Package: JumpDiffSim (via r-universe)

June 3, 2026

Type Package

Title Jump Diffusion Simulation and Calibration for Merton and Kou Models

Version 0.1.0

Description Implements the Merton (1976) <doi:10.1016/0304-405X(76)90022-2> and Kou (2002) <doi:10.1287/mnsc.48.8.1086.166> jump-diffusion models through a unified S4 object-oriented interface. Provides exact compound-Poisson asset price simulation, maximum likelihood parameter estimation with Hessian-based standard errors, Wald-type confidence intervals, European option pricing via the Merton analytic series expansion, and publication-quality diagnostic plots. All functionality operates entirely offline without market data dependencies.

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.3

Collate 'generics.R' 'kou_fit.R' 'kou_model.R' 'kou_sim.R' 'merton_model.R' 'merton_fit.R' 'merton_price.R' 'merton_sim.R' 'utils.R'

Depends R (>= 4.1.0)

Imports methods, stats, ggplot2 (>= 4.0.2), numDeriv (>= 2016.8.1.1)

Suggests knitr, pkgdown, rmarkdown, covr, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

URL https://kennedy2244.github.io/JumpDiffSim/, https://github.com/kennedy2244/JumpDiffSim

BugReports https://github.com/kennedy2244/JumpDiffSim/issues

NeedsCompilation no

Author Kennedy Titus Kayaki [aut, cre], Dohyun Oh [aut], Ju Seong Hyeon [aut], Lee Se Eun [aut], Choi Jiwoo [aut], Yuri Shin [aut]

Maintainer Kennedy Titus Kayaki <kennedy_2244@yu.ac.kr>

Repository https://cran.r-universe.dev

Date/Publication 2026-06-03 17:23:06 UTC

RemoteUrl https://github.com/cran/JumpDiffSim

RemoteRef HEAD

RemoteSha 981778af06a2bb705e306da5b77993239c8edea5

Contents

`diagnosticPlots` *Diagnostic Plots for a JDSimResult Object*

Description

Creates three diagnostic plots for a JDSimResult object: a simulated path fan chart, a histogram of log-returns with a normal density overlay, and the autocorrelation function of squared log-returns.

Usage

```
diagnosticPlots(object)
```

Arguments

`object`	A JDSimResult object.

Value

A named list of three ggplot objects:

- fan_chart: simulated path quantile fan chart.
- density: histogram of log-returns with a normal overlay.
- acf_sq: autocorrelation of squared log-returns with 95\

Examples

```
m    <- MertonModel()
sim  <- simulateMerton(m, n = 50, T_ = 1, steps = 100)
plts <- diagnosticPlots(sim)
print(plts$fan_chart)
print(plts$density)
print(plts$acf_sq)
```

---

fit *Fit Model to Log-Returns via MLE*

Description

Fit Model to Log-Returns via MLE

Usage

```
fit(object, log_returns, ...)
```

Arguments

object	A JumpDiffModel object.
log_returns	Numeric vector of observed log returns.
...	Passed to optim.

Value

A JDFitResult object.

`fitMerton` *Fit Merton Jump-Diffusion Model via MLE*

Description

Maximises the log-likelihood of the Merton (1976) model using L-BFGS-B optimisation. Returns a JDFitResult object containing parameter estimates, standard errors, and convergence info.

Usage

```
fitMerton(
  log_returns,
  start = c(mu = 0.05, sigma = 0.2, lambda = 1, mu_j = -0.1, sigma_j = 0.15),
  dt = 1/252,
  N_max = 50L,
  verbose = FALSE
)
```

Arguments

`log_returns`	Numeric vector of observed log asset returns.
`start`	Named numeric vector of starting values. Defaults to `c(mu=0.05, sigma=0.20, lambda=1, mu_j=-0.10, sigma_j=0.15)`.
`dt`	Numeric. Time step length (default 1/252).
`N_max`	Integer. Mixture truncation (default 50).
`verbose`	Logical. Print progress? (default FALSE).

Value

A JDFitResult object.

Examples

```
ret <- jdSampleData("merton", n = 500, seed = 42)
fit <- fitMerton(ret, verbose = TRUE)
print(fit)
confint(fit)
```

`JDFitResult-class` *JDFitResult: MLE Estimation Output Class*

Description

JDFitResult: MLE Estimation Output Class

Usage

```
## S4 method for signature 'JDFitResult'
show(object)

## S4 method for signature 'JDFitResult'
confint(object, parm, level = 0.95, ...)
```

Arguments

`object`	A JDFitResult object.
`parm`	Ignored (all parameters returned).
`level`	Confidence level (default 0.95).
`...`	Unused.

