Package 'finlabR'

May 8, 2026

Type Package

Title Portfolio Analytics and Simulation Toolkit

Version 1.0.0

Description Tools for portfolio construction and risk analytics, including mean-variance optimization, conditional value at risk (expected shortfall) minimization, risk parity, regime clustering, correlation analysis, Monte Carlo simulation, and option pricing. Includes utilities for portfolio evaluation, clustering, and risk reporting. Methods are based in part on Markowitz (1952) <doi:10.1111/j.1540-6261.1952.tb01525.x>, Rockafellar and Uryasev (2000) <doi:10.21314/JOR.2000.038>, Maillard et al. (2010) <doi:10.3905/jpm.2010.36.4.060>, Black and Scholes (1973) <doi:10.1086/260062>, and Cox et al. (1979) <doi:10.1016/0304-405X(79)90015-1>.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Imports stats, utils, quadprog, ggplot2, PerformanceAnalytics, zoo, class, quantmod, reshape2, mclust, shiny

Suggests bslib, TTR, DT, xts, yfR, cryptoQuotes, tidyquant, Rtsne, umap, testthat, knitr, rmarkdown

VignetteBuilder knitr

RoxygenNote 7.3.3

NeedsCompilation no

Author Suyash Jindal [aut, cre]

Maintainer Suyash Jindal <jindalsuyash7@gmail.com>

Depends R (>= 3.5.0)

Repository CRAN

Date/Publication 2026-04-22 13:20:14 UTC

Contents

`american_option_binomial` *American Option Pricing via Binomial Tree*

Description

Extends the CRR binomial tree with early exercise at each node. Correctly prices American puts and calls.

Usage

```
american_option_binomial(
  S,
  K,
  time_to_expiry,
  r,
  sigma,
  n = 200,
  type = "put",
  q = 0
)
```

Arguments

`S`	Current stock price.
`K`	Strike price.
`time_to_expiry`	Time to expiry (years).
`r`	Risk-free rate (annual, continuous).
`sigma`	Volatility (annual).
`n`	Number of time steps. Default 200.
`type`	"call" or "put". American puts are commonly priced here.
`q`	Continuous dividend yield. Default 0.

Value

A list: price, early_exercise_boundary, european_price, early_premium.

Examples

```
# American put (early exercise premium should be positive for ITM put)
am <- american_option_binomial(S = 100, K = 105, time_to_expiry = 1,
                               r = 0.05, sigma = 0.25, n = 200,
                               type = "put")
cat("American put:", am$price, "  European put:", am$european_price,
    "  Early premium:", am$early_premium, "\n")
```

---

annualize_returns *Annualise Returns*

Description

Annualise Returns

Usage

```
annualize_returns(returns, freq = 252, type = "geometric")
```

Arguments

returns	Numeric. Per-period returns.
freq	Integer. Periods per year. Default 252.
type	Character. "arithmetic" or "geometric". Default "geometric".

Value

Numeric. Annualised return.

---

asset_clustering *Asset Clustering with Optional PCA Reduction*

Description

Groups assets by similarity of their return distributions using k-means or Gaussian mixture EM, with an optional PCA dimensionality reduction step applied before clustering.

Usage

```
asset_clustering(
  returns,
  method = c("kmeans", "em"),
  k = 3,
  reduce = c("pca", "none"),
  n_components = 3,
  seed = 123
)
```

Arguments

returns	Matrix or data.frame of returns (rows = time, cols = assets).
method	Character. "kmeans" (default) or "em".
k	Integer. Number of clusters. Default 3.
reduce	Character. Dimensionality reduction: "pca" (default) or "none".
n_components	Integer. Number of PCA components to retain. Default 3.
seed	Integer. Random seed. Default 123.

Value

A list with elements clusters (named integer vector), model (fitted model object), and pca (the prcomp object, or NULL if reduce = "none").

Examples

```
set.seed(2)
R <- matrix(rnorm(300 * 6, 0.0003, 0.01), 300, 6)
colnames(R) <- paste0("A", 1:6)
res <- asset_clustering(R, method = "kmeans", k = 3)
res$clusters
```

---

asset_correlation *Correlation Analysis Across Asset Groups*

Description

Computes a Pearson, Spearman, or Kendall correlation matrix from either a single return matrix or a named list of return matrices (one per asset group). When a list is supplied, matrices are column-bound and column names are prefixed with the group index.

Usage

```
asset_correlation(returns_list, method = c("pearson", "spearman", "kendall"))
```

Arguments

returns_list	A numeric matrix/data.frame of returns (T x N), OR a named list of such matrices — one element per asset group.
method	Character. Correlation method: "pearson" (default), "spearman", or "kendall".

Value

A symmetric numeric correlation matrix (N x N).

Examples

```
set.seed(1)
R <- matrix(rnorm(300 * 4, 0.0003, 0.01), 300, 4)
colnames(R) <- c("EQ1", "EQ2", "CMD1", "BOND1")
asset_correlation(R)
```

`asset_correlation_matrix`

Compute Cross-Asset Correlation Matrix

Description

Computes the Pearson, Spearman, or Kendall correlation matrix across multiple asset classes (equities, commodities, crypto, bonds).

Usage

```
asset_correlation_matrix(
  returns,
  method = "pearson",
  use_ = "pairwise.complete.obs"
)
```

Arguments

`returns`	Numeric matrix (T x N) of asset returns.
`method`	Character. "pearson", "spearman", or "kendall". Default "pearson".
`use_`	Character. Passed to `cor()`. Default "pairwise.complete.obs".

Value

A list: cor_matrix, p_values, labels.

Examples

```
set.seed(42)
R <- matrix(rnorm(300 * 6, 0.0003, 0.01), 300, 6)
colnames(R) <- c("SPY","GLD","BTC","OIL","TLT","ETH")
res <- asset_correlation_matrix(R)
round(res$cor_matrix, 3)
```

binomial_tree_option *Binomial Tree Option Pricing (European)*

Description

Uses the Cox-Ross-Rubinstein (CRR) binomial model.

Usage

```
binomial_tree_option(S, K, time_to_expiry, r, sigma, n = 100, type = "call")
```

Arguments

S	Current stock price.
K	Strike price.
time_to_expiry	Time to expiry (years).
r	Risk-free rate (annual, continuous).
sigma	Volatility (annual).
n	Number of time steps. Default 100.
type	"call" or "put".

Value

A list: price, u, d, p, tree (stock price tree), option_tree.

Examples

```
binomial_tree_option(S = 100, K = 100, time_to_expiry = 1,
                     r = 0.05, sigma = 0.20, n = 50)
```

bootstrap_returns *Bootstrap Returns*

Description

Generates a bootstrap distribution of portfolio statistics.

Usage

```
bootstrap_returns(
  returns,
  weights = NULL,
  stat_fn = NULL,
  n_boot = 1000,
  block_size = 1,
  freq = 252,
  seed = 42
)
```

Arguments

returns	Numeric vector or matrix of returns.
weights	Optional weight vector. Default equal weight.
stat_fn	Function applied to each bootstrap sample. Default: mean * freq.
n_boot	Integer. Number of bootstrap replicates. Default 1000.
block_size	Integer. Block bootstrap size (1 = iid). Default 1.
freq	Integer. Default 252.
seed	Integer. Default 42.

Value

A list: boot_stat, mean, se, ci_95.

---

bs_option_price *Black-Scholes Option Price*

Description

Black-Scholes Option Price

Usage

```
bs_option_price(S, K, time_to_expiry, r, sigma, type = "call", q = 0)
```

Arguments

S	Current stock price.
K	Strike price.
time_to_expiry	Time to expiry in years.
r	Risk-free rate (continuous, annualised).
sigma	Volatility (annualised).
type	"call" or "put". Default "call".
q	Dividend yield. Default 0.

Value

Numeric. Option price.

Examples

```
bs_option_price(100, 100, 1, 0.05, 0.20, "call")
bs_option_price(100, 105, 0.5, 0.04, 0.25, "put")
```

`calc_returns` *Compute Asset Returns from a Price Series*

Description

Converts a matrix or data frame of asset prices into a matrix of period returns. The first row is dropped (no prior price to diff against), so the result has one fewer row than the input.

Usage

```
calc_returns(prices, method = c("log", "simple"), na.rm = TRUE)
```

Arguments

`prices`	Numeric matrix, data.frame, xts, or zoo of prices (T x N). Remove any non-numeric columns (e.g. a `date` column) before calling: `calc_returns(prices[, -1])`.
`method`	Character. Return type: `"log"` (default, continuously compounded) or `"simple"` (arithmetic).
`na.rm`	Logical. If `TRUE` (default), rows that still contain `NA` after differencing are dropped.

Value

A numeric matrix of returns with `nrow(prices) - 1` rows and `ncol(prices)` columns. Column names are preserved.

