Módulo 3 / Semana 11

Simulación de Monte Carlo en finanzas

Introducción

La empresa administradora de fondos de inversión La Mezcla Inteligente para ahorradores e inversionistas, ofrece una amplia variedad de alternativas de inversión para que sus clientes tengan opciones de la conformación de su patrimonio y obtener atractivos rendimientos. De todos es sabido el comportamiento histórico de los precios de cada uno de los fondos, pero también todos se preguntan cuál será el futuro comportamiento del patrimonio, por consiguiente mayor satisfacción de los clientes, y por supuesto, el reconocimiento de su clientela.

El director de estrategia de La Mezcla Inteligente asistió a una conferencia y conoció la técnica estadística de simulación de patrones de comportamiento de precios de activos financieros, la cual le interesó desde el primer momento porque sabe lo valioso que será para su empresa y para sus clientes. Por lo tanto, quiere aplicarla cuanto antes, y con esto, incluirla como parte de sus estrategias para la selección de activos en cada fondo de inversión.

Explicación

La simulación en términos estadísticos y financieros es como tener la experiencia de explorar en la información y su análisis, experiencia en construir portafolios, experiencia como *trader*, incluso experiencia en perder dinero de manera virtual. En la simulación no se requieren millones de pesos para adquirir considerable experiencia en valuación de activos financieros y probar la oferta y demanda del mercado.

En este tema se dará a conocer una técnica particular de la simulación denominada *Monte Carlo*. El término Monte Carlo es típicamente asociado con el proceso de modelar y simular un sistema afectado por aleatoriedad. Se generan escenarios aleatorios y herramientas estadísticas para establecer, por ejemplo, el valor de un activo financiero.

Simulación dinámica del precio de acciones mediante movimiento browniano geométrico

Se toma el caso como inversionista que desea modelar el comportamiento del valor de un portafolio compuesto de n activos financieros como acciones, bonos, metales, etc. Al tiempo inicial (t=0), se conoce el valor del activo i como P_i^0, y la cantidad de títulos de cada activo en el portafolio es h_1, por lo tanto, el valor inicial del portafolio (VP) es la suma del producto de la cantidad de títulos multiplicada por el precio:

$$VP_0 = \sum_{i=1}^{n} h_i * P_i^0$$

El precio de cada activo al final un periodo de planeación depende de un conjunto de factores dinámicos de riesgo, como las tasas de interés, inflación, precios del petróleo, variables macroeconómicas, etc. Para hacer más práctico el ejemplo, supón que el único factor considerado es el factor mercado, el cual se representa como una variable aleatoria β con un comportamiento f(β), por ejemplo, como distribución normal. Por lo tanto, el precio de cada activo al tiempo T es determinado como una función de β:

$$P_i^T * f(\beta)$$

Así que, el valor del portafolio completo VF en el tiempo T es representado de la siguiente forma:

$$VF_T = \sum_{i=1}^{n} h_i * P_i^T * f(\beta)$$

Por medio de la simulación Monte Carlo se hacen interacciones para generar valores aleatorios de β por medio de la distribución normal para pronosticar el valor esperado del patrimonio en el tiempo T.

Una aplicación de Monte Carlo es cuando se requiere generar posibles rutas o trayectorias futuras para una variable, por ejemplo, del precio de una acción. El comportamiento de los precios de una acción en el tiempo es análogo a un movimiento secuencial aleatorio, en donde el valor actual está íntimamente relacionado con su valor anterior. Este fenómeno de movimiento aleatorio es denominado como *movimiento browniano*, en honor al escoses Robert Brown (Whiteside, 2008), quien realizó experimentos en los movimientos del polen en el agua. Posteriormente, Albert Einstein explicó con detalle cómo las micropartículas se mueven por moléculas de agua individuales. Un ejemplo de la ruta del polen en el agua se observa en la siguiente figura:

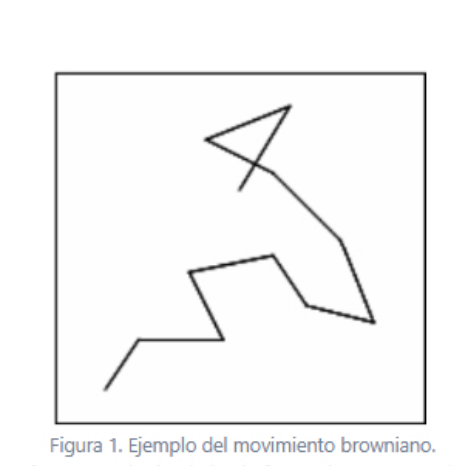


Figura 1. Ejemplo del movimiento browniano.
Fuente: Whiteside, J. (2008). *A practical application of Monte Carlo simulation in forecasting*. Recuperado de https://www.proquest.com/docview/208193177

A pesar de que la ruta es irregular, el movimiento browniano puede ser calculado mediante una ecuación, y se expondrá mediante un caso del valor del dinero en el tiempo. Se inicia con un valor actual P con una tasa de interés r capitalizable continuamente. Se quiere modelar el valor de la inversión en determinado tiempo t, para lo cual se utiliza la siguiente ecuación de valor de dinero en el tiempo:

$$F(t) = P * e^{rt}$$

Suponiendo que r es constante 6% anual y P son $100, se visualiza el siguiente comportamiento del valor futuro para cada año.

p	100	
t	f	t
0	100.00	0.06
1	106.18	0.06
2	112.75	0.06
3	119.72	0.06
4	127.12	0.06
5	134.99	0.06
6	143.33	0.06
7	152.20	0.06
8	161.61	0.06
9	171.60	0.06
10	182.21	0.06

Tabla 1. Valores futuros con tasa de interés fija con capitalización continua.

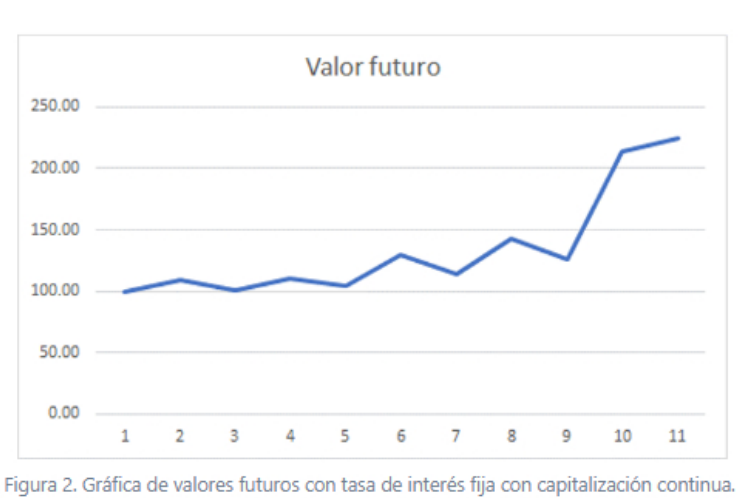


Figura 2. Gráfica de valores futuros con tasa de interés fija con capitalización continua.
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Ahora supón que la tasa de interés no es constante en el tiempo, y se incluye en el modelo como una función f(r) con un comportamiento similar a una determinada distribución de probabilidad. Ahora el exponente rt se convierte en una función del tiempo y, por lo tanto, para determinar su comportamiento se debe entrar en el campo del cálculo diferencial.

A continuación, se generarán valores aleatorios entre 0 y 1 a la tasa de interés y se obtendrá el siguiente comportamiento:

	100	
t	f	t
0	100.00	0.0880
1	109.33	0.0892
2	100.68	0.0034
3	110.27	0.0326
4	103.85	0.0094
5	129.31	0.0514
6	114.23	0.0222
7	143.01	0.0511
8	126.06	0.0289
9	214.00	0.0845
10	224.93	0.0911

Tabla 2. Valores futuros con tasa de interés aleatoria con capitalización continua.

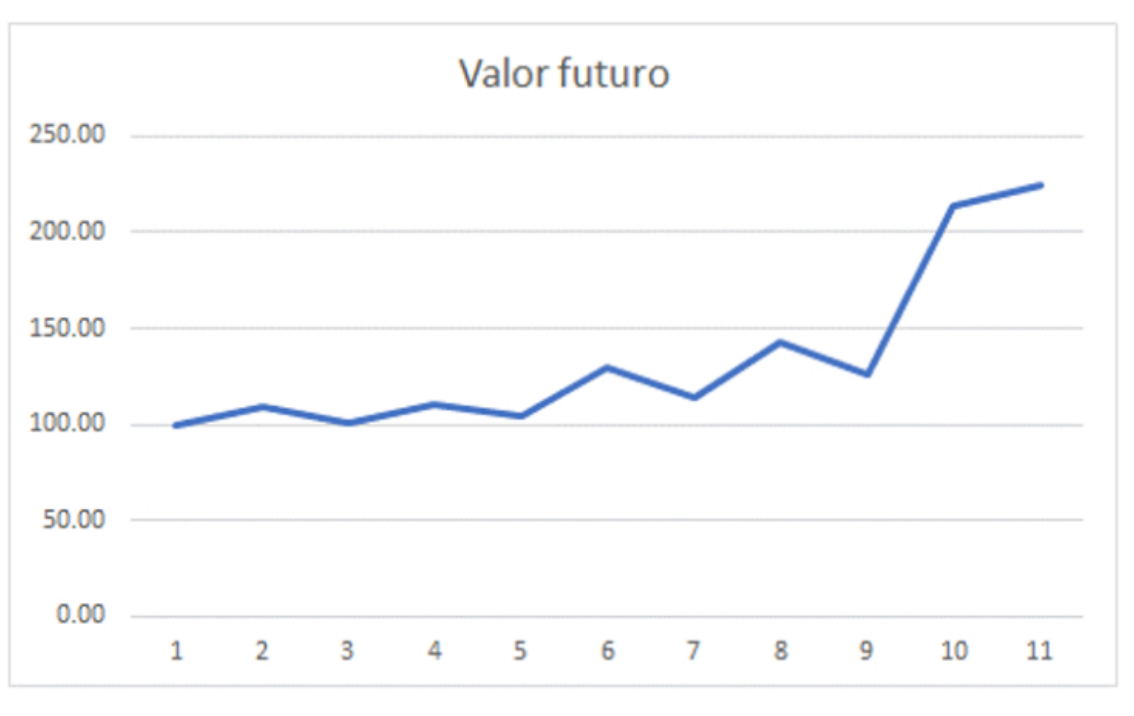


Figura 3. Gráfica de valores futuros con tasa de interés aleatoria con capitalización continua.
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

El método Monte Carlo entra al juego cuando se introduce aleatoriedad en una ecuación diferencial, generando lo que se denomina la ecuación estocástica diferencial (Lewinson, 2020). Un proceso estocástico es cuando el estado actual del sistema depende del estado anterior y, por lo tanto, se utiliza para entender futuros resultados que dependen sobre distribuciones de probabilidad. Ahora se va a estimar el precio de una acción mediante la ecuación estocástica diferencial.

Un proceso estocástico se dice que sigue un movimiento browniano geométrico (GBM, *geometric brownian motion*) cuando satisface la siguiente ecuación estocástica diferencial (Lewinson, 2020):

$$dS = \mu S dt + \sigma S dW_t$$

En donde *d* significa la derivada de la variable, *S* es el precio de la acción, *μ* es el retorno promedio sobre un periodo de tiempo como el retorno esperado inicial, σ es la volatilidad del precio o desviación estándar y W_t es el movimiento browniano. Los incrementos brownianos son calculados como un resultado de la variable aleatoria de la distribución normal N(0,t), donde *t* es el incremento de tiempo.

Utilizando el cálculo diferencial, se puede determinar el precio en el tiempo t:

$$S(t) = S_0 e^{\left(\mu - \frac{1}{2}\sigma^2\right)t + \sigma W_t}$$

En donde *S(t)* es el próximo valor del activo financiero y S_0 es el valor o precio inicial del activo financiero. Se observa que es similar a la ecuación que se utilizó anteriormente para determinar el valor futuro de una inversión con tasa de interés capitalizable continuamente.

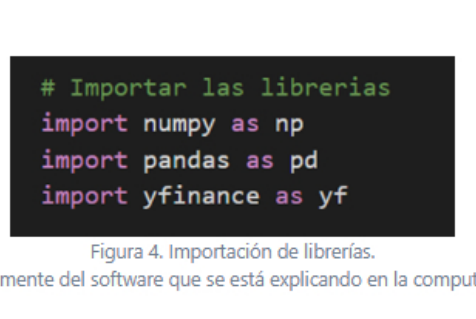
Para desarrollar el proceso de las simulaciones, se utilizará la siguiente fórmula recursiva:

$$S_{t+1} = S_t \exp\left(\mu - \frac{1}{2}\sigma^2\right)\Delta t + \sigma\sqrt{\Delta t}Z_{t+1}$$

En donde Z es una variable aleatoria con distribución normal y *Δt* es el diferencial de tiempo.

A continuación, se usará Monte Carlo y el movimiento geométrico browniano para simular los precios de la acción de NVIDIA Corporation, empresa fabricante de semiconductores que cotiza en la bolsa de valores Nasdaq de Estados Unidos. El código Python está tomado como referencia en la publicación de Lewinson (2020).

