Package 'randomizr'

July 23, 2025

Title Easy-to-Use Tools for Common Forms of Random Assignment and Sampling

Version 1.0.0

Description Generates random assignments for common experimental designs and random samples for common sampling designs.

URL https://declaredesign.org/r/randomizr/, https://github.com/DeclareDesign/randomizr

BugReports https://github.com/DeclareDesign/randomizr/issues

Depends R (>= 3.5.0)

License MIT + file LICENSE

Encoding UTF-8

Suggests knitr, dplyr, testthat, rmarkdown

VignetteBuilder knitr

RoxygenNote 7.2.3

NeedsCompilation yes

Author Alexander Coppock [aut, cre] (ORCID: <https://orcid.org/0000-0002-5733-2386>), Jasper Cooper [ctb] (ORCID: <https://orcid.org/0000-0002-8639-3188>), Neal Fultz [ctb] (C version of restricted partitions), Graeme Blair [ctb] (ORCID: <https://orcid.org/0000-0001-9164-2102>)

Maintainer Alexander Coppock <acoppock@gmail.com>

Repository CRAN

Date/Publication 2023-08-10 06:30:02 UTC

Contents

block_and_cluster_ra 2
block_and_cluster_ra_probabilities 5
block_ra 8
block_ra_probabilities 11

cluster_ra . . . 13
cluster_ra_probabilities . . . 15
cluster_rs . . . 17
cluster_rs_probabilities . . . 19
complete_ra . . . 20
complete_ra_probabilities . . . 22
complete_rs . . . 24
complete_rs_probabilities . . . 26
conduct_ra . . . 27
custom_ra . . . 29
custom_ra_probabilities . . . 30
declare_ra . . . 31
declare_rs . . . 34
draw_rs . . . 36
obtain_condition_probabilities . . . 38
obtain_inclusion_probabilities . . . 41
obtain_num_permutations . . . 43
obtain_permutation_matrix . . . 44
obtain_permutation_probabilities . . . 46
randomizr . . . 46
simple_ra . . . 47
simple_ra_probabilities . . . 48
simple_rs . . . 50
simple_rs_probabilities . . . 51
strata_and_cluster_rs . . . 52
strata_and_cluster_rs_probabilities . . . 54
strata_rs . . . 55
strata_rs_probabilities . . . 57

Index **59**

`block_and_cluster_ra` *Blocked and Clustered Random Assignment*

Description

A random assignment procedure in which units are assigned as clusters and clusters are nested within blocks.

Usage

```
block_and_cluster_ra(
  blocks = NULL,
  clusters = NULL,
  prob = NULL,
  prob_unit = NULL,
  prob_each = NULL,
  m = NULL,
```

```
  m_unit = NULL,
  block_m = NULL,
  block_m_each = NULL,
  block_prob = NULL,
  block_prob_each = NULL,
  num_arms = NULL,
  conditions = NULL,
  check_inputs = TRUE
)
```

Arguments

blocks	A vector of length N that indicates which block each unit belongs to.
clusters	A vector of length N that indicates which cluster each unit belongs to.
prob	Use for a two-arm design in which either floor(N_clusters_block*prob) or ceiling(N_clusters_block*prob) clusters are assigned to treatment within each block. The probability of assignment to treatment is exactly prob because with probability 1-prob, floor(N_clusters_block*prob) clusters will be assigned to treatment and with probability prob, ceiling(N_clusters_block*prob) clusters will be assigned to treatment. prob must be a real number between 0 and 1 inclusive. (optional)
prob_unit	Use for a two arm design. Must of be of length N. tapply(prob_unit, blocks, unique) will be passed to block_prob.
prob_each	Use for a multi-arm design in which the values of prob_each determine the probabilities of assignment to each treatment condition. prob_each must be a numeric vector giving the probability of assignment to each condition. All entries must be nonnegative real numbers between 0 and 1 inclusive and the total must sum to 1. Because of integer issues, the exact number of clusters assigned to each condition may differ (slightly) from assignment to assignment, but the overall probability of assignment is exactly prob_each. (optional)
m	Use for a two-arm design in which the scalar m describes the fixed number of clusters assigned in each block. This number does not vary across blocks.
m_unit	Use for a two-arm design. Must be of length N. tapply(m_unit, blocks, unique) will be passed to block_m.
block_m	Use for a two-arm design in which block_m describes the number of clusters to assign to treatment within each block. block_m must be a numeric vector that is as long as the number of blocks and is in the same order as sort(unique(blocks)).
block_m_each	Use for a multi-arm design in which the values of block_m_each determine the number of clusters assigned to each condition. block_m_each must be a matrix with the same number of rows as blocks and the same number of columns as treatment arms. Cell entries are the number of clusters to be assigned to each treatment arm within each block. The rows should respect the ordering of the blocks as determined by sort(unique(blocks)). The columns should be in the order of conditions, if specified.
block_prob	Use for a two-arm design in which block_prob describes the probability of assignment to treatment within each block. Must be in the same order as sort(unique(blocks)). Differs from prob in that the probability of assignment can vary across blocks.

block_prob_each
: Use for a multi-arm design in which the values of block_prob_each determine the probabilities of assignment to each treatment condition. block_prob_each must be a matrix with the same number of rows as blocks and the same number of columns as treatment arms. Cell entries are the probabilities of assignment to treatment within each block. The rows should respect the ordering of the blocks as determined by sort(unique(blocks)). Use only if the probabilities of assignment should vary by block, otherwise use prob_each. Each row of block_prob_each must sum to 1.

num_arms
: The number of treatment arms. If unspecified, num_arms will be determined from the other arguments. (optional)

conditions
: A character vector giving the names of the treatment groups. If unspecified, the treatment groups will be named 0 (for control) and 1 (for treatment) in a two-arm trial and T1, T2, T3, in a multi-arm trial. An exception is a two-group design in which num_arms is set to 2, in which case the condition names are T1 and T2, as in a multi-arm trial with two arms. (optional)

check_inputs
: logical. Defaults to TRUE.

Value

A vector of length N that indicates the treatment condition of each unit.

Examples

```
clusters <- rep(letters, times=1:26)

blocks <- rep(NA, length(clusters))
blocks[clusters %in% letters[1:5]] <- "block_1"
blocks[clusters %in% letters[6:10]] <- "block_2"
blocks[clusters %in% letters[11:15]] <- "block_3"
blocks[clusters %in% letters[16:20]] <- "block_4"
blocks[clusters %in% letters[21:26]] <- "block_5"


table(blocks, clusters)

Z <- block_and_cluster_ra(blocks = blocks,
                          clusters = clusters)

table(Z, blocks)
table(Z, clusters)

Z <- block_and_cluster_ra(blocks = blocks,
                          clusters = clusters,
                          num_arms = 3)

table(Z, blocks)
table(Z, clusters)

Z <- block_and_cluster_ra(blocks = blocks,
```

```
                          clusters = clusters,
                          prob_each = c(.2, .5, .3))

block_m_each <- rbind(c(2, 3),
                      c(1, 4),
                      c(3, 2),
                      c(2, 3),
                      c(5, 1))

Z <- block_and_cluster_ra(blocks = blocks,
                          clusters = clusters,
                          block_m_each = block_m_each)

table(Z, blocks)
table(Z, clusters)
```

`block_and_cluster_ra_probabilities` *probabilities of assignment: Blocked and Clustered Random Assignment*

Description

probabilities of assignment: Blocked and Clustered Random Assignment

Usage

```
block_and_cluster_ra_probabilities(
  blocks = NULL,
  clusters = NULL,
  prob = NULL,
  prob_unit = NULL,
  prob_each = NULL,
  m = NULL,
  m_unit = NULL,
  block_m = NULL,
  block_m_each = NULL,
  block_prob = NULL,
  block_prob_each = NULL,
  num_arms = NULL,
  conditions = NULL,
  check_inputs = TRUE
)
```

Arguments

`blocks`	A vector of length N that indicates which block each unit belongs to.

`clusters` A vector of length N that indicates which cluster each unit belongs to.

`prob` Use for a two-arm design in which either floor(N_clusters_block*prob) or ceiling(N_clusters_block*prob) clusters are assigned to treatment within each block. The probability of assignment to treatment is exactly prob because with probability 1-prob, floor(N_clusters_block*prob) clusters will be assigned to treatment and with probability prob, ceiling(N_clusters_block*prob) clusters will be assigned to treatment. prob must be a real number between 0 and 1 inclusive. (optional)

`prob_unit` Use for a two arm design. Must of be of length N. tapply(prob_unit, blocks, unique) will be passed to `block_prob`.

`prob_each` Use for a multi-arm design in which the values of prob_each determine the probabilities of assignment to each treatment condition. prob_each must be a numeric vector giving the probability of assignment to each condition. All entries must be nonnegative real numbers between 0 and 1 inclusive and the total must sum to 1. Because of integer issues, the exact number of clusters assigned to each condition may differ (slightly) from assignment to assignment, but the overall probability of assignment is exactly prob_each. (optional)

`m` Use for a two-arm design in which the scalar m describes the fixed number of clusters assigned in each block. This number does not vary across blocks.

`m_unit` Use for a two-arm design. Must be of length N. tapply(m_unit, blocks, unique) will be passed to `block_m`.

`block_m` Use for a two-arm design in which block_m describes the number of clusters to assign to treatment within each block. block_m must be a numeric vector that is as long as the number of blocks and is in the same order as sort(unique(blocks)).

`block_m_each` Use for a multi-arm design in which the values of block_m_each determine the number of clusters assigned to each condition. block_m_each must be a matrix with the same number of rows as blocks and the same number of columns as treatment arms. Cell entries are the number of clusters to be assigned to each treatment arm within each block. The rows should respect the ordering of the blocks as determined by sort(unique(blocks)). The columns should be in the order of conditions, if specified.

`block_prob` Use for a two-arm design in which block_prob describes the probability of assignment to treatment within each block. Must be in the same order as sort(unique(blocks)). Differs from prob in that the probability of assignment can vary across blocks.

`block_prob_each`
Use for a multi-arm design in which the values of block_prob_each determine the probabilities of assignment to each treatment condition. block_prob_each must be a matrix with the same number of rows as blocks and the same number of columns as treatment arms. Cell entries are the probabilities of assignment to treatment within each block. The rows should respect the ordering of the blocks as determined by sort(unique(blocks)). Use only if the probabilities of assignment should vary by block, otherwise use prob_each. Each row of block_prob_each must sum to 1.

`num_arms` The number of treatment arms. If unspecified, num_arms will be determined from the other arguments. (optional)

conditions	A character vector giving the names of the treatment groups. If unspecified, the treatment groups will be named 0 (for control) and 1 (for treatment) in a two-arm trial and T1, T2, T3, in a multi-arm trial. An exception is a two-group design in which num_arms is set to 2, in which case the condition names are T1 and T2, as in a multi-arm trial with two arms. (optional)
check_inputs	logical. Defaults to TRUE.

Value

A matrix of probabilities of assignment

Examples

```
clusters <- rep(letters, times=1:26)
blocks <- rep(NA, length(clusters))
blocks[clusters %in% letters[1:5]] <- "block_1"
blocks[clusters %in% letters[6:10]] <- "block_2"
blocks[clusters %in% letters[11:15]] <- "block_3"
blocks[clusters %in% letters[16:20]] <- "block_4"
blocks[clusters %in% letters[21:26]] <- "block_5"


prob_mat <- block_and_cluster_ra_probabilities(clusters = clusters,
                                               blocks = blocks)
head(prob_mat)

prob_mat <- block_and_cluster_ra_probabilities(clusters = clusters,
                                               blocks = blocks,
                                               num_arms = 3)
head(prob_mat)

prob_mat <- block_and_cluster_ra_probabilities(clusters = clusters,
                                               blocks = blocks,
                                               prob_each = c(.2, .5, .3))
head(prob_mat)

block_m_each <- rbind(c(2, 3),
                      c(1, 4),
                      c(3, 2),
                      c(2, 3),
                      c(5, 1))

prob_mat <- block_and_cluster_ra_probabilities(clusters = clusters,
                                               blocks = blocks,
                                               block_m_each = block_m_each)
head(prob_mat)
```

block_ra *Block Random Assignment*

Description

block_ra implements a random assignment procedure in which units that are grouped into blocks defined by pre-treatment covariates are assigned using complete random assignment within block. For example, imagine that 50 of 100 men are assigned to treatment and 75 of 200 women are assigned to treatment.

