Anàlisi de Dades Financeres

Universitat Autònoma de Barcelona

Entrega 5

Informe

Abril Pérez Martí - 1600601

Arnau Perich Iglesias - 1603567

Eric Jiménez Barril - 1599092

Joan González Martínez - 1597201

Laia Escursell Rof - 1600578

13 de novembre del 2023

Exercici 1

Construirem un simulador de moviment brownià.

(a) **Camí de mostra de llançament de monedes**: creeu una funció que prengui un paràmetre N i produeixi un vector, que quan es dibuixa és el camí dels guanys i pèrdues d'un joc de llançaments. Tingueu en compte les regles següents:

- El joc realitza N apostes en l'interval de temps $[0, 1]$.
- El benefici o la pèrdua de cada aposta és $\frac{1}{\sqrt{N}}$

(i) Traceu un camí de mostra per als valors següents de $N = 5, 10, 50, 100, 10000$

Definim una funció que símula els beneficis o pèrdues al tirar N vegades una moneda. A continuació podem veure com queda la funció:

```
coin_tossing_path <- function(N) {
  increments <- sample(c(-1, 1), size = N, replace = TRUE)/sqrt(N)
  path <- c(0, cumsum(increments))
  time <- seq(0, 1, length.out = N+1)

  return(list(time=time, path=path))
}
```

La figura 1 mostra els camins generats i com els beneficis o pèrdues van variant al llarg de les tirades, per els diferents valors de N:

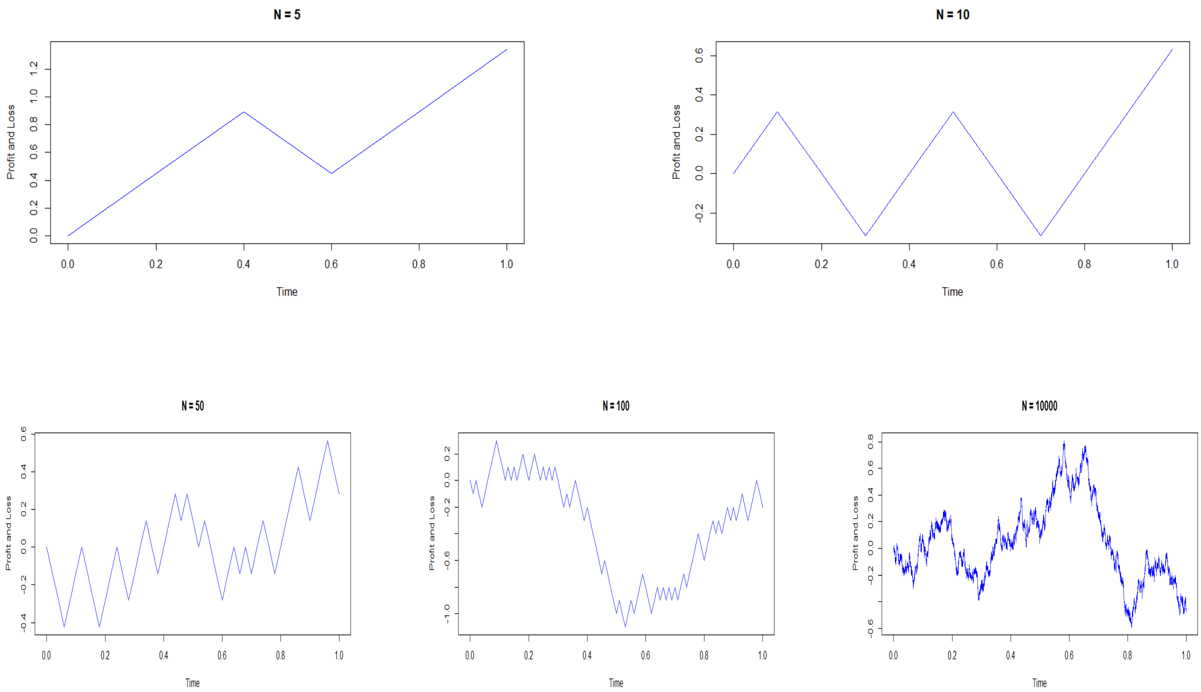

Figura 1: Plot del camí de mostra per diferents valors de N. Es pot veure com evoluciona el benefici i la pèrdua al llarg del temps

(b) **Distribució de camins de mostra**: Codifiqueu una funció que pren els valors N i m i crida m vegades la funció anterior amb el paràmetre N. Feu aquesta funció per mantenir l'últim valor per a cada camí de mostra.