Se importan las librerías requeridas:



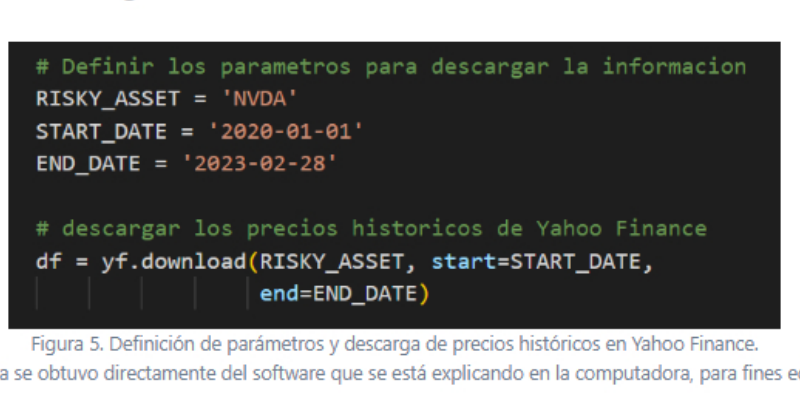
```
# Importar las librerias
import numpy as np
import pandas as pd
import yfinance as yf
```

Figura 4. Importación de librerías.
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Enseguida se incluye el texto para fines prácticos:

# Importar las librerias
import numpy as np
import pandas as pd
import yfinance as yf

Se identifica el símbolo de la acción de NVIDIA (NVDA), se establece el periodo de los precios históricos del 1 de junio del 2020 al 28 de febrero del 2023, y se descarga de Yahoo Finance.



```
# Definir los parametros para descargar la informacion
RISKY_ASSET = 'NVDA'
START_DATE = '2020-01-01'
END_DATE = '2023-02-28'

# descargar los precios historicos de Yahoo Finance
df = yf.download(RISKY_ASSET, start=START_DATE,
                 end=END_DATE)
```

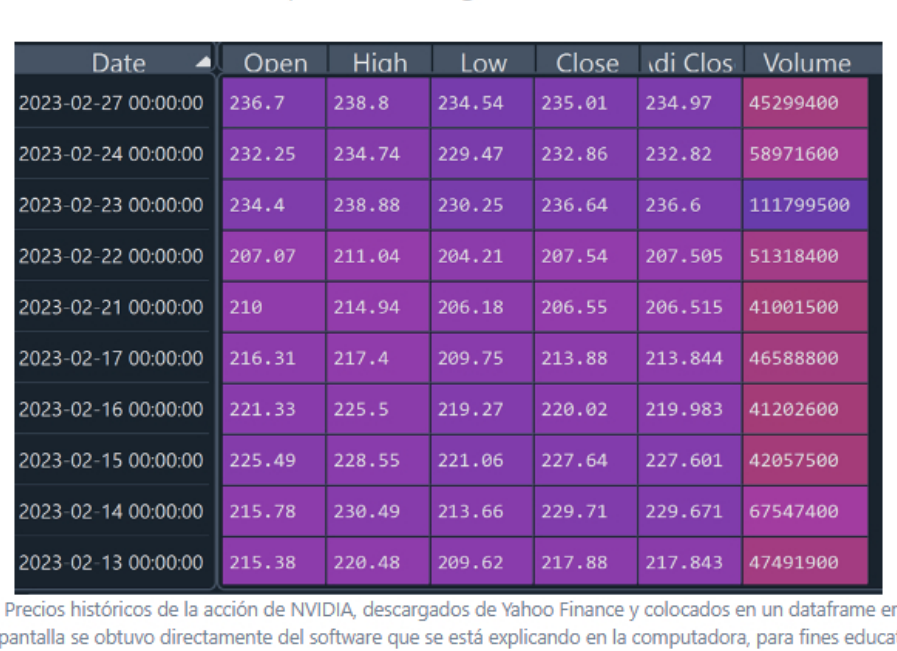
Figura 5. Definición de parámetros y descarga de precios históricos en Yahoo Finance.
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Enseguida se incluye el texto para fines prácticos:

# Definir los parametros para descargar la informacion
RISKY_ASSET = 'NVDA'
START_DATE = '2020-01-01'
END_DATE = '2023-02-28'

# descargar los precios historicos de Yahoo Finance
df = yf.download(RISKY_ASSET, start=START_DATE,
end=END_DATE)

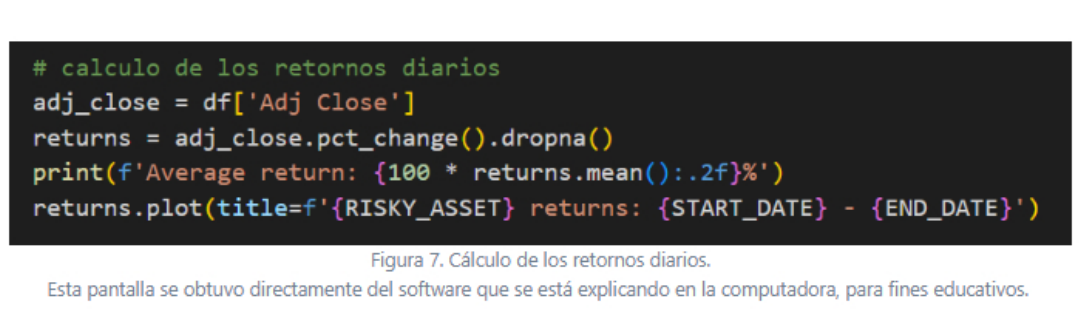
El dataframe df parcial con la información de la empresa es el siguiente:



Date	Open	High	Low	Close	\di Clos	Volume
2023-02-27 00:00:00	236.7	238.8	234.54	235.01	234.97	45299400
2023-02-24 00:00:00	232.25	234.74	229.47	232.86	232.82	58971600
2023-02-23 00:00:00	234.4	238.88	230.25	236.64	236.6	111799500
2023-02-22 00:00:00	207.07	211.04	204.21	207.54	207.505	51318400
2023-02-21 00:00:00	210	214.94	206.18	206.55	206.515	41001500
2023-02-17 00:00:00	216.31	217.4	209.75	213.88	213.844	46588800
2023-02-16 00:00:00	221.33	225.5	219.27	220.02	219.983	41202600
2023-02-15 00:00:00	225.49	228.55	221.06	227.64	227.601	42057500
2023-02-14 00:00:00	215.78	230.49	213.66	229.71	229.671	67547400
2023-02-13 00:00:00	215.38	220.48	209.62	217.88	217.843	47491900

Figura 6. Precios históricos de la acción de NVIDIA, descargados de Yahoo Finance y colocados en un dataframe en Python.
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Se calculan los retornos diarios, se asignan a la variable returns y genera la gráfica de retornos diarios.



```
# calculo de los retornos diarios
adj_close = df['Adj Close']
returns = adj_close.pct_change().dropna()
print(f'Average return: {100 * returns.mean():.2f}%')
returns.plot(title=f'{RISKY_ASSET} returns: {START_DATE} - {END_DATE}')
```

Figura 7. Cálculo de los retornos diarios.
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Enseguida se incluye el texto para fines prácticos:

# calculo de los retornos diarios
adj_close = df['Adj Close']
returns = adj_close.pct_change().dropna()
print(f'Average return: {100 * returns.mean():.2f}%')
returns.plot(title=f'{RISKY_ASSET} returns: {START_DATE} - {END_DATE}')

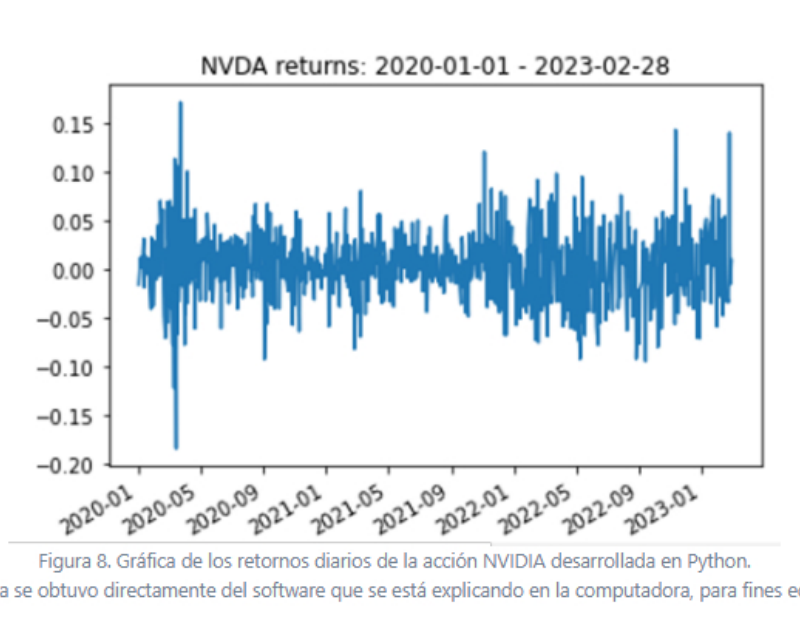
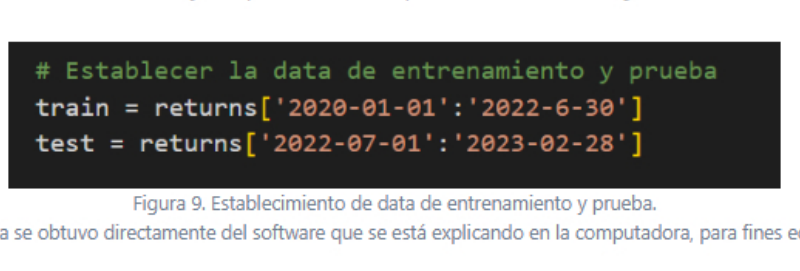


Figura 8. Gráfica de los retornos diarios de la acción NVIDIA desarrollada en Python.
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Enseguida se separa el periodo de tiempo para entrenamiento y prueba del modelo. El periodo de entrenamiento se define del 1 de enero del 2020 al 30 de junio del 2022 y el periodo de prueba del 1 de julio del 2022 al 28 de febrero del 2023.



```
# Establecer la data de entrenamiento y prueba
train = returns['2020-01-01':'2022-6-30']
test = returns['2022-07-01':'2023-02-28']
```

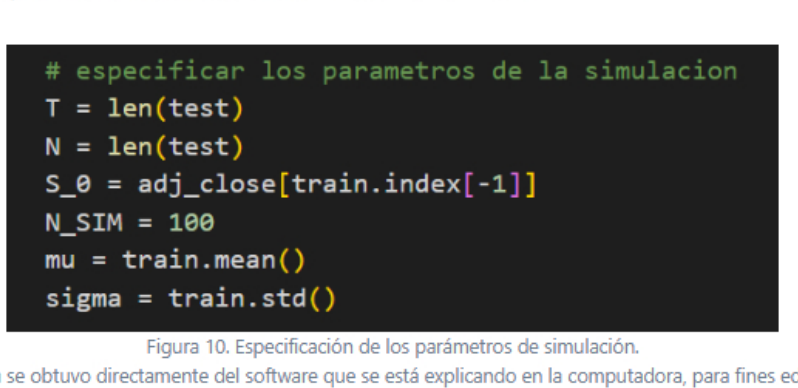
Figura 9. Establecimiento de data de entrenamiento y prueba.
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Enseguida se incluye el texto para fines prácticos:

# Establecer la data de entrenamiento y prueba
train = returns['2020-01-01':'2022-6-30']
test = returns['2022-07-01':'2023-02-28']

Se calcula el promedio (mu) y desviación estándar (sigma) de los retornos diarios del periodo de entrenamiento, que se utilizarán más adelante para las simulaciones. Adicionalmente se determinarán los parámetros del modelo:

T: horizonte del pronóstico, en este caso el número de días en los datos de prueba.
N: número de incrementos en el periodo del pronóstico, en este caso, diario.
S_0: el precio inicial. Para esta simulación se toma el último dato del periodo de entrenamiento.
N_SIM: número de rutas simuladas, en este caso se determina como 100.



```
# especificar los parametros de la simulacion
T = len(test)
N = len(test)
S_0 = adj_close[train.index[-1]]
N_SIM = 100
mu = train.mean()
sigma = train.std()
```

Figura 10. Especificación de los parámetros de simulación.
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Enseguida se incluye el texto para fines prácticos:

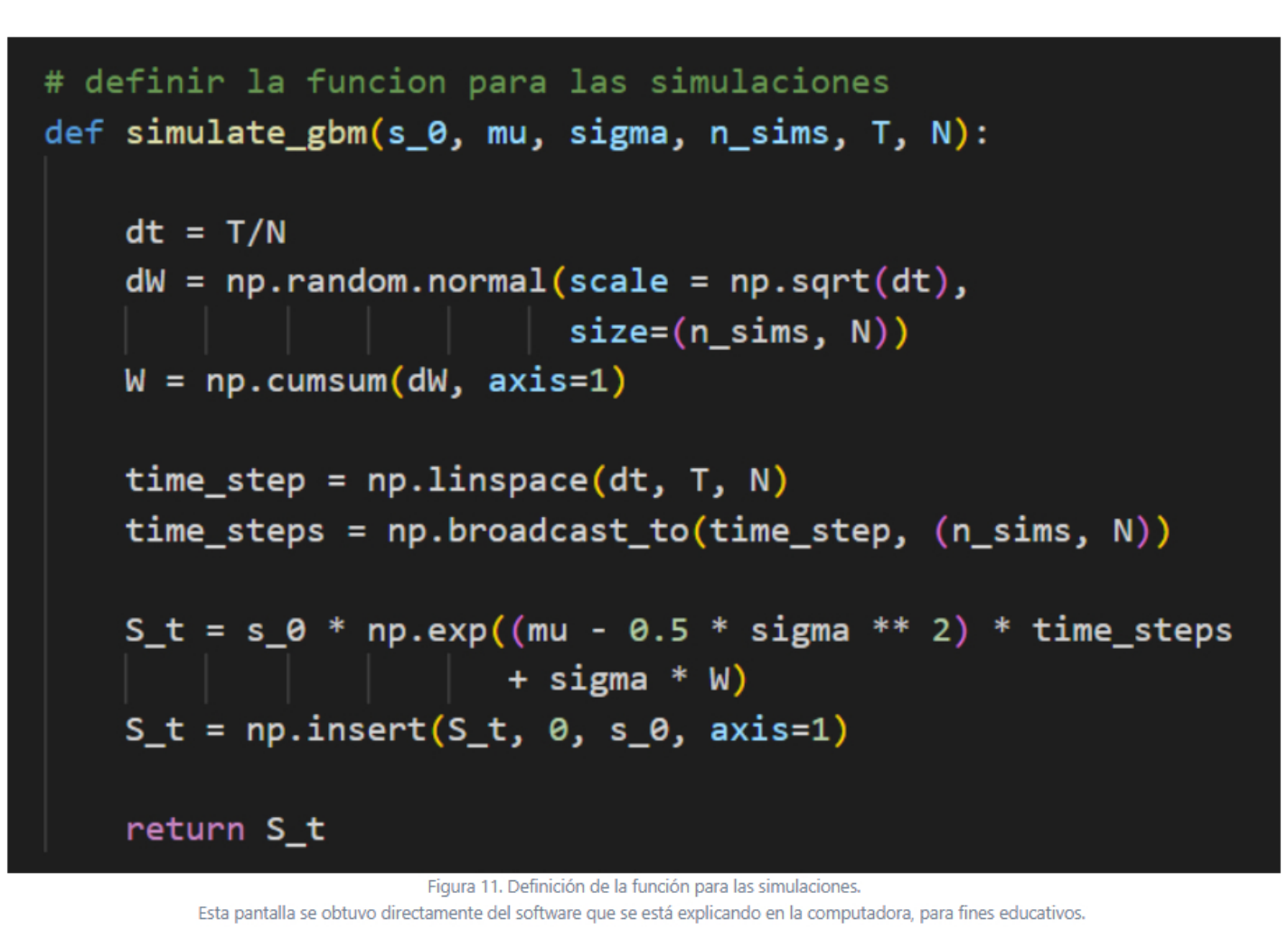
# especificar los parametros de la simulacion
T = len(test)
N = len(test)
S_0 = adj_close[train.index[-1]]
N_SIM = 100
mu = train.mean()
sigma = train.std()

El siguiente paso es el módulo para configurar y ejecutar las simulaciones, para lo cual se define la función simulate_gbm con los parámetros previamente definidos. Se comienza por determinar el incremento de tiempo (dt), en este caso de 1 día, y los incrementos brownianos (dw) utilizando el módulo np.random.normal de Python. Se calculan las rutas brownianas (W) por medio de una suma acumulativa (np.cumsum). Se crea una matriz que contiene los intervalos de tiempo (time_steps) espaciados uniformemente dentro de un intervalo (el horizonte de la simulación), usando np.linspace. Luego, se muestra la matriz con la forma deseada usando np.broadcast.

