# Package 'CustomDerivative'

July 28, 2026

**Type** Package

**Title** Extensible Derivative Pricing and Risk Analytics

**Version** 0.2.0

**Description** Tools for pricing and analysing financial derivatives under the classical lognormal diffusion model and geometric Brownian motion assumptions. The package provides analytical European option prices, Monte Carlo pricing with antithetic and control variates, confidence intervals, finite-difference Greeks, and path simulation for path-dependent payoffs. The simulation interfaces accept user-defined payoff functions, enabling transparent construction of custom contracts while reporting numerical uncertainty.

**License** MIT + file LICENSE

**URL** https://github.com/AIM-IT4/CustomDerivative

**BugReports** https://github.com/AIM-IT4/CustomDerivative/issues

**Encoding** UTF-8

**Depends** R (>= 4.1.0)

**Imports** R6, stats

**Suggests** testthat (>= 3.0.0)

**Config/testthat/edition** 3

**Config/roxygen2/version** 8.0.0

**NeedsCompilation** no

**Author** Amit Kumar Jha [aut, cre, cph]

**Maintainer** Amit Kumar Jha <jha.8@iitj.ac.in>

**Repository** CRAN

**Date/Publication** 2026-07-28 07:00:02 UTC

# Contents

asian_call_payoff . . . . . 2
black_scholes_price . . . . . 2
CustomDerivative . . . . . 3
down_and_out_call_payoff . . . . . 4
finite_difference_greeks . . . . . 5
payoff_helpers . . . . . 6
price_european_mc . . . . . 6
price_path_dependent_mc . . . . . 7
simulate_gbm_paths . . . . . 8

**Index** **10**

---

`asian_call_payoff` *Arithmetic-average Asian call payoff*

---

## Description

Arithmetic-average Asian call payoff

## Usage

```
asian_call_payoff(strike, include_spot = FALSE)
```

## Arguments

| | |
|---|---|
| `strike` | Strike price. |
| `include_spot` | Whether the initial spot column participates in the average. |

## Value

A payoff function accepting a path matrix.

---

`black_scholes_price` *Black-Scholes price for a European option*

---

## Description

Computes the analytical Black-Scholes-Merton price of a European call or put with a continuous dividend yield.

## Usage

```
black_scholes_price(
  spot,
  strike,
  maturity,
  rate,
  volatility,
  dividend_yield = 0,
  type = c("call", "put")
)
```

## Arguments

| | |
|---|---|
| `spot` | Current underlying price. |
| `strike` | Strike price. |
| `maturity` | Time to maturity in years. |
| `rate` | Continuously compounded risk-free rate. |
| `volatility` | Annualized volatility. |
| `dividend_yield` | Continuous dividend yield. |
| `type` | Either `"call"` or `"put"`. |

## Value

A numeric option price.

---

# `CustomDerivative` *Legacy CustomDerivative R6 interface*

---

## Description

Backward-compatible wrapper for users of versions 0.1.x. New code should generally use `price_european_mc()` and the payoff helper functions directly.

## Methods

### Public methods:

- `CustomDerivative$new()`
- `CustomDerivative$price()`
- `CustomDerivative$clone()`

**`CustomDerivative$new()`:** Create a legacy custom derivative object.

*Usage:*

```
CustomDerivative$new(
  underlying_price,
  strike_price,
  time_to_maturity,
  volatility,
  risk_free_rate,
  payoff_function
)
```

*Arguments:*

`underlying_price` Initial underlying price.

`strike_price` Strike price retained for compatibility.

`time_to_maturity` Time to maturity in years.

`volatility` Annualized volatility.

`risk_free_rate` Continuously compounded risk-free rate.

`payoff_function` Vectorized terminal payoff function.

`CustomDerivative$price()`**:** Price the derivative using risk-neutral Monte Carlo.

*Usage:*

```
CustomDerivative$price(n_simulations = 10000L, seed = NULL)
```

*Arguments:*

`n_simulations` Number of simulations.

`seed` Optional random seed.

*Returns:* Numeric derivative price.

`CustomDerivative$clone()`**:** The objects of this class are cloneable with this method.

*Usage:*

```
CustomDerivative$clone(deep = FALSE)
```

*Arguments:*

`deep` Whether to make a deep clone.

---

`down_and_out_call_payoff` *Down-and-out European call payoff*

---

## Description

Down-and-out European call payoff

## Usage

```
down_and_out_call_payoff(strike, barrier)
```

### Arguments

| | |
|---|---|
| strike | Strike price. |
| barrier | Lower barrier level. |

### Value

A payoff function accepting a path matrix.

---

`finite_difference_greeks` *Finite-difference Greeks for a pricing function*

---

### Description

The pricing function must accept named arguments `spot`, `maturity`, `rate`, and `volatility`, and return either a numeric price or a `cd_pricing_result`.

