

06-ESTACIONARIEDAD

Emiliano Pérez Caullieres

2022-09-20

Contents

1	Mínimos Cuadrados Ordinarios	5
1.1	El problema	5
1.2	Estimación R	6
1.3	Ejercicio	9
2	Máxima Verosimilitud	11
2.1	El problema	11
2.2	Estimación y simulación	12
3	Método Generalizado de Momentos (MGM)	15
3.1	El problema	15
4	CAPITAL ASSET PRICING MODEL (CAPM)	17
4.1	El problema	17
4.2	Estimación R	18
4.3	ESTIMACIÓN	18
4.4	Ejercicio Compara con TSLA con el APPLE	22
5	ESTACIONARIEDAD	25

Chapter 1

Mínimos Cuadrados Ordinarios

1.1 El problema

Recordando que el método de MCO resulta en encontrar la combinación de valores de los estimadores de los parámetros $\hat{\beta}$ que permita minimizar la suma de los residuales (estimadores de los términos de error ε) al cuadrado dada por:

$$\sum_{i=1}^N e_i^2 = \sum_{i=1}^N (y_i - \mathbf{X}_i' \hat{\beta})^2$$

Donde $\hat{\beta}$ denota el vector de estimadores $\hat{\beta}_1, \dots, \hat{\beta}_K$ y dado que $(e_1, e_2, \dots, e_n)'(e_1, e_2, \dots, e_n) = \mathbf{e}'\mathbf{e}$, el problema del método de MCO consiste en resolver el problema de optimización:

$$\begin{aligned} \text{Minimizar}_{\hat{\beta}} S(\hat{\beta}) &= \text{Minimizar}_{\hat{\beta}} \mathbf{e}'\mathbf{e} \\ &= \text{Minimizar}_{\hat{\beta}} (\mathbf{Y} - \mathbf{X}\hat{\beta})'(\mathbf{Y} - \mathbf{X}\hat{\beta}) \end{aligned}$$

Expandiendo la expresión $\mathbf{e}'\mathbf{e}$ obtenemos:

$$\mathbf{e}'\mathbf{e} = \mathbf{Y}'\mathbf{Y} - 2\mathbf{Y}'\mathbf{X}\hat{\beta} + \hat{\beta}'\mathbf{X}'\mathbf{X}\hat{\beta}$$

De esta forma obtenemos que las condiciones necesarias de un mínimo son:

$$\frac{\partial S(\hat{\beta})}{\partial \hat{\beta}} = -2\mathbf{X}'\mathbf{Y} + 2\mathbf{X}'\mathbf{X}\hat{\beta} = \mathbf{0}$$

Y se pueden despejar las *ecuaciones normales* dadas por:

Debido a que el objetivo es encontrar la matriz $\hat{\beta}$ despejamos:

$$\hat{\beta} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{Y}$$

$$\mathbf{X}'\mathbf{X}\hat{\beta} = \mathbf{X}'\mathbf{Y}$$

1.2 Estimación R

Para la estimación utilizaremos el paquete “BatchGetSymbols”. Este paquete nos permitirá descargar información acerca de la bolsa de valores internacional.

1.2.1 Dependencias

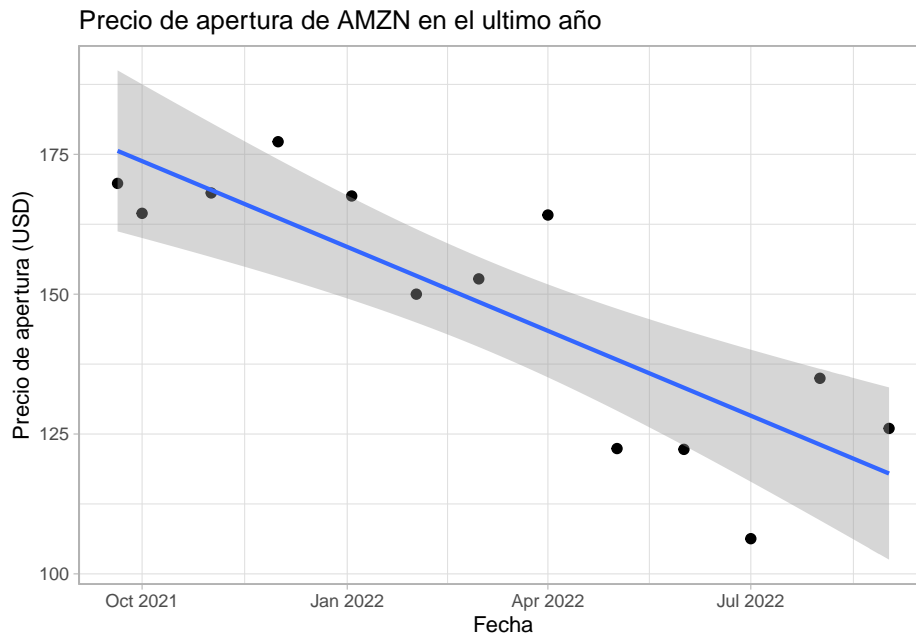
```
#install.packages("pacman")  
#pacman nos permite cargar varias librerías en una sola línea  
library(pacman)  
pacman::p_load(tidyverse,BatchGetSymbols,ggplot2, lubridate)
```