El precio de la acción en cada punto del tiempo es determinado con la misma ecuación descrita anteriormente:

$$S_{t+1} = S_t \exp\left(\mu - \frac{1}{2}\sigma^2\right)\Delta t + \sigma\sqrt{\Delta t}Z_{t+1}$$

Se define la función simulate_gbm para generar las simulaciones:



```
# definir la funcion para las simulaciones
def simulate_gbm(s_0, mu, sigma, n_sims, T, N):

    dt = T/N
    dW = np.random.normal(scale = np.sqrt(dt),
                          size=(n_sims, N))
    W = np.cumsum(dW, axis=1)

    time_step = np.linspace(dt, T, N)
    time_steps = np.broadcast_to(time_step, (n_sims, N))

    S_t = s_0 * np.exp((mu - 0.5 * sigma ** 2) * time_steps
                       + sigma * W)
    S_t = np.insert(S_t, 0, s_0, axis=1)

    return S_t
```

Figura 11. Definición de la función para las simulaciones.
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Enseguida se incluye el texto para fines prácticos:

# definir la funcion para las simulaciones
def simulate_gbm(s_0, mu, sigma, n_sims, T, N):

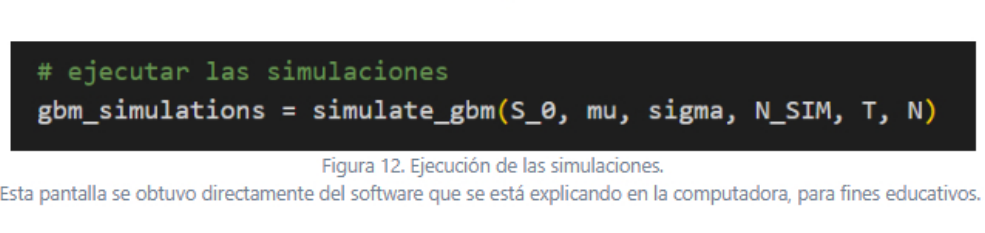
dt = T/N
dW = np.random.normal(scale = np.sqrt(dt),
size=(n_sims, N))
W = np.cumsum(dW, axis=1)

time_step = np.linspace(dt, T, N)
time_steps = np.broadcast_to(time_step, (n_sims, N))

S_t = s_0 * np.exp((mu - 0.5 * sigma ** 2) * time_steps
+ sigma * W)
S_t = np.insert(S_t, 0, s_0, axis=1)

return S_t

Se ejecutan las simulaciones. Se inserta el valor inicial S_0 en la primera columna de la matriz de simulaciones.



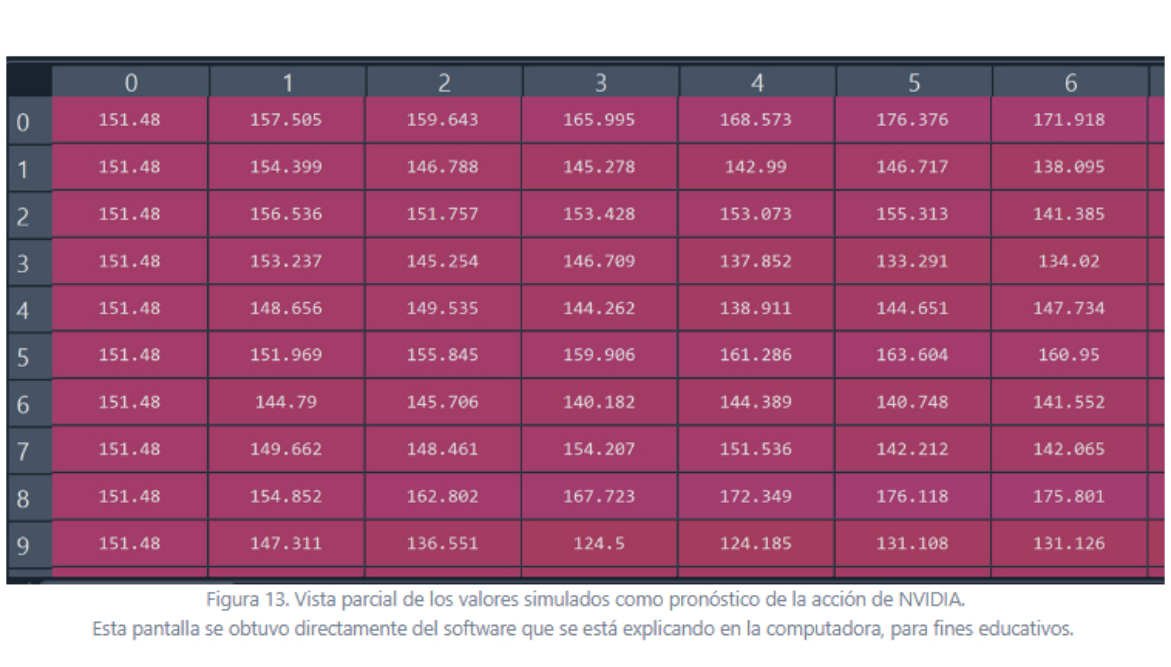
```
# ejecutar las simulaciones
gbm_simulations = simulate_gbm(S_0, mu, sigma, N_SIM, T, N)
```

Figura 12. Ejecución de las simulaciones.
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Enseguida se incluye el texto para fines prácticos:

# ejecutar las simulaciones
gbm_simulations = simulate_gbm(S_0, mu, sigma, N_SIM, T, N)

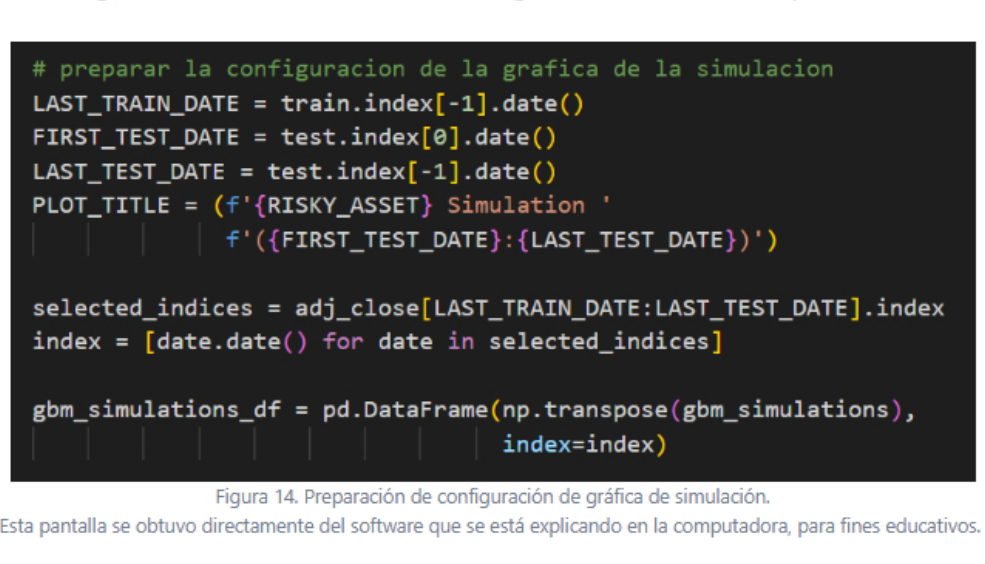
Lo cual genera una matriz (gbm_simulations) de 100 renglones con los valores pronosticados para los días en el periodo de prueba.



	0	1	2	3	4	5	6
0	151.48	157.505	159.643	165.995	168.573	176.376	171.918
1	151.48	154.399	146.788	145.278	142.99	146.717	138.095
2	151.48	156.536	151.757	153.428	153.073	155.313	141.385
3	151.48	153.237	145.254	146.709	137.852	133.291	134.02
4	151.48	148.656	149.535	144.262	138.911	144.651	147.734
5	151.48	151.969	155.845	159.906	161.286	163.604	160.95
6	151.48	144.79	145.706	140.182	144.389	140.748	141.552
7	151.48	149.662	148.461	154.207	151.536	142.212	142.065
8	151.48	154.852	162.802	167.723	172.349	176.118	175.801
9	151.48	147.311	136.551	124.5	124.185	131.108	131.126

Figura 13. Vista parcial de los valores simulados como pronóstico de la acción de NVIDIA.
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Se prepara la configuración de la gráfica, creando un dataframe gbm_simulations_df para facilitar la elaboración de la misma.



```
# preparar la configuracion de la grafica de la simulacion
LAST_TRAIN_DATE = train.index[-1].date()
FIRST_TEST_DATE = test.index[0].date()
LAST_TEST_DATE = test.index[-1].date()
PLOT_TITLE = (f'{RISKY_ASSET} Simulation '
              f'({FIRST_TEST_DATE}:{LAST_TEST_DATE})')

selected_indices = adj_close[LAST_TRAIN_DATE:LAST_TEST_DATE].index
index = [date.date() for date in selected_indices]

gbm_simulations_df = pd.DataFrame(np.transpose(gbm_simulations),
                                  index=index)
```

Figura 14. Preparación de configuración de gráfica de simulación.
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

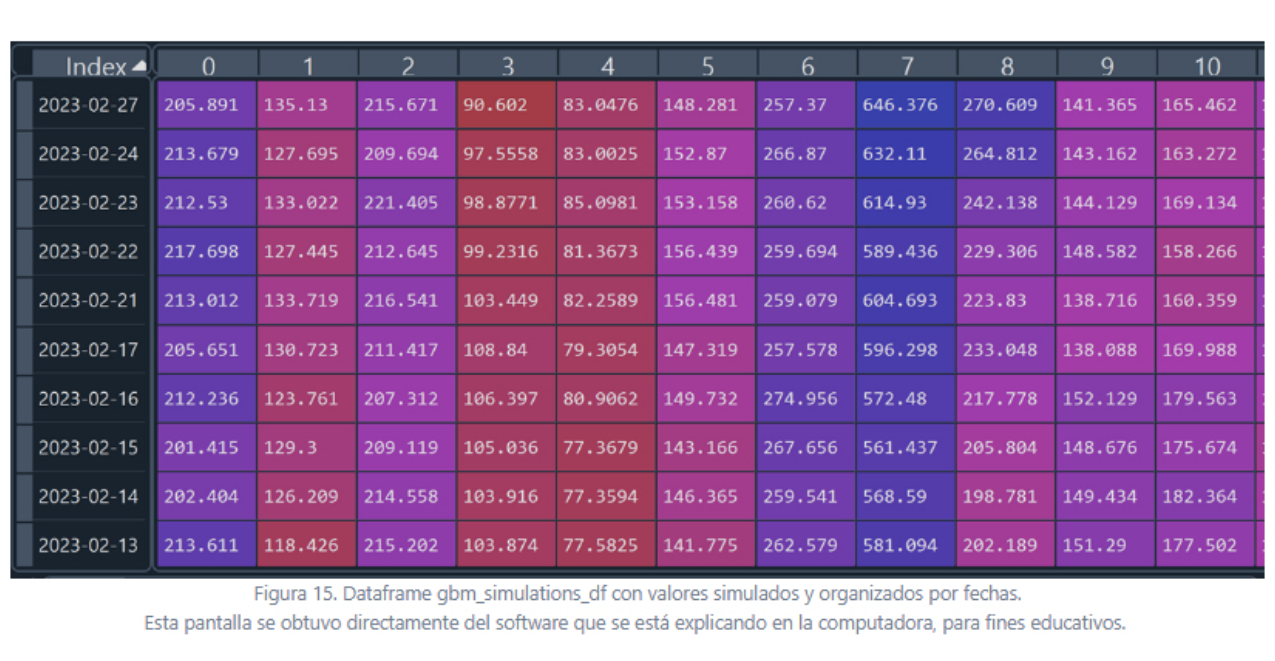
Enseguida se incluye el texto para fines prácticos:

# preparar la configuracion de la grafica de la simulacion
LAST_TRAIN_DATE = train.index[-1].date()
FIRST_TEST_DATE = test.index[0].date()
LAST_TEST_DATE = test.index[-1].date()
PLOT_TITLE = (f'{RISKY_ASSET} Simulation '
f'({FIRST_TEST_DATE}:{LAST_TEST_DATE})')

selected_indices = adj_close[LAST_TRAIN_DATE:LAST_TEST_DATE].index
index = [date.date() for date in selected_indices]

gbm_simulations_df = pd.DataFrame(np.transpose(gbm_simulations),
index=index)

Se cambia el dataframe de las simulaciones con las fechas como renglones y las columnas con las 100 simulaciones.



Index	0	1	2	3	4	5	6	7	8	9	10
2023-02-27	205.891	135.13	215.671	90.602	83.0476	148.281	257.37	646.376	270.609	141.365	165.462
2023-02-24	213.679	127.695	209.694	97.5558	83.0025	152.87	266.87	632.11	264.812	143.162	163.272
2023-02-23	212.53	133.022	221.405	98.8771	85.0981	153.158	260.62	614.93	242.138	144.129	169.134
2023-02-22	217.698	127.445	212.645	99.2316	81.3673	156.439	259.694	589.436	229.306	148.582	158.266
2023-02-21	213.012	133.719	216.541	103.449	82.2589	156.481	259.079	604.693	223.83	138.716	160.359
2023-02-17	205.651	130.723	211.417	108.84	79.3054	147.319	257.578	596.298	233.048	138.088	169.988
2023-02-16	212.236	123.761	207.312	106.397	80.9062	149.732	274.956	572.48	217.778	152.129	179.563
2023-02-15	201.415	129.3	209.119	105.036	77.3679	143.166	267.656	561.437	205.804	148.676	175.674
2023-02-14	202.404	126.209	214.558	103.916	77.3594	146.365	259.541	568.59	198.781	149.434	182.364
2023-02-13	213.611	118.426	215.202	103.874	77.5825	141.775	262.579	581.094	202.189	151.29	177.502

Figura 15. Dataframe gbm_simulations_df con valores simulados y organizados por fechas.
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Enseguida se incluye el texto para fines prácticos:

# graficas
ax = gbm_simulations_df.plot(alpha=0.2, legend=False)
line_1, = ax.plot(index, gbm_simulations_df.mean(axis=1),
color='red')
line_2, = ax.plot(index, adj_close[LAST_TRAIN_DATE:LAST_TEST_DATE],
color='blue')
ax.set_title(PLOT_TITLE, fontsize=16)
ax.legend((line_1, line_2), ('mean', 'actual'))

Y finalmente se obtiene la gráfica deseada.

