

Package: esgtoolkit (via r-universe)

October 13, 2026

Type Package

Title Toolkit for Monte Carlo Simulations

Version 1.10.2

Date 2026-02-12

URL <https://github.com/Techtonique/esgtoolkit>

BugReports <https://github.com/Techtonique/esgtoolkit/issues>

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

Description A toolkit for Monte Carlo Simulations in Finance, Economics, Insurance, Physics. Multiple simulation models can be created by combining building blocks provided in the package.

License BSD_3_clause Clear + file LICENSE

Depends ggplot2, gridExtra, methods, reshape2, VineCopula, randtoolbox, zoo, data.table

Imports Rcpp(>= 0.11.0)

Suggests devtools, ggplot2, testthat, knitr, misc, tseries

VignetteBuilder knitr

LinkingTo Rcpp

NeedsCompilation yes

RoxygenNote 7.3.2

Remotes thierrymoudiki/misc

Config/pak/sysreqs libicu-dev

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

Date/Publication 2026-05-16 02:03:49 UTC

RemoteUrl <https://github.com/Techtonique/esgtoolkit>

RemoteRef HEAD

RemoteSha 08b2e8f3f18ab45bbaca7532b5e74ea7d0cffe7d

Contents

calculate_V	2
calculatereturns	3
enforce_monotonicity	4
esgcortest	4
esgdiscountfactor	5
esgfwdrates	6
esgmartingaletest	7
esgmccv	9
esgmcpprices	10
esgplotbands	11
esgplotshocks	12
esgplots	13
forwardrates	13
generate_diverse_sv_paths	14
generate_svjd	15
generate_synthetic_returns	16
plot.rvine_simulation	17
print.rvine_simulation	18
rg2plus	18
rpriceg2plus	19
rsvjd	20
rsvjdg2plus	21
sim_to_zoo	22
simdiff	23
simshocks	24
simulate_rvine	26
ycextra	27
ycinter	29
Index	31

calculate_V	<i>Calculate variance term $V(t,T)$ for G2++ model</i>
-------------	---

Description

Calculate variance term $V(t,T)$ for G2++ model

Usage

calculate_V(tau, a, b, sigma, eta, rho)

calculatereturns	<i>Calculate returns or log-returns for multivariate time series</i>
------------------	--

Description

Calculate returns or log-returns for multivariate time series

Usage

```
calculatereturns(x, type = c("basic", "log"))
```

Arguments

x	Multivariate time series
type	Type of return: basic return ("basic") or log-return ("log")

Value

A vector of returns

Examples

```
kappa <- 1.5
V0 <- theta <- 0.04
sigma_v <- 0.2
theta1 <- kappa*theta
theta2 <- kappa
n_simulations <- 10

eps0 <- simshocks(n = n_simulations, horizon = 5, frequency = "quart")
sim_GBM <- simdiff(n = n_simulations, horizon = 5, frequency = "quart",
                  model = "GBM",
                  x0 = 100, theta1 = 0.03, theta2 = 0.1, eps = eps0)

returns <- calculatereturns(sim_GBM)
log_returns <- diff(log(sim_GBM))

print(identical(returns, log_returns))

par(mfrow=c(1, 2))
matplot(returns, type = 'l')
matplot(log_returns, type = 'l')
```

enforce_monotonicity	<i>Enforce monotonicity: $P(t, T1) \geq P(t, T2)$ for $T1 < T2$</i>
----------------------	--

Description

Enforce monotonicity: $P(t, T1) \geq P(t, T2)$ for $T1 < T2$

Usage

```
enforce_monotonicity(bond_prices, maturities)
```

esgcortest	<i>Correlation test for shocks</i>
------------	------------------------------------

Description

Performs correlation tests between two sets of shocks at each time point.

Usage

```
esgcortest(
  x,
  alternative = c("two.sided", "less", "greater"),
  method = c("pearson", "kendall", "spearman"),
  conf.level = 0.95
)
```

Arguments

x	A list containing two time series objects to test for correlation
alternative	A character string specifying the alternative hypothesis: "two.sided" (default), "less", or "greater"
method	A character string indicating which correlation coefficient is to be used: "pearson" (default), "kendall", or "spearman"
conf.level	Confidence level for the returned confidence interval

Details

The function performs correlation tests between two sets of shocks at each time point. It returns both the correlation estimates and confidence intervals for the correlation coefficient at each time point.

Value

A list containing:

- `cor.estimate`: A time series of correlation estimates
- `conf.int`: A time series matrix containing the lower and upper bounds of the confidence intervals

Examples

```
# Generate sample shocks
x <- ts(matrix(rnorm(1000), 100, 10))
y <- ts(matrix(rnorm(1000), 100, 10))

# Test correlation between shocks
result <- esgctest(list(x, y))

# Plot correlation estimates with confidence intervals
plot(result$cor.estimate)
lines(result$conf.int[,1], col="red")
lines(result$conf.int[,2], col="red")
```

esgdiscountfactor	<i>Stochastic Discount Factors</i>
-------------------	------------------------------------

Description

Computes stochastic discount factors or discounted values based on interest rates and cash flows.