(i) Crida la funció anterior per als parells $(N, m) = (100, 100), (1000, 1000), (10000, 10000)$. Traceu la funció de densitat de la sèrie temporal resultant i feu una prova de normalitat.

Definim la funció `simulate_multiple_paht` que s'encarrega d'anar cridant la funció `coin_tossing_paths` m vegades i guardar el benefici o pèrdua final de cada vegada.

```
simulate_multiple_paths <- function(N, m) {
  last_values <- numeric(m)

  for (i in 1:m) {
    result <- coin_tossing_path(N)
    last_values[i] <- tail(result$path, n=1)
  }

  return(last_values)
}
```

D'aquesta menera podem veure cap a on tendeixen els beneficis o pèrdues. A la figura 2, tendèixen a 0.

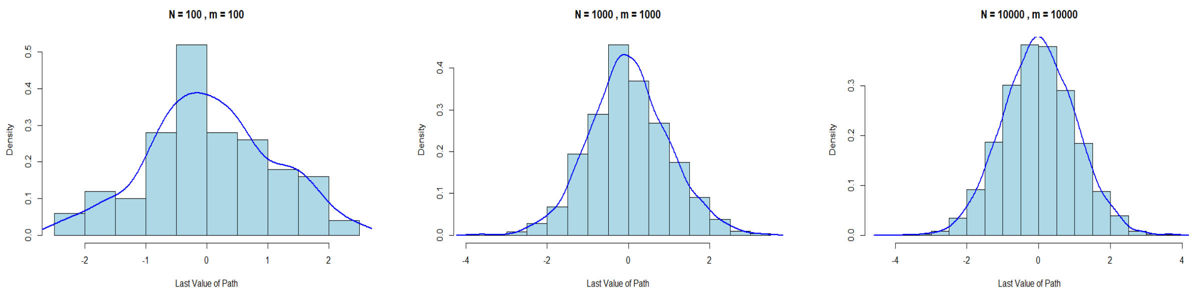

Figura 2: Plot de la funció de densitat per diferents valors de N i m.

Com podem apreciar a la Figura 2 a mesura que augmentem tant la quantitat del nombre d'apostes com les vegades que repetim el procés, la gràfica de densitat s'aproxima més a la gràfica d'una distribució normal. Aquest resultat és esperat, degut a que un moviment brownià té distribució normal i s'obté de fer el pas al límit a un procés discret com el estavem realitzant anteriorment. Per confirmar la nostra hipótesi, podem realitzar un *Shapiro-Wilk Normality Test* sobre cadascun dels procesos. Aquest prova la hipòtesi nul·la que la població donada està normalmente distribuida contra l'alternativa, que no ho està. A la Taula veiem els *p-valors* per les proves amb 100 i 1000 repeticions de processos amb 100 i 1000 apostes, respectivament.

(N, m)	**p-value**
(100,100)	0.5524
(1000,1000)	0.5829

Observem que en ambdós casos obtenim *p-valors* superiors a 0.05 per tant, no tenim evidències significatives per rebutjar que les poblacions siguin normals. Per tant, podem suposar que el resultat de les mostres són normals, tant per *N=m=100* com per *N=m=1000*

La prova del *Shapiro-Wilk Normality Test* test no admet poblacions de 10000 individus. És per això que a continuació es realitzen *qqplots* corresponents (veure Figura 3) per determinar si els quantils teòrics s'aproximen als quantils mostrals.

