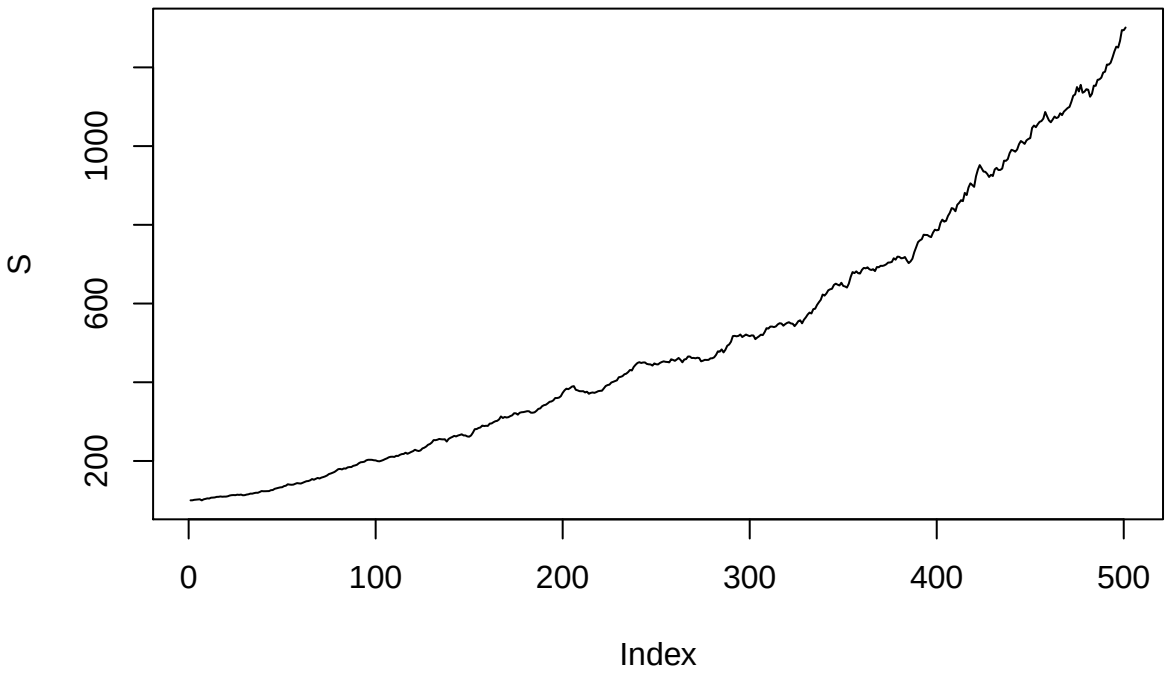
TP 8 : Modèle de Black-Scholes discret, application au cours du Bitcoin

Première partie : Un modèle probabiliste pour le prix d'un actif.

1. Réponse : il y a la tendance b, la volatilité σ, et l'initialisation S_0 qui sont des paramètres.

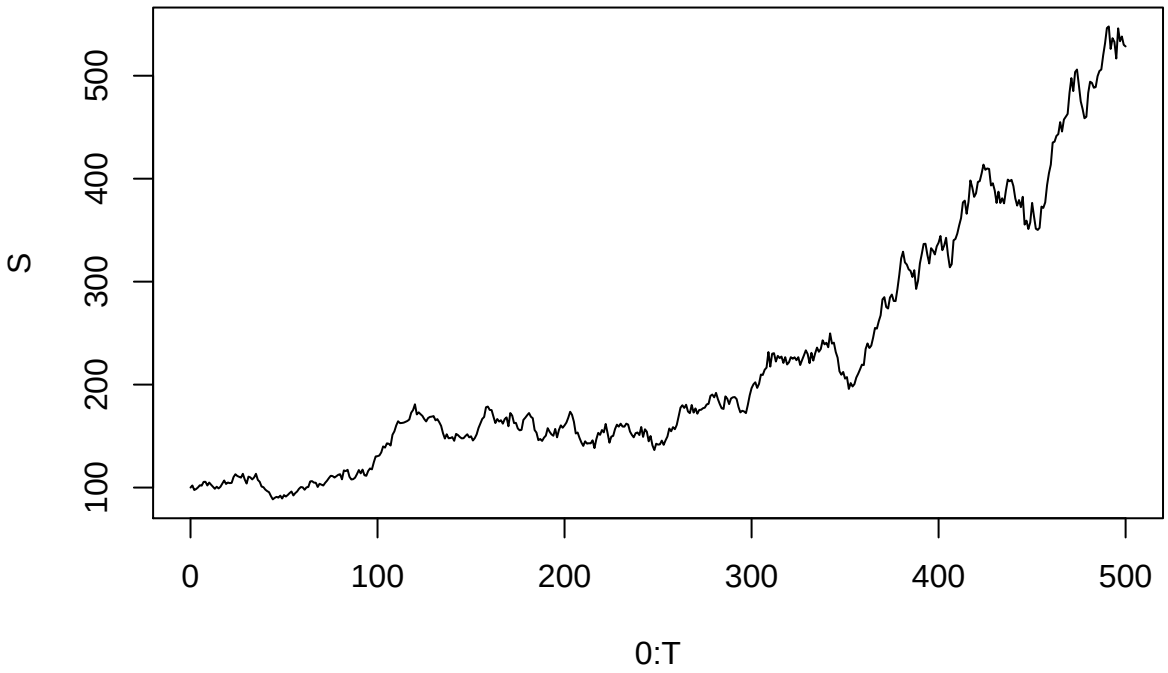
```
traj = function(T,S0,b,sigma){
  S = replicate(T+1,0) # on initialise avec T+1 cases dans le vecteur S
  Z = rnorm(T,mean=0,sd=1) # on simule les variables gaussiennes
  S[1] = S0 # on rentre le premier terme
  for (t in 2:(T+1)){
    S[t] = S[t-1] + S[t-1]*(b+sigma*Z[t-1]) # on écrit la récurrence
  }
  S # on renvoie le vecteur des prix
}

T=500
S = traj(T,S0=100,b=0.005,sigma=0.01)
plot(S,type="l")
```