Methods (by generic)

- `confint(JDFitResult)`: Wald-type 95\

Slots

`estimates` Named numeric. Parameter estimates.

`se` Named numeric. Standard errors.

`loglik` Numeric. Log-likelihood at optimum.

`converged` Logical. Did optimisation converge?

`model_type` Character. Model name.

`jdSampleData` *Generate Synthetic Log-Returns from a Jump-Diffusion Model*

Description

Convenience function for generating reproducible synthetic log-returns for use in examples, tests, and vignettes. All package examples and tests use this function to avoid any dependency on live market data.

Usage

```
jdSampleData(
  model = "merton",
  n = 500,
  mu = 0.05,
  sigma = 0.2,
  lambda = 1,
  mu_j = -0.1,
  sigma_j = 0.15,
  dt = 1/252,
  seed = 42L
)
```

Arguments

`model`	Character. Model type: `"merton"` (default).
`n`	Integer. Number of log-returns to generate (default 500).
`mu`	Numeric. Drift (default 0.05).
`sigma`	Positive numeric. Diffusion vol (default 0.20).
`lambda`	Non-negative numeric. Jump intensity (default 1).
`mu_j`	Numeric. Mean log-jump (default -0.10).
`sigma_j`	Positive numeric. Std dev of log-jumps (default 0.15).
`dt`	Numeric. Time step (default 1/252).
`seed`	Integer. Random seed (default 42).

Value

Numeric vector of `n` log-returns.

Examples

```
ret <- jdSampleData("merton", n = 200, seed = 42)
head(ret)
hist(ret, breaks = 40, main = "Synthetic Merton Returns")
```

JDSimResult-class *JDSimResult: Simulation Output Class*

Description

JDSimResult: Simulation Output Class

Slots

`paths` Matrix. Simulated price paths (n x steps+1).

`times` Numeric vector. Time grid.

`params` List. Model parameters used in simulation.

`model_type` Character. Model name.

JumpDiffModel-class *JumpDiffModel: Virtual Base Class for Jump-Diffusion Models*

Description

Abstract base class. Do not instantiate directly. Concrete subclasses: MertonModel.

Slots

`mu` Numeric. Drift parameter (annualised).

`sigma` Positive numeric. Diffusion volatility.

`lambda` Non-negative numeric. Jump intensity (jumps per year).

jumpMoments *Theoretical Moments of the Log-Return Distribution*

Description

Theoretical Moments of the Log-Return Distribution

Usage

```
jumpMoments(object)
```

Arguments

`object`	A JumpDiffModel object.

Value

Named numeric vector: mean, variance, skewness, kurtosis.

jumpMoments,MertonModel-method

Theoretical Moments of the Merton Jump-Diffusion Log-Return

Description

Returns the theoretical mean, variance, skewness, and excess kurtosis of the log-return distribution under the Merton (1976) model.

Usage

```
## S4 method for signature 'MertonModel'
jumpMoments(object)
```

Arguments

object	A MertonModel object.

Value

Named numeric vector with elements mean, variance, skewness, kurtosis.

Examples

```
m <- MertonModel(mu = 0.05, sigma = 0.20, lambda = 1,
                 mu_j = -0.10, sigma_j = 0.15)
jumpMoments(m)
```

loglik

Compute Negative Log-Likelihood

Description

Compute Negative Log-Likelihood

Usage

```
loglik(object, log_returns, ...)
```

Arguments

object	A JumpDiffModel object.
log_returns	Numeric vector of log asset returns.
...	Passed to methods.

Value

Scalar negative log-likelihood value.

`mertonLogLik` *Negative Log-Likelihood for the Merton Jump-Diffusion Model*

Description

Evaluates the negative log-likelihood of observing `log_returns` under the Merton (1976) model. The density is approximated by a Gaussian mixture truncated at `N_max = 50` terms.

Usage

```
mertonLogLik(params, log_returns, dt = 1/252, N_max = 50L)
```

Arguments

`params`	Named numeric vector: `c(mu, sigma, lambda, mu_j, sigma_j)`.
`log_returns`	Numeric vector of observed log asset returns.
`dt`	Numeric. Length of each time step (default 1/252).
`N_max`	Integer. Number of mixture terms (default 50).

Value

Scalar. Negative log-likelihood. Returns `Inf` for invalid parameter values.

