Package 'saeSim'

May 16, 2014

Type Package

Title Simulation Tools for Small Area Estimation

Date 2014-05-12

Version 0.2

Author Sebastian Warnholz

Maintainer Sebastian Warnholz <Sebastian.Warnholz@fu-berlin.de>

URL wahani.github.io/saeSim

Depends R(>= 3.0),dplyr,spdep,ggplot2,MASS,methods

Suggests testthat

Description Tools for the simulation of data in the context of small area estimation.

Roxygen list(wrap = FALSE)

License GPL-2 | file LICENSE

R topics documented:

`agg_standard` *Aggregate function*

Description

This is one implementation for aggregating data in a simulation set-up. It is designed to be used as argument to `sim_agg`. See the examples there.

Usage

```
agg_standard(splitVars = "idD")
```

Arguments

`splitVars`	variable names as character to split the data

Details

This function follows the split-apply-combine idiom. I.e. each data set is split by the defined variables. Then the variables within each subset are aggregated (reduced to one row). Logical variables are reduced by `any`; for characters and factors the most frequent value is taken; and for numerics the mean (removing NAs).

See Also

`sim_agg`

`calc_var` *Calculator function*

Description

This function can be used to calculate some new variables on the data. This function is supposed to be used with `sim_calc`. See the examples there.

Usage

```
calc_var(varName = "y", funList = list(mean = mean, var = var),
  exclude = NULL, by = "idD", newName = varName)
```

Arguments

`varName`	a chracter giving the name of the variable in the data, on which a function is applied.
`funList`	list of functions to be applied on `varName`. Can be named, see `newName`
`exclude`	charcter vector of variable names in the data used to exclude observations from the calculation or `NULL`. The variables must be `logical`, `TRUE`s will be excluded.
`by`	variable names as character for which the data is split. Computed values will be constant within each subset

`newName`	name of the new variable. If `length(funList) > 1` it can be vector. If equal to `varName` the name will be pasted with the function name (if the list is named) or appended by a increasing sequence of integer

See Also

`sim_calc`

`gen_norm` *Generator functions*

Description

This function family is designed to draw random numbers according to the setting for domains and units. fe stands for fixed effects, e for the model error and v for an additional error component defined on area level (random effect). These functions are used in combination with the `sim_gen` function family. These functions are not called directly but via `sim_gen`

Usage

```
gen_norm(mean = 0, sd = 1)

gen_v_norm(mean = 0, sd = 1)

gen_v_sar(mean = 0, sd = 1, rho = 0.5, type = "rook")
```

Arguments

`mean`	the mean passed to the random number generator, for example `rnorm`
`sd`	the standard deviation passed to the random number generator, for example rnorm
`rho`	the correlation used to create the variance covariance matrix for a SAR process - see `cell2nb`
`type`	either "rook" or "queen". See `cell2nb` for detials

Details

`gen_norm` is used to draw random numbers from a homoscedastic normal distribution. This generator is used for the fixed effects part and model error.

`gen_v_norm` and `gen_v_sar` will create an area-level random component. In the case of `v_norm`, the error component will be from a normal distribution and i.i.d. from an area-level perspective (all units in an area will have same value, all areas are independent). v_sar will also be from a normal distribution, but the errors are correlated. The variance covariance matrix is constructed for a SAR(1) - spatial/simultanous autoregressive process. mvrnorm is used for the random number generation.

See Also

For examples: `sim_gen`, `sim_gen_fe`, `sim_gen_e`, `sim_gen_ec`, `sim_gen_re` and `sim_gen_rec`

make_id *Construct ID-Variables*

Description

This function can be used to construct a data frame with `id*` variables. This is helpful for user written generator functions.

