# Package 'AsianOption'

December 22, 2025

**Type** Package

**Title** Asian Option Pricing with Price Impact

**Version** 0.1.0

**Date** 2025-12-04

**Maintainer** Priyanshu Tiwari <tiwari.priyanshu.iitk@gmail.com>

**Description** Implements binomial tree pricing for geometric and arithmetic Asian options incorporating market price impact from hedging activities. Uses the Cox-Ross-Rubinstein (CRR) model with the replicating portfolio method. Provides exact pricing for geometric Asian options and bounds for arithmetic Asian options based on Jensen's inequality. The price impact mechanism models how hedging volumes affect stock prices, leading to modified risk-neutral probabilities. Based on the methodology described in Tiwari and Majumdar (2025) <doi:10.48550/arXiv.2512.07154>.

**License** GPL (>= 3)

**URL** https://github.com/plato-12/AsianOption

**BugReports** https://github.com/plato-12/AsianOption/issues

**Encoding** UTF-8

**Depends** R (>= 4.0.0)

**Imports** Rcpp (>= 1.0.0)

**LinkingTo** Rcpp

**Suggests** testthat (>= 3.0.0), covr

**RoxygenNote** 7.3.3

**NeedsCompilation** yes

**Author** Priyanshu Tiwari [aut, cre] (ORCID: <https://orcid.org/0009-0007-8917-4689>),
Sourav Majumdar [ctb]

**Repository** CRAN

**Date/Publication** 2025-12-22 18:20:02 UTC

# Contents



---

`arithmetic_asian_bounds` *Bounds for Arithmetic Asian Option with Price Impact*

---

## Description

Computes lower and upper bounds for the arithmetic Asian option (call or put) using the relationship between arithmetic and geometric means (Jensen's inequality).

## Usage

```
arithmetic_asian_bounds(
  S0,
  K,
  r,
  u,
  d,
  lambda,
  v_u,
  v_d,
  n,
  option_type = "call",
  compute_path_specific = FALSE,
  validate = TRUE
)
```

## Arguments

| | |
|---|---|
| `S0` | Initial stock price (must be positive) |
| `K` | Strike price (must be positive) |
| `r` | Gross risk-free rate per period (e.g., 1.05) |
| `u` | Base up factor in CRR model (must be > d) |
| `d` | Base down factor in CRR model (must be positive) |
| `lambda` | Price impact coefficient (non-negative) |
| `v_u` | Hedging volume on up move (non-negative) |
| `v_d` | Hedging volume on down move (non-negative) |
| `n` | Number of time steps (positive integer, recommended n <= 20) |
| `option_type` | Character; either "call" (default) or "put" |
| `compute_path_specific` | Logical. If TRUE, computes the tighter path-specific upper bound using exact enumeration of all 2^n paths. Default is FALSE. |
| `validate` | Logical; if TRUE, performs input validation (default TRUE) |

## Details

Computes rigorous upper and lower bounds for arithmetic Asian options using Jensen's inequality. The lower bound is the geometric Asian option price (from AM-GM inequality). Two types of upper bounds are available:

**Global upper bound:** Uses a worst-case spread parameter applicable to all paths.

**Path-specific upper bound:** Computes tighter bounds by using path-specific spread parameters. This requires exact enumeration of all 2^n paths in the binomial tree (no sampling or approximation). The path-specific bound is typically much tighter than the global bound.

For detailed mathematical formulations, see the package vignettes and the reference paper.

## Value

List containing:

**lower_bound** Lower bound for arithmetic option (= geometric option price)

**upper_bound** Upper bound for arithmetic option (global bound, for backward compatibility)

**upper_bound_global** Global upper bound using $\rho^*$

**upper_bound_path_specific** Path-specific upper bound (only if compute_path_specific=TRUE, otherwise NA)

**rho_star** Spread parameter $\rho^*$

**EQ_G** Expected geometric average under risk-neutral measure

**V0_G** Geometric Asian option price (same as lower_bound)

**n_paths_used** Number of paths used for path-specific bound (2^n if computed, 0 otherwise)

## References

Tiwari, P., & Majumdar, S. (2025). Asian option valuation under price impact. *arXiv preprint*. doi:10.48550/arXiv.2512.07154