Examples

```
prices <- get_example_prices()
rets   <- calc_returns(prices[, -1])
rets_s <- calc_returns(prices[, -1], method = "simple")
dim(rets)
```

`clt_demonstration` *CLT Demonstration*

Description

Illustrates the Central Limit Theorem by drawing repeated samples of increasing size from a population and plotting the distribution of sample means against the theoretical Normal curve.

Usage

```
clt_demonstration(
  data,
  n_samples = c(5, 10, 30, 100),
  n_reps = 2000,
  seed = 123
)
```

Arguments

`data`	Numeric vector. Population data (e.g. historical returns).
`n_samples`	Integer vector. Sample sizes to test. Default `c(5, 10, 30, 100)`.
`n_reps`	Integer. Repetitions per sample size. Default 2000.
`seed`	Integer. Default 123.

Value

A list with:

`results`	data.frame of sample means and Shapiro-Wilk statistics.
`clt_plot`	A ggplot2 faceted histogram with Normal overlay.
`pop_mean`	Population mean.
`pop_sd`	Population standard deviation.

Examples

```
set.seed(1)
r <- rnorm(500, 0.0004, 0.012)
clt <- clt_demonstration(r)
clt$clt_plot
```

clt_pnl_ci *Evaluate Strategy PnL using CLT Confidence Intervals*

Description

Applies the Central Limit Theorem to a vector of simulated daily profit-and-loss values and returns the sample mean together with a symmetric confidence interval.

Usage

```
clt_pnl_ci(pnl_vector, confidence_level = 0.95)
```

Arguments

pnl_vector	Numeric vector of simulated daily PnL values.
confidence_level	Numeric. Desired confidence level. Default 0.95.

Value

A list with three elements: `mean_pnl`, `lower_ci`, and `upper_ci`.

Examples

```
set.seed(10)
pnl <- rnorm(252, mean = 50, sd = 300)
clt_pnl_ci(pnl, confidence_level = 0.95)
```

clt_sample_means *CLT Sample Means*

Description

Returns sample mean statistics for the CLT demonstration.

Usage

```
clt_sample_means(data, n, n_reps = 1000, seed = 42)
```

Arguments

data	Numeric vector.
n	Integer. Sample size.
n_reps	Integer. Replications. Default 1000.
seed	Integer. Default 42.

Value

Numeric vector of sample means.

`cluster_book_kmeans` *Cluster Order Book States using K-Means*

Description

Identifies market regimes by running k-means clustering on the four microstructure features produced by `extract_features()`. Features are z-score scaled before clustering.

Usage

```
cluster_book_kmeans(features, centers = 4)
```

Arguments

`features`	A data.frame of extracted features (output of `extract_features()`).
`centers`	Integer. Number of clusters (regimes). Default 4.

Value

A list with two elements: `model` (the `kmeans` object) and `data` (the input data frame with a new `regime` column).

Examples

```
set.seed(7)
book <- extract_features(simulate_orderbook(500))
result <- cluster_book_kmeans(book, centers = 3)
table(result$data$regime)
```

`cluster_summary` *Cluster Summary*

Description

Cluster Summary

Usage

```
cluster_summary(cl_obj)
```

Arguments

cl_obj	Output from asset_clustering().

Value

Data.frame of cluster statistics.

---

compute_efficient_frontier

Compute the Efficient Frontier

Description

Solves a sequence of minimum-variance QP problems (via quadprog) across a grid of target returns to trace the efficient frontier. Returns and risk are annualised.

Usage

```
compute_efficient_frontier(
  returns,
  n_points = 100,
  allow_short = FALSE,
  risk_free = 0,
  freq = 252
)
```

Arguments

returns	Numeric matrix or xts of asset returns (T x N).
n_points	Integer. Number of frontier points. Default 100.
allow_short	Logical. Allow short selling? Default FALSE.
risk_free	Numeric. Risk-free rate (annualised). Default 0.
freq	Integer. Periods per year (252 = daily, 12 = monthly). Default 252.

Value

A list with:

frontier	data.frame of Return, Risk, Sharpe, and asset weights.
cov_matrix	Annualised sample covariance matrix.
mu	Annualised expected return vector.
assets	Asset names.

Examples

```
set.seed(42)
R <- matrix(rnorm(500 * 5, 0.0005, 0.01), 500, 5)
colnames(R) <- paste0("Asset", 1:5)
ef <- compute_efficient_frontier(R)
head(ef$frontier)
```

---

consistency_check *Consistency Check*

Description

Tests whether an estimator is consistent: SE decreases as n grows.

Usage

```
consistency_check(estimates_by_n)
```

Arguments

estimates_by_n	Named list where names are sample sizes and values are numeric vectors of estimates.

Value

A data.frame with SE by n and consistency result.

---

cross_asset_analysis *Cross-Asset Correlation Analysis*

Description

Comprehensive cross-asset study across equities, commodities, and crypto.

Usage

```
cross_asset_analysis(returns, asset_classes = NULL, window = 60, freq = 252)
```

Arguments

returns	Matrix (T x N) with named columns.
asset_classes	Named list mapping asset names to class labels, e.g. list(SPY="Equity", GLD="Commodity", BTC="Crypto").
window	Integer. Rolling window for rolling correlations. Default 60.
freq	Integer. Periods per year. Default 252.

Value

A list: static_cor, rolling_cors, within_class, between_class, plots.

`cross_validate_portfolio` *Time-Series Cross-Validation for Portfolio Models*

Description

Implements rolling-window and expanding-window cross-validation for portfolio construction, measuring out-of-sample Sharpe and consistency / unbiasedness of estimates.

Usage

```
cross_validate_portfolio(
  returns,
  model_fn,
  n_folds = 5,
  window_type = "expanding",
  min_train = 60,
  eval_period = 21,
  freq = 252
)
```

Arguments

`returns`	Numeric matrix (T x N) of returns.
`model_fn`	Function taking a training returns matrix and returning a weight vector. E.g. `function(R) min_variance_portfolio(R)$weights`.
`n_folds`	Integer. Number of CV folds. Default 5.
`window_type`	Character. "rolling" or "expanding". Default "expanding".
`min_train`	Integer. Minimum training observations. Default 60.
`eval_period`	Integer. Evaluation (test) window per fold. Default 21.
`freq`	Integer. Periods per year. Default 252.

Value

A list: fold_results, oos_sharpe, consistency, unbiasedness.

Examples

```
set.seed(42)
R <- matrix(rnorm(500 * 4, 0.0003, 0.012), 500, 4)
colnames(R) <- paste0("A", 1:4)
cv <- cross_validate_portfolio(
  R,
  model_fn = function(r) min_variance_portfolio(r)$weights,
  n_folds  = 5
)
cat("OOS Sharpe:", cv$oos_sharpe, "\n")
```

---

cvar_frontier *CVaR-Return Frontier*

Description

Sweeps over target returns to build a CVaR-efficient frontier.

Usage

```
cvar_frontier(returns, alpha = 0.95, n_points = 30, freq = 252)
```

Arguments

returns	Numeric matrix of asset returns.
alpha	Confidence level. Default 0.95.
n_points	Integer. Number of frontier points. Default 30.
freq	Integer. Periods per year. Default 252.

Value

A data.frame with columns: Return, Risk, CVaR, VaR, Sharpe, weights.

---

cvar_minimize *CVaR-Minimising Portfolio (Softmax / Lightweight)*

Description

A simpler CVaR minimiser. For long-only portfolios the weights are parameterised through a softmax transformation so the simplex constraint is satisfied automatically (BFGS). For short-selling the weights are optimised directly with box constraints and a quadratic penalty for $\sum w = 1$ (L-BFGS-B).

Usage

```
cvar_minimize(
  returns,
  alpha = 0.95,
  allow_short = FALSE,
  bounds = c(-1, 1),
  penalty = 1000,
  max_iter = 500
)
```

Arguments

returns	Matrix/data.frame of asset returns (T x N).
alpha	Confidence level (e.g. 0.95). Default 0.95.
allow_short	Logical. Allow short positions. Default FALSE.
bounds	Length-2 numeric vector of [lower, upper] weight bounds when allow_short = TRUE. Default c(-1, 1).
penalty	Quadratic penalty weight for the sum-to-one constraint when allow_short = TRUE. Default 1000.
max_iter	Maximum iterations passed to stats::optim. Default 500.

Value

A list with elements:

weights	Named numeric vector of portfolio weights.
cvar	Scalar CVaR value at the optimal portfolio.

Examples

```
set.seed(7)
R <- matrix(rnorm(400 * 4, 0.0003, 0.01), 400, 4)
colnames(R) <- c("A", "B", "C", "D")
res <- cvar_minimize(R)
res$weights
res$cvar
```

---

detect_regimes *Detect Regimes (General Interface)*

Description

Detect Regimes (General Interface)

Usage

```
detect_regimes(returns, method = "kmeans", k = 3, ...)
```

Arguments

returns	Numeric vector or matrix column of returns.
method	Character. "kmeans" or "em". Default "kmeans".
k	Integer. Number of regimes.
...	Additional args passed to the method function.