Usage

```
esgdiscountfactor(r, X)
```

Arguments

<code>r</code>	A numeric vector or time series of interest rates
<code>X</code>	A numeric vector or time series of cash flows or values to be discounted

Details

This function handles various combinations of scalar and time series inputs for both interest rates and cash flows. For scalar inputs, it performs simple exponential discounting. For time series inputs, it computes cumulative discounting over time.

Value

A time series of discounted values

Examples

```
# Simple scalar discounting
esgdiscountfactor(0.05, 100)

# Time series discounting
#r <- ts(rep(0.05, 10), start = 0, deltat = 1)
#X <- ts(rep(100, 10), start = 0, deltat = 1)
#esgdiscountfactor(r, X)
```

esgfwdrates	<i>Instantaneous forward rates</i>
-------------	------------------------------------

Description

Computes instantaneous forward rates from zero rates using various interpolation methods.

Usage

```
esgfwdrates(
  in.maturities,
  in.zerorates,
  n,
  horizon,
  out.frequency = c("annual", "semi-annual", "quarterly", "monthly", "weekly", "daily"),
  method = c("fmm", "periodic", "natural", "monoH.FC", "hyman", "HCSPL", "SW"),
  ...
)
```

Arguments

<code>in.maturities</code>	Vector of input maturities
<code>in.zerorates</code>	Vector of input zero rates
<code>n</code>	Number of simulations
<code>horizon</code>	Time horizon for forward rates
<code>out.frequency</code>	Output frequency for forward rates. One of: "annual" (default) "semi-annual" "quarterly" "monthly" "weekly" "daily"
<code>method</code>	Interpolation method. One of: "fmm" (default)

- "periodic"
- "natural"
- "monoH.FC"
- "hyman"
- "HCSPL"
- "SW"

... Additional arguments passed to interpolation functions

Details

The function computes instantaneous forward rates from zero rates using various interpolation methods. It first converts zero rates to zero-coupon prices, then interpolates these prices using the specified method. The forward rates are then computed from the interpolated prices.

The function supports different output frequencies and interpolation methods to suit various needs.

Value

A time series object containing the instantaneous forward rates

Examples

```
# Generate sample data
maturities <- c(1, 2, 3, 5, 7, 10)
zero_rates <- c(0.01, 0.015, 0.02, 0.025, 0.03, 0.035)

# Compute forward rates with annual frequency
fwd_rates <- esgfwdrates(in.maturities = maturities,
                        in.zerorates = zero_rates,
                        n = 1000,
                        horizon = 10,
                        out.frequency = "annual",
                        method = "fmm")
```

esgmartingaletest	<i>Test for martingale property and market consistency</i>
-------------------	--

Description

Performs statistical tests to verify if a time series follows a martingale property and maintains market consistency. Three different testing methods are available.

Usage

```
esgmartingaletest(
  r,
  X,
  p0,
  alpha = 0.05,
  method = c("onevsone", "trend", "ratio")
)
```

Arguments

<code>r</code>	Risk-free rate (scalar or time series)
<code>X</code>	Time series of asset prices or values
<code>p0</code>	Initial price(s) for comparison
<code>alpha</code>	Significance level for hypothesis tests (default: 0.05)
<code>method</code>	Testing method: • "onevsone": One-to-one comparison test • "trend": Trend analysis test • "ratio": Ratio-based test

Details

The function implements three different approaches to test for martingale properties:

- "onevsone": Compares each observation against its expected value
- "trend": Analyzes trends in the martingale differences
- "ratio": Tests the ratio between discounted values and initial prices

The test results help determine if the time series maintains the martingale property and market consistency under the risk-neutral measure.

Value

A list containing test results, which may include:

- t-statistic and p-value for ratio test
- trend analysis results
- mean and standard deviation of martingale differences

Examples

```
# Generate sample data
r <- 0.05
X <- ts(matrix(rnorm(1000), 100, 10), start = 0, deltat = 0.1)
p0 <- 100

# Perform martingale test using ratio method
```

```

result <- esgmartingaletest(r, X, p0, method = "ratio")

# Perform trend analysis
trend_result <- esgmartingaletest(r, X, p0, method = "trend")

```

esgmccv

*Convergence of Monte Carlo prices***Description**

Analyzes the convergence of Monte Carlo prices by computing average prices and confidence intervals as the number of simulations increases.

Usage

```
esgmccv(r, X, maturity, plot = TRUE, ...)
```

Arguments

<code>r</code>	Interest rate time series or constant
<code>X</code>	Price time series
<code>maturity</code>	Maturity time point
<code>plot</code>	Logical indicating whether to plot the convergence (default: TRUE)
<code>...</code>	Additional arguments passed to plotting functions

Details

The function computes the average price and confidence intervals for increasing numbers of simulations to analyze the convergence of Monte Carlo estimates. It discounts the price series using the provided interest rate and evaluates at the specified maturity.