1.2.2 Descarga de los valores

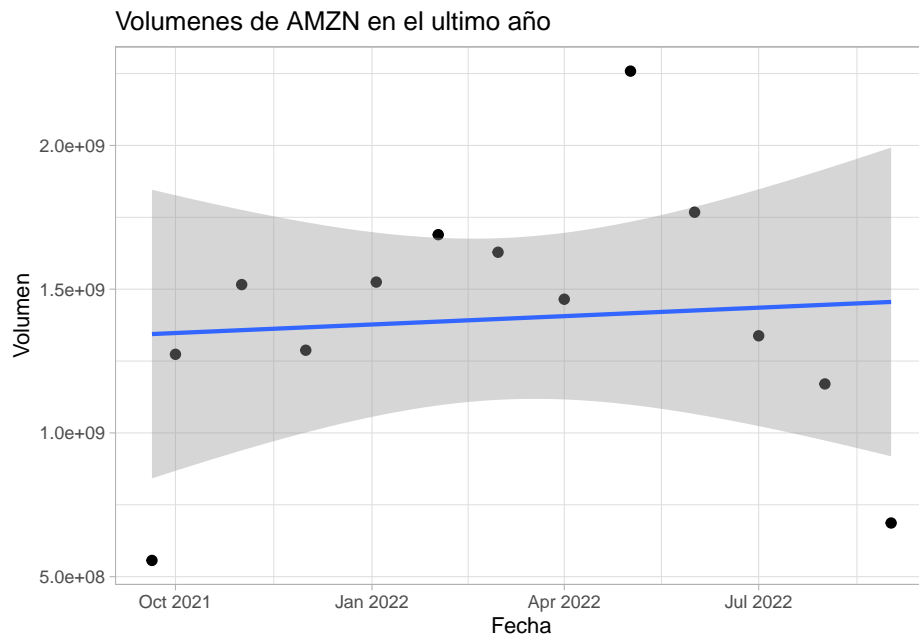
```
#Primero determinamos el lapso de tiempo  
pd<-Sys.Date()-365 #primer fecha  
pd  
#> [1] "2021-09-20"  
ld<-Sys.Date() #última fecha  
ld  
#> [1] "2022-09-20"  
#Intervalos de tiempo  
int<-"monthly"  
#Datos a elegir  
dt<-c("AMZN")  
  
#Descargando los valores  
?BatchGetSymbols()  
data<- BatchGetSymbols(tickers = dt,  
                        first.date = pd,  
                        last.date = ld,  
                        freq.data = int,  
                        do.cache = FALSE,  
                        thresh.bad.data = 0)  
  
#Generando data frame con los valores
```

```
data_precio<-data$df.tickers
colnames(data_precio)
#> [1] "ticker"           "ref.date"
#> [3] "volume"           "price.open"
#> [5] "price.high"        "price.low"
#> [7] "price.close"       "price.adjusted"
#> [9] "ret.adjusted.prices" "ret.closing.prices"
```

```
sp_precio<-ggplot(data_precio, aes(x=ref.date, y=price.open))+geom_point(size =2, colour = "black")
sp_precio
```



```
sp_volumen<-ggplot(data_precio, aes(x=ref.date, y=volume))+geom_point(size =2, colour = "black")+
sp_volumen
```



1.2.4 Regresión lineal que optiene los coeficientes $\hat{\beta}$

```
#datos estadísticos
summary(data_precio[c("price.open","volume")])
#>   price.open      volume
#>   Min.   :106.3   Min.   :5.565e+08
#>   1st Qu.:126.0   1st Qu.:1.273e+09
#>   Median :152.7   Median :1.465e+09
#>   Mean   :148.1   Mean   :1.397e+09
#>   3rd Qu.:167.6   3rd Qu.:1.628e+09
#>   Max.   :177.2   Max.   :2.258e+09
#análisis de regresión lineal lm() y=precio,x=fecha
reg_tiempo_precio<-lm(price.open~ref.date, data=data_precio)
#¡Siempre se pone dentro de lm() la variable dependiente primero y luego la independiente
summary(reg_tiempo_precio)
#>
#> Call:
#> lm(formula = price.open ~ ref.date, data = data_precio)
#>
#> Residuals:
#>      Min       1Q   Median       3Q      Max
#> -21.983  -9.331  -0.524   9.438  20.707
#>
#> Coefficients:
```



```
#>               Estimate Std. Error t value Pr(>|t|)
#> (Intercept) 3324.56599   626.77652   5.304 0.000251 ***
#> ref.date    -0.16670     0.03289  -5.068 0.000362 ***
#> ---
#> Signif. codes:
#> 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 13.19 on 11 degrees of freedom
#> Multiple R-squared:  0.7001, Adjusted R-squared:  0.6729
#> F-statistic: 25.68 on 1 and 11 DF,  p-value: 0.0003617

#análisis de regresión lineal lm() y=volumen,x=fecha
reg_tiempo_volumen<-lm(volume~ref.date, data=data_precio)
summary(reg_tiempo_volumen)
#>
#> Call:
#> lm(formula = volume ~ ref.date, data = data_precio)
#>
#> Residuals:
#>      Min       1Q   Median       3Q      Max
#> -787378903 -97754889  58779600 232265861 842239339
#>
#> Coefficients:
#>               Estimate Std. Error t value Pr(>|t|)
#> (Intercept) -4.755e+09  2.186e+10  -0.217   0.832
#> ref.date     3.228e+05  1.147e+06   0.281   0.784
#>
#> Residual standard error: 460200000 on 11 degrees of freedom
#> Multiple R-squared:  0.007146, Adjusted R-squared: -0.08311
#> F-statistic: 0.07918 on 1 and 11 DF,  p-value: 0.7836
```

1.3 Ejercicio

El objetivo de este ejercicio es simplemente que indiquen y modifiquen los errores en el código. Así pues, deberán descomentar -quitar las #antes del código- para empezar el ejercicio.

