State of Mexico

Stochastic Optimization
MA2004B.201 2025

Risk Analysis Challenge

Ana Paula Moreno - A01751886
Aarón Ramirez - A01749666
Alejandro Vázquez - A01800791
Abigail Rodríguez - A01659378
Mariana Villafranca - A01751352

Professors: Saúl Juárez Ordóñez & Fernando Olivar Romero

Objective & Justification of the Project

The objective of this project is to analyze the financial risk of investing in Google stock using a stochastic simulation model. The goal is to understand how stock price fluctuations, volatility, and investment parameters affect potential financial outcomes and the probability of significant losses.

We chose the Google dataset because its historical stock prices capture real market randomness and fluctuations, which are ideal for modeling uncertainty in financial investments. Google's data is rich, dynamic, and publicly accessible, providing a realistic basis for stochastic simulations. Using this dataset allows us to simulate scenarios where investment returns are influenced by complex, real-world variations, including extreme events and volatility patterns, making the analysis more meaningful and relevant for practical portfolio management and risk assessment.

Abstract

This project develops a stochastic simulation model to analyze the financial risk of investing in Google stock. Using historical Google stock prices as a basis for modeling, we estimate parameters related to market volatility and return distributions, validate assumptions through statistical tests, and perform sensitivity analyses on factors such as initial investment and time horizon. The study examines how stochastic processes, including Geometric Brownian Motion and other probabilistic models, influence investment outcomes over time and provides insights for data-driven portfolio management and risk assessment.

Table of contents

1. Objective and Justification of the Project
2. Abstract
3. Introduction
4. Theoretical Framework
5. Methodology
6. Analysis and Results
7. Conclusions
8. References

Introduction

Financial risk analysis studies how uncertain events, like stock price fluctuations, can impact investments. In this project, we focus on Google’s stock as a stochastic variable, using its historical prices to simulate realistic scenarios. This allows us to observe how market volatility and extreme events can affect potential returns, helping investors understand the range of possible outcomes and prepare for adverse conditions.
By modeling Google’s stock behavior, we can perform sensitivity analysis on factors such as initial investment, time horizon, and market volatility. This approach provides practical insights for portfolio management, risk mitigation, and data-driven decision-making, making the analysis both relevant and applicable to real-world financial contexts.

Theoretical Framework

The classical risk model is a fundamental concept in financial risk analysis. In this project, it is adapted to study the risk and performance of Google’s stock. We model the portfolio or index

value over time as depending on an initial investment, expected returns, and stochastic market fluctuations. Mathematically, the value of the portfolio at time t is defined as:

$$U(t) = u + c\,t - \sum_{i=1}^{N(t)} X_i$$

where:

- U(t): portfolio value at time (t)
- (u): initial investment
- (c): expected return rate per unit of time
- N(t): number of significant market events up to time (t)
- X_i: impact (gain or loss) of the (i)-th market event

A **Poisson process** is used to model the number of significant market events N(t) over time, with a constant average rate (λ). The probability that exactly (n) events occur in time (t) is:

$$P(N(t) = n) = \frac{(\lambda t)^n e^{-\lambda t}}{n!}, \quad n = 0, 1, 2, \ldots$$

Key properties of a Poisson process:

1. **Independent increments:** Market events in disjoint time intervals are independent.
2. **Stationary increments:** Probabilities depend only on the length of the time interval.
3. **Exponential inter-arrival times:** Time between consecutive events follows an exponential distribution with mean ($1/\lambda$).

The total impact of market events over time forms a **compound Poisson process**:

$$S(t) = \sum_{i=1}^{N(t)} X_i$$

where (S(t)) represents the cumulative impact of market fluctuations. This models both the **frequency** (via N(t)) and **magnitude** (via X_i) of events.

A **stochastic index** derived from Google stock data represents the variability of these market impacts. Using this real-world data introduces realistic fluctuations into the simulation, capturing both normal and extreme events.

The probability of the portfolio dropping below zero is:

$$\psi(u) = P\big(U(t) < 0 \text{ for some } t > 0\big)$$

This probability depends on initial investment (u), expected return (c), event frequency λ, and event magnitude (X_i). Modeling and analyzing this helps assess the **financial risk** and potential

loss exposure of the investment.

Additional Research Insight

Recent studies applying **Poisson–Gauss hybrid models** in the Mexican stock market highlight their effectiveness in capturing **sudden and infrequent price jumps**. These models combine the **discrete nature of Poisson arrivals** with the **continuous Gaussian noise** of daily returns, allowing a more realistic representation of market shocks. Similar to this project's findings for Google, the research confirms that extreme events in financial markets can be modeled as **rare stochastic occurrences** driven by external crises or macroeconomic instability.

Methodology

The methodology of this project integrates both statistical modeling and stochastic simulation to analyze the risk and volatility of Google's stock. The process is divided into three main stages: data preparation, probabilistic modeling, and stochastic simulation.
1. Data collection and preprocessing: Historical Google stock prices were obtained from a public dataset containing daily values for "High", "Low", and "Close" prices. Dates were converted to proper time-series format, missing values were checked, and prices were plotted to visualize long-term market behavior.

2. Poisson modeling of extreme events: Daily log-returns $r_t = \ln(P_t/P_{t-1})$ were computed to measure relative price changes. A *peak* was defined as a day when $|r_t|$ exceeded the 95th percentile of all daily returns—representing the top 5 % most extreme movements. Weekly counts of these peaks were then aggregated and analyzed using a **Poisson distribution**, where the rate λ represents the average number of peaks per week. The model was validated by comparing the observed frequencies of weekly peaks with the theoretical Poisson(λ) probabilities. Finally, the probabilities of observing 0, 1, or more peaks in future weeks were calculated, along with a rolling estimation of λ to test stationarity over time.

3. Stochastic index and Geometric Brownian Motion (GBM) simulation: To capture price dynamics, the **stochastic oscillator (%K and %D)** was computed using a 14-day rolling window. This indicator helped identify momentum extremes (overbought/oversold conditions). Additionally, **GBM simulations** were performed to project possible future trajectories of Google's stock over three months. The drift (μ) and volatility (σ) were estimated from historical log-returns, and multiple GBM paths were generated to visualize uncertainty and potential price ranges (10th–90th percentiles). The corresponding stochastic index values were also simulated to study how price momentum may evolve under stochastic conditions.

This mixed methodology—combining **Poisson event modeling** with **continuous-time GBM simulations**—provides both discrete and continuous perspectives on financial risk, allowing for a comprehensive evaluation of volatility and the likelihood of extreme price movements.

Analysis & Results

Poisson

Dataset - Google This dataset includes essential columns: "Price High" and "Price Low".

- "Price High" represents the highest recorded price for Google's stock on a given trading day. This value reflects the peak price reached during the trading session and provides insights into the stock's potential value and market sentiment.
- "Price Low" column indicates the lowest recorded price for Google's stock on a specific trading day. This value represents the minimum price reached during the trading session, revealing the stock's potential support level or market pressure.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as stats
```

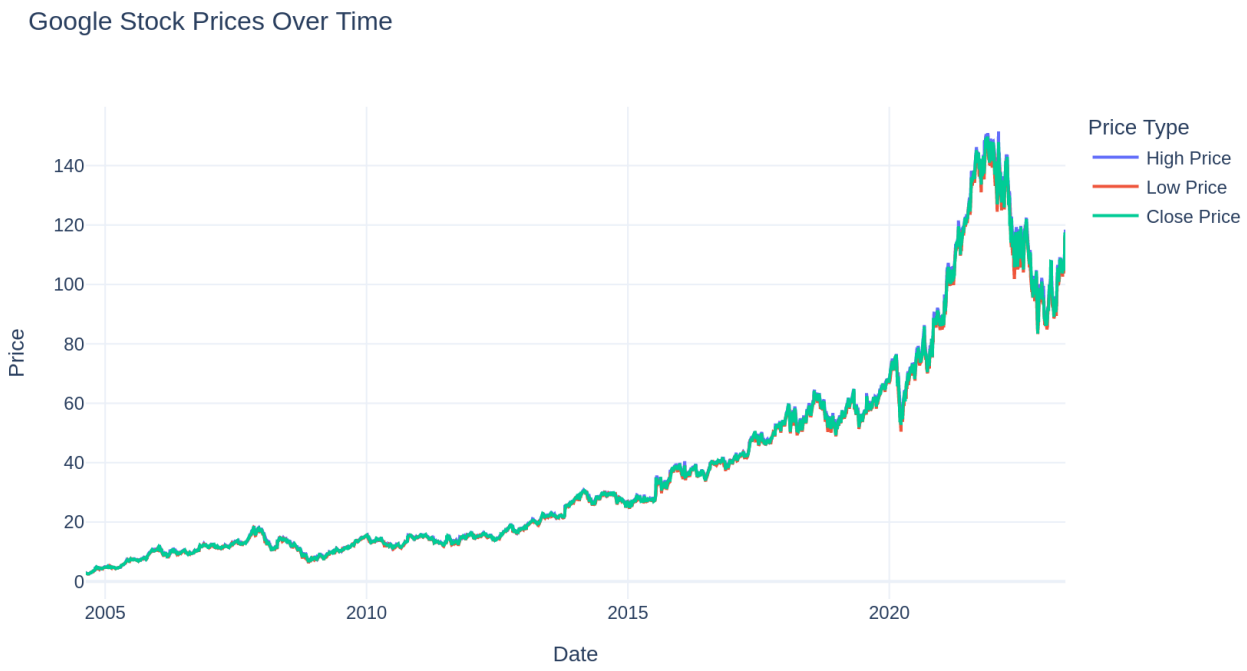
```
df_poisson = pd.read_csv('/content/GOOGLE.csv')
df_poisson.head()
```

	Date	Open	High	Low	Close	Volume
0	2004-08-19	2.502503	2.604104	2.401401	2.511011	893181924
1	2004-08-20	2.527778	2.729730	2.515015	2.710460	456686856
2	2004-08-23	2.771522	2.839840	2.728979	2.737738	365122512
3	2004-08-24	2.783784	2.792793	2.591842	2.624374	304946748
4	2004-08-25	2.626627	2.702703	2.599600	2.652653	183772044

