

Package: simulatetimeseries (via r-universe)

October 17, 2026

Type Package

Title Time Series dataset collection (real-world and synthetic)

Version 0.2.1

Date 2025-10-02

Description Collection of functions for simulating various types of time series data and accessing real-world time series datasets.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Imports stats, utils, ahead, astsa, expsmooth, fma, forecast, fpp2, MASS, tswge, Rcpp, misc, tensorflow

LinkingTo Rcpp

Suggests rugarch, keras3

RoxygenNote 7.3.2

Depends R (>= 3.5), keras3

Remotes thierrymoudiki/misc

Config/pak/sysreqs cmake libfontconfig1-dev libfreetype6-dev libfribidi-dev make libharfbuzz-dev texlive libicu-dev libjpeg-dev libpng-dev libtiff-dev libuv1-dev libwebp-dev libxml2-dev libssl-dev python3 libx11-dev zlib1g-dev

Repository <https://techtonique.r-universe.dev>

Date/Publication 2026-04-23 14:24:16 UTC

RemoteUrl <https://github.com/thierrymoudiki/simulatetimeseries>

RemoteRef HEAD

RemoteSha a2313321a908d554cef62afc776db34bae0de1cc

Contents

simulatetimeseries-package	2
check_list_contains	3

create_correlation_matrix	3
debug_print	4
get_data_1	4
lstmf	5
martingale_test	5
meboot	6
rbootstrap	8
rcpp_hello_world	9
removenas	9
rfitdistr	10
rgan	10
rgaussiandens	11
rsurrogate	12
simulate_correlated_gaussian	12
simulate_time_series_1	13
simulate_time_series_2	14
simulate_time_series_3	15
simulate_time_series_4	15
splitts	16
train_gan	16
ts_dataset	17
Index	18

simulatetimeseries-package
<i>Time Series dataset collection (real-world and synthetic)</i>

Description

Collection of functions for simulating various types of time series data and accessing real-world time series datasets.

Package Content

Index: This package was not yet installed at build time.

Maintainer

T. Moudiki <thierry.moudiki@gmail.com>

Author(s)

T. Moudiki [aut, cre]

check_list_contains	<i>Check if a list contains an object</i>
---------------------	---

Description

Checks whether a list of time series objects already contains a given element, comparing values, start, and frequency.

Usage

```
check_list_contains(list_elts, new_elt)
```

```
check_list_contains(list_elts, new_elt)
```

Arguments

list_elts	List of time series objects.
new_elt	A time series object to check for presence in the list.
lst	list to check
obj	object to look for

Value

logical indicating if object is in list
 Logical. TRUE if the element is found, FALSE otherwise.

Examples

```
x <- ts(1:10)
y <- ts(1:10)
check_list_contains(list(x), y)
```

create_correlation_matrix	<i>create correlation matrix</i>
---------------------------	----------------------------------

Description

create correlation matrix

Usage

```
create_correlation_matrix(cor_values)
```

debug_print	<i>Print a variable for debugging</i>
-------------	---------------------------------------

Description

Prints the name and value of a variable to the console, with a newline for clarity.

Usage

```
debug_print(x)
```

Arguments

x Any R object to print.

Value

Invisibly returns the input object.

Examples

```
debug_print(iris)
```

get_data_1	<i>Get data 1</i>
------------	-------------------

Description

Data from Task Views + synthetic

Usage

```
get_data_1(diffs = TRUE, install_pkgs = FALSE)
```

Arguments

diffs return the differentiated series or not? (lag = 1)

Value

a list of time series objects

`lstmf`*LSTM Forecasting for Financial Returns*

Description

LSTM Forecasting for Financial Returns

Usage

```
lstmf(  
  y,  
  h = 10,  
  level = c(80, 95),  
  lookback = 20,  
  units = 50,  
  epochs = 100,  
  batch_size = 32  
)
```

Arguments

<code>y</code>	Univariate time series of returns
<code>h</code>	Forecast horizon
<code>level</code>	Confidence levels for prediction intervals
<code>lookback</code>	Number of previous periods to use for prediction
<code>units</code>	Number of LSTM units
<code>epochs</code>	Training epochs
<code>batch_size</code>	Batch size for training

Value

Object of class 'forecast' with LSTM predictions

`martingale_test`*Martingale Test for Time Series*

Description

Performs a set of martingale tests on a numeric vector, using polynomial and inverse polynomial trends.

Usage

```
martingale_test(x)
```

Arguments

x Numeric vector. The time series to test.

Value

A named list of p-values for each martingale test.