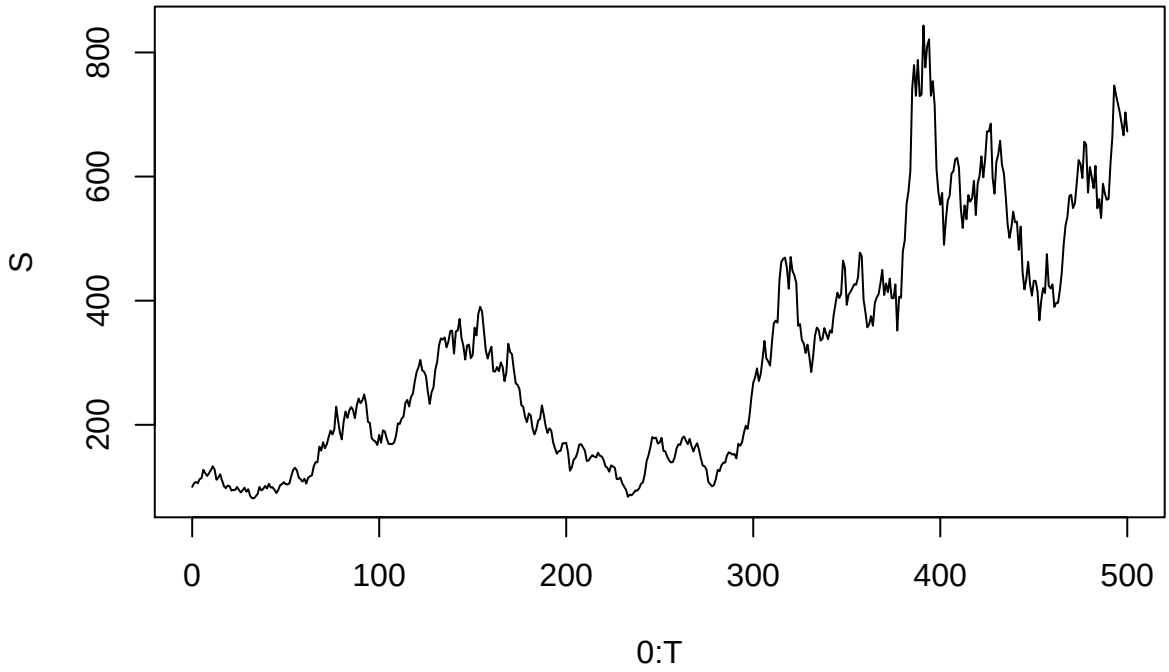
2. Réponse : le prix de l'actif augmente nettement, sans beaucoup de volatilité. Le paramètre $b = 0.005$ est quand même assez ambitieux (pour un pas de temps d'un jour).

```
S = traj(T,S0=100,b=0.005,sigma=0.03)
plot(0:T,S,type="l")
```