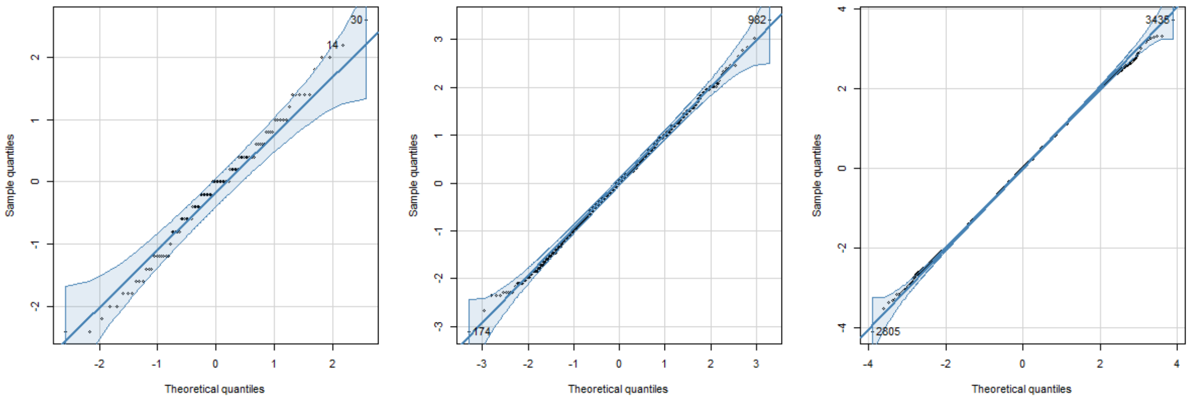

Figura 3: Gràfics dels *qqplots* per N = m = 100, 1000, 1000; respectivament

En els tres casos s'observa que els quantils mostrals es situen dins l'interval de confiança del 95%. Pero cal notar que, a mesura que s'agumenten els valors de N i m, millor s'ajusten a la recta i l'interval de confiança és més estret. Això ens dona indicis que la població tendeix a una distribució normal, tal i com voliem demostrar.

Exercici 2

Programarem un algorisme de Montecarlo per calcular els preus de les opcions.

(a) Construcció d'un camí de mostra sota el model de Black-Scholes: Escriu una funció que mostri un camí a partir de l'equació diferencial estocàstica de Black-Scholes:

$$dS_t = rS_t dt + \sigma S_t dW_t$$

usant una discretització d'Euler. La funció ha de prendre les següentsvariables:

- N: : nombre de passos de temps a la mostra.
- t_0: temps inicial expressat com una fracció d'any.
- t_n: temps d'expiració expressat com una fracció d'any.
- S_0: preu inicial de l'Stock.
- r: taxa sense risc anualitzada expressada en percentatge.
- σ: volatilitat anual expressada en percentatge.

Escriu una funció amb el següent prototip que retorni un camí del procés estocàstic.

```
path_sample=function(N,t0,tn,S0,r,sigma)
```

- Explorar les propietats del camí en funció de la variable r i σ.

El nostre objectiu és escriure una funció que mostri un camí a partir de l'equació diferencial estocàstica de Black-Scholes:

$$dS_t = rS_t dt + \sigma S_t dW_t \tag{1}$$

on S_t és el preu de l'actiu subjacent en temps t, r és la taxa del tipus d'interès, σ la volatilitat i W_t un Moviment Brownià.

Per tal de calcular el preu de l'actiu en cada instant de temps, usem la *discretització d'Euler*. Aquest procediment es basa en el següent.

Sigui una equació diferencial estocàstica de la forma

$$dX_t = a(X_t)dt + b(X_t)dW_t$$

per a algunes funcions $a(\cdot)$ i $b(\cdot)$. Podem simular el camí del procés a l'interval $[0, T]$ discretitzant-lo en N passos de mateix tamany

$$0 \leq t0 < t1 < ... < tn = T$$

i calculant el valor de cada X_{t_i} mitjançant l'algorisme:

$$\begin{cases} X_{t_0} = x_{t_0} \\ X_{t+\Delta T} = X_t + a(X_t)\Delta T + b(X_t)\sqrt{\Delta T}\,\varepsilon_t, \quad \varepsilon_t \sim N(0,1) \quad \Delta T = \frac{tn-t0}{N} \end{cases} \tag{2}$$

Per tant, usem l'equació 2 per simular el camí del preu de l'actiu, tenint en compte que $S_{t_0} = S_0$, $a(S) = rS$ i $b(S) = \sigma S$. El codi de `R` implementat es mostra a continuació

```
path_sample <- function(N, t0, tn, S0, r, sigma) {
  path <- numeric(N+1)
  path[1] = S0

  dt = (tn-t0)/N

  for (i in 2:(N+1)) {
    path[i] = path[i-1] + r*path[i-1]*dt + sigma*path[i-1]*sqrt(dt)*rnorm(1)
  }

  return(path)
}
```

Considerem l'exemple per $N = 1000$, $t_0 = 0$, $t_n = 1$, $S_0 = 100$, $r = 0.01$, $sigma = 0.3$. Si fixem `seed(1234)` i usem la funció `path_sample`, obtenim el camí que es mostra a la Figura 4.

