Primeros pasos de programación

Francisco Tejeda Dominguez, Bautista Carelli, Julian Pages, Francisco Crisafulli

2022-04-27

#Tecnicas y Herramientas modernas

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.

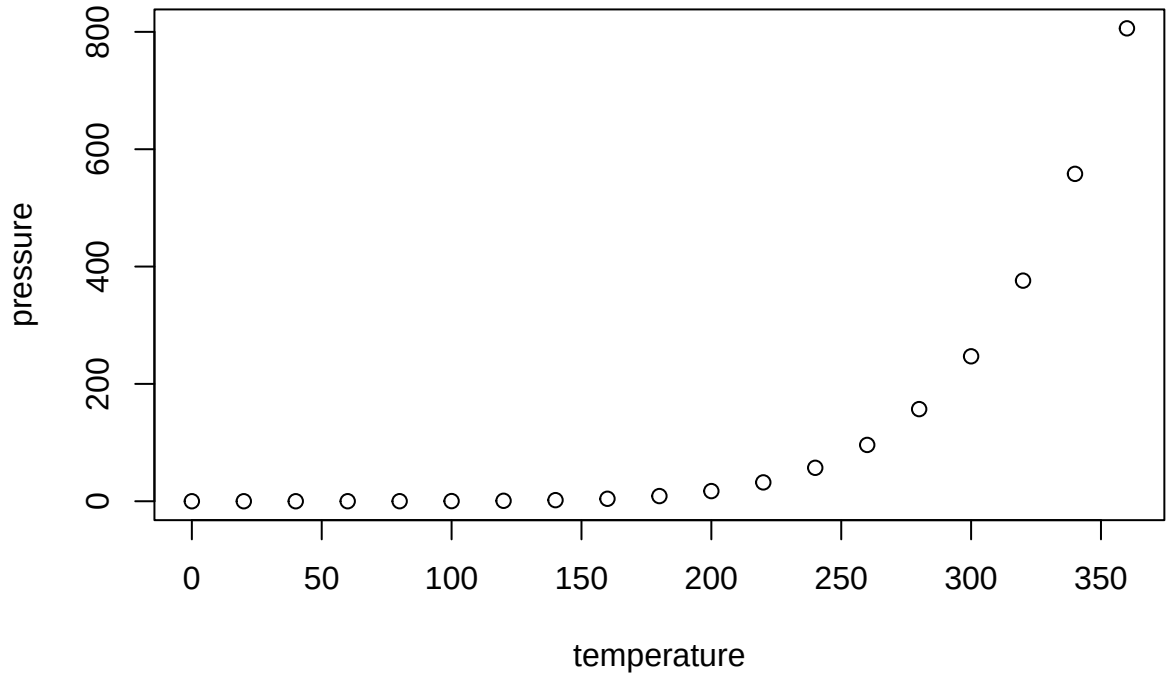
Este es el ejercicio Nº1

```
summary(cars)
```

```
##      speed           dist       
##  Min.   : 4.0   Min.   :  2.00  
##  1st Qu.:12.0   1st Qu.: 26.00  
##  Median :15.0   Median : 36.00  
##  Mean   :15.4   Mean   : 42.98  
##  3rd Qu.:19.0   3rd Qu.: 56.00  
##  Max.   :25.0   Max.   :120.00
```

Including Plots

You can also embed plots, for example:




```
##DESDE ACA ES LO QUE HICIMOS EN CLASE, TUVIMOS QUE CREAR UN . rm PARA PODER ESCRIBIR LOS CODIGOS QUE I


A <- 0
B <- 1
F[1]<-A
F[2]<-B
for( i in 3:100) {

  F[i]<- (F[i-1]+F[i-2])

}
head (F)
```

```
## [1] 0 1 1 2 3 5
```

Este es el ejercicio Nº 2 ordenar un vector

```
start_time <- Sys.time()
#Tomo una muestra de 10 numeros entre 1 y 100
x <- sample(1:100,10)
x
```

```
##  [1] 52 44 64  4 21 10 99 75 36 74
```

```r
#Creo una funcion para ordenar
burbuja <- function(x){
n<-length(x)
start_time <- Sys.time()
for( j in 1:(n-1)){
    for(i in 1:(n-j)){
       if(x[i]>x[i+1]){
        temp <- x[i]
        x[i]<- x[i+1]
        x[i+1]<- temp
       }
    }
  }
 return(x)
}
res<-burbuja(x)
res
```

```
##  [1]  4 10 21 36 44 52 64 74 75 99
```

```r
#Muestra obtenida
x
```

```
##  [1] 52 44 64  4 21 10 99 75 36 74
```

```r
#Ahora vamos a verificar el tiempo que tardo en correr
end_time <- Sys.time( )

end_time - start_time
```

```
## Time difference of 0.184922 secs
```

