

Introducción a R y Python

Miguel Ángel Orjuela Rocha



pythonTM



Universidad del
Rosario

UR INTERNATIONAL
SUMMER SCHOOL

Workshop
Big Data CO

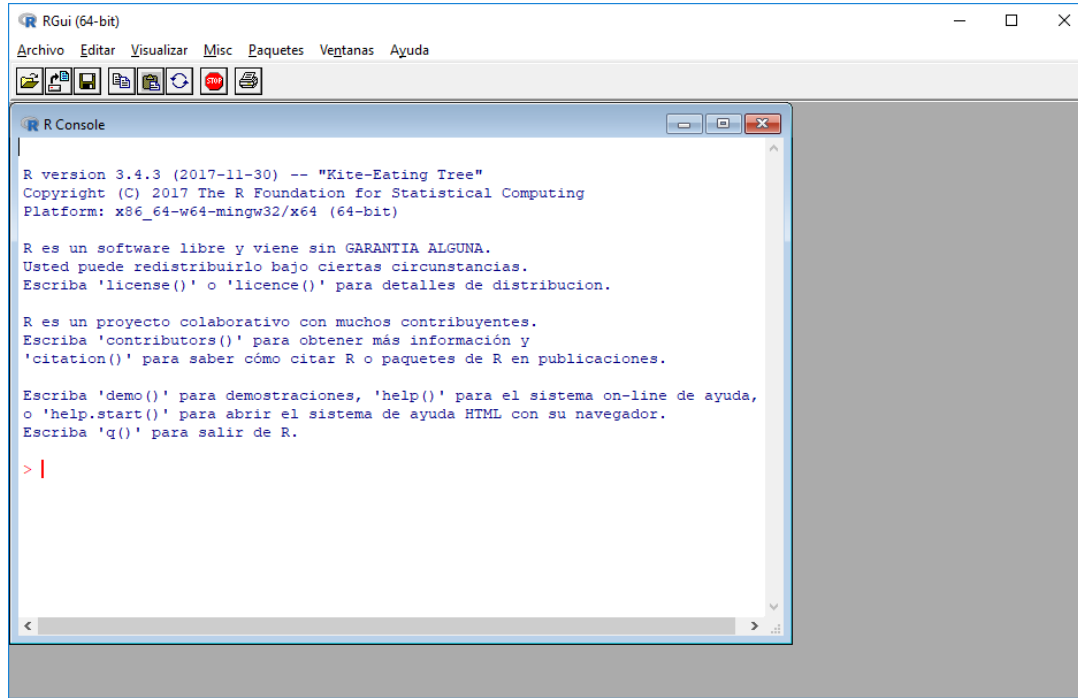
Mayo 30 a Junio 1 de 2018

Historia y descripción general de R

- R es un dialecto de S
- Creado en 1991 por Ross Ihaka y Robert Gentleman de la Universidad de Auckland (Nueva Zelanda)
- Actualmente el R Core Group controla el código fuente: <http://www.r-project.org/>
- Su licenciamiento es Open Source
- Se ejecuta en la mayoría de plataformas de computación y sistemas operativos
- Capacidades gráficas sofisticadas
- Filosofía: trabajo interactivo + desarrollo de nuevas herramientas

Inicializando R

R



The screenshot shows the RGui (64-bit) window. The title bar reads "RGui (64-bit)". The menu bar includes "Archivo", "Editar", "Visualizar", "Misc", "Paquetes", "Ventanas", and "Ayuda". Below the menu bar is a toolbar with icons for file operations. The main area is the "R Console", which displays the following text:

```
R version 3.4.3 (2017-11-30) -- "Kite-Eating Tree"
Copyright (C) 2017 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

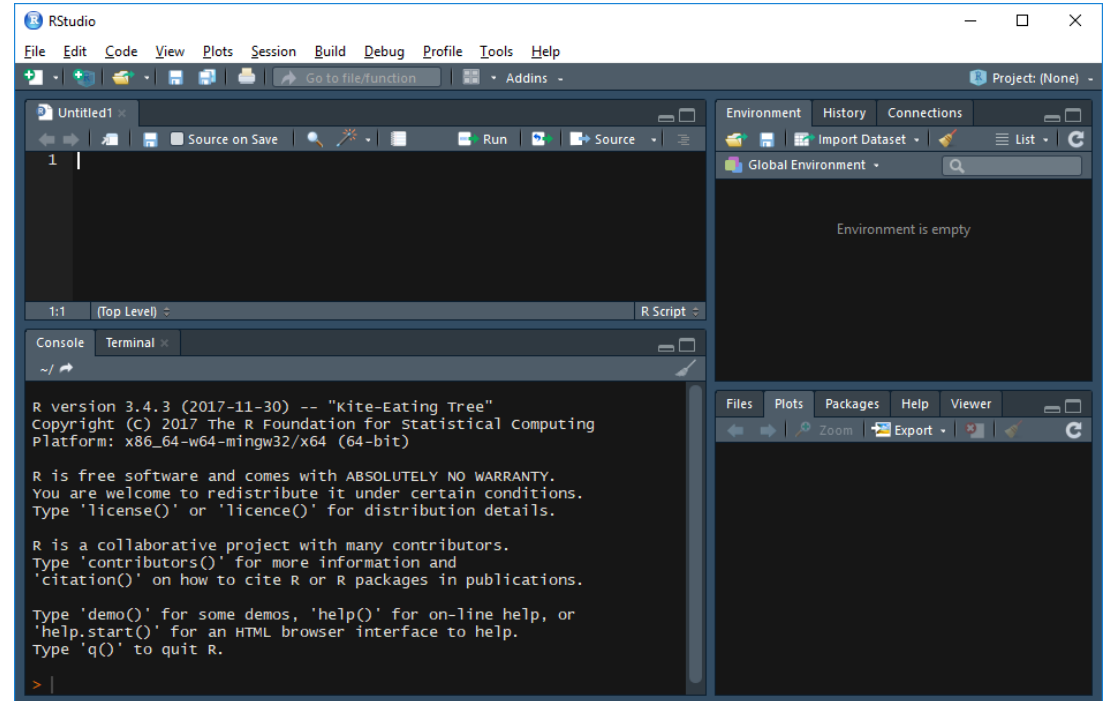
R es un software libre y viene sin GARANTIA ALGUNA.
Usted puede redistribuirlo bajo ciertas circunstancias.
Escriba 'license()' o 'licence()' para detalles de distribucion.

R es un proyecto colaborativo con muchos contribuyentes.
Escriba 'contributors()' para obtener más información y
'citation()' para saber cómo citar R o paquetes de R en publicaciones.

Escriba 'demo()' para demostraciones, 'help()' para el sistema on-line de ayuda,
o 'help.start()' para abrir el sistema de ayuda HTML con su navegador.
Escriba 'q()' para salir de R.

> |
```

RStudio



The screenshot shows the RStudio IDE. The title bar reads "RStudio". The menu bar includes "File", "Edit", "Code", "View", "Plots", "Session", "Build", "Debug", "Profile", "Tools", and "Help". Below the menu bar is a toolbar with icons for file operations and a "Go to file/function" search bar. The main area is the "Console", which displays the following text:

```
R version 3.4.3 (2017-11-30) -- "Kite-Eating Tree"
Copyright (C) 2017 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