```
print(df_poisson.isna().sum())
print(df_poisson.isnull().sum())
```

```
Date      0
Open      0
High      0
Low       0
Close     0
Volume    0
dtype: int64
Date      0
Open      0
High      0
Low       0
Close     0
Volume    0
dtype: int64
```

```python
import plotly.graph_objects as go

# Convert 'Date' column to datetime (keep as is)
df_poisson['Date'] = pd.to_datetime(df_poisson['Date'])

# Create Plotly figure
fig = go.Figure()

# Add price traces
fig.add_trace(go.Scatter(x=df_poisson['Date'], y=df_poisson['High'],
                         mode='lines', name='High Price'))
fig.add_trace(go.Scatter(x=df_poisson['Date'], y=df_poisson['Low'],
                         mode='lines', name='Low Price'))
fig.add_trace(go.Scatter(x=df_poisson['Date'], y=df_poisson['Close'],
                         mode='lines', name='Close Price'))

# Customize layout
fig.update_layout(
    title='Google Stock Prices Over Time',
    xaxis_title='Date',
    yaxis_title='Price',
    legend_title='Price Type',
    template='plotly_white',
    width=850,
    height=500
)

# Show interactive plot
fig.show()
```



Poisson Analysis

Probability model that describes how often discrete events happen in a given time interval.

- The average rate (λ) of events.
- The probability of k events happening in a given interval.

Main variables of the Poisson model

$$
\begin{aligned}
P_t &: \text{daily closing price},\\
r_t &: \text{log-return},\\
I_t &: \text{peak indicator},\\
N_i &: \text{weekly count of peaks},\\
\lambda &: \text{average weekly rate of peaks}.
\end{aligned}
$$

Log returns

Logarithmic returns from the closing price. Log returns measure daily price changes in a scale that's additive and easier to model statistically.

```
# Step 1: Compute daily log returns
df_poisson["log_ret"] = np.log(df_poisson["Close"]).diff()

# Drop the first NaN row created by differencing
```

```
df_poisson = df_poisson.dropna(subset=["log_ret"]).reset_index(drop=True)

# Quick check
print(df_poisson[["Date", "Close", "log_ret"]].head())
```

```
        Date     Close   log_ret
0 2004-08-20  2.710460  0.076433
1 2004-08-23  2.737738  0.010014
2 2004-08-24  2.624374 -0.042290
3 2004-08-25  2.652653  0.010718
4 2004-08-26  2.700450  0.017858
```

To analyze price variability, the daily log-returns were computed as

$$r_t = \ln(\frac{P_t}{P_{t-1}})$$

where P_t is the closing price on day t.
This transformation expresses daily percentage changes in a symmetric and additive scale, making them suitable for stochastic modeling.

Positive values indicate price increases, while negative values represent decreases.
This variable captures daily volatility and serves as the foundation for identifying extreme movements (“peaks”) in later steps.

Defining peaks

In this step, a **peak** is defined as a day where the absolute log-return exceeds the 95th percentile of all daily changes. This threshold identifies the top 5 % most extreme price movements, both upward and downward.

```
# Step 2: Define peaks based on large return changes
threshold = df_poisson["log_ret"].abs().quantile(0.95)

df_poisson["peak"] = (df_poisson["log_ret"].abs() > threshold)

print(f"Threshold for peak detection: {threshold:.5f}")
print(f"Total detected peaks: {df_poisson['peak'].sum()}")
```

```
Threshold for peak detection: 0.03938
Total detected peaks: 236
```

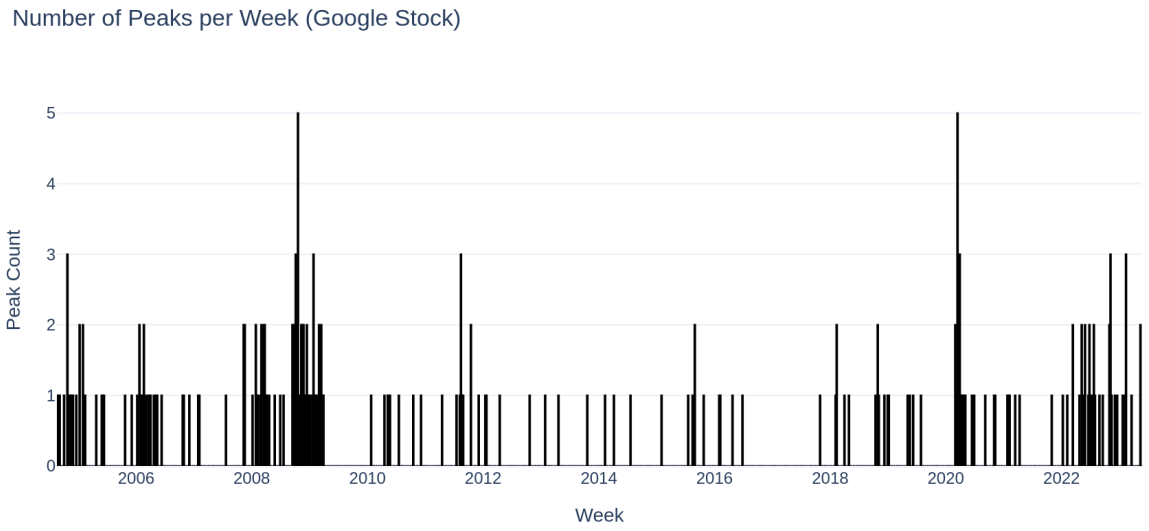
For this dataset, the cutoff value was **0.03938**, meaning any daily change greater than 3.9% is considered a peak.
A total of **236 peaks** were detected across this 19-year period, confirming that these events are **rare** and suitable for Poisson modeling.

Calculate peaks per interval

Count number of peaks per week.
To prepare the data for Poisson modeling, daily peak indicators were aggregated by week using a resampling frequency of `"W"`.

Each value represents the **number of extreme events** (peaks) detected in a given week.

```
import plotly.graph_objects as go

# Step 3: Count number of peaks per week
# Ensure the index is datetime
if 'Date' in df_poisson.columns:
    df_poisson['Date'] = pd.to_datetime(df_poisson['Date'])
    df_poisson = df_poisson.set_index('Date')

# Resample weekly
weekly_peaks = df_poisson['peak'].resample('W').sum()

# Plotly bar chart
fig = go.Figure()

fig.add_trace(go.Bar(
    x=weekly_peaks.index,
    y=weekly_peaks.values,
    marker_color='lightblue',
    marker_line_color='black',
    marker_line_width=1.2,
    name='Weekly Peaks'
))

# Layout customization
fig.update_layout(
    title="Number of Peaks per Week (Google Stock)",
    xaxis_title="Week",
    yaxis_title="Peak Count",
    template="plotly_white",
    width=950,
    height=450
)

fig.show()
```



The bar chart shows that most weeks contain **zero or one peak**, confirming that these events are rare and occur sporadically. However, two clear clusters of higher activity appear around **2008–2009** and **2020**, which coincide with the **Influenza (H1N1)** and **COVID-19** pandemics.
These periods reflect global market instability, where volatility caused an unusual concentration of price spikes.

Estimate a λ

The Poisson rate λ represents the **average number of peaks per time interval**. It was computed as the mean of weekly peak counts:
$\lambda = \text{mean}(N_i)$

