A Comparative Analysis of the Black-Scholes Model and Monte Carlo Simulation on European Call Options

Ryan Mastropaolo

Fall 2025

MA 398 Senior Capstone

Mentor: Dr. Bernadette Boyle

December 9, 2025

Abstract

In this paper, we will compare the Black-Scholes model and the Monte Carlo simulation, which are two methods for estimating the fair value of a European call option. Although both methods are based on the same stochastic foundation of geometric Brownian motion, the Black-Scholes method provides a closed-form analytical solution, whereas the Monte Carlo method relies on repeated numerical sampling. We will outline the mathematical framework behind each model and then examine how Monte Carlo estimates converge to the Black-Scholes price under identical assumptions. We will look at quantitative simulations using real market data when a sufficiently large number of paths are used. This demonstrates the consistency between the two methods when pricing European call options, highlighting the trade-off between analytical precision and computational flexibility. Finally, we will mathematically prove that stock prices follow a log-normal distribution.

1 Introduction

Quantitative finance rests on one key idea: randomness can be measured, modeled, and even predicted through mathematics. Among its most powerful tools, the Black-Scholes equation and Monte Carlo simulation stand as two distinct paths toward the same goal of understanding and pricing risk in an uncertain financial world. The Black-Scholes model represents one of the most influential breakthroughs in financial mathematics, offering a closed-form solution for the pricing of European options under specific market assumptions. In contrast, the Monte Carlo simulation provides a flexible numerical method capable of approximating expected option payoffs through repeated random sampling. Both models have their strengths and weaknesses. Black-Scholes is limited to European options but is significantly faster than Monte Carlo. On the other hand, Monte Carlo is more flexible and can handle more complex financial instruments but is significantly slower to get precise results. Both of these models are widely used in the financial industry to produce an expected fair value for an option or derivative. Despite their methodological differences, analytical vs. numerical, both frameworks are grounded in the same stochastic foundation. In section 2, we discuss the basic financial terms necessary to understand option pricing and risk-neutral valuation. Next, in sections 3 and 4, we provide background information on the Black-Scholes model and the Monte Carlo Simulation, respectively. In section 5, we explore the mathematical conditions under which the Monte Carlo approach converges to the Black-Scholes solution as the number of simulations approaches infinity, given that you simulate it under the same assumptions including geometric Brownian motion, volatility, and risk neutral measure. To conclude, in section 6, we will mathematically prove that stock prices follow a log-normal distribution.

2 Definitions in Financial Mathematics

In this section, we discuss a few terms which are commonly used in the context of financial mathematics.

Definition 2.1. *An **option** is a financial contract that gives the holder the right to buy or sell an asset at a predetermined strike price on or before a specified expiration date.*

(i) *A **call option** is a contract that gives the holder the right to buy the underlying asset at a predetermined strike price on or before a specified expiration time.*

(ii) *A **put option** is a contract that gives the holder the right to sell the underlying asset at a predetermined strike price on or before a specified expiration time* [1].

Definition 2.2. *The **expiration time** or **maturity** is the fixed future time on a given date at which the option contract expires. The expiration time is often denoted as T.*

(i) *An **American option** may be exercised at any time prior to its expiration* *date.*

(ii) *A* ***European option*** *can only be exercised at its expiration date* [1][6].

Definition 2.3. *The* ***strike price*** *is the fixed, predetermined price at which the underlying asset will be delivered should the holder choose to buy or sell the option. The strike price is often denoted as K* [2].

Definition 2.4. *The* ***risk-free rate*** *is the constant drift term applied to the simulated price process under the risk-neutral measure. This represents the rate at which money can be borrowed or lent without risk and is used to discount the expected payoff of the derivative. Risk-free rate is often denoted as r* [3].

(i) The risk-neutral measure *is the assumption that investors are indifferent to risk, so all assets are expected to grow according to the risk-free rate* [2].

Definition 2.5. *The* ***volatility*** *is a measure of the degree of variation in the price of a financial asset or market over time, often interpreted as a "measure of the speed of the market." Volatility is often denoted as σ* [2].