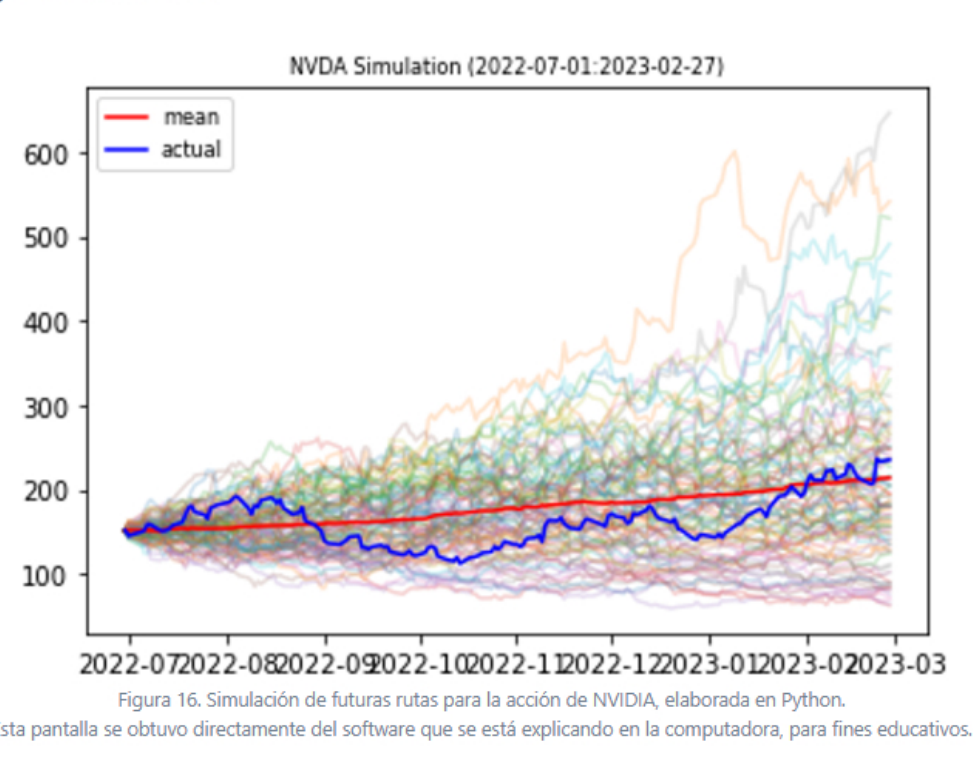
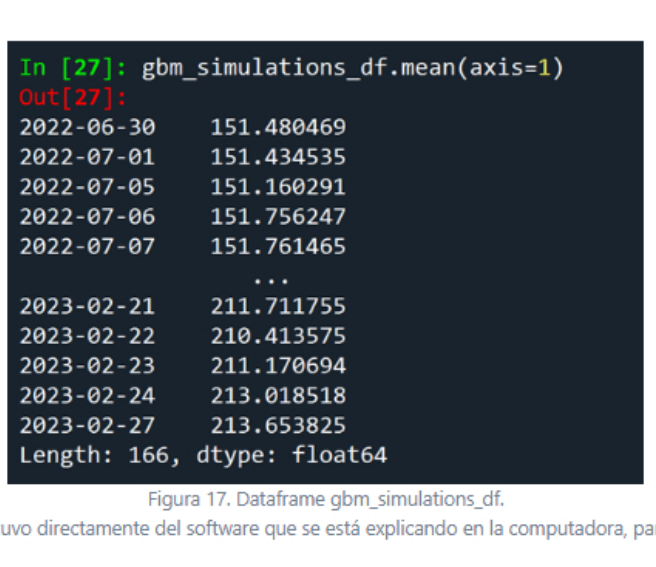


Figura 16. Simulación de futuras rutas para la acción de NVIDIA, elaborada en Python.
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

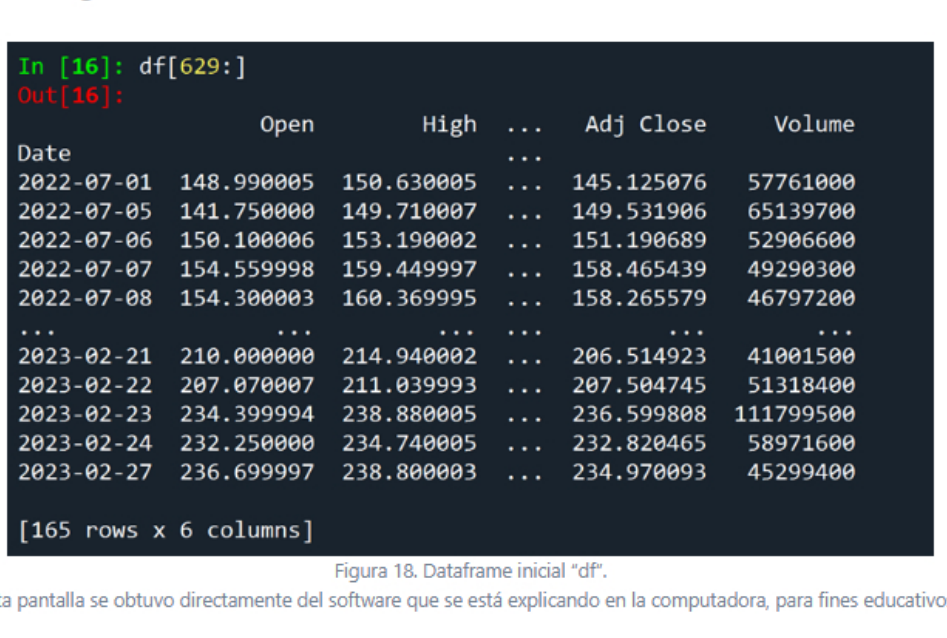
En esta gráfica se muestran las trayectorias simuladas por el método de movimiento browniano geométrico para el periodo de prueba. El valor promedio de las simulaciones muestra una tendencia creciente (línea color rojo), inicia de 151.48 y termina en 213.65. Los valores completos de la línea promedio se obtienen del dataframe gbm_simulations_df:



```
In [27]: gbm_simulations_df.mean(axis=1)
Out[27]:
2022-06-30    151.480469
2022-07-01    151.434535
2022-07-05    151.160291
2022-07-06    151.756247
2022-07-07    151.761465
                 ...
2023-02-21    211.711755
2023-02-22    210.413575
2023-02-23    211.170694
2023-02-24    213.018518
2023-02-27    213.653825
Length: 166, dtype: float64
```

Figura 17. Dataframe gbm_simulations_df.
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

La línea azul representa los valores reales del periodo de prueba, iniciando en 145.13 y finalizando en 234.97, los cuales están en el dataframe inicial "df" del renglón 629 al final.



```
In [16]: df[629:]
Out[16]:
                  Open        High  ...   Adj Close     Volume
Date                                ...
2022-07-01  148.990005  150.630005  ...  145.125076   57761000
2022-07-05  141.750000  149.710007  ...  149.531906   65139700
2022-07-06  150.100006  153.190002  ...  151.190689   52906600
2022-07-07  154.559998  159.449997  ...  158.465439   49290300
2022-07-08  154.300003  160.369995  ...  158.265579   46797200
...                ...         ...  ...         ...        ...
2023-02-21  210.000000  214.940002  ...  206.514923   41001500
2023-02-22  207.070007  211.039993  ...  207.504745   51318400
2023-02-23  234.399994  238.880005  ...  236.599808  111799500
2023-02-24  232.250000  234.740005  ...  232.820465   58971600
2023-02-27  236.699997  238.800003  ...  234.970093   45299400

[165 rows x 6 columns]
```

Figura 18. Dataframe inicial "df".
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Esta visualización es visible para un número razonable de rutas de la muestra. En los casos de la vida real, se necesita usar significativamente más de 100 rutas. Mientras mayor sea el número de rutas muestra en la simulación, los resultados serán de mayor veracidad/confiabilidad.

Fijación de opciones europeas mediante simulaciones

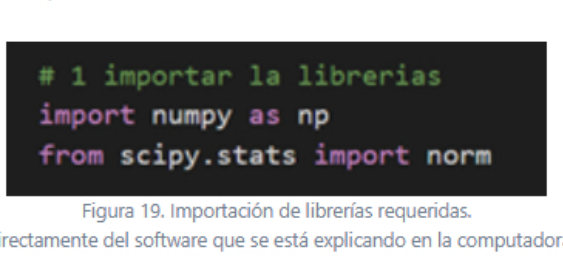
Los *instrumentos opciones* son un tipo de instrumento derivado porque su precio está ligado al precio subyacente del activo financiero. Por medio de la compra de contratos de opciones se garantiza el derecho, pero no la obligación, de comprar o vender (*call/put*) un activo subyacente en un precio establecido (conocido como *strike*) con cierto tiempo de anticipación. La principal razón de la demanda de instrumentos opciones es porque es un instrumento de cobertura contra la exposición de los movimientos de un activo en forma no deseada (Lewinson, 2020).

Específicamente las opciones europeas son diferentes porque tienen una fórmula analítica establecida para su valuación, lo cual no es el caso de derivados más avanzados, como las opciones americanas y exóticas (Chen, 2022). Por lo tanto, es viable utilizar Monte Carlo para simular su precio en un periodo de tiempo.

Utilizando simulaciones Monte Carlo para valuar opciones, es mediante una valuación neutral del riesgo, se asume que el premio de la opción incrementa al mismo ritmo que la tasa libre de riesgo, la cual se utilizará para descontar el pago futuro y calcular su valor presente. Para cada una de las rutas simuladas, se calcula el pago futuro a vencimiento, se determina el promedio de todas las rutas y se descuenta este promedio a valor presente.

El siguiente código Python está basado en la publicación de Lewinson (2020).

Se inicia con la importación de las librerías requeridas.



```
# 1 importar la librerias
import numpy as np
from scipy.stats import norm
```

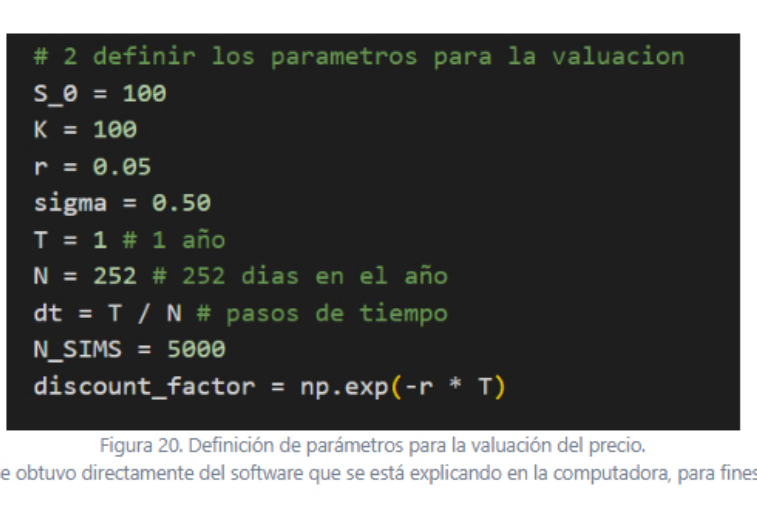
Figura 19. Importación de librerías requeridas.
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Enseguida se incluye el texto para fines prácticos:

# 1 importar la librerias
import numpy as np
from scipy.stats import norm

Se definen los parámetros para la valuación del precio:

- S_0 = precio inicial de la acción.
- K = precio de ejercer la opción, en este caso igual que el precio inicial $100.
- r = tasa de interés libre de riesgo.
- sigma= volatilidad de la acción (desviación estándar) anualizada.
- T = tiempo al vencimiento en años.
- N = número de incrementos de tiempo para las simulaciones.
- n_sims = número de rutas simuladas.
- discount_factor = factor para calcular el valor presente del futuro precio.



```
# 2 definir los parametros para la valuacion
S_0 = 100
K = 100
r = 0.05
sigma = 0.50
T = 1 # 1 año
N = 252 # 252 dias en el año
dt = T / N # pasos de tiempo
N_SIMS = 5000
discount_factor = np.exp(-r * T)
```

Figura 20. Definición de parámetros para la valuación del precio.
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Enseguida se incluye el texto para fines prácticos:

# 2 definir los parametros para la valuacion
S_0 = 100
K = 100
r = 0.05
sigma = 0.50
T = 1 # 1 año
N = 252 # 252 dias en el año
dt = T / N # pasos de tiempo
N_SIMS = 5000
discount_factor = np.exp(-r * T)

La opción *call* es cuando se tiene la opción de comprar un activo financiero en la fecha de vencimiento a un precio predeterminado, también conocido como *strike price*. Por el contrario, la opción *put* es cuando se vende un activo financiero con fecha y precio predeterminado.