Usage

```
block_ra(
  blocks = NULL,
  prob = NULL,
  prob_unit = NULL,
  prob_each = NULL,
  m = NULL,
  m_unit = NULL,
  block_m = NULL,
  block_m_each = NULL,
  block_prob = NULL,
  block_prob_each = NULL,
  num_arms = NULL,
  conditions = NULL,
  check_inputs = TRUE
)
```

Arguments

blocks	A vector of length N that indicates which block each unit belongs to. Can be a character, factor, or numeric vector. (required)
prob	Use for a two-arm design in which either floor(N_block*prob) or ceiling(N_block*prob) units are assigned to treatment within each block. The probability of assignment to treatment is exactly prob because with probability 1-prob, floor(N_block*prob) units will be assigned to treatment and with probability prob, ceiling(N_block*prob) units will be assigned to treatment. prob must be a real number between 0 and 1 inclusive. (optional)
prob_unit	Use for a two arm design. Must of be of length N. tapply(prob_unit, blocks, unique) will be passed to block_prob.
prob_each	Use for a multi-arm design in which the values of prob_each determine the probabilities of assignment to each treatment condition. prob_each must be a numeric vector giving the probability of assignment to each condition. All entries must be nonnegative real numbers between 0 and 1 inclusive and the total must sum to 1. Because of integer issues, the exact number of units assigned to each condition may differ (slightly) from assignment to assignment, but the overall probability of assignment is exactly prob_each. (optional)

`m` Use for a two-arm design in which the scalar m describes the fixed number of units to assign in each block. This number does not vary across blocks.

`m_unit` Use for a two-arm design. Must be of length N. tapply(m_unit, blocks, unique) will be passed to `block_m`.

`block_m` Use for a two-arm design in which the vector block_m describes the number of units to assign to treatment within each block. block_m must be a numeric vector that is as long as the number of blocks and is in the same order as sort(unique(blocks)).

`block_m_each` Use for a multi-arm design in which the values of block_m_each determine the number of units assigned to each condition. block_m_each must be a matrix with the same number of rows as blocks and the same number of columns as treatment arms. Cell entries are the number of units to be assigned to each treatment arm within each block. The rows should respect the ordering of the blocks as determined by sort(unique(blocks)). The columns should be in the order of conditions, if specified.

`block_prob` Use for a two-arm design in which block_prob describes the probability of assignment to treatment within each block. Must be in the same order as sort(unique(blocks)). Differs from prob in that the probability of assignment can vary across blocks.

`block_prob_each`
Use for a multi-arm design in which the values of block_prob_each determine the probabilities of assignment to each treatment condition. block_prob_each must be a matrix with the same number of rows as blocks and the same number of columns as treatment arms. Cell entries are the probabilities of assignment to treatment within each block. The rows should respect the ordering of the blocks as determined by sort(unique(blocks)). Use only if the probabilities of assignment should vary by block, otherwise use prob_each. Each row of block_prob_each must sum to 1.

`num_arms` The number of treatment arms. If unspecified, num_arms will be determined from the other arguments. (optional)

`conditions` A character vector giving the names of the treatment groups. If unspecified, the treatment groups will be named 0 (for control) and 1 (for treatment) in a two-arm trial and T1, T2, T3, in a multi-arm trial. An exception is a two-group design in which num_arms is set to 2, in which case the condition names are T1 and T2, as in a multi-arm trial with two arms. (optional)

`check_inputs` logical. Defaults to TRUE.

Value

A vector of length N that indicates the treatment condition of each unit. Is numeric in a two-arm trial and a factor variable (ordered by conditions) in a multi-arm trial.

Examples

```
# Two-arm Designs

blocks <- rep(c("A", "B","C"), times = c(50, 100, 200))
Z <- block_ra(blocks = blocks)
```

```
table(blocks, Z)

Z <- block_ra(blocks = blocks, prob = .3)
table(blocks, Z)

Z <- block_ra(blocks = blocks, block_prob = c(.1, .2, .3))
table(blocks, Z)

Z <- block_ra(blocks = blocks,
              prob_unit = rep(c(.1, .2, .3),
                              times = c(50, 100, 200)))
table(blocks, Z)

Z <- block_ra(blocks = blocks, m = 20)
table(blocks, Z)

Z <- block_ra(blocks = blocks, block_m = c(20, 30, 40))
table(blocks, Z)

Z <- block_ra(blocks = blocks,
              m_unit = rep(c(20, 30, 40),
                           times = c(50, 100, 200)))
table(blocks, Z)

block_m_each <- rbind(c(25, 25),
                 c(50, 50),
                 c(100, 100))

Z <- block_ra(blocks = blocks, block_m_each = block_m_each)
table(blocks, Z)

block_m_each <- rbind(c(10, 40),
                 c(30, 70),
                 c(50, 150))

Z <- block_ra(blocks = blocks, block_m_each = block_m_each,
              conditions = c("control", "treatment"))
table(blocks, Z)

# Multi-arm Designs
Z <- block_ra(blocks = blocks, num_arms = 3)
table(blocks, Z)

block_m_each <- rbind(c(10, 20, 20),
                 c(30, 50, 20),
                 c(50, 75, 75))
Z <- block_ra(blocks = blocks, block_m_each = block_m_each)
table(blocks, Z)

Z <- block_ra(blocks = blocks, block_m_each = block_m_each,
              conditions = c("control", "placebo", "treatment"))
table(blocks, Z)
```

```
Z <- block_ra(blocks = blocks, prob_each = c(.1, .1, .8))
table(blocks, Z)
```

`block_ra_probabilities` *probabilities of assignment: Block Random Assignment*

Description

probabilities of assignment: Block Random Assignment

Usage

```
block_ra_probabilities(
  blocks = NULL,
  prob = NULL,
  prob_unit = NULL,
  prob_each = NULL,
  m = NULL,
  m_unit = NULL,
  block_m = NULL,
  block_m_each = NULL,
  block_prob = NULL,
  block_prob_each = NULL,
  num_arms = NULL,
  conditions = NULL,
  check_inputs = TRUE
)
```

Arguments

`blocks`	A vector of length N that indicates which block each unit belongs to. Can be a character, factor, or numeric vector. (required)
`prob`	Use for a two-arm design in which either floor(N_block*prob) or ceiling(N_block*prob) units are assigned to treatment within each block. The probability of assignment to treatment is exactly prob because with probability 1-prob, floor(N_block*prob) units will be assigned to treatment and with probability prob, ceiling(N_block*prob) units will be assigned to treatment. prob must be a real number between 0 and 1 inclusive. (optional)
`prob_unit`	Use for a two arm design. Must of be of length N. tapply(prob_unit, blocks, unique) will be passed to `block_prob`.

prob_each — Use for a multi-arm design in which the values of prob_each determine the probabilities of assignment to each treatment condition. prob_each must be a numeric vector giving the probability of assignment to each condition. All entries must be nonnegative real numbers between 0 and 1 inclusive and the total must sum to 1. Because of integer issues, the exact number of units assigned to each condition may differ (slightly) from assignment to assignment, but the overall probability of assignment is exactly prob_each. (optional)

m — Use for a two-arm design in which the scalar m describes the fixed number of units to assign in each block. This number does not vary across blocks.

m_unit — Use for a two-arm design. Must be of length N. tapply(m_unit, blocks, unique) will be passed to block_m.

block_m — Use for a two-arm design in which the vector block_m describes the number of units to assign to treatment within each block. block_m must be a numeric vector that is as long as the number of blocks and is in the same order as sort(unique(blocks)).

block_m_each — Use for a multi-arm design in which the values of block_m_each determine the number of units assigned to each condition. block_m_each must be a matrix with the same number of rows as blocks and the same number of columns as treatment arms. Cell entries are the number of units to be assigned to each treatment arm within each block. The rows should respect the ordering of the blocks as determined by sort(unique(blocks)). The columns should be in the order of conditions, if specified.

block_prob — Use for a two-arm design in which block_prob describes the probability of assignment to treatment within each block. Must be in the same order as sort(unique(blocks)). Differs from prob in that the probability of assignment can vary across blocks.

block_prob_each — Use for a multi-arm design in which the values of block_prob_each determine the probabilities of assignment to each treatment condition. block_prob_each must be a matrix with the same number of rows as blocks and the same number of columns as treatment arms. Cell entries are the probabilities of assignment to treatment within each block. The rows should respect the ordering of the blocks as determined by sort(unique(blocks)). Use only if the probabilities of assignment should vary by block, otherwise use prob_each. Each row of block_prob_each must sum to 1.

num_arms — The number of treatment arms. If unspecified, num_arms will be determined from the other arguments. (optional)

conditions — A character vector giving the names of the treatment groups. If unspecified, the treatment groups will be named 0 (for control) and 1 (for treatment) in a two-arm trial and T1, T2, T3, in a multi-arm trial. An exception is a two-group design in which num_arms is set to 2, in which case the condition names are T1 and T2, as in a multi-arm trial with two arms. (optional)

check_inputs — logical. Defaults to TRUE.

Value

A matrix of probabilities of assignment

Examples

```
blocks <- rep(c("A", "B","C"), times = c(50, 100, 200))
prob_mat <- block_ra_probabilities(blocks = blocks)
head(prob_mat)

prob_mat <- block_ra_probabilities(blocks = blocks, m = 20)
head(prob_mat)

block_m_each <- rbind(c(25, 25),
                c(50, 50),
                c(100, 100))

prob_mat <- block_ra_probabilities(blocks = blocks, block_m_each = block_m_each)
head(prob_mat)

block_m_each <- rbind(c(10, 40),
                c(30, 70),
                c(50, 150))

prob_mat <- block_ra_probabilities(blocks = blocks,
                                   block_m_each = block_m_each,
                                   conditions = c("control", "treatment"))
head(prob_mat)

prob_mat <- block_ra_probabilities(blocks = blocks, num_arms = 3)
head(prob_mat)

block_m_each <- rbind(c(10, 20, 20),
                c(30, 50, 20),
                c(50, 75, 75))
prob_mat <- block_ra_probabilities(blocks = blocks, block_m_each = block_m_each)
head(prob_mat)

prob_mat <- block_ra_probabilities(blocks=blocks, block_m_each=block_m_each,
                      conditions=c("control", "placebo", "treatment"))
head(prob_mat)

prob_mat <- block_ra_probabilities(blocks=blocks, prob_each=c(.1, .1, .8))
head(prob_mat)
```

`cluster_ra` *Cluster Random Assignment*

Description

cluster_ra implements a random assignment procedure in which groups of units are assigned together (as a cluster) to treatment conditions. This function conducts complete random assignment at the cluster level, unless simple = TRUE, in which case `simple_ra` analogues are used.