1.3.1 1

El objetivo de este código es explicar la variable “**volume**” con la variable “**price.high**”.

```
#reg_tiempo_ej1<-lm(price.high~volume, data=data_precio)
#summary(reg_tiempo_ej1)
```

1.3.2 2

El objetivo de este código es explicar la variable “**volume**” con la variable “**price.low**”.

```
#reg_tiempo_ej2<-lm(price.low~volume, data=data_precio)
#summary(reg_tiempo_ej1)
```

1.3.3 3 (opcional)

El objetivo de este ejercicio es descargar los valores del stock de Tesla *BMV: TSLA* en los últimos *dos años*.

```
#dt_ej3<-("TSLA")
#pdej<-Sys.Date()-(365*3) #primer fecha
#pdej
#Descargando los valores
#dataej3<- BatchgetSymbols(tickers = dt_ej3,
                           #first.date = pdej,
                           #last.date = ld,
                           #freq.data = int,
                           #do.cache = FALSE,
                           #thresh.bad.data = 0)

#Generando data frame con los valores
#data_precio_ej2<-dataej3$df.tickers
#1colnames(data_precio_ej2)
```

Chapter 2

Máxima Verosimilitud

2.1 El problema

Recordemos que dado $f(y_i|\mathbf{x}_i)$ la función de densidad condicional de y_i dado \mathbf{x}_i . Sea θ un conjunto de parámetros de la función. Entonces la función de densidad conjunta de variables aleatorias independientes $\{y_i : y_i \in \mathbb{R}\}$ dados los valores $\{\mathbf{x}_i : \mathbf{x}_i \in \mathbb{R}^K\}$ estará dada por:

$$\Pi_{i=1}^n f(y_i|\mathbf{x}_i; \theta) = f(y_1, y_2, \dots, y_n | \mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n; \theta) = L(\theta) \quad (2.1)$$

A la ecuación (2.1) se le conoce como ecuación de verosimilitud. El problema de máxima verosimilitud entonces será:

$$\max_{\theta \in \Theta} \Pi_{i=1}^n f(y_i|\mathbf{x}_i; \theta) = \max_{\theta \in \Theta} L(\theta) \quad (2.2)$$

Dado que el logaritmo natural es una transformación monótona, podemos decir que el problema de la ecuación (2.2) es equivalente a:

$$\max_{\theta \in \Theta} \ln L(\theta) = \max_{\theta \in \Theta} \ln \Pi_{i=1}^n f(y_i|\mathbf{x}_i; \theta) = \max_{\theta \in \Theta} \sum_{i=1}^n \ln f(y_i|\mathbf{x}_i; \theta) \quad (2.3)$$

Para solucionar el problema se tiene que determinar las condiciones de primer y segundo orden, las cuales serán:

$$\frac{\partial}{\partial \theta} \ln L(\theta) = \nabla \ln L(\theta) \quad (2.4)$$

$$\frac{\partial^2}{\partial^2 \theta} \ln L(\theta) = \frac{\partial}{\partial \theta} \ln L(\theta) \cdot \frac{\partial}{\partial \theta} \ln L(\theta') = H(\theta) \quad (2.5)$$

La solución estará dada por aquel valor de $\hat{\theta}$ que hace:

$$\frac{\partial}{\partial \theta} \ln L(\hat{\theta}) = 0$$

A su vez, la varianza será aquella que resulta de:

$$\text{Var}[\hat{\theta}|\mathbf{X}] = (-\mathbb{E}_{\hat{\theta}}[H(\theta)])^{-1}$$

2.2 Estimación y simulación

2.2.1 Lanzar una moneda

```
set.seed(1234)#esto sirve para siempre generar los mismos numeros aleatorios
#rbinom(numero observaciones,numero de ensayos,probabilidad de exito en cada ensayo)
cara<-rbinom(1,100,0.5)
cara#esto nos dice de los 100 ensayos cuantos fueron cara
#> [1] 47
sol<-100-cara
sol
#> [1] 53

#Ahora definiremos la función que encontrará la función de verosimilitud para determinar
#
verosimilitud <- function(p){
  dbinom(cara, 100, p)
}

#si suponemos que la probabilidad sesgada de que caiga cara es 40%
prob_sesgada<-0.4
#es posible calcular la función de que salga cara
verosimilitud(prob_sesgada)
#> [1] 0.02919091
#ahora es posible generar una función de verosimilitud negativa
#para maximizar el valor de la verosimilitud
neg_verosimilitud <- function(p){
  dbinom(cara, 100, p)*-1
}
neg_verosimilitud(prob_sesgada)
#> [1] -0.02919091
# usamos la función nlm() para maximizar esta función no lineal
#?nlm()
nlm(neg_verosimilitud,0.5,stepmax=0.5)#se pone un parametro porque sabemos que hay un
#> $minimum
```

```
#> [1] -0.07973193
#>
#> $estimate
#> [1] 0.47
#>
#> $gradient
#> [1] 1.589701e-10
#>
#> $code
#> [1] 1
#>
#> $iterations
#> [1] 4
```

Si bien el ejercicio anterior es un tanto repetitivo debido a que sabemos que hay un 50% de que caiga una moneda de un lado o otro. Esto ejemplifica la manera en la que se utiliza el metodo de maximización de máxima verosimilitud.

Chapter 3

Método Generalizado de Momentos (MGM)

3.1 El problema

Retomemos el modelo de regresión lineal tal que:

$$y_i = X_i\beta + u_i \quad (3.1)$$

Tomando en cuenta los principios de ortogonalidad ($E(Z_i u_i) = 0$) y ($\text{rank}E(Z_i' X_i) = 0$) sabemos que β es el único vector de $N \times 1$ que resuelve las condiciones de momento de determinada población. En otras palabras, $E[z_i'(y_i - x_i\beta)] = 0$ es una solución y $E[z_i'(y_i - x_i\beta)] \neq 0$ NO es una solución. Debido a que la media muestral son estimadores consistentes de momentos de una población, se puede:

$$N^{-1} \sum_{i=1}^N z_i'(y_i - x_i\beta) = 0 \quad (3.2)$$

Asumiendo que la ecuación (3.2) tiene L ecuaciones lineales y K coeficientes β desconocidos y $K = L$, entonces la matriz $\sum_{i=1}^N z_i' x_i$ debe ser no singular para encontrar los coeficientes de la siguiente manera.

$$\hat{\beta} = N^{-1} \left[\sum_{i=1}^N z_i' x_i \right]^{-1} \left[\sum_{i=1}^N z_i' y_i \right] \quad (3.3)$$