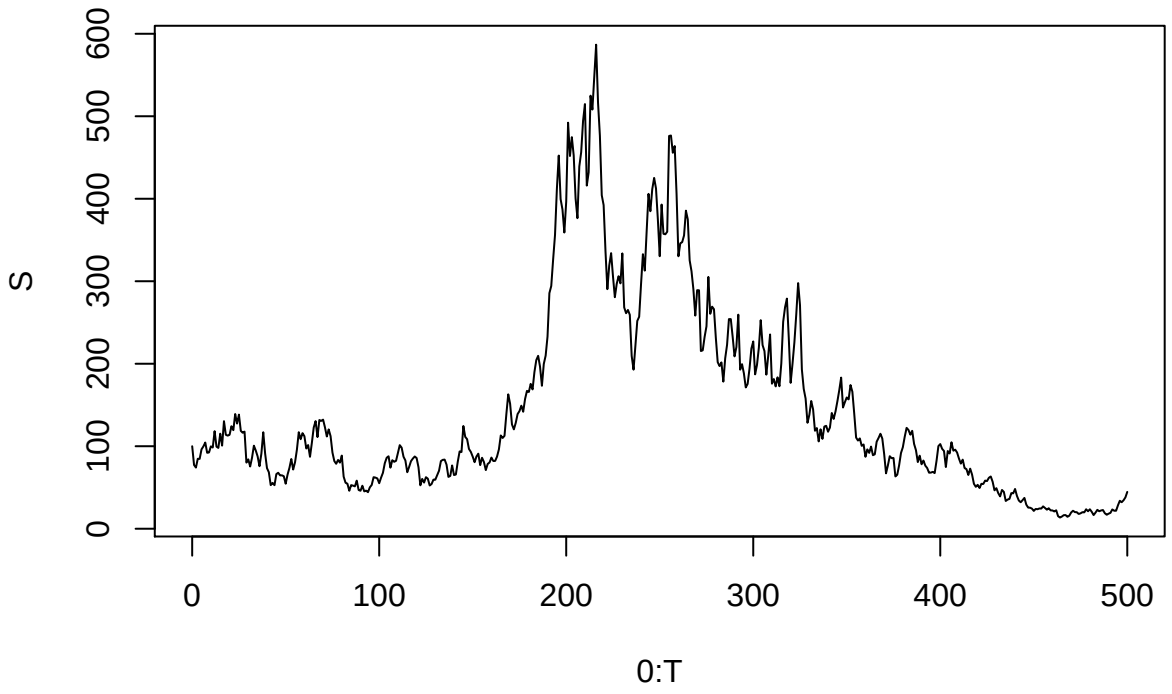


3.

```
S = traj(T,S0=100,b=0.005,sigma=0.07)
plot(0:T,S,type="l")
```

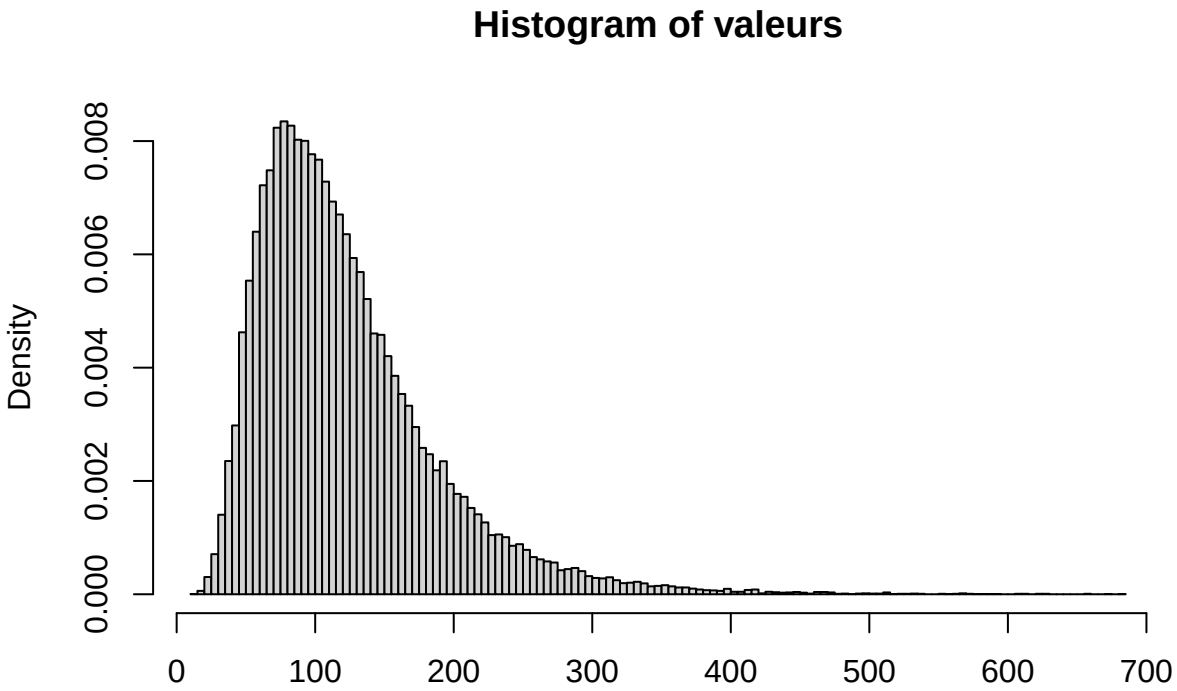



```
S = traj(T,S0=100,b=0.005,sigma=0.12)
plot(0:T,S,type="l")
```