Definition 2.6. *A* ***stochastic differential equation (SDE)*** *is the mathematical foundation for modeling the price of a financial asset, including options. This equation describes how a random process X_t evolves in continuous time with a deterministic trend and a random "noise" term. The equation is defined by*

$$dX_t = m(t, X_t)\, dt + \sigma(t, X_t)\, dB_t,$$

where B_t is Brownian motion, $m(t, X_t)$ is drift (the deterministic trend), and $\sigma(t, X_t)$ is volatility (the random "noise") [4].

Definition 2.7. ***Geometric Brownian Motion (GBM)*** *is a stochastic process in which asset prices change continuously over very small intervals of time, and the position, specifically how the asset's price changes from one amount to the next, is being altered by random amounts due to drift and random noise. Because GBM is defined through a stochastic differential equation with drift and volatility terms proportional to the current asset price, it is a special case of SDEs used to model asset prices* [5].

Geometric Brownian Motion follows the formula:

$$dS = \mu S\, dt + \sigma S\, dW, \tag{1}$$

where S denotes the price of the asset at time t, μ denotes the drift, otherwise known as the average rate of return of the underlying stock, σ denotes the volatility of the stock, and W denotes a standard Brownian motion. A Brownian motion $W(t)$, also known as a Wiener process, is a continuous stochastic process that is a source of randomness. $W(t)$ has a normal distribution of $N(0, t)$ for all $t > 0$, where $N(\mu, \sigma^2)$ represents the normal distribution with mean μ and variance σ^2. In addition, dS denotes the change in the price of the asset, and dW denotes the change in the standard Brownian motion [6].

Definition 2.8. ***Itô's Process*** *is a continuous stochastic process used to model the evolution of random variables whose behavior depends on both time and a Wiener process. It follows the form*

$$dX = a(x, t)dt + b(x, t)dz, \tag{2}$$

where dz *is an increment of the Wiener process,* $a(x, t)$ *is the drift term, and* $b(x, t)$ *is the volatility, which controls the magnitude of randomness. Itô's process forms the foundation for Itô's lemma and the stochastic differential equations used in models such as geometric Brownian motion previously defined in equation* (2) [10].

Definition 2.9. ***Itô's lemma*** *describes how a function of a stochastic process evolves over time. If a variable* x *follows Itô's process in equation* (2)*, then* x *has a drift rate of* a *and a variance rate of* b^2*. Itô's lemma shows that a function* $G(x, t)$ *follows the stochastic differential equation*

$$dG = \left(\frac{\partial G}{\partial x} a + \frac{\partial G}{\partial t} + \frac{1}{2} \frac{\partial^2 G}{\partial x^2} b^2 \right) dt + \frac{\partial G}{\partial x} b \, dz. \tag{3}$$

This result shows that $G(x, t)$ *follows an Itô process driven by the Wiener source of randomness. Moreover, Itô's lemma determines how any function of a stochastic variable inherits drift and volatility.*

(i) Special case: Black-Scholes model. If the underlying variable is the stock price S, which follows geometric the Brownian motion formula stated in equation (1), then applying Itô's lemma to a function $G(S, t)$, which represents a derivative whose value depends on the stock price S and time t, yields the Black-Scholes form

$$dG = \left(\frac{\partial G}{\partial S} \mu S + \frac{\partial G}{\partial t} + \frac{1}{2} \frac{\partial^2 G}{\partial S^2} \sigma^2 S^2 \right) dt + \frac{\partial G}{\partial S} \sigma S \, dz. \tag{4}$$

This special case is fundamental in option pricing because it shows that any derivative $G(S, t)$ is driven by the same underlying uncertainty as the stock price [10].