Value

Result list from the chosen method.

---

download_prices *Download Prices via quantmod*

Description

Wraps `quantmod::getSymbols` for clean price downloads.

Usage

```
download_prices(
  symbols,
  from = "2020-01-01",
  to = Sys.Date(),
  src = "yahoo",
  return_type = "returns"
)
```

Arguments

symbols	Character vector of ticker symbols.
from	Date string "YYYY-MM-DD". Default "2020-01-01".
to	Date string. Default today.
src	Character. Data source. Default "yahoo".
return_type	Character. "prices" or "returns". Default "returns".

Value

An xts object.

Examples

```
prices <- download_prices(c("SPY","GLD","BTC-USD"), from="2021-01-01")
```

embedding_2d *2D Embedding for Visualisation*

Description

Projects a numeric matrix down to two dimensions using PCA, UMAP, or t-SNE for visual exploration of asset or regime clusters.

Usage

```
embedding_2d(X, method = c("pca", "umap", "tsne"), seed = 123)
```

Arguments

X	Numeric matrix (N x D).
method	Character. "pca" (default), "umap", or "tsne".
seed	Integer. Random seed. Default 123.

Value

A numeric matrix with two columns (N x 2).

Examples

```
set.seed(3)
X <- matrix(rnorm(100 * 5), 100, 5)
emb <- embedding_2d(X, method = "pca")
plot(emb, pch = 19, col = "steelblue")
```

em_clustering *EM (Gaussian Mixture) Clustering*

Description

Fits a Gaussian Mixture Model via Expectation-Maximisation using the **mclust** package. Automatically selects the number of components using BIC if k = NULL.

Usage

```
em_clustering(X, k = 3, scale_ = TRUE, max_k = 8)
```

Arguments

X	Numeric matrix (T x P) of features.
k	Integer or NULL. Number of components. If NULL, auto-selects. Default 3.
scale_	Logical. Standardise features. Default TRUE.
max_k	Integer. Max components to try when auto-selecting. Default 8.

Value

A list: labels, probabilities, bic, means, covariances, model.

Examples

```
if (requireNamespace("mclust", quietly = TRUE)) {
  ok <- tryCatch({
    mclust::Mclust(matrix(rnorm(20), 10, 2), G = 2, verbose = FALSE)
    TRUE
  }, error = function(e) FALSE)
  if (ok) {
    set.seed(42)
    X <- rbind(matrix(rnorm(100*2, c(-2,0), 0.5), 100),
               matrix(rnorm(100*2, c( 2,0), 0.5), 100))
    res <- em_clustering(X, k = 2)
    table(res$labels)
  }
}
```

---

em_regime *EM Algorithm (Gaussian Mixture) Regime Detection*

Description

Uses Expectation-Maximisation to fit a Gaussian Mixture Model on rolling features. Requires the **mclust** package.

Usage

```
em_regime(returns, k = 3, window = 21, freq = 252)
```

Arguments

returns	Numeric vector of returns.
k	Integer. Number of components. Default 3.
window	Integer. Rolling window. Default 21.
freq	Integer. Periods per year. Default 252.

Value

A list: labels, probabilities, model, stats.

Examples

```
if (requireNamespace("mclust", quietly = TRUE)) {
  ok <- tryCatch({
    mclust::Mclust(matrix(rnorm(20), 10, 2), G = 2, verbose = FALSE)
    TRUE
  }, error = function(e) FALSE)
  if (ok) {
    set.seed(1)
    r <- c(rnorm(200, 0.001, 0.01), rnorm(150, -0.002, 0.02))
    res <- em_regime(r, k = 2)
  }
}
```

---

equal_risk_contribution

Equal Risk Contribution (Risk Parity) Portfolio

Description

Solves the equal-risk-contribution problem using cyclical coordinate descent. Supports custom risk budgets. Each asset (by default) contributes equally to total portfolio risk.

Usage

```
equal_risk_contribution(
  returns,
  budget = NULL,
  freq = 252,
  tol = 1e-08,
  max_iter = 1000
)
```

Arguments

returns	Numeric matrix (T x N) of asset returns.
budget	Numeric vector of risk budget weights (default: equal, length N). Will be normalised to sum to 1.
freq	Integer. Periods per year. Default 252.
tol	Numeric. Convergence tolerance. Default 1e-8.
max_iter	Integer. Maximum iterations. Default 1000.

Value

A list with elements: weights, risk_contrib, risk, return, budget, iterations, method.

Examples

```
set.seed(42)
R <- matrix(rnorm(300 * 4, 0.0003, 0.01), 300, 4)
colnames(R) <- c("SPY", "GLD", "TLT", "VNQ")
rp <- equal_risk_contribution(R)
rp$weights
```

---

example_prices *Example synthetic price dataset*

Description

Example synthetic price dataset

Usage

```
example_prices
```

Format

A data frame with 1000 rows (dates) and 6 assets. Columns: date, EQ1, EQ2, EQ3, CMD1, CRYPTO1, BOND1

---

extract_features *Extract Market Microstructure Features*

Description

Appends derived microstructure columns — spread, order-flow imbalance, total depth, and a volatility proxy — to an order book snapshot data frame.

Usage

```
extract_features(book)
```

Arguments

book	A data.frame produced by `simulate_orderbook()`.

Value

The input data.frame with four additional columns: `spread`, `imbalance`, `depth_total`, and `volatility`.

Examples

```
set.seed(1)
book <- simulate_orderbook(200)
book <- extract_features(book)
head(book[, c("spread", "imbalance", "volatility")])
```

`fetch_yahoo_prices` *Fetch Yahoo Finance close prices (wrapper around quantmod)*

Description

Fetch Yahoo Finance close prices (wrapper around quantmod)

Usage

```
fetch_yahoo_prices(symbols, from, to)
```

Arguments

`symbols`	Character vector of tickers.
`from`	Start date.
`to`	End date.

Value

xts object of close prices.

`format_weights` *Format Portfolio Weights*

Description

Prints a formatted weight table with percentages.

Usage

```
format_weights(weights, digits = 4)
```

Arguments

`weights`	Named numeric vector of portfolio weights.
`digits`	Integer. Decimal places. Default 4.

Value

Invisibly returns the formatted data.frame.

gaussian_mixture_em *Gaussian Mixture Model via EM (Diagonal Covariance)*

Description

Fits a Gaussian mixture model with diagonal covariance using the Expectation-Maximisation algorithm. Suitable for soft clustering of asset return features.

Usage

```
gaussian_mixture_em(X, k = 3, max_iter = 100, tol = 1e-06, seed = 123)
```

Arguments

X	Numeric matrix (N x D) of observations.
k	Integer. Number of mixture components. Default 3.
max_iter	Integer. Maximum EM iterations. Default 100.
tol	Numeric. Log-likelihood convergence tolerance. Default 1e-6.
seed	Integer. Random seed. Default 123.

Value

A list with elements `weights`, `means`, `variances`, `loglik`, and `cluster` (hard assignments via `which.max`).

Examples

```
set.seed(5)
X <- matrix(rnorm(200 * 3), 200, 3)
res <- gaussian_mixture_em(X, k = 2)
table(res$cluster)
```

gbm_simulation *Geometric Brownian Motion Simulation (Rich Output)*

Description

Simulates stock price paths under GBM with a choice of exact log-normal or Euler-Maruyama discretisation. Returns paths as a paths x time matrix together with terminal-price summary statistics.

Usage

```
gbm_simulation(
  S0 = 100,
  mu = 0.08,
  sigma = 0.2,
  time_horizon = 1,
  n_steps = 252,
  n_paths = 1000,
  seed = NULL,
  method = "exact"
)
```

Arguments

S0	Initial stock price. Default 100.
mu	Annual drift. Default 0.08.
sigma	Annual volatility. Default 0.20.
time_horizon	Time horizon in years. Default 1.
n_steps	Number of time steps. Default 252.
n_paths	Number of simulated paths. Default 1000.
seed	Integer seed. Default NULL.
method	Character. "exact" (default) or "euler".

Value

A list with:

paths	Numeric matrix (n_paths x n_steps+1) of simulated prices.
time_grid	Numeric vector of time points.
stats	Named list of terminal-price statistics.

Examples

```
sim <- gbm_simulation(S0 = 100, mu = 0.08, sigma = 0.20, n_paths = 500)
sim$stats$prob_profit
```

gd_max_sharpe *Gradient Descent Maximum Sharpe Portfolio*

Description

Maximises the Sharpe ratio using Adam-style gradient descent on the negative Sharpe objective.