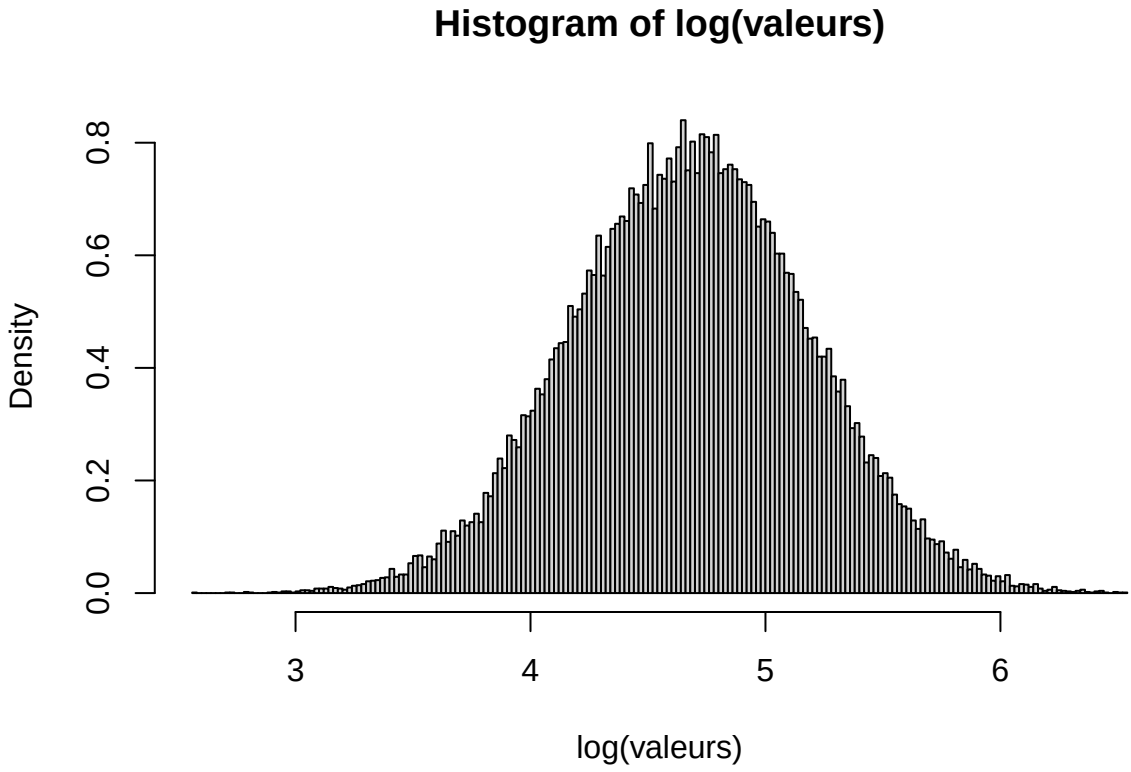
Réponse : quand la volatilité augmente, le cours de l'actif s'éloigne nettement de la tendance. En fait, on peut conjecturer que l'évolution globale dépend de b mais AUSSI de σ.

```
N=50000
valeurs = replicate(N,0) # on initialise le vecteur
for (i in 1:N){
  valeurs[i] = traj(T=100,S0=100,b=0.002,sigma=0.05)[101] # on ne regarde que la
  # dernière valeur : 101 car il y a T+1 valeurs dans S
}
hist(valeurs, breaks=200,freq=FALSE)
```



4.

```
hist(log(valeurs), breaks=200,freq=FALSE)
```



Réponse : le log du prix final semble suivre une loi normale dont la moyenne croît avec b et décroît avec σ.

```r
N=50000
valeurs = replicate(N,0)
for (i in 1:N){
  valeurs[i] = traj(T=100,S0=100,b=0.0,sigma=0.15)[101] # on ne regarde que la dernière valeur
}

length(which(valeurs <= 50))/N # diminué d'au moins 50%
```

5.

```
## [1] 0.61402
```

```r
length(which(valeurs >= 150))/N # augmenté d'au moins 50%
```

```
## [1] 0.1571
```

```r
length(which(valeurs >= 200))/N # au moins doublé
```

```
## [1] 0.1153
```

```r
logV = function(b,sigma,S){
  T = length(S)-1
  Splus = S[2:(T+1)] # valeurs des S_{t+1}
  Smoins = S[1:T] # valeurs des S_{t} : ainsi on n'a pas besoin de boucle for et
  # on utilise la vectorialisation sous R
  -(T/2)*log(2*pi*sigma^2) - sum(((Splus-Smoins)/Smoins - b)^2)/(2*sigma^2)
}
```

6.

7. Réponse : il suffit juste de remarquer que le vecteur des $V_t := \frac{S_t - S_{t-1}}{S_{t-1}}$ est un vecteur de Gaussiennes indépendantes de moyenne b et de variance σ^2. On peut donc calculer la moyenne empirique de V et la variance emprique (sans biais) de V pour accéder à des estimateurs de b et de σ^2.