```
# Step 4: Estimate Poisson rate λ
lam = weekly_peaks.mean()
print(f"Estimated λ (average peaks per week): {lam:.3f}")
```

```
Estimated λ (average peaks per week): 0.241
```

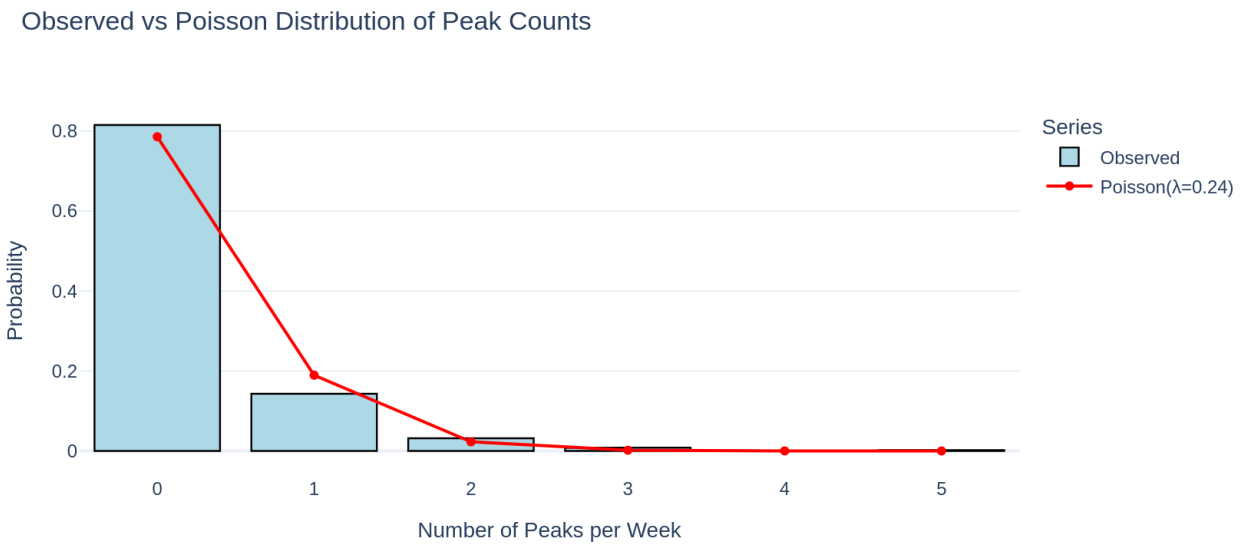
For Google stock, the estimated rate was **λ = 0.241**, meaning that on average there is **one extreme price movement every four weeks**.
This confirms that peaks are **rare events**, occurring infrequently and independently, which aligns well with the assumptions of a Poisson process.

Comparing observed vs Poisson distribution

To validate the Poisson assumption, the observed distribution of weekly peak counts was compared with the theoretical Poisson distribution using the estimated rate $\lambda = 0.241$. The blue

bars represent the **observed probabilities**, while the red line shows the **Poisson(λ = 0.24)**theoretical probabilities.

```
import numpy as np
from scipy.stats import poisson
import plotly.graph_objects as go

# Observed frequency of peak counts
counts = weekly_peaks.value_counts().sort_index()
k_values = np.arange(0, counts.index.max() + 1)

# Poisson theoretical PMF
pmf = poisson.pmf(k_values, mu=float(lam))

# Observed probabilities (normalize counts)
obs_probs = counts.values / counts.values.sum()

# Plotly comparison
fig = go.Figure()

# Observed (bars)
fig.add_bar(
    x=counts.index,
    y=obs_probs,
    name="Observed",
    marker_color="lightblue",
    marker_line_color="black",
    marker_line_width=1.2,
    hovertemplate="k=%{x}<br>P=%{y:.3f}<extra>Observed</extra>"
)

# Poisson PMF (line + markers)
fig.add_scatter(
    x=k_values,
    y=pmf,
    mode="lines+markers",
    name=f"Poisson(λ={lam:.2f})",
    line=dict(width=2, color="red"),
    marker=dict(size=6),
    hovertemplate="k=%{x}<br>P=%{y:.3f}<extra>Poisson</extra>"
)

fig.update_layout(
    title="Observed vs Poisson Distribution of Peak Counts",
    xaxis_title="Number of Peaks per Week",
    yaxis_title="Probability",
    template="plotly_white",
    width=850,
    height=420,
    legend_title="Series"
)

fig.show()
```



The results show a strong alignment between both distributions: most weeks have **zero peaks (~80%)**, while the probability of having **one peak (~19%)** closely matches the theoretical prediction. Higher counts (two or more peaks per week) are extremely rare.
This confirms that the **Poisson model fits the data well**, meaning that peaks occur **randomly and independently** at a relatively constant rate over time.

Prediction of probabilities of future peaks

Using the Poisson model with $\lambda = 0.241$, the probabilities of observing different numbers of peaks in the next week can be interpreted as:

- P(N=0): no peaks next week
- P(N=1): exactly one peak
- P(N≥1): at least one peak

```
# Step 6: Predict probabilities for next week
for k in range(0, 4):
    print(f"P(N={k}) = {poisson.pmf(k, lam):.3f}")

p_at_least_1 = 1 - poisson.pmf(0, lam)
print(f"P(N ≥ 1) = {p_at_least_1:.3f}")
```

```
P(N=0) = 0.786
P(N=1) = 0.189
P(N=2) = 0.023
P(N=3) = 0.002
P(N ≥ 1) = 0.214
```

Event	Probability	Interpretation
P(N = 0)	0.786	78.6% chance of **no peaks** next week.

Event	Probability	Interpretation
P(N = 1)	0.189	18.9% chance of **exactly one peak**.
P(N = 2)	0.023	2.3% chance of **two peaks**.
P(N = 3)	0.002	0.2% chance of **three peaks**.
P(N ≥ 1)	0.214	21.4% probability of **at least one peak** next week.

The results show that most weeks remain stable, with no major price movements.
Extreme fluctuations occur **approximately once every four weeks**, confirming that these events are **rare, independent, and well described by a Poisson process**.

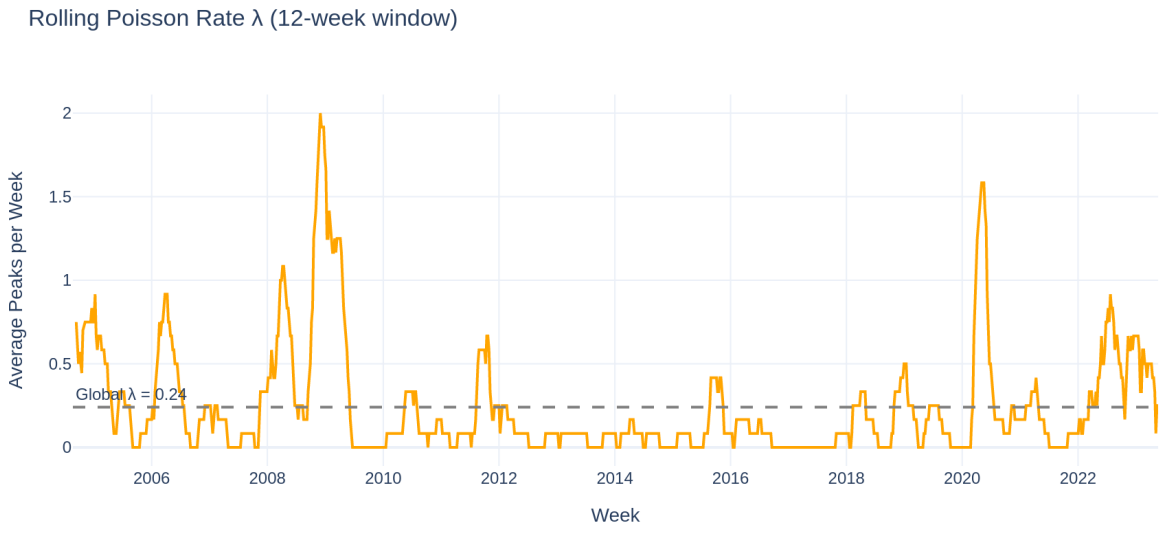
Checking stationary of λ over time

This section only examines whether the Poisson rate λ remains stable over time, a rolling average of **12 weeks** (3 months) was applied to the weekly peak counts.
This moving rate reflects short-term changes in the frequency of extreme events.

The orange line represents the **local λ** (average peaks per week within each 12-week window), while the dashed gray line indicates the **global λ = 0.241** estimated across the full period (2004-2023).