En el siguiente paso, se definirá en Python una función para calcular el premio de la opción, usando las fórmulas del modelo Black-Scholes basada en procesos estocásticos. La solución analítica para determinar el precio de la opción de compra call (C(St,t)) y la opción de venta put (P(St,t)) es la siguiente (Lewinson, 2020):

$$C(S_t, t) = N(d_1)\, S_t - N(d_2)Ke^{-r(T-t)}$$
$$P(S_t, t) = N(-d_2)Ke^{-r(T-t)} - N(-d_1)S_t$$

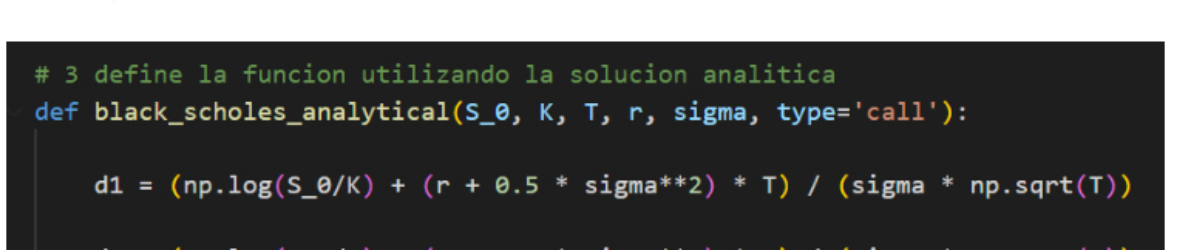
N(d₁) y N(d₂) = son valores de la función de distribución acumulada (CDF) normal estándar para ponderar el precio del activo en el tiempo t y al vencimiento. Se plantean d_1 y d_2 como funciones estocásticas de movimiento browniano.
S_t = precio del activo al tiempo t.
K = precio al vencimiento (strike price). $Ke^{-r(T-t)}$ es el valor presente del precio al vencimiento.
r = tasa libre de riesgo.
σ = desviación estándar.
(T - t) = plazo para el vencimiento en el tiempo t.

Las ecuaciones que aplican como funciones estocásticas de movimiento browniano son las siguientes:

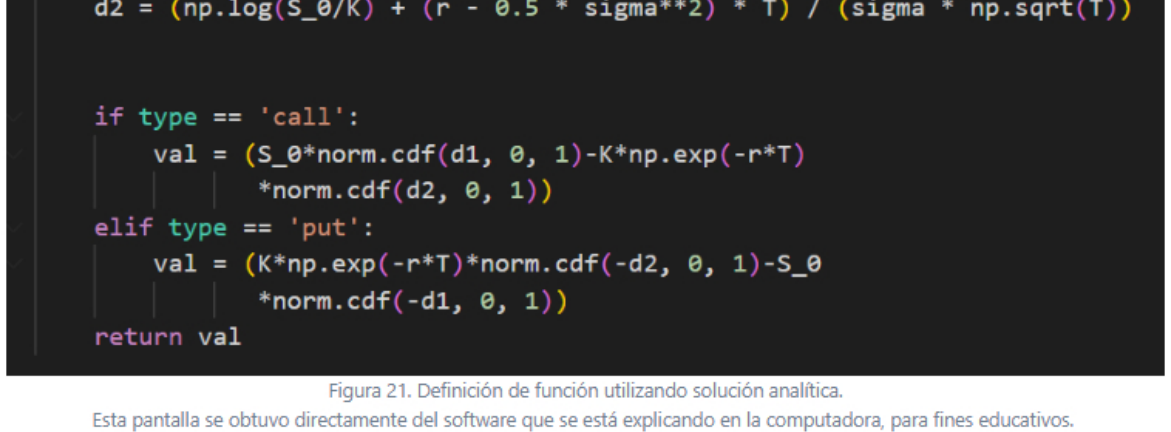
$$d_1 = \frac{1}{\sigma\sqrt{T-t}}\left[ln\left(\frac{S_t}{K}\right) + \left(r + \frac{\sigma^2}{2}\right)(T-t)\right]$$

$$d_2 = d1 - \sigma\sqrt{T-t}$$

Se aplican las fórmulas en Python.



```
# 3 define la funcion utilizando la solucion analitica
def black_scholes_analytical(S_0, K, T, r, sigma, type='call'):

    d1 = (np.log(S_0/K) + (r + 0.5 * sigma**2) * T) / (sigma * np.sqrt(T))
```



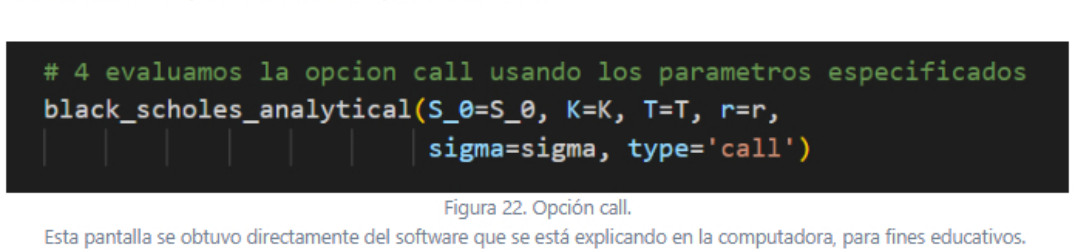
```
    d2 = (np.log(S_0/K) + (r - 0.5 * sigma**2) * T) / (sigma * np.sqrt(T))

    if type == 'call':
        val = (S_0*norm.cdf(d1, 0, 1)-K*np.exp(-r*T)
              *norm.cdf(d2, 0, 1))
    elif type == 'put':
        val = (K*np.exp(-r*T)*norm.cdf(-d2, 0, 1)-S_0
              *norm.cdf(-d1, 0, 1))
    return val
```

Figura 21. Definición de función utilizando solución analítica.
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Enseguida se incluye el texto para fines prácticos:

```
# 3 define la funcion utilizando la solucion analitica
def black_scholes_analytical(S_0, K, T, r, sigma, type='call'):

  d1 = (np.log(S_0/K) + (r + 0.5 * sigma**2) * T) / (sigma * np.sqrt(T))

  d2 = (np.log(S_0/K) + (r - 0.5 * sigma**2) * T) / (sigma * np.sqrt(T))


  if type == 'call':
    val = (S_0*norm.cdf(d1, 0, 1)-K*np.exp(-r*T)
        *norm.cdf(d2, 0, 1))
  elif type == 'put':
    val = (K*np.exp(-r*T)*norm.cdf(-d2, 0, 1)-S_0
        *norm.cdf(-d1, 0, 1))
  return val
```

Se utilizará la función creada en el paso anterior black-scholes_analytical para calcular las simulaciones Monte Carlo. En este caso, se evalúa la opción *call* con los parámetros especificados.



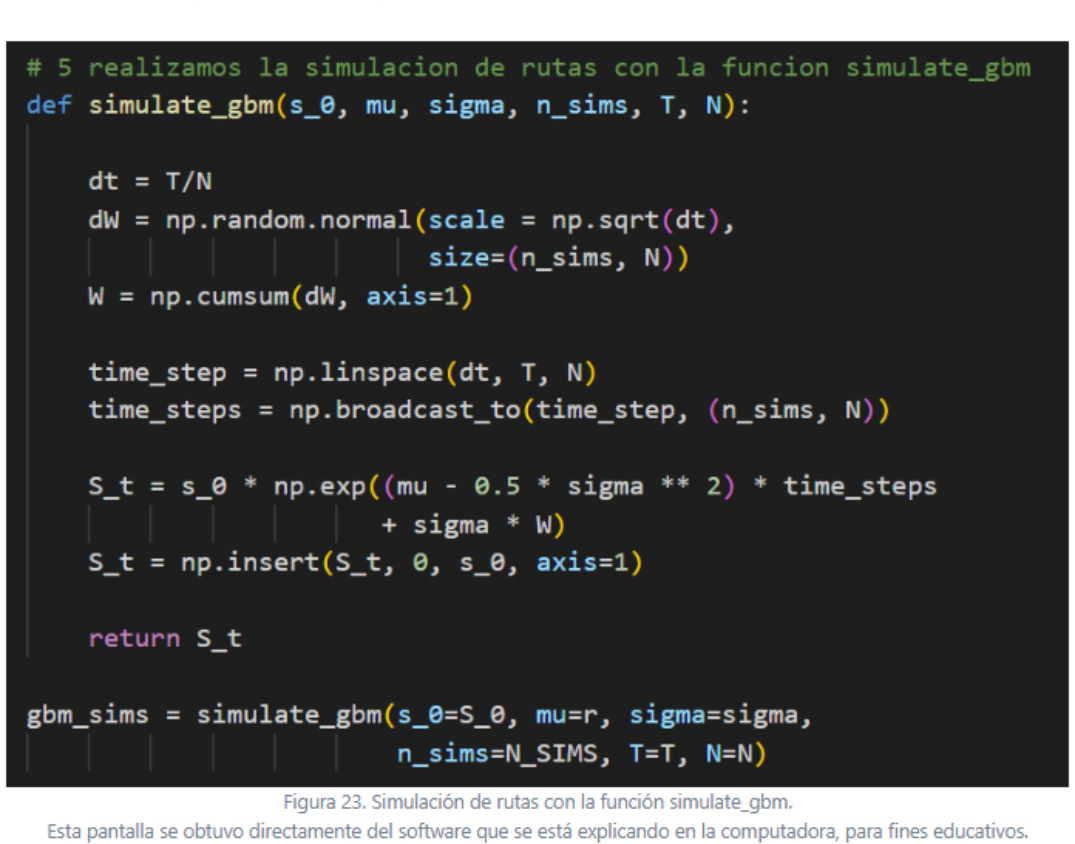
```
# 4 evaluamos la opcion call usando los parametros especificados
black_scholes_analytical(S_0=S_0, K=K, T=T, r=r,
                         sigma=sigma, type='call')
```

Figura 22. Opción call.
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Enseguida se incluye el texto para fines prácticos:

```
# 4 evaluamos la opcion call usando los parametros especificados
black_scholes_analytical(S_0=S_0, K=K, T=T, r=r,
                 sigma=sigma, type='call')
```

Se crea la función para realizar las simulaciones simulate_gbm.



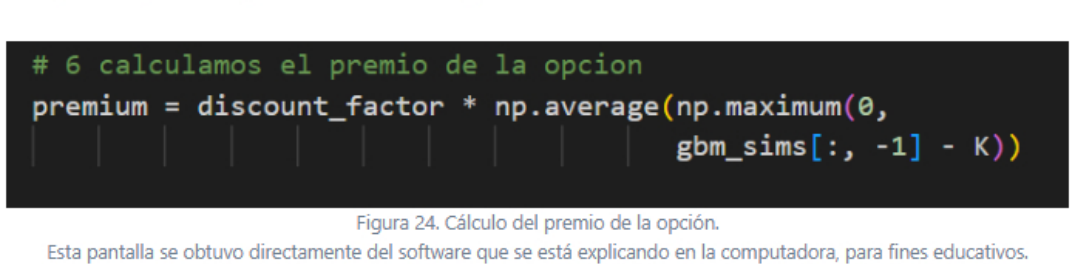
```
# 5 realizamos la simulacion de rutas con la funcion simulate_gbm
def simulate_gbm(s_0, mu, sigma, n_sims, T, N):

    dt = T/N
    dW = np.random.normal(scale = np.sqrt(dt),
                          size=(n_sims, N))
    W = np.cumsum(dW, axis=1)

    time_step = np.linspace(dt, T, N)
    time_steps = np.broadcast_to(time_step, (n_sims, N))

    S_t = s_0 * np.exp((mu - 0.5 * sigma ** 2) * time_steps
                       + sigma * W)
    S_t = np.insert(S_t, 0, s_0, axis=1)

    return S_t

gbm_sims = simulate_gbm(s_0=S_0, mu=r, sigma=sigma,
                        n_sims=N_SIMS, T=T, N=N)
```

Figura 23. Simulación de rutas con la función simulate_gbm.
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Enseguida se incluye el texto para fines prácticos:

```
# 5 realizamos la simulacion de rutas con la funcion simulate_gbm
def simulate_gbm(s_0, mu, sigma, n_sims, T, N):

  dt = T/N
  dW = np.random.normal(scale = np.sqrt(dt),
            size=(n_sims, N))
  W = np.cumsum(dW, axis=1)

  time_step = np.linspace(dt, T, N)
  time_steps = np.broadcast_to(time_step, (n_sims, N))

  S_t = s_0 * np.exp((mu - 0.5 * sigma ** 2) * time_steps
           + sigma * W)
  S_t = np.insert(S_t, 0, s_0, axis=1)

  return S_t

gbm_sims = simulate_gbm(s_0=S_0, mu=r, sigma=sigma,
           n_sims=N_SIMS, T=T, N=N)
```

Se calcula el premio de la opción y se asigna a la variable premium.



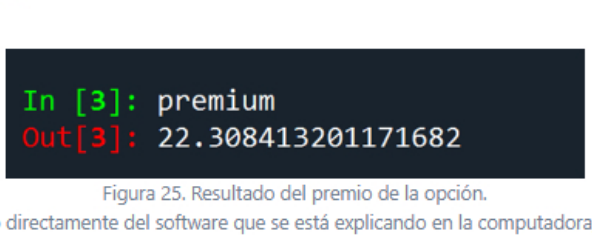
```
# 6 calculamos el premio de la opcion
premium = discount_factor * np.average(np.maximum(0,
                                       gbm_sims[:, -1] - K))
```

Figura 24. Cálculo del premio de la opción.
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Enseguida se incluye el texto para fines prácticos:

```
# 6 calculamos el premio de la opcion
premium = discount_factor * np.average(np.maximum(0,
                 gbm_sims[:, -1] - K))
```

El premio calculado de la opción es $22.31.



```
In [3]: premium
Out[3]: 22.308413201171682
```

Figura 25. Resultado del premio de la opción.
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Nuevamente la veracidad del modelo está en función de la cantidad de rutas simuladas (utilizando el parámetro N_SIMS).

Estimación del valor en riesgo VaR utilizando Monte Carlo

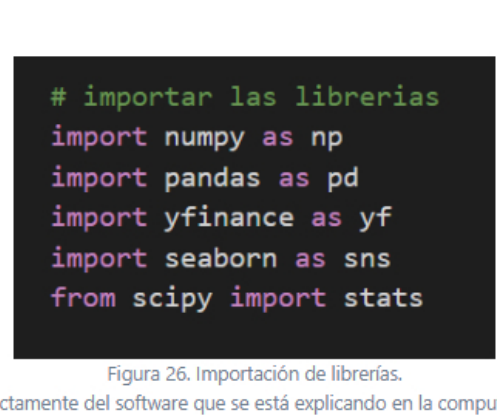
El valor en riesgo (VaR) es un métrico financiero importante que mide el riesgo asociado con un conjunto de activos financieros, posición de activos o cartera. Este nuevo métrico del riesgo, determina la pérdida potencial de la cartera con un nivel determinado de confianza, regularmente 95%, sobre un predeterminado horizonte de tiempo y bajo condiciones normales de mercado.

De acuerdo con Kenton (2023), el VaR puede ser calculado de tres formas: valores históricos, varianza-covarianza y simulación Monte Carlo. Por ejemplo, supón que el VaR de la cartera es de $100, significa que con un nivel de confianza del 95% (bajo condiciones normales de mercado) no se perderá más de $100 si se mantiene un portafolio por un plazo determinado.