Usage

```
gd_max_sharpe(
  returns,
  risk_free = 0.02,
  lr = 0.001,
  max_iter = 3000,
  tol = 1e-08,
  freq = 252,
  beta1 = 0.9,
  beta2 = 0.999,
  eps_adam = 1e-08
)
```

Arguments

returns	Numeric matrix of returns.
risk_free	Numeric. Risk-free rate. Default 0.02.
lr	Numeric. Adam learning rate. Default 0.001.
max_iter	Integer. Default 3000.
tol	Numeric. Default 1e-8.
freq	Integer. Default 252.
beta1	Numeric. Adam beta1. Default 0.9.
beta2	Numeric. Adam beta2. Default 0.999.
eps_adam	Numeric. Adam epsilon. Default 1e-8.

Value

A list: weights, loss_history, sharpe, risk, return.

gd_min_variance *Gradient Descent Minimum Variance Portfolio*

Description

Minimises portfolio variance using projected gradient descent with simplex projection (long-only constraint).

Usage

```
gd_min_variance(
  returns,
  lr = 0.01,
  max_iter = 2000,
  tol = 1e-08,
  freq = 252,
  momentum = 0.9,
  verbose = FALSE
)
```

Arguments

returns	Numeric matrix (T x N) of asset returns.
lr	Numeric. Learning rate. Default 0.01.
max_iter	Integer. Max iterations. Default 2000.
tol	Numeric. Convergence tolerance. Default 1e-8.
freq	Integer. Periods per year. Default 252.
momentum	Numeric. Momentum parameter (0 = vanilla GD). Default 0.9.
verbose	Logical. Print iteration info. Default FALSE.

Value

A list: weights, loss_history, convergence, risk, return.

Examples

```
set.seed(42)
R <- matrix(rnorm(300 * 5, 0.0003, 0.012), 300, 5)
colnames(R) <- paste0("A", 1:5)
gd <- gd_min_variance(R, lr = 0.05, max_iter = 1000)
gd$risk
```

get_example_prices *Example Price Data*

Description

Returns a data frame of simulated daily closing prices for six synthetic assets (three equities, one commodity, one crypto, one bond) covering approximately 1 000 trading days starting 2018-01-01.

Usage

```
get_example_prices()
```

Value

A data frame with columns `date`, `EQ1`, `EQ2`, `EQ3`, `CMD1`, `CRYPTO1`, and `BOND1`.

Examples

```
prices <- get_example_prices()
head(prices)
```

get_returns *Compute Returns from Prices*

Description

Compute Returns from Prices

Usage

```
get_returns(prices, type = "log", lag = 1)
```

Arguments

`prices`	Numeric vector, matrix, or xts of prices.
`type`	Character. "log" or "simple". Default "log".
`lag`	Integer. Return lag. Default 1.

Value

Returns of the same type as input.

Examples

```
prices <- c(100, 102, 101, 105, 103)
get_returns(prices)
get_returns(prices, type = "simple")
```

`gradient_descent` *Simple gradient descent optimizer*

Description

Simple gradient descent optimizer

Usage

```
gradient_descent(f, grad = NULL, x0, lr = 0.01, max_iter = 1000, tol = 1e-06)
```

Arguments

`f`	Objective function.
`grad`	Gradient function (optional).
`x0`	Initial parameter vector.
`lr`	Learning rate.
`max_iter`	Maximum iterations.
`tol`	Convergence tolerance.

Value

List with optimized parameters and history.

`gradient_descent_portfolio`
General Gradient Descent Portfolio (wrapper)

Description

General Gradient Descent Portfolio (wrapper)

Usage

```
gradient_descent_portfolio(returns, objective = "min_variance", ...)
```

Arguments

returns	Numeric matrix of returns.
objective	Character. "min_variance" or "max_sharpe". Default "min_variance".
...	Additional arguments passed to the objective function.

Value

List from the chosen optimisation.

---

kmeans_regime *K-Means Market Regime Detection*

Description

Clusters market observations into regimes using k-means on engineered features: rolling return, rolling volatility, and rolling Sharpe ratio.

Usage

```
kmeans_regime(
  returns,
  k = 3,
  window = 21,
  n_starts = 50,
  seed = 42,
  freq = 252
)
```

Arguments

returns	Numeric vector or single-column matrix of returns.
k	Integer. Number of regimes. Default 3.
window	Integer. Rolling window for feature computation. Default 21.
n_starts	Integer. k-means restarts. Default 50.
seed	Integer. Random seed. Default 42.
freq	Integer. Periods per year. Default 252.

Value

A list: labels, centers, features, stats, dates (if xts).

Examples

```
set.seed(42)
r <- c(rnorm(200, 0.001, 0.01),   # bull
       rnorm(100, -0.002, 0.025),  # bear
       rnorm(150, 0.0005, 0.008))  # low-vol
res <- kmeans_regime(r, k = 3)
table(res$labels)
```

---

knn_classify *kNN Classifier for Financial Signals*

Description

Trains a k-Nearest Neighbours classifier on financial features. Supports time-series aware train/test splitting.

Usage

```
knn_classify(X, y, k = 5, train_frac = 0.7, scale_ = TRUE, seed = 42)
```

Arguments

X	Numeric matrix (T x P) of features.
y	Factor or integer vector of labels.
k	Integer. Number of neighbours. Default 5.
train_frac	Numeric. Training set fraction. Default 0.7.
scale_	Logical. Standardise features. Default TRUE.
seed	Integer. Default 42.

Value

A list: predictions, actual, accuracy, confusion_matrix, k.

Examples

```
set.seed(42)
X <- matrix(rnorm(500 * 5), 500, 5)
y <- factor(ifelse(rowSums(X[, 1:2]) > 0, "Up", "Down"))
res <- knn_classify(X, y, k = 7)
res$accuracy
```

knn_money_flow *kNN Money Flow Analysis*

Description

Uses k-Nearest Neighbours to classify future market direction based on historical money-flow features (volume, price momentum, volatility). Provides a regime-based money flow signal.

Usage

```
knn_money_flow(
  returns,
  volume = NULL,
  k = 5,
  window = 10,
  horizon = 5,
  train_frac = 0.7,
  seed = 42
)
```

Arguments

returns	Numeric vector or matrix column of returns.
volume	Optional numeric vector of trading volume.
k	Integer. Number of nearest neighbours. Default 5.
window	Integer. Feature rolling window. Default 10.
horizon	Integer. Prediction horizon (steps ahead). Default 5.
train_frac	Numeric. Fraction for training. Default 0.7.
seed	Integer. Default 42.

Value

A list: predictions, accuracy, confusion_matrix, feature_importance.

Examples

```
set.seed(1)
r <- rnorm(500, 0.0003, 0.012)
res <- knn_money_flow(r, k = 7, horizon = 5)
cat("Accuracy:", res$accuracy, "\n")
```

knn_predict *k-Nearest Neighbors prediction*

Description

k-Nearest Neighbors prediction

Usage

```
knn_predict(train_X, train_y, new_X, k = 5)
```

Arguments

train_X	Training features matrix.
train_y	Training labels (numeric or factor).
new_X	New features matrix.
k	Number of neighbors.

Value

Predictions.

market_regime_kmeans *Market Regime Clustering with K-Means on Rolling Features*

Description

Detects market regimes by computing rolling distributional features (mean, volatility, skewness, kurtosis) from a return series and applying k-means clustering to the resulting feature matrix.

Usage

```
market_regime_kmeans(
  returns,
  k = 3,
  window = 60,
  features = c("mean", "vol"),
  seed = 123
)
```

Arguments

returns	Matrix, data.frame, or xts of returns (T x N).
k	Integer. Number of clusters (regimes). Default 3.
window	Integer. Rolling window length. Default 60.
features	Character vector of features to compute. Any subset of c("mean", "vol", "skew", "kurt"). Default c("mean", "vol").
seed	Integer. Random seed for reproducibility. Default 123.

Value

A list with elements features (feature matrix), kmeans (the fitted kmeans object), and labels (integer cluster assignments).

Examples

```
set.seed(1)
R <- matrix(rnorm(500 * 4, 0.0003, 0.01), 500, 4)
colnames(R) <- paste0("A", 1:4)
res <- market_regime_kmeans(R, k = 3, window = 60)
table(res$labels)
```

---

max_sharpe_portfolio *Maximum Sharpe Ratio Portfolio*

Description

Finds the tangency portfolio by minimising negative Sharpe ratio via L-BFGS-B optimisation with box constraints.

Usage

```
max_sharpe_portfolio(
  returns,
  risk_free = 0.02,
  freq = 252,
  allow_short = FALSE
)
```

Arguments

returns	Numeric matrix or xts of asset returns (T x N).
risk_free	Numeric. Annualised risk-free rate. Default 0.02.
freq	Integer. Periods per year. Default 252.
allow_short	Logical. Allow short selling? Default FALSE.