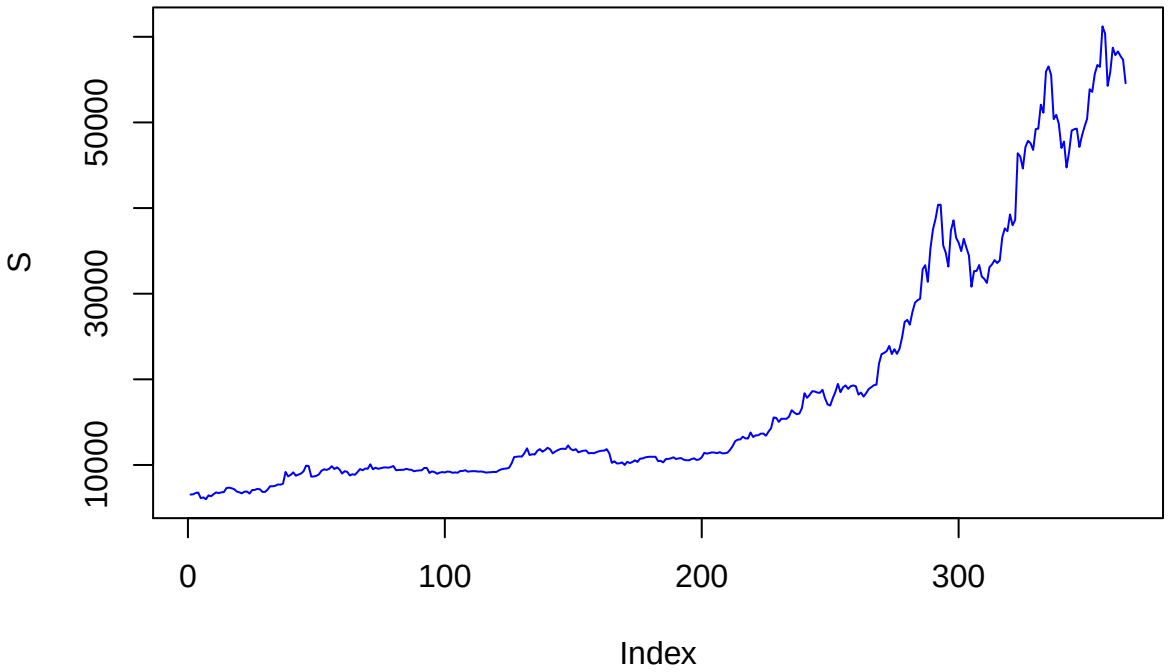
```r
data = read.csv("bitcoin.csv")
head(data)
```

Deuxième partie : application à l'analyse et à la prédiction du cours du Bitcoin.

```
##         Date    Last Volume    Open    High     Low
## 1 03/22/2021 54577.0    N/A 57558.1 55287.5 53842.1
## 2 03/21/2021 57316.4    N/A 57809.8 57443.4 56357.5
## 3 03/20/2021 57754.5    N/A 58419.7 58583.8 57471.4
## 4 03/19/2021 58296.2    N/A 57773.0 58877.5 57863.0
## 5 03/18/2021 57859.3    N/A 58724.5 57876.5 56305.1
## 6 03/17/2021 58727.6    N/A 55489.5 59489.4 58621.9
```

Ce dataframe retrace l'évolution du prix d'un Bitcoin (en dollars) depuis une année. Il a plusieurs colonnes : la date, le prix à la fin de la journée (c'est celui qui nous intéresse dans notre étude), le prix en début de journée, ainsi que les valeurs extrêmes sur 24h.

```r
S = rev(data$Last) # le jeu de données
plot(S,type="l",main="Cours du Bitcoin sur les derniers 365 jours",col="blue")
```



7.

8.

```
T = length(S)-1
Splus = S[2:(T+1)]
Smoins = S[1:T]
V = (Splus-Smoins)/Smoins
hatb =  mean(V)
hatsigma = sd(V)
hatb
```

```
## [1] 0.00651585
```

```
hatsigma
```

```
## [1] 0.03704547
```

Commentaires : le modèle nous renvoie $b \sim 0.007$, soit une évolution moyenne quotienne de $+0,7\%$. Toutefois, σ est assez élevé ($\sigma \sim 0.04$), et de l'ordre de 5 fois le biais, ce qui rend le cours très volatile, et l'actif très risqué.