If `plot=TRUE`, it creates a plot showing:

- Confidence intervals as shaded area
- Average price as a blue line
- X-axis showing number of simulations
- Y-axis showing Monte Carlo estimated prices

Value

A list containing:

- `avg.price`: Vector of average prices for different numbers of simulations
- `conf.int`: Matrix of confidence intervals (lower and upper bounds)

Examples

```
# Generate sample data
r <- 0.05
X <- ts(matrix(rnorm(1000), 100, 10), start = 0, deltat = 0.1)

# Analyze convergence
convergence <- esgmccv(r, X, maturity = 1)
```

esgmcprices

Estimation of discounted asset prices

Description

Calculates the discounted asset prices using simulated interest rates and asset values.

Usage

```
esgmcprices(r, X, maturity = NULL)
```

Arguments

r	Interest rate time series or constant value
X	Asset price time series or constant value
maturity	Optional maturity date for the calculation

Details

The function takes interest rates and asset prices as inputs and calculates the discounted asset prices. It handles both time series and constant values for both inputs.

If a maturity date is specified, the function returns the discounted price at that maturity. Otherwise, it returns the time series of discounted prices averaged across all scenarios.

Value

A time series object containing the discounted asset prices, or a single value if a maturity date is specified

Examples

```
# Using constant values
r <- 0.05
X <- 100
price <- esgmcprices(r, X)

# Using time series
r_ts <- ts(matrix(rnorm(100), 10, 10), start = 0, deltat = 0.1)
X_ts <- ts(matrix(rnorm(100), 10, 10), start = 0, deltat = 0.1)
```

```
prices_ts <- esgmcprices(r_ts, X_ts)

# With maturity
price_at_maturity <- esgmcprices(r_ts, X_ts, maturity = c(0, 5))
```

esgplotbands

Plot time series percentiles and confidence intervals

Description

Creates a plot showing time series data with confidence intervals, percentiles, and mean values.

Usage

```
esgplotbands(x, ...)
```

Arguments

x	A time series object containing the data to plot
...	Additional arguments passed to the plotting function

Details

The function creates a plot with:

- Mean values as the central line
- 95th and 5th percentiles as outer bands
- Upper and lower quartiles as inner bands
- Confidence intervals based on t-tests

The plot uses a color gradient from light yellow to light green for the different bands.

Value

A plot object

Examples

```
# Create sample time series
x <- ts(matrix(rnorm(1000), ncol=10), start=0, deltat=1)

# Plot with default settings
esgplotbands(x)

# Plot with custom title
esgplotbands(x, main="Custom Title")
```

`esgplotshocks`*Visualize the dependence between Gaussian shocks*

Description

Creates a scatter plot with marginal density plots to visualize the relationship between two sets of Gaussian shocks.

Usage

```
esgplotshocks(x, y = NULL)
```

Arguments

<code>x</code>	A matrix or list containing two columns of Gaussian shocks
<code>y</code>	Optional second matrix or list of Gaussian shocks to compare with <code>x</code>

Details

The function creates a scatter plot of the shocks with marginal density plots on the top and right sides. If `y` is provided, both sets of shocks are plotted with different colors (blue for `x`, red for `y`). The marginal density plots show the distribution of shocks along each dimension.

Value

A `ggplot2` object containing the scatter plot with marginal densities

Examples

```
# Generate sample Gaussian shocks
x <- matrix(rnorm(1000), 500, 2)

# Plot single set of shocks
esgplotshocks(x)

# Plot two sets of shocks for comparison
y <- matrix(rnorm(1000), 500, 2)
esgplotshocks(x, y)
```

`esgplotts`*Plot time series objects*

Description

Creates a line plot of time series data using `ggplot2`, with each series represented by a different color.

Usage

```
esgplotts(x)
```

Arguments

`x` A time series object (ts) containing the data to plot

Details

The function converts the time series data into a format suitable for `ggplot2` plotting. Each column of the time series is plotted as a separate line with a different color. The x-axis represents time/maturity and the y-axis shows the values.

Value

A `ggplot2` object containing the line plot

Examples

```
# Create sample time series data
x <- ts(matrix(rnorm(100), 20, 5), start = 0, deltat = 0.1)

# Plot the time series
esgplotts(x)
```

`forwardrates`*Forward rates extraction*

Description

This function extracts the forward rates from the results obtained with [ycinter](#) and [ycextra](#)

Usage

```
forwardrates(.Object)
```

Arguments

.Object An S4 object created by [ycinter](#) or [ycextra](#).

Value

A time series object giving the instantaneous forward rates for methods "NS", "SV" and the forward rates for methods "HCSPL", "SW"

Author(s)

Thierry Moudiki

See Also

[ycinter](#), [ycextra](#)

Examples

```
# Prices
p <- c(0.9859794,0.9744879,0.9602458,0.9416551,0.9196671,0.8957363,0.8716268,0.8482628,
0.8255457,0.8034710,0.7819525,0.7612204,0.7416912,0.7237042,0.7072136
,0.6922140,0.6785227,0.6660095,0.6546902,0.6441639,0.6343366,0.6250234,0.6162910,0.6080358,
0.6003302,0.5929791,0.5858711,0.5789852,0.5722068,0.5653231)

# Observed maturities
u <- 1:30

# Output maturities
t <- seq(from = 1, to = 60, by = 0.5)

# Svensson interpolation
yc <- ycextra(p = p, matsin = u, matsout = t,
method="SV", typeres="prices", UFR = 0.018)

plot(forwardrates(yc))
```

```
generate_diverse_sv_paths
```

Generate a list of synthetic return paths using stochastic volatility models with randomized parameters.

