

Un curso introductorio de R

Scripts, funciones y loops

Mario Gavidia-Calderón

3/31/2021

El Menú de hoy

- ▶ Operaciones lógicas
- ▶ subset()
- ▶ Scripts
- ▶ Funciones
- ▶ Loops
- ▶ packages

Operaciones lógicas

Operaciones lógicas

- ▶ TRUE = 1
- ▶ FALSE = 0
- ▶ Conjunción es **y** , en R es **&**, equivale a *****
- ▶ Disjunción es **o** , en R es **|** , equivale a **+**

Operaciones lógicas

- ▶ Entonces:
 - ▶ TRUE & TRUE
 - ▶ TRUE & FALSE
 - ▶ FALSE & TRUE
 - ▶ FALSE & FALSE

Operaciones lógicas

► Entonces:

```
TRUE & TRUE
```

```
## [1] TRUE
```

```
TRUE & FALSE
```

```
## [1] FALSE
```

```
FALSE & TRUE
```

```
## [1] FALSE
```

```
FALSE & FALSE
```

```
## [1] FALSE
```

Operaciones lógicas

- ▶ Entonces:
 - ▶ TRUE | TRUE
 - ▶ TRUE | FALSE
 - ▶ FALSE | TRUE
 - ▶ FALSE | FALSE

Operaciones lógicas

► Entonces:

```
TRUE | TRUE
```

```
## [1] TRUE
```

```
TRUE | FALSE
```

```
## [1] TRUE
```

```
FALSE | TRUE
```

```
## [1] TRUE
```

```
FALSE | FALSE
```

```
## [1] FALSE
```


Operaciones lógicas

- ▶ Mayor o menor que: $>$, $<$
- ▶ Mayor o menor igual que: $>=$, $<=$
- ▶ Es igual que: $==$
- ▶ No es igual que: $!=$

subset()

subset()

- ▶ De ?subset:
 - ▶ Retorna un subconjunto de vectores, matrices o data frames que **satisfacen las condiciones**
- ▶ **condiciones = operaciones lógicas**
- ▶ `subset(data, subset = _condiciones_)`

subset(): Leyendo encuesta

```
survey <- read.table(  
  "../02_data/respuestas27.csv",  
  header = T,  
  sep = ",",  
)
```

subset(): Leyendo encuesta

```
names(survey) <- c("date", "name", "last.name",  
                  "age", "district", "molinero",  
                  "faculty", "year", "program",  
                  "prog.lang", "os", "labs",  
                  "Excel", "R", "why")  
  
names(survey)
```

```
## [1] "date"      "name"      "last.name" "age"      "di  
## [7] "faculty"   "year"      "program"   "prog.lang" "os  
## [13] "Excel"    "R"         "why"
```

subset(): Leyendo encuesta

- ▶ Cuántos estudian ing. ambiental

```
ing_amb <- subset(  
  survey,  
  subset = faculty == "Ingeniería ambiental")  
class(ing_amb)
```

```
## [1] "data.frame"
```

```
nrow(ing_amb)
```

```
## [1] 9
```

subset(): Leyendo encuesta

- ▶ Cuántos son menores que la edad média

```
menores_edad_media <- subset(  
  survey,  
  subset = age < mean(survey$age)  
)
```

```
nrow(menores_edad_media)
```

```
## [1] 12
```

subset(): Leyendo encuesta

- ▶ Qué estudia el más valiente

```
par_fin <- "No hago las prácticas me defiendo en el parcial"
temerario <- subset(
  survey,
  subset = labs == par_fin
)

temerario$faculty

## [1] "Meteorología"
```


subset(): Leyendo encuesta

- Programan y saben Python

```
program_py <- subset(  
  survey,  
  subset = (program == "Sí" | program == "Algo" &  
            prog.lang == "Python")  
)  
  
head(program_py[, c("program", "prog.lang")], 5)
```

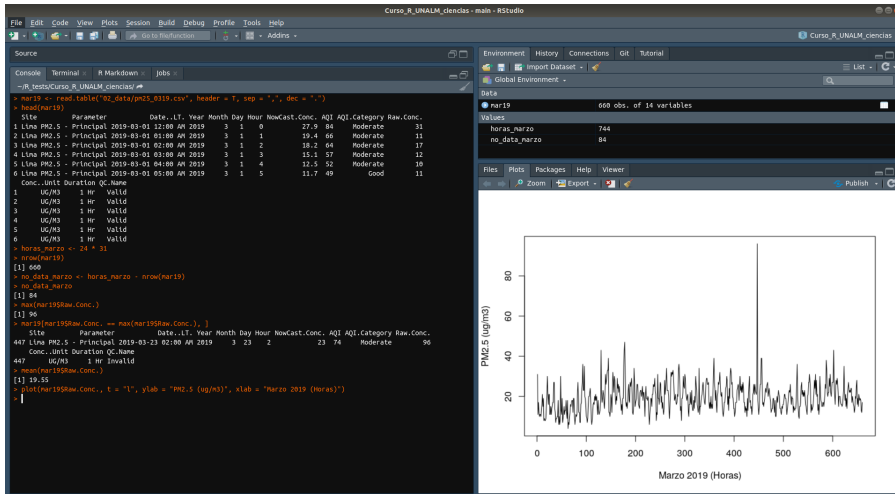
```
##      program prog.lang  
## 1      Algo    Python  
## 3      Algo    Python  
## 4      Algo    Python  
## 9      Algo    Python  
## 16     Sí      Python
```

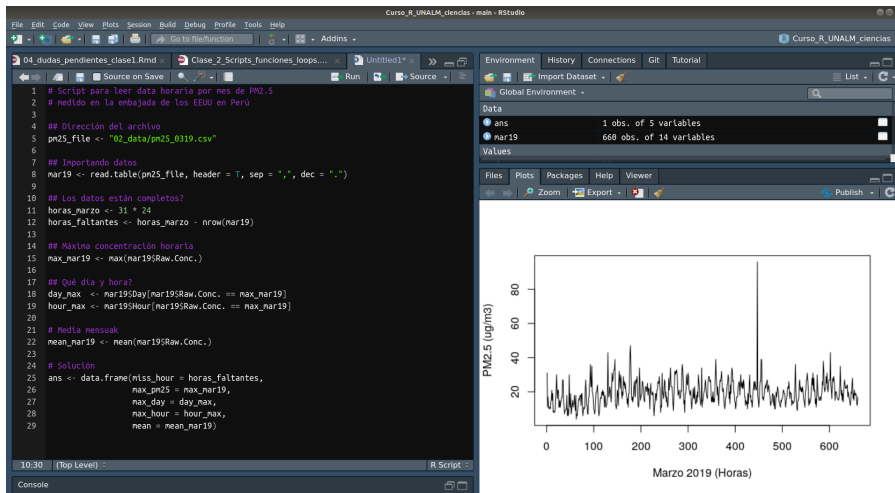
Scripts

Qué es un script?

- ▶ Una **receta**: *Script para analizar resultados de encuesta*.
 - ▶ Serie de operaciones (statements) en un archivo de texto (*.R)
- ▶ Consola de R para **experimentar**: *Qué hace está función?*
- ▶ Editor de texto¹:
 - ▶ **Espacio** para trabajar
 - ▶ Código que **funciona** y nos **importa**

¹Wickham & Golemund, 2016





Scripts

- ▶ Un proceso ordenado
- ▶ Suceptible a mejoras
- ▶ Más fácil de compartir

Funciones

Funciones

```
functionName <- function(arg1, arg2, ...){  
  statements1  
  statements2  
  ...  
  return(object)  
}
```


Funciones: Ejemplos prácticos

► Cubo de un número

```
CuboNumero <- function(x){  
  cubo <- x * x * x  
  return(cubo)  
}
```

```
CuboNumero(3)
```

```
## [1] 27
```

Funciones: Ejemplos prácticos

► Más argumentos

```
MyAge <- function(this_year, born_year){  
  my_age <- this_year - born_year  
  return(my_age)  
}
```

```
MyAge(2020, 1988)
```

```
## [1] 32
```

Funciones: Ejemplos prácticos

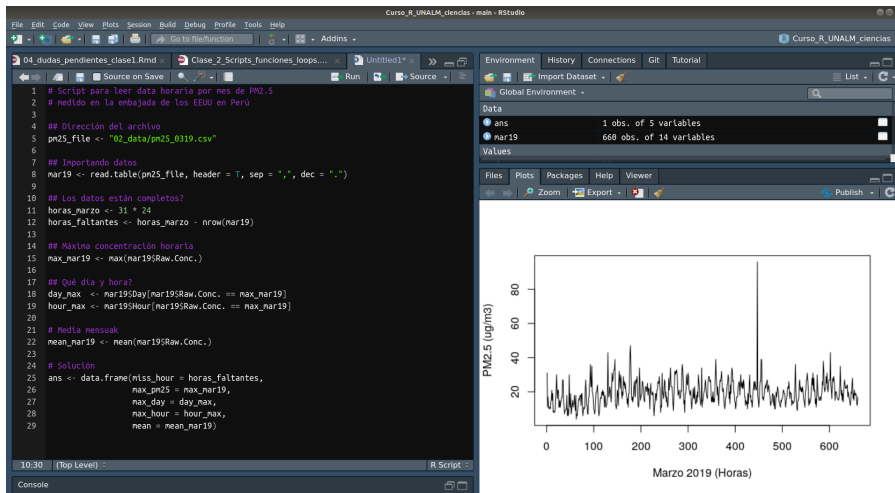
```
MediaVector <- function(vec){  
  suma <- sum(vec)  
  n <- length(vec)  
  media <- suma / n  
  return(media)  
}
```

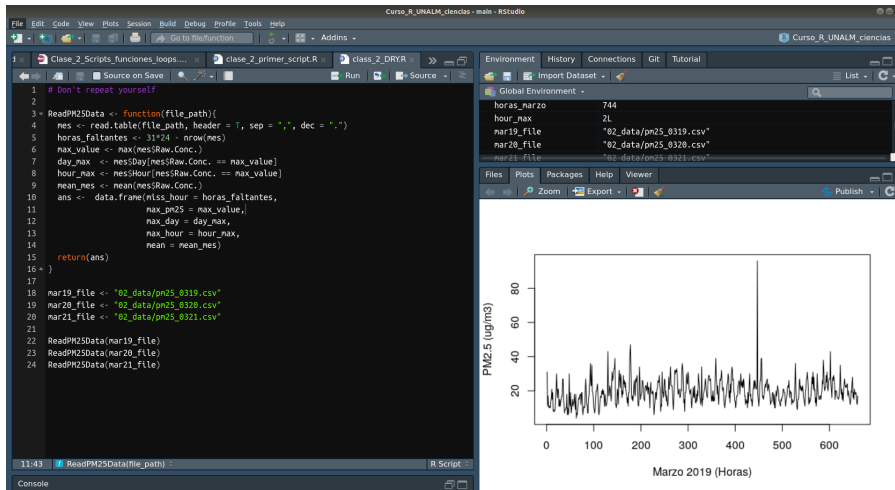
```
mi_vector <- c(1, 4, 5, 6, 15)  
MediaVector(mi_vector)
```

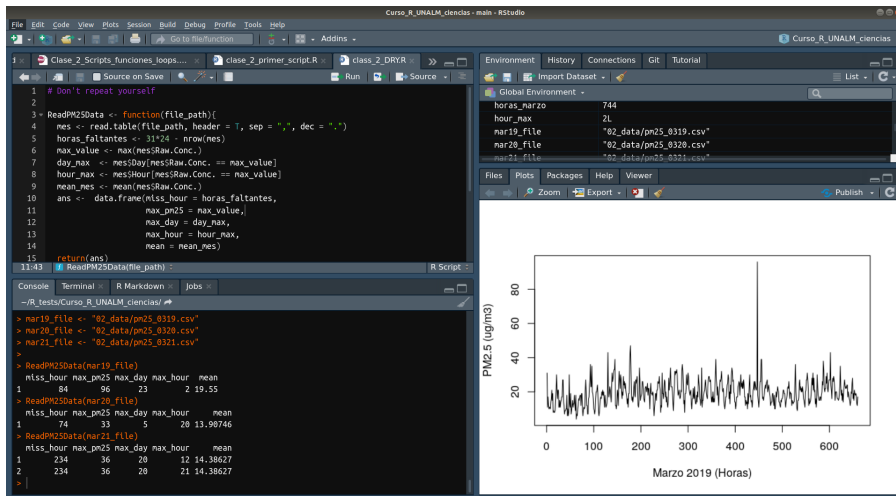
```
## [1] 6.2
```

Scripts y funciones

Usualmente un **script** puede transformarse en una **función**.







Loops

Loops



From: Bart the Genius

Season 1, Episode 2

In: <http://bartsblackboard.com/tag/chalk/>

Loops

► Sintaxis

```
for (var in sequence){  
  statements  
}
```

Loops

- Haciendo la tarea de Bart

```
for (i in seq(1, 10)){  
  print("Ya no gastaré más tiza")  
}
```

```
## [1] "Ya no gastaré más tiza"  
## [1] "Ya no gastaré más tiza"  
## [1] "Ya no gastaré más tiza"  
## [1] "Ya no gastaré más tiza"  
## [1] "Ya no gastaré más tiza"  
## [1] "Ya no gastaré más tiza"  
## [1] "Ya no gastaré más tiza"  
## [1] "Ya no gastaré más tiza"  
## [1] "Ya no gastaré más tiza"  
## [1] "Ya no gastaré más tiza"
```

Loops: Ejemplos prácticos

► Por **posición**

```
tempC <- c(20, 22, 23, 25)
for (i in seq(1, length(tempC))) {
  print(tempC[i] + 273.15)
}
```

```
## [1] 293.15
```

```
## [1] 295.15
```

```
## [1] 296.15
```

```
## [1] 298.15
```

Loops: Ejemplos prácticos

► Por elemento

```
tempC <- c(20, 22, 23, 25)
for (t in tempC){
  print(t + 273.15)
}
```

```
## [1] 293.15
```

```
## [1] 295.15
```

```
## [1] 296.15
```

```
## [1] 298.15
```

Loops: Ejemplos prácticos

- Una función a un vector

```
for (i in mi_vector){  
  print(CuboNumero(i))  
}
```

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

```
## [1] 64
```

```
## [1] 125
```

```
## [1] 216
```

```
## [1] 3375
```

Loops: Advertencia

- Obviamente es más sencillo:

```
tempC + 273.15
```

```
## [1] 293.15 295.15 296.15 298.15
```

```
CuboNumero(mi_vector)
```

```
## [1]      1    64   125   216  3375
```

- Loops son importantes para *automatizar*

Don't repeat yourself (DRY)

- ▶ Es un principio de programación
- ▶ Si estás reescribiendo código es momento de:
 - ▶ Crear una **función**
 - ▶ Hacer un **loop**

Packages

Packages

- ▶ Conjunto de funciones desarrolladas para tareas *ad hoc*:
 - ▶ Análisis de datos contaminación de aire: `openair`
 - ▶ Crear aplicaciones: `shiny`
 - ▶ GIS: `sf` y `raster`
 - ▶ Webscrapping: `RCurl`
 - ▶ *Etcetera*

Packages

► Instalación

```
install.packages("openair") # Siempre colocar las " "
```

► Utilizar

```
library(openair) # No es necesario las " "
```

► Buena práctica: Colocarlas al inicio de cada script