Consigna Nº3

Repetir el ejercicio anterior utilizando la biblioteca tic toc y microbenchmark

```r
tictoc::tic()

#Tomo una muestra de 10 numeros entre 1 y 100
x <- sample(1:100,10)
x
```

```
##  [1] 83 37 31 88 48 30 53 97  7  3
```

```r
#Creo una funcion para ordenar
burbuja <- function(x){
n<-length(x)
start_time <- Sys.time()
for( j in 1:(n-1)){
    for(i in 1:(n-j)){
```

```
       if(x[i]>x[i+1]){
        temp <- x[i]
        x[i]<- x[i+1]
        x[i+1]<- temp
       }
    }
  }
 return(x)
}
res<-burbuja(x)
res
```

```
##  [1]  3  7 30 31 37 48 53 83 88 97
```

```
#Muestra obtenida
x
```

```
##  [1] 83 37 31 88 48 30 53 97  7  3
```

```
#Ahora vamos a verificar el tiempo que tardo en correr
tictoc::toc( )
```

```
## 0.07 sec elapsed
```

```
library(microbenchmark)

start_time <- Sys.time()

set.seed(2017)
n <- 10000
p <- 100
X <- matrix(rnorm(n*p), n, p)
y <- X %*% rnorm(p) + rnorm(100)
check_for_equal_coefs <- function(values) {
tol <- 1e-12
max_error <- max(c(abs(values[[1]] - values[[2]]),
                   abs(values[[2]] - values[[3]]),
                   abs(values[[1]] - values[[3]])))
  max_error < tol
}

mbm <- microbenchmark("lm" = { b <- lm(y ~ X + 0)$coef },

         "pseudoinverse" = {
              b <- solve(t(X) %*% X) %*% t(X) %*% y
            },
         "linear system" = {
              b <- solve(t(X) %*% X, t(X) %*% y)
            },
        check = check_for_equal_coefs)
#Muestra obtenida
mbm
```

```
## Unit: milliseconds
##           expr      min       lq     mean   median       uq      max neval
##             lm 250.7993 280.9916 305.9708 302.1421 325.4887 398.3052   100
##  pseudoinverse 359.5632 411.4447 439.8936 437.6612 464.7589 604.2702   100
##  linear system 195.8391 220.8556 248.1290 244.0419 269.4367 382.7491   100
```

```
#Ahora vamos a verificar el tiempo que tardo en correr
end_time <- Sys.time( )

end_time - start_time
```

```
## Time difference of 1.678492 mins
```

Creación de vectores

```
v1 <- c(1,2,3,4,5)
```

Creación de un vector de 9 componentes

```
v2 <- c(1,2,3,4,5,6,7,8,9)
```

Creación de matrices

```
m1 <- matrix(v2,ncol=3,byrow = FALSE)
```

byrow =TRUE: lo hace por fila byrow =FALSE: lo hace por columna

Averiguar que clase de objeto hemos creado

Para saber de que clase es un objeto se utiliza el comando 'class(nombre_del_objeto)'

```
class(v1)
```

```
## [1] "numeric"
```

```
class(m1)
```

```
## [1] "matrix" "array"
```

Creación de un vector de palabras

```r
v3 <- c("a","b","c")
class(v3)
```

```
## [1] "character"
```

```r
v3
```

```
## [1] "a" "b" "c"
```

##Asi elijo la columna 2 fila 2 m1[2,2] [1] 5

##Asi elijo todas las filas y la columna 2 m1[,2] [1] 4 5 6

##Asi elijo toda la matriz excepto la columna 2 m1[,-2] [,1] [,2] [1,] 1 7 [2,] 2 8 [3,] 3 9

Asi elijo las primeras 4 componentes del vector

v2[1:4] [1] 1 2 3 4

Asi calculo la desviacion standard de la matriz 1 en la fila 2 y todas las columnas

sd(m1[2,]) [1] 3

Importar dato de la red o de excel

```r
library(readr)
casos <- read_delim("casos.csv",
    delim = ";", escape_double = FALSE, col_types = cols(`Covid Argentina` = col_date(format = "%m/%d/%
        ...2 = col_integer()), trim_ws = TRUE)
```

```
## New names:
## * `` -> `...2`
## * `` -> `...3`
```

```
## Warning: One or more parsing issues, see `problems()` for details
```

Plote de datos

```r
casos$...2
```

```
##  [1]   NA    1    2    2   12   17   19   21   31   34   45   56   65   79   98
## [16]  128  158  225  266  301  387  502  589  690  745  820 1054 1054 1133 1265
## [31] 1353 1451 1554 1628 1715
```

```
plot(casos$...2, main="Contagios 2020", ylab="Semana", xlab="Casos Positivos")
```