Usage

```
cluster_ra(
  clusters = NULL,
  m = NULL,
  m_unit = NULL,
  m_each = NULL,
  prob = NULL,
  prob_unit = NULL,
  prob_each = NULL,
  num_arms = NULL,
  conditions = NULL,
  simple = FALSE,
  check_inputs = TRUE
)
```

Arguments

clusters	A vector of length N that indicates which cluster each unit belongs to.
m	Use for a two-arm design in which m clusters are assigned to treatment and N_clusters-m clusters are assigned to control. (optional)
m_unit	Use for a two-arm design in which exactly unique(m_unit) clusters are assigned to treatment and the remainder are assigned to control. m_unit must be of length N and must be the same for all units (optional)
m_each	Use for a multi-arm design in which the values of m_each determine the number of clusters assigned to each condition. m_each must be a numeric vector in which each entry is a nonnegative integer that describes how many clusters should be assigned to the 1st, 2nd, 3rd... treatment condition. m_each must sum to N. (optional)
prob	Use for a two-arm design in which either floor(N_clusters*prob) or ceiling(N_clusters*prob) clusters are assigned to treatment. The probability of assignment to treatment is exactly prob because with probability 1-prob, floor(N_clusters*prob) clusters will be assigned to treatment and with probability prob, ceiling(N_clusters*prob) clusters will be assigned to treatment. prob must be a real number between 0 and 1 inclusive. (optional)
prob_unit	Use for a two-arm design. unique(prob_unit) will be passed to the prob argument and must be the same for all units.
prob_each	Use for a multi-arm design in which the values of prob_each determine the probabilities of assignment to each treatment condition. prob_each must be a numeric vector giving the probability of assignment to each condition. All entries must be nonnegative real numbers between 0 and 1 inclusive and the total must sum to 1. Because of integer issues, the exact number of clusters assigned to each condition may differ (slightly) from assignment to assignment, but the overall probability of assignment is exactly prob_each. (optional)
num_arms	The total number of treatment arms. If unspecified, will be determined from the length of m_each or conditions.

conditions	A character vector giving the names of the treatment groups. If unspecified, the treatment groups will be named T1, T2, T3, etc.
simple	logical, defaults to FALSE. If TRUE, simple random assignment of clusters to conditions is used. When simple = TRUE, please do not specify m or m_each.
check_inputs	logical. Defaults to TRUE.

Value

A vector of length N that indicates the treatment condition of each unit.

Examples

```
# Two Group Designs
clusters <- rep(letters, times=1:26)

Z <- cluster_ra(clusters = clusters)
table(Z, clusters)

Z <- cluster_ra(clusters = clusters, m = 13)
table(Z, clusters)

Z <- cluster_ra(clusters = clusters, m_each = c(10, 16),
                conditions = c("control", "treatment"))
table(Z, clusters)

# Multi-arm Designs
Z <- cluster_ra(clusters = clusters, num_arms = 3)
table(Z, clusters)

Z <- cluster_ra(clusters = clusters, m_each = c(7, 7, 12))
table(Z, clusters)

Z <- cluster_ra(clusters = clusters, m_each = c(7, 7, 12),
                conditions = c("control", "placebo", "treatment"))
table(Z, clusters)

Z <- cluster_ra(clusters = clusters,
                conditions = c("control", "placebo", "treatment"))
table(Z, clusters)
```

---

cluster_ra_probabilities

probabilities of assignment: Cluster Random Assignment

Description

probabilities of assignment: Cluster Random Assignment

Usage

```
cluster_ra_probabilities(
  clusters = NULL,
  m = NULL,
  m_unit = NULL,
  m_each = NULL,
  prob = NULL,
  prob_unit = NULL,
  prob_each = NULL,
  num_arms = NULL,
  conditions = NULL,
  simple = FALSE,
  check_inputs = TRUE
)
```

Arguments

clusters	A vector of length N that indicates which cluster each unit belongs to.
m	Use for a two-arm design in which m clusters are assigned to treatment and N_clusters-m clusters are assigned to control. (optional)
m_unit	Use for a two-arm design in which exactly unique(m_unit) clusters are assigned to treatment and the remainder are assigned to control. m_unit must be of length N and must be the same for all units (optional)
m_each	Use for a multi-arm design in which the values of m_each determine the number of clusters assigned to each condition. m_each must be a numeric vector in which each entry is a nonnegative integer that describes how many clusters should be assigned to the 1st, 2nd, 3rd... treatment condition. m_each must sum to N. (optional)
prob	Use for a two-arm design in which either floor(N_clusters*prob) or ceiling(N_clusters*prob) clusters are assigned to treatment. The probability of assignment to treatment is exactly prob because with probability 1-prob, floor(N_clusters*prob) clusters will be assigned to treatment and with probability prob, ceiling(N_clusters*prob) clusters will be assigned to treatment. prob must be a real number between 0 and 1 inclusive. (optional)
prob_unit	Use for a two-arm design. unique(prob_unit) will be passed to the prob argument and must be the same for all units.
prob_each	Use for a multi-arm design in which the values of prob_each determine the probabilities of assignment to each treatment condition. prob_each must be a numeric vector giving the probability of assignment to each condition. All entries must be nonnegative real numbers between 0 and 1 inclusive and the total must sum to 1. Because of integer issues, the exact number of clusters assigned to each condition may differ (slightly) from assignment to assignment, but the overall probability of assignment is exactly prob_each. (optional)
num_arms	The total number of treatment arms. If unspecified, will be determined from the length of m_each or conditions.

conditions	A character vector giving the names of the treatment groups. If unspecified, the treatment groups will be named T1, T2, T3, etc.
simple	logical, defaults to FALSE. If TRUE, simple random assignment of clusters to conditions is used. When simple = TRUE, please do not specify m or m_each.
check_inputs	logical. Defaults to TRUE.

Value

A matrix of probabilities of assignment

Examples

```
# Two Group Designs
clusters <- rep(letters, times = 1:26)
prob_mat <- cluster_ra_probabilities(clusters = clusters)
head(prob_mat)

prob_mat <- cluster_ra_probabilities(clusters = clusters, m = 10)
head(prob_mat)

prob_mat <- cluster_ra_probabilities(clusters = clusters,
                                     m_each = c(9, 17),
                                     conditions = c("control", "treatment"))

# Multi-arm Designs
prob_mat <- cluster_ra_probabilities(clusters = clusters, num_arms = 3)
head(prob_mat)

prob_mat <- cluster_ra_probabilities(clusters = clusters, m_each = c(7, 7, 12))
head(prob_mat)

prob_mat <- cluster_ra_probabilities(clusters = clusters, m_each = c(7, 7, 12),
                          conditions=c("control", "placebo", "treatment"))
head(prob_mat)

prob_mat <- cluster_ra_probabilities(clusters = clusters,
                          conditions=c("control", "placebo", "treatment"))
head(prob_mat)

prob_mat <- cluster_ra_probabilities(clusters = clusters,
                                     prob_each = c(.1, .2, .7))
head(prob_mat)
```

---

cluster_rs *Cluster Random Sampling*

Description

cluster_rs implements a random sampling procedure in which groups of units are sampled together (as a cluster). This function conducts complete random sampling at the cluster level, unless simple = TRUE, in which case simple_rs analogues are used.

Usage

```
cluster_rs(
  clusters = NULL,
  n = NULL,
  n_unit = NULL,
  prob = NULL,
  prob_unit = NULL,
  simple = FALSE,
  check_inputs = TRUE
)
```

Arguments

clusters	A vector of length N that indicates which cluster each unit belongs to.
n	Use for a design in which n clusters are sampled. (optional)
n_unit	unique(n_unit) will be passed to n. Must be the same for all units (optional)
prob	Use for a design in which either floor(N_clusters*prob) or ceiling(N_clusters*prob) clusters are sampled. The probability of being sampled is exactly prob because with probability 1-prob, floor(N_clusters*prob) clusters will be sampled and with probability prob, ceiling(N_clusters*prob) clusters will be sampled. prob must be a real number between 0 and 1 inclusive. (optional)
prob_unit	unique(prob_unit) will be passed to the prob argument and must be the same for all units.
simple	logical, defaults to FALSE. If TRUE, simple random sampling of clusters. When simple = TRUE, please do not specify n.
check_inputs	logical. Defaults to TRUE.

Value

A numeric vector of length N that indicates if a unit is sampled (1) or not (0).

Examples

```
clusters <- rep(letters, times=1:26)

S <- cluster_rs(clusters = clusters)
table(S, clusters)

S <- cluster_rs(clusters = clusters, n = 13)
table(S, clusters)
```

cluster_rs_probabilities
Inclusion Probabilities: Cluster Sampling

Description

Inclusion Probabilities: Cluster Sampling

Usage

```
cluster_rs_probabilities(
  clusters = NULL,
  n = NULL,
  n_unit = NULL,
  prob = NULL,
  prob_unit = NULL,
  simple = FALSE,
  check_inputs = TRUE
)
```

Arguments

clusters	A vector of length N that indicates which cluster each unit belongs to.
n	Use for a design in which n clusters are sampled. (optional)
n_unit	unique(n_unit) will be passed to n. Must be the same for all units (optional)
prob	Use for a design in which either floor(N_clusters*prob) or ceiling(N_clusters*prob) clusters are sampled. The probability of being sampled is exactly prob because with probability 1-prob, floor(N_clusters*prob) clusters will be sampled and with probability prob, ceiling(N_clusters*prob) clusters will be sampled. prob must be a real number between 0 and 1 inclusive. (optional)
prob_unit	unique(prob_unit) will be passed to the prob argument and must be the same for all units.
simple	logical, defaults to FALSE. If TRUE, simple random sampling of clusters. When simple = TRUE, please do not specify n.
check_inputs	logical. Defaults to TRUE.

Value

A vector length N indicating the probability of being sampled.

Examples

```
# Two Group Designs
clusters <- rep(letters, times = 1:26)
probs <- cluster_rs_probabilities(clusters = clusters)
table(probs, clusters)
```

```
prob_mat <- cluster_rs_probabilities(clusters = clusters, n = 10)
table(probs, clusters)

prob_mat <- cluster_rs_probabilities(clusters = clusters, prob = .3)
table(probs, clusters)
```

`complete_ra` *Complete Random Assignment*

Description

complete_ra implements a random assignment procedure in which fixed numbers of units are assigned to treatment conditions. The canonical example of complete random assignment is a procedure in which exactly m of N units are assigned to treatment and N-m units are assigned to control.

Users can set the exact number of units to assign to each condition with m or m_each. Alternatively, users can specify probabilities of assignment with prob or prob_each and complete_ra will infer the correct number of units to assign to each condition. In a two-arm design, complete_ra will either assign floor(N*prob) or ceiling(N*prob) units to treatment, choosing between these two values to ensure that the overall probability of assignment is exactly prob. In a multi-arm design, complete_ra will first assign floor(N*prob_each) units to their respective conditions, then will assign the remaining units using simple random assignment, choosing these second-stage probabilities so that the overall probabilities of assignment are exactly prob_each.

In most cases, users should specify N and not more than one of m, m_each, prob, prob_each, or num_arms.

If only N is specified, a two-arm trial in which N/2 units are assigned to treatment is assumed. If N is odd, either floor(N/2) units or ceiling(N/2) units will be assigned to treatment.

