Universidad de Alcalá

Programa de Doctorado

Más Programación

Manejo y visualización de datos en R

Paloma Ruiz Benito, Verónica Cruz Alonso,
Julen Astigarraga Urcelay

Enero - 2022

Contenido

Día 2

Parte I.- Introducción a R

- Entornos de programación, introducción a R y RStudio.
- Tipos de variables y datos, operaciones aritméticas y lógicas, creación de vectores, matrices, listas y tablas. Selección de datos.

Parte II.- Manejo de datos

- Flujo y funciones en la gestión de bases de datos.
- Recomendaciones para la estructura de las bases de datos y creación de código de programación.
- Introducción a la gestión de datos. Recomendaciones para su generación.
- Estructuras de programación: condicionales, bucles y funciones.

Parte III.- Visualización de datos

- Funciones básicas e introducción a ggplot para la visualización de datos.
- Generación de gráficos unidimensionales: histograma, dispersión, gráfico de cajas y bigotes, etc.
- Generación de gráficos bidimensionales: dispersión, boxplot, gráficos de barras, etc.
- Ejemplos y prácticas de visualización de gráficos en mapas.

Parte IV.- Trabajo reproducible

- Trabajo reproducible.
- Introducción a git y github.
- Rmarkdown.

Más programación

#Condicionales

#Bucles

Más programación

#Condicionales

Igualdad	Desigualdad
`> TRUE == TRUE` `TRUE`	`> TRUE != TRUE` `FALSE`
`> TRUE == FALSE` `FALSE`	`> TRUE != FALSE` `TRUE`
`> “coche” == “moto”` `FALSE`	`> “coche” != “moto”` `TRUE`
`> 4 == 1` `FALSE`	`> 4 != 1` `TRUE`

Más programación

#Condicionales

Menor	Mayor
> 3 < 5 TRUE	> 3 > 5 FALSE
> “coche” < “moto” TRUE Orden alfabético	> “coche” > “moto” FALSE
> TRUE < FALSE FALSE TRUE se transforma a 1	> TRUE > FALSE TRUE FALSE se transforma a 0

Más programación

#Condicionales

```
> price<-c(1:15)
> price > 10
[1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE FALSE FALSE TRUE
[12] TRUE TRUE TRUE TRUE
> threshold<-rep(10,15)
> threshold
[1] 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
> threshold == price
[1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE FALSE TRUE FALSE
[12] FALSE FALSE FALSE FALSE
```

Más programación

#Condicionales: operadores lógicos

Y	O
`> TRUE & TRUE` `TRUE`	`> TRUE | TRUE` `TRUE`
`> TRUE & FALSE` `FALSE`	`> TRUE | FALSE` `TRUE`
`> FALSE & FALSE` `FALSE`	`> FALSE | FALSE` `FALSE`

`> y <- 19`
`> y < 10 & y > 5`
`FALSE`
	`> y <- 19`
	`> y < 10 | y > 5`
	`TRUE`

Más programación

#Condicionales: operadores lógicos

! (operador "no")	Comparando vectores

`> !TRUE`
`FALSE`
	`> c(TRUE, TRUE, FALSE) &`
	`c(TRUE, FALSE, FALSE)`
	`[1] TRUE FALSE FALSE`

`> !FALSE`
`TRUE`
	`> c(TRUE, TRUE, FALSE) |`
	`c(TRUE, FALSE, FALSE)`
	`[1] TRUE TRUE FALSE`

`> is.numeric("coche")`
`FALSE`
`> !is.numeric("coche")`
`TRUE`	

`> is.numeric(5)`
`TRUE`
`> !is.numeric(5)`
`FALSE`	

Más programación

SENTENCIA CONDICIONAL "IF"

```
if(argumento){
        expression
        }
```

```
> ##-----SENTENCIA IF
> x<-1
> if(x==1){"se cumple la condición"}
[1] "se cumple la condición"
> x<-2
> if(x==1){"se cumple la condición"}
```

4- Más programación

SENTENCIA "ELSE"

```
if(argumento){expression1} else{expression2}
```

```
>#--------SENTENCIA ELSE
> x<-2
> if(x==1){"se cumple la condición"} else{"no se
cumple"}
[1] "no se cumple"
```

Más programación

SENTENCIA "IFELSE"

```
ifelse(condicion,expression1,expression2)
```

```
> ##-----SENTENCIA IFELSE
> x<-1
> ifelse(x==1,"se cumple la condición","no se cumple")
[1] "se cumple la condición"
```

Más programación

SENTENCIA "while"

```
while(condicion){expression}
```

```
> precio<-1
> while(precio<=7){
                print("precio bajo")
                                }
```

4- Más programación

SENTENCIA "while"

```
while(condicion){expression}
```

¡OJO! La condición debe volverse falsa en algún momento

```
> precio<-1
> while(precio<=7){
                print("precio bajo") +
                precio <- precio + 1
                }
[1] "precio bajo"
[1] "precio bajo"
[1] "precio bajo"
[1] "precio bajo"
[1] "precio bajo"
[1] "precio bajo"
[1] "precio bajo"
```

Más programación

SENTENCIA "while"

```
while(condicion){expression}
```

¡OJO! La condición debe volverse falsa en algún momento

```
> precio<-1
> while(precio<=7){
                print(paste("precio bajo" + precio))
                precio <- precio + 1
                }
[1] "precio bajo 1"
[1] "precio bajo 2"
[1] "precio bajo 3"
[1] "precio bajo 4"
[1] "precio bajo 5"
[1] "precio bajo 6"
[1] "precio bajo 7"
```

Más programación

SENTENCIA "for"

```
for(var in seq){expression}
```

Repite una acción un número determinado de veces

```
> precios <- c(1, 1.2, 1.3, 1.5, 1.2, 1.7,
1.4)
> for(i in 1:length(precios)){
        print(p[i])
        }
[1] 1
[1] 1.2
[1] 1.3
[1] 1.5
[1] 1.2
[1] 1.7
[1] 1.4
```

Más programación

SENTENCIA "for"

```
for(var in seq){expression}
```

Repite una acción un número determinado de veces

```
> precios <- list(1, 1.2, 1.3, 1.5, 1.2, 1.7,
1.4)
> for(i in 1:length(precios)){
      print(p[[i]])
      }
[1] 1
[1] 1.2
[1] 1.3
[1] 1.5
[1] 1.2
[1] 1.7
[1] 1.4
```

Más programación

SENTENCIA "for"

```
for(var in seq){expression}
```

Repite una acción un número determinado de veces

```
> ciudades <- list("Madrid", "Paris",
"Londres", "Tokio",
"Rio de Janeiro")

> for(c in ciudades) {
    print(ciudades))
   }
```

Más programación

SENTENCIA "for"

```
for(var in seq){expression}
```

Repite una acción un número determinado de veces

```
> ciudades <- list("Madrid", "Paris",
"Londres", "Tokio",
"Rio de Janeiro")

> for(i in 1:length(ciudades)) {
    if(ciudades[i]==6){
       break
       }
       print(paste(ciudades[i], "está en la posición",
       i, "en el vector ciudades"))
   }
```

Más programación

Más funciones útiles

which() devuelve la posición de los valores que cumplan con la condición

```
> x <- c(1,2,3,4,5,NA,NA,NA,9)
> which(x>1)
[1] 2 3 4 5 9
> which(is.na(x))
[1] 6 7 8
> which(!is.na(x))
[1] 1 2 3 4 5 9
```

Crea un vector llamado vec1 con 100 números siguiendo una distribución normal con media 2 y desviación estándar 0.1.

Crea después una matriz de datos llamada mat1 con 20 filas y 10 columnas. Utilizando for(), genera 10 muestras aleatorias de 20 elementos del vector vec1 con la función sample() y sálvalas en cada una de las columnas de la matriz mat1. Sírvete siempre que lo necesites de las paginas de ayuda de las funciones utilizadas para conocer que argumentos requieren.


useR!