La mayoría de las instituciones intermediarias financieras integran y ofrecen fondos de inversión a los ahorradores e inversionistas, con muy variadas características y especialización de su cartera, como fondos de renta fija, de renta variable, nacional e internacional, de materias primas, tipos de cambio, metales, etc. Regularmente se publica el valor del VaR del fondo de inversión a todo el público en su prospecto de colocación. Los administradores de riesgo lo utilizan para medir y controlar el nivel de exposición al riesgo. Usualmente, hay dos métodos para estimar el VaR. El primero se basa en el supuesto de que los retornos de nuestro portafolio siguen una distribución normal, y el segundo depende del *ranking* de los retornos históricos (Yan, 2017).

A continuación, se desarrollará el cálculo del VaR de un día para un portafolio compuesto por dos activos financieros (Facebook y Google), utilizando Python, tomando como referencia el código publicado por Lewinson (2020).

Se importa las librerías requeridas.



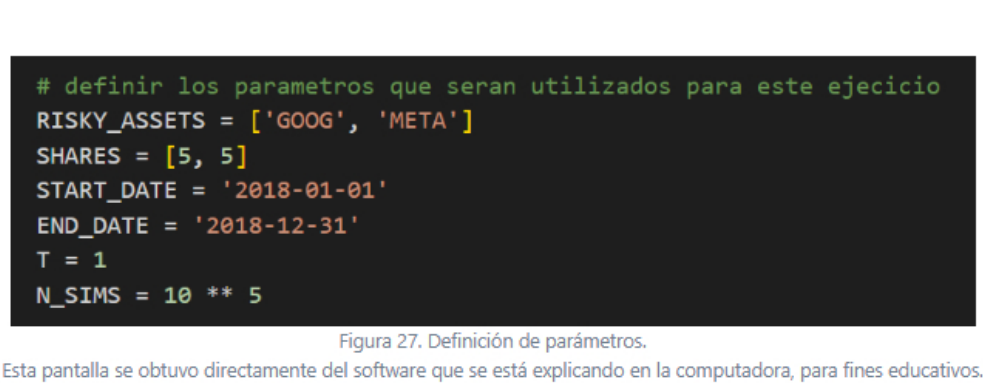
```
# importar las librerias
import numpy as np
import pandas as pd
import yfinance as yf
import seaborn as sns
from scipy import stats
```

Figura 26. Importación de librerías.
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Enseguida se incluye el texto para fines prácticos:

```
# importar las librerias
import numpy as np
import pandas as pd
import yfinance as yf
import seaborn as sns
from scipy import stats
```

Definir los parámetros, incluyendo los símbolos de Google (GOOG) y Facebook (META), la cantidad de acciones de cada activo en el portafolio, el periodo de análisis, el plazo del VaR que es un día y el número de simulaciones que se van a realizar.



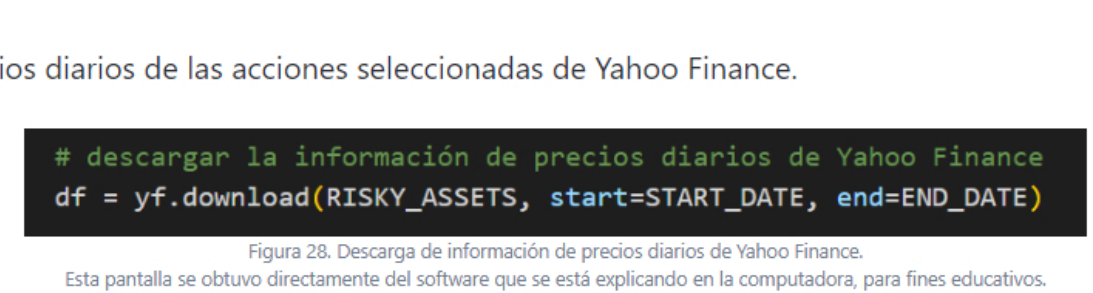
```
# definir los parametros que seran utilizados para este ejercicio
RISKY_ASSETS = ['GOOG', 'META']
SHARES = [5, 5]
START_DATE = '2018-01-01'
END_DATE = '2018-12-31'
T = 1
N_SIMS = 10 ** 5
```

Figura 27. Definición de parámetros.
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Enseguida se incluye el texto para fines prácticos:

```
# definir los parametros que seran utilizados para este ejercicio
RISKY_ASSETS = ['GOOG', 'META']
SHARES = [5, 5]
START_DATE = '2018-01-01'
END_DATE = '2018-12-31'
T = 1
N_SIMS = 10 ** 5
```

Se descargan los precios diarios de las acciones seleccionadas de Yahoo Finance.



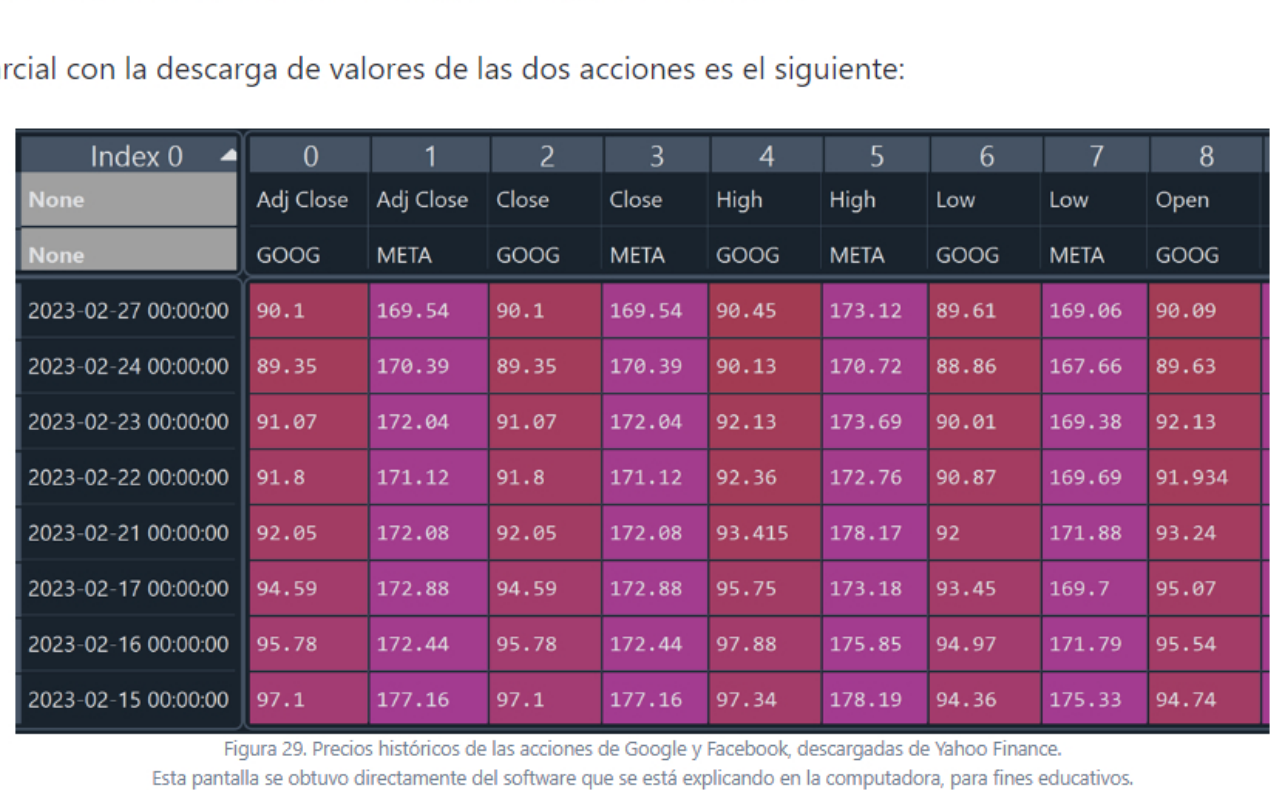
```
# descargar la información de precios diarios de Yahoo Finance
df = yf.download(RISKY_ASSETS, start=START_DATE, end=END_DATE)
```

Figura 28. Descarga de información de precios diarios de Yahoo Finance.
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Enseguida se incluye el texto para fines prácticos:

```
# descargar la información de precios diarios de Yahoo Finance
df = yf.download(RISKY_ASSETS, start=START_DATE, end=END_DATE)
```

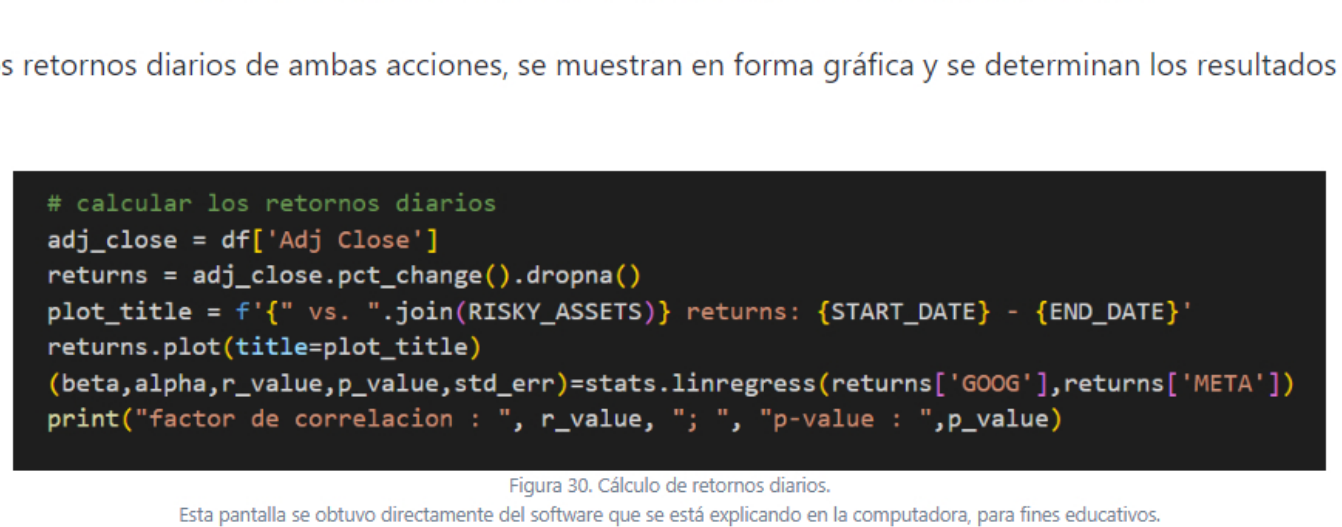
El dataframe parcial con la descarga de valores de las dos acciones es el siguiente:



Index 0	0	1	2	3	4	5	6	7	8
None	Adj Close	Adj Close	Close	Close	High	High	Low	Low	Open
None	GOOG	META	GOOG	META	GOOG	META	GOOG	META	GOOG
2023-02-27 00:00:00	90.1	169.54	90.1	169.54	90.45	173.12	89.61	169.06	90.09
2023-02-24 00:00:00	89.35	170.39	89.35	170.39	90.13	170.72	88.86	167.66	89.63
2023-02-23 00:00:00	91.07	172.04	91.07	172.04	92.13	173.69	90.01	169.38	92.13
2023-02-22 00:00:00	91.8	171.12	91.8	171.12	92.36	172.76	90.87	169.69	91.934
2023-02-21 00:00:00	92.05	172.08	92.05	172.08	93.415	178.17	92	171.88	93.24
2023-02-17 00:00:00	94.59	172.88	94.59	172.88	95.75	173.18	93.45	169.7	95.07
2023-02-16 00:00:00	95.78	172.44	95.78	172.44	97.88	175.85	94.97	171.79	95.54
2023-02-15 00:00:00	97.1	177.16	97.1	177.16	97.34	178.19	94.36	175.33	94.74

Figura 29. Precios históricos de las acciones de Google y Facebook, descargadas de Yahoo Finance.
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Se calculan los retornos diarios de ambas acciones, se muestran en forma gráfica y se determinan los resultados de la correlación.



```
# calcular los retornos diarios
adj_close = df['Adj Close']
returns = adj_close.pct_change().dropna()
plot_title = f'{" vs. ".join(RISKY_ASSETS)} returns: {START_DATE} - {END_DATE}'
returns.plot(title=plot_title)
(beta,alpha,r_value,p_value,std_err)=stats.linregress(returns['GOOG'],returns['META'])
print("factor de correlacion : ", r_value, "; ", "p-value : ",p_value)
```

Figura 30. Cálculo de retornos diarios.
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Enseguida se incluye el texto para fines prácticos:

```
# calcular los retornos diarios
adj_close = df['Adj Close']
returns = adj_close.pct_change().dropna()
plot_title = f'{" vs. ".join(RISKY_ASSETS)} returns: {START_DATE} - {END_DATE}'
returns.plot(title=plot_title)
(beta,alpha,r_value,p_value,std_err)=stats.linregress(returns['GOOG'],returns['META'])
print("factor de correlacion : ", r_value, "; ", "p-value : ",p_value)
```

La gráfica de los retornos diarios de cada activo se muestra a continuación.