Usage

```
complete_ra(
  N,
  m = NULL,
  m_unit = NULL,
  m_each = NULL,
  prob = NULL,
  prob_unit = NULL,
  prob_each = NULL,
  num_arms = NULL,
  conditions = NULL,
  check_inputs = TRUE
)
```

Arguments

N	The number of units. N must be a positive integer. (required)
m	Use for a two-arm design in which m units are assigned to treatment and N-m units are assigned to control. (optional)
m_unit	Use for a two-arm design in which exactly unique(m_unit) units are assigned to treatment and the remainder are assigned to control. m_unit must be of length N and must be the same for all units (optional)
m_each	Use for a multi-arm design in which the values of m_each determine the number of units assigned to each condition. m_each must be a numeric vector in which each entry is a nonnegative integer that describes how many units should be assigned to the 1st, 2nd, 3rd... treatment condition. m_each must sum to N. (optional)
prob	Use for a two-arm design in which either floor(N*prob) or ceiling(N*prob) units are assigned to treatment. The probability of assignment to treatment is exactly prob because with probability 1-prob, floor(N*prob) units will be assigned to treatment and with probability prob, ceiling(N*prob) units will be assigned to treatment. prob must be a real number between 0 and 1 inclusive. (optional)
prob_unit	Use for a two-arm design. unique(prob_unit) will be passed to the prob argument and must be the same for all units.
prob_each	Use for a multi-arm design in which the values of prob_each determine the probabilities of assignment to each treatment condition. prob_each must be a numeric vector giving the probability of assignment to each condition. All entries must be nonnegative real numbers between 0 and 1 inclusive and the total must sum to 1. Because of integer issues, the exact number of units assigned to each condition may differ (slightly) from assignment to assignment, but the overall probability of assignment is exactly prob_each. (optional)
num_arms	The number of treatment arms. If unspecified, num_arms will be determined from the other arguments. (optional)
conditions	A character vector giving the names of the treatment groups. If unspecified, the treatment groups will be named 0 (for control) and 1 (for treatment) in a two-arm trial and T1, T2, T3, in a multi-arm trial. An exception is a two-group design in which num_arms is set to 2, in which case the condition names are T1 and T2, as in a multi-arm trial with two arms. (optional)
check_inputs	logical. Defaults to TRUE.

Value

A vector of length N that indicates the treatment condition of each unit. Is numeric in a two-arm trial and a factor variable (ordered by conditions) in a multi-arm trial.

Examples

```
# Two-arm Designs
Z <- complete_ra(N = 100)
table(Z)
```

```
Z <- complete_ra(N = 100, m = 50)
table(Z)

Z <- complete_ra(N = 100, m_unit = rep(50, 100))
table(Z)

Z <- complete_ra(N = 100, prob = .111)
table(Z)

Z <- complete_ra(N = 100, prob_unit = rep(0.1, 100))
table(Z)

Z <- complete_ra(N = 100, conditions = c("control", "treatment"))
table(Z)


# Multi-arm Designs
Z <- complete_ra(N = 100, num_arms = 3)
table(Z)

Z <- complete_ra(N = 100, m_each = c(30, 30, 40))
table(Z)

Z <- complete_ra(N = 100, prob_each = c(.1, .2, .7))
table(Z)

Z <- complete_ra(N = 100, conditions = c("control", "placebo", "treatment"))
table(Z)

# Special Cases
# Two-arm trial where the conditions are by default "T1" and "T2"
Z <- complete_ra(N = 100, num_arms = 2)
table(Z)

# If N = m, assign with 100% probability
complete_ra(N=2, m=2)

# Up through randomizr 0.12.0,
complete_ra(N=1, m=1) # assigned with 50% probability
# This behavior has been deprecated
```

`complete_ra_probabilities` *probabilities of assignment: Complete Random Assignment*

Description

probabilities of assignment: Complete Random Assignment

Usage

```
complete_ra_probabilities(
  N,
  m = NULL,
  m_unit = NULL,
  m_each = NULL,
  prob = NULL,
  prob_unit = NULL,
  prob_each = NULL,
  num_arms = NULL,
  conditions = NULL,
  check_inputs = TRUE
)
```

Arguments

N	The number of units. N must be a positive integer. (required)
m	Use for a two-arm design in which m units are assigned to treatment and N-m units are assigned to control. (optional)
m_unit	Use for a two-arm design in which exactly unique(m_unit) units are assigned to treatment and the remainder are assigned to control. m_unit must be of length N and must be the same for all units (optional)
m_each	Use for a multi-arm design in which the values of m_each determine the number of units assigned to each condition. m_each must be a numeric vector in which each entry is a nonnegative integer that describes how many units should be assigned to the 1st, 2nd, 3rd... treatment condition. m_each must sum to N. (optional)
prob	Use for a two-arm design in which either floor(N*prob) or ceiling(N*prob) units are assigned to treatment. The probability of assignment to treatment is exactly prob because with probability 1-prob, floor(N*prob) units will be assigned to treatment and with probability prob, ceiling(N*prob) units will be assigned to treatment. prob must be a real number between 0 and 1 inclusive. (optional)
prob_unit	Use for a two-arm design. unique(prob_unit) will be passed to the prob argument and must be the same for all units.
prob_each	Use for a multi-arm design in which the values of prob_each determine the probabilities of assignment to each treatment condition. prob_each must be a numeric vector giving the probability of assignment to each condition. All entries must be nonnegative real numbers between 0 and 1 inclusive and the total must sum to 1. Because of integer issues, the exact number of units assigned to each condition may differ (slightly) from assignment to assignment, but the overall probability of assignment is exactly prob_each. (optional)
num_arms	The number of treatment arms. If unspecified, num_arms will be determined from the other arguments. (optional)
conditions	A character vector giving the names of the treatment groups. If unspecified, the treatment groups will be named 0 (for control) and 1 (for treatment) in a two-arm trial and T1, T2, T3, in a multi-arm trial. An exception is a two-group design in

which num_arms is set to 2, in which case the condition names are T1 and T2, as in a multi-arm trial with two arms. (optional)

check_inputs logical. Defaults to TRUE.

Value

A matrix of probabilities of assignment

Examples

```
# 2-arm designs
prob_mat <- complete_ra_probabilities(N=100)
head(prob_mat)

prob_mat <- complete_ra_probabilities(N=100, m=50)
head(prob_mat)

prob_mat <- complete_ra_probabilities(N=100, prob = .3)
head(prob_mat)

prob_mat <- complete_ra_probabilities(N=100, m_each = c(30, 70),
                          conditions = c("control", "treatment"))
head(prob_mat)

# Multi-arm Designs
prob_mat <- complete_ra_probabilities(N=100, num_arms=3)
head(prob_mat)

prob_mat <- complete_ra_probabilities(N=100, m_each=c(30, 30, 40))
head(prob_mat)

prob_mat <- complete_ra_probabilities(N=100, m_each=c(30, 30, 40),
                          conditions=c("control", "placebo", "treatment"))
head(prob_mat)

prob_mat <- complete_ra_probabilities(N=100, conditions=c("control", "placebo", "treatment"))
head(prob_mat)

prob_mat <- complete_ra_probabilities(N=100, prob_each = c(.2, .7, .1))
head(prob_mat)
```

---

complete_rs *Complete Random Sampling*

Description

complete_rs implements a random sampling procedure in which fixed numbers of units are sampled. The canonical example of complete random sampling is a procedure in which exactly n of N

units are sampled.

Users can set the exact number of units to sample with n. Alternatively, users can specify the probability of being sampled with prob and complete_rs will infer the correct number of units to sample. complete_rs will either sample floor(N*prob) or ceiling(N*prob) units, choosing between these two values to ensure that the overall probability of being sampled is exactly prob. Users should specify N and not more than one of n or prob.

If only N is specified, N/2 units will be sampled. If N is odd, either floor(N/2) units or ceiling(N/2) units will be sampled.

Usage

```
complete_rs(
  N,
  n = NULL,
  n_unit = NULL,
  prob = NULL,
  prob_unit = NULL,
  check_inputs = TRUE
)
```

Arguments

N	The number of units. N must be a positive integer. (required)
n	Use for a design in which exactly n units are sampled. (optional)
n_unit	unique(n_unit) will be passed to n. Must be the same for all units (optional)
prob	Use for a design in which either floor(N*prob) or ceiling(N*prob) units are sampled. The probability of being sampled is exactly prob because with probability 1-prob, floor(N*prob) units will be sampled and with probability prob, ceiling(N*prob) units will be sampled. prob must be a real number between 0 and 1 inclusive. (optional)
prob_unit	unique(prob_unit) will be passed to the prob argument and must be the same for all units.
check_inputs	logical. Defaults to TRUE.

Value

A numeric vector of length N that indicates if a unit is sampled (1) or not (0).

Examples

```
S <- complete_rs(N = 100)
table(S)

S <- complete_rs(N = 100, n = 50)
table(S)
```

```
S <- complete_rs(N = 100, n_unit = rep(50, 100))
table(S)

S <- complete_rs(N = 100, prob = .111)
table(S)

S <- complete_rs(N = 100, prob_unit = rep(.1, 100))
table(S)

# If N = n, sample with 100% probability...
complete_rs(N=2, n=2)

# Up through randomizr 0.12.0,
# This behavior has been deprecated
complete_rs(N=1, n=1) # sampled with 50% probability
```

`complete_rs_probabilities` *Inclusion Probabilities: Complete Random Sampling*

Description

Inclusion Probabilities: Complete Random Sampling

Usage

```
complete_rs_probabilities(
  N,
  n = NULL,
  n_unit = NULL,
  prob = NULL,
  prob_unit = NULL,
  check_inputs = TRUE
)
```

Arguments

`N`	The number of units. N must be a positive integer. (required)
`n`	Use for a design in which exactly n units are sampled. (optional)
`n_unit`	unique(n_unit) will be passed to n. Must be the same for all units (optional)
`prob`	Use for a design in which either floor(N*prob) or ceiling(N*prob) units are sampled. The probability of being sampled is exactly prob because with probability 1-prob, floor(N*prob) units will be sampled and with probability prob, ceiling(N*prob) units will be sampled. prob must be a real number between 0 and 1 inclusive. (optional)

prob_unit	unique(prob_unit) will be passed to the prob argument and must be the same for all units.
check_inputs	logical. Defaults to TRUE.

Value

A vector length N indicating the probability of being sampled.

Examples

```
probs <- complete_rs_probabilities(N = 100)
table(probs)

probs <- complete_rs_probabilities(N = 100, n = 50)
table(probs)

probs <- complete_rs_probabilities(N=100, prob = .3)
table(probs)
```

---

conduct_ra *Conduct a random assignment*

Description

You can either give conduct_ra() an declaration, as created by declare_ra or you can specify the other arguments to describe a random assignment procedure.

Usage

```
conduct_ra(
  declaration = NULL,
  N = NULL,
  blocks = NULL,
  clusters = NULL,
  m = NULL,
  m_unit = NULL,
  m_each = NULL,
  prob = NULL,
  prob_unit = NULL,
  prob_each = NULL,
  block_m = NULL,
  block_m_each = NULL,
  block_prob = NULL,
  block_prob_each = NULL,
  num_arms = NULL,
  conditions = NULL,
  simple = FALSE,
```

```
  permutation_matrix = NULL,
  check_inputs = TRUE
)
```

Arguments

declaration	A random assignment declaration, created by declare_ra.
N	The number of units. N must be a positive integer. (required)
blocks	A vector of length N that indicates which block each unit belongs to.
clusters	A vector of length N that indicates which cluster each unit belongs to.
m	Use for a two-arm design in which m units (or clusters) are assigned to treatment and N-m units (or clusters) are assigned to control. In a blocked design, exactly m units in each block will be treated. (optional)
m_unit	Use for a two-arm trial. Under complete random assignment, must be constant across units. Under blocked random assignment, must be constant within blocks.
m_each	Use for a multi-arm design in which the values of m_each determine the number of units (or clusters) assigned to each condition. m_each must be a numeric vector in which each entry is a nonnegative integer that describes how many units (or clusters) should be assigned to the 1st, 2nd, 3rd... treatment condition. m_each must sum to N. (optional)
prob	Use for a two-arm design in which either floor(N*prob) or ceiling(N*prob) units (or clusters) are assigned to treatment. The probability of assignment to treatment is exactly prob because with probability 1-prob, floor(N*prob) units (or clusters) will be assigned to treatment and with probability prob, ceiling(N*prob) units (or clusters) will be assigned to treatment. prob must be a real number between 0 and 1 inclusive. (optional)
prob_unit	Use for a two arm design. Must of be of length N. Under simple random assignment, can be different for each unit or cluster. Under complete random assignment, must be constant across units. Under blocked random assignment, must be constant within blocks.
prob_each	Use for a multi-arm design in which the values of prob_each determine the probabilities of assignment to each treatment condition. prob_each must be a numeric vector giving the probability of assignment to each condition. All entries must be nonnegative real numbers between 0 and 1 inclusive and the total must sum to 1. Because of integer issues, the exact number of units assigned to each condition may differ (slightly) from assignment to assignment, but the overall probability of assignment is exactly prob_each. (optional)
block_m	Use for a two-arm design in which block_m describes the number of units to assign to treatment within each block. Note that in previous versions of randomizr, block_m behaved like block_m_each.
block_m_each	Use for a multi-arm design in which the values of block_m_each determine the number of units (or clusters) assigned to each condition. block_m_each must be a matrix with the same number of rows as blocks and the same number of columns as treatment arms. Cell entries are the number of units (or clusters) to

be assigned to each treatment arm within each block. The rows should respect the ordering of the blocks as determined by sort(unique(blocks)). The columns should be in the order of conditions, if specified.

