Volatilidades

Agustin Muñoz Gonzalez

7/7/2020

Preparamos el entorno.

```
rm(list=ls())
library(ggplot2)
```

Preliminares:

Recordemos que la fórmula del retorno del i-ésimo día es

$$R_i = \frac{S_i - S_{i-1}}{S_{i-1}}.$$

Defino una función para calcular los retornos diarios (o mensuales, anuales, etc, dependiendo del tiempo de nuestros datos) de nuestros datos.

```
# La siguiente función devuelve el retorno diario (o mensual)
# dependiendo de los datos price del data.frame
# Asume que los datos tienen una categoría Price y una categoría Año
retornos_año=function(datos,año){
  retornos=c()
  for(i in 2:length(datos$Price[datos$Año==año])){
    R_i=(datos$Price[datos$Año==año][i]-
           datos$Price[datos$Año==año][i-1])/
      datos$Price[datos$Año==año][i-1]
    retornos=c(retornos,R_i)
  }
  retornos
}
```

Vamos a definir y probar los distintos métodos del Capítulo 9 del Wilmott. Primero los definimos y al final los probamos.

Constant volatility/Moving window:.

Si suponemos que la volatilidad es constante o que varía muy lentamente, y tenemos datos de N días (rolling window of N returns), entonces la fórmula del método de volatilidad constante (**sin anualizar**) es

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N} R_i^2.$$

El problema con esta medida de volatilidad es que todos los retornos tienen el mismo peso, entonces podríamos tener saltos en la volatilidad si por ejemplo un día tenemos un retorno muy alto.

```
# Definimos el método de volatilidad cte a partir de los retornos
vol_cte=function(retornos){
  sqrt(mean(retornos^2))
}
```

Incorporating mean reversion:

Consideremos ahora una volatilidad que varía con el tiempo. No tenemos una única σ sino que tenemos la estimación de la volatilidad del n-ésimo día, σ_n, que hallamos usando datos de los días anteriores. Si creemos que la volatilidad tiende a variar alrededor de cierta media a largo plazo $\overline{\sigma}$, entonces podemos usar la fórmula siguiente

$$\sigma_n^2 = \alpha\overline{\sigma}^2 + (1-\alpha)\frac{1}{n}\sum_{i=1}^{n} R_i^2.$$

Notar que hay un peso asignado a cada estimación de volatilidad a largo plazo y a la estimación de la volatilidad actual, basados en los últimos n retornos.

Este modelo se llama ARCH model, por Autoregressive Conditional Heteroscedasticity.

```
vol_ARCH=function(retornos,n,alpha,sigma_medio){
  sqrt(alpha*sigma_medio^2+(1-alpha)*mean(retornos[1:n]^2))
}
```

Exponentially weighted estimate:

Consideremos ahora la siguiente estimación de la volatilidad

$$\sigma_n^2 = (1-\lambda)\sum_{i=1}^{\infty}\lambda^{i-1}R_{n-i+1}^2 \underset{\substack{\downarrow \\ n-i+1\geq 1}}{=} (1-\lambda)\sum_{i=1}^{n}\lambda^i R_{n-i}^2 = (1-\lambda)\sum_{i=0}^{n-1}\lambda^i R_{n-i}^2,$$

donde $0 < \lambda < 1$ debe ser. Notar que cuanto mas reciente es el retorno, más peso tiene. El coeficiente $1-\lambda$ asegura que todos los pesos sumen 1. La suma se extiende hasta el principio del tiempo.