Para simplificar (3.3) se puede nombrar Z juntando z_i N veces para crear una matriz de tamaño $NG \times L$. Lo mismo hacemos con X juntando x_i para obtener una de $NG \times K$ y Y obteniendo una $NG \times 1$. Obteniendo:

$$\hat{\beta} = [Z' X]^{-1} [Z' Y] \quad (3.4)$$

Es importante tomar en cuenta cuando el caso en el que hay más ecuaciones lineales que coeficientes β ; es decir, $L \geq K$. En estos casos es muy probable que no haya solución, por lo que mejor que se puede estimar es poner la ecuación (3.2), tan pequeña como sea posible. Por lo mismo el paso que nos lleva a la ecuación (3.3), debe eliminarse N^{-1} . El objetivo:

$$\min_{\beta} \left[\sum_{i=1}^N z'_i x_i \beta \right]^{-1} \left[\sum_{i=1}^N z'_i y_i \beta \right] \quad (3.5)$$

Así pues nombramos a W como una matriz simétrica de $W \times W$ donde se genera la variable b que debemos minimizar que sustituye a β creando una función cuadrática en la ecuación (3.3).

$$\min_b \left[\sum_{i=1}^N z'_i x_i b \right]^{-1} \left[\sum_{i=1}^N z'_i y_i b \right] \quad (3.6)$$

$$\therefore \hat{\beta} = [X' Z \hat{W} Z' X]^{-1} [X' Z \hat{W} Z' Y] \quad (3.7)$$

Sin embargo, $X' Z \hat{W} Z' X$ debe ser no singular para que haya una solución. Para esto se asume que \hat{W} tiene un límite de probabilidad no singular. Esto se describe como $\hat{W} \xrightarrow{p} W$ y $N \rightarrow W\infty$ donde W no es aleatorio, es una matriz positiva definida simétrica de $L \times L$.

Chapter 4

CAPITAL ASSET PRICING MODEL (CAPM)

4.1 El problema

Una vez que hemos establecido la manera en la que se pueden estimar algunos valores –como las regresiones lineales y el método de máxima verosimilitud–, además de la naturaleza de los retornos de algunos activos en el capítulo 4, es posible comenzar a hablar de maneras en la que se pueden estimar los valores futuros de los rendimientos de activos y –de esta manera– poder tomar mejores decisiones de inversiones. Por ello, hablaremos del modelo de **Capital Asset Pricing Model**. El modelo es muy sencillo y pretende estimar su rentabilidad esperada en función del **riesgo sistemático**. Por lo mismo, en este modelo se utilizan los valores de los precios de los activos a lo largo del tiempo y utiliza la intuición con la que derivamos la ecuación lineal con los Mínimos cuadrados ordinarios (MCO).

$$R_{jt} - R_{ft} = \alpha_j + \beta_j(R_{mt} - R_{ft}) + u_{jt} \quad (4.1)$$

En la ecuación (4.1)

- R_{jt} es el retorno del portafolio j en el tiempo t
- R_{ft} es el retorno de un bono sin riesgo gubernamental en un año. **Parecido a los CETES.**
- R_{mt} es el retorno en un portafolio de mercado.
- u_{jt} es el retorno en un portafolio de mercado.

- α_j, β_j son los coeficientes que queremos obtener.

De esta manera, α_j es el coeficiente que más nos interesa debido a que queremos ver si el activo supera o no el index del mercado con base en el activo fijo.

Si α_j es positivo entonces sabemos que el retorno tiene buenos rendimientos y uno negativo significa que no. Por tanto $H_0 : \alpha_j = 0$

4.2 Estimación R

Para la estimación utilizaremos el paquete “BatchGetSymbols”. Este paquete nos permitirá descargar información acerca de la bolsa de valores internacional.

4.3 ESTIMACIÓN

4.3.1 Dependencias

```
#install.packages("pacman")
#pacman nos permite cargar varias librerías en una sola línea
library(pacman)
pacman::p_load(tidyverse, BatchGetSymbols, ggplot2, lubridate, readxl, tidyquant)
```

4.3.2 Descarga de los valores

```
#Primero determinamos el lapso de tiempo
pd<-as.Date("2021/09/18") #primer fecha
pd
#> [1] "2021-09-18"
ld<-as.Date("2022/09/18") #última fecha
ld
#> [1] "2022-09-18"
#Intervalos de tiempo
int<-"monthly"
#Datos a elegir
dt<-c("AMZN")
dt2<-c("TSLA")
#Descargando los valores
?BatchGetSymbols()
data1<- BatchGetSymbols(tickers = dt,
                        first.date = pd,
                        last.date = ld,
                        freq.data = int,
                        do.cache = FALSE,
                        thresh.bad.data = 0)
data2<- BatchGetSymbols(tickers = dt2,
```

```

first.date = pd,
last.date = ld,
freq.data = int,
do.cache = FALSE,
thresh.bad.data = 0)

#Generando data frame con los valores
data_precio_amzn<-data1$df.tickers
colnames(data_precio_amzn)
#> [1] "ticker"          "ref.date"
#> [3] "volume"          "price.open"
#> [5] "price.high"      "price.low"
#> [7] "price.close"     "price.adjusted"
#> [9] "ret.adjusted.prices" "ret.closing.prices"
data_precio_tls<-data2$df.tickers
colnames(data_precio_tls)
#> [1] "ticker"          "ref.date"
#> [3] "volume"          "price.open"
#> [5] "price.high"      "price.low"
#> [7] "price.close"     "price.adjusted"
#> [9] "ret.adjusted.prices" "ret.closing.prices"
#necesitamos convertir la serie de tiempo de precios en retornos continuos compuestos de los pre
data_precio_amzn$ccrAMZN<-c(NA ,100*diff(log(data_precio_amzn$price.open)))#agregamos un valor NA
data_precio_amzn$ccrAMZN#estos son los retornos
#> [1] NA -3.2011678 2.1889913 5.3061639
#> [5] -5.6279333 -11.0646538 1.8052718 7.2089622
#> [9] -29.3475086 -0.1185366 -13.9945965 23.8807273
#> [13] -6.8696583

data_precio_tls$ccrTSLA<-c(NA ,100*diff(log(data_precio_tls$price.open)))#agregamos un valor NA
data_precio_tls$ccrTSLA
#> [1] NA 5.796888 38.591933 1.361865 -1.121972
#> [6] -20.478772 -7.264574 21.765521 -22.795320 -13.089771
#> [11] -10.336735 28.307900 -10.009692
#formateando por año y mes
data_precio_tls$ref.date=format(as.Date(data_precio_tls$ref.date), "%m/%Y")
data_precio_amzn$ref.date=format(as.Date(data_precio_amzn$ref.date), "%m/%Y")
#Compararemos con los CETES
CETES_sep2021_2022<-read_excel("BD/CETES-sep2021-2022.xlsx", skip=17)
head(CETES_sep2021_2022)
#> # A tibble: 6 x 2
#> Fecha SF43936
#> <dtm> <dbl>
#> 1 2021-09-15 00:00:00 4.6
#> 2 2021-09-23 00:00:00 4.58