3 Black-Scholes Mathematical Model

The Black-Scholes model is a mathematical formula used to estimate the fair price of a European option, which gives the right to buy or sell an asset at a specific price in the future. It assumes that stock prices move randomly but continuously over time and follows geometric Brownian motion. To calculate how much the option should be worth today, the model takes into account several key factors, including the current stock price (S), the option's strike price (K), the time until expiration (T), the risk-free interest rate (r), and the asset's volatility (σ). In a risk-neutral world, the expected growth rate of any asset is the risk-free rate r, not the real-world drift μ. This substitution arises from delta-hedging, which eliminates randomness and forces the derivative to earn the risk-free return. To price a European call option, we apply the standard Black–Scholes PDE under the terminal payoff condition

$$C(S,T) = max(S-K,0) \tag{5}$$

where max returns the larger of the two values $S-K$ or 0. Therefore, if the stock price is above the strike, the payoff is $S-K$, and if the stock price is below the strike, the payoff is 0. This yields the closed-form Black-Scholes formula for value of a European call option $C(S,t)$ as follows: [8]

$$C(S,t) = N(d_1)S - N(d_2)Ke^{-r(T-t)} \tag{6}$$

Where $N(d_1)$ and $N(d_2)$ are the normal distributions of d_1 and d_2, respectively. Also,

$$d_1 = \frac{\ln(\frac{S}{K}) + (r + \frac{\sigma^2}{2})(T-t)}{\sigma\sqrt{T-t}} \tag{7}$$

and

$$d_2 = \frac{\ln(\frac{S}{K}) + (r - \frac{\sigma^2}{2})(T-t)}{\sigma\sqrt{T-t}} \tag{8}$$

Also, d_2 can be written as:

$$d_2 = d_1 - \sigma\sqrt{T-t} \tag{9}$$

In essence, the Black-Scholes model helps traders and investors understand how market uncertainty and time affect the value of financial options. The Black-Scholes Model follows seven restrictive assumptions listed below [7].

1. The risk-free interest rate, r, is known and constant through time.

2. The stock price follows geometric Brownian motion with constant drift μ and volatility σ.

3. The stock pays no dividends or other distributions.

4. It is a European option, thus it can only be exercised at the expiration time.

5. It is a frictionless market; therefore, there are no transaction costs in buying or selling the stock or the option.

6. Fractional shares are permitted, thus it is possible to borrow any fraction of the price of a security to buy it or to hold it, at the short-term interest rate.

7. There are no direct consequences for short selling. When a seller does not own a security, they accept the price of the security from the buyer and agree to settle later by paying an amount equal to its value on that future date.

4 Monte Carlo Simulation

A Monte Carlo simulation is a computational method that employs repeated random sampling and statistical analysis to estimate possible outcomes of uncertain processes. Since stock returns are inherently uncertain and influenced by randomness, the Monte Carlo method is a natural fit for modeling risk, volatility, and potential future price scenarios. Similar to real option analysis, the Monte Carlo simulation can be applied to financial instruments, specifically options [9]. The Monte Carlo method prices European call options by first simulating thousands or sometimes millions, of price paths of the option following the risk-neutral measure and geometric Brownian motion. Next, the payoff for each path is calculated, and a single mean value of the payoffs is calculated to obtain the expected payoff in a risk-neutral world. Finally, the mean is discounted at the risk-free rate to obtain the current fair value of the option [8]. The complete Monte Carlo pricing formula is as follows: [10]

$$C_0 = e^{-rT} \frac{1}{N} \sum_{i=1}^{N} (S_i(T) - K)^+ \tag{10}$$

Where C_0 is the estimated fair value of the European call option at time 0, and N is the total number of simulated stock price paths generated. In addition, e^{-rT} is the discount factor applied to the expected payoff at the expiration time T, and $S_i(T)$ is the stock price at maturity, so $(S_i(T) - K)^+$ is the payoff at maturity.

5 Qualitative Comparison of Convergence: Monte Carlo Simulation vs. Black-Scholes Model

The significance behind the Black-Scholes model and the Monte Carlo simulation is that they both estimate the fair value of a financial derivative; however, they handle uncertainty and risk in different ways [6]. Both methods are based on the stochastic foundation of geometric Brownian motion, but they differ in that one offers a closed-form analytical solution where uncertainty is embedded in the formula through parameters like volatility and the risk free rate [7][10]. Moreover, the other offers a numerical approximation through repeated random sampling, handling uncertainty by directly simulating many possible price paths, each incorporating random shocks from the Wiener process [9][3].