Description

This function produces a collection of synthetic return series, each simulated with randomized parameters for drift, volatility, jump characteristics, and regime-switching behavior. Useful for pretraining models or scenario testing.

Usage

```
generate_diverse_sv_paths(
  n_paths = 10000,
  horizon = 252 * 5,
  frequency = "daily",
  jump_type = "mixed",
  include_regime_switching = TRUE,
  random_seed = NULL
)
```

Arguments

n_paths	Number of return paths to generate.
horizon	Length of each return path in trading days.
frequency	Frequency of the data (e.g., "daily").
jump_type	Type of jump size distribution: "normal", "log_normal", "exponential", or "mixed".
include_regime_switching	Logical; whether to include Markov regime switching.
random_seed	Optional seed for reproducibility.

Value

A list of class 'diverse_sv_paths' containing: - 'paths': A list of numeric vectors (returns) - 'parameters': The parameter sets used to generate each path - 'horizon': The path length - 'frequency': The data frequency - 'n_paths': Number of paths - 'generation_date': Timestamp of generation

generate_svjd	<i>Generate random SVJD scenarios with Feller-compliant parameters</i>
---------------	--

Description

Generate random SVJD scenarios with Feller-compliant parameters

Usage

```
generate_svjd(
  n_series = 100,
  horizon = 252,
  frequency = "daily",
  type_return = c("log", "basic"),
  seed = NULL
)
```

Arguments

n_series	Number of scenarios (default=1000).
horizon	Time steps per scenario (default=252).
type_return	Type of return: basic return ("basic") or log-return ("log")
seed	Random seed (optional).
freq	Frequency ("daily", "weekly", "monthly").

Value

List of scenarios (each with 'log_returns', 'vol', and 'params').

generate_synthetic_returns

Generate a single synthetic stock return path using a stochastic volatility model with jumps and regime switching.

Description

This function simulates returns based on a Heston-style stochastic volatility process, optionally incorporating jumps (with configurable distributions), microstructure noise, and regime switching via a Markov chain.

Usage

```
generate_synthetic_returns(
  n_days = 252 * 10,
  mu = 2e-04,
  kappa = 0.05,
  theta = 1e-04,
  sigma_v = 0.01,
  rho = -0.7,
  lambda_jump = 0.05,
  jump_size_dist = "normal",
  sigma_jump = 0.02,
  noise_dist = "normal",
  noise_scale = 5e-04,
  noise_df = 5,
  regime_params = NULL,
  random_seed = NULL
)
```

Arguments

n_days	Number of trading days to simulate (default: 252 * 10).
mu	Drift (mean return) per time step.
kappa	Mean-reversion speed of the variance process.
theta	Long-run average variance.
sigma_v	Volatility of volatility (vol of variance process).
rho	Correlation between volatility and returns (leverage effect).
lambda_jump	Jump intensity (expected jumps per day).
jump_size_dist	Distribution of jump sizes ("normal", "log_normal", or "exponential").
sigma_jump	Scale parameter for jump size distribution.
noise_dist	Type of microstructure noise ("normal" or "student_t").
noise_scale	Standard deviation (or scale) of added microstructure noise.
noise_df	Degrees of freedom for Student-t noise (if used).
regime_params	List specifying regime switching behavior. Includes a transition matrix, and multipliers for 'theta' and 'kappa' during high-volatility regimes.
random_seed	Optional random seed for reproducibility.

Value

A 'data.table' with columns: 'date', 'returns', 'variance', 'volatility', 'regime'.

plot.rvine_simulation *Plot method for rvine_simulation objects*

Description

Creates diagnostic plots for R-vine copula simulation results.

Usage

```
## S3 method for class 'rvine_simulation'
plot(x, type = "distribution", ...)
```

Arguments

x	An object of class 'rvine_simulation'
type	Type of plot: "distribution" (default), "correlation", or "both"
...	Additional arguments passed to plotting functions

Value

A ggplot2 object or grid of plots

```
print.rvine_simulation
```

Print method for rvine_simulation objects

Description

Print method for rvine_simulation objects

Usage

```
## S3 method for class 'rvine_simulation'
print(x, ...)
```

Arguments

x An object of class 'rvine_simulation'
 ... Additional arguments passed to print

```
rg2plus
```

Simulate G2++ short rates model

Description

Simulate G2++ short rates model

Usage

```
rg2plus(
  n = 10L,
  horizon = 5L,
  freq = "semi-annual",
  u = 1:30,
  txZC = c(0.01422, 0.01309, 0.0138, 0.01549, 0.01747, 0.0194, 0.02104, 0.02236, 0.02348,
    0.02446, 0.02535, 0.02614, 0.02679, 0.02727, 0.0276, 0.02779, 0.02787, 0.02786,
    0.02776, 0.02762, 0.02745, 0.02727, 0.02707, 0.02686, 0.02663, 0.0264, 0.02618,
    0.02597, 0.02578, 0.02563),
  a = 0.5,
  b = 0.3541203,
  sigma = 0.09416266,
  eta = 0.08439934,
  rho = -0.99855687,
  methodyc = c("fmm", "hyman", "HCSPL", "SW"),
  ...
)
```