```

```

#> 3 2021-09-30 00:00:00    4.69
#> 4 2021-10-07 00:00:00    4.81
#> 5 2021-10-14 00:00:00    4.79
#> 6 2021-10-21 00:00:00    4.83
#indice sp500
SP500 <- read_csv("BD/Download Data - INDEX_US_S&P US_SPX.csv")
SP500$ccrSP500<-c(NA ,100*diff(log(SP500$Open)))
names(SP500)[1]<-paste('ref.date')
#formateando por año y mes

#cetes
cete_1_año<-10.10#esto es el rendimiento a un año de un cete gubernamental seguro

#Juntamos el df
CAPM_2<-merge(data_precio_amzn, data_precio_tls, by = c('ref.date'))
CAPM_4<-merge(SP500, CAPM_2, by = c('ref.date'))
CAPM<-data.frame(CAPM_4)

#exceso de retorno
CAPM$excess_ret_AMZN<-CAPM$ccrAMZN-cete_1_año
CAPM$excess_ret_SP500<-CAPM$ccrSP500-cete_1_año
CAPM$excess_ret_TSLA<-CAPM$ccrTSLA-cete_1_año

#relacion entre los excesos de demanda
ggplot(CAPM, aes(x=excess_ret_AMZN, y=excess_ret_SP500))+geom_point()+labs(title="Rela
#> Warning: Removed 2 rows containing missing values
#> (geom_point).

#relacion entre los excesos de demanda
ggplot(CAPM, aes(x=excess_ret_TSLA, y=excess_ret_SP500))+geom_point()+labs(title="Rela
#> Warning: Removed 2 rows containing missing values
#> (geom_point).

#veamos la regresion lineal
CAPM_lr<-lm(excess_ret_TSLA~excess_ret_SP500,data = CAPM)
summary(CAPM_lr)
#>
#> Call:
#> lm(formula = excess_ret_TSLA ~ excess_ret_SP500, data = CAPM)
#>
#> Residuals:
#>      Min       1Q   Median       3Q      Max
#> -23.980 -13.391  -5.548  11.572  37.067
#>
#> Coefficients:
#>                                Estimate Std. Error t value Pr(>|t|)

```



Figure 4.1: Relación de excesos de retornos entre AMZN y SP500

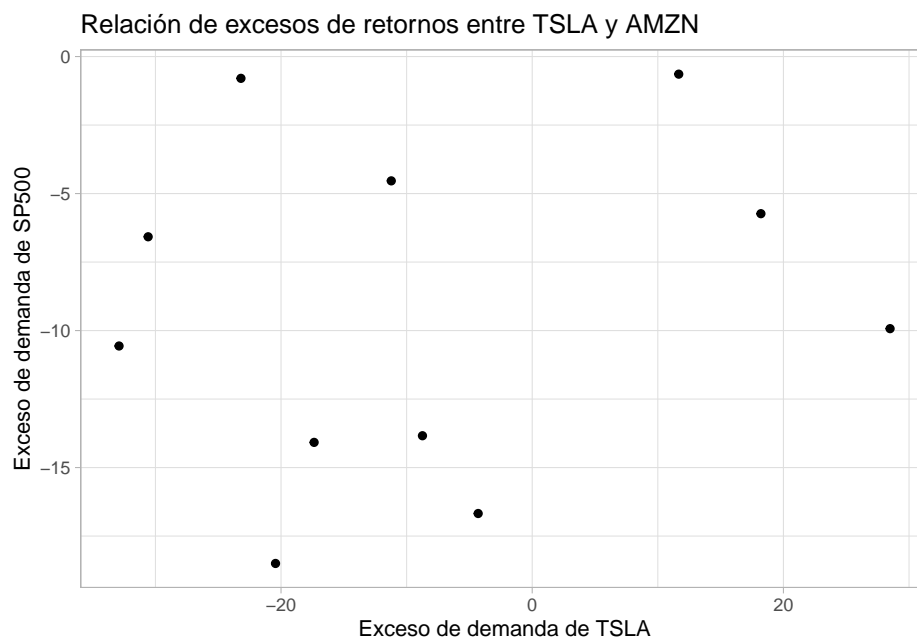


Figure 4.2: Relación de excesos de retornos entre TSLA y SP500

```
#> (Intercept)      -3.2334      11.8097  -0.274      0.79
#> excess_ret_SP500  0.5379      1.0789   0.499      0.63
#>
#> Residual standard error: 20.88 on 9 degrees of freedom
#> (2 observations deleted due to missingness)
#> Multiple R-squared:  0.02687,    Adjusted R-squared:  -0.08125
#> F-statistic: 0.2486 on 1 and 9 DF,  p-value: 0.6301
alpha1<-coefficients(CAPM_lr)[1]
alpha1<0
#> (Intercept)
#>      TRUE
```

De esta manera sabemos que el rendimiento de TSLA NO es mayor debido a que el coeficiente $\alpha = -2.9534$, lo cual indica peores rendimientos al resto del SP500.