Value

A list: weights, return, risk, sharpe, method.

Examples

```
set.seed(1)
R <- matrix(rnorm(252 * 4, 0.0003, 0.012), 252, 4)
colnames(R) <- c("SPY", "GLD", "BTC", "BND")
ms <- max_sharpe_portfolio(R, risk_free = 0.05)
ms$sharpe
```

---

mc_price_simulation *Monte Carlo Price / Return Simulation*

Description

Simulates portfolio or single-asset cumulative return paths from a historical return series. Computes the terminal return distribution, CLT convergence statistics, and k-fold forward-chain cross-validation estimates of annualised return.

Usage

```
mc_price_simulation(
  returns,
  n_paths = 5000,
  horizon = 252,
  freq = 252,
  seed = 42,
  use_hist = FALSE,
  n_cv_folds = 5
)
```

Arguments

returns	Numeric vector of historical returns.
n_paths	Integer. Number of simulation paths. Default 5000.
horizon	Integer. Steps forward. Default 252.
freq	Integer. Periods per year. Default 252.
seed	Integer. Default 42.
use_hist	Logical. Bootstrap from historical returns (block resampling) instead of Normal draws. Default FALSE.
n_cv_folds	Integer. Number of forward-chain CV folds. Default 5.

Value

A list with elements: paths_matrix, terminal_returns, mu_historical, sg_historical, perc5, perc95, prob_profit, clt_stats (data.frame), cv_estimates (data.frame).

Examples

```
set.seed(1)
r <- rnorm(500, 0.0004, 0.012)
mc <- mc_price_simulation(r, n_paths = 1000, horizon = 252)
mc$prob_profit
```

---

mc_return_distribution *Monte Carlo Return Distribution Table*

Description

Extracts a percentile table of annualised terminal returns from the output of `mc_price_simulation()`.

Usage

```
mc_return_distribution(mc_obj)
```

Arguments

`mc_obj`	Output from `mc_price_simulation()`.

Value

A data.frame with columns Percentile and AnnReturn.

---

mc_statistics *Monte Carlo Statistics Summary*

Description

Prints a formatted console report of key statistics from `mc_price_simulation()` output, including CLT convergence and cross-validation estimates.

Usage

```
mc_statistics(mc_obj)
```

Arguments

`mc_obj`	Output from `mc_price_simulation()`.

Value

Invisibly returns `mc_obj`.

`minimize_cvar` *Minimise Portfolio CVaR (Multi-Restart)*

Description

Uses the Rockafellar-Uryasev linear programming reformulation of CVaR minimisation with multiple random restarts via `stats::optim`.

Usage

```
minimize_cvar(
  returns,
  alpha = 0.95,
  target_ret = NULL,
  allow_short = FALSE,
  freq = 252,
  n_restarts = 5
)
```

Arguments

`returns`	Numeric matrix (T x N) of asset returns.
`alpha`	Confidence level. Default 0.95.
`target_ret`	Optional numeric. Minimum target return constraint.
`allow_short`	Logical. Default FALSE.
`freq`	Integer. Periods per year. Default 252.
`n_restarts`	Integer. Number of random restarts. Default 5.

Value

A list: weights, CVaR, VaR, return, risk.

Examples

```
set.seed(1)
R <- matrix(rnorm(300 * 5, 0.0004, 0.011), 300, 5)
colnames(R) <- paste0("Asset", 1:5)
result <- minimize_cvar(R, alpha = 0.95)
result$CVaR
```

min_variance_portfolio *Global Minimum Variance Portfolio*

Description

Solves the unconstrained (or long-only) minimum-variance QP problem. Returns and risk are annualised.

Usage

```
min_variance_portfolio(returns, freq = 252, allow_short = FALSE)
```

Arguments

returns	Numeric matrix or xts of asset returns (T x N).
freq	Integer. Periods per year. Default 252.
allow_short	Logical. Allow short selling? Default FALSE.

Value

A list: weights, return, risk, sharpe, method.

Examples

```
set.seed(7)
R <- matrix(rnorm(252 * 6, 0.0002, 0.01), 252, 6)
colnames(R) <- paste0("A", 1:6)
mv <- min_variance_portfolio(R)
mv$risk
```

money_flow_knn *Money Flow Index + kNN signal*

Description

Money Flow Index + kNN signal

Usage

```
money_flow_knn(high, low, close, volume, k = 5, horizon = 1)
```

Arguments

high	High prices.
low	Low prices.
close	Close prices.
volume	Volumes.
k	Number of neighbors.
horizon	Forecast horizon for label.

Value

List with MFI series and kNN prediction.

---

monte_carlo_option *Monte Carlo Option Pricing*

Description

Prices European options via GBM simulation.

Usage

```
monte_carlo_option(
  S,
  K,
  time_to_expiry,
  r,
  sigma,
  n_sim = 1e+05,
  n_steps = 252,
  type = "call",
  seed = NULL
)
```

Arguments

S	Stock price.
K	Strike.
time_to_expiry	Time to expiry.
r	Risk-free rate.
sigma	Volatility.
n_sim	Integer. Number of simulations. Default 100000.
n_steps	Integer. Number of time steps. Default 252.
type	"call" or "put".
seed	Random seed for reproducibility. Default NULL.

Value

A list: price, std_error, conf_interval, paths (sample).

`mvo_efficient_frontier`

Efficient Frontier - Lightweight Scan (Raw Scale)

Description

Traces the efficient frontier by solving a target-return QP at each point. Works on raw (non-annualised) return scale. Prefer `compute_efficient_frontier()` for annualised, production-grade output.

Usage

```
mvo_efficient_frontier(
  returns,
  n = 50,
  rf = 0,
  allow_short = FALSE,
  target_returns = NULL
)
```

Arguments

`returns`	Matrix or data.frame of asset returns (T x N).
`n`	Integer. Number of frontier points. Default 50.
`rf`	Numeric. Risk-free rate for Sharpe. Default 0.
`allow_short`	Logical. Default FALSE.
`target_returns`	Optional numeric vector of specific target returns.

Value

A list: frontier (data.frame), weights (list), mu, Sigma.

Examples

```
set.seed(5)
R <- matrix(rnorm(300 * 4, 0.0003, 0.01), 300, 4)
colnames(R) <- c("A", "B", "C", "D")
ef <- mvo_efficient_frontier(R, n = 30)
head(ef$frontier)
```

mvo_max_sharpe *Maximum Sharpe Portfolio - Frontier Scan*

Description

Identifies the max-Sharpe portfolio by scanning the efficient frontier computed via `mvo_efficient_frontier()`. A simpler alternative to `max_sharpe_portfolio()` when a frontier object is already available.

Usage

```
mvo_max_sharpe(returns, rf = 0, allow_short = FALSE, n = 50)
```

Arguments

`returns`	Matrix or data.frame of asset returns (T x N).
`rf`	Numeric. Risk-free rate. Default 0.
`allow_short`	Logical. Default FALSE.
`n`	Integer. Number of frontier points. Default 50.

Value

A list: weights, mean (raw-scale), vol (raw-scale), sharpe.

Examples

```
set.seed(9)
R <- matrix(rnorm(300 * 4, 0.0003, 0.01), 300, 4)
colnames(R) <- c("A", "B", "C", "D")
mvo_max_sharpe(R)
```

mvo_min_variance *Minimum Variance Portfolio - Lightweight (Raw Scale)*

Description

A simpler GMV solver that works on raw (non-annualised) return scale. Uses quadprog directly with a ridge-regularised covariance matrix. Prefer `min_variance_portfolio()` for annualised results.

Usage

```
mvo_min_variance(returns, allow_short = FALSE)
```

Arguments

returns	Matrix or data.frame of asset returns (T x N).
allow_short	Logical. Allow short selling? Default FALSE.

Value

A list: weights, mean (raw-scale), vol (raw-scale).

Examples

```
set.seed(3)
R <- matrix(rnorm(300 * 4, 0.0003, 0.01), 300, 4)
colnames(R) <- c("A", "B", "C", "D")
mvo_min_variance(R)
```

---

mvo_summary *MVO Summary*

Description

Prints a formatted summary of the max-Sharpe and min-variance portfolios derived from an efficient frontier object.

Usage

```
mvo_summary(ef_obj, risk_free = 0.02)
```

Arguments

ef_obj	Output of compute_efficient_frontier().
risk_free	Numeric. Risk-free rate. Default 0.02.

Value

Invisibly returns a list with max_sharpe and min_variance rows.

optimize_quotes_gd *Optimize Quoting Parameters via Gradient Descent*

Description

Learns optimal market-making quoting parameters by minimising the mean squared error between a linear spread model and a target spread vector. The model is: $spread = a + b \cdot vol + c \cdot |inventory| + d \cdot imbalance$.