Arguments

n	Number of scenarios to simulate.
horizon	Time steps (semi-annual, default = 20).
freq	Frequency of simulation (default is "semi-annual").
u	Observed maturities (vector).
txZC	Yield to maturities (vector, same length as u).
a	G2++ mean reversion for factor x.
b	G2++ mean reversion for factor y.
sigma	Volatility of factor x.
eta	Volatility of factor y.
rho	Correlation between factors.
methodyc	Interpolation method for forward rates ("fmm", "hyman", "HCSPL", "SW").
...	Additional parameters to be passed to simdiff or simshocks

Value

A time series of simulated short rates for each scenario.

rpriceg2plus	<i>Simulate Zero-Coupon Bond Prices using G2++ Model</i>
--------------	--

Description

Simulate Zero-Coupon Bond Prices using G2++ Model

Usage

```
rpriceg2plus(
  n = 10L,
  horizon = 5L,
  freq = "semi-annual",
  u = 1:30,
  txZC = c(0.01422, 0.01309, 0.0138, 0.01549, 0.01747, 0.0194, 0.02104, 0.02236, 0.02348,
    0.02446, 0.02535, 0.02614, 0.02679, 0.02727, 0.0276, 0.02779, 0.02787, 0.02786,
    0.02776, 0.02762, 0.02745, 0.02727, 0.02707, 0.02686, 0.02663, 0.0264, 0.02618,
    0.02597, 0.02578, 0.02563),
  a = 0.5,
  b = 0.3541203,
  sigma = 0.09416266,
  eta = 0.08439934,
  rho = -0.99855687,
  maturities = c(5, 7, 10),
  start_ = c(2000, 1),
  seed = NULL,
  ...
)
```

Arguments

n	Number of scenarios to simulate.
horizon	Time steps for simulation (e.g., 5 for 5 years).
freq	Frequency of simulation (default is "semi-annual").
u	Observed maturities (vector).
txZC	Yield to maturities (vector, same length as u).
a	G2++ mean reversion for factor x.
b	G2++ mean reversion for factor y.
sigma	Volatility of factor x.
eta	Volatility of factor y.
rho	Correlation between factors.
maturities	Bond maturity times.
start_	Starting time for ts object.
seed	Random seed for reproducibility.
...	Additional parameters to be passed to simdiff or simshocks

Value

A time series of simulated bond prices for each scenario.

rsvjd

Simulate SVJD model (with Feller condition check)

Description

Simulate SVJD model (with Feller condition check)

Usage

```
rsvjd(
  n = 10L,
  horizon = 5L,
  freq = "daily",
  S0 = 100,
  V0 = 0.04,
  r0 = 0.02,
  kappa = 5,
  theta = 0.04,
  volvol = 0.5,
  rho = -0.7,
  lambda = 0.1,
  mu_J = 0,
  sigma_J = 0.1,
  start_ = NULL,
  ...
)
```

Arguments

n	Number of simulations.
horizon	Time steps (e.g., 252 for daily data).
freq	Frequency ("daily", "weekly", "monthly").
S0	Initial price (default=100).
V0	Initial volatility (default=0.04).
r0	Risk-free rate (default=0.02).
kappa	Mean reversion speed (must satisfy Feller: $2 * \kappa * \theta \geq \text{volvol}^2$).
theta	Long-run mean volatility.
volvol	Volatility of volatility (must satisfy Feller).
rho	Leverage effect (correlation between shocks).
lambda	Jump intensity.
mu_J	Mean jump size.
sigma_J	Std dev of jump size.
start_	Starting time (optional), better used with numeric frequency
...	Additional parameters to be passed to simdiff or simshocks

Value

A time series object (ts).

rsvjdg2plus	<i>Simulate SVJD model (with Feller condition check) with G2++ short rates model</i>
-------------	--

Description

Simulate SVJD model (with Feller condition check) with G2++ short rates model

Usage

```
rsvjdg2plus(
  n = 10,
  horizon = 5,
  freq = "quarterly",
  S0 = 100,
  rho_stock_vol = -0.7,
  rho_g2plus = -0.99855687,
  ...
)
```

Arguments

n	Number of simulations.
horizon	Time steps (e.g., 252 for daily data).
freq	Frequency ("daily", "weekly", "monthly").
S0	Initial stock price (default=100).
rho_stock_vol	Correlation stock price / stochastic volatility
rho_g2plus	Correlation between G2++ factors
...	Additional parameters to be passed to rsvjd and rg2plus

Value

A of time series objects (ts).

sim_to_zoo	<i>Convert Simulation Output to a zoo Time Series</i>
------------	---

Description

Converts the output of 'simdiff()' or 'simshocks()' (typically a 'ts' object or matrix) into a 'zoo' time series with proper calendar-based indexing.