4.4 Ejercicio Compara con TSLA con el APPLE

```
dt3<-"AAPL"
data3<-BatchGetSymbols(tickers = dt3,
                        first.date = pd,
                        last.date = ld,
                        freq.data = int,
                        do.cache = FALSE,
                        thresh.bad.data = 0)
#> Warning: `BatchGetSymbols()` was deprecated in BatchGetSymbols 2.6.4.
#> Please use `yfR::yf_get()` instead.
#> 2022-05-01: Package BatchGetSymbols will soon be replaced by yfR.
#> More details about the change is available at github <<www.github.com/msperlin/yfR>
#> You can install yfR by executing:
#>
#> remotes::install_github('msperlin/yfR')
#>
#> Running BatchGetSymbols for:
#>   tickers =AAPL
#>   Downloading data for benchmark ticker
#> ^GSPC | yahoo (1/1)
#> AAPL | yahoo (1/1) - Got 100% of valid prices | Good job!
data_precio_AAPL<-data3$df.tickers
colnames(data_precio_AAPL)
#> [1] "ticker"      "ref.date"
#> [3] "volume"      "price.open"
#> [5] "price.high"  "price.low"
#> [7] "price.close" "price.adjusted"
```

```

#> [9] "ret.adjusted.prices" "ret.closing.prices"
data_precio_AAPL$ccrAAPL<-c(NA ,100*diff(log(data_precio_AAPL$price.open)))#agregamos un valor NA
data_precio_AAPL$ccrAAPL
#> [1] NA -1.330092 4.875668 11.698469 5.996413
#> [6] -2.171531 -5.498712 5.510207 -10.483068 -4.442864
#> [11] -9.701946 16.851754 -2.751627
data_precio_AAPL$ref.date=format(as.Date(data_precio_AAPL$ref.date), "%m/%Y")
CAPM_3<-merge(data_precio_AAPL, CAPM, by = c('ref.date'))
CAPM_3$excess_ret_AAPL<-CAPM_3$ccrAAPL-cete_1_año
#veamos la regresion lineal
CAPM3_lr<-lm(excess_ret_AAPL~excess_ret_SP500,data = CAPM_3)
summary(CAPM3_lr)
#>
#> Call:
#> lm(formula = excess_ret_AAPL ~ excess_ret_SP500, data = CAPM_3)
#>
#> Residuals:
#>      Min       1Q   Median       3Q      Max
#> -10.9038  -5.4365   0.4641   3.4629  14.1798
#>
#> Coefficients:
#>                Estimate Std. Error t value Pr(>|t|)
#> (Intercept)      -4.7542      4.9105  -0.968   0.358
#> excess_ret_SP500   0.4663      0.4486   1.039   0.326
#>
#> Residual standard error: 8.682 on 9 degrees of freedom
#> (2 observations deleted due to missingness)
#> Multiple R-squared:  0.1072, Adjusted R-squared:  0.007964
#> F-statistic: 1.08 on 1 and 9 DF, p-value: 0.3258
alpha2<-coefficients(CAPM3_lr)[1]
alpha2<0
#> (Intercept)
#>      TRUE

```


Chapter 5

ESTACIONARIEDAD

5.0.1 Dependencias

```
#install.packages("pacman")
#pacman nos permite cargar varias librerías en una sola línea
library(pacman)
pacman::p_load(tidyverse, BatchGetSymbols, ggplot2, lubridate, readxl, forecast, stats)

#Primero determinamos el lapso de tiempo
pd<-Sys.Date()-(365*20) #primer fecha
pd
#> [1] "2002-09-25"
ld<-Sys.Date() #última fecha
ld
#> [1] "2022-09-20"
#Intervalos de tiempo
int<-"monthly"
#Datos a elegir
dt<-c("AMZN")
dt2<-c("TSLA")
#Descargando los valores
data1<- BatchGetSymbols(tickers = dt,
                        first.date = pd,
                        last.date = ld,
                        freq.data = int,
                        do.cache = FALSE,
                        thresh.bad.data = 0)
#Generando data frame con los valores
data_precio_amzn<-data1$df.tickers
colnames(data_precio_amzn)
```

```

#> [1] "ticker"          "ref.date"
#> [3] "volume"          "price.open"
#> [5] "price.high"      "price.low"
#> [7] "price.close"     "price.adjusted"
#> [9] "ret.adjusted.prices" "ret.closing.prices"

#necesitamos convertir la serie de tiempo de precios en retornos continuos compuestos
dataPrecioAmzn$ccrAMZN<-c(NA ,100*diff(log(dataPrecioAmzn$price.open)))#agregamos
dataPrecioAmzn$ccrAMZN#estos son los retornos
#> [1] NA 0.37037079 16.90900220 22.83329766
#> [5] -22.98950701 13.39221448 0.95260416 14.27998202
#> [9] 11.55626991 24.11122449 -0.46684144 13.08785513
#> [13] 11.63599301 3.89974592 12.48104071 -0.73260401
#> [17] -3.06108260 -4.08140972 -16.32741069 0.99457279
#> [21] 0.06902105 9.61879412 11.67844638 -33.59147712
#> [25] -0.57381482 7.69997922 -18.78100714 15.60691856
#> [29] 11.66713068 -4.43506452 -20.41392362 -1.23405211
#> [33] -6.96531282 9.64353557 -6.77486160 30.02382912
#> [37] -5.40177077 6.39945115 -12.58398922 20.12391421
#> [41] -2.92703823 -7.77281354 -15.93630872 -2.10477279
#> [45] -4.11970307 -1.60415929 10.64572285 -37.21478392
#> [49] 15.01070027 3.59739571 17.58906665 5.43570463
#> [53] -4.00357496 -1.90531667 3.54637914 1.33891100
#> [57] 42.77167396 11.98170332 -0.13070948 12.66409736
#> [61] 2.27857959 15.63296023 -6.26135927 2.56510858
#> [65] 5.74113839 -18.78533471 -21.72447597 13.78662202
#> [69] 7.15014819 3.44753666 -11.63053849 5.54650897
#> [73] 8.53074641 -14.71605773 -24.20236452 -29.39126222
#> [77] 20.09953181 13.15576837 8.77225235 13.27882326
#> [81] 9.60320128 -2.73678724 7.64068237 2.50334747
#> [85] -6.96036997 13.59745291 24.90536163 14.32806129
#> [89] -0.50514401 -10.08447333 -3.70473986 13.45839135
#> [93] 1.02565002 -9.33660068 -13.76436786 8.99531736
#> [97] 5.87517724 21.76202538 4.58513429 8.56726887
#> [101] 1.22598823 -6.16865517 1.74979032 4.53458328
#> [105] 7.93222740 -0.25978671 4.72685785 9.04110889
#> [109] -4.41608958 0.80039290 -4.18766494 -8.13529694
#> [113] -8.68550170 -1.18960511 3.43828044 9.60224827
#> [117] 14.70991690 -9.58159747 9.53799598 2.08880372
#> [121] 5.85976366 2.83140800 -8.65274050 7.52661134
#> [125] 1.39202437 4.89612270 -2.12710012 1.39936277
#> [129] -5.02332605 5.76221809 3.66491122 8.27850152
#> [133] -6.24554343 9.85520147 15.15285156 8.73395739
#> [137] -0.05013788 -10.51934673 -0.06687288 -5.92858190
#> [141] -10.58568315 2.74371861 4.15753522 -3.80625428