Usage

```
make_id(nDomains, nUnits, ...)

## S4 method for signature 'numeric,numeric'
make_id(nDomains, nUnits, ...)
```

Arguments

`nDomains`	The number of domains. Can be considered as cluster variable.
`nUnits`	The number of units in each domain. If `length(nUnits) > 1` each elemnt is the number of units in each domain respectively.
`...`	arguments passed to methods

Examples

```
make_id(2, 2)
make_id(2, c(2, 3))
```

sample_csrs *Sampling function*

Description

This function controls the sampling mechanism. They are designed to be used with `sim_sample`. `sample_csrs` will draw with simple random sampling in each cluster. The cluster is hard coded as `idD`.

Usage

```
sample_csrs(size = 0.05)
```

Arguments

`size`	can either be >= 1 giving the sample size (in each cluster) or < 1 where it is treated as proportion (in each cluster). Additionally size can have `length(size) > 1` which will be interpreted as different sample sizes in each cluster/domain.

See Also

`sample_srs`, `sample_sampleWrapper`, `sim_sample`

`sample_sampleWrapper` *Sampling function*

Description

These function control the sampling mechanism. They are designed to be used with `sim_sample`. `sample_sampleWrapper` is a wrapper of the sample function already implemented in R. The arguments will simply be passed to `sample`

Usage

```
sample_sampleWrapper(...)
```

Arguments

`...`	Arguments passed to `sample`

See Also

`sample_srs`, `sample_csrs`, `sim_sample`

`sample_srs` *Sampling function*

Description

This function controls the sampling mechanism. They are designed to be used with `sim_sample`. `sample_srs` will draw with simple random sampling.

Usage

```
sample_srs(size = 0.05, ...)
```

Arguments

`size`	can either be >= 1 giving the sample size or < 1 where it is treated as proportion
`...`	Arguments passed to `sample.int`

See Also

`sample_sampleWrapper`, `sample_csrs`, `sim_sample`

sim *Start simulation*

Description

This function can be applied to a sim_setup object or sim_base. It will start the simulation. Use the printing method as long as you are testing the scenario.

Usage

```
sim(x, ...)

## S4 method for signature 'sim_base'
sim(x, ...)

## S4 method for signature 'sim_setup'
sim(x, ..., R = NULL)
```

Arguments

x	a sim_setup or sim_base constructed with sim_setup() or sim_base_standard()
...	simulation components added with sim_*
R	the number of desired repetitions in the simulation

Examples

```
setup <- sim_lm()
resultList <- sim(setup, R = 1)

# Will return a data frame
dat <- sim(sim_base_standard(), sim_gen_fe(), sim_gen_e())
```

sim_agg *Aggregation component*

Description

Aggregating the data is another component which can be used on the population or sample. The aggregation will simply be done after the sampling, if you haven't specified any sampling component, the population is aggregated (makes sense if you draw samples directly from the model). The unit identifier idU will be lost.

Usage

```
sim_agg(aggFun = agg_standard())
```

Arguments

aggFun	function which controls the aggregation process. At the moment only agg_standard is defined.

See Also

agg_standard

Examples

```
# Aggregating the population:
sim_lm() %+% sim_agg()

# Aggregating after sampling:
sim_lm() %+% sim_sample() %+% sim_agg()
```

---

sim_base_standard *Basics for a simulation setup*

Description

Use the ‘sim_base_*‘ functions to start a new simulation setup. Everything else are just preconfigured setups.

Usage

```
sim_base_standard(nDomains = 100, nUnits = 100)

sim_lm()

sim_lmm()

sim_lmc()

sim_lmmc()
```

Arguments

nDomains	the number of domains
nUnits	the number of units

---

sim_calc *Add new variables*

Description

These functions can be used for adding new variables to the data.

Usage

```
sim_calc(calcFun = calc_var(), level = "population")

sim_n()

sim_N()

sim_popMean(exclude = NULL)

sim_popVar(exclude = NULL)
```

Arguments

calcFun	a function used for calculation
level	character given the level on which the variable is to be calculated. One in c("population", "sample", "agg")
exclude	charcter vector of variable names in the data used to exclude observations from the calculation or NULL. The variables must be logical, TRUEs will be excluded.

