Package: synthesizer (via r-universe)

March 9, 2025

Type Package

Title Synthesize Data Based on Empirical Quantile Functions and Rank Order Matching

Version 0.4.0

Maintainer Mark van der Loo <mark.vanderloo@gmail.com>

Description Data is synthesized using a combination of inverse transform sampling using the empirical quantile functions for each variable, and then copying the rank order structure from the original dataset. The syntesizer method has a tunable parameter allowing to gradually move from realistic and possibly unsafe synthetic data to decorrelated data of less utility.

License EUPL

URL https://github.com/markvanderloo/synthesizer

Imports stats, randomForest

VignetteBuilder simplermarkdown

Depends R (>= 3.5.0)

Suggests tinytest, simplermarkdown

Encoding UTF-8

RoxygenNote 7.3.2

NeedsCompilation no

Author Mark van der Loo [aut, cre] (<https://orcid.org/0000-0002-9807-4686>)

Repository CRAN

Date/Publication 2025-02-07 11:30:04 UTC

Contents

`make_synthesizer` *Create a function that generates synthetic data*

Description

Create a function that accepts a non-negative integer `n`, and that returns synthetic data sampled from the emperical (multivariate) distribution of `x`.

Usage

```
make_synthesizer(x, ...)

## S3 method for class 'numeric'
make_synthesizer(x, ...)

## S3 method for class 'integer'
make_synthesizer(x, ...)

## S3 method for class 'logical'
make_synthesizer(x, ...)

## S3 method for class 'factor'
make_synthesizer(x, ...)

## S3 method for class 'character'
make_synthesizer(x, ...)

## S3 method for class 'ts'
make_synthesizer(x, ...)

## S3 method for class 'data.frame'
make_synthesizer(x, rankcor = 1, ...)
```

Arguments

`x`	`[vector|data.frame]` Template data to be synthesized.
`...`	arguments passed to other methods
`rankcor`	`[numeric]` in $(0, 1]$ The correlations between the ranks of the real data and synthetic data. Either a single number or a vector of the form `c("variable1"=x1,...)`. Only used if `x` is a data frame.

Value

A `function` accepting a single integer argument: the number of synthesized values or records to return.

See Also

Other synthesis: `synthesize()`

Examples

```
synth <- make_synthesizer(cars$speed)
synth(10)


synth <- make_synthesizer(iris)
synth(6)
synth(150)
synth(250)
```

`pmse` *Compute the pMSE metric between synthetic and real data*

Description

The propensity mean squared error is defined as $\frac{1}{N}\sum_{i=1}^{N}(p_i - c)^2$, where c is the number of synthetic records, divided by the sum of the number of synthetic and real records.

Usage

```
pmse(synth, real, model = c("lr", "rf"), nrep = NULL)
```

Arguments

`synth`	`[data.frame]` Synthesized data.
`real`	`[real]` Data to compare with the synthesized data.
`model`	`[character]` Model used to compute propensity scores. Options are `"lr"`: logistic regression, and `"rf"`: random forest.
`nrep`	`[integer]` Number of model repetitions to average the pMSE value over. Ignored for `lr`.

Value

`[numeric]` scalar.

Examples

```
scars <- synthesize(cars)
pmse(scars, cars)
```

`synthesize` *Create synthetic version of a dataset*

Description

Create `n` values or records based on the emperical (multivariate) distribution of `y`. For data frames it is possible to decorrelate synthetic from the original variables by lowering the value for the `rankcor` parameter.

Usage

```
synthesize(x, n = NROW(x), rankcor = 1)
```

Arguments

`x`	`[vector|data.frame]` data to synthesize.
`n`	`[integer]` Number of values or records to synthesize.
`rankcor`	`[numeric]` in $[0, 1]$. Either a single rank correlation value that is applied to all variables, or a vector of the form `c(variable1=ut1lity1,...)`. Variables not explicitly mentioned will have `rankcor=1`. See also the note below.

Value

A data object of the same type and structure as `x`.

Note

The utility of a synthetic variable is lowered by decorelating the rank correlation between the real and synthetic data. If `rankcor=1`, the synthetic data will ordered such that it has the same rank order as the original data. If `rankcor=0`, no such reordering will take place. For values between 0 and 1, blocks of data are randomly selected and randomly permuted iteratively until the rank correlation between original and synthetic data drops below the parameter.

See Also

Other synthesis: `make_synthesizer()`

Examples

```
synthesize(cars$speed,10)
synthesize(cars)
synthesize(cars,25)

s1 <- synthesize(iris, rankcor=1)
s2 <- synthesize(iris, rankcor=0.5)
s3 <- synthesize(iris, rankcor=c("Species"=0.5))

oldpar <- par(mfrow=c(2,2), pch=16, las=1)
```

```
plot(Sepal.Length ~ Sepal.Width, data=iris, col=iris$Species, main="Iris")
plot(Sepal.Length ~ Sepal.Width, data=s1, col=s1$Species, main="Synthetic Iris")
plot(Sepal.Length ~ Sepal.Width, data=s2, col=s2$Species, main="Low utility Iris")
plot(Sepal.Length ~ Sepal.Width, data=s3, col=s3$Species, main="Low utility Species")
par(oldpar)
```

Index