The Black-Scholes model is widely known for its simplicity and efficiency, due to its explicit formula. Moreover, its strength is also a limitation because this model uses restrictive assumptions, including constant volatility, frictionless markets, and the absence of early exercise. As a result, its accuracy diminishes when applied to more complex, path-dependent, or American-style derivatives [2].

Alternatively, the Monte Carlo simulation allows for greater modeling flexibility at the expense of analytical simplicity. This method allows for more realistic market scenarios, including time-varying volatility, discrete dividends, early exercise opportunities, and multi-asset correlations, which can all be incorporated without altering the core framework. The key limitation of Monte Carlo simulations is the computational intensity since this method requires a large number of paths to reach a precise outcome, in turn resulting in a slower process [9].

The convergence of the two methods occurs when the same assumptions are made and there is a significant number of simulations. Black-Scholes produces a direct value from its formula, whereas Monte Carlo approaches this value as repeated sampling smooths out randomness, and the estimate stabilizes. In the simulation, error never fully disappears, but with enough runs and consistent modeling of parameters, including volatility, drift, and the risk-free rate, the simulated price from Monte Carlo aligns with the analytical price from Black-Scholes. Essentially, the closed-form Black-Scholes formula marks the target value, and the Monte Carlo simulation closes the gap until its estimate matches that target under the same market and stochastic setup [6].

In essence, the contrast between the two methods illustrates a fundamental divide between theory and computation. The Black-Scholes model represents an idealized framework in which mathematics delivers a closed-form solution, while Monte Carlo simulations reflect a practical, data-driven method in which uncertainty is managed through statistical approximation. Despite their differences, both methods ultimately converge on the same financial truth: under the same assumptions, the expected value of the simulated payoffs in Monte Carlo approaches the Black-Scholes price as the number of simulations trends to infinity.

5.1 Table demonstrating this convergence

To further show the relationship between the analytical Black-Scholes solution and the numerical Monte Carlo estimator, this section presents a quantitative comparison of option prices computed under identical market assumptions. In this table, we used real market data through the yfinance Python library that retrieves historical and current financial information from Yahoo Finance to evaluate a standard (one-dimensional) European call option for some of the largest corporations including Apple (AAPL), Tesla (TSLA), Microsoft (MSFT), Google (GOOG), and Amazon (AMZN). A python program was created to automatically extract the latest stock price (S_0), historical volatility (σ) computed from log returns, dividend yield (q), and the time to maturity (T) based on a chosen expiration date. The strike price (K) was selected to be close to the current market price, allowing for a consistent comparison across the stocks. Also, the risk-free rate (r) was set using publically available Treasury data. After these variables were collected, the pricing methods were integrated into the code. First, a Monte Carlo simulation was coded to model the stock price following geometric Brownian motion by generating a large number of stochastic price paths and averaging the discounted payoff $max(S_T - K, 0)$, where max returns the larger of the two values $(S_T - k)$ or 0. Therefore, if the stock price is above the strike, the payoff is $(S_T - K)$, and if the stock price is below the strike, the payoff is 0. Second, the closed-form Black-Scholes formula, previously stated in equations $(6-9)$, was coded using the same inputs to compute the analytical value of a European call option. After both prices were computed, we were able to compare both values directly, which are displayed in Table 1 below.

Table 1: One-dimensional European call option price using actual data with the risk-free rate $r = 0.042$

Ticker	S_0	**K**	**T**	$\boldsymbol{\sigma}$	**q**	**MC**	**BS**
AAPL	270.08	270.00	0.008774	0.2590	0.002851	2.7000	2.7132
TSLA	445.08	445.00	0.008839	0.7416	0.000	12.5000	12.5640
MSFT	504.63	505.00	0.008832	0.2275	0.006579	4.3000	4.3214
GOOG	290.27	290.00	0.008779	0.3331	0.002824	3.8000	3.8187
AMZN	248.80	250.00	0.008768	0.3521	0.000	2.7550	2.7694

The significance of this table is that the Monte Carlo and Black-Scholes prices are nearly identical for every ticker, with a difference of only a few cents. Therefore, this table supports evidence of convergence: for simple European calls, the Monte Carlo estimator converges to the Black-Scholes price as the number of simulated paths grows.