`block_prob` Use for a two-arm design in which block_prob describes the probability of assignment to treatment within each block. Differs from prob in that the probability of assignment can vary across blocks.

`block_prob_each`
Use for a multi-arm design in which the values of block_prob_each determine the probabilities of assignment to each treatment condition. block_prob_each must be a matrix with the same number of rows as blocks and the same number of columns as treatment arms. Cell entries are the probabilities of assignment to treatment within each block. The rows should respect the ordering of the blocks as determined by sort(unique(blocks)). Use only if the probabilities of assignment should vary by block, otherwise use prob_each. Each row of block_prob_each must sum to 1.

`num_arms` The number of treatment arms. If unspecified, num_arms will be determined from the other arguments. (optional)

`conditions` A character vector giving the names of the treatment groups. If unspecified, the treatment groups will be named 0 (for control) and 1 (for treatment) in a two-arm trial and T1, T2, T3, in a multi-arm trial. An exception is a two-group design in which num_arms is set to 2, in which case the condition names are T1 and T2, as in a multi-arm trial with two arms. (optional)

`simple` logical, defaults to FALSE. If TRUE, simple random assignment is used. When `simple = TRUE`, please do not specify m, m_each, block_m, or block_m_each. If `simple = TRUE`, `prob` and `prob_each` may vary by unit.

`permutation_matrix`
for custom random assignment procedures.

`check_inputs` logical. Defaults to TRUE.

Examples

```
declaration <- declare_ra(N = 100, m_each = c(30, 30, 40))
Z <- conduct_ra(declaration = declaration)
table(Z)

# equivalent to

Z <- conduct_ra(N = 100, m_each = c(30, 30, 40))
table(Z)
```

`custom_ra` *Custom Random Assignment*

Description

TODO

Usage

```
custom_ra(permutation_matrix)
```

Arguments

permutation_matrix	A permutation matrix

Value

A vector of length N that indicates the treatment condition of each unit. Is numeric in a two-arm trial and a factor variable (ordered by conditions) in a multi-arm trial.

Examples

```
# TODO
```

---

custom_ra_probabilities

probabilities of assignment: Custom Random Assignment

Description

probabilities of assignment: Custom Random Assignment

Usage

```
custom_ra_probabilities(permutation_matrix)
```

Arguments

permutation_matrix	A permutation matrix

Value

A matrix of probabilities of assignment

Examples

```
# TODO
```

declare_ra *Declare a random assignment procedure.*

Description

Declare a random assignment procedure.

Usage

```
declare_ra(
  N = NULL,
  blocks = NULL,
  clusters = NULL,
  m = NULL,
  m_unit = NULL,
  m_each = NULL,
  prob = NULL,
  prob_unit = NULL,
  prob_each = NULL,
  block_m = NULL,
  block_m_each = NULL,
  block_prob = NULL,
  block_prob_each = NULL,
  num_arms = NULL,
  conditions = NULL,
  simple = FALSE,
  permutation_matrix = NULL,
  check_inputs = TRUE
)
```

Arguments

N	The number of units. N must be a positive integer. (required)
blocks	A vector of length N that indicates which block each unit belongs to.
clusters	A vector of length N that indicates which cluster each unit belongs to.
m	Use for a two-arm design in which m units (or clusters) are assigned to treatment and N-m units (or clusters) are assigned to control. In a blocked design, exactly m units in each block will be treated. (optional)
m_unit	Use for a two-arm trial. Under complete random assignment, must be constant across units. Under blocked random assignment, must be constant within blocks.
m_each	Use for a multi-arm design in which the values of m_each determine the number of units (or clusters) assigned to each condition. m_each must be a numeric vector in which each entry is a nonnegative integer that describes how many units (or clusters) should be assigned to the 1st, 2nd, 3rd... treatment condition. m_each must sum to N. (optional)

`prob` Use for a two-arm design in which either floor(N*prob) or ceiling(N*prob) units (or clusters) are assigned to treatment. The probability of assignment to treatment is exactly prob because with probability 1-prob, floor(N*prob) units (or clusters) will be assigned to treatment and with probability prob, ceiling(N*prob) units (or clusters) will be assigned to treatment. prob must be a real number between 0 and 1 inclusive. (optional)

`prob_unit` Use for a two arm design. Must of be of length N. Under simple random assignment, can be different for each unit or cluster. Under complete random assignment, must be constant across units. Under blocked random assignment, must be constant within blocks.

`prob_each` Use for a multi-arm design in which the values of prob_each determine the probabilities of assignment to each treatment condition. prob_each must be a numeric vector giving the probability of assignment to each condition. All entries must be nonnegative real numbers between 0 and 1 inclusive and the total must sum to 1. Because of integer issues, the exact number of units assigned to each condition may differ (slightly) from assignment to assignment, but the overall probability of assignment is exactly prob_each. (optional)

`block_m` Use for a two-arm design in which block_m describes the number of units to assign to treatment within each block. Note that in previous versions of randomizr, block_m behaved like block_m_each.

`block_m_each` Use for a multi-arm design in which the values of block_m_each determine the number of units (or clusters) assigned to each condition. block_m_each must be a matrix with the same number of rows as blocks and the same number of columns as treatment arms. Cell entries are the number of units (or clusters) to be assigned to each treatment arm within each block. The rows should respect the ordering of the blocks as determined by sort(unique(blocks)). The columns should be in the order of conditions, if specified.

`block_prob` Use for a two-arm design in which block_prob describes the probability of assignment to treatment within each block. Differs from prob in that the probability of assignment can vary across blocks.

`block_prob_each`
Use for a multi-arm design in which the values of block_prob_each determine the probabilities of assignment to each treatment condition. block_prob_each must be a matrix with the same number of rows as blocks and the same number of columns as treatment arms. Cell entries are the probabilities of assignment to treatment within each block. The rows should respect the ordering of the blocks as determined by sort(unique(blocks)). Use only if the probabilities of assignment should vary by block, otherwise use prob_each. Each row of block_prob_each must sum to 1.

`num_arms` The number of treatment arms. If unspecified, num_arms will be determined from the other arguments. (optional)

`conditions` A character vector giving the names of the treatment groups. If unspecified, the treatment groups will be named 0 (for control) and 1 (for treatment) in a two-arm trial and T1, T2, T3, in a multi-arm trial. An exception is a two-group design in which num_arms is set to 2, in which case the condition names are T1 and T2, as in a multi-arm trial with two arms. (optional)

simple — logical, defaults to FALSE. If TRUE, simple random assignment is used. When simple = TRUE, please do not specify m, m_each, block_m, or block_m_each. If simple = TRUE, prob and prob_each may vary by unit.

permutation_matrix — for custom random assignment procedures.

check_inputs — logical. Defaults to TRUE.

Value

A list of class "declaration". The list has five entries: $ra_function, a function that generates random assignments according to the declaration. $ra_type, a string indicating the type of random assignment used $probabilities_matrix, a matrix with N rows and num_arms columns, describing each unit's probabilities of assignment to conditions. $blocks, the blocking variable. $clusters, the clustering variable.

Examples

```
# The declare_ra function is used in three ways:

# 1. To obtain some basic facts about a randomization:
declaration <- declare_ra(N=100, m_each=c(30, 30, 40))
declaration

# 2. To conduct a random assignment:

Z <- conduct_ra(declaration)
table(Z)

# 3. To obtain observed condition probabilities

probs <- obtain_condition_probabilities(declaration, Z)
table(probs, Z)

# Simple Random Assignment Declarations

declare_ra(N=100, simple = TRUE)
declare_ra(N=100, prob = .4, simple = TRUE)
declare_ra(N=100, prob_each=c(0.3, 0.3, 0.4),
           conditions=c("control", "placebo", "treatment"), simple=TRUE)

# Complete Random Assignment Declarations

declare_ra(N=100)
declare_ra(N=100, m_each = c(30, 70),
           conditions = c("control", "treatment"))
declare_ra(N=100, m_each=c(30, 30, 40))


# Block Random Assignment Declarations

blocks <- rep(c("A", "B","C"), times = c(50, 100, 200))
```

```
block_m_each <- rbind(c(10, 40),
                      c(30, 70),
                      c(50, 150))
declare_ra(blocks = blocks, block_m_each = block_m_each)


# Cluster Random Assignment Declarations

clusters <- rep(letters, times = 1:26)
declare_ra(clusters = clusters)
declare_ra(clusters = clusters, m_each = c(7, 7, 12))

# Blocked and Clustered Random Assignment Declarations

clusters <- rep(letters, times=1:26)
blocks <- rep(NA, length(clusters))
blocks[clusters %in% letters[1:5]] <- "block_1"
blocks[clusters %in% letters[6:10]] <- "block_2"
blocks[clusters %in% letters[11:15]] <- "block_3"
blocks[clusters %in% letters[16:20]] <- "block_4"
blocks[clusters %in% letters[21:26]] <- "block_5"

table(blocks, clusters)

declare_ra(clusters = clusters, blocks = blocks)
declare_ra(clusters = clusters, blocks = blocks, prob_each = c(.2, .5, .3))
```

`declare_rs` *Declare a random sampling procedure.*

Description

Declare a random sampling procedure.

Usage

```
declare_rs(
  N = NULL,
  strata = NULL,
  clusters = NULL,
  n = NULL,
  n_unit = NULL,
  prob = NULL,
  prob_unit = NULL,
  strata_n = NULL,
  strata_prob = NULL,
  simple = FALSE,
```

```
  check_inputs = TRUE
)
```

Arguments

N	The number of units. N must be a positive integer. (required)
strata	A vector of length N that indicates which stratum each unit belongs to.
clusters	A vector of length N that indicates which cluster each unit belongs to.
n	Use for a design in which n units (or clusters) are sampled. In a stratified design, exactly n units in each stratum will be sampled. (optional)
n_unit	Under complete random sampling, must be constant across units. Under stratified random sampling, must be constant within strata.
prob	Use for a design in which either floor(N*prob) or ceiling(N*prob) units (or clusters) are sampled. The probability of being sampled is exactly prob because with probability 1-prob, floor(N*prob) units (or clusters) will be sampled and with probability prob, ceiling(N*prob) units (or clusters) will be sampled. prob must be a real number between 0 and 1 inclusive. (optional)
prob_unit	Must of be of length N. Under simple random sampling, can be different for each unit or cluster. Under complete random sampling, must be constant across units. Under stratified random sampling, must be constant within strata.
strata_n	Use for a design in which strata_n describes the number of units to sample within each stratum.
strata_prob	Use for a design in which strata_prob describes the probability of being sampled within each stratum. Differs from prob in that the probability of being sampled can vary across strata.
simple	logical, defaults to FALSE. If TRUE, simple random sampling is used. When `simple = TRUE`, please do not specify n or strata_n. When `simple = TRUE`, prob may vary by unit.
check_inputs	logical. Defaults to TRUE.