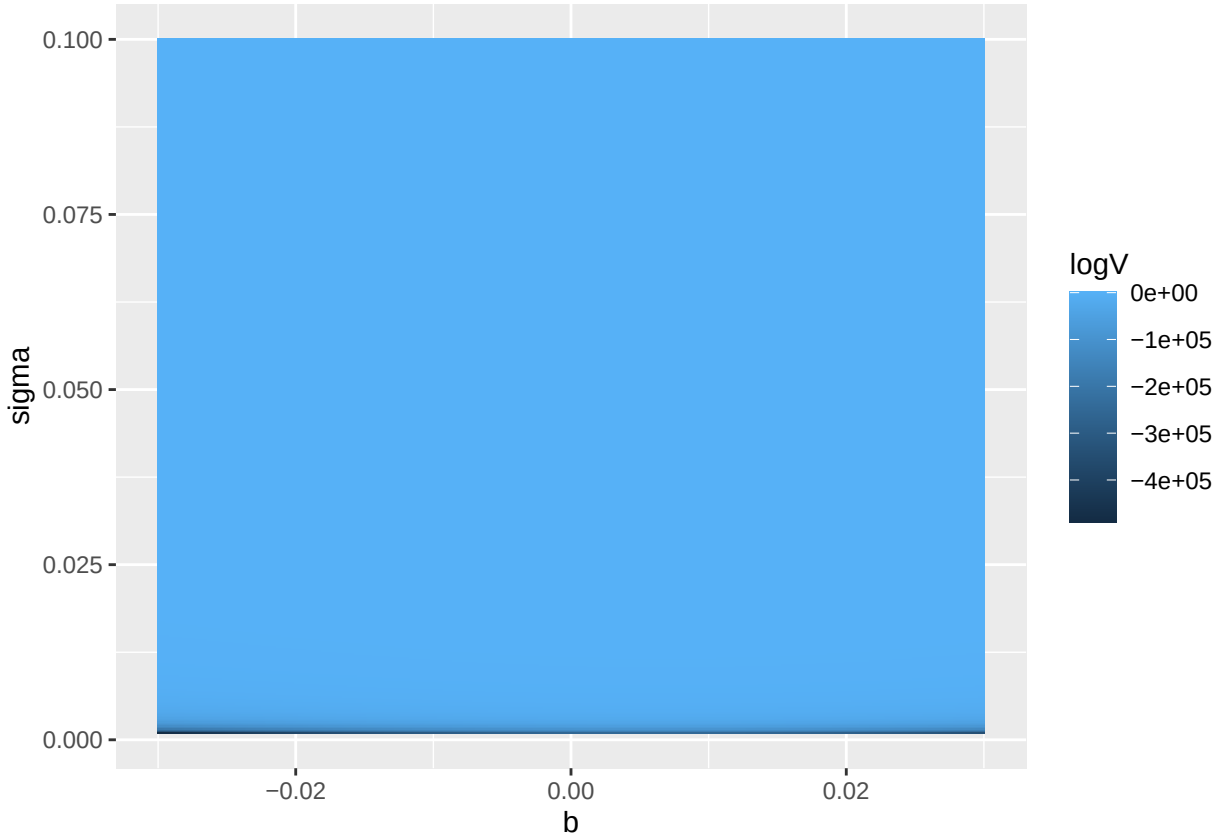
```
size = 500
bs = seq(-0.03,0.03,length.out=size) # le paramètre length.out donne le nombre
# d'éléments dans le subdivision de l'intervalle.
sigmas = seq(0.001,0.1,length.out=size)

logVs = matrix(data = replicate(size*size,0),nrow=size,ncol=size) # on crée une
# matrice de taille size*size ne contenant que des zéros
for (i in 1:size){
  b = bs[i]
  for (j in 1:size){
    sigma = sigmas[j]
```

```
    logVs[i,j] = logV(b,sigma,S) #on utilise la fonction précedemment définie
  }
}
```

9.

```
library(ggplot2)
data = expand.grid(b=bs,sigma=sigmas) # on crée une matrice contenant les
# valeurs de b croisées avec celles de sigma.
data$logV = matrix(logVs,ncol = 1,nrow=size*size) # on ajoute à ce dataframe
# un champ logV qui correspond aux logvraisemblances précedemment calculées
ggplot(data, aes(b, sigma, fill= logV)) + geom_tile() # on dessine le heatmap
```



10.

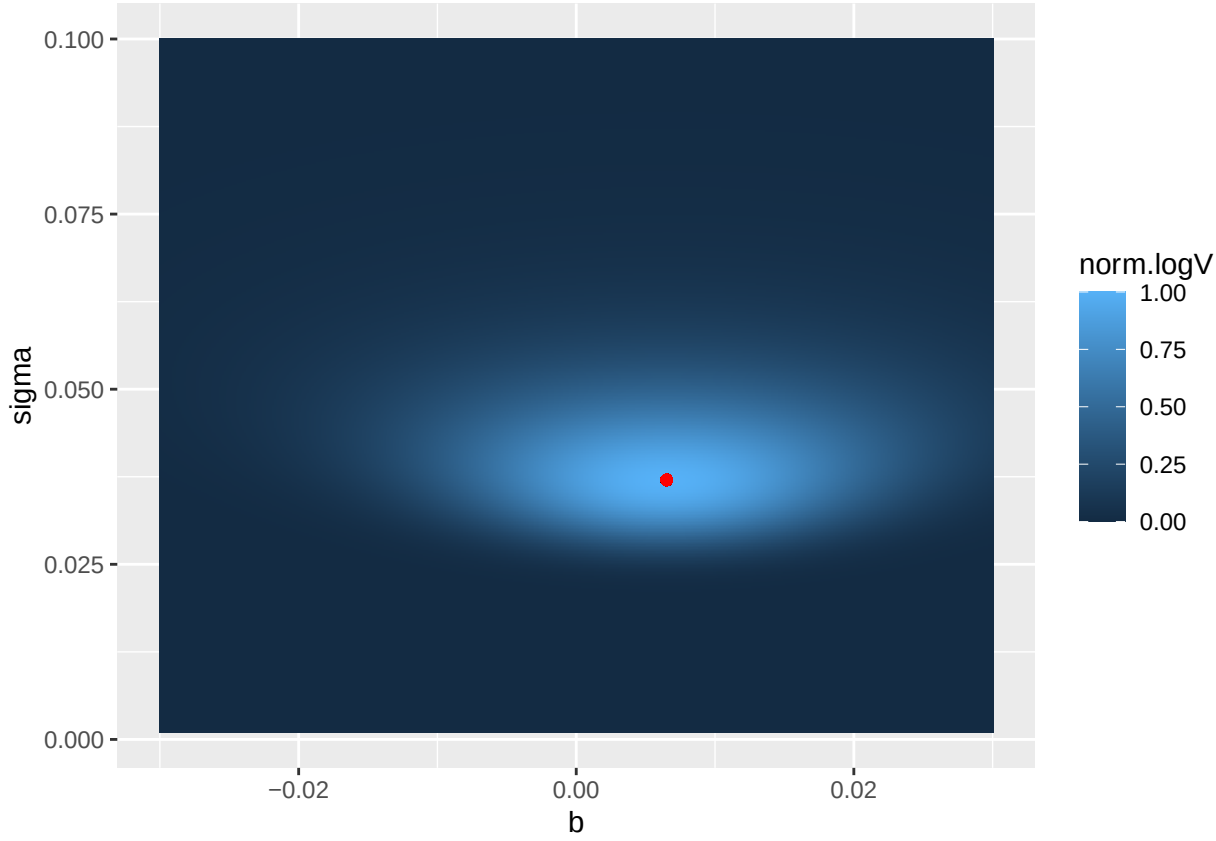
Réponse : on ne peut pas y voir clair, car la variation de la log vraisemblance est trop violente : tout quasiment apparait en bleu clair, alors que les valeurs sont de l'ordre de -10000 ou de 300...

