Package ‘rakeR’

October 14, 2022

Title Easy Spatial Microsimulation (Raking) in R
Version 0.2.1
Date 2017-10-10
Description Functions for performing spatial microsimulation (‘raking’) in R.
Depends R (>= 3.4.0)
License GPL-3
Encoding UTF-8
LazyData true
RoxygenNote 6.0.1
Imports ipfp, wrswoR
Suggests testthat, readr
URL https://philmikejones.github.io/rakeR/
BugReports https://github.com/philmikejones/rakeR/issues
NeedsCompilation no
Author Phil Mike Jones [aut, cre] (<https://orcid.org/0000-0001-5173-3245>), Robin Lovelace [aut] (Many functions are based on code by Robin Lovelace and Morgane Dumont), Morgane Dumont [aut] (Many functions are based on code by Robin Lovelace and Morgane Dumont), Andrew Smith [ctb]
Maintainer Phil Mike Jones <philmikejones@gmail.com>
Repository CRAN
Date/Publication 2017-10-10 13:44:46 UTC

R topics documented:

check_constraint .......................................................... 2
check_ind ................................................................. 3
check_constraint

Description
Checks a constraint table for common errors.

Usage
check_constraint(constraint_var, num_zones)

Arguments
- constraint_var: The constraint table to check, usually a data frame
- num_zones: The number of zones that should be present in the table

Details
Checks a constraint table for the following common errors:
- Ensures all zone codes are unique
- Ensures there are the expected number of zones
- Ensures all but the zone column are numeric (integer or double)

Value
If no errors are detected the function returns silently. Any errors will stop the function or script to be investigated.

Examples
cons <- data.frame(
    "zone" = letters[1:3],
    "age_0_49" = c(8, 2, 7),
    "age_gt_50" = c(4, 8, 4),
    "sex_f" = c(6, 6, 8),
    "sex_m" = c(6, 4, 3)
)
check_constraint(cons, 3) # no errors
Description
Checks an individual (survey) variable for common errors.

Usage
check_ind(ind_var)

Arguments
ind_var the individual (survey) variable you want to check

Details
Checks an individual (survey) variable for the following common errors:
• That each row sums to 1 (i.e. correctly converted to a dummy variable)

Value
If no errors are detected the function returns silently. Any errors will stop the function or script to be investigated.

Examples
## check_ind(ind_var)

Description
Extract aggregate weights from individual weight table

Usage
extract(weights, inds, id)

Arguments
weights A weight table, typically produced by rakeR::weight()
inds The individual level data
id The unique id variable in the individual level data (inds), usually the first column
**Details**

Extract aggregate weights from individual weight table, typically produced by rakeR::weight(). Extract cannot operate with numeric variables because it creates a new variable for each unique factor of each variable. If you want numeric information, like income, use integerise() instead.

**Value**

A data frame with zones and aggregated simulated values for each variable

**Examples**

```r
## Not run
## Use weights object from weights()
## ext_weights <- extract(weights = weights, inds = inds, id = "id")
```

---

**Description**

Deprecated: use rakeR::extract()

**Usage**

```r
extract_weights(weights, inds, id)
```

**Arguments**

- `weights`: A weight table, typically produced by rakeR::weight()
- `inds`: The individual level data
- `id`: The unique id variable in the individual level data (inds), usually the first column

**Value**

A data frame with zones and aggregated simulated values for each variable

**Examples**

```r
## Not run
## extract_weights() is deprecated, use extract() instead
```
integerise

Description

Generate integer cases from numeric weights matrix.

Usage

integerise(weights, inds, method = "trs", seed = 42)

Arguments

weights
A matrix or data frame of fractional weights, typically provided by rakeR::weight()

inds
The individual–level data (i.e. one row per individual)

method
The integerisation method specified as a character string. Defaults to "trs"; currently other methods are not implemented.

seed
The seed to use, defaults to 42.

Details

Extracted weights (using rakeR::extract()) are more 'precise' than integerised weights (although the user should be careful this is not spurious precision based on context) as they return fractions. Nevertheless, integerised weights are useful in cases when:

- Numeric information (such as income) is required, as this needs to be cut() to work with rakeR::extract()
- Simulated 'individuals' are required for case studies of key areas.
- Input individual-level data for agent-based or dynamic models are required

The default integerisation method uses the 'truncate, replicate, sample' method developed by Robin Lovelace and Dimitris Ballas http://www.sciencedirect.com/science/article/pii/S0198971513000240

Other methods (for example proportional probabilities) may be implemented at a later date.