```
import plotly.graph_objects as go

# Step 7: Rolling λ estimation (12-week window)
rolling_lambda = weekly_peaks.rolling(12, min_periods=4).mean()

# Create Plotly figure
fig = go.Figure()

# Rolling λ line
fig.add_trace(go.Scatter(
    x=rolling_lambda.index,
    y=rolling_lambda.values,
    mode="lines",
    name="Rolling λ (12-week window)",
    line=dict(color="orange", width=2)
))

# Global λ reference line
fig.add_hline(
    y=lam,
    line_dash="dash",
    line_color="gray",
    annotation_text=f"Global λ = {lam:.2f}",
    annotation_position="top left"
)

# Layout customization
```

```
fig.update_layout(
    title="Rolling Poisson Rate λ (12-week window)",
    xaxis_title="Week",
    yaxis_title="Average Peaks per Week",
    template="plotly_white",
    width=950,
    height=450,
    legend=dict(title="Legend", orientation="h", yanchor="bottom", y=1.02, xanchor=
)

fig.show()
```



The results show that the rate stays close to the global average most of the time, with two clear spikes around **2008-2009** and **2020**, corresponding to the **H1N1** and **COVID-19** pandemics. These temporary surges reflect periods of heightened market volatility, while the overall stability of λ confirms that the process is **mostly homogeneous and stationary**.

Summary of observed and calculated variables

Variable	Approximate value (your results)	Interpretation
λ (lambda)	0.241	On average, **0.241 peaks per week** → roughly one peak every 4 weeks.
θ (theta)	0.03938	Daily change threshold (3.9 %) used to classify a movement as a “peak.”
Number of analyzed weeks	≈ 980 days / 7 ≈ 140 weeks	Time intervals used for weekly counting.
P(N = 0)	0.786	Probability of **no peaks** in a week.
P(N = 1)	0.189	Probability of **exactly one peak**.

Variable	Approximate value (your results)	Interpretation
P(N ≥ 1)	0.214	Probability of **at least one peak** in a week.
Var(N□) and E[N□]	var ≈ mean	Confirms that the process **follows a homogeneous Poisson law**.

Next 3 months prediction

To extend the analysis forward, the Poisson model was used to estimate the **expected number of peaks** during the next three months (≈ 12 weeks).

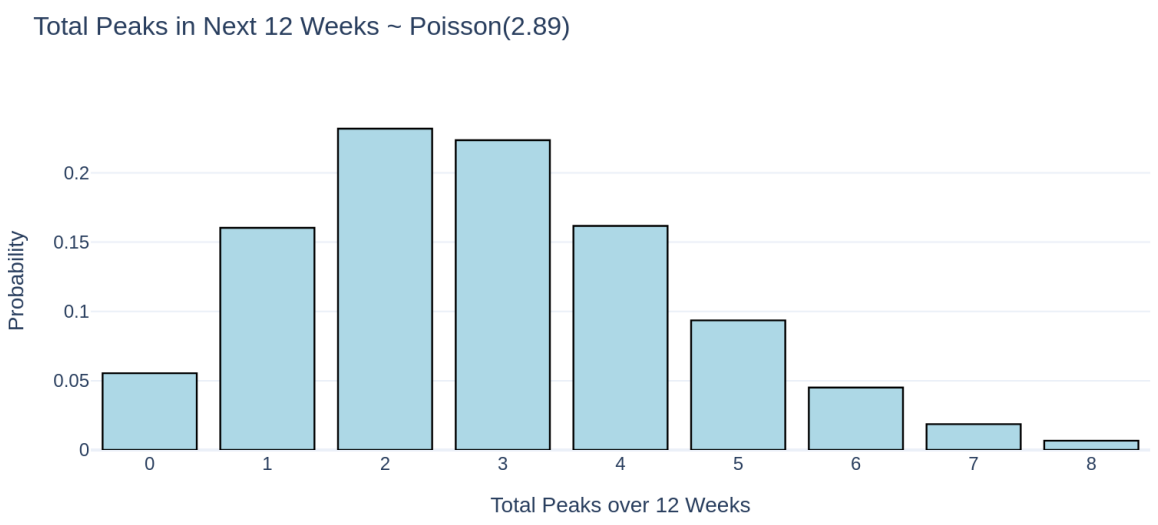
The weekly Poisson rate was previously estimated as
$\lambda = 0.241$ peaks/week.

Thus, the expected total rate for 12 weeks is

$\lambda_{\text{total}} = 12 \times 0.241 = 3.13.$

```
import plotly.graph_objects as go
from scipy.stats import poisson
import numpy as np

# --- Prediction ---
# lam previously obtained
H = 12       # 3 months range
lam_total = H * lam  # lambda for 3 months

# --- Poisson Distribution for total peaks in 12 weeks ---
k = np.arange(0, max(6, int(lam_total * 3) + 1))
pmf_total = poisson.pmf(k, mu=lam_total)

# --- Plotly graph ---
fig = go.Figure()

fig.add_trace(go.Bar(
    x=k,
    y=pmf_total,
    name="Poisson PMF",
    marker_color="lightblue",
    marker_line_color="black",
    marker_line_width=1.2,
    hovertemplate="Peaks=%{x}<br>Probability=%{y:.3f}<extra></extra>"
))

fig.update_layout(
    title=f"Total Peaks in Next 12 Weeks ~ Poisson({lam_total:.2f})",
    xaxis_title="Total Peaks over 12 Weeks",
    yaxis_title="Probability",
    template="plotly_white",
    width=850,
    height=400
)
```

```
fig.show()

# --- Expected result ---
print(f"\nExpected total peaks in 12 weeks: {lam_total:.2f}")
```




```
Expected total peaks in 12 weeks: 2.89
```

The histogram shows the probability of observing 0, 1, 2, 3, … total peaks in that period.

The most likely outcome is **around 3 peaks**, with most scenarios falling between **2 and 4**.

This confirms that, under current market behavior, Google's stock tends to experience only a few significant movements over a 3-month window.

Expected total peaks in 12 weeks: 2.89

Stochastic Index

The stochastic oscillator is a widely used technical indicator in financial analysis that measures the momentum of price movements. Developed by George Lane in the late 1950s, it compares a security's closing price to its price range over a given period, providing insights into potential overbought or oversold market conditions.

This indicator is based on the idea that prices tend to close near their highs during upward trends and near their lows during downward trends. It consists of two main components: the %K line, which represents the current position of the closing price within the recent trading range, and the %D line, which is a moving average of %K used to identify buy or sell signals.

By analyzing the relationship between these two lines, investors can detect possible trend reversals and make more informed trading decisions. In this project, we explore the calculation, visualization, and

interpretation of the stochastic index to better understand its role in technical analysis and decision-making in financial markets.

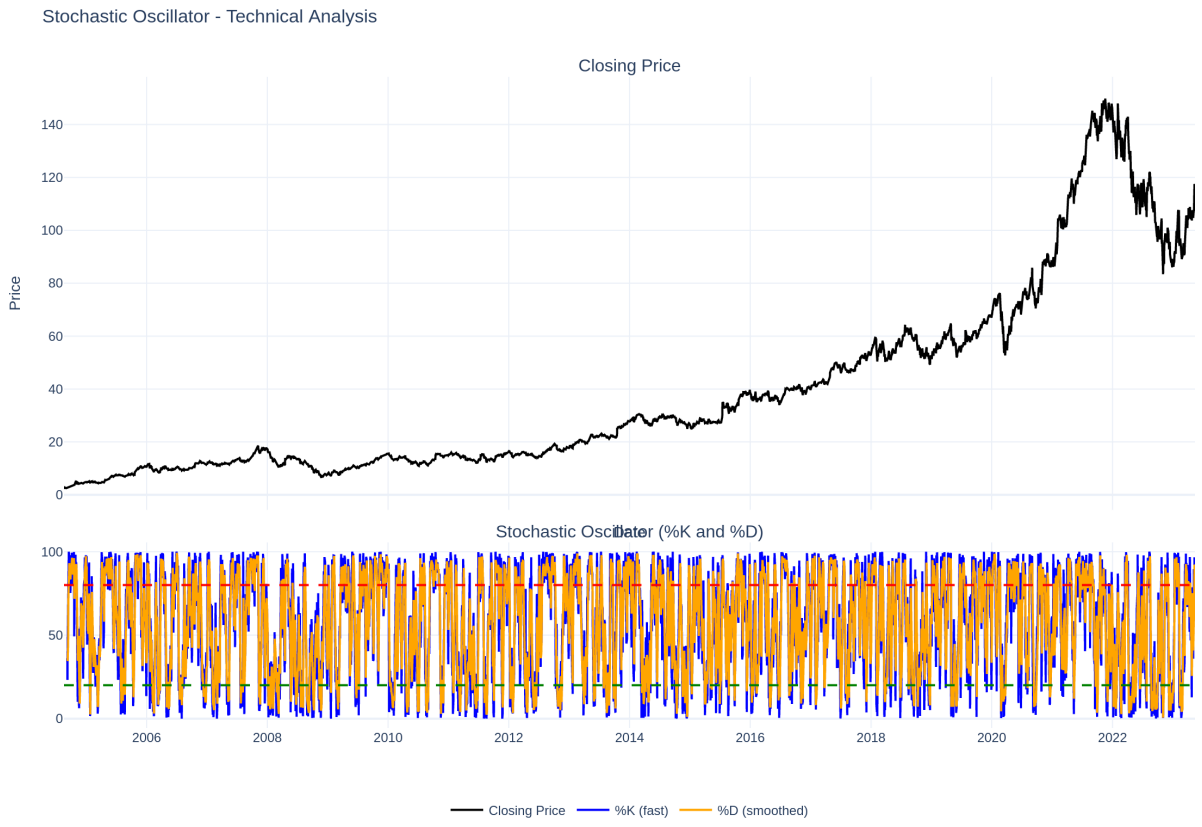
```
df = pd.read_csv('/content/GOOGLE.csv')
df.head()
```

	Date	Open	High	Low	Close	Volume
0	2004-08-19	2.502503	2.604104	2.401401	2.511011	893181924
1	2004-08-20	2.527778	2.729730	2.515015	2.710460	456686856
2	2004-08-23	2.771522	2.839840	2.728979	2.737738	365122512
3	2004-08-24	2.783784	2.792793	2.591842	2.624374	304946748
4	2004-08-25	2.626627	2.702703	2.599600	2.652653	183772044

Stochastic Index