Usage

```
optimize_quotes_gd(data, target_spread, learning_rate = 0.01, epochs = 1000)
```

Arguments

data	A data.frame of historical book data containing columns volatility and imbalance (output of extract_features()).
target_spread	Numeric vector of ideal/observed spreads (length equal to nrow(data)).
learning_rate	Numeric. Gradient descent step size (alpha). Default 0.01.
epochs	Integer. Number of gradient descent iterations. Default 1000.

Value

A named numeric vector of four optimised parameters: a (intercept), b (volatility), c (inventory), d (imbalance).

Examples

```
set.seed(2)
book   <- extract_features(simulate_orderbook(300))
target <- book$spread + rnorm(300, 0, 0.001)
optimize_quotes_gd(book, target, learning_rate = 0.005, epochs = 500)
```

option_greeks *Option Greeks (Black-Scholes Analytical)*

Description

Option Greeks (Black-Scholes Analytical)

Usage

```
option_greeks(S, K, time_to_expiry, r, sigma, type = "call", q = 0)
```

Arguments

S	Stock price.
K	Strike.
time_to_expiry	Time to expiry.
r	Risk-free rate.
sigma	Volatility.
type	"call" or "put".
q	Dividend yield. Default 0.

Value

Named numeric vector: Delta, Gamma, Theta, Vega, Rho.

Examples

```
option_greeks(100, 100, 1, 0.05, 0.20, "call")
```

---

option_price_simulation

Option Price Simulation: CLT Demonstration (Large N)

Description

Simulates option prices over a large grid of (mean, sd) parameters and demonstrates the Central Limit Theorem convergence.

Usage

```
option_price_simulation(
  S = 100,
  K = 100,
  time_to_expiry = 1,
  r = 0.05,
  sigma_grid = c(0.1, 0.2, 0.3, 0.4),
  n_sim_vec = c(100, 500, 1000, 5000, 10000),
  type = "call",
  n_rep = 200,
  seed = 123
)
```

Arguments

S	Stock price.
K	Strike.
time_to_expiry	Time to expiry.
r	Risk-free rate.
sigma_grid	Numeric vector of sigmas to test.
n_sim_vec	Integer vector of simulation counts (e.g. CLT progression).
type	"call" or "put".
n_rep	Number of repetitions per configuration. Default 200.
seed	Integer seed. Default 123.

Value

A list: results data.frame, clt_plot, convergence_plot.

---

option_price_summary *Option Pricing Summary via Repeated Monte Carlo*

Description

Runs `price_option_mc()` multiple times and summarises the distribution of price estimates across repetitions.

Usage

```
option_price_summary(n_reps = 30, ...)
```

Arguments

n_reps	Integer. Number of independent MC repetitions. Default 30.
...	Additional parameters passed to `price_option_mc()`.

Value

A list with:

mean	Mean price across repetitions.
sd	Standard deviation of prices across repetitions.
prices	Numeric vector of all repetition prices.

Examples

```
option_price_summary(n_reps = 20, S0 = 100, K = 100,
                     r = 0.05, sigma = 0.2, n_sims = 5000)
```

performance_summary *Performance summary using PerformanceAnalytics*

Description

Performance summary using PerformanceAnalytics

Usage

```
performance_summary(returns)
```

Arguments

returns	Matrix/data.frame/xts of returns.

Value

PerformanceAnalytics table of annualized returns.

plot_asset_clusters *Plot Asset Clusters*

Description

Plot Asset Clusters

Usage

```
plot_asset_clusters(cl_obj, type = "heatmap", title = "Asset Clustering")
```

Arguments

cl_obj	Output from `asset_clustering()`.
type	Character. "heatmap", "dendrogram", or "network". Default "heatmap".
title	Character.

Value

A ggplot2 object or base R plot.

plot_binomial_tree *Plot Binomial Tree (Small Trees)*

Description

Visualises the stock price nodes for a binomial tree. Only recommended for n <= 15.

Usage

```
plot_binomial_tree(bt_obj, show = "stock", title = "Binomial Tree")
```

Arguments

bt_obj	Output from binomial_tree_option() or american_option_binomial().
show	Character. "stock" or "option". Default "stock".
title	Character.

Value

A ggplot2 object.

plot_correlation_heatmap *Plot Correlation Heatmap*

Description

Plot Correlation Heatmap

Usage

```
plot_correlation_heatmap(
  cor_obj,
  title = "Cross-Asset Correlation",
  show_values = TRUE,
  low_col = "#2166ac",
  high_col = "#d6604d"
)
```

Arguments

cor_obj	Output from asset_correlation_matrix(), or a correlation matrix directly.
title	Character. Plot title.
show_values	Logical. Show numeric values in cells. Default TRUE.
low_col	Colour for low correlation.
high_col	Colour for high correlation.

Value

A ggplot2 object.

`plot_cvar_frontier` *Plot CVaR Frontier*

Description

Plot CVaR Frontier

Usage

```
plot_cvar_frontier(cvar_obj, title = "CVaR-Efficient Frontier")
```

Arguments

`cvar_obj`	Output from `cvar_frontier()`.
`title`	Character. Plot title.

Value

A ggplot2 object.

`plot_efficient_frontier`
Plot the Efficient Frontier

Description

Visualises the efficient frontier with optional overlays for max-Sharpe and min-variance portfolios, and individual asset positions.

Usage

```
plot_efficient_frontier(
  ef_obj,
  highlight_portfolios = TRUE,
  risk_free = 0.02,
  show_assets = TRUE,
  title = "Efficient Frontier"
)
```

Arguments

ef_obj	Output from compute_efficient_frontier().
highlight_portfolios	Logical. Overlay max-Sharpe and min-var portfolios. Default TRUE.
risk_free	Numeric. Risk-free rate. Default 0.02.
show_assets	Logical. Show individual assets. Default TRUE.
title	Character. Plot title.

Value

A ggplot2 object.

---

plot_embedding *Plot 2D Embedding (t-SNE or UMAP)*

Description

Plot 2D Embedding (t-SNE or UMAP)

Usage

```
plot_embedding(
  embed_obj,
  labels = NULL,
  title = "2D Embedding",
  method = "t-SNE"
)
```

Arguments

embed_obj	Output from portfolio_tsne() or portfolio_umap().
labels	Optional factor / character vector for colouring. Default NULL.
title	Character.
method	Character. Label for subtitle. Default "t-SNE".

Value

A ggplot2 object.

plot_gd_convergence *Plot Gradient Descent Convergence*

Description

Plot Gradient Descent Convergence

Usage

```
plot_gd_convergence(gd_obj, title = "Gradient Descent Convergence")
```

Arguments

gd_obj	Output from gd_min_variance() or gd_max_sharpe().
title	Character.

Value

A ggplot2 object.

plot_mc_paths *Plot Monte Carlo Paths*

Description

Plots a random sample of simulated price or cumulative-return paths. Accepts output from either gbm_simulation() (App 2) or mc_price_simulation() (App 2), making it usable across both apps.

Usage

```
plot_mc_paths(mc_obj, n_show = 200, title = "Monte Carlo Price Paths")
```

Arguments

mc_obj	Output from gbm_simulation() or mc_price_simulation().
n_show	Integer. Maximum number of paths to display. Default 200.
title	Character. Plot title.

Value

A ggplot2 object.

plot_option_simulation *Plot Monte Carlo Option Paths*

Description

Plot Monte Carlo Option Paths

Usage

```
plot_option_simulation(mc_obj, n_paths = 100, title = "Monte Carlo GBM Paths")
```

Arguments

mc_obj	Output from monte_carlo_option().
n_paths	Integer. Number of paths to show. Default 100.
title	Character.

Value

A ggplot2 object.

plot_pca_biplot *Plot PCA Biplot*

Description

Plot PCA Biplot

Usage

```
plot_pca_biplot(
  pca_obj,
  pc_x = 1,
  pc_y = 2,
  colour_by = NULL,
  title = "PCA Biplot"
)
```

Arguments

pca_obj	Output from portfolio_pca().
pc_x	Integer. PC on x-axis. Default 1.
pc_y	Integer. PC on y-axis. Default 2.
colour_by	Optional numeric vector to colour observations by. Default NULL.
title	Character.

Value

A ggplot2 object.

`plot_regimes` *Plot Market Regimes*

Description

Plots a time series of returns coloured by detected regime.

Usage

```
plot_regimes(regime_obj, dates = NULL, title = "Market Regime Detection")
```

Arguments

`regime_obj`	Output from `kmeans_regime()` or `em_regime()`.
`dates`	Optional Date/POSIXct vector. Uses index if xts.
`title`	Character. Plot title.

Value

A ggplot2 object.

`plot_risk_contribution` *Plot Risk Contributions*

Description

Visualises portfolio weights alongside risk contributions using a bar chart (side-by-side) or a pie chart.