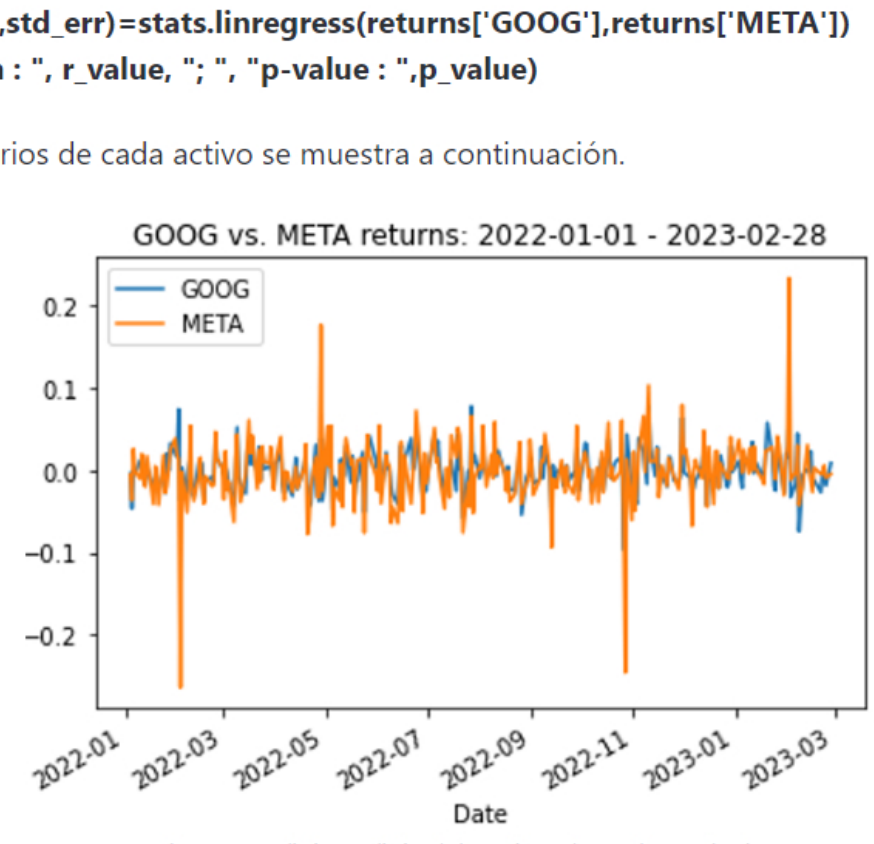
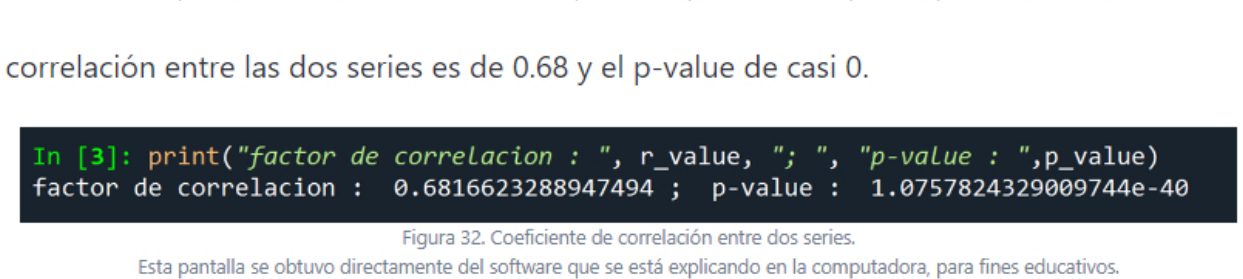


Figura 31. Rendimientos diarios de las acciones de Google y Facebook.
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

El coeficiente de correlación entre las dos series es de 0.68 y el p-value de casi 0.



```
In [3]: print("factor de correlacion : ", r_value, "; ", "p-value : ",p_value)
factor de correlacion :  0.6816623288947494 ;  p-value :  1.0757824329009744e-40
```

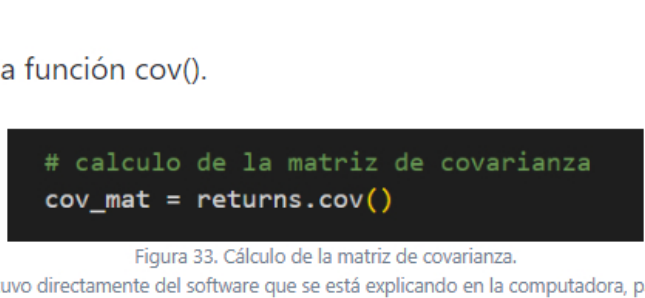
Figura 32. Coeficiente de correlación entre dos series.
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Hay dos formas para enfocar los cálculos del VaR:

- Cálculo del VaR sobre los precios. Con el número de acciones y el precio de cada activo se puede calcular el valor del portafolio actual y su posible valor X días después.
- Cálculo del VaR sobre los retornos. Usando el ranking de los retornos de cada activo en el portafolio y los rendimientos esperados se puede calcular el retorno esperado del portafolio en X días después. Con esto, se expresa VaR como la cantidad en dinero basado en ese retorno y el valor actual del portafolio.

El enfoque de simulaciones Monte Carlo que se utiliza para estimar el precio de un activo emplea valores aleatorios sobre la distribución normal estándar. Para el caso de calcular el VaR del portafolio, se aplica la covarianza histórica de los retornos diarios a los valores aleatorios generados por la simulación. Para tal efecto, se utiliza la descomposición de matrices Cholesky y se multiplica por la matriz de variables aleatorias. De esta forma se obtiene la nueva matriz de variables aleatorias ajustada por la covarianza de los activos en el portafolio.

Se calcula la matriz de covarianza con la función cov().



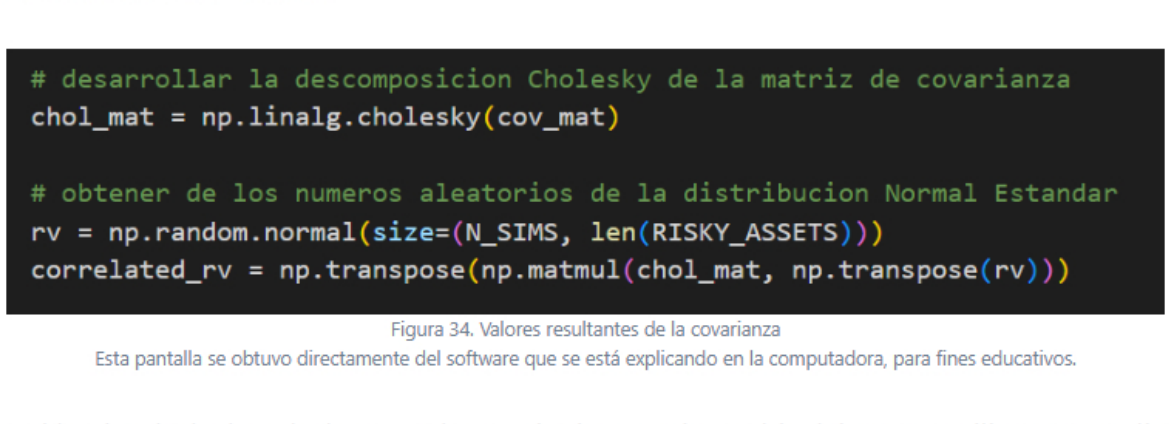
```
# calculo de la matriz de covarianza
cov_mat = returns.cov()
```

Figura 33. Cálculo de la matriz de covarianza.
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Enseguida se incluye el texto para fines prácticos:

```
# calculo de la matriz de covarianza
cov_mat = returns.cov()
```

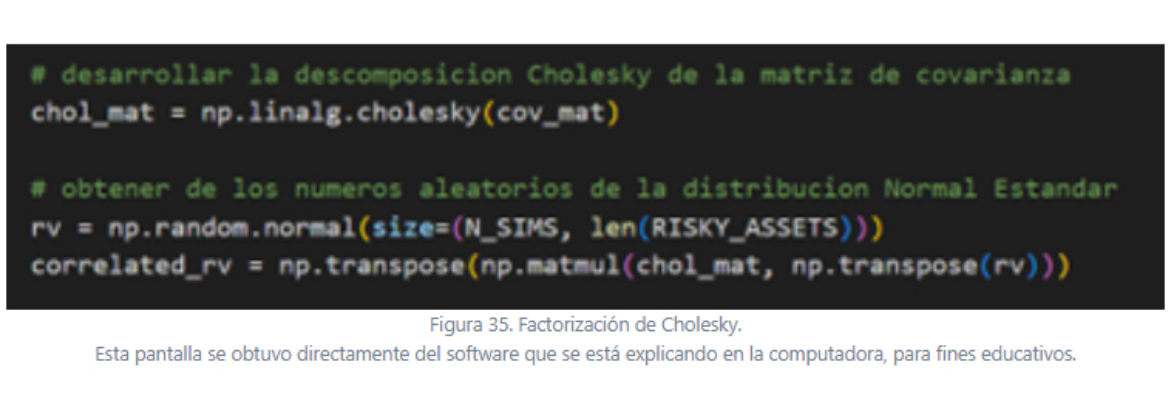
Los valores resultantes de covarianza indican que los activos tienen una relación positiva 0.000695. Los valores en el mismo activo son la varianza de los retornos diarios.



```
# desarrollar la descomposicion Cholesky de la matriz de covarianza
chol_mat = np.linalg.cholesky(cov_mat)

# obtener de los numeros aleatorios de la distribucion Normal Estandar
rv = np.random.normal(size=(N_SIMS, len(RISKY_ASSETS)))
correlated_rv = np.transpose(np.matmul(chol_mat, np.transpose(rv)))
```

Figura 34. Valores resultantes de la covarianza
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Se ejecuta la factorización de Cholesky (chol_mat) a la matriz de covarianza histórica y se utiliza para aplicar la covarianza histórica a la matriz de números aleatorios de la distribución normal estándar (correlated_rv).



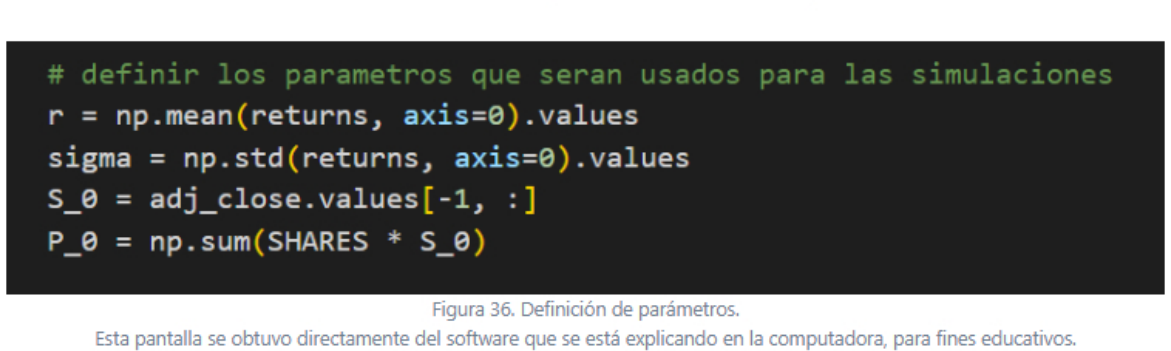
```
# desarrollar la descomposicion Cholesky de la matriz de covarianza
chol_mat = np.linalg.cholesky(cov_mat)

# obtener de los numeros aleatorios de la distribucion Normal Estandar
rv = np.random.normal(size=(N_SIMS, len(RISKY_ASSETS)))
correlated_rv = np.transpose(np.matmul(chol_mat, np.transpose(rv)))
```

Figura 35. Factorización de Cholesky.
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Enseguida se incluye el texto para fines prácticos:

```
# desarrollar la descomposicion Cholesky de la matriz de covarianza
chol_mat = np.linalg.cholesky(cov_mat)

# obtener de los numeros aleatorios de la distribucion Normal Estandar
rv = np.random.normal(size=(N_SIMS, len(RISKY_ASSETS)))
correlated_rv = np.transpose(np.matmul(chol_mat, np.transpose(rv)))
```

Se definen los parámetros a utilizar para las simulaciones. Se calcula el promedio y la desviación estándar de cada activo, se obtiene el valor más reciente conocido de cada activo (S_0) y el valor inicial del portafolio (P_0).



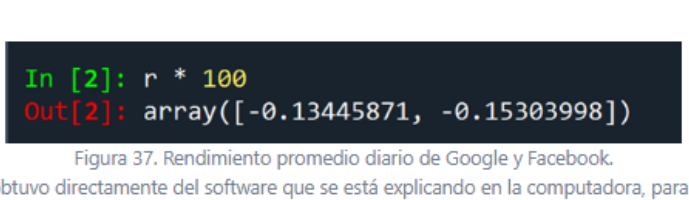
```
# definir los parametros que seran usados para las simulaciones
r = np.mean(returns, axis=0).values
sigma = np.std(returns, axis=0).values
S_0 = adj_close.values[-1, :]
P_0 = np.sum(SHARES * S_0)
```

Figura 36. Definición de parámetros.
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Enseguida se incluye el texto para fines prácticos:

```
# definir los parametros que seran usados para las simulaciones
r = np.mean(returns, axis=0).values
sigma = np.std(returns, axis=0).values
S_0 = adj_close.values[-1, :]
P_0 = np.sum(SHARES * S_0)
```

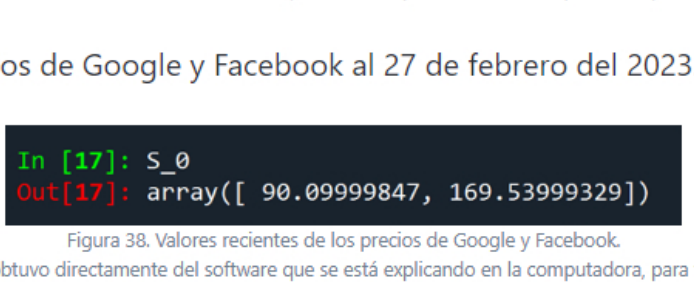
El rendimiento promedio diario de Google y Facebook es -0.13% y -0.15%, respectivamente.



```
In [2]: r * 100
Out[2]: array([-0.13445871, -0.15303998])
```

Figura 37. Rendimiento promedio diario de Google y Facebook.
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

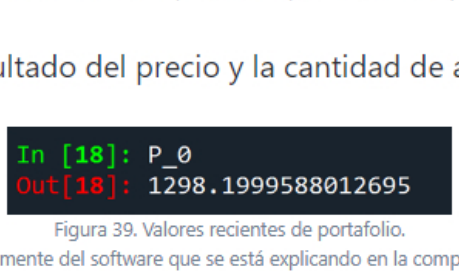
Los valores más recientes de los precios de Google y Facebook al 27 de febrero del 2023.



```
In [17]: S_0
Out[17]: array([ 90.09999847, 169.53999329])
```

Figura 38. Valores recientes de los precios de Google y Facebook.
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

El valor más reciente del portafolio, como resultado del precio y la cantidad de acciones.