```
# Convert the 'Date' column to datetime format and sort the DataFrame
df['Date'] = pd.to_datetime(df['Date'])
df = df.sort_values('Date')

# === 2. Calculate the Stochastic Oscillator ===
n = 14  # 14-day rolling window

# Lowest low over the past n periods
df['L_n'] = df['Low'].rolling(window=n).min()
# Highest high over the past n periods
df['H_n'] = df['High'].rolling(window=n).max()

# %K line: shows the current closing price's position within the recent high-low ra
df['%K'] = 100 * (df['Close'] - df['L_n']) / (df['H_n'] - df['L_n'])
# %D line: 3-period moving average of %K, used as a smoothed signal line
df['%D'] = df['%K'].rolling(window=3).mean()

# === 3. Create the figure with subplots ===
import plotly.graph_objects as go
from plotly.subplots import make_subplots

# Define two vertically stacked subplots (price and stochastic index)
fig = make_subplots(
    rows=2, cols=1,
    shared_xaxes=True,       # share the same X-axis (dates)
    vertical_spacing=0.05,   # space between plots
    row_heights=[0.7, 0.3],  # relative height of each subplot
    subplot_titles=('Closing Price', 'Stochastic Oscillator (%K and %D)')
)

# --- Plot 1: Closing Price ---
fig.add_trace(
    go.Scatter(x=df['Date'], y=df['Close'], mode='lines', name='Closing Price', lin
    row=1, col=1
)

# --- Plot 2: %K and %D lines ---
```

```
fig.add_trace(
    go.Scatter(x=df['Date'], y=df['%K'], mode='lines', name='%K (fast)', line=dict(
    row=2, col=1
)

fig.add_trace(
    go.Scatter(x=df['Date'], y=df['%D'], mode='lines', name='%D (smoothed)', line=d
    row=2, col=1
)

# --- Overbought / Oversold threshold lines ---
fig.add_hline(y=80, line_dash='dash', line_color='red', row=2, col=1)   # overbough
fig.add_hline(y=20, line_dash='dash', line_color='green', row=2, col=1) # oversold

# === 4. Layout and styling adjustments ===
fig.update_layout(
    title='Stochastic Oscillator - Technical Analysis',
    xaxis_title='Date',
    yaxis_title='Price',
    height=800,
    template='plotly_white',
    legend=dict(orientation='h', yanchor='bottom', y=-0.15, xanchor='center', x=0.5
)

# === 5. Display the plot ===
fig.show()
```



This code calculates and plots the Stochastic Oscillator, a momentum indicator that shows whether a stock may be overbought or oversold. Using a 14-day window, it computes the %K and %D lines from the high, low, and closing prices. The chart displays both the price trend and the

stochastic lines, with reference levels at 80 (overbought) and 20 (oversold), indicating potential selling and buying signals, respectively.

The Fast Stochastic (%K) measures the position of the current closing price within the range of the last n days (for example, 14).

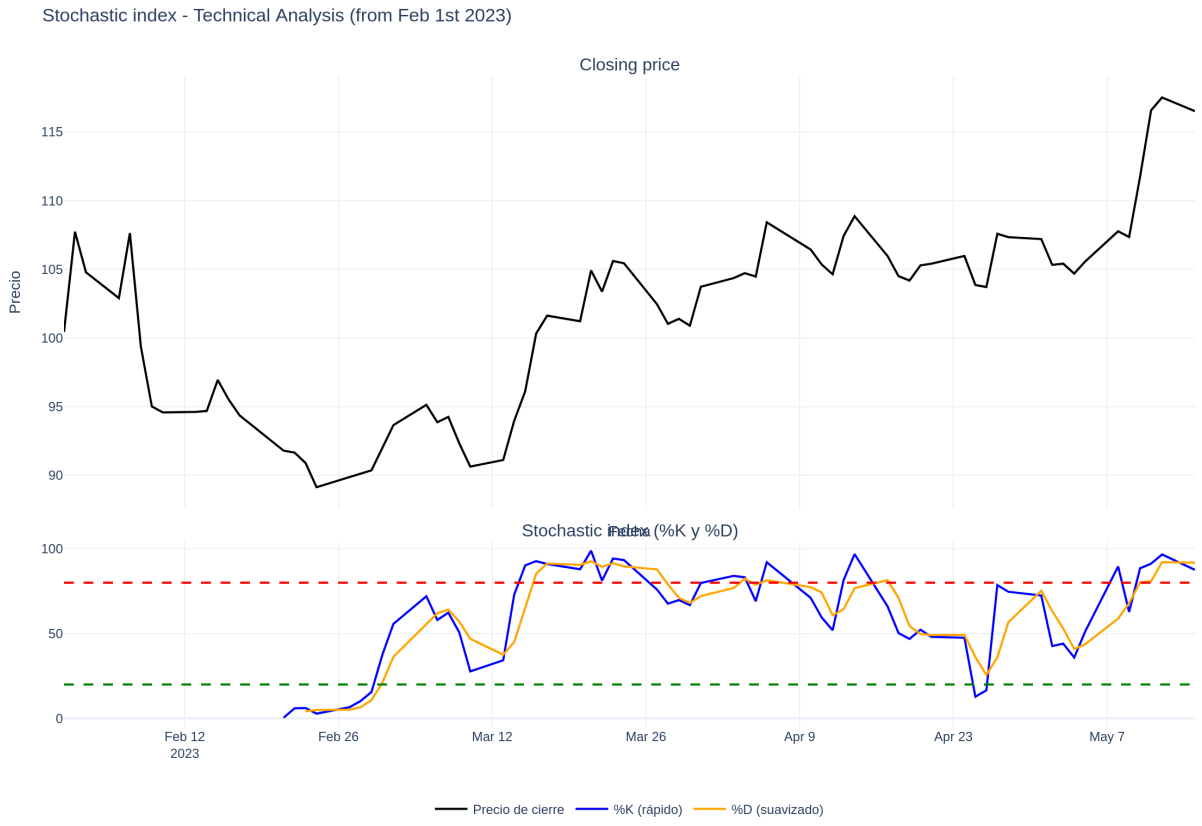
If %K = 100, the current price is at its highest point in the past 14 days. If %K = 0, the price is at its lowest point. If %K = 50, the price is in the middle of the range — neither high nor low.

High values (>80) indicate that the price is close to its recent maximum, suggesting a possible overbought condition. Low values (<20) indicate that the price is near its recent minimum, suggesting a possible oversold condition.

The %D line is a moving average of %K (usually over 3 periods). When %K crosses above %D, it may signal a buying opportunity, as the price may start to rise. When %K crosses below %D, it may signal a selling opportunity, as the price may start to fall.