6 Proof that Stock Prices Follow a Log-Normal Distribution

As explored earlier, both Black-Scholes and Monte Carlo follow geometric Brownian motion to price European call options, and geometric Brownian motion gives us that stock prices follow a log-normal distribution. Therefore, since the Monte Carlo simulation and the Black-Scholes formula both depend on the same GBM framework to model the evolving stock price, the log-normal result is the foundation connecting these two pricing methods [10]. Moreover, in this section, we will mathematically prove that stock prices follow a log-normal distribution.

Theorem 6.1. *Let the stock price process $S(t)$ satisfy the geometric Brownian motion*

$$dS(t) = \mu S(t)\,dt + \sigma S(t)\,dW(t),$$

where μ and $\sigma > 0$ are constants and $W(t)$ is a standard Brownian motion. Then the explicit solution is

$$S(t) = S(0)\exp\left(\left(\mu - \tfrac{1}{2}\sigma^2\right)t + \sigma W(t)\right),$$

which implies that $S(t)$ is log-normally distributed for every $t > 0$.

Proof. Assume that the stock price process $S(t)$ satisfies the geometric Brownian motion equation

$$dS(t) = \mu S(t)\,dt + \sigma S(t)\,dW(t), \qquad t \geq 0, \tag{11}$$

where $\mu \in \mathbb{R}$ and $\sigma > 0$ are constants, and $W(t)$ is a standard Brownian motion for all $t \geq 0$. We also assume $S(0) > 0$. Let $f(S,t) = \ln(S)$. Then the partial derivative $f_S = \frac{1}{S}$ and the second partial derivative $f_{SS} = -\frac{1}{S^2}$. By Itô's lemma,[10]

$$
\begin{aligned}
df &= \left[f_t + \mu S f_S + \frac{1}{2}\sigma^2 S^2 f_{SS} \right] dt + \sigma S f_S dW \\
&= \left[0 + \mu S \frac{1}{S} + \frac{1}{2}\sigma^2 S^2 \frac{-1}{S^2} \right] dt + \sigma S \frac{1}{S} dW \\
&= \left[\mu - \frac{1}{2}\sigma^2 \right] dt + \sigma \, dW.
\end{aligned}
$$

Integrating this stochastic equation from 0 to t, we yield the following:

$$
\ln(S(t)) - \ln(S(0)) = (\mu - \frac{1}{2}\sigma^2)t + \sigma W(t).
$$

Therefore,

$$
S(t) = S(0) \exp\left((\mu - \frac{1}{2}\sigma^2)t + \sigma W(t) \right).
$$

Since μ and σ are fixed parameters and t is the fixed time, the term $(\mu - \frac{1}{2}\sigma^2)t$ is a finite number with no randomness, otherwise known as "deterministic." However, for any fixed t, the standard Brownian motion $W(t)$ has a normal distribution of $N(0, t)$ and is a source of randomness. Thus, the term $\sigma W(t)$ is normally distributed with mean 0 and variance $\sigma^2 t$ and introduces randomness into the exponential.

Let

$$
Y = (\mu - \tfrac{1}{2}\sigma^2)t + \sigma W(t).
$$

Since $W(t)$ is normal and Y, the sum of a constant and a normal random variable, is also normal, it follows that Y is normally distributed. Because $S(t) = S(0)e^Y$, and any random variable of the form $X = e^Y$ is log-normally distributed whenever Y is normally distributed, the process $S(t)$ follows a log-normal distribution [10].

Therefore, we obtain the log-normal model of stock prices,

$$S(t) = S(0) \exp\left((\mu - \tfrac{1}{2}\sigma^2)t + \sigma W(t) \right). \tag{12}$$

$\square$

References

[1] P. Wilmott, S. Howison, J. Dewynne, *The Mathematics of Financial Derivatives.* Cambridge University Press, Cambridge, 1997.

[2] S. Natenberg, *Option Volatility and Pricing: Advanced Trading Strategies and Techniques.* McGraw-Hill Education, New York, NY, 2015.

[3] Bouchaud, J.-P., Potters, M., *Hedge Your Monte Carlo.* Capital Fund Management, Paris, 2001.