Value

A list of class "declaration". The list has five entries: $rs_function, a function that generates random samplings according to the declaration. $rs_type, a string indicating the type of random sampling used $probabilities_vector, A vector length N indicating the probability of being sampled. $strata, the stratification variable. $clusters, the clustering variable.

Examples

```
# The declare_rs function is used in three ways:

# 1. To obtain some basic facts about a sampling procedure:
declaration <- declare_rs(N = 100, n = 30)
declaration

# 2. To draw a random sample:
```

```
S <- draw_rs(declaration)
table(S)

# 3. To obtain inclusion probabilities

probs <- obtain_inclusion_probabilities(declaration)
table(probs, S)

# Simple Random Sampling Declarations

declare_rs(N = 100, simple = TRUE)
declare_rs(N = 100, prob = .4, simple = TRUE)

# Complete Random Sampling Declarations

declare_rs(N = 100)
declare_rs(N = 100, n = 30)

# Stratified Random Sampling Declarations

strata <- rep(c("A", "B","C"), times=c(50, 100, 200))
declare_rs(strata = strata)
declare_rs(strata = strata, prob = .5)


# Cluster Random Sampling Declarations

clusters <- rep(letters, times = 1:26)
declare_rs(clusters = clusters)
declare_rs(clusters = clusters, n = 10)

# Stratified and Clustered Random Sampling Declarations

clusters <- rep(letters, times = 1:26)
strata <- rep(NA, length(clusters))
strata[clusters %in% letters[1:5]] <- "stratum_1"
strata[clusters %in% letters[6:10]] <- "stratum_2"
strata[clusters %in% letters[11:15]] <- "stratum_3"
strata[clusters %in% letters[16:20]] <- "stratum_4"
strata[clusters %in% letters[21:26]] <- "stratum_5"

table(strata, clusters)

declare_rs(clusters = clusters, strata = strata)
declare_rs(clusters = clusters, strata = strata, prob = .3)
```

`draw_rs` *Draw a random sample*

Description

You can either give draw_rs() an declaration, as created by declare_rs or you can specify the other arguments to describe a random sampling procedure.

Usage

```
draw_rs(
  declaration = NULL,
  N = NULL,
  strata = NULL,
  clusters = NULL,
  n = NULL,
  n_unit = NULL,
  prob = NULL,
  prob_unit = NULL,
  strata_n = NULL,
  strata_prob = NULL,
  simple = FALSE,
  check_inputs = TRUE
)
```

Arguments

declaration	A random sampling declaration, created by declare_rs.
N	The number of units. N must be a positive integer. (required)
strata	A vector of length N that indicates which stratum each unit belongs to.
clusters	A vector of length N that indicates which cluster each unit belongs to.
n	Use for a design in which n units (or clusters) are sampled. In a stratified design, exactly n units in each stratum will be sampled. (optional)
n_unit	Under complete random sampling, must be constant across units. Under stratified random sampling, must be constant within strata.
prob	Use for a design in which either floor(N*prob) or ceiling(N*prob) units (or clusters) are sampled. The probability of being sampled is exactly prob because with probability 1-prob, floor(N*prob) units (or clusters) will be sampled and with probability prob, ceiling(N*prob) units (or clusters) will be sampled. prob must be a real number between 0 and 1 inclusive. (optional)
prob_unit	Must of be of length N. Under simple random sampling, can be different for each unit or cluster. Under complete random sampling, must be constant across units. Under stratified random sampling, must be constant within strata.
strata_n	Use for a design in which strata_n describes the number of units to sample within each stratum.
strata_prob	Use for a design in which strata_prob describes the probability of being sampled within each stratum. Differs from prob in that the probability of being sampled can vary across strata.

simple	logical, defaults to FALSE. If TRUE, simple random sampling is used. When simple = TRUE, please do not specify n or strata_n. When simple = TRUE, prob may vary by unit.
check_inputs	logical. Defaults to TRUE.

Examples

```
declaration <- declare_rs(N = 100, n = 30)
S <- draw_rs(declaration = declaration)
table(S)

# equivalent to
S <- draw_rs(N = 100, n = 30)
table(S)
```

`obtain_condition_probabilities` *Obtain the probabilities of units being in the conditions that they are in.*

Description

You can either give obtain_condition_probabilities() an declaration, as created by declare_ra or you can specify the other arguments to describe a random assignment procedure.

This function is especially useful when units have different probabilities of assignment and the analyst plans to use inverse-probability weights.

Usage

```
obtain_condition_probabilities(
  declaration = NULL,
  assignment,
  N = NULL,
  blocks = NULL,
  clusters = NULL,
  m = NULL,
  m_unit = NULL,
  m_each = NULL,
  prob = NULL,
  prob_unit = NULL,
  prob_each = NULL,
  block_m = NULL,
  block_m_each = NULL,
  block_prob = NULL,
  block_prob_each = NULL,
  num_arms = NULL,
```

```
  conditions = NULL,
  simple = FALSE,
  permutation_matrix = NULL,
  check_inputs = TRUE
)
```

Arguments

declaration	A random assignment declaration, created by declare_ra.
assignment	A vector of random assignments, often created by conduct_ra.
N	The number of units. N must be a positive integer. (required)
blocks	A vector of length N that indicates which block each unit belongs to.
clusters	A vector of length N that indicates which cluster each unit belongs to.
m	Use for a two-arm design in which m units (or clusters) are assigned to treatment and N-m units (or clusters) are assigned to control. In a blocked design, exactly m units in each block will be treated. (optional)
m_unit	Use for a two-arm trial. Under complete random assignment, must be constant across units. Under blocked random assignment, must be constant within blocks.
m_each	Use for a multi-arm design in which the values of m_each determine the number of units (or clusters) assigned to each condition. m_each must be a numeric vector in which each entry is a nonnegative integer that describes how many units (or clusters) should be assigned to the 1st, 2nd, 3rd... treatment condition. m_each must sum to N. (optional)
prob	Use for a two-arm design in which either floor(N*prob) or ceiling(N*prob) units (or clusters) are assigned to treatment. The probability of assignment to treatment is exactly prob because with probability 1-prob, floor(N*prob) units (or clusters) will be assigned to treatment and with probability prob, ceiling(N*prob) units (or clusters) will be assigned to treatment. prob must be a real number between 0 and 1 inclusive. (optional)
prob_unit	Use for a two arm design. Must of be of length N. Under simple random assignment, can be different for each unit or cluster. Under complete random assignment, must be constant across units. Under blocked random assignment, must be constant within blocks.
prob_each	Use for a multi-arm design in which the values of prob_each determine the probabilities of assignment to each treatment condition. prob_each must be a numeric vector giving the probability of assignment to each condition. All entries must be nonnegative real numbers between 0 and 1 inclusive and the total must sum to 1. Because of integer issues, the exact number of units assigned to each condition may differ (slightly) from assignment to assignment, but the overall probability of assignment is exactly prob_each. (optional)
block_m	Use for a two-arm design in which block_m describes the number of units to assign to treatment within each block. Note that in previous versions of randomizr, block_m behaved like block_m_each.

`block_m_each` Use for a multi-arm design in which the values of block_m_each determine the number of units (or clusters) assigned to each condition. block_m_each must be a matrix with the same number of rows as blocks and the same number of columns as treatment arms. Cell entries are the number of units (or clusters) to be assigned to each treatment arm within each block. The rows should respect the ordering of the blocks as determined by sort(unique(blocks)). The columns should be in the order of conditions, if specified.

`block_prob` Use for a two-arm design in which block_prob describes the probability of assignment to treatment within each block. Differs from prob in that the probability of assignment can vary across blocks.

`block_prob_each`
Use for a multi-arm design in which the values of block_prob_each determine the probabilities of assignment to each treatment condition. block_prob_each must be a matrix with the same number of rows as blocks and the same number of columns as treatment arms. Cell entries are the probabilities of assignment to treatment within each block. The rows should respect the ordering of the blocks as determined by sort(unique(blocks)). Use only if the probabilities of assignment should vary by block, otherwise use prob_each. Each row of block_prob_each must sum to 1.

`num_arms` The number of treatment arms. If unspecified, num_arms will be determined from the other arguments. (optional)

`conditions` A character vector giving the names of the treatment groups. If unspecified, the treatment groups will be named 0 (for control) and 1 (for treatment) in a two-arm trial and T1, T2, T3, in a multi-arm trial. An exception is a two-group design in which num_arms is set to 2, in which case the condition names are T1 and T2, as in a multi-arm trial with two arms. (optional)

`simple` logical, defaults to FALSE. If TRUE, simple random assignment is used. When `simple = TRUE`, please do not specify m, m_each, block_m, or block_m_each. If `simple = TRUE`, `prob` and `prob_each` may vary by unit.

`permutation_matrix`
for custom random assignment procedures.

`check_inputs` logical. Defaults to TRUE.

Examples

```
# Conduct a block random assignment
blocks <- rep(c("A", "B","C"), times=c(50, 100, 200))
block_m_each <- rbind(c(10, 40),
                c(30, 70),
                c(50, 150))
declaration <- declare_ra(blocks = blocks, block_m_each = block_m_each)
Z <- conduct_ra(declaration = declaration)
table(Z, blocks)

observed_probabilities <-
   obtain_condition_probabilities(declaration = declaration, assignment = Z)
```

```
# Probabilities in the control group:
table(observed_probabilities[Z == 0], blocks[Z == 0])

# Probabilities in the treatment group:
table(observed_probabilities[Z == 1], blocks[Z == 1])


# Sometimes it is convenient to skip the declaration step
Z <- conduct_ra(blocks = blocks, block_m_each = block_m_each)
observed_probabilities <-
   obtain_condition_probabilities(assignment = Z,
                                  blocks = blocks,
                                  block_m_each = block_m_each)
table(observed_probabilities[Z == 0], blocks[Z == 0])
table(observed_probabilities[Z == 1], blocks[Z == 1])
```

`obtain_inclusion_probabilities` *Obtain inclusion probabilities*

Description

You can either give obtain_inclusion_probabilities() an declaration, as created by `declare_rs` or you can specify the other arguments to describe a random sampling procedure.

This function is especially useful when units have different inclusion probabilities and the analyst plans to use inverse-probability weights.

Usage

```
obtain_inclusion_probabilities(
  declaration = NULL,
  N = NULL,
  strata = NULL,
  clusters = NULL,
  n = NULL,
  n_unit = NULL,
  prob = NULL,
  prob_unit = NULL,
  strata_n = NULL,
  strata_prob = NULL,
  simple = FALSE,
  check_inputs = TRUE
)
```

Arguments

declaration	A random sampling declaration, created by declare_rs.
N	The number of units. N must be a positive integer. (required)
strata	A vector of length N that indicates which stratum each unit belongs to.
clusters	A vector of length N that indicates which cluster each unit belongs to.
n	Use for a design in which n units (or clusters) are sampled. In a stratified design, exactly n units in each stratum will be sampled. (optional)
n_unit	Under complete random sampling, must be constant across units. Under stratified random sampling, must be constant within strata.
prob	Use for a design in which either floor(N*prob) or ceiling(N*prob) units (or clusters) are sampled. The probability of being sampled is exactly prob because with probability 1-prob, floor(N*prob) units (or clusters) will be sampled and with probability prob, ceiling(N*prob) units (or clusters) will be sampled. prob must be a real number between 0 and 1 inclusive. (optional)
prob_unit	Must of be of length N. Under simple random sampling, can be different for each unit or cluster. Under complete random sampling, must be constant across units. Under stratified random sampling, must be constant within strata.
strata_n	Use for a design in which strata_n describes the number of units to sample within each stratum.
strata_prob	Use for a design in which strata_prob describes the probability of being sampled within each stratum. Differs from prob in that the probability of being sampled can vary across strata.
simple	logical, defaults to FALSE. If TRUE, simple random sampling is used. When simple = TRUE, please do not specify n or strata_n. When simple = TRUE, prob may vary by unit.
check_inputs	logical. Defaults to TRUE.

