-85.

Littlechild, S.C. and Owen, G. (1973). A simple expression for the Shapley value in a special case. *Management Science*, 20, 370-372.

Thomson, W. (2024). Cost allocation and airport problems. *Mathematical Social Sciences*, 31(C), 17–31.

## See Also

basicrule, weightedrule, clonesrule, hierarchicalrule

## Examples

```
c <- c(1, 3, 7, 10) # Cost vector
SECrule(c)
```

`SFCrule` *Sequential full contributions rule*

## Description

`SFCrule` calculates the contribution vector selected by the SFC rule.

## Usage

```
SFCrule(c)
```

## Arguments

| | |
|---|---|
| `c` | A numeric cost vector. |

## Details

For each $c \in C^N$ and each $i \in N$, let $N^i_- = \{j \in N : c_j < c_i\}$ and let $N^i(c) \subseteq N$ be defined by $N^i(c) = \{j \in N : c_j = c_i\}$, the sequential full contribution rule is defined by

$$\text{SFC}_i(c) = \frac{c_i - \max\{c_j : j \in N^i_-\}}{|N_i(c)|}$$

According to this rule, an agent does not assist other agents with smaller needs than his own in covering the costs they require, even though he uses the same segments they use (similarly, he does not receive any assistance in covering his own segmental cost from agents with greater needs than his, even though these agents also use his segment). If multiple agents have the same cost parameters, they equally share the cost of the common segment they use.

## Value

A numeric contribution vector, where each element represents the payment of the different agents.

## References

Bernárdez Ferradás, A., Mirás Calvo, M. Á., Quinteiro Sandomingo, C., and Sánchez-Rodríguez, E. (2025). Airport problems with cloned agents. [Preprint manuscript].

Thomson, W. (2024). Cost allocation and airport problems. *Mathematical Social Sciences*, 31(C), 17–31.

## See Also

`PRIORrule`, `basicrule`, `weightedrule`, `hierarchicalrule`

## Examples

```
c <- c(1, 3, 7, 10) # Cost vector
SFCrule(c)
```

`SIGMArule` *Parametric family of rules*

## Description

`SIGMArule` calculates the contribution vector selected by a SIGMA rule.

## Usage

```
SIGMArule(c, a = 0.5)
```

## Arguments

| | |
|---|---|
| `c` | A numeric cost vector. |
| `a` | A numeric value in the range [0,1], controlling the parameterization of the rule. By default, `a = 0.5`. |

## Details

Let $N^i_- = \{j \in N : j < i\}$. For each $a \in [0,1]$, each $c \in C^N$, and each $i \in N \backslash \{n\}$, a $\sigma^a$ rule is defined by

$$\sigma^a_i(c) = \min\left\{ \min\left\{ \frac{1}{r-(i-1)+a}\Big(c_r - \sum_{j \in N^i_-} \sigma^a_j(c)\Big) : r = i, \ldots, n-1 \right\}, \frac{1}{n+1-i}\Big(c_n - \sum_{j \in N^i_-} \sigma^a_j(c)\Big) \right\},$$

$$\text{and } \sigma^a_n(c) = c_n - \sum_{i=1}^{n-1} \sigma^a_i(c).$$

In this rule, the closer the parameter $a$ is to 0, the more equal the distribution of payments among the agents will be, and vice versa. In fact, it is easy to verify that $\sigma^0 = \text{CEC}$ and $\sigma^1 = \text{SM}$.

## Value

A numeric contribution vector, where each element represents the payment of the different agents.

## References

Thomson, W. (2024). Cost allocation and airport problems. *Mathematical Social Sciences*, 31(C), 17–31.

van Gellekom, J. R. G. and Potters, J. A. M. (1999). Consistent solution rules for standard tree enterprises. Technical Report 9910, University of Nijmegen.

## See Also

`CECrule`, `SMrule`, `basicrule`, `hierarchicalrule`

## Examples

```
c <- c(1, 3, 7, 10) # Cost vector
SIGMArule(c) # a=0.5

# The SIGMA rule with a=0 is the CEC rule
a <- 0
all.equal(SIGMArule(c, a), CECrule(c))

# The SIGMA rule with a=1 is the SM rule
a <- 1
all.equal(SIGMArule(c, a), SMrule(c))
```

---

`SMrule` *Slack maximizer rule*

---

## Description

`SMrule` calculates the contribution vector selected by the SM rule.

## Usage

```
SMrule(c)
```

## Arguments

| | |
|---|---|
| `c` | A numeric cost vector. |

## Details

For each $c \in C^N$ and each $i \in N \backslash \{n\}$, the slack maximizer rule is defined by

$$\text{SM}_i(c) = \min\left\{ \frac{1}{r-i+2}\Big(c_r - \sum_{j \in N^i_-} \text{SM}_j(c)\Big) : r = i, \ldots, n-1 \right\},\ \text{SM}_n(c) = c_n - \sum_{i=1}^{n-1} SM_i(c)$$

This rule aims to maximize the 'slacks', that is, the available margin for each agent within the imposed constraints.

The contribution selected by the SM rule for a problem $c \in C^N$ coincides with the payoff vector assigned by the nucleolus to the associated cost game $v \in G^N$, that is, $\text{SM}(c) = \text{Nu}(v)$.

## Value

A numeric contribution vector, where each element represents the payment of the different agents.

## References

Littlechild, S. C. (1974). A simple expression for the nucleolus in a special case. *International Journal of Game Theory*, 3(1), 21-29.