La expresión puede ser simplificada a

$$\sigma_n^2 = \lambda\sigma_{n-1}^2 + (1-\lambda)R_n^2.$$

```
# Definimos el método de Exponentially weighted moving average
# Supongo que retornos son los retornos diarios de todos los años
# i.e. los retornos diarios históricos, no divido por año.
# modo=='media' devuelvo la media de las vol
# modo=='vol' devuelvo la tira de volatilidades, con \vol\=\retornos\
vol_EWMA=function(retornos,lambda,modo){
  vol=c()
  n=length(retornos)
  for(i in 1:n){
    # una forma
    vol[i]=sqrt((1-lambda)*sum(retornos[1:i]^2*lambda^((i-1):0)))
    # otra
    # vol[i]=(1-lambda)*sum(retornos[n:1]^2*lambda^(i))
  }
  # devuelvo el promedio de las volatilidades calculadas
  if(modo=='media'){mean(vol)}else if(modo=='vol'){vol}else{'Ingrese el modo'}
}
```

ARREGLARLA PARA QUE TOME n COMO PARÁMETRO Y REALICE LA SUMA COMPLETA SOBRE EL RETORNO.

A simple GARCH model:

Si ponemos los modelos anteriores juntos obtenemos

$$\sigma_n^2 = \alpha\overline{\sigma}^2 + (1-\alpha)\left(\lambda\sigma_{n-1}^2 + (1-\lambda)R_n^2\right) = \alpha\overline{\sigma}^2 + (1-\alpha)(1-\lambda)\sum_{i=0}^{n-1}\lambda^i R_{n-i}^2.$$

Este modelos se llama Generalized Autoregressive Conditional Heteroscedasticity o GARCH model.

```
vol_GARCH=function(retornos,n,alpha,lambda,sigma_medio){
  # primero creo un vector de lambdas para la sumatoria
  sqrt(alpha*sigma_medio^2+
         (1-alpha)*(1-lambda)*sum(retornos[n:1]^2*lambda^(0:(n-1))))
}
```

Traditional close-to-close measure:

vol_cc se usa cuando el drift es chico; vol_acc está ajustado por el drift

$$\sigma_{cc}^2 = \frac{1}{n}\sum_{i=1}^{n}\left(\log\left(\frac{C_i}{C_{i-1}}\right)\right)^2 = \frac{1}{n}\sum_{i=1}^{n}\left(\log(C_i)-\log(C_{i-1})\right)^2;$$

$$\sigma_{acc}^2 = \frac{1}{n-1}\sum_{i=1}^{n}\left(\left(\log\left(\frac{C_i}{C_{i-1}}\right)\right)^2 - \frac{\left(\log\left(\frac{C_n}{C_0}\right)\right)^2}{n(n-1)}\right) = \frac{1}{n-1}\left(\left(\sum_{i=1}^{n}\left(\log(C_i)-\log(C_{i-1})\right)^2\right) - \frac{\left(\log(C_n)-\log(C_0)\right)^2}{n(n-1)}\right),$$

donde C_i es el precio de cierre del i-ésimo día.

Cabe decir que mis datos arrancan en $i = 1$ por lo que tomo por ej

$$\frac{1}{n}\sum_{i=2}^{n}\left(\log\left(\frac{C_i}{C_{i-1}}\right)\right)^2.$$

```
vol_cc=function(datos_C,n){
  # m=length(datos_C)
  # n <= length(datos_C)
  sqrt(mean((log(datos_C[2:n])-log(datos_C[1:(n-1)]))^2))
}
vol_acc=function(datos_C,n){
  sqrt(
    (1/(n-1)) *
      ( sum( ( log(datos_C[2:n])-log(datos_C[1:(n-1)]) )^2 )-
        ( log(datos_C[n])-log(datos_C[1]) )^2/(n*(n-1)) )
       )
}
```

Parkinson (1980):

Este estimador usar valores extremos, los altos H y bajos L durante el día

$$\sigma_p^2 = \frac{1}{4n\log(2)} \sum_{i=1}^{n} \left(\log\left(\frac{H_i}{L_i} \right) \right)^2.$$

```
vol_p=function(datos_H,datos_L,n){
  sqrt(
    1/(4*n*log(2)) * sum(log(datos_H/datos_L)^2)
    )
}
```

Este método es 5 veces mas eficiente que el close-to-close estimate. O sea, para la misma cantidad de datos la varianza de estos datos es $\frac{1}{5}$ la varianza de los datos del método close-to-close.