Examples

```
ret <- jdSampleData("merton", n = 200, seed = 42)
p   <- c(mu = 0.05, sigma = 0.2, lambda = 1, mu_j = -0.1, sigma_j = 0.15)
mertonLogLik(p, ret)
```

`MertonModel` *Create a MertonModel Object*

Description

Create a MertonModel Object

Usage

```
MertonModel(mu = 0.05, sigma = 0.2, lambda = 1, mu_j = -0.1, sigma_j = 0.15)
```

Arguments

mu	Numeric. Drift (default 0.05).
sigma	Positive numeric. Diffusion volatility (default 0.20).
lambda	Non-negative numeric. Jump intensity (default 1).
mu_j	Numeric. Mean log-jump size (default -0.10).
sigma_j	Positive numeric. Std dev of log-jumps (default 0.15).

Value

A validated MertonModel object.

Examples

```
m <- MertonModel(mu = 0.05, sigma = 0.20, lambda = 1,
                 mu_j = -0.10, sigma_j = 0.15)
show(m)
```

MertonModel-class *MertonModel: Merton (1976) Jump-Diffusion Model*

Description

Extends JumpDiffModel with log-normal jump parameters.

Usage

```
## S4 method for signature 'MertonModel'
show(object)
```

Arguments

object	A MertonModel object.

Slots

mu_j Numeric. Mean log-jump size.

sigma_j Positive numeric. Std dev of log-jump sizes.

References

Merton, R.C. (1976). Option pricing when underlying stock returns are discontinuous. *Journal of Financial Economics*, 3(1-2), 125-144.

`priceEuropean` *Price a European Option under the Merton Jump-Diffusion Model*

Description

Computes the European call or put price under the Merton (1976) jump-diffusion model using the analytic series expansion. The price is expressed as a Poisson-weighted sum of Black-Scholes prices with jump-adjusted drift and volatility.

The formula is:

$$C = \sum_{n=0}^{N} \frac{e^{-\lambda' T}(\lambda' T)^n}{n!} \cdot BS(S, K, r_n, \sigma_n, T)$$

where $\lambda' = \lambda(1 + \bar{\mu}_J)$, $r_n = r - \lambda\bar{\mu}_J + n\log(1 + \bar{\mu}_J)/T$, and $\sigma_n^2 = \sigma^2 + n\sigma_J^2/T$.

Usage

```
priceEuropean(
  fit,
  S = 100,
  K = 100,
  r = 0.05,
  T_ = 1,
  type = "call",
  N_max = 50L
)
```

Arguments

`fit`	A JDFitResult object from `fitMerton`.
`S`	Positive numeric. Current asset price.
`K`	Positive numeric. Strike price.
`r`	Numeric. Continuously compounded risk-free rate.
`T_`	Positive numeric. Time to expiry in years.
`type`	Character. Either `"call"` or `"put"`.
`N_max`	Integer. Number of series terms (default 50).

Value

A named list with elements:

`price`	Merton jump-diffusion option price.
`bs_price`	Black-Scholes benchmark price (no jumps).
`jump_premium`	Difference between Merton and BS price.
`params`	List of inputs and fitted parameters used.

References

Merton, R.C. (1976). Option pricing when underlying stock returns are discontinuous. *Journal of Financial Economics*, 3(1-2), 125-144.

Examples

```
ret <- jdSampleData("merton", n = 500, seed = 42)
fit <- fitMerton(ret)
price <- priceEuropean(fit, S = 100, K = 100,
                       r = 0.05, T_ = 1, type = "call")
print(price)
```

`simulate` *Simulate Asset Price Paths*

Description

Simulate Asset Price Paths

Usage

```
simulate(object, n, T_, steps, seed = NULL, ...)
```

Arguments

`object`	A JumpDiffModel object.
`n`	Integer. Number of simulated paths.
`T_`	Positive numeric. Time horizon in years.
`steps`	Positive integer. Number of time steps.
`seed`	Optional integer seed for reproducibility.
`...`	Additional arguments passed to methods.

Value

A JDSimResult object.

`simulateMerton` *Simulate Merton Jump-Diffusion Asset Paths*

Description

Generates `n` exact compound-Poisson asset price paths under the Merton (1976) jump-diffusion model. Simulation is exact in the sense that jump arrivals are drawn directly from the Poisson process rather than approximated via Euler discretisation.

Usage

```
simulateMerton(object, n = 100, T_ = 1, steps = 252, S0 = 1, seed = NULL, ...)
```

Arguments

`object`	A MertonModel object.
`n`	Integer. Number of paths to simulate.
`T_`	Positive numeric. Time horizon in years (default 1).
`steps`	Positive integer. Number of time steps (default 252).
`S0`	Numeric. Initial asset price (default 1).
`seed`	Optional integer. Random seed for reproducibility.
`...`	Unused.

Value

A JDSimResult object.

Examples

```
m   <- MertonModel()
sim <- simulateMerton(m, n = 10, T_ = 1, steps = 252, seed = 42)
dim(sim@paths)   # should be 200 x 253
```

Index