Examples

```
x <- rnorm(100)
martingale_test(x)
```

meboot

Maximum Entropy Bootstrap for Time Series using Rcpp

Description

Generates bootstrap replicates of a time series using the maximum entropy bootstrap algorithm with Rcpp implementation for improved performance. This method is particularly useful for non-stationary time series and preserves the dependence structure of the original data.

Usage

```
meboot(
  x,
  reps = 999,
  trim = list(trim = 0.1, xmin = NULL, xmax = NULL),
  reachbnd = TRUE,
  expand.sd = TRUE,
  force.clt = TRUE,
  scl.adjustment = FALSE,
  sym = FALSE,
  colsubj,
  coldata,
  coltimes,
  ...
)
```

Arguments

x A numeric vector or time series object to be bootstrapped.

reps Number of bootstrap replicates to generate (default: 999).

trim Controls tail behavior. Can be a single numeric value specifying the trim proportion (e.g., 0.10 for 10
trim Trim proportion for tail calculation (default: 0.10)

	xmin Lower bound for generated values (optional)
	xmax Upper bound for generated values (optional)
reachbnd	Logical indicating whether to allow generated values to reach the boundaries xmin and xmax (default: TRUE).
expand.sd	Logical indicating whether to expand the standard deviation of the ensemble (default: TRUE).
force.clt	Logical indicating whether to force the central limit theorem compliance by centering each replicate (default: TRUE).
scl.adjustment	Logical indicating whether to adjust the scale of the ensemble to match the original data's variance (default: FALSE).
sym	Logical indicating whether to force symmetry in the maximum entropy density (default: FALSE).
colsubj	Deprecated parameter from original meboot (included for compatibility).
coldata	Deprecated parameter from original meboot (included for compatibility).
coltimes	Deprecated parameter from original meboot (included for compatibility).
...	Additional arguments passed to expansion functions.

Details

The maximum entropy bootstrap algorithm generates replicates that:

- Preserve the dependence structure of the original time series
- Can handle non-stationary time series
- Satisfy the ergodic theorem and central limit theorem
- Maintain the mean and autocorrelation structure

The Rcpp implementation provides significant performance improvements over the original R implementation, especially for large datasets and many replications.

Value

A list with the following components:

x	Original time series data
ensemble	Matrix of bootstrap replicates (n x reps)
xx	Sorted original data
z	Intermediate points between sorted values
dv	Absolute differences between consecutive observations
dvtrim	Trimmed mean of differences
xmin	Lower bound used for generation
xmax	Upper bound used for generation
desintxb	Interval means satisfying mean-preserving constraint
ordxx	Ordering index of original data
kappa	Scale adjustment factor (if scl.adjustment = TRUE)

References

Vinod, H. D., & Lopez-de-Lacalle, J. (2009). Maximum entropy bootstrap for time series: The meboot R package. *Journal of Statistical Software*, 29(5), 1-19.

Examples

```
# Basic usage with a time series
set.seed(123)
x <- ts(rnorm(100), start = c(2000, 1), frequency = 12)
result <- meboot(x, reps = 1000)

# Plot first few replicates
matplot(result$ensemble[, 1:5], type = "l", lty = 1)
lines(result$x, col = "black", lwd = 2)

# With custom bounds
result_bounded <- meboot(x, reps = 100,
                        trim = list(trim = 0.1, xmin = -3, xmax = 3))

# With scale adjustment
result_scaled <- meboot(x, reps = 100, scl.adjustment = TRUE)
```

rbootstrap

Simulate using bootstrap resampling

Description

Generates bootstrap samples from a numeric vector, optionally producing multiple replicates.

Usage

```
rbootstrap(x, n = length(x), p = 1, seed = 123)
```

Arguments

x	Numeric vector to resample.
n	Integer. Number of samples per replicate (default: length of x).
p	Integer. Number of replicates (default: 1).
seed	Integer. Random seed for reproducibility (default: 123).

Value

A vector or matrix of bootstrap samples.

Examples

```
x <- rnorm(10)
rbootstrap(x, n = 10, p = 3)
```

rcpp_hello_world	<i>Simple function using Rcpp</i>
------------------	-----------------------------------

Description

Simple function using Rcpp

Usage

```
rcpp_hello_world()
```

Examples

```
## Not run:
rcpp_hello_world()

## End(Not run)
```

removenas	<i>Remove NA values from a time series</i>
-----------	--

Description

Remove NA values from a time series
remove NAs by linear interpolation

Usage

```
removenas(y)

removenas(y)
```

Arguments

x	time series object
---	--------------------

Value

time series object without NA values

rfitdistr	<i>Simulate from parametric distribution</i>
-----------	--

Description

Simulate from parametric distribution

Usage

```
rfitdistr(x, n = length(x), p = 1)
```

rgan	<i>Generate Synthetic Data using GANs</i>
------	---

Description

Creates synthetic data using Generative Adversarial Networks with predefined architectures optimized for different data types.