11. Pour y voir plus clair, on opère une renormalisation de la log-vraisemblance.

```
exposant = 20
norm.logV = pmax(0,logVs/max(logVs))^exposant
```

Réponse : on enlève les valeurs trop négatives en le reportant à 0, puis on renormalise les valeurs positives pour qu'elles soient entre 0 et 1. L'effet de l'exposant est de faire chuter les valeurs trop loin du maximum de vraisemblance (on aura par exemple 0.7^8 qui sera tout petit). Ne vont rester que les valeurs de vraisemblance qui sont très proches du maximum. Cette renormalisation est d'autant plus violente que l'exposant est grand.

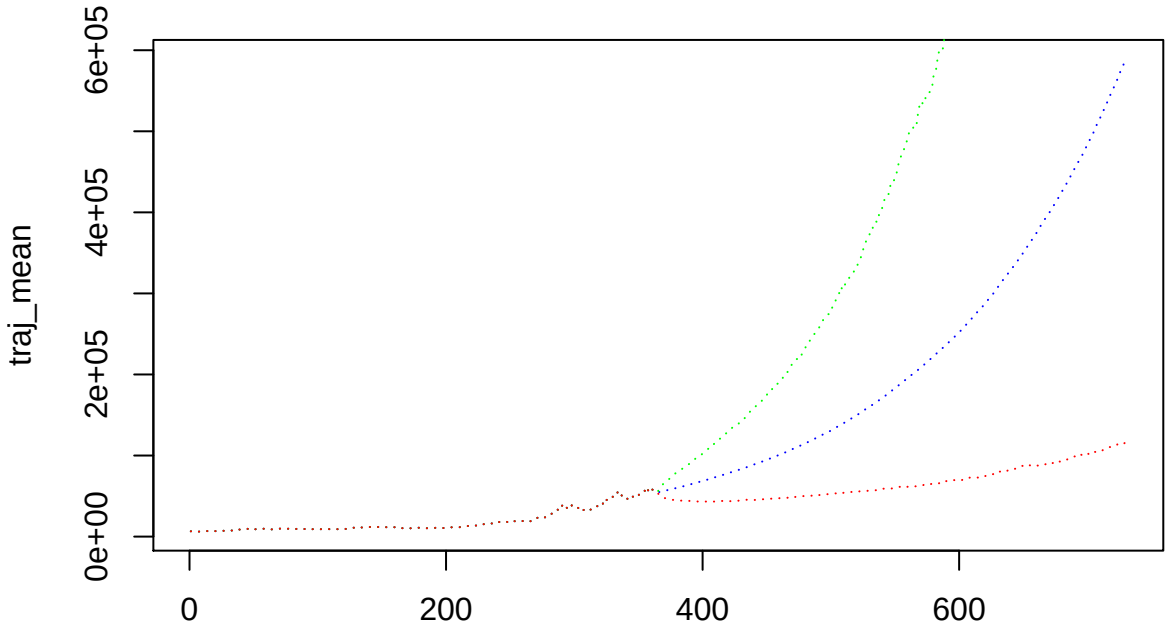
```
data = expand.grid(b=bs,sigma=sigmas)
data$norm.logV = matrix(norm.logV,ncol = 1,nrow=size*size)
ggplot(data, aes(b, sigma, fill= norm.logV)) + geom_tile() + geom_point(aes(x=hatb, y=hatsigma), colour=
```




```
N = 5000
S365 = S[365] # la valeur de S avec laquelle on intialise nos simulations
b = hatb # les valeurs des paramètres estimés
sigma = hatsigma
T=365+365 # temps non pas d'un an, mais deux
trajectoires = matrix(replicate(N*T,0),nrow = N, ncol = T)
for (i in 1:N){
  trajectoires[i,1:365] = S
  trajectoires[i,365:T] = traj(365,S0=S365,b,sigma)
}
```

12.