Usage

```
sim_to_zoo(sim_data, start_ = NULL, frequency = NULL, end_ = NULL)
```

Arguments

sim_data	A time series ('ts'), matrix, or vector representing the simulation output.
start_	Optional. The starting point of the time index, as a 'Date' string ("YYYY-MM-DD") or a numeric cycle ('c(year, period)'). If 'sim_data' is a 'ts' object and 'start_' is 'NULL', the function will attempt to infer the start date automatically.
frequency	Optional. The frequency of the time series. Can be numeric (e.g., '12', '4', '260') or character ("monthly", "quarterly", "daily", etc.). Inferred from 'sim_data' if missing.
end_	Optional. A date string or cycle indicating the end of the time series. If provided, it overrides the number of observations used to construct the time index.

Details

The function supports flexible time specification using either a 'Date' string (e.g. "2025-01-01") or a cycle notation (e.g. 'c(2025, 4)' for 4th quarter of 2025), along with a frequency that can be numeric (e.g. '12', '4', '260') or string-based ("monthly", "quarterly", etc.).

Value

A 'zoo' object with the same data as 'sim_data' and a time index based on calendar dates.

simdiff	<i>Simulation of diffusion processes</i>
---------	--

Description

Simulates various diffusion processes including Geometric Brownian Motion (GBM), Cox-Ingersoll-Ross (CIR), and Ornstein-Uhlenbeck (OU) processes.

Usage

```
simdiff(
  n,
  horizon,
  frequency = c("annual", "semi-annual", "quarterly", "monthly", "weekly", "daily"),
  model = c("GBM", "CIR", "OU"),
  x0,
  theta1 = NULL,
  theta2 = NULL,
  theta3 = NULL,
  lambda = NULL,
  mu_z = NULL,
  sigma_z = NULL,
  p = NULL,
  eta_up = NULL,
  eta_down = NULL,
  eps = NULL,
  start_ = NULL,
  end_ = NULL,
  seed = 123,
  return_zoo = FALSE
)
```

Arguments

n	Number of simulations/scenarios
horizon	Time horizon for simulation
frequency	Frequency of observations: if numeric (as for ts objects), better used with start_ and end_. Otherwise a string, either "annual", "semi-annual", "quarterly", "monthly", "weekly", or "daily"
model	Type of diffusion process: "GBM", "CIR", or "OU"
x0	Initial value
theta1	First parameter (drift for GBM, mean reversion for CIR/OU)
theta2	Second parameter (volatility for GBM, long-term mean for CIR/OU)
theta3	Third parameter (optional, used for some models)
lambda	Jump intensity parameter (optional)

mu_z	Mean of jump size (optional)
sigma_z	Standard deviation of jump size (optional)
p	Probability parameter (optional)
eta_up	Upward jump size (optional)
eta_down	Downward jump size (optional)
eps	Pre-generated random shocks (optional)
start_	Starting time (optional), better used with numeric frequency
end_	Ending time (optional), better used with numeric frequency
seed	Random seed for reproducibility
return_zoo	Boolean; Either return a zoo object or not

Details

The function simulates various diffusion processes with different frequencies. For GBM, theta1 represents the drift and theta2 the volatility. For CIR and OU processes, theta1 is the mean reversion speed and theta2 is the long-term mean.

Value

A time series object containing the simulated paths

Examples

```
# Simulate GBM process
eps <- simshocks(n = 10, horizon = 5, frequency = "quarterly")
sim_GBM <- simdiff(n = 10, horizon = 5, frequency = "quarterly",
                  model = "GBM", x0 = 100, theta1 = 0.03, theta2 = 0.1,
                  eps = eps)

# Simulate CIR process
sim_CIR <- simdiff(n = 10, horizon = 5, frequency = "quarterly",
                  model = "CIR", x0 = 0.04, theta1 = 1.5, theta2 = 0.04,
                  theta3 = 0.2, eps = eps)
```

simshocks

Simulation of Gaussian shocks for risk factors

Description

Generates Gaussian shocks for simulating risk factors using various methods including classic Monte Carlo, antithetic variates, moment matching, and hybrid approaches.

Usage

```

simshocks(
  n,
  horizon,
  frequency = c("annual", "semi-annual", "quarterly", "monthly", "weekly", "daily"),
  method = c("classic", "antithetic", "mm", "hybridantimm", "TAG"),
  family = NULL,
  par = NULL,
  par2 = rep(0, length(par)),
  RVM = NULL,
  type = c("CVine", "DVine", "RVine"),
  start_ = NULL,
  end_ = NULL,
  seed = 123,
  return_zoo = FALSE
)

```

Arguments

n	Number of simulations/scenarios
horizon	Time horizon for simulation
frequency	Frequency of observations: if numeric (as for ts objects), better used with start_ and end_. Otherwise a string, either "annual", "semi-annual", "quarterly", "monthly", "weekly", or "daily"
method	Simulation method: "classic", "antithetic", "mm" (moment matching), "hybridantimm" (hybrid antithetic-moment matching), or "TAG"
family	Vector of copula families (optional)
par	Vector of copula parameters (optional)
par2	Vector of second copula parameters (optional)
RVM	RVineMatrix object for R-vine copula (optional)
type	Type of vine copula: "CVine", "DVine", or "RVine"
start_	Starting time (optional), better used with numeric frequency
end_	Ending time (optional), better used with numeric frequency
seed	Random seed for reproducibility
return_zoo	Boolean; Either return a zoo object or not

Details

The function generates Gaussian shocks that can be used for simulating various risk factors. Different simulation methods are available:

- "classic": Standard Monte Carlo simulation
- "antithetic": Uses antithetic variates to reduce variance
- "mm": Moment matching to improve statistical properties

- "hybridantimm": Combines antithetic variates and moment matching
- "TAG": TAG method

When using copulas (by specifying family and par), the function supports C-vine, D-vine, and R-vine copula structures.