Usage

```
rgan(x, n, p = 1, type_input = c("unimodal", "mixture", "financial"))
```

Arguments

x	Matrix or data.frame. Input data to learn distribution from.
n	Integer. Number of synthetic samples (rows) to generate.
p	Integer. Number of features (columns) to generate. Currently must match input dimension.
type_input	Character. Type of data distribution: "unimodal", "mixture", or "financial".

Value

A matrix of synthetic data with n rows and p columns.

Examples

```
## Not run:
# Unimodal normal distribution
real_data <- matrix(rnorm(1000, mean = 2, sd = 4))
synthetic <- rgan(real_data, n = 500, p = 1, type_input = "unimodal")

# Mixture distribution
mixture <- rbinom(1000, 1, 0.5)
real_mixture <- matrix(mixture * rnorm(1000, 2, 1) + (1-mixture) * rnorm(1000, 8, 2))
synthetic <- rgan(real_mixture, n = 500, p = 1, type_input = "mixture")
```

```
# Financial data
synthetic <- rgan(EuStockMarkets[1:500,1], n = 500, p = 1, type_input = "financial")

## End(Not run)
```

rgaussiandens

Simulate Gaussian Kernel Density

Description

Generates samples from a Gaussian kernel density estimate of a numeric vector.

Usage

```
rgaussiandens(
  x,
  n = length(x),
  p = 1,
  seed = 123,
  method = c("antithetic", "traditional")
)
```

Arguments

x	Numeric vector to estimate density from.
n	Integer. Number of samples per replicate (default: length of x).
p	Integer. Number of replicates (default: 1).
seed	Integer. Random seed for reproducibility (default: 123).
method	Character. Sampling method: "antithetic" or "traditional".

Value

A vector or matrix of samples from the estimated density.

Examples

```
x <- rnorm(10)
rgaussiandens(x, n = 10, p = 3)
```

rsurrogate	<i>Simulate using surrogate data</i>
------------	--------------------------------------

Description

Simulate using surrogate data

Usage

```
rsurrogate(x, n = length(x), p = 1, seed = 123)
```

simulate_correlated_gaussian	<i>Simulate Correlated Gaussian Random Variables</i>
------------------------------	--

Description

Generates a matrix of correlated Gaussian random variables using a specified correlation matrix.

Usage

```
simulate_correlated_gaussian(n = 100L, cor_matrix = diag(3))
```

Arguments

n	Integer. Number of samples to generate (rows of output matrix).
cor_matrix	Numeric matrix. Desired correlation matrix (must be positive definite).

Value

A numeric matrix of dimension n x p, where p is the number of variables (columns in cor_matrix).

Examples

```
cor_matrix <- matrix(c(1, 0.6, 0.6, 1), nrow = 2, byrow = TRUE)
correlated_gaussian <- simulate_correlated_gaussian(n=100, cor_matrix=cor_matrix)
print(cor(correlated_gaussian))
```

simulate_time_series_1
<i>Simulate Univariate Time Series Dataset 1</i>

Description

Generates a univariate time series with specified trend, seasonality, and noise distribution.

Usage

```
simulate_time_series_1(  
  n,  
  trend = c("linear", "quadratic"),  
  seasonality = c("none", "sinusoidal"),  
  distribution = c("normal", "student"),  
  noise_sd = 10,  
  seed = 123  
)
```

Arguments

n	Integer. Number of data points.
trend	Character. "linear" or "quadratic".
seasonality	Character. "none" or "sinusoidal".
distribution	Character. "normal" or "student".
noise_sd	Numeric. Standard deviation of noise.
seed	Integer. Random seed for reproducibility.

Value

A time series object.