[4] G. Lawler, *Stochastic Calculus: An Introduction with Applications.* University of Chicago, 2000.

[5] A. Ermogenous, *Brownian Motion and Its Applications in the Stock Market.* Undergraduate Mathematics Day, Electronic Proceedings, University of Dayton, 2006.

[6] Q. Jiang, *Comparison of Black–Scholes Model and Monte-Carlo Simulation on Stock Price Modeling.* Atlantis Press, ICEMCI, 2019

[7] F. Black, M. Scholes, *The Pricing of Options and Corporate Liabilities.* The Uniersity of Chicago Press, 1973.

[8] H.Shahbandarzadeh, K.Salimifard, R.Moghdani *Application of Monte Carlo Simulation in the Assessment of European Call Options.* Iranian Journal of Management Studies, vol. 6, no. 1, 2013.

[9] S. Raychaudhuri, *Introduction to Monte Carlo Simulation*, IEEE, Piscataway, NJ, 2008.

[10] J. Hull, *Options, Futures, and Other Derivatives 8th ed*, Pearson, 2012.

Appendix

Black-Scholes Monte Carlo Price Calculator Code

Attached below is the full Python code to calculate the expected prices of the stocks using Black-Scholes equation and the Monte Carlo simulation.

```python
# --- Consistent BS vs MC pipeline (call option) ---
import yfinance as yf
import numpy as np
from datetime import datetime, timezone, timedelta
from math import log, sqrt, exp
from scipy.stats import norm
from scipy.optimize import brentq
import pandas as pd

# -------------------------
# Black-Scholes Call Price
# -------------------------
def bs_call_price(S, K, T, r, q, sigma):
    if T <= 0:
        raise ValueError(f"T must be > 0, got {T}")
    if sigma <= 0:
        raise ValueError(f"sigma must be > 0, got {sigma}")
    d1 = (log(S / K) + (r - q + 0.5 * sigma**2) * T) / (sigma * sqrt(T))
    d2 = d1 - sigma * sqrt(T)
```

```python
    return S * exp(-q * T) * norm.cdf(d1) - K * exp(-r * T) * norm.cdf(d2)


# -------------------------
# Implied Volatility Solver
# -------------------------
def implied_vol_call(S, K, T, r, q, market_price, lo=1e-6, hi=5.0):
    def f(sig):
        return bs_call_price(S, K, T, r, q, sig) - market_price
    f_lo, f_hi = f(lo), f(hi)
    if f_lo * f_hi > 0:
        raise RuntimeError("IV solver failed to bracket the root.")
    return brentq(f, lo, hi, maxiter=200, xtol=1e-10)


# -------------------------
# Risk-neutral Monte Carlo (GBM)
# -------------------------
def mc_call_price(S0, K, T, r, q, sigma, paths=200_000, steps=252, seed=42):
    if T <= 0:
        raise ValueError("T must be > 0 for simulation.")
    rng = np.random.default_rng(seed)
    dt = T / steps
    mu = r - q
    Z = rng.standard_normal((paths, steps))
    increments = (mu - 0.5 * sigma**2) * dt + sigma * np.sqrt(dt) * Z
```

```python
    ST = S0 * np.exp(np.cumsum(increments, axis=1)[:, -1])

    payoff = np.maximum(ST - K, 0.0)

    disc = np.exp(-r * T)

    price = disc * payoff.mean()

    stderr = disc * payoff.std(ddof=1) / np.sqrt(paths)

    return price, stderr


# -------------------------

# Auto dividend yield (q)

# -------------------------

def fetch_dividend_yield_from_yf(tkr, S0=None):

    """

    Dividend yield q (decimal):

      1) fast_info['dividend_yield']  (ideal)

      2) TTM cash dividends / S0

      3) Last 4 dividends / S0

      4) Else 0.0

    """

    tk = yf.Ticker(tkr)


    # --- 1) fast_info

    try:

        fi = tk.fast_info

        dy = fi.get("dividend_yield", None)
```

```
    if dy is not None and np.isfinite(dy) and dy >= 0:
        return float(dy)
except:
    pass