Value

A time series object containing the simulated Gaussian shocks

Examples

```
# Generate shocks using classic Monte Carlo
shocks <- simshocks(n = 1000, horizon = 5, frequency = "quarterly")

# Generate shocks using antithetic variates
shocks_antithetic <- simshocks(n = 1000, horizon = 5, frequency = "quarterly",
                               method = "antithetic")

# Generate shocks using C-vine copula
#shocks_copula <- simshocks(n = 1000, horizon = 5, frequency = "quarterly",
#                           family = c(1, 1), par = c(0.5, 0.3),
#                           type = "CVine")
```

simulate_rvine	<i>Enhanced R-vine copula simulation with proper distribution preservation</i>
----------------	--

Description

Simulates multivariate time series data using R-vine copulas while preserving the dependence structure and marginal distributions of the original data.

Usage

```
simulate_rvine(
  data,
  n = 100,
  seed = 123,
  verbose = FALSE,
  n_trials = 10,
  oversample_factor = 1.5,
  score_weights = c(0.4, 0.2, 0.2, 0.1, 0.1)
)
```

Arguments

data	Matrix or data.frame of multivariate time series
n	Number of simulations to generate
seed	Random seed for reproducibility
verbose	Whether to print fitting information
n_trials	Number of trials to select best fit
oversample_factor	Factor to oversample for better selection (default 1.5)
score_weights	Vector of weights for scoring criteria. Must be length 5 and sum to 1. Order: [Kendall_cor, Pearson_cor, KS_test, mean_error, sd_error]

Value

A list of class 'rvine_simulation' containing:

- simulated_data - The simulated multivariate time series
- diagnostics - Comprehensive diagnostic information including:
 - Correlation matrices and errors
 - Distribution comparison metrics
 - Quality scores and trial results
 - Model information

Examples

```
## Not run:
# Basic usage with default weights
result <- simulate_rvine(uschange, n = 200)

# Custom weights emphasizing correlation preservation
result <- simulate_rvine(uschange, n = 200,
                        score_weights = c(0.6, 0.2, 0.1, 0.05, 0.05))

# Plot results
plot(result)

## End(Not run)
```

 ycextra

Yield curve or zero-coupon prices extrapolation

Description

Yield curve or zero-coupon bonds prices curve extrapolation using the Nelson-Siegel, Svensson, Smith-Wilson models.

Usage

```
ycextra(yM = NULL, p = NULL, matsin, matsout,
        method = c("NS", "SV", "SW"),
        typeres = c("rates", "prices"), UFR, T_UFR = NULL)
```

Arguments

yM	A vector of non-negative numerical quantities, containing the yield to maturities.
p	A vector of non-negative numerical quantities, containing the zero-coupon prices.
matsin	A vector containing the observed maturities.
matsout	the output maturities needed.
method	A character string giving the type of method used for interpolation and extrapolation. method can be either "NS" for Nelson-Siegel, "SV" for Svensson, or "SW" Smith-Wilson.
typeres	A character string, giving the type of return. Either "prices" or "rates".
UFR	The ultimate forward rate.
T_UFR	The number of years after which the yield curve converges to the UFR. T_UFR is used only when method is "SW".

Details

This function interpolates between observed points of a yield curve, or zero-coupon prices, and extrapolates the curve using the Nelson-Siegel, Svensson, Smith-Wilson models. The result can be either prices or zero rates. For the purpose of extrapolation, an ultimate forward rate (UFR) to which the yield curve converges must be provided. With the Smith-Wilson method, a period of convergence (number of years) to the ultimate forward rate, after the last liquid point, must be provided.

Value

An S4 Object

Author(s)

Thierry Moudiki

Examples

```
# Yield to maturities
txZC <- c(0.01422, 0.01309, 0.01380, 0.01549, 0.01747, 0.01940, 0.02104, 0.02236, 0.02348,
0.02446, 0.02535, 0.02614, 0.02679, 0.02727, 0.02760, 0.02779, 0.02787, 0.02786, 0.02776
, 0.02762, 0.02745, 0.02727, 0.02707, 0.02686, 0.02663, 0.02640, 0.02618, 0.02597, 0.02578, 0.02563)

# Prices
p <- c(0.9859794, 0.9744879, 0.9602458, 0.9416551, 0.9196671, 0.8957363, 0.8716268, 0.8482628,
0.8255457, 0.8034710, 0.7819525, 0.7612204, 0.7416912, 0.7237042, 0.7072136
, 0.6922140, 0.6785227, 0.6660095, 0.6546902, 0.6441639, 0.6343366, 0.6250234, 0.6162910, 0.6080358,
0.6003302, 0.5929791, 0.5858711, 0.5789852, 0.5722068, 0.5653231)
```

```

# Observed maturities
u <- 1:30

# Output maturities
t <- seq(from = 1, to = 60, by = 0.5)

# Svensson extrapolation
(yc <- ycextra(p = p, matsin = u, matsout = t,
method="SV", typeres="prices", UFR = 0.018))

#Smith-Wilson extrapolation
(yc <- ycextra(p = p, matsin = u, matsout = t,
method="SW", typeres="rates", UFR = 0.019, T_UFR = 20))

# Nelson-Siegel extrapolation
(yc <- ycextra(yM = txZC, matsin = u, matsout = t,
method="NS", typeres="prices", UFR = 0.029))