Examples

```
# Draw a stratified random sample
strata <- rep(c("A", "B","C"), times=c(50, 100, 200))

declaration <- declare_rs(strata = strata)

observed_probabilities <-
   obtain_inclusion_probabilities(declaration = declaration)

table(strata, observed_probabilities)


# Sometimes it is convenient to skip the declaration step
observed_probabilities <-
   obtain_inclusion_probabilities(strata = strata)

table(strata, observed_probabilities)
```

obtain_num_permutations *Obtain the Number of Possible Permutations from a Random Assignment Declaration*

Description

Obtain the Number of Possible Permutations from a Random Assignment Declaration

Usage

```
obtain_num_permutations(declaration)
```

Arguments

declaration	A random assignment or sampling declaration, created by declare_ra or declare_rs.

Value

a scalar

Examples

```
# Random assignment
## complete

declaration <- declare_ra(N = 4)
perms <- obtain_permutation_matrix(declaration)
dim(perms)
obtain_num_permutations(declaration)

## blocked

blocks <- c("A", "A", "B", "B", "C", "C", "C")
declaration <- declare_ra(blocks = blocks)
perms <- obtain_permutation_matrix(declaration)
dim(perms)
obtain_num_permutations(declaration)

## clustered

clusters <- c("A", "B", "A", "B", "C", "C", "C")
declaration <- declare_ra(clusters = clusters)
perms <- obtain_permutation_matrix(declaration)
dim(perms)
obtain_num_permutations(declaration)

## large

declaration <- declare_ra(20)
```

```
choose(20, 10)
perms <- obtain_permutation_matrix(declaration)
dim(perms)

# Random sampling
## complete

declaration <- declare_rs(N = 4)
perms <- obtain_permutation_matrix(declaration)
dim(perms)
obtain_num_permutations(declaration)

## stratified

strata <- c("A", "A", "B", "B", "C", "C", "C")
declaration <- declare_rs(strata = strata)
perms <- obtain_permutation_matrix(declaration)
dim(perms)
obtain_num_permutations(declaration)

## clustered

clusters <- c("A", "B", "A", "B", "C", "C", "C")
declaration <- declare_rs(clusters = clusters)
perms <- obtain_permutation_matrix(declaration)
dim(perms)
obtain_num_permutations(declaration)

## large

declaration <- declare_rs(N = 20)
perms <- obtain_permutation_matrix(declaration)
dim(perms)
```

`obtain_permutation_matrix` *Obtain Permutation Matrix from a Random Assignment Declaration*

Description

Obtain Permutation Matrix from a Random Assignment Declaration

Usage

```
obtain_permutation_matrix(declaration, maximum_permutations = 10000)
```

Arguments

declaration	A random assignment declaration, created by declare_ra.
maximum_permutations	If the number of possible random assignments exceeds maximum_permutations, obtain_permutation_matrix will return a random sample of maximum_permutations permutations. Defaults to 10,000.

Value

a matrix of all possible (or a random sample of all possible) random assignments consistent with a declaration.

Examples

```
# complete

declaration <- declare_ra(N = 4)
perms <- obtain_permutation_matrix(declaration)
dim(perms)
obtain_num_permutations(declaration)

# blocked

blocks <- c("A", "A", "B", "B", "C", "C", "C")
declaration <- declare_ra(blocks = blocks)
perms <- obtain_permutation_matrix(declaration)
dim(perms)
obtain_num_permutations(declaration)

# clustered

clusters <- c("A", "B", "A", "B", "C", "C", "C")
declaration <- declare_ra(clusters = clusters)
perms <- obtain_permutation_matrix(declaration)
dim(perms)
obtain_num_permutations(declaration)

# large

declaration <- declare_ra(20)
choose(20, 10)
perms <- obtain_permutation_matrix(declaration)
dim(perms)
```

`obtain_permutation_probabilities` *Obtain the probabilities of permutations*

Description

Obtain the probabilities of permutations

Usage

```
obtain_permutation_probabilities(declaration)
```

Arguments

`declaration`	A random assignment declaration, created by `declare_ra`.

Value

a vector of probabilities

Examples

```
declaration <- declare_ra(N = 5, prob_each = c(.49, .51))
obtain_num_permutations(declaration)
perm_probs <- obtain_permutation_probabilities(declaration)
perms <- obtain_permutation_matrix(declaration)

# probabilities of assignment from declaration *should* match the average over all permutations
true_probabilities <- declaration$probabilities_matrix[,2]
true_probabilities

# correctly WRONG because the perms have different probs!
rowMeans(perms)

# correctly correct!
perms %*% perm_probs
```

`randomizr` *randomizr*

Description

Easy-to-Use Tools for Common Forms of Random Assignment and Sampling

simple_ra *Simple Random Assignment*

Description

simple_ra implements a random assignment procedure in which units are independently assigned to treatment conditions. Because units are assigned independently, the number of units that are assigned to each condition can vary from assignment to assignment. For most experimental applications in which the number of experimental units is known in advance, complete_ra is better because the number of units assigned to each condition is fixed across assignments.

In most cases, users should specify N and not more than one of prob, prob_each, or num_arms.

If only N is specified, a two-arm trial with prob = 0.5 is assumed.

Usage

```
simple_ra(
  N,
  prob = NULL,
  prob_unit = NULL,
  prob_each = NULL,
  num_arms = NULL,
  conditions = NULL,
  check_inputs = TRUE,
  simple = TRUE
)
```

Arguments

N	The number of units. N must be a positive integer. (required)
prob	Use for a two-arm design. prob is the probability of assignment to treatment and must be a real number between 0 and 1 inclusive and must be length 1. (optional)
prob_unit	Use for a two-arm design. prob is the probability of assignment to treatment and must be a real number between 0 and 1 inclusive and must be length N. (optional)
prob_each	Use for a multi-arm design in which the values of prob_each determine the probabilities of assignment to each treatment condition. prob_each must be a numeric vector giving the probability of assignment to each condition. All entries must be nonnegative real numbers between 0 and 1 inclusive and the total must sum to 1. It may be a conditions-length vector or a N-by-conditions matrix. (optional)
num_arms	The number of treatment arms. If unspecified, num_arms will be determined from the other arguments. (optional)

conditions	A character vector giving the names of the treatment groups. If unspecified, the treatment groups will be named 0 (for control) and 1 (for treatment) in a two-arm trial and T1, T2, T3, in a multi-arm trial. An exception is a two-group design in which num_arms is set to 2, in which case the condition names are T1 and T2, as in a multi-arm trial with two arms. (optional)
check_inputs	logical. Defaults to TRUE.
simple	logical. internal use only.

Value

A vector of length N that indicates the treatment condition of each unit. Is numeric in a two-arm trial and a factor variable (ordered by conditions) in a multi-arm trial.

Examples

```
# Two Group Designs

Z <- simple_ra(N=100)
table(Z)

Z <- simple_ra(N=100, prob=0.5)
table(Z)

Z <- simple_ra(N=100, prob_each = c(0.3, 0.7),
               conditions = c("control", "treatment"))
table(Z)

# Multi-arm Designs
Z <- simple_ra(N=100, num_arms=3)
table(Z)

Z <- simple_ra(N=100, prob_each=c(0.3, 0.3, 0.4))
table(Z)

Z <- simple_ra(N=100, prob_each=c(0.3, 0.3, 0.4),
               conditions=c("control", "placebo", "treatment"))
table(Z)

Z <- simple_ra(N=100, conditions=c("control", "placebo", "treatment"))
table(Z)
```

---

simple_ra_probabilities

probabilities of assignment: Simple Random Assignment

Description

probabilities of assignment: Simple Random Assignment

Usage

```
simple_ra_probabilities(
  N,
  prob = NULL,
  prob_unit = NULL,
  prob_each = NULL,
  num_arms = NULL,
  conditions = NULL,
  check_inputs = TRUE,
  simple = TRUE
)
```

Arguments

N	The number of units. N must be a positive integer. (required)
prob	Use for a two-arm design. prob is the probability of assignment to treatment and must be a real number between 0 and 1 inclusive and must be length 1. (optional)
prob_unit	Use for a two-arm design. prob is the probability of assignment to treatment and must be a real number between 0 and 1 inclusive and must be length N. (optional)
prob_each	Use for a multi-arm design in which the values of prob_each determine the probabilities of assignment to each treatment condition. prob_each must be a numeric vector giving the probability of assignment to each condition. All entries must be nonnegative real numbers between 0 and 1 inclusive and the total must sum to 1. It may be a conditions-length vector or a N-by-conditions matrix. (optional)
num_arms	The number of treatment arms. If unspecified, num_arms will be determined from the other arguments. (optional)
conditions	A character vector giving the names of the treatment groups. If unspecified, the treatment groups will be named 0 (for control) and 1 (for treatment) in a two-arm trial and T1, T2, T3, in a multi-arm trial. An exception is a two-group design in which num_arms is set to 2, in which case the condition names are T1 and T2, as in a multi-arm trial with two arms. (optional)
check_inputs	logical. Defaults to TRUE.
simple	logical. internal use only.

Value

A matrix of probabilities of assignment

Examples

```
# Two Group Designs
prob_mat <- simple_ra_probabilities(N=100)
head(prob_mat)
```

```
prob_mat <- simple_ra_probabilities(N=100, prob=0.5)
head(prob_mat)

prob_mat <- simple_ra_probabilities(N=100, prob_each = c(0.3, 0.7),
                        conditions = c("control", "treatment"))
head(prob_mat)

# Multi-arm Designs
prob_mat <- simple_ra_probabilities(N=100, num_arms=3)
head(prob_mat)

prob_mat <- simple_ra_probabilities(N=100, prob_each=c(0.3, 0.3, 0.4))
head(prob_mat)

prob_mat <- simple_ra_probabilities(N=100, prob_each=c(0.3, 0.3, 0.4),
                        conditions=c("control", "placebo", "treatment"))
head(prob_mat)

prob_mat <- simple_ra_probabilities(N=100, conditions=c("control", "placebo", "treatment"))
head(prob_mat)
```

`simple_rs` *Simple Random Sampling*

Description

simple_rs implements a random sampling procedure in which units are independently sampled. Because units are sampled independently, the number of units that are sampled can vary from sample to sample. For most applications in which the number of units in the sampling frame is known in advance, `complete_rs` is better because the number of units sampled is fixed across sampled.

Usage

```
simple_rs(N, prob = NULL, prob_unit = NULL, check_inputs = TRUE, simple = TRUE)
```

Arguments

`N`	The number of units. N must be a positive integer. (required)
`prob`	prob is the probability of being sampled must be a real number between 0 and 1 inclusive, and must be of length 1. (optional)
`prob_unit`	prob is the probability of being sampled must be a real number between 0 and 1 inclusive, and must be of length N. (optional)
`check_inputs`	logical. Defaults to TRUE.
`simple`	logical. internal use only.

Value

A numeric vector of length N that indicates if a unit is sampled (1) or not (0).

Examples

```
S <- simple_rs(N = 100)
table(S)

S <- simple_rs(N = 100, prob = 0.3)
table(S)
```

---

simple_rs_probabilities

Inclusion Probabilities: Simple Random Sampling

Description

Inclusion Probabilities: Simple Random Sampling

Usage

```
simple_rs_probabilities(
  N,
  prob = NULL,
  prob_unit = NULL,
  check_inputs = TRUE,
  simple = TRUE
)
```

Arguments

N	The number of units. N must be a positive integer. (required)
prob	prob is the probability of being sampled must be a real number between 0 and 1 inclusive, and must be of length 1. (optional)
prob_unit	prob is the probability of being sampled must be a real number between 0 and 1 inclusive, and must be of length N. (optional)
check_inputs	logical. Defaults to TRUE.
simple	logical. internal use only.