# --- Ensure S0 exists
if S0 is None or not np.isfinite(S0):
    try:
        S0 = float(tk.history(period="5d")["Close"].dropna().iloc[-1])
    except:
        return 0.0

# --- 2) TTM cash dividends
try:
    divs = tk.dividends
    if divs is not None and not divs.empty:
        divs.index = pd.to_datetime(divs.index).tz_localize(None)
        cutoff = pd.Timestamp.utcnow().tz_localize(None) - pd.Timedelta(days=365)
        ttm = float(divs[divs.index >= cutoff].sum())
        if ttm > 0:
            return ttm / S0

        # --- 3) Last 4 payouts (typical quarterly dividends)
        last4 = float(divs.tail(4).sum())
```

```
            if last4 > 0:
                return last4 / S0
    except:
        pass

    # No dividends
    return 0.0

# -------------------------
# Collecting S0, K, T, market price
# -------------------------
def fetch_inputs_from_yf(tkr, exp=None, use_put_chain=False):
    tk = yf.Ticker(tkr)

    S0 = float(tk.history(period="1d")["Close"].iloc[-1])

    all_exps = tk.options
    if not all_exps:
        raise RuntimeError("No listed options.")
    if exp is None:
        exp = all_exps[0]

    ch = tk.option_chain(exp)
    table = ch.puts if use_put_chain else ch.calls
```

```
table = table.dropna(subset=["strike"])


atm_idx = (table["strike"] - S0).abs().idxmin()

row = table.loc[atm_idx]

K = float(row["strike"])


bid = float(row.get("bid", np.nan))

ask = float(row.get("ask", np.nan))

last = float(row.get("lastPrice", np.nan))


if np.isfinite(bid) and np.isfinite(ask) and bid > 0 and ask > 0:

    market_price = 0.5 * (bid + ask)

elif np.isfinite(last) and last > 0:

    market_price = last

else:

    mark = float(row.get("mark", np.nan))

    if np.isfinite(mark) and mark > 0:

        market_price = mark

    else:

        raise RuntimeError("Could not determine market option price.")


now = datetime.now(timezone.utc)

exp_dt = datetime.fromisoformat(exp).replace(tzinfo=timezone.utc)

T = (exp_dt - now).total_seconds() / (365.25 * 24 * 3600)
```

```python
    if T <= 0:
        raise ValueError(f"T <= 0 for {exp}")

    return {
        "tkr": tkr,
        "S0": S0,
        "K": K,
        "exp": exp,
        "T": T,
        "market_price": market_price,
        "row": row,
        "chain_table": table
    }

# -------------------------
# Example run
# -------------------------
tkr = "AMZN"
r = 0.042   # 4.2% risk-free rate

data = fetch_inputs_from_yf(tkr)
S0 = data["S0"]; K = data["K"]; T = data["T"]; market_price = data["market_price"]

# AUTO dividend yield
```

```
q = fetch_dividend_yield_from_yf(tkr, S0=S0)

# IV from market price
iv = implied_vol_call(S0, K, T, r, q, market_price)

# Black-Scholes price
bs_price = bs_call_price(S0, K, T, r, q, iv)

# Monte Carlo price
mc_price, mc_se = mc_call_price(S0, K, T, r, q, iv)

print("Dividend yield q =", q)
print(f"Ticker: {tkr}")
print(f"S0={S0:.4f}, K={K:.4f}, T={T:.6f}, r={r:.4%}, q={q:.4%}")
print(f"Market option price ≈ {market_price:.4f}")
print(f"Implied Vol (call): {iv:.4%}")
print(f"BS price:          {bs_price:.4f}")
print(f"MC price:          {mc_price:.4f} (± {1.96*mc_se:.4f})")
```

Example Output

Below is an example output of the stock ticker AMZN representing the Amazon stock:

```
Dividend yield q = 0.0

Ticker: AMZN

S0=248.8000, K=250.0000, T=0.008768, r=4.2000\%, q=0.0000%,

Market option price ≈ 2.7550

Implied Vol (call): 35.2057%

BS price: 2.7550

MC price: 2.7694 (± 0.0199)
```