```

ycinter

*Yield curve or zero-coupon prices interpolation***Description**

Yield curve or zero-coupon bonds prices curve interpolation using the Nelson-Siegel , Svensson, Smith-Wilson models and an Hermite cubic spline.

Usage

```

ycinter(yM = NULL, p = NULL, matsin, matsout,
method = c("NS", "SV", "SW", "HCSPL"),
typeres = c("rates", "prices"))

```

Arguments

yM	A vector of non-negative numerical quantities, containing the yield to maturities.
p	A vector of non-negative numerical quantities, containing the zero-coupon prices.
matsin	A vector containing the observed maturities.
matsout	the output maturities needed.
method	A character string giving the type of method used fo interpolation. method can be either "NS" for Nelson-Siegel, "SV" for Svensson, "HCSPL" for Hermite cubic spline, or "SW" Smith-Wilson.
typeres	A character string, giving the type of return. Either "prices" or "rates".

Details

This function interpolates between observed points of a yield curve, or zero-coupon prices, using the Nelson-Siegel, Svensson, Smith-Wilson models and an Hermite cubic spline. The result can be either prices or zero rates.

Value

An S4 Object

Author(s)

Thierry Moudiki

Examples

```
## Interpolation of yields to maturities with prices as outputs

# Yield to maturities
txZC <- c(0.01422,0.01309,0.01380,0.01549,0.01747,0.01940,0.02104,0.02236,0.02348,
0.02446,0.02535,0.02614,0.02679,0.02727,0.02760,0.02779,0.02787,0.02786,0.02776
,0.02762,0.02745,0.02727,0.02707,0.02686,0.02663,0.02640,0.02618,0.02597,0.02578,0.02563)

# Zero-coupon prices
p <- c(0.9859794,0.9744879,0.9602458,0.9416551,0.9196671,0.8957363,0.8716268,0.8482628,
0.8255457,0.8034710,0.7819525,0.7612204,0.7416912,0.7237042,0.7072136
,0.6922140,0.6785227,0.6660095,0.6546902,0.6441639,0.6343366,0.6250234,0.6162910,0.6080358,
0.6003302,0.5929791,0.5858711,0.5789852,0.5722068,0.5653231)

# Observed maturities
u <- 1:30

# Output maturities
t <- seq(from = 1, to = 30, by = 0.5)

# Cubic splines interpolation
(yc <- ycinter(yM = txZC, matsin = u, matsout = t,
method="HCSPL", typeres="rates"))

# Nelson-Siegel interpolation
(yc <- ycinter(yM = txZC, matsin = u, matsout = t,
method="NS", typeres="prices"))

# Svensson interpolation
(yc <- ycinter(p = p, matsin = u, matsout = t,
method="SV", typeres="prices"))

#Smith-Wilson interpolation
(yc <- ycinter(p = p, matsin = u, matsout = t,
method="SW", typeres="rates"))
```

Index

- * **curve**
 - forwardrates, [13](#)
 - ycextra, [27](#)
 - ycinter, [29](#)
- * **extrapolation,**
 - forwardrates, [13](#)
 - ycextra, [27](#)
- * **interpolation,**
 - forwardrates, [13](#)
 - ycinter, [29](#)
- * **yield**
 - forwardrates, [13](#)
 - ycextra, [27](#)
 - ycinter, [29](#)

calculate_V, [2](#)
calculatereturns, [3](#)

enforce_monotonicity, [4](#)
esgctest, [4](#)
esgdiscountfactor, [5](#)
esgfwdrates, [6](#)
esgmartingaletest, [7](#)
esgmccv, [9](#)
esgmcprices, [10](#)
esgplotbands, [11](#)
esgplotshocks, [12](#)
esgplots, [13](#)

forwardrates, [13](#)

generate_diverse_sv_paths, [14](#)
generate_svjd, [15](#)
generate_synthetic_returns, [16](#)

plot.rvine_simulation, [17](#)
print.rvine_simulation, [18](#)

rg2plus, [18](#), [22](#)
rpriceg2plus, [19](#)
rsvjd, [20](#), [22](#)
rsvjdg2plus, [21](#)
sim_to_zoo, [22](#)
simdiff, [23](#)
simshocks, [24](#)
simulate_rvine, [26](#)
ycextra, [13](#), [14](#), [27](#)
ycinter, [13](#), [14](#), [29](#)