```
import pandas as pd
import plotly.graph_objects as go
from plotly.subplots import make_subplots

# === 1. Convertir la columna Date a formato fecha y ordenar ===
df['Date'] = pd.to_datetime(df['Date'])
df = df.sort_values('Date')

# === 1.1 Filtrar desde el 1 de febrero de 2023 hasta la última fecha ===
fecha_inicio = pd.Timestamp('2023-02-01')
fecha_final = df['Date'].max()
df = df[(df['Date'] >= fecha_inicio) & (df['Date'] <= fecha_final)]

# === 2. Calcular el índice estocástico ===
n = 14  # ventana de 14 días

df['L_n'] = df['Low'].rolling(window=n).min()
df['H_n'] = df['High'].rolling(window=n).max()
df['%K'] = 100 * (df['Close'] - df['L_n']) / (df['H_n'] - df['L_n'])
df['%D'] = df['%K'].rolling(window=3).mean()

# === 3. Crear la figura con subplots ===
fig = make_subplots(
    rows=2, cols=1,
    shared_xaxes=True,
    vertical_spacing=0.05,
    row_heights=[0.7, 0.3],
    subplot_titles=('Closing price', 'Stochastic index (%K y %D)')
)

# --- Gráfico 1: Precio de cierre ---
fig.add_trace(
    go.Scatter(x=df['Date'], y=df['Close'], mode='lines', name='Precio de cierre',
    row=1, col=1
)
```

```
# --- Gráfico 2: %K y %D ---
fig.add_trace(
    go.Scatter(x=df['Date'], y=df['%K'], mode='lines', name='%K (rápido)', line=dic
    row=2, col=1
)

fig.add_trace(
    go.Scatter(x=df['Date'], y=df['%D'], mode='lines', name='%D (suavizado)', line=
    row=2, col=1
)

# --- Líneas de sobrecompra/sobreventa ---
fig.add_hline(y=80, line_dash='dash', line_color='red', row=2, col=1)
fig.add_hline(y=20, line_dash='dash', line_color='green', row=2, col=1)

# === 4. Ajustes de diseño ===
fig.update_layout(
    title='Stochastic index - Technical Analysis (from Feb 1st 2023)',
    xaxis_title='Fecha',
    yaxis_title='Precio',
    height=800,
    template='plotly_white',
    legend=dict(orientation='h', yanchor='bottom', y=-0.15, xanchor='center', x=0.5
)

# === 5. Mostrar la gráfica ===
fig.show()
```



This is the same stochastic index, but only taking data from February 1st to May 15th, 2023, in order to visualize it better.

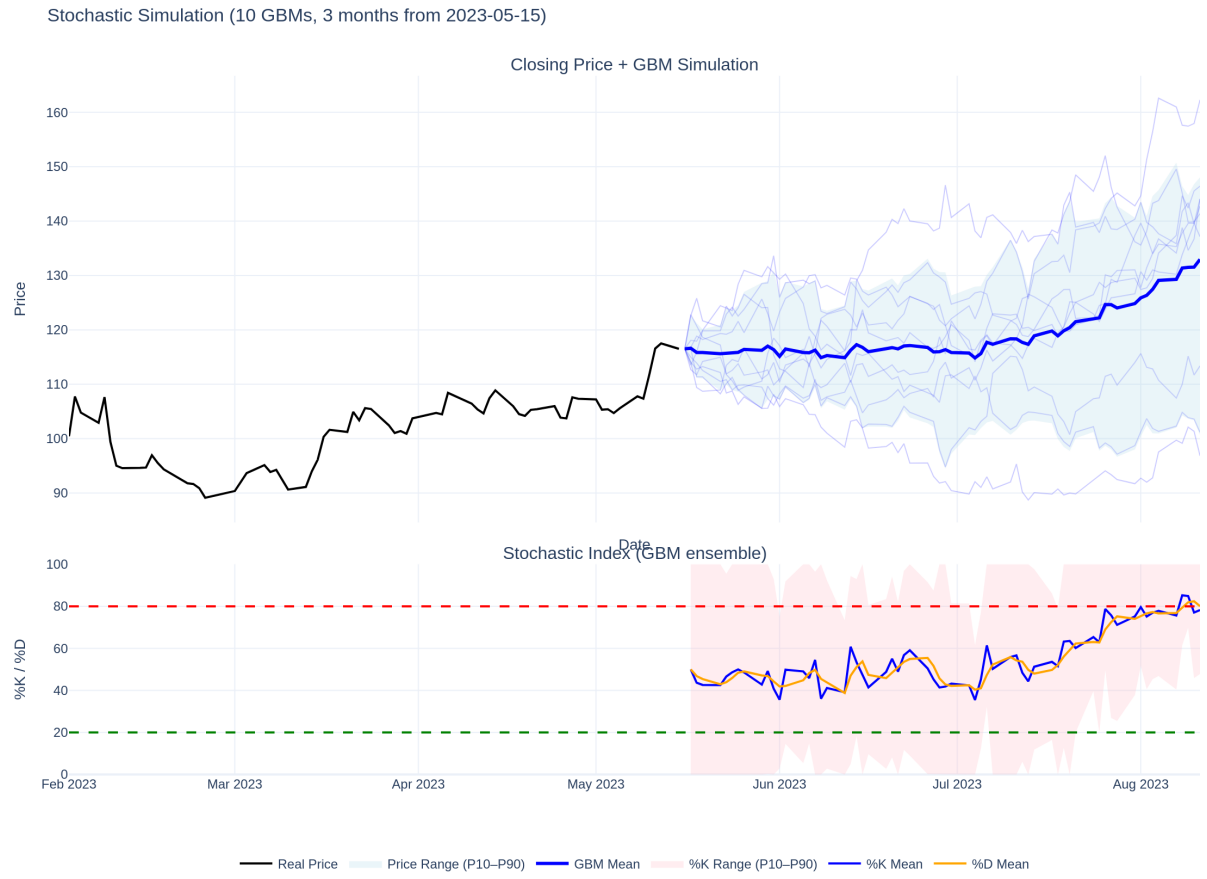
Geometric Brownian Motion (GBM)

```
import numpy as np
import pandas as pd
import plotly.graph_objects as go
from plotly.subplots import make_subplots

# --- 1. Prepare data ---
df['Date'] = pd.to_datetime(df['Date'])  # Ensure 'Date' column is in datetime form
df = df.sort_values('Date')              # Sort data by date ascending

# Filter data for the period of interest
fecha_inicio = pd.Timestamp('2023-02-01')  # Start date for visualization
fecha_final = df['Date'].max()             # End date (e.g., latest available)
df = df[(df['Date'] >= fecha_inicio) & (df['Date'] <= fecha_final)].copy()

# --- 2. Estimate mu and sigma from log-returns ---
df['log_ret'] = np.log(df['Close'] / df['Close'].shift(1))  # Log returns
log_rets = df['log_ret'].dropna()                            # Drop NaN from first
mu = log_rets.mean() * 252                                   # Annualized mean retu
sigma = log_rets.std() * np.sqrt(252)                        # Annualized volatilit

# --- 3. GBM parameters ---
S0 = df['Close'].iloc[-1]  # Last observed price
days = 63                  # ~3 months of trading days
n_paths = 10               # Number of simulated GBM paths
dt = 1/252                 # Time step in years (1 trading day)
seed = 42
np.random.seed(seed)       # For reproducibility

# --- 4. Simulate GBMs ---
# Generate random increments for GBM
increments = np.random.normal(
    loc=(mu - 0.5 * sigma**2) * dt,  # Drift term
    scale=sigma * np.sqrt(dt),        # Diffusion term
    size=(n_paths, days)
)
log_paths = np.cumsum(increments, axis=1)  # Cumulative sum to get log-price paths
log_paths = np.concatenate([np.zeros((n_paths, 1)), log_paths], axis=1)  # Include
S_paths = S0 * np.exp(log_paths)           # Convert log-prices to actual prices

# --- 5. Price statistics ---
S_mean = S_paths.mean(axis=0)              # Mean price across simulations
S_p10 = np.percentile(S_paths, 10, axis=0) # 10th percentile
S_p90 = np.percentile(S_paths, 90, axis=0) # 90th percentile

# --- 6. Calculate %K for each path (stochastic oscillator) ---
window_n = 14                              # Lookback period for stochastic oscilla
T = S_paths.shape[1]                        # Number of time points
K_matrix = np.full((n_paths, T), np.nan)   # Initialize matrix to store %K

for i in range(n_paths):
    s = pd.Series(S_paths[i, :])
    L_n = s.rolling(window=window_n, min_periods=1).min()  # Rolling lowest low
    H_n = s.rolling(window=window_n, min_periods=1).max()  # Rolling highest high
    denom = (H_n - L_n).replace(0, np.nan)                 # Avoid division by zero
    K = 100 * (s - L_n) / denom                             # %K formula
    K_matrix[i, :] = K.values
```

```python
# Compute mean and percentiles of %K across all GBM paths
K_mean = np.nanmean(K_matrix, axis=0)
K_p10 = np.nanpercentile(K_matrix, 10, axis=0)
K_p90 = np.nanpercentile(K_matrix, 90, axis=0)
D_mean = pd.Series(K_mean).rolling(window=3, min_periods=1).mean()  # %D: 3-period

# --- 7. Future dates for plotting ---
start_date = df['Date'].max()
future_dates = pd.bdate_range(start=start_date + pd.Timedelta(days=1), periods=days
# Adjust in case of mismatch
if len(future_dates) != T:
    future_dates = pd.bdate_range(start=start_date + pd.Timedelta(days=1), periods=