Value

A data frame of integerised cases

Examples

cons <- data.frame(
  "zone" = letters[1:3],
  "age_0_49" = c(8, 2, 7),
  "age_gt_50" = c(4, 8, 4),
  "sex_f" = c(6, 6, 8),
  "sex_m" = c(6, 4, 3),
stringsAsFactors = FALSE
inds <- data.frame(
  "id" = LETTERS[1:5],
  "age" = c("age_gt_50", "age_gt_50", "age_0_49", "age_gt_50", "age_0_49"),
  "sex" = c("sex_m", "sex_m", "sex_m", "sex_f", "sex_f"),
  "income" = c(2868, 2474, 2231, 3152, 2473),
  stringsAsFactors = FALSE
)
vars <- c("age", "sex")
weights <- weight(cons = cons, inds =_inds, vars = vars)
weights_int <- integerise(weights, inds = inds)

rake

Description

A convenience function wrapping weight() and extract() or weight() and integerise()

Usage

rake(cons, inds, vars, output = "fraction", iterations = 10, ...)

Arguments

cons A data frame of constraint variables
inds A data frame of individual–level (survey) data
vars A character string of variables to iterate over
output A string specifying the desired output, either "fraction" (extract()) or "integer" (integerise())
iterations The number of iterations to perform. Defaults to 10.
... Additional arguments to pass to depending on desired output:
  • if "fraction" specify 'id' (see extract() documentation)
  • if "integer" specify 'method' and 'seed' (see integerise() documentation)

Value

A data frame with extracted weights (if output == "fraction", the default) or integerised cases (if output == "integer")
## Examples

```r
## not run
## frac_weights <- rake(cons, inds, vars, output = "fraction",
##                     id = "id")

## int_weight <- rake(cons, inds, vars, output = "integer",
##                     method = "trs", seed = "42")
```

### simulate

#### Description

Deprecated: integerise() %>% simulate() has been replaced by simply integerise() to be consistent with extract().

#### Usage

`simulate(...)`

#### Arguments

... arguments previously passed to `simulate()`

#### Value

Returns an error if used. Just use integerise()

### weight

#### Description

Produces fractional weights using the iterative proportional fitting algorithm.

#### Usage

`weight(cons, inds, vars = NULL, iterations = 10)`
Arguments

- *cons* A data frame containing all the constraints. This should be in the format of one row per zone, one column per constraint category. The first column should be a zone code; all other columns must be numeric counts.

- *inds* A data frame containing individual-level (survey) data. This should be in the format of one row per individual, one column per constraint. The first column should be an individual ID.

- *vars* A character vector of variables that constrain the simulation (i.e. independent variables)

- *iterations* The number of iterations the algorithm should complete. Defaults to 10

Details

The first column of each data frame should be an ID. The first column of *cons* should contain the zone codes. The first column of *inds* should contain the individual unique identifier.

Both data frames should only contain:

- an ID column (zone ID *cons* or individual ID *inds*).
- constraints *inds* or constraint category *cons*.
- *inds* can optionally contain additional dependent variables that do not influence the weighting process.

No other columns should be present (the user can merge these back in later).

It is essential that the levels in each *inds* constraint (i.e. column) match exactly with the column names in *cons*. In the example below see how the column names in *cons* (`'age_0_49', 'sex_f', ...`) match exactly the levels in * inds* variables.

The columns in *cons* must be in alphabetical order because these are created alphabetically when they are 'spread' in the individual-level data.

Value

A data frame of fractional weights for each individual in each zone with zone codes recorded in column names and individual id recorded in row names.

Examples

```r
# SimpleWorld
cons <- data.frame(
  "zone" = letters[1:3],
  "age_0_49" = c(8, 2, 7),
  "age_gt_50" = c(4, 8, 4),
  "sex_f" = c(6, 6, 8),
  "sex_m" = c(6, 4, 3),
  stringsAsFactors = FALSE
)
inds <- data.frame(
  "id" = LETTERS[1:5],
  "age" = c("age_gt_50", "age_gt_50", "age_0_49", "age_gt_50", "age_0_49"),
```
weight

```
"sex" = c("sex_m", "sex_m", "sex_m", "sex_f", "sex_f"),
"income" = c(2868, 2474, 2231, 3152, 2473),
stringsAsFactors = FALSE
```

# Set variables to constrain over
vars <- c("age", "sex")
weights <- weight(cons = cons, inds = inds, vars = vars)
print(weights)

Index

check_constraint, 2
check_ind, 3

eextract, 3
extract_weights, 4

integerise, 5
integerize (integerise), 5

rake, 6

simulate, 7

weight, 7