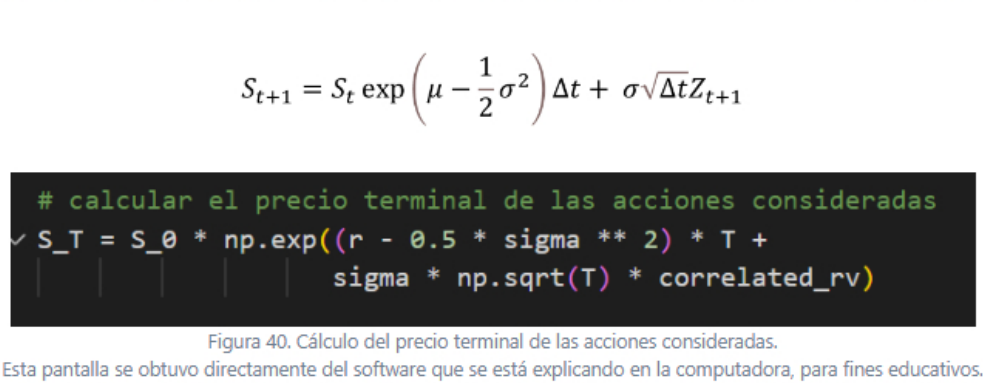


```
In [18]: P_0
Out[18]: 1298.1999588012695
```

Figura 39. Valores recientes de portafolio.
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Se aplica la solución analítica del movimiento browniano geométrico (GBM) con la ecuación diferencial estocástica para calcular el valor probable del precio de ambos activos con base en las simulaciones. El precio S del activo en el tiempo t+1 está en función del precio actual St, así como en el promedio de rendimientos diarios μ, su desviación estándar σ, el incremento de tiempo Δt y el valor de la distribución normal estándar S_t+1, expresado con la ecuación diferencial estocástica.

$$S_{t+1} = S_t \exp\left(\mu - \frac{1}{2}\sigma^2\right)\Delta t + \sigma\sqrt{\Delta t}Z_{t+1}$$



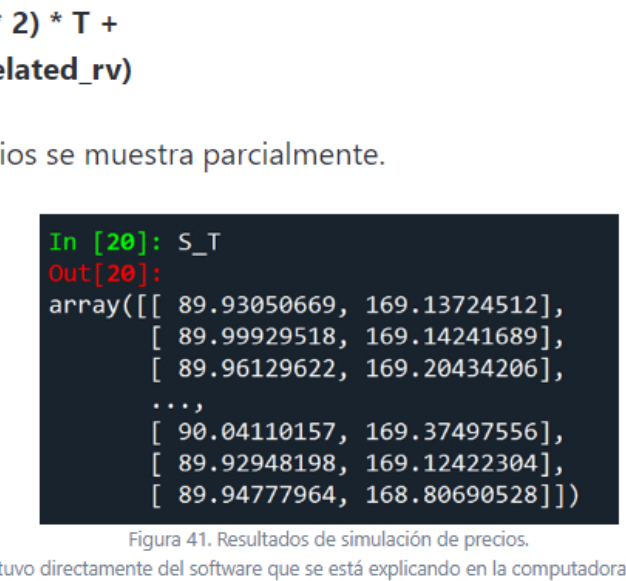
```
# calcular el precio terminal de las acciones consideradas
S_T = S_0 * np.exp((r - 0.5 * sigma ** 2) * T +
                   sigma * np.sqrt(T) * correlated_rv)
```

Figura 40. Cálculo del precio terminal de las acciones consideradas.
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Enseguida se incluye el texto para fines prácticos:

```
# calcular el precio terminal de las acciones consideradas
S_T = S_0 * np.exp((r - 0.5 * sigma ** 2) * T +
        sigma * np.sqrt(T) * correlated_rv)
```

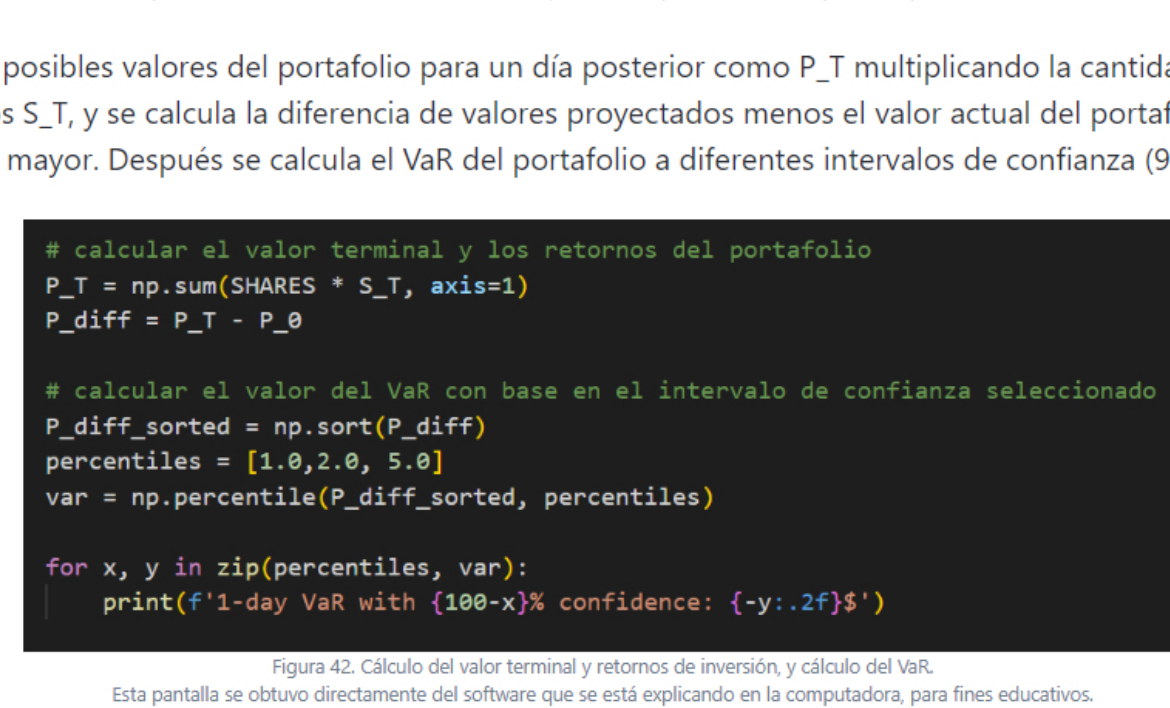
Y el resultado de la simulación de precios se muestra parcialmente.



```
In [20]: S_T
Out[20]:
array([[ 89.93050669, 169.13724512],
       [ 89.99929518, 169.14241689],
       [ 89.96129622, 169.20434206],
       ...,
       [ 90.04110157, 169.37497556],
       [ 89.92948198, 169.12422304],
       [ 89.94777964, 168.80690528]])
```

Figura 41. Resultados de simulación de precios.
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Se determinan los posibles valores del portafolio para un día posterior como P_T multiplicando la cantidad de acciones por los posibles precios S_T, y se calcula la diferencia de valores proyectados menos el valor actual del portafolio (P_T – P0) y se sortea de menor a mayor. Después se calcula el VaR del portafolio a diferentes intervalos de confianza (99%, 98% y 95%).



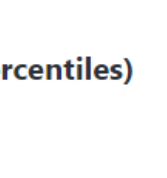
```
# calcular el valor terminal y los retornos del portafolio
P_T = np.sum(SHARES * S_T, axis=1)
P_diff = P_T - P_0

# calcular el valor del VaR con base en el intervalo de confianza seleccionado
P_diff_sorted = np.sort(P_diff)
percentiles = [1.0,2.0, 5.0]
var = np.percentile(P_diff_sorted, percentiles)

for x, y in zip(percentiles, var):
    print(f'1-day VaR with {100-x}% confidence: {-y:.2f}$')
```

Figura 42. Cálculo del valor terminal y retornos de inversión, y cálculo del VaR.
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Enseguida se incluye el texto para fines prácticos:

```
# calcular el valor terminal y los retornos del portafolio
P_T = np.sum(SHARES * S_T, axis=1)
P_diff = P_T - P_0

# calcular el valor del VaR con base en el intervalo de confianza seleccionado
P_diff_sorted = np.sort(P_diff)
percentiles = [1.0,2.0, 5.0]
var = np.percentile(P_diff_sorted, percentiles)

for x, y in zip(percentiles, var):
  print(f'1-day VaR with {100-x}% confidence: {-y:.2f}$')
```

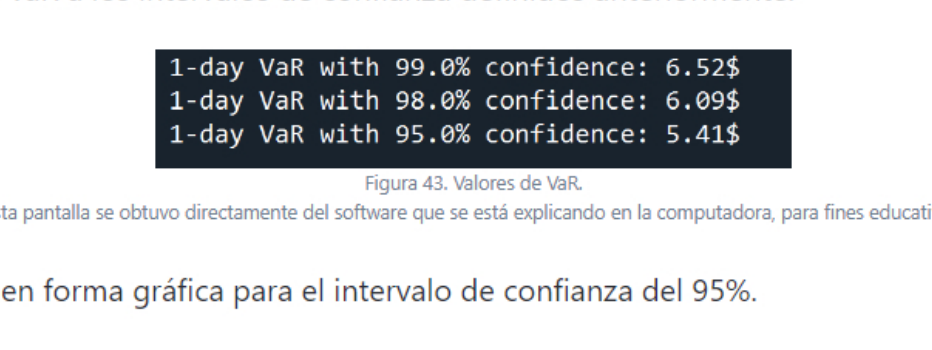
Y se obtienen los valores de VaR a los intervalos de confianza definidos anteriormente.



```
1-day VaR with 99.0% confidence: 6.52$
1-day VaR with 98.0% confidence: 6.09$
1-day VaR with 95.0% confidence: 5.41$
```

Figura 43. Valores de VaR.
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Se presentan los resultados en forma gráfica para el intervalo de confianza del 95%.



```
# presentar los resultados sobre una grafica
ax = sns.distplot(P_diff, kde=False)
ax.set_title('''Distribution of possible 1-day changes
             in portfolio value
             1-day 95% VaR''', fontsize=10)
ax.axvline(var[2], 0, 10000);
```

Figura 44. Presentación de resultados sobre una gráfica.
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Enseguida se incluye el texto para fines prácticos:

```
# presentar los resultados sobre una grafica
ax = sns.distplot(P_diff, kde=False)
ax.set_title('''Distribution of possible 1-day changes
      in portfolio value
      1-day 99% VaR''', fontsize=10)
ax.axvline(var[2], 0, 10000);
```

La siguiente figura muestra la distribución de los posibles valores del portafolio en 1 día. La línea vertical es el valor en riesgo (VaR), en este caso, $5.41.

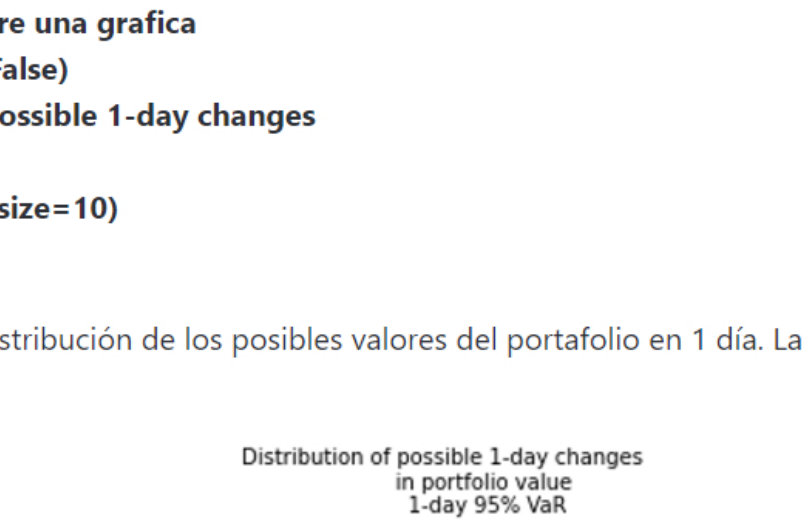


Figura 45. Distribución de los posibles valores del portafolio en un plazo de 1 día.
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

El valor de VaR es una importante referencia para comparar diferentes portafolios y estimar con anticipación el impacto en los rendimientos esperados.

Cierre

En este tema se abordaron diferentes herramientas para medir el riesgo de la inversión en activos financieros, apoyados con métodos de simulación Monte Carlo, para tratar de encontrar un balance entre riesgo y rendimiento, de acuerdo con el perfil del inversionista. En cuanto al riesgo, se sabe que el riesgo total tiene dos componentes: riesgo del mercado y riesgo de un activo especifico (Yan, 2017). En ambos casos, una vez que se identifica el comportamiento de sus fluctuaciones en periodos de tiempo, se utilizan diversas pruebas para ajustar ese comportamiento a una distribución de probabilidad, y con esto, se expanden las posibilidades de aplicaciones de la simulación para estimar posibles valores futuros que, a final de cuentas, ayudan a incrementar la precisión de las decisiones de inversión.

Desde el día que asistió el director de estrategia a la conferencia, la empresa La Mezcla Inteligente ha mejorado mucho en la selección de activos para incluirlos en sus portafolios y mayor prestigio entre el público inversionista.

Checkpoints

Asegúrate de:

- Comprender los conceptos y aplicaciones de la simulación en las finanzas.
- Comprender el concepto y los términos de la ecuación diferencial estocástica.
- Identificar las aplicaciones de la simulación Monte Carlos para estimar los precios de activos financieros.
- Calcular los premios de las opciones europeas y el cálculo del valor en riesgo de un activo o portafolio de inversiones (VaR).

Referencias

- Chen, J. (2022). *Exotic option: definition and comparation to traditional options.* Recuperado de https://www.investopedia.com/terms/e/exoticoption.asp
- Kenton, W. (2023). *Understanding value at risk (var) and how it´s computed.* Recuperado de https://www.investopedia.com/terms/v/var.asp
- Lewinson, E. (2020). *Python for finance cookbook.* Packt Publishing.
- Whiteside, J. (2008). *A practical application of Monte Carlo simulation in forecasting.* Recuperado de https://www.proquest.com/docview/208193177
- Yan, Y. (2017). *Python for finance* (2ª ed.). Packt Publishing.

Recursos requeridos

Los siguientes enlaces son externos a la Universidad Tecmilenio, al acceder a ellos considera que debes apegarte a sus términos y condiciones.

- Chen, J. (2022). *Exotic option: definition and comparation to traditional options.* Recuperado de https://www.investopedia.com/terms/e/exoticoption.asp
- Sánchez, G. (2020). *El método Monte Carlo.* Recuperado de https://gsnchez.com/blog/article/El-metodo-de-monte-carlo
- Trading con Opciones. (2022, 27 de mayo). *Opciones AMERICANAS o EUROPEAS❓ (NO es lo que PIENSAS❗)* [Archivo de video]. Recuperado de https://www.youtube.com/watch?v=mqr0KmF0YzQ

"Tecmilenio no guarda relación alguna con las marcas mencionadas como ejemplo. Las marcas son propiedad de sus titulares conforme a la legislación aplicable, estas se utilizan con fines académicos y didácticos, por lo que no existen fines de lucro, relación publicitaria o de patrocinio".

Cerrar