Usage

```
plot_risk_contribution(
  rp_obj,
  type = c("bar", "pie"),
  title = "Risk Contribution"
)
```

Arguments

`rp_obj`	Output from `equal_risk_contribution()` or `risk_parity_portfolio()`.
`type`	Character. `"bar"` (default) or `"pie"`.
`title`	Character. Plot title.

Value

A ggplot2 object.

`portfolio_asset_clustering` *Portfolio Asset Clustering (Extended)*

Description

Clusters portfolio assets using k-means, hierarchical, or EM methods on correlation-transformed distance in PCA-reduced space. Returns rich statistics including per-cluster summary, correlation matrix, and dendrogram. For the general-purpose version see `asset_clustering`.

Usage

```
portfolio_asset_clustering(
  returns,
  k = 3,
  method = "kmeans",
  n_pca = 5,
  seed = 42,
  freq = 252
)
```

Arguments

`returns`	Numeric matrix (T x N) of asset returns.
`k`	Integer. Number of clusters. Default 3.
`method`	Character. "kmeans", "hierarchical", or "em". Default "kmeans".
`n_pca`	Integer. PCA dims before clustering. Default 5.
`seed`	Integer. Default 42.
`freq`	Integer. Default 252.

Value

A list: labels, cluster_stats, dendrogram, centers, cor_matrix.

Examples

```
set.seed(42)
R <- matrix(rnorm(300 * 8, 0.0002, 0.01), 300, 8)
colnames(R) <- paste0("Asset", 1:8)
cl <- portfolio_asset_clustering(R, k = 3)
cl$cluster_stats
```

portfolio_clustering *Portfolio Clustering (full pipeline)*

Description

Runs the complete asset clustering pipeline including PCA reduction, k-means clustering, and regime assignment.

Usage

```
portfolio_clustering(returns, k = 3, freq = 252, plot_all = TRUE, seed = 42)
```

Arguments

returns	Numeric matrix of returns.
k	Integer. Clusters. Default 3.
freq	Integer. Default 252.
plot_all	Logical. Return all plots. Default TRUE.
seed	Integer. Default 42.

Value

A comprehensive list with clustering, plots, and statistics.

portfolio_cvar *Portfolio CVaR (Expected Shortfall)*

Description

Computes the Conditional Value-at-Risk (CVaR / Expected Shortfall) for a given weight vector using historical simulation.

Usage

```
portfolio_cvar(
  returns,
  weights,
  alpha = 0.95,
  method = c("historical", "parametric", "monte_carlo"),
  n_sim = 10000
)
```

Arguments

returns	Numeric matrix (T x N) of asset returns.
weights	Numeric vector of portfolio weights (length N).
alpha	Confidence level (e.g. 0.95). Default 0.95.
method	Character. "historical", "parametric", or "monte_carlo".
n_sim	Integer. Number of MC simulations (if method = "monte_carlo").

Value

A named numeric vector: VaR, CVaR (both as positive loss figures).

Examples

```
set.seed(42)
R <- matrix(rnorm(500 * 4, 0.0003, 0.012), 500, 4)
w <- c(0.25, 0.25, 0.25, 0.25)
portfolio_cvar(R, w, alpha = 0.95)
```

---

portfolio_pca *Portfolio PCA Analysis*

Description

Performs Principal Component Analysis on asset returns. Identifies dominant factors, factor loadings, and variance explained.

Usage

```
portfolio_pca(returns, scale_ = TRUE, n_comp = NULL, freq = 252)
```

Arguments

returns	Numeric matrix (T x N) of asset returns.
scale_	Logical. Standardise returns before PCA. Default TRUE.
n_comp	Integer. Number of components to retain. Default NULL (all).
freq	Integer. Periods per year. Default 252.

Value

A list: pca_obj, loadings, scores, var_explained, cumulative_var.

Examples

```
set.seed(42)
R <- matrix(rnorm(300 * 8, 0.0002, 0.01), 300, 8)
colnames(R) <- paste0("Asset", 1:8)
pca <- portfolio_pca(R)
pca$var_explained
```

`portfolio_performance` *Portfolio Performance Summary*

Description

Computes key performance metrics for a portfolio.

Usage

```
portfolio_performance(returns, weights = NULL, risk_free = 0.02, freq = 252)
```

Arguments

`returns`	Numeric vector of portfolio returns.
`weights`	Optional weight vector (for display).
`risk_free`	Numeric. Risk-free rate. Default 0.02.
`freq`	Integer. Periods per year. Default 252.

Value

A named list of performance metrics.

Examples

```
set.seed(1)
r <- rnorm(252, 0.0004, 0.012)
portfolio_performance(r, risk_free = 0.02)
```

`portfolio_tsne` *Portfolio t-SNE Embedding*

Description

Reduces high-dimensional return data to 2D using t-SNE. Useful for visualising clusters in asset return space.

Usage

```
portfolio_tsne(
  returns,
  perplexity = 30,
  n_iter = 1000,
  dims = 2,
  scale_ = TRUE,
  seed = 42,
  use_pca_first = TRUE,
  pca_dims = 20
)
```

Arguments

`returns`	Numeric matrix (T x N) of returns.
`perplexity`	Numeric. t-SNE perplexity. Default 30.
`n_iter`	Integer. t-SNE iterations. Default 1000.
`dims`	Integer. Output dimensions (2 or 3). Default 2.
`scale_`	Logical. Standardise inputs. Default TRUE.
`seed`	Integer. Default 42.
`use_pca_first`	Logical. Pre-reduce with PCA. Default TRUE.
`pca_dims`	Integer. PCA dims before t-SNE. Default 20.

Value

A list: embedding (T x dims), perplexity, note.

Examples

```
set.seed(42)
R <- matrix(rnorm(300 * 10, 0.0002, 0.01), 300, 10)
ts <- portfolio_tsne(R, perplexity = 30)
head(ts$embedding)
```

portfolio_umap *Portfolio UMAP Embedding*

Description

Reduces returns matrix to 2D using UMAP.

Usage

```
portfolio_umap(
  returns,
  n_neighbors = 15,
  min_dist = 0.1,
  dims = 2,
  scale_ = TRUE,
  seed = 42
)
```

Arguments

returns	Numeric matrix of returns.
n_neighbors	Integer. UMAP neighbours. Default 15.
min_dist	Numeric. UMAP min_dist. Default 0.1.
dims	Integer. Output dimensions. Default 2.
scale_	Logical. Default TRUE.
seed	Integer. Default 42.

Value

A list: embedding, config.

predict_regime_knn *Predict Regime using kNN*

Description

Classifies a new order book snapshot into one of the regimes learned by `cluster_book_kmeans()` using k-nearest-neighbour classification.

Usage

```
predict_regime_knn(train_data, new_snapshot, k = 5)
```

Arguments

train_data	A data.frame of historical snapshots that includes a regime column (i.e. output of cluster_book_kmeans()$data).
new_snapshot	A single-row data.frame containing the same feature columns as train_data.
k	Integer. Number of nearest neighbours. Default 5.

Value

A factor of length 1 with the predicted regime label.

Examples

```
set.seed(3)
book  <- extract_features(simulate_orderbook(500))
res   <- cluster_book_kmeans(book, centers = 3)
pred  <- predict_regime_knn(res$data, res$data[1, ], k = 5)
```

---

price_option_binomial *Binomial Tree Option Pricing (European / American)*

Description

Prices European or American options using the Cox-Ross-Rubinstein (CRR) binomial tree model.

Usage

```
price_option_binomial(
  S0,
  K,
  r,
  sigma,
  time_to_maturity = 1,
  n_steps = 100,
  type = c("call", "put"),
  american = FALSE
)
```

Arguments

S0	Spot price.
K	Strike price.
r	Risk-free rate (annualised).
sigma	Volatility (annualised).
time_to_maturity	Time to maturity in years. Default 1.

n_steps	Number of binomial tree steps. Default 100.
type	Character. "call" or "put".
american	Logical. TRUE for American option. Default FALSE.

Value

Scalar option price.

Examples

```
price_option_binomial(S0 = 100, K = 100, r = 0.05, sigma = 0.2)
price_option_binomial(S0 = 100, K = 100, r = 0.05, sigma = 0.2,
                      american = TRUE, type = "put")
```

---

price_option_mc *Monte Carlo European Option Pricing*

Description

Prices a European call or put option using Monte Carlo simulation under risk-neutral GBM dynamics.

Usage

```
price_option_mc(
  S0,
  K,
  r,
  sigma,
  time_to_maturity = 1,
  n_sims = 10000,
  type = c("call", "put"),
  seed = NULL
)
```

Arguments

S0	Spot price.
K	Strike price.
r	Risk-free rate (annualised).
sigma	Volatility (annualised).
time_to_maturity	Time to maturity in years. Default 1.
n_sims	Number of simulations. Default 10000.
type	Character. "call" or "put".
seed	Integer random seed. Default NULL.