Thomson, W. (2024). Cost allocation and airport problems. *Mathematical Social Sciences*, 31(C), 17–31.

## See Also

SIGMArule, basicrule, hierarchicalrule

## Examples

```
c <- c(1, 3, 7, 10) # Cost vector
SMrule(c)
```

---

`universitybus` *University bus travel data*

---

## Description

Data containing 4 variables with information about the round-trip journeys of university students traveling by bus from the Morrazo region (Galicia, Spain) to the University of Vigo.

## Usage

```
universitybus
```

## Format

A data frame with 13 rows and 4 columns (variables):

**district** Area where each student boards and alights from the bus

**town** Town to which each student belongs

**distance** Distance in kilometers of the road route (bus path) between a district and the University of Vigo

**passengers** Number of passengers per district

## Source

Own elaboration based on data retrieved from Google Maps (accessed on April 25, 2025).

## Examples

```
# We assume that the bus is entirely funded by the municipalities
# How are the cots distributed among them?

# Allocation by district (in km)
by_district <- multiclonesrules(universitybus$distance, universitybus$passengers,
agents_names = universitybus$district, labels = FALSE)

# Allocation by town (in km)
district_to_town <- setNames(universitybus$town, universitybus$district)
by_town <- t(rowsum(t(by_district), group = district_to_town[colnames(by_district)]))
print(by_town)

# Cost allocation by town (in euros)
cT <- 500 # Total cost of the journey (bus rental cost + variable costs)
max_dist <- max(universitybus$distance) # maximum distance
by_town_cost <- round(by_town * (cT / max_dist), 2) # km to euros
print(by_town_cost)
```

---

# weightedrule *Weighted allocation rule*

---

## Description

`weightedrule` calculates the contribution vector resulting from the payment allocation among the different agents using one of the various predefined weighted rules.

## Usage

```
weightedrule(c, w, rule)
```

## Arguments

| | |
|---|---|
| `c` | A numeric cost vector. |
| `w` | A numeric weight vector. |
| `rule` | A character string specifying the rule to apply.The rules that can be selected are: `"SFC"`, `"SEC"`, `"CSEC"`, `"CEC"`, `"CP"` and `"CEB"`. |

## Details

Let $w = (w_i)_{i \in N} \in \mathbb{R}^n$ be a positive weight vector, satisfying $w_i > 0$ for all $i \in N$ and $w(N) = 1$. Consider the $(n-1)$-standard simplex, defined as $\Delta_n = \{x \in \mathbb{R}^n : x \geq 0,\ x_1 + \cdots + x_n = 1\}$. Then, the set of all positive weight vectors corresponds to $\text{Int}(\Delta_n)$, the interior of the $(n-1)$-standard simplex.

A weighted rule is a mapping $\mathcal{R} : C^N \times \text{Int}(\Delta_N) \to \mathbb{R}^N$ which associates with a problem $c \in C^N$ and a positive weight vector $w \in \text{Int}(\Delta_n)$ a contribution vector $\mathcal{R}(c, w) \in X(c)$.

It is possible to define weighted versions of the rules: SFC, SEC, CEC, CP and CEB. In fact, two different rules emerge from the standard SEC rule: the weighted SEC rule and the coalition-weighted SEC rule. If $w_i = w_j$ for all $i, j \in N$, then the solution of weighted SEC$(c)$ and weighted CSEC$(c)$ coincides.

In all the rules, the higher the weight $w_i$, the more the corresponding agent will have to pay, except for the weighted CEB rule (the construction of this rule is based on the concept of allocating 'benefits', so it is logical that it is set up this way). Furthermore, as previously stated, all the rules, except for the weighted CSEC rule, require the weights to be positive. Although the weighted CSEC rule allows zero weights, it requires that at least one weight must be positive.

The weighted version of the SFC, SEC, CEC and CP rules is equal to their respective versions for clones, so the formulation of these rules will be the same for the version with clones. Only the CEB rule has a weighted version and a clone version that are different.

## Value

A numeric contribution vector, where each element represents the payment of the different agents.

## References

Bernárdez Ferradás, A., Mirás Calvo, M. Á., Quinteiro Sandomingo, C., and Sánchez-Rodríguez, E. (2025a). Airport problems with cloned agents. [Preprint manuscript].

Bernárdez Ferradás, A., Mirás Calvo, M. Á., Quinteiro Sandomingo, C., and Sánchez-Rodríguez, E. (2025b). A characterization of the CSEC rule for airport problems. [Prepint manuscript].

Sánchez-Rodríguez, E., Mirás Calvo, M. Á., Quinteiro Sandomingo, C., and Núñez Lugilde, I. (2024). Coalition-weighted Shapley values. *International Journal of Game Theory*, 53, 547-577.

Thomson, W. (2024). Cost allocation and airport problems. *Mathematical Social Sciences*, 31(C), 17–31.

## See Also

NScheck, NSset, basicrule, clonesrule

## Examples

```
c <- c(1, 3, 3, 7, 10) # Cost vector
w <- c(1, 4, 1, 2, 8)  # Weight vector

# Weighted SFC rule
weightedrule(c, w, "SFC")

# Weighted CEB rule
weightedrule(c, w, "CEB")

# Weighted SEC rule
weightedrule(c, w, "SEC")

# Weighted CSEC rule
w <- c(0, 4, 1, 2, 8) # New weight vector
weightedrule(c, w, "CSEC")
```

# Index