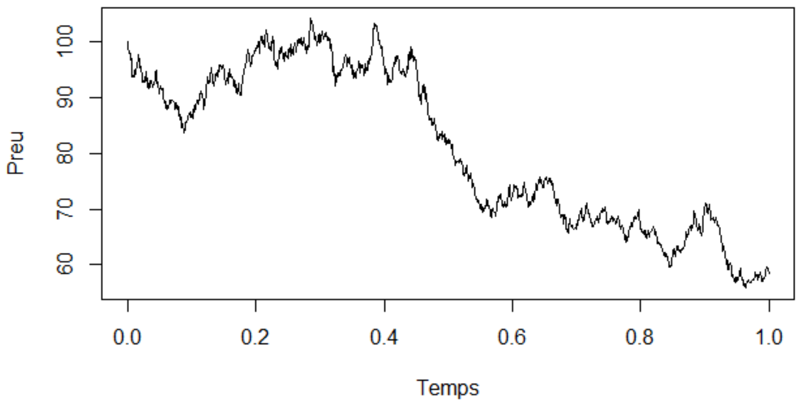

Figura 4: Simulació d'un camí del preu de l'actiu que segueix l'equació diferencial estocàstica de Black-Scholes 1.

Per últim, explorem les propietats del camí en funció de la variable r i σ.

Conseqüències de la variació de volatilitat

En primer lloc, mantenim constant el tipus d'interès r i fem variar el de la volatilitat σ. A la figura 5 s'observa una tendència creixent en la variabilitat dels valors que pren S a mesura que augmenta el valor de σ. Notem com, per valors petits de σ, la variació màxima mitjana del preu de l'actiu és relativament petita (no supera una oscil·lació màxima major a 50 per $\sigma < 0.25$). Per altra banda, per valors grans de σ, aquesta variació esdevé significant (arriba a superar una oscil·lació màxima major a 150 per alguns valors $\sigma > 0.75$).

Per tant, podem concloure que a major volatilitat, més ampli és el rang de valors que pot pendre l'actiu.

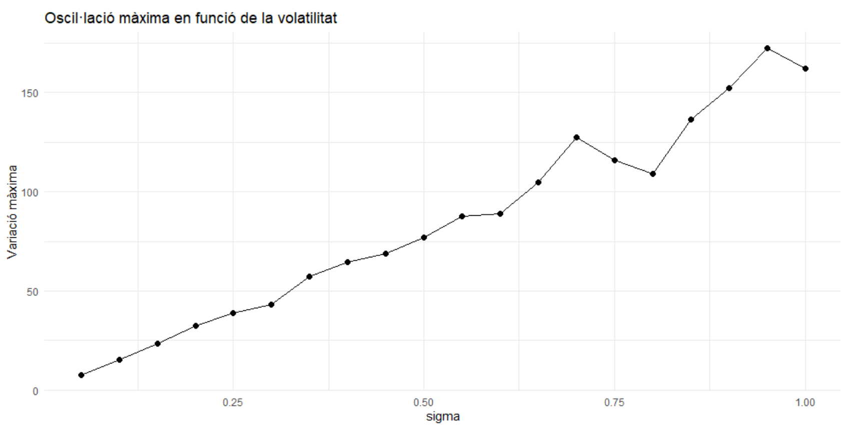

Figura 5: Variació màxima mitjana del preu d'un actiu en funció del valor de σ. Per cada σ, es simulen 100 camins, es calcula la variació màxima del preu de cada camí i es fa la mitjana

Conseqüències de la variació del tipus d'interès

A continuació, mantenim constant la volatilitat σ i fem variar el tipus d'interés r. En aquest cas, el fenomen que s'observa al variar r és un augment de la tendència creixent del camí del preu de l'actiu. A la figura 6 es pot veure la diferència entre el valor final i inicial de l'actiu. Aquest valor ens mostra, de manera implícita, si el camí ha seguit una tendència constant (el preu no ha variat) o una tendència creixent (el preu final és major que el preu inicial). Notem que per valors petits de r, aquesta diferència és significativament petita; és a dir que el preu ha romàs constant. Per altra banda, a mesura que el valor de r augmenta, aquesta diferència també esdevé més gran. És a dir que la tendència del camí és creixent (el preu final és major que l'inicial).