# Combine results into a DataFrame
df_future = pd.DataFrame({
    'Date': future_dates,
    'S_mean': S_mean,
    'S_p10': S_p10,
    'S_p90': S_p90,
    '%K_mean': K_mean,
    '%K_p10': K_p10,
    '%K_p90': K_p90,
    '%D_mean': D_mean
})

# --- 8. Plot using Plotly: price + bands and stochastic oscillator ---
fig = make_subplots(
    rows=2, cols=1,
    shared_xaxes=True,
    vertical_spacing=0.06,
    row_heights=[0.68, 0.32],
    subplot_titles=('Closing Price + GBM Simulation', 'Stochastic Index (GBM ensemb
)

# Historical real price
fig.add_trace(
    go.Scatter(x=df['Date'], y=df['Close'], mode='lines', name='Real Price', line=d
    row=1, col=1
)

# Plot individual GBM paths (light blue)
for i in range(10):  # Show 10 random paths
    fig.add_trace(
        go.Scatter(
            x=df_future['Date'],
            y=S_paths[i, :],
            mode='lines',
            line=dict(width=1, color='rgba(0,0,255,0.2)'),
            showlegend=False
        )
    )

# Price band (P10-P90) for GBM simulation
fig.add_trace(
    go.Scatter(
        x=pd.concat([df_future['Date'], df_future['Date'][::-1]]),
        y=pd.concat([df_future['S_p90'], df_future['S_p10'][::-1]]),
        fill='toself',
        fillcolor='rgba(173,216,230,0.25)',
        line=dict(color='rgba(255,255,255,0)'),
```

```
        showlegend=True,
        name='Price Range (P10–P90)'
    ),
    row=1, col=1
)

# GBM mean path
fig.add_trace(
    go.Scatter(x=df_future['Date'], y=df_future['S_mean'], mode='lines', name='GBM
    row=1, col=1
)

# --- Stochastic oscillator: %K band and averages ---
# %K band (P10-P90)
fig.add_trace(
    go.Scatter(
        x=pd.concat([df_future['Date'], df_future['Date'][::-1]]),
        y=pd.concat([df_future['%K_p90'], df_future['%K_p10'][::-1]]),
        fill='toself',
        fillcolor='rgba(255,182,193,0.25)',
        line=dict(color='rgba(255,255,255,0)'),
        showlegend=True,
        name='%K Range (P10–P90)'
    ),
    row=2, col=1
)

# %K mean
fig.add_trace(
    go.Scatter(x=df_future['Date'], y=df_future['%K_mean'], mode='lines', name='%K
    row=2, col=1
)

# %D mean (3-day SMA of %K)
fig.add_trace(
    go.Scatter(x=df_future['Date'], y=df_future['%D_mean'], mode='lines', name='%D
    row=2, col=1
)

# Overbought / oversold lines
fig.add_hline(y=80, line_dash='dash', line_color='red', row=2, col=1)
fig.add_hline(y=20, line_dash='dash', line_color='green', row=2, col=1)

# Layout settings
fig.update_layout(
    title=f'Stochastic Simulation (10 GBMs, 3 months from {fecha_final.date()})',
    xaxis_title='Date',
    yaxis_title='Price',
    height=850,
    template='plotly_white',
    legend=dict(orientation='h', yanchor='bottom', y=-0.15, xanchor='center', x=0.5
)

# %K / %D y-axis range
fig.update_yaxes(range=[0, 100], row=2, col=1, title='%K / %D')

fig.show()
```

```
/tmp/ipython-input-1819524520.py:59: RuntimeWarning:

Mean of empty slice

/usr/local/lib/python3.12/dist-packages/numpy/lib/_nanfunctions_impl.py:1634: Runt
imeWarning:

All-NaN slice encountered
```



The figure presents a stochastic simulation of the stock price over approximately three months, starting from May 15, 2023, using a Geometric Brownian Motion (GBM) ensemble approach. The simulation consists of 10 GBM paths, which model potential future price trajectories based on the historical volatility and mean return calculated from log-returns.

Feedback Corrections

Additional explanation - Drift, Volatility, and Market Stability

In the Geometric Brownian Motion (GBM) model, two parameters define the expected stock behavior: drift (μ) and volatility (σ). The drift represents the average rate of return—the tendency of the stock price to grow over time under normal market conditions—while volatility measures the degree of random fluctuation or uncertainty around that trend. A higher σ means larger and more unpredictable price swings, while a lower σ indicates steadier movement and reduced short-term risk.

For Google’s stock, the calculated volatility (σ) was relatively low compared to other technology companies such as Tesla or Nvidia. This can be justified by Google’s diversified business model—which includes advertising, cloud services, and AI research—and its strong market capitalization within the Technology sector of the U.S. stock market (NASDAQ). These factors make Google less sensitive to speculative trading and rapid valuation changes, which typically drive volatility in high-growth firms.

While the model focuses on a single stock, “market stability” in this context refers to the consistency of Google’s own price evolution over time. Its limited frequency of extreme events (confirmed by the Poisson analysis, λ ≈ 0.24) and narrow simulation bands under GBM both reflect an asset that moves predictably even during broader market uncertainty. In short, Google’s stability does not contradict the high volatility seen in other tech stocks—it highlights how company maturity, investor confidence, and diversification can produce lower risk dynamics within the same market sector.

Additional Explanation – The Wiener Process in GBM

In the **Geometric Brownian Motion (GBM)** model, the **Wiener process** W_t — also known as *Brownian motion* — represents the random component that introduces uncertainty into the stock price evolution. It is a continuous-time stochastic process with the following key properties:

1. $W_0 = 0$ — the process starts at zero.
2. **Independent increments:** changes in W_t over disjoint time intervals are statistically independent.
3. **Normally distributed increments:**

$$W_t - W_s \sim \mathcal{N}(0,\, t-s), \quad \text{for } 0 \le s < t.$$

   Each increment follows a normal distribution with mean 0 and variance equal to the elapsed time.

4. **Continuous paths:** W_t is continuous in time but nowhere differentiable, capturing the natural irregularity of financial markets.

The GBM model is expressed as:

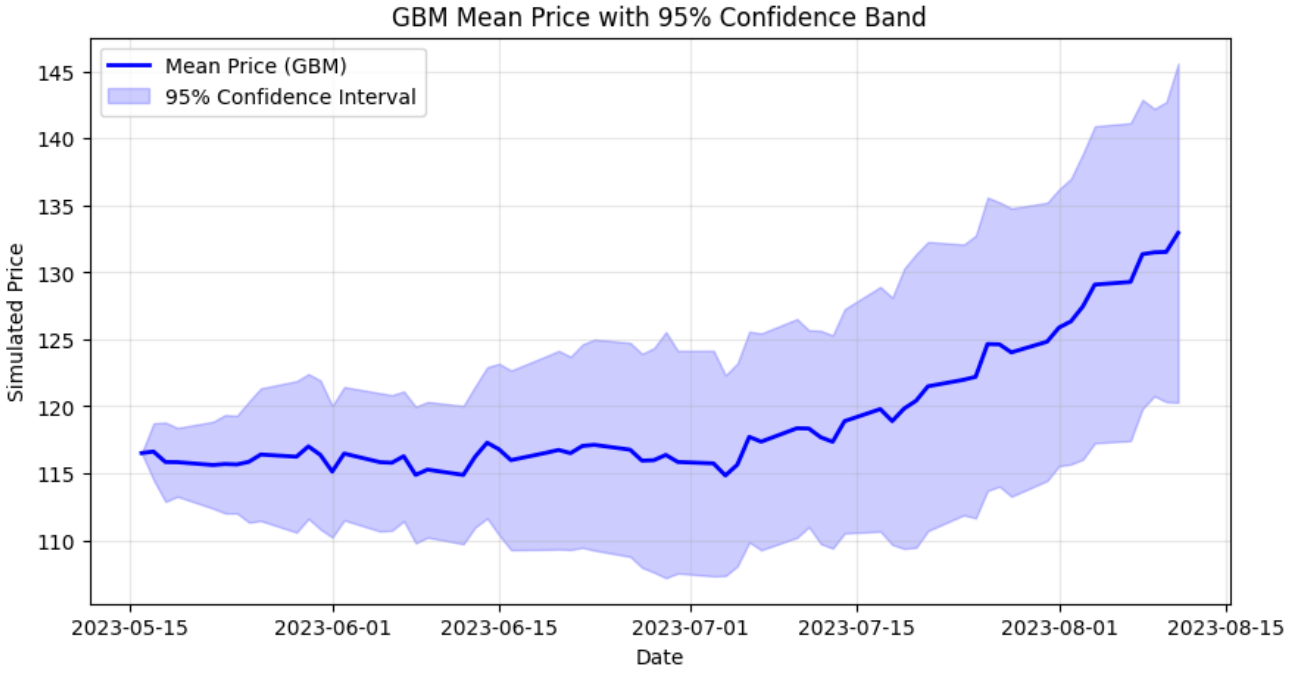
$$S_t = S_0 \, e^{(\mu - \frac{1}{2}\sigma^2)t + \sigma W_t}$$

Where:

- μ (**drift**) determines the *average rate of growth* of the stock price.
- σ (**volatility**) scales the *intensity of random fluctuations* generated by the Wiener process.
- W_t introduces randomness, creating a probabilistic range of future outcomes around the mean path.

This randomness allows GBM to realistically simulate the uncertain nature of financial markets, where short-term variations are driven by unpredictable shocks, while the long-term drift reflects consistent expected growth.