## See Also

price_geometric_asian

## Examples

```
# Compute basic bounds (global bound only) for call option
bounds <- arithmetic_asian_bounds(
  S0 = 100, K = 100, r = 1.05, u = 1.2, d = 0.8,
  lambda = 0.1, v_u = 1, v_d = 1, n = 3, option_type = "call"
)

print(bounds)

# Compute bounds for put option
bounds_put <- arithmetic_asian_bounds(
  S0 = 100, K = 100, r = 1.05, u = 1.2, d = 0.8,
  lambda = 0.1, v_u = 1, v_d = 1, n = 3, option_type = "put"
)

# Compute with path-specific bound (uses exact enumeration of all 2^n paths)
bounds_ps <- arithmetic_asian_bounds(
  S0 = 100, K = 100, r = 1.05, u = 1.2, d = 0.8,
  lambda = 0.1, v_u = 1, v_d = 1, n = 5,
  compute_path_specific = TRUE
)

print(bounds_ps)

# Estimate arithmetic option price as midpoint of path-specific bounds
if (!is.na(bounds_ps$upper_bound_path_specific)) {
  estimated_price <- mean(c(bounds_ps$lower_bound,
                            bounds_ps$upper_bound_path_specific))
  cat("Estimated price:", estimated_price, "\n")
}
```

---

check_no_arbitrage *Check No-Arbitrage Condition*

---

## Description

Verifies that the no-arbitrage condition $\tilde{d} < r < \tilde{u}$ holds.

### Usage

```
check_no_arbitrage(r, u, d, lambda, v_u, v_d)
```

### Arguments

| | |
|---|---|
| r | Gross risk-free rate per period |
| u | Base up factor |
| d | Base down factor |
| lambda | Price impact coefficient |
| v_u | Hedging volume on up move |
| v_d | Hedging volume on down move |

### Value

Logical: TRUE if condition holds, FALSE otherwise

### Examples

```
check_no_arbitrage(r = 1.05, u = 1.2, d = 0.8, lambda = 0.1, v_u = 1, v_d = 1)
```

---

## compute_adjusted_factors *Compute Adjusted Up and Down Factors*

---

### Description

Calculates the modified up and down factors after incorporating price impact from hedging.

### Usage

```
compute_adjusted_factors(u, d, lambda, v_u, v_d)
```

### Arguments

| | |
|---|---|
| u | Base up factor |
| d | Base down factor |
| lambda | Price impact coefficient |
| v_u | Hedging volume on up move |
| v_d | Hedging volume on down move |

### Value

List with elements `u_tilde` and `d_tilde`

### Examples

```
compute_adjusted_factors(u = 1.2, d = 0.8, lambda = 0.1, v_u = 1, v_d = 1)
```

## compute_p_adj *Compute Adjusted Risk-Neutral Probability*

### Description

Calculates the adjusted risk-neutral probability incorporating price impact from hedging activities.

### Usage

```
compute_p_adj(r, u, d, lambda, v_u, v_d)
```

### Arguments

| | |
|---|---|
| r | Gross risk-free rate per period |
| u | Base up factor |
| d | Base down factor |
| lambda | Price impact coefficient |
| v_u | Hedging volume on up move |
| v_d | Hedging volume on down move |

### Value

Adjusted risk-neutral probability (numeric)

### Examples

```
compute_p_adj(r = 1.05, u = 1.2, d = 0.8, lambda = 0.1, v_u = 1, v_d = 1)
```

## price_black_scholes_call *Black-Scholes European Call Option Price*

### Description

Computes the exact price of a European call option using the classical Black-Scholes (1973) analytical formula. This is the continuous-time benchmark for comparison with discrete binomial models.

### Usage

```
price_black_scholes_call(S0, K, r, sigma, time_to_maturity)
```

## Arguments

- `S0` Initial stock price (must be positive)
- `K` Strike price (must be positive)
- `r` Continuously compounded risk-free rate (e.g., 0.05 for 5% annual rate)
- `sigma` Volatility (annualized standard deviation, must be non-negative)
- `time_to_maturity` Time to maturity in years (must be positive)

## Details

The Black-Scholes formula for a European call option is:

$$C = S_0 N(d_1) - K e^{-rT} N(d_2)$$

where:

$$d_1 = \frac{\log(S_0/K) + (r + \sigma^2/2)T}{\sigma\sqrt{T}}$$

$$d_2 = d_1 - \sigma\sqrt{T}$$

and $N(\cdot)$ is the cumulative standard normal distribution function.