Value

A vector length N indicating the probability of being sampled.

Examples

```
probs <- simple_ra_probabilities(N = 100)
table(probs)

probs <- simple_ra_probabilities(N = 100, prob = 0.3)
table(probs)
```

---

strata_and_cluster_rs *Stratified and Clustered Random Sampling*

Description

A random sampling procedure in which units are sampled as clusters and clusters are nested within strata.

Usage

```
strata_and_cluster_rs(
  strata = NULL,
  clusters = NULL,
  prob = NULL,
  prob_unit = NULL,
  n = NULL,
  n_unit = NULL,
  strata_n = NULL,
  strata_prob = NULL,
  check_inputs = TRUE
)
```

Arguments

strata	A vector of length N that indicates which stratum each unit belongs to.
clusters	A vector of length N that indicates which cluster each unit belongs to.
prob	Use for a design in which either floor(N_clusters_stratum*prob) or ceiling(N_clusters_stratum*prob) clusters are sampled within each stratum. The probability of being sampled is exactly prob because with probability 1-prob, floor(N_clusters_stratum*prob) clusters will be sampled and with probability prob, ceiling(N_clusters_stratum*prob) clusters will be sampled. prob must be a real number between 0 and 1 inclusive. (optional)
prob_unit	Must of be of length N. tapply(prob_unit, blocks, unique) will be passed to strata_prob.
n	Use for a design in which the scalar n describes the fixed number of units to sample in each stratum. This number does not vary across strata.
n_unit	Must be of length N. tapply(m_unit, blocks, unique) will be passed to strata_n.

strata_n	Use for a design in which strata_n describes the number of units to sample within each stratum.
strata_prob	Use for a design in which strata_prob describes the probability of being sampled within each stratum. Differs from prob in that the probability of being sampled can vary across strata.
check_inputs	logical. Defaults to TRUE.

Value

A numeric vector of length N that indicates if a unit is sampled (1) or not (0).

Examples

```
clusters <- rep(letters, times = 1:26)

strata <- rep(NA, length(clusters))
strata[clusters %in% letters[1:5]] <- "stratum_1"
strata[clusters %in% letters[6:10]] <- "stratum_2"
strata[clusters %in% letters[11:15]] <- "stratum_3"
strata[clusters %in% letters[16:20]] <- "stratum_4"
strata[clusters %in% letters[21:26]] <- "stratum_5"

table(strata, clusters)

S <- strata_and_cluster_rs(strata = strata,
                           clusters = clusters)

table(S, strata)
table(S, clusters)


S <- strata_and_cluster_rs(clusters = clusters,
                           strata = strata,
                           prob = .5)

table(S, clusters)
table(S, strata)

S <- strata_and_cluster_rs(clusters = clusters,
                           strata = strata,
                           strata_n = c(2, 3, 2, 3, 2))

table(S, clusters)
table(S, strata)

S <- strata_and_cluster_rs(clusters = clusters,
                           strata = strata,
                           strata_prob = c(.1, .2, .3, .4, .5))

table(S, clusters)
table(S, strata)
```

`strata_and_cluster_rs_probabilities` *Inclusion Probabilities: Stratified and Clustered Random Sampling*

Description

Inclusion Probabilities: Stratified and Clustered Random Sampling

Usage

```
strata_and_cluster_rs_probabilities(
  strata = NULL,
  clusters = NULL,
  prob = NULL,
  prob_unit = NULL,
  n = NULL,
  n_unit = NULL,
  strata_n = NULL,
  strata_prob = NULL,
  check_inputs = TRUE
)
```

Arguments

`strata`	A vector of length N that indicates which stratum each unit belongs to.
`clusters`	A vector of length N that indicates which cluster each unit belongs to.
`prob`	Use for a design in which either floor(N_clusters_stratum*prob) or ceiling(N_clusters_stratum*prob) clusters are sampled within each stratum. The probability of being sampled is exactly prob because with probability 1-prob, floor(N_clusters_stratum*prob) clusters will be sampled and with probability prob, ceiling(N_clusters_stratum*prob) clusters will be sampled. prob must be a real number between 0 and 1 inclusive. (optional)
`prob_unit`	Must of be of length N. tapply(prob_unit, blocks, unique) will be passed to `strata_prob`.
`n`	Use for a design in which the scalar n describes the fixed number of units to sample in each stratum. This number does not vary across strata.
`n_unit`	Must be of length N. tapply(m_unit, blocks, unique) will be passed to `strata_n`.
`strata_n`	Use for a design in which strata_n describes the number of units to sample within each stratum.
`strata_prob`	Use for a design in which strata_prob describes the probability of being sampled within each stratum. Differs from prob in that the probability of being sampled can vary across strata.
`check_inputs`	logical. Defaults to TRUE.

Value

A vector length N indicating the probability of being sampled.

Examples

```
clusters <- rep(letters, times = 1:26)

strata <- rep(NA, length(clusters))
strata[clusters %in% letters[1:5]] <- "stratum_1"
strata[clusters %in% letters[6:10]] <- "stratum_2"
strata[clusters %in% letters[11:15]] <- "stratum_3"
strata[clusters %in% letters[16:20]] <- "stratum_4"
strata[clusters %in% letters[21:26]] <- "stratum_5"

table(strata, clusters)

probs <- strata_and_cluster_rs_probabilities(strata = strata,
                                             clusters = clusters)

table(probs, strata)
table(probs, clusters)


probs <- strata_and_cluster_rs_probabilities(clusters = clusters,
                                             strata = strata,
                                             prob = .5)

table(probs, clusters)
table(probs, strata)

probs <- strata_and_cluster_rs_probabilities(clusters = clusters,
                                             strata = strata,
                                             strata_n = c(2, 3, 2, 3, 2))

table(probs, clusters)
table(probs, strata)

probs <- strata_and_cluster_rs_probabilities(clusters = clusters,
                                             strata = strata,
                                             strata_prob = c(.1, .2, .3, .4, .5))

table(probs, clusters)
table(probs, strata)
```

`strata_rs` *Stratified Random Sampling*

Description

strata_rs implements a random sampling procedure in which units that are grouped into strata defined by covariates are sample using complete random sampling within stratum For example, imagine that 50 of 100 men are sampled and 75 of 200 women are sampled.

Usage

```
strata_rs(
  strata = NULL,
  prob = NULL,
  prob_unit = NULL,
  n = NULL,
  n_unit = NULL,
  strata_n = NULL,
  strata_prob = NULL,
  check_inputs = TRUE
)
```

Arguments

strata	A vector of length N that indicates which stratum each unit belongs to. Can be a character, factor, or numeric vector. (required)
prob	Use for a design in which either floor(N_stratum*prob) or ceiling(N_stratum*prob) units are sampled within each stratum. The probability of being sampled is exactly prob because with probability 1-prob, floor(N_stratum*prob) units will be sampled and with probability prob, ceiling(N_stratum*prob) units will be sampled. prob must be a real number between 0 and 1 inclusive. (optional)
prob_unit	Must of be of length N. tapply(prob_unit, strata, unique) will be passed to strata_prob.
n	Use for a design in which the scalar n describes the fixed number of units to sample in each stratum. This number does not vary across strata.
n_unit	Must be of length N. tapply(m_unit, strata, unique) will be passed to strata_n.
strata_n	Use for a design in which the numeric vector strata_n describes the number of units to sample within each stratum.
strata_prob	Use for a design in which strata_prob describes the probability of being sampled within each stratum. Differs from prob in that the probability of being sampled can vary across strata.
check_inputs	logical. Defaults to TRUE.

Value

A numeric vector of length N that indicates if a unit is sampled (1) or not (0).

Examples

```
strata <- rep(c("A", "B","C"), times = c(50, 100, 200))
Z <- strata_rs(strata = strata)
```

```
table(strata, Z)

Z <- strata_rs(strata = strata, prob = .3)
table(strata, Z)

Z <- strata_rs(strata = strata, n = 20)
table(strata, Z)

Z <- strata_rs(strata = strata, strata_prob = c(.1, .2, .3))
table(strata, Z)

Z <- strata_rs(strata = strata,
               prob_unit = rep(c(.1, .2, .3), times = c(50, 100, 200)))
table(strata, Z)

Z <- strata_rs(strata = strata, strata_n = c(20, 30, 40))
table(strata, Z)

Z <- strata_rs(strata = strata,
               n_unit = rep(c(20, 30, 40), times = c(50, 100, 200)))
table(strata, Z)
```

`strata_rs_probabilities` *Inclusion Probabilities: Stratified Random Sampling*

Description

Inclusion Probabilities: Stratified Random Sampling

Usage

```
strata_rs_probabilities(
  strata = NULL,
  prob = NULL,
  prob_unit = NULL,
  n = NULL,
  n_unit = NULL,
  strata_n = NULL,
  strata_prob = NULL,
  check_inputs = TRUE
)
```

Arguments

`strata`	A vector of length N that indicates which stratum each unit belongs to. Can be a character, factor, or numeric vector. (required)

prob	Use for a design in which either floor(N_stratum*prob) or ceiling(N_stratum*prob) units are sampled within each stratum. The probability of being sampled is exactly prob because with probability 1-prob, floor(N_stratum*prob) units will be sampled and with probability prob, ceiling(N_stratum*prob) units will be sampled. prob must be a real number between 0 and 1 inclusive. (optional)
prob_unit	Must of be of length N. tapply(prob_unit, strata, unique) will be passed to strata_prob.
n	Use for a design in which the scalar n describes the fixed number of units to sample in each stratum. This number does not vary across strata.
n_unit	Must be of length N. tapply(m_unit, strata, unique) will be passed to strata_n.
strata_n	Use for a design in which the numeric vector strata_n describes the number of units to sample within each stratum.
strata_prob	Use for a design in which strata_prob describes the probability of being sampled within each stratum. Differs from prob in that the probability of being sampled can vary across strata.
check_inputs	logical. Defaults to TRUE.

Value

A vector length N indicating the probability of being sampled.

Examples

```
strata <- rep(c("A", "B","C"), times = c(50, 100, 200))
probs <- strata_rs_probabilities(strata = strata)
table(strata, probs)

probs <- strata_rs_probabilities(strata = strata, prob = .2)
table(strata, probs)

probs <- strata_rs_probabilities(strata = strata, strata_prob = c(.1, .2, .3))
table(strata, probs)

probs <- strata_rs_probabilities(strata = strata, strata_n = c(10, 40, 70))
table(strata, probs)
```

Index

block_and_cluster_ra, 2
block_and_cluster_ra_probabilities, 5
block_ra, 8
block_ra_probabilities, 11

cluster_ra, 13
cluster_ra_probabilities, 15
cluster_rs, 17
cluster_rs_probabilities, 19
complete_ra, 20, *47*
complete_ra_probabilities, 22
complete_rs, 24, *50*
complete_rs_probabilities, 26
conduct_ra, 27, *39*
custom_ra, 29
custom_ra_probabilities, 30

declare_ra, *27, 28*, 31, *38, 39*, *43*, *45, 46*
declare_rs, 34, *37*, *41–43*
draw_rs, 36

obtain_condition_probabilities, 38
obtain_inclusion_probabilities, 41
obtain_num_permutations, 43
obtain_permutation_matrix, 44
obtain_permutation_probabilities, 46

randomizr, 46

simple_ra, *13*, 47
simple_ra_probabilities, 48
simple_rs, *18*, 50
simple_rs_probabilities, 51
strata_and_cluster_rs, 52
strata_and_cluster_rs_probabilities, 54
strata_rs, 55
strata_rs_probabilities, 57