Garman & Klass (1980):

Para una eficiencia de 7.4 veces mayor a las de close to close, tenemos

$$\begin{aligned}\sigma_{gk}^2 &= \frac{1}{n}\sum_{i=1}^{n}\left(0.511\left(\log\left(\frac{H_i}{L_i}\right)\right)^2 - 0.019\log\left(\frac{C_i}{O_i}\right)\log\left(\frac{H_i L_i}{O_i^2}\right) - 2\log\left(\frac{H_i}{O_i}\right)\log\left(\frac{L_i}{O_i}\right)\right)\\ &= \frac{1}{n}\sum_{i=1}^{n}\Big(0.511\left(\log(H_i)-\log(L_i)\right)^2 - 0.019\left(\log(C_i)-\log(O_i)\right)\left(\log(H_i)\log(L_i)-\log(O_i)^2\right)\\ &\quad - 2\left(\log(H_i)-\log(O_i)\right)\left(\log(L_i)-\log(O_i)\right)\Big),\end{aligned}$$

donde O_i es el precio de apertura.

```
vol_gk=function(datos_C,datos_O,datos_H,datos_L,n){
  sqrt(
    mean(
      0.511*(log(datos_H)-log(datos_L))^2
      -0.019*(log(datos_C)-log(datos_O))*
        (log(datos_H)*log(datos_L)-log(datos_O)^2)
      -2*(log(datos_H)-log(datos_O))*(log(datos_L)-log(datos_O))
    )
  )
}
```

Rogers & Satchell (1980):

Parkinson y Garman & Klass no son independientes del drift. Un último estimador simple de volatilidad es

$$\begin{aligned}\sigma_{rs}^2 &= \frac{1}{n}\sum_{i=1}^{n}\left(\log\left(\frac{H_i}{C_i}\right)\log\left(\frac{H_i}{O_i}\right)+\log\left(\frac{L_i}{C_i}\right)\log\left(\frac{L_i}{O_i}\right)\right)\\ &= \frac{1}{n}\sum_{i=1}^{n}\Big(\left(\log(H_i)-\log(C_i)\right)\left(\log(H_i)-\log(O_i)\right)+\left(\log(L_i)-\log(C_i)\right)\left(\log(L_i)-\log(O_i)\right)\Big)\end{aligned}$$

```
vol_rs=function(datos_C,datos_O,datos_H,datos_L,n){
  sqrt(
    mean(
      (log(datos_H)-log(datos_C))*(log(datos_H)-log(datos_O))
      +(log(datos_L)-log(datos_C))*(log(datos_L)-log(datos_O))
    )
  )
}
```

Ejemplos prácticos.

Vamos a trabajar con los datos de YPF que descargamos de http://www.investing.com.

```
datos_YPF=read.csv('YPF Historical Data.csv',header=TRUE)
datos_BRKb=read.csv('BRKb Historical Data.csv',header=TRUE)
```