This formula assumes:

- Stock price follows geometric Brownian motion: $dS_t = rS_t dt + \sigma S_t dW_t$
- No dividends
- Constant risk-free rate and volatility
- Continuous trading with no transaction costs or price impact

## Value

European call option price (numeric)

## References

Black, F., & Scholes, M. (1973). The Pricing of Options and Corporate Liabilities. *Journal of Political Economy*, 81(3), 637-654. doi:10.1086/260062

## Examples

```
price_black_scholes_call(S0 = 100, K = 100, r = 0.05, sigma = 0.2,
                         time_to_maturity = 1)
```

price_black_scholes_put

*Black-Scholes European Put Option Price*

## Description

Computes the exact price of a European put option using the classical Black-Scholes (1973) analytical formula.

## Usage

```
price_black_scholes_put(S0, K, r, sigma, time_to_maturity)
```

## Arguments

| | |
|---|---|
| `S0` | Initial stock price (must be positive) |
| `K` | Strike price (must be positive) |
| `r` | Continuously compounded risk-free rate (e.g., 0.05 for 5% annual rate) |
| `sigma` | Volatility (annualized standard deviation, must be non-negative) |
| `time_to_maturity` | Time to maturity in years (must be positive) |

## Details

The Black-Scholes formula for a European put option is:

$$P = Ke^{-rT}N(-d_2) - S_0N(-d_1)$$

where:

$$d_1 = \frac{\log(S_0/K) + (r + \sigma^2/2)T}{\sigma\sqrt{T}}$$

$$d_2 = d_1 - \sigma\sqrt{T}$$

and $N(\cdot)$ is the cumulative standard normal distribution function.

Alternatively, the put price can be derived from put-call parity:

$$P = C - S_0 + Ke^{-rT}$$

## Value

European put option price (numeric)

## Put-Call Parity

The Black-Scholes put and call prices satisfy:

$$C - P = S_0 - Ke^{-rT}$$

This relationship holds exactly for European options without dividends.

### References

Black, F., & Scholes, M. (1973). The Pricing of Options and Corporate Liabilities. *Journal of Political Economy*, 81(3), 637-654. doi:10.1086/260062

### See Also

`price_black_scholes_call`

### Examples

```
price_black_scholes_put(S0 = 100, K = 100, r = 0.05, sigma = 0.2,
                        time_to_maturity = 1)
```

---

`price_european` *Price European Option with Price Impact*

---

### Description

Computes the exact price of a European option (call or put) using the Cox-Ross-Rubinstein (CRR) binomial model with price impact from hedging activities.

### Usage

```
price_european(
  S0,
  K,
  r,
  u,
  d,
  lambda,
  v_u,
  v_d,
  n,
  option_type = "call",
  validate = TRUE
)
```

### Arguments

| | |
|---|---|
| S0 | Initial stock price (must be positive) |
| K | Strike price (must be positive) |
| r | Gross risk-free rate per period (e.g., 1.05 for 5% rate) |
| u | Base up factor in CRR model (must be > d) |
| d | Base down factor in CRR model (must be positive) |
| lambda | Price impact coefficient (non-negative) |

| | |
|---|---|
| v_u | Hedging volume on up move (non-negative) |
| v_d | Hedging volume on down move (non-negative) |
| n | Number of time steps (positive integer) |
| option_type | Character; either "call" (default) or "put" |
| validate | Logical; if TRUE, performs input validation |

## Details

Computes exact prices for European options (call or put) using the binomial model with price impact. Price impact from hedging activities modifies the stock dynamics through adjusted up/down factors and risk-neutral probability.

Unlike path-dependent Asian options, European options only depend on the terminal stock price, allowing for efficient O(n) computation instead of O(2^n). See the package vignettes and reference paper for detailed mathematical formulations.

## Value

European option price (numeric)

## References

Tiwari, P., & Majumdar, S. (2025). Asian option valuation under price impact. *arXiv preprint*. doi:10.48550/arXiv.2512.07154

## See Also

price_geometric_asian, compute_p_adj

## Examples

```
# Call option with no price impact
price_european(
  S0 = 100, K = 100, r = 1.05, u = 1.2, d = 0.8,
  lambda = 0, v_u = 0, v_d = 0, n = 10, option_type = "call"
)

# Put option with price impact
price_european(
  S0 = 100, K = 100, r = 1.05, u = 1.2, d = 0.8,
  lambda = 0.1, v_u = 1, v_d = 1, n = 10, option_type = "put"
)

# Verify put-call parity
call <- price_european(100, 100, 1.05, 1.2, 0.8, 0.1, 1, 1, 10, "call")
put <- price_european(100, 100, 1.05, 1.2, 0.8, 0.1, 1, 1, 10, "put")
```

## `price_geometric_asian` *Price Geometric Asian Option with Price Impact*

### Description

Computes the exact price of a geometric Asian option (call or put) using the Cox-Ross-Rubinstein (CRR) binomial model with price impact from hedging activities. Uses exact enumeration of all 2^n paths.