```

```
#> [145] 8.04816141 -5.42111518 -5.03065911 9.90317538
#> [149] -7.85404256 11.32156221 8.43295383 -2.32429615
#> [153] 13.01462014 1.54061721 2.05813986 20.15392877
#> [157] -7.39490376 2.34828935 20.47843365 7.17052347
#> [161] -2.62563883 -12.67694180 -3.85435390 5.96629237
#> [165] 11.72089295 8.23387458 -0.49783030 5.76252931
#> [169] 1.44112456 8.10699776 -4.52676184 -6.00796092
#> [173] 0.72964776 8.98955770 2.83447462 4.01536256
#> [177] 4.38443827 7.35281333 -2.61760725 2.36894617
#> [181] -1.20285851 -2.07377928 13.68712240 5.85471090
#> [185] -0.00427124 20.93976645 4.63815978 -6.55115525
#> [189] 9.77684681 4.61358018 2.75160333 5.84583306
#> [193] 12.74521370 -0.22279328 -21.94794383 8.60716489
#> [197] -18.86826361 11.20213025 0.98664739 8.39682297
#> [201] 7.12720047 -9.38002536 8.85565506 -2.70183034
#> [205] -5.58782369 -1.36520548 2.37757442 0.91249016
#> [209] 3.83805218 6.98245156 -5.31693095 1.37938043
#> [213] 18.97247680 4.64889431 11.92308161 14.25393454
#> [217] 9.27398656 -8.41337555 -4.66642316 4.05671846
#> [221] 2.52393793 -0.84885499 -3.59427728 -0.31861156
#> [225] 11.12179968 -7.17375312 5.72503641 -2.40180912
#> [229] 4.18486227 -6.11473001 2.18899134 5.30616390
#> [233] -5.62793333 -11.06465380 1.80527180 7.20896224
#> [237] -29.34750864 -0.11853658 -13.99459654 23.88072732
#> [241] -6.86965832
#tenemos 20 retornos a lo largo de 20 años
```

Veamos la serie de tiempo

```
ret_20_amzn<-ggplot(data=data_precio_amzn, aes(x=ref.date))+geom_line(aes(y=ccrAMZN))+labs(title=
ret_20_amzn
```

5.0.2 Serie de tiempo

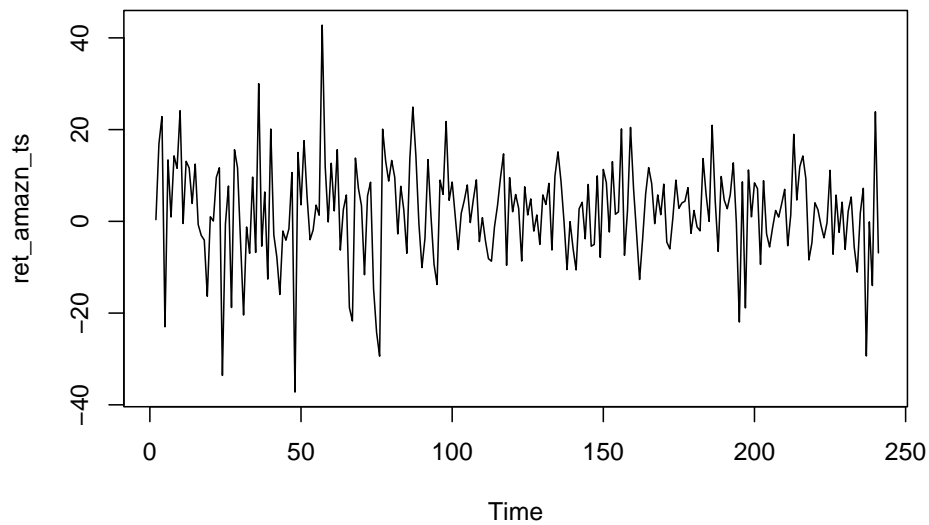
Primero que nada es importante cargar los datos a un objeto series de tiempo. Esto nos lo permite la función `ts()`. Además debemos cerciorarnos de que los datos esten en orden cronológico.

```
data_precio_amzn<-data_precio_amzn[order(data_precio_amzn$ref.date),]
head(data_precio_amzn)#dado que ya estaba en orden cronológico nuestro df no cambia
#> # A tibble: 6 x 11
#>   ticker ref.date      volume price~1 price~2 price~3 price~4
#>   <chr>   <date>         <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
#> 1 AMZN   2002-09-25   7.45e8   0.808   0.87    0.764   0.796
#> 2 AMZN   2002-10-01   4.07e9   0.812   1.01    0.800   0.968
#> 3 AMZN   2002-11-01   4.13e9   0.961   1.23    0.91    1.17
```