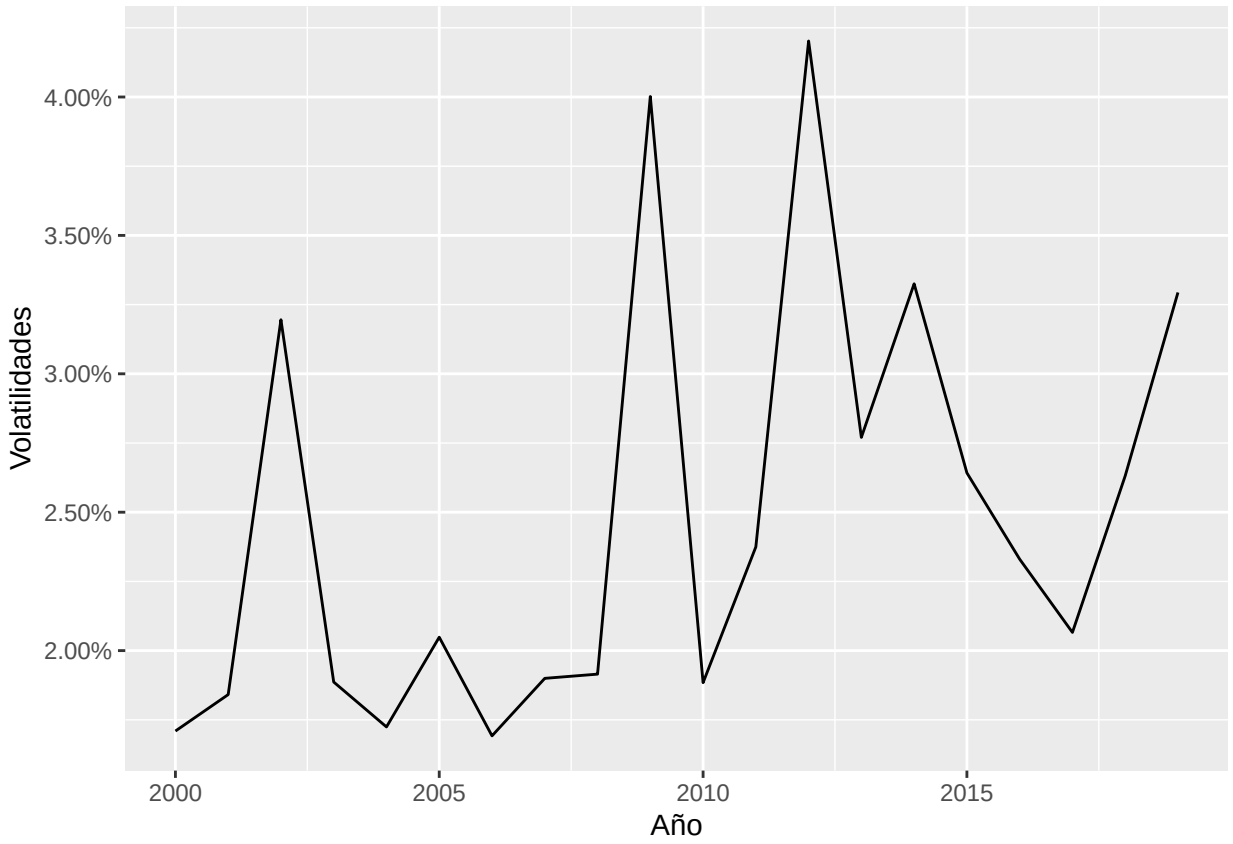
Vamos a acomodar un poco los datos. Estamos usando datos diarios.

```
# Revierto el orden para acomodar las fechas
datos_YPF=datos_YPF[dim(datos_YPF)[1]:1,]
# Creemos una categoría Año y organicemosla
datos_YPF$Año=rep(0,dim(datos_YPF)[1])
for(i in 1:length(datos_YPF$Date)){
  for(j in 2000:2019){
    if(grepl(j, datos_YPF$Date[i], fixed = TRUE))
      {datos_YPF$Año[i]=j}
  }
}
```