Value

A list with elements:

price	Discounted Monte Carlo option price.
std_error	Standard error of the price estimate.

Examples

```
price_option_mc(S0 = 100, K = 100, r = 0.05, sigma = 0.2, seed = 1)
```

---

regime_statistics *Regime Statistics*

Description

Computes per-regime return, volatility, Sharpe, and duration statistics.

Usage

```
regime_statistics(regime_obj)
```

Arguments

regime_obj	Output from `kmeans_regime()` or `em_regime()`.

Value

A data.frame with per-regime statistics.

---

risk_contribution *Compute Risk Contributions*

Description

Computes the marginal and percentage risk contribution of each asset to total portfolio volatility.

Usage

```
risk_contribution(weights, cov_matrix)
```

Arguments

weights	Numeric weight vector (length N).
cov_matrix	N x N covariance matrix.

Value

A data.frame with columns: Asset, Weight, MargRC, RiskContrib, PercRC.

Examples

```
Sig <- matrix(c(0.04, 0.02, 0.02, 0.09), 2, 2)
risk_contribution(c(0.5, 0.5), Sig)
```

---

risk_parity_portfolio *Risk Parity Portfolio (Convenience Wrapper)*

Description

Alias for `equal_risk_contribution()` with named budget support. Use this when you want to specify a custom (possibly named) risk budget.

Usage

```
risk_parity_portfolio(returns, budget = NULL, freq = 252, ...)
```

Arguments

`returns`	Numeric matrix (T x N) of asset returns.
`budget`	Named numeric vector. Risk budget per asset (sums to 1). Default is equal budget.
`freq`	Integer. Periods per year. Default 252.
`...`	Additional arguments passed to `equal_risk_contribution()`.

Value

Same list as `equal_risk_contribution()`.

Examples

```
set.seed(1)
R <- matrix(rnorm(300 * 3, 0.0003, 0.01), 300, 3)
colnames(R) <- c("Equity", "Bond", "Gold")
rp <- risk_parity_portfolio(R, budget = c(0.5, 0.3, 0.2))
rp$weights
```

risk_parity_weights *Risk Parity Weights — Fast Iterative Solver*

Description

Computes equal-risk-contribution weights directly from a covariance matrix using a simple iterative proportional scaling algorithm. This is a lightweight alternative to `equal_risk_contribution()` when only a covariance matrix is available (no return series needed).

Usage

```
risk_parity_weights(covmat, max_iter = 1000, tol = 1e-08)
```

Arguments

`covmat`	Square numeric covariance matrix (N x N).
`max_iter`	Integer. Maximum number of iterations. Default 1000.
`tol`	Numeric. Convergence tolerance on normalised risk contribution deviation. Default 1e-8.

Value

A list with elements:

`weights`	Named numeric vector of risk-parity weights.
`iterations`	Number of iterations until convergence.

Examples

```
Sig <- matrix(c(0.04, 0.02, 0.02, 0.09), 2, 2)
colnames(Sig) <- rownames(Sig) <- c("A", "B")
risk_parity_weights(Sig)
```

rolling_correlation *Rolling Correlation*

Description

Computes rolling pairwise correlations between two assets over time.

Usage

```
rolling_correlation(r1, r2, window = 60, method = "pearson")
```

Arguments

r1	Numeric vector. Returns of asset 1.
r2	Numeric vector. Returns of asset 2.
window	Integer. Rolling window. Default 60.
method	Character. "pearson" etc. Default "pearson".

Value

A numeric vector of rolling correlations.

---

rolling_cv_forecast *Rolling cross-validation for return forecasts*

Description

Rolling cross-validation for return forecasts

Usage

```
rolling_cv_forecast(
  returns,
  window = 252,
  horizon = 21,
  step = 21,
  estimator = c("mean", "median")
)
```

Arguments

returns	Matrix/data.frame of returns.
window	Training window length.
horizon	Forecast horizon.
step	Step size between folds.
estimator	"mean" or "median".

Value

Data frame with fold metrics.

run_quantportr_app *Launch the QuantPortR Interactive Dashboard*

Description

Launch the QuantPortR Interactive Dashboard

Usage

```
run_quantportr_app(host = "127.0.0.1", port = 3838, launch.browser = TRUE)
```

Arguments

host	Character. Host address. Default "127.0.0.1".
port	Integer. Port to listen on. Default 3838.
launch.browser	Logical. Open browser on launch. Default TRUE.

Value

No return value. Called for the side effect of launching a Shiny application. Internally this starts a Shiny app object and blocks the current R session until the app is stopped.

sampling_distribution *Sampling Distribution Demonstration*

Description

Demonstrates properties of sampling distributions (mean, variance) through simulation, verifying theoretical results.

Usage

```
sampling_distribution(
  pop,
  stat = "mean",
  n_samples = c(10, 30, 50, 100, 200),
  n_reps = 2000,
  seed = 42
)
```

Arguments

pop	Numeric vector. Population of returns.
stat	Character. "mean" or "variance". Default "mean".
n_samples	Integer. Sample sizes to test.
n_reps	Integer. Repetitions. Default 2000.
seed	Integer. Default 42.

Value

A list: results, plots, theoretical_vs_empirical.

`scree_plot` *Scree Plot*

Description

Scree Plot

Usage

```
scree_plot(pca_obj, title = "Scree Plot (PCA Variance Explained)")
```

Arguments

`pca_obj`	Output from `portfolio_pca()`.
`title`	Character.

Value

A ggplot2 object.

`simulate_gbm_paths` *Simulate GBM Price Paths*

Description

Simulates stock price paths under Geometric Brownian Motion using the exact log-normal increments. Returns a matrix with rows as time steps and columns as simulations (compatible with App 1 option pricing workflow).

Usage

```
simulate_gbm_paths(
  S0,
  mu,
  sigma,
  time_horizon = 1,
  n_steps = 252,
  n_sims = 1000,
  seed = NULL
)
```

Arguments

S0	Initial price.
mu	Annual drift.
sigma	Annual volatility.
time_horizon	Time horizon in years. Default 1.
n_steps	Number of time steps. Default 252.
n_sims	Number of simulation paths. Default 1000.
seed	Integer random seed. Default NULL.

Value

Numeric matrix of simulated prices (n_steps+1 rows x n_sims cols).

Examples

```
paths <- simulate_gbm_paths(S0 = 100, mu = 0.08, sigma = 0.2, seed = 42)
dim(paths)   # 253 x 1000
```

---

simulate_orderbook *Simulate a Basic Order Book*

Description

Generates synthetic limit order book (LOB) snapshots over time by combining a random-walk mid-price with Poisson-distributed depth and a stochastic bid-ask spread.

Usage

```
simulate_orderbook(n_steps = 1000, p0 = 100)
```

Arguments

n_steps	Integer. Number of time steps to simulate. Default 1000.
p0	Numeric. Initial mid-price. Default 100.

Value

A data.frame with columns time, mid_price, bid, ask, bid_depth, and ask_depth.

Examples

```
set.seed(42)
book <- simulate_orderbook(n_steps = 500, p0 = 100)
head(book)
```

unbiasedness_check *Unbiasedness Check*

Description

Tests whether a list of estimates is unbiased w.r.t. a true value.

Usage

```
unbiasedness_check(estimates, true_val, tol = 0.05)
```

Arguments

estimates	Numeric vector of estimates.
true_val	Numeric. True parameter value.
tol	Relative tolerance. Default 0.05 (5%).

Value

A list: bias, rel_bias, is_unbiased.

var_cvar *VaR and CVaR analysis*

Description

VaR and CVaR analysis

Usage

```
var_cvar(
  returns,
  alpha = 0.95,
  method = c("historical", "gaussian"),
  portfolio_weights = NULL
)
```

Arguments

returns	Matrix/data.frame of returns.
alpha	Confidence level.
method	"historical" or "gaussian".
portfolio_weights	Optional weights to compute portfolio risk.

Value

Data frame with VaR and CVaR.

`var_cvar_analysis` *Full VaR / CVaR Portfolio Analysis*

Description

Computes VaR and CVaR using three methods and returns a comprehensive report.

Usage

```
var_cvar_analysis(
  returns,
  weights,
  alphas = c(0.9, 0.95, 0.99),
  n_sim = 50000,
  freq = 252
)
```

Arguments

`returns`	Numeric matrix (T x N) of asset returns.
`weights`	Numeric vector of portfolio weights.
`alphas`	Numeric vector of confidence levels. Default c(0.90, 0.95, 0.99).
`n_sim`	Integer. MC simulations. Default 50000.
`freq`	Integer. Periods per year. Default 252.

Value

A list with tables and ggplot objects.

Index