### Usage

```
finite_difference_greeks(
  pricer,
  spot,
  maturity,
  rate,
  volatility,
  ...,
  spot_bump = 1e-04,
  volatility_bump = 1e-04,
  rate_bump = 1e-04,
  time_bump = 1/365
)
```

### Arguments

| | |
|---|---|
| pricer | Pricing function. |
| spot | Current underlying price. |
| maturity | Time to maturity in years. |
| rate | Risk-free rate. |
| volatility | Annualized volatility. |
| ... | Additional arguments passed to `pricer`. |
| spot_bump | Relative spot bump. |
| volatility_bump | Absolute volatility bump. |
| rate_bump | Absolute rate bump. |
| time_bump | Time bump in years. |

## Value

Named numeric vector containing delta, gamma, vega, rho, and theta.

---

`payoff_helpers` *Standard terminal payoff functions*

---

## Description

Standard terminal payoff functions

## Usage

```
call_payoff(strike)

put_payoff(strike)

digital_call_payoff(strike, cash = 1)
```

## Arguments

| | |
|---|---|
| `strike` | Strike price. |
| `cash` | Cash amount paid by the digital call when in the money. |

## Value

A vectorized payoff function accepting terminal prices.

---

`price_european_mc` *Monte Carlo price for a European custom payoff*

---

## Description

Prices a terminal-value payoff under risk-neutral geometric Brownian motion. The estimator can use antithetic variates and a discounted terminal-underlying control variate whose expectation is known analytically.

## Usage

```
price_european_mc(
  payoff,
  spot,
  maturity,
  rate,
  volatility,
  dividend_yield = 0,
  n_simulations = 100000L,
  seed = NULL,
  antithetic = TRUE,
  control_variate = TRUE,
  confidence_level = 0.95
)
```

## Arguments

| | |
|---|---|
| payoff | Vectorized function of terminal prices. |
| spot | Current underlying price. |
| maturity | Time to maturity in years. |
| rate | Continuously compounded risk-free rate. |
| volatility | Annualized volatility. |
| dividend_yield | Continuous dividend yield. |
| n_simulations | Number of Monte Carlo scenarios. |
| seed | Optional integer random seed. The caller's RNG state is restored. |
| antithetic | Whether to use antithetic normal variates. |
| control_variate | Whether to use the discounted terminal underlying as a control variate. |
| confidence_level | Confidence level for the normal-approximation interval. |

## Value

An object of class `cd_pricing_result`.

---

# price_path_dependent_mc

*Monte Carlo price for a path-dependent custom payoff*

---

## Description

Monte Carlo price for a path-dependent custom payoff

## Usage

```
price_path_dependent_mc(
  payoff,
  spot,
  maturity,
  rate,
  volatility,
  dividend_yield = 0,
  n_steps = 252L,
  n_simulations = 10000L,
  seed = NULL,
  antithetic = TRUE,
  confidence_level = 0.95
)
```

## Arguments

| | |
|---|---|
| payoff | Function accepting a path matrix and returning one payoff per row. |
| spot | Current underlying price. |
| maturity | Time horizon in years. |
| rate | Continuously compounded risk-free rate. |
| volatility | Annualized volatility. |
| dividend_yield | Continuous dividend yield. |
| n_steps | Number of monitoring intervals. |
| n_simulations | Number of paths. |
| seed | Optional integer random seed. |
| antithetic | Whether to use antithetic normal innovations. |
| confidence_level | Confidence level for the normal-approximation interval. |

## Value

An object of class `cd_pricing_result`.

---

# simulate_gbm_paths *Simulate geometric Brownian motion paths*

---

## Description

Simulate geometric Brownian motion paths

## Usage

```
simulate_gbm_paths(
  spot,
  maturity,
  rate,
  volatility,
  dividend_yield = 0,
  n_steps = 252L,
  n_simulations = 10000L,
  seed = NULL,
  antithetic = TRUE
)
```

## Arguments

| | |
|---|---|
| `spot` | Current underlying price. |
| `maturity` | Time horizon in years. |
| `rate` | Continuously compounded risk-free rate. |
| `volatility` | Annualized volatility. |
| `dividend_yield` | Continuous dividend yield. |
| `n_steps` | Number of monitoring intervals. |
| `n_simulations` | Number of paths. |
| `seed` | Optional integer random seed. |
| `antithetic` | Whether to use antithetic normal innovations. |

## Value

A matrix with one path per row, including the initial price in column 1.

# Index

asian_call_payoff, 2

black_scholes_price, 2

call_payoff *(*payoff_helpers*)*, 6
CustomDerivative, 3

digital_call_payoff *(*payoff_helpers*)*, 6
down_and_out_call_payoff, 4

finite_difference_greeks, 5

payoff_helpers, 6
price_european_mc, 6
price_european_mc(), *3*
price_path_dependent_mc, 7
put_payoff *(*payoff_helpers*)*, 6

simulate_gbm_paths, 8