Constant Volatility

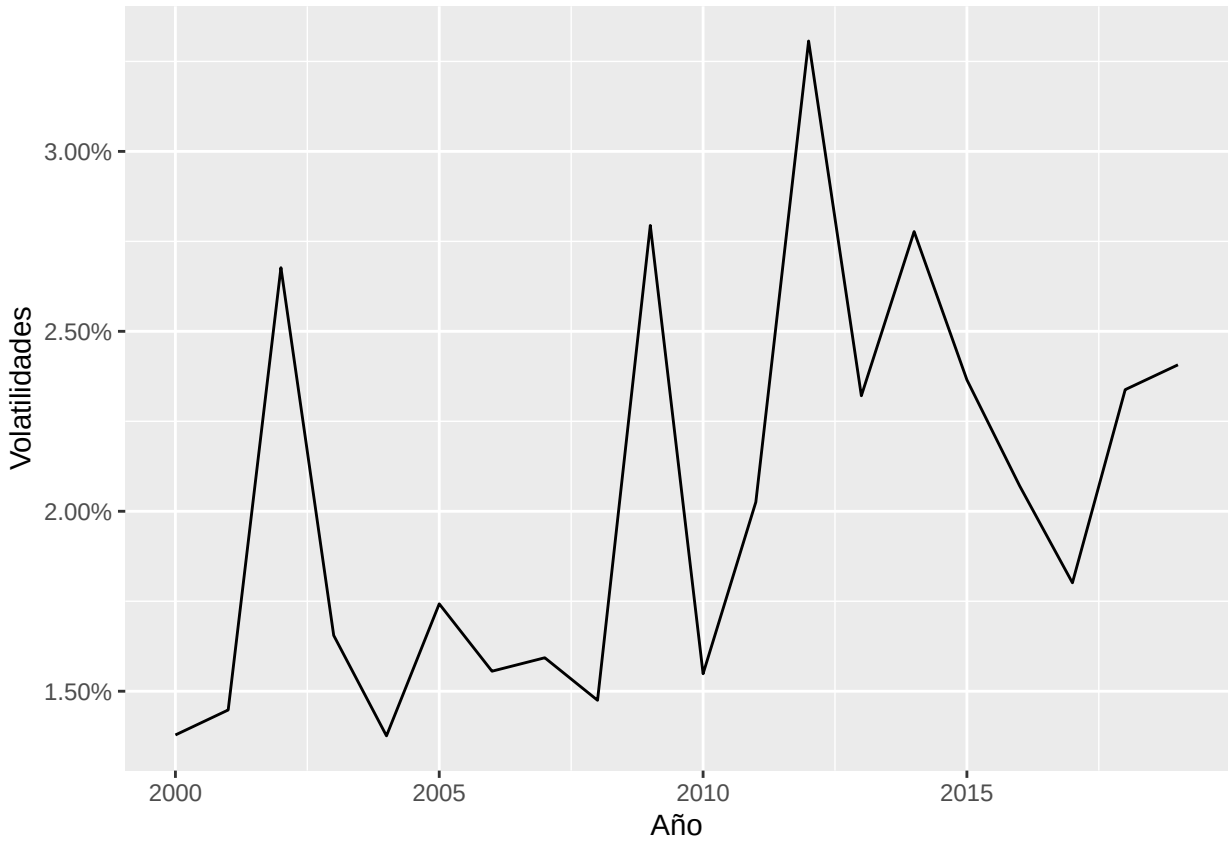
Grafiquemos.

```
años=2000:2019
volatilidades=c()
for(i in años){
  volatilidades=c(volatilidades,vol_cte(retornos_año(datos_YPF,i)))
}
datos_plot=data.frame(cbind(volatilidades,años))
names(datos_plot)=c('Volatilidades','Año')
ggplot(datos_plot,aes(x=Año,y=Volatilidades))+
  geom_line()+
  scale_y_continuous(labels = scales::percent)
```