Examples

```
ts_data <- simulate_time_series_1(n = 100L, trend = "quadratic", seasonality = "sinusoidal", noise_sd = 2500, distr  
plot(ts_data, type = "l", main = "Simulated Time Series")
```

`simulate_time_series_2`*Simulate a univariate time series dataset 2*

Description

Simulate a univariate time series dataset 2

Usage

```
simulate_time_series_2(  
  n,  
  trend = c("linear", "sinusoidal"),  
  seasonality = FALSE,  
  noise_sd = 0.1,  
  ar = 0,  
  ma = 0,  
  seed = 123  
)
```

Arguments

<code>n</code>	numerical, number of data points
<code>trend</code>	string, "linear" or "sinusoidal"
<code>seasonality</code>	string, "none" or "sinusoidal"
<code>noise_sd</code>	numerical, standard deviation of noise
<code>ar</code>	autoregressive order
<code>ma</code>	moving average order
<code>seed</code>	int, reproducibility seed

Value

a native time series object

Examples

```
ts_data <-  
simulate_time_series_2(  
  n = 100L,  
  trend = "sinusoidal",  
  seasonality = TRUE,  
  noise_sd = runif(n = 1, min = 20, max=50)  
)  
plot(ts_data, type = "l", main = "Simulated Time Series")
```

`simulate_time_series_3`*Simulate a univariate time series dataset 3*

Description

Simulate a univariate time series dataset 3

Usage

```
simulate_time_series_3(n = 100, seed = 123)
```

Arguments

n	numerical, number of data points
seed	int, reproducibility seed

Value

a native time series object

Examples

```
print(simulate_time_series_3(10))
```

`simulate_time_series_4`*Simulate a univariate time series dataset 4*

Description

Simulate a univariate time series dataset 4

Usage

```
simulate_time_series_4(n = 600, psi = 0.1, theta = 0.1, seed = 123)
```

Arguments

n	numerical, number of data points
psi	1st parameter for innovation variance (in [0, 1])
theta	2nd parameter for innovation variance (in [0, 1])
seed	int, reproducibility seed

Value

a native time series object

Examples

```
plot(simulate_time_series_4())
```

splitts	<i>split time series sequentially</i>
---------	---------------------------------------

Description

split time series sequentially

Usage

```
splitts(y, split_prob = 0.5, return_indices = FALSE, ...)
```

train_gan	<i>Train a Simple GAN Model</i>
-----------	---------------------------------

Description

Trains a simple Generative Adversarial Network (GAN) using user-supplied generator and discriminator functions.

Usage

```
train_gan(
  train_dat,
  generator_fn,
  discriminator_fn,
  n_iter = 5,
  epochs_per_iter = 30,
  num_resamples = 500,
  seed = 123L
)
```

Arguments

train_dat	Matrix or data.frame. Training data for the GAN.
generator_fn	Function. Returns a generator model given latent dimension.
discriminator_fn	Function. Returns a discriminator model given input dimension.
n_iter	Integer. Number of training iterations (default: 5).
epochs_per_iter	Integer. Number of epochs per iteration (default: 30).
num_resamples	Integer. Number of synthetic samples to generate after training (default: 500).
seed	Integer. Random seed for reproducibility (default: 123).

Details

Requires both 'tensorflow' and 'keras3' packages to be installed. The generator and discriminator functions must return valid Keras models.

Value

A list containing the trained GAN model ('model'), a matrix of synthetic resamples ('resamples'), and elapsed training time in seconds ('time').

Examples

```
# Example usage (requires keras3 and tensorflow)
# train_gan(train_dat = matrix(rnorm(100)), generator_fn = my_gen, discriminator_fn = my_disc)
```

ts_dataset	<i>Time Series Dataset Collection</i>
------------	---------------------------------------

Description

A collection of univariate time series from various R packages and synthetic data

Usage

```
ts_dataset
```

Format

A list containing multiple time series objects

Source

Various R packages including astsa, datasets, expsmooth, fma, forecast, fpp2, MASS, tswge, and synthetic data

Index

- * **bootstrap**
 - meboot, [6](#)
- * **datasets**
 - ts_dataset, [17](#)
- * **package**
 - simulatetimeseries-package, [2](#)
- * **ts**
 - meboot, [6](#)

check_list_contains, [3](#)
create_correlation_matrix, [3](#)

debug_print, [4](#)

get_data_1, [4](#)

lstmf, [5](#)

martingale_test, [5](#)
meboot, [6](#)

rbootstrap, [8](#)
rcpp_hello_world, [9](#)
removenas, [9](#)
rfitdistr, [10](#)
rgan, [10](#)
rgaussiandens, [11](#)
rsurrogate, [12](#)

simulate_correlated_gaussian, [12](#)
simulate_time_series_1, [13](#)
simulate_time_series_2, [14](#)
simulate_time_series_3, [15](#)
simulate_time_series_4, [15](#)
simulatetimeseries
 (simulatetimeseries-package), [2](#)
simulatetimeseries-package, [2](#)
splitts, [16](#)

train_gan, [16](#)
ts_dataset, [17](#)