See Also

calc_var

Examples

```
# Standard behavior
sim_base_standard() %+% sim_gen_fe() %+% sim_calc()

# Custom data modifications
## Add predicted values of a linear model
library(saeSim)

calc_lm <- function(dat) {
  dat$linearPredictor <- predict(lm(y ~ x, data = dat))
  dat
}

sim_base_standard() %+% sim_gen_fe() %+% sim_gen_e() %+% sim_calc(calc_lm)
```

---

sim_gen *Add generated data*

Description

These functions can be used to add data to a setup. The terminology comes from mixed models.

Usage

```
sim_gen(generator, const = 0, slope = 1, name = "variableName",
  nCont = NULL, level = NULL, fixed = NULL)

sim_gen_fe(generator = gen_norm(0, 4), const = 100, slope = 1,
  name = "x")

sim_gen_e(generator = gen_norm(0, 4), name = "e")

sim_gen_ec(generator = gen_norm(mean = 0, sd = 150), nCont = 0.05,
  level = "unit", fixed = TRUE, name = "e")

sim_gen_re(generator = gen_v_norm(), name = "v")

sim_gen_rec(generator = gen_v_norm(mean = 0, sd = 40), nCont = 0.05,
  level = "area", fixed = TRUE, name = "v")
```

Arguments

`generator`	generator function used to generate random numbers
`const`	constant/intercept in a fixed effects part
`slope`	slope in a fixed effects part
`name`	variable name used in the resulting `data.frame`
`level`	"unit", "area" or "none" - is the whole area contaminated, units inside an area or random observations in the data
`nCont`	gives the number of contaminated observations. Values between 0 and 1 will be trated as proportion. If length is larger 1, the expected length is the number of domains, you can specify something else in each domain. Integers are expected in that cas - numeric will be converted to integer
`fixed`	TRUE fixes the observations which will be contaminated. FALSE will result in a random selection of contaminated observations. Default is NULL for non-contaminated scenarios.

See Also

`gen_norm`, `gen_v_norm`, `gen_v_sar`

Examples

```
# Data setup for a mixed model
sim_base_standard() %+% sim_gen_fe() %+% sim_gen_re %+% sim_gen_e()
# Adding contamination in the model error
sim_base_standard() %+% sim_gen_fe() %+% sim_gen_re %+% sim_gen_e() %+% sim_gen_ec()
```

sim_sample *Sampling component*

Description

This component can be used to add a sampling mechanism to the simulation set-up. A sample will be drawn after the population is generated (sim_gen) and variables on the population are computed (sim_calc)

Usage

```
sim_sample(smplFun = sample_csrs(size = 5L))
```

Arguments

smplFun	function which controls the sampling process. sample_csrs is the default

See Also

sample_srs, sample_csrs, sample_sampleWrapper

Examples

```
# Simple random sample - 5% sample:
sim_lm() %+% sim_sample(sample_srs())

# Simple random sampling proportional to size - 5% in each domain:
sim_lm() %+% sim_sample(sample_csrs())
```

sim_setup *Construct a simulation set-up*

Description

This function is used to construct a new simulation set-up. There are several ways to work with it. Please see the examples and documentation.

Usage

```
sim_setup(base, ...)

## S4 method for signature 'sim_base'
sim_setup(base, ..., R = 500, simName = "test")

## S4 method for signature 'sim_setup'
sim_setup(base, ...)
```

Arguments

`base`	a object constructed by the `sim_base_*` family or a `sim_setup` object, constructed with `sim_setup`
`...`	simulation components, like `sim_gen`, etc.
`R`	the number of desired repetitions in the simulation
`simName`	the name of the simulation. It is simply added as character to the data

Value

An objects of class `sim_setup`. Should be used in conjunction with methods for this class.

See Also

`sim`, `sim_base_standard`

Index