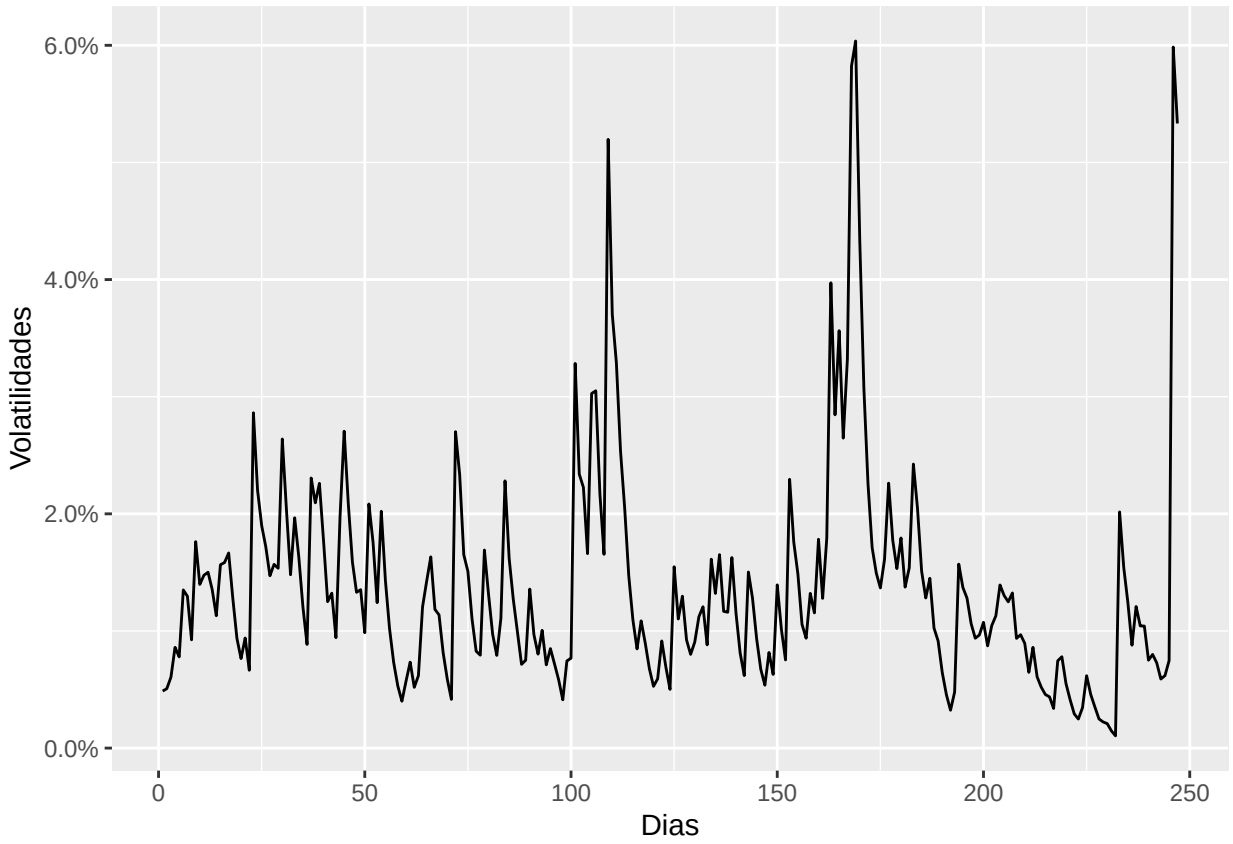
EWMA

```
años=2000:2019
volatilidades=c()
for(i in años){
  volatilidades=c(volatilidades,vol_EWMA(retornos_año(datos_YPF,i),0.5,'media'))
}
datos_plot=data.frame(cbind(volatilidades,años))
names(datos_plot)=c('Volatilidades','Año')
ggplot(datos_plot,aes(x=Año,y=Volatilidades))+
  geom_line()+
  scale_y_continuous(labels = scales::percent)
```



Juguemos un poco con la función. Plotiemos la volatilidad diaria del año 2000.

```
retornos=retornos_año(datos_YPF,2000)
datos_plot=data.frame(cbind('Volatilidades'=vol_EWMA(retornos,0.5,'vol'),
                            'Dias'=1:length(retornos)))
ggplot(datos_plot,aes(x=Dias,y=Volatilidades))+
  geom_line()+
  scale_y_continuous(labels = scales::percent)
```


6.0%
4.0%
2.0%
0.0%
Volatilidades
0
50
100
150
200
250
Dias