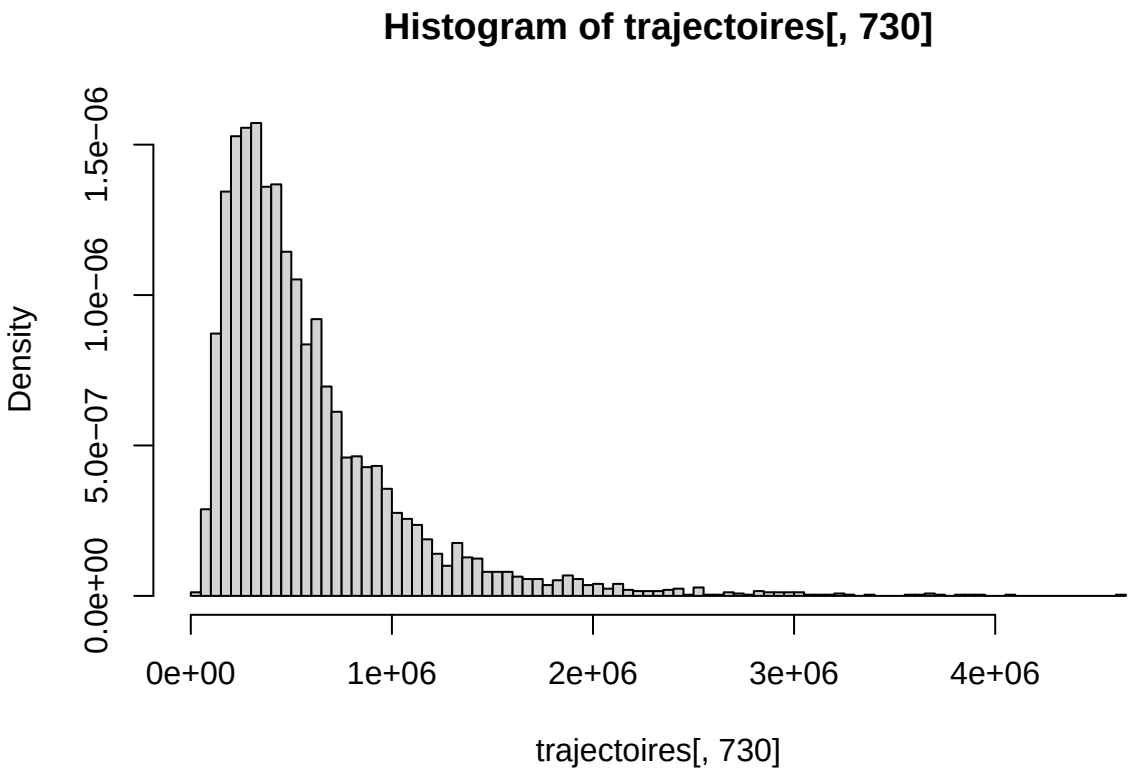
```
T=365+365
x=1:T
traj_inf = apply(trajectoires,2,quantile,0.025) # prendre la trajectoire inférieure
traj_sup = apply(trajectoires,2,quantile,0.975) # prendre la trajectoire supérieure (avec une certaine
traj_mean = colMeans(trajectoires)
plot(x,traj_mean,col="blue",type="l",lty=3) #prédiction moyenne
lines(x, traj_sup, col="green", lty = 3) #prédiction optimiste
lines(x, traj_inf, col="red", lty = 3) #prédiction pessimiste
```



13.

Commentaires : la prédiction semble indiquer une claire tendance à la hausse, et ce même dans un intervalle de confiance à 95%. On pourrait donc être tenté d'acheter.

```
hist(trajectoires[,730],freq=FALSE,breaks=150)
```



14.

```
mean(trajectoires[,730])
```

```
## [1] 589365.2
```

```
sd(trajectoires[,730])
```

```
## [1] 475027.4
```

```
length(which(trajectoires[,730]>=500000))/N
```

15.

```
## [1] 0.4478
```

Réponse : la probabilité n'est que d'environ 0.45, donc... pas plutôt non.

16. Réponse : il y a énormément d'hypothèses faites dans ce modèle. La plus grosse d'entre elles est que le volatilité et la tendance sont constantes dans le temps. En pratique, cela n'est pas vérifié sur de longues périodes. De plus, l'estimation de ces paramètres est inparfaite et comme le montre la question 11, nombreux sont les paramètres candidats pour bien fitter le modèle (y compris des cas où $b < 0$). Ce qui est certain en revanche, c'est que le Bitcoin est un actif risqué !