```
import numpy as np
import matplotlib.pyplot as plt

# Assuming you already have your simulated paths in S_paths (shape: [n_paths, T])
# and your time vector or future dates in df_future['Date']

# Compute the mean and standard deviation at each time step
S_mean = S_paths.mean(axis=0)
S_std = S_paths.std(axis=0)

# 95% confidence interval for the mean (approx. normal assumption)
conf_int_95 = 1.96 * (S_std / np.sqrt(S_paths.shape[0]))
upper_ci = S_mean + conf_int_95
lower_ci = S_mean - conf_int_95

# Plot mean and confidence band
plt.figure(figsize=(10,5))
plt.plot(df_future['Date'], S_mean, label='Mean Price (GBM)', color='blue', linewid
plt.fill_between(df_future['Date'], lower_ci, upper_ci, color='blue', alpha=0.2, la
plt.title('GBM Mean Price with 95% Confidence Band')
plt.xlabel('Date')
plt.ylabel('Simulated Price')
plt.legend()
plt.grid(True, alpha=0.3)
plt.show()
```



Conclusions

Poisson Conclusion

Interpretation - Practical Meaning of the Poisson Model

Google's stock behaves as a low-volatility asset with rare, unpredictable shocks. It's better suited for long-term investment stability than short-term speculation.

The estimated rate $\lambda = 0.241$ indicates that large price changes happen about **once every four weeks**, showing that Google's stock is **relatively stable** and experiences **low volatility** under normal conditions.

Stability is a positive signal for **long-term investors**, as it suggests few abrupt or unpredictable movements.

However, the Poisson model does **not** predict whether Google's stock price will rise or fall, but it predicts how **often** significant movements (peaks) occur over time. It only describes the **frequency** of extreme events, not their direction or magnitude.

Therefore, while Google's stock appears to be a **low-risk, long-term asset**, short-term trading decisions would require additional models that incorporate **trend direction** and **market context**.

The interpretation of the log-return peaks observed in **2008** and **2020** strongly aligns with real-world market crises. The spike in 2008 corresponds to the **global financial crisis**, while the increase in 2020 matches the **COVID-19 pandemic shock**. These events validate the Poisson model's ability to capture **rare but impactful disruptions** in market behavior, reinforcing the reliability of the estimated rate $\lambda = 0.241$ as an accurate reflection of Google’s typical volatility pattern.

Stochastic Index Conclusion

Using the GBM ensemble method combined with the stochastic oscillator allows both probabilistic forecasting of prices and an assessment of potential momentum extremes. The simulations indicate a mild upward trend in price, with a growing probability of reaching overbought levels as reflected in the %K/%D behavior, while acknowledging significant uncertainty in individual paths.

General Conclusion

This project successfully applied stochastic optimization techniques to analyze financial risk and price volatility in Google’s stock. By combining discrete-event modeling through the Poisson process and continuous-time simulation via Geometric Brownian Motion (GBM), it was possible to capture both the frequency of extreme price movements and the long-term dynamics of Google's market behavior.

The results showed that Google’s stock exhibits low volatility and infrequent shocks, with extreme fluctuations occurring approximately once every four weeks ($\lambda \approx 0.24$). This stability suggests that the asset is well suited for long-term investment strategies, providing steady growth with limited exposure to sudden risks.

Through GBM simulations, the model also revealed realistic uncertainty bands for short-term price evolution, highlighting the role of randomness and historical volatility in shaping potential future outcomes. These findings reinforce the usefulness of stochastic models for quantitative risk management, portfolio optimization, and data-driven financial decision-making.

Overall, this project demonstrates that integrating probabilistic and stochastic modeling provides a powerful framework for understanding financial uncertainty, supporting more informed and resilient investment strategies in dynamic markets.

Individual Conclusions

Abigail: During this project, the use of ChatGPT was mainly for validating the models and identifying ways to improve them. AI tools also helped me clarify statistical concepts, and improve the structure of written explanations. However, every suggestion was verified, adapted, and rewritten to ensure I fully understood its logic and that all results were consistent with my dataset and objectives. I applied responsible AI literacy by using these tools as a guide, not as a source to copy from, ensuring that the final work reflected my own understanding and reasoning. Following Article IX of the General Student Regulations, I maintained academic integrity, avoiding plagiarism and ensuring transparency in my use of AI. This experience reinforced the importance of using AI critically and consciously, as a tool that supports learning and promotes ethical and original work in data science.

Ana Pau: This project allowed me to apply theoretical and computational concepts to analyze stochastic processes in a practical and data-driven context. Through the use of models such as the Poisson process and Geometric Brownian Motion, I was able to understand how randomness influences financial and real-world systems. The integration of AI tools supported the learning process by mainly clarifying complex concepts but also giving advice to improve the efficiency of model development, while maintaining full academic integrity and critical thinking.

Overall, this experience strengthened my technical skills and my ability to interpret probabilistic results, but also highlighted the importance of ethical and responsible use of AI in data analysis. Beyond the numerical results, the main takeaway was learning to combine analytical reasoning with technological support to produce transparent, rigorous, and meaningful work in data science.

Mariana: Personally, I used Gemini to understand how the Geometric Brownian Motion (GBM) model applies to stock price analysis. It served as a guide to clarify equations, parameters, and assumptions behind stochastic simulations. However, I made sure to recreate and verify the

logic myself, checking that the explanations matched the mathematical and financial context of our data. Following the principles of responsible AI literacy, I used the tool for comprehension rather than automation, ensuring that every part of my work reflected my own understanding. This process made me aware that AI should not replace critical thinking, but rather support analytical growth and ethical learning within data science practice.

Alex: I used ChatGPT mainly to gather references and context about the stochastic index and how it relates to financial modeling. The AI helped me explore academic explanations and compare definitions, but I made sure to understand and reformulate everything in my own way before applying it. Rather than relying on the tool to solve tasks, I used it to broaden my comprehension and verify the consistency of my interpretations. I followed the principles of academic integrity from Article IX by avoiding direct copying and clearly recognizing AI as a learning assistant. This experience showed me that generative AI can be a valuable ally when used critically, helping transform complex ideas into meaningful knowledge while maintaining originality and ethical responsibility.

Aarón: During this project, I used Gemini and ChatGPT in Google Colab mainly to clarify doubts, improve code efficiency, and review explanations about the Poisson process and GBM. These tools helped me learn faster, but I always verified, modified, and tested their suggestions before including them in the notebook.

I made sure to apply responsible AI literacy, understanding every piece of code instead of copying it directly. All results and interpretations were written by me and my team, following Article IX of the General Student Regulations, ensuring originality and transparency.

I believe that AI should support, not replace, human reasoning. Used consciously and ethically, it becomes a powerful resource for learning and developing rigorous, responsible work as future data scientists.

References

- Koivu, M. (2004). **A stochastic optimization approach to financial decision making** (Doctoral dissertation, Helsinki School of Economics).

Acta Universitatis Oeconomicae Helsingiensis A-234. Retrieved from https://aaltodoc.aalto.fi/server/api/core/bitstreams/e717afad-de92-4203-8ee8-bbfea5c4f0e0/content

- Gallager, R. (2011, Spring). **Chapter 2: Poisson processes**. In *Discrete Stochastic Processes (Course 6.262)*. Massachusetts Institute of Technology. Retrived from https://ocw.mit.edu/courses/6-262-discrete-stochastic-processes-spring-2011/3a19ce0e02d0008877351bfa24f3716a_MIT6_262S11_chap02.pdf

- Daniel, J. W. (2008, June 26). **Poisson processes (and mixture distributions)**. Austin Actuarial Seminars. Retrived from https://www.casact.org/sites/default/files/database/studynotes_3_poisson_2 (casact.org)

- Bilal Waseer. (2025). **Google Stocks Complete** [Dataset]. Kaggle. Recuperado de https://www.kaggle.com/datasets/bilalwaseer/google-stocks-complete