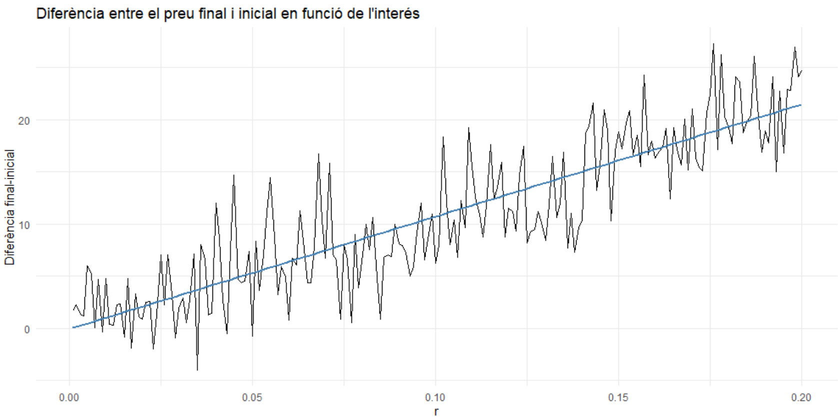

Figura 6: Diferència entre el preu final i inicial de l'actiu en funció del tipus d'interés. Per cada r, es simulen 100 camins, es calcula la diferència entre el valor final i l'inicial ($S_0 - S_T$) i es fa la mitjana.

(b) Funció de pagament: Implementeu la funció següent que avalua un benefici per a una *Call-option* amb una *strike* K determinada.

Donada que la funció de pagament d'un *Call-option* amb strike K és

$$\max(S - K, 0)$$

s'implementa la següent funció a R:

```
payoff_function=function(S,K){
    return(max(S-K,0))
}
```

(c) Algorisme de Monte Carlo: Implementeu un algorisme de Montecarlo tal com es descriu a les notes de la conferència. La funció hauria de prendre un paràmetre M que determini el nombre de camins de mostra, així com una funció que determini el benefici de l'opció. També hauria de prendre tots els paràmetres de la funció *path sample* per poder cridar-lo dins del procediment. Utilitzeu el següent prototip:

```
monte_carlo(M,N,t0,tn,S0,r,sigma,payoff_function,K)
```

Recordem, en primer lloc, l'algorisme de Monte Carlo pel preu d'una opció.

Algorisme de Monte Carlo:

1. Simular una ruta per a S aplicant l'esquema d'Euler (13) amb
   - $S_{t_0} = S_0$
   - $a(S) = rS$
   - $b(S) = \sigma S$

per a una malla de discretització donada $N \in \mathbb{N}$

2. Apliqueu la funció de pagament, V, a S_{t_N}
3. Repetiu els passos 1 i 2 $M \in \mathbb{N}$ vegades
4. El preu de l'opció és

$$\left(\frac{1}{M}\sum_{i=1}^{M} V\left(S_N^i\right)\right) e^{-r(T-t)} \quad \text{quan } N, M \to \infty$$

A continuació s'implementa aquest algorisme a `R` pel nostre cas.

```
monte_carlo=function(M,N,t0,tn,S0,r,sigma,payoff_function,K){
  v <- c()
  for(i in 1:M){
    path<-path_sample(N,t0,tn,S0,r,sigma)
    v[i]<-payoff_function(path[N+1],K)
    }
  preu = mean(v)*exp(-r*(tn-t0))
}
```

La funció `monte_carlo` realitza el següent: es repeteix M cops la simulació d'un camí usant la funció `path_sample` de l'apartat (a), es calcula el preu de l'opció en base a l'últim valor del camí, amb la funció de pagament prèviament definida, i finalment s'aplica l'última fórmula per saber el preu de l'acció.

(d) ***Put option*** Modifica la funció `payoff_function` i dona el preu d'una *Put-option* per 1 any, amb un 5% de taxa d'interés i un 40% de volatilitat. L'*stock* cotitza actualment a 90 EUR i l'*strike* de la *Put-option* és de 75.

En primer lloc, recordem que la funció de pagament d'una *Put-option* sense prima és

$$\max(K - S_T, 0)$$

Un cop definida la `payoff_function`, i fixant

- $r = 0.05$.
- $\sigma = 0.4$.
- $S_0 = 90$.
- $K = 75$.
- N=1000.
- M=1000.
- $t_0 = 0$, $t_n = 1$, doncs hi ha 1 any.

obtenim que el preu de la *Put-option* al moment inicial és de 5.89EUR.

Per tal de seguir el model de Monte-Carlo, podem cridar més cops la funció i calcular la mitjana. Si ho fem amb 10 repeticions, obtenim un preu de 5.60EUR, mentre que si ho fem amb 100 repeticions, obtenim que el preu de la *Put-option* és de 5.55EUR