Figure 5.1: Serie de tiempo de los retornos de año en los últimos 20 años

```
#> 4 AMZN 2002-12-02 3.11e9 1.21 1.25 0.922 0.944
#> 5 AMZN 2003-01-02 3.38e9 0.960 1.16 0.928 1.09
#> 6 AMZN 2003-02-03 2.32e9 1.10 1.12 0.980 1.10
#> # ... with 4 more variables: price.adjusted <dbl>,
#> #   ret.adjusted.prices <dbl>, ret.closing.prices <dbl>,
#> #   ccrAMZN <dbl>, and abbreviated variable names
#> #   1: price.open, 2: price.high, 3: price.low,
#> #   4: price.close
#hagamos el objeto ts
ret_amzn_ts<-ts(data_precio_amzn$ccrAMZN)
plot(ret_amzn_ts)#de esta manera podemos ver que se cargo bien debido a que es igual
```



5.0.3 Estacionariedad

```
#MA_m5<-forecast::ma(ret_amazn_ts,order=11,centre=TRUE)
#plot(ret_amazn_ts)+lines(MA_m5, col="red", lwd=2)
gglagplot(ret_amazn_ts,lags=20,do.lines=FALSE,colour=FALSE)+theme_minimal()
```

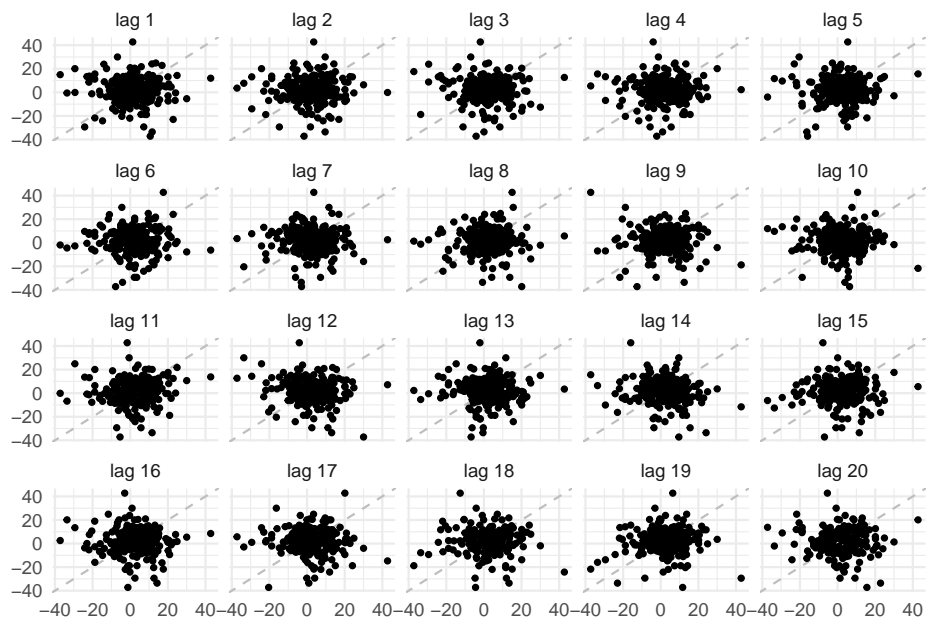


Figure 5.2: Lag Plot que nos muestra la correlación entre 20 lags

```
ACF_ret_amazn_ts<-acf(ret_amazn_ts,na.action = na.pass)
```

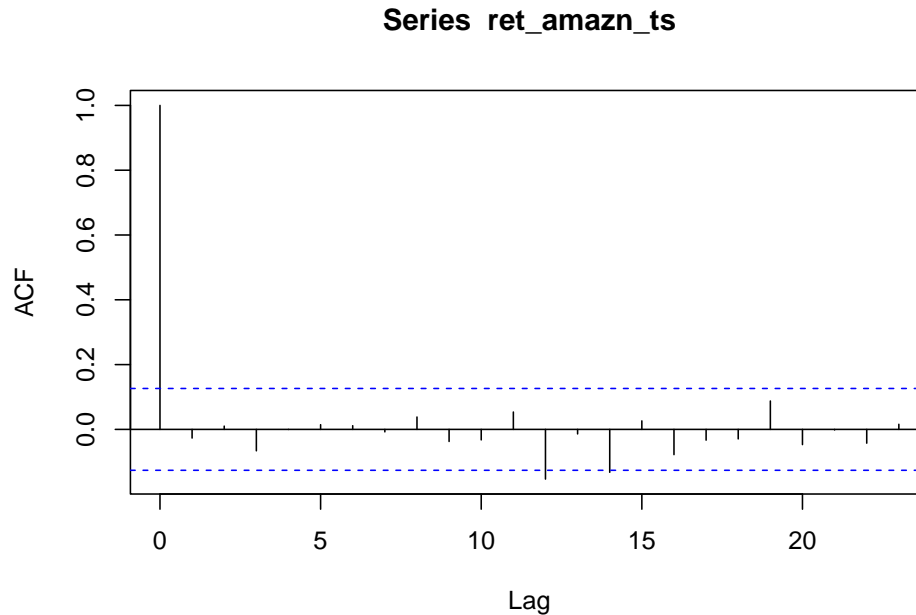


Figure 5.3: Función de Autocorrelación de los retornos de AMZN en los ultimos 20 años

La Figura 5.2 nos indica la manera en la que se correlacionan los lags, evidentemente no se puede ver ningún tipo de correlación visible. Similarmente la Figura 5.3 en donde se muestra la función de autocorrelación. Excepto al primer lag —que muestra correlación debido a que se está comparando consigo mismo— es evidente que no hay correlación fuerte entre ninguno de los lags. Por lo mismo, sería difícil poder encontrar y estimar valores futuros debido a que la Figura 5.2 y la Figura 5.3 indican que la serie de tiempo de los retornos de AMZN de la Figura 5.1 es **completamente aleatorio y no hay estacionariedad**.