### Usage

```
price_geometric_asian(
  S0,
  K,
  r,
  u,
  d,
  lambda,
  v_u,
  v_d,
  n,
  option_type = "call",
  validate = TRUE
)
```

### Arguments

| | |
|---|---|
| `S0` | Initial stock price (must be positive) |
| `K` | Strike price (must be positive) |
| `r` | Gross risk-free rate per period (e.g., 1.05) |
| `u` | Base up factor in CRR model (must be > d) |
| `d` | Base down factor in CRR model (must be positive) |
| `lambda` | Price impact coefficient (non-negative) |
| `v_u` | Hedging volume on up move (non-negative) |
| `v_d` | Hedging volume on down move (non-negative) |
| `n` | Number of time steps (positive integer) |
| `option_type` | Character; either "call" (default) or "put" |
| `validate` | Logical; if TRUE, performs input validation |

## Details

Computes exact prices for geometric Asian options using complete path enumeration in a binomial tree. Price impact from hedging activities modifies the stock dynamics through adjusted up/down factors and risk-neutral probability.

This function enumerates all 2^n possible paths in the binomial tree for exact pricing (no approximation or sampling). For large n (> 20), this requires significant computation time and memory. See the package vignettes and reference paper for detailed mathematical formulations.

## Value

Geometric Asian option price (numeric).

## References

Tiwari, P., & Majumdar, S. (2025). Asian option valuation under price impact. *arXiv preprint*. doi:10.48550/arXiv.2512.07154

## See Also

`arithmetic_asian_bounds`, `compute_p_adj`

## Examples

```
# Basic example
price_geometric_asian(
  S0 = 100, K = 100, r = 1.05, u = 1.2, d = 0.8,
  lambda = 0, v_u = 0, v_d = 0, n = 10
)

# With price impact
price_geometric_asian(
  S0 = 100, K = 100, r = 1.05, u = 1.2, d = 0.8,
  lambda = 0.1, v_u = 1, v_d = 1, n = 15
)

# Put option
price_geometric_asian(
  S0 = 100, K = 100, r = 1.05, u = 1.2, d = 0.8,
  lambda = 0.1, v_u = 1, v_d = 1, n = 10,
  option_type = "put"
)
```

price_kemna_vorst_arithmetic
*Kemna-Vorst Arithmetic Average Asian Option*

## Description

Calculates the price of an arithmetic average Asian option using Monte Carlo simulation with variance reduction via the geometric average control variate. This implements the Kemna & Vorst (1990) method WITHOUT price impact.

## Usage

```
price_kemna_vorst_arithmetic(
  S0,
  K,
  r,
  sigma,
  T0,
  T_mat,
  n,
  M = 10000,
  option_type = "call",
  use_control_variate = TRUE,
  seed = NULL,
  return_diagnostics = FALSE
)
```

## Arguments

| | |
|---|---|
| `S0` | Numeric. Initial stock price at time T0 (start of averaging period). Must be positive. |
| `K` | Numeric. Strike price. Must be positive. |
| `r` | Numeric. Continuously compounded risk-free rate (e.g., 0.05 for 5%). Use `log(r_gross)` to convert from gross rate. |
| `sigma` | Numeric. Volatility (annualized standard deviation). Must be non-negative. |
| `T0` | Numeric. Start time of averaging period. Must be non-negative. |
| `T_mat` | Numeric. Maturity time. Must be greater than T0. |
| `n` | Integer. Number of averaging points (observations). Must be positive. |
| `M` | Integer. Number of Monte Carlo simulations. Default is 10000. Larger values give more accurate results but take longer. |
| `option_type` | Character. Type of option: "call" (default) or "put". |
| `use_control_variate` | Logical. If TRUE (default), uses the geometric average as a control variate for variance reduction. This dramatically improves accuracy. |

`seed` Integer. Random seed for reproducibility. Default is NULL (no seed).

`return_diagnostics`
Logical. If TRUE, returns additional diagnostic information including confidence intervals, correlation, and variance reduction factor. Default is FALSE.

### Value

If `return_diagnostics = FALSE`, returns a numeric value (the estimated option price). If `return_diagnostics = TRUE`, returns a list with components:

**price** Estimated option price

**std_error** Standard error of the estimate

**lower_ci** Lower 95% confidence interval

**upper_ci** Upper 95% confidence interval

**geometric_price** Analytical geometric average price (control variate)

**correlation** Correlation between arithmetic and geometric payoffs

**variance_reduction_factor** Ratio of variances (with/without control)

**n_simulations** Number of Monte Carlo simulations used

**n_steps** Number of time steps in each simulation

### References

Kemna, A.G.Z. and Vorst, A.C.F. (1990). "A Pricing Method for Options Based on Average Asset Values." *Journal of Banking and Finance*, 14, 113-129.

### Examples

```
price_kemna_vorst_arithmetic(
  S0 = 100, K = 100, r = 0.05, sigma = 0.2,
  T0 = 0, T_mat = 1, n = 50, M = 10000
)
```

---

## price_kemna_vorst_geometric

*Kemna-Vorst Geometric Average Asian Option*

---

### Description

Calculates the price of a geometric average Asian call option using the closed-form analytical solution from Kemna & Vorst (1990). This is the standard benchmark implementation WITHOUT price impact.

### Usage

```
price_kemna_vorst_geometric(S0, K, r, sigma, T0, T_mat, option_type = "call")
```

### Arguments

| | |
|---|---|
| `S0` | Numeric. Initial stock price at time T0 (start of averaging period). Must be positive. |
| `K` | Numeric. Strike price. Must be positive. |
| `r` | Numeric. Gross risk-free interest rate per period (e.g., 1.05 for 5 Must be positive. |
| `sigma` | Numeric. Volatility (annualized standard deviation). Must be non-negative. |
| `T0` | Numeric. Start time of averaging period. Must be non-negative. |
| `T_mat` | Numeric. Maturity time. Must be greater than T0. |
| `option_type` | Character. Type of option: "call" (default) or "put". |

### Details

The geometric average at maturity is defined as:

$$G_T = \exp\left(\frac{1}{T - T_0}\int_{T_0}^{T} \log(S(\tau))d\tau\right)$$

For the discrete case with n+1 observations:

$$G_T = \left(\prod_{i=0}^{n} S(T_i)\right)^{1/(n+1)}$$

The closed-form solution for a call option is:

$$C = S_0 e^{d^*} N(d) - KN(d - \sigma_G\sqrt{T - T_0})$$

where:

$$d^* = \frac{1}{2}(r - \frac{\sigma^2}{6})(T - T_0)$$

$$d = \frac{\log(S_0/K) + \frac{1}{2}(r + \frac{\sigma^2}{6})(T - T_0)}{\sigma\sqrt{(T - T_0)/3}}$$

and $N(\cdot)$ is the cumulative standard normal distribution function.

### Value

Numeric. The analytical price of the geometric average Asian option.

### References

Kemna, A.G.Z. and Vorst, A.C.F. (1990). "A Pricing Method for Options Based on Average Asset Values." *Journal of Banking and Finance*, 14, 113-129.

### Examples

```
price_kemna_vorst_geometric(
  S0 = 100, K = 100, r = 0.05, sigma = 0.2,
  T0 = 0, T_mat = 1, option_type = "call"
)
```

---

## print.arithmetic_bounds *Print Method for Arithmetic Asian Bounds*

---

### Description

Print Method for Arithmetic Asian Bounds

### Usage

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

### Arguments

| | |
|---|---|
| x | Object of class `arithmetic_bounds` |
| ... | Additional arguments (unused) |

### Value

Invisible x

---

## print.kemna_vorst_arithmetic *Print Method for Kemna-Vorst Arithmetic Results*

---

### Description

Print Method for Kemna-Vorst Arithmetic Results

### Usage

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

### Arguments

| | |
|---|---|
| x | Object of class "kemna_vorst_arithmetic" |
| ... | Additional arguments (ignored) |

## Value

Invisibly returns the input object x. Called for side effects (printing).

---

summary.kemna_vorst_arithmetic
*Summary Method for Kemna-Vorst Arithmetic Results*

---

## Description

Summary Method for Kemna-Vorst Arithmetic Results

## Usage

```
## S3 method for class 'kemna_vorst_arithmetic'
summary(object, ...)
```

## Arguments

| | |
|---|---|
| object | Object of class "kemna_vorst_arithmetic" |
| ... | Additional arguments (ignored) |

## Value

Invisibly returns the input object object. Called for side effects (printing).

# Index