Actividades en Finanzas 2: Valuación de Opciones y Simulaciones de Monte Carlo en R

Semana 3: R Script I

Ricardo Huamán

2022-03-10

R File Manipulation Commands

Limpiar el espacio de trabajo

```
rm(list=ls())
```

Para ubicar el espacio de trabajo actual, use el comando **getwd()**

```
getwd()
```

```
## [1] "C:/Users/sandr/Dropbox/PUCP/2022-0/Actividades en Finanzas/Dictado/1FIN17/clase3"
```

Para hacer un listado de los directorios y subdirectorios dentro del espacio de trabjo, use el comando **list.files()**

```
list.files()
```

```
## [1] "clase3.pdf" "clase3.Rmd"
```

Si se desea cambiar el espacio de trabajo, se puede settear usando el comando **setwd()**, y dentro de los paréntesis, puede ubicar el path del nuevo espacio de trabajo.

```
# setwd()
```

Instalar paquetes

```
install.packages("dplyr")
```

Luego de haber instalado el paquete, se debe llamar a la librería para poder ser usado en el presente script.

```
library("dplyr")
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

R como un espacio de calculadora

R usa los símbolos usuales de aritmética como + para suma, - para restas, * para multiplicaciónes, / para división, y ^ para exponente. Asimismo, se pueden usar paréntesis () para especificar el orden de las operaciones. R también usa %% para poder operar módulos y %/% para divisiones enteras.

```
(1 + 1/100)^100
```

```
## [1] 2.704814
```

```
17 %% 5
```

```
## [1] 2
```

```
17 %/% 5
```

```
## [1] 3
```

R tiene integrado cierto número de funciones: `sin(x)`, `cos(x)`, `tan(x)`, (todo en radiones), `exp(x)`, `log(x)`, y `sqrt(x)`. Algunas constantes especiales como `pi` también ya están predefinidas.

```
exp(1)
```

```
## [1] 2.718282
```

```
options(digits = 16)
exp(1)
```

```
## [1] 2.718281828459045
```

```
 pi
```

```
## [1] 3.141592653589793
```

```
sin(pi/6)
```

```
## [1] 0.4999999999999999
```

Las funciiones `floor(x)`y `ceiling(x)`redondean hacia arriba y hacia abajo, respectivamente, hasta el entero más cercano.

```
floor(sin(pi/6))
```

```
## [1] 0
```

```
ceiling(sin(pi/6))
```

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

Variables

Tipos de datos

Definiendo objetos

```
x <- 100  # entero
```

```
(1 + 1/x)^x
```

```
## [1] 2.704813829421528
```

Actualizando el valor de la variable `x`

```
x <- 200
```

```
(1 + 1/x)^x
```

```
## [1] 2.711517122929317
```

Podemos almacenar el resultado de estas operaciones en una variable

```
y <- (1 + 1/x)^x
y
```

```
## [1] 2.711517122929317
```

Funciones

En matemática, una función necesita de uno o más argumentos (inputs) para producir uno o más outputs.

Para llamar a una función en R, se necesita escribir el nombre de la función, seguida de sus argumentos dentro de paréntesis y separados por comas.

La función `seq`genera una secuencia arismética.

```
seq(from = 1, to = 9, by = 2)
```

```
## [1] 1 3 5 7 9
```

Algunos argumentos son opcionales, y tienen valores predeterminador. Por ejemplo, si se omite el argumento `by`, R asume que `by = 1`.

```
seq(from = 1, to = 9)
```

```
## [1] 1 2 3 4 5 6 7 8 9
```

Para obtener más información sobre cualquier función o paquete inslatado, puede tipear `help(name)`o `?name`

```
help(seq)
```

```
## starting httpd help server ... done
```

Cada funciòn tiene un orden determinado para sus argumentos. En este sentido, si se ingresan argumentos en ese orden, no necesitan ser llamados. Por otro lado, se puede elegir el orden en el que se ingresan los argumentos; para esto, se le debe agregar el nombre de dicho argumento.

```
seq(1, 9, 2)
```

```
## [1] 1 3 5 7 9
```

```
seq(to = 9, from = 1)
```

```
## [1] 1 2 3 4 5 6 7 8 9
```

```
seq(by = -2, 9, 1)
```

```
## [1] 9 7 5 3 1
```

```
x <- 9
seq(1, x, x/3)
```

```
## [1] 1 4 7
```

Vectores

```r
(x <- seq(1, 20, by = 2))
```

```
##  [1]  1  3  5  7  9 11 13 15 17 19
```

```r
(y <- rep(3, 4))
```

```
## [1] 3 3 3 3
```

```r
(z <- c(y, x))
```

```
##  [1]  3  3  3  3  1  3  5  7  9 11 13 15 17 19
```

```r
(x <- 100:110)
```

```
##  [1] 100 101 102 103 104 105 106 107 108 109 110
```

```r
i <- c(1, 3, 2)
x[i]
```

```
## [1] 100 102 101
```

```r
j <- c(-1, -2, -3)
x[j]
```

```
## [1] 103 104 105 106 107 108 109 110
```

Creando un vector

```r
carreras = c("comunicacion para el desarrollo",
             "sociologia",
             "ciencia politica",
             "economia",
             "sistemas",
             "antropologia")  # c significa concatenar
```

¿Qué tipo de dato es?

```r
is.integer(carreras)
```

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

```r
is.character(carreras)
```

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

```r
is.factor(carreras)
```

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

Transformando el tipo de dato

```r
carreras1 = as.factor(carreras) # variables categóricas
carreras1
```

```
## [1] comunicacion para el desarrollo sociologia
## [3] ciencia politica                economia
## [5] sistemas                        antropologia
## 6 Levels: antropologia ciencia politica ... sociologia
```

```r
carreras2 = as.numeric(carreras1)
carreras2
```

```
## [1] 3 6 2 4 5 1
```

Verificando el tipo de dato

```
class(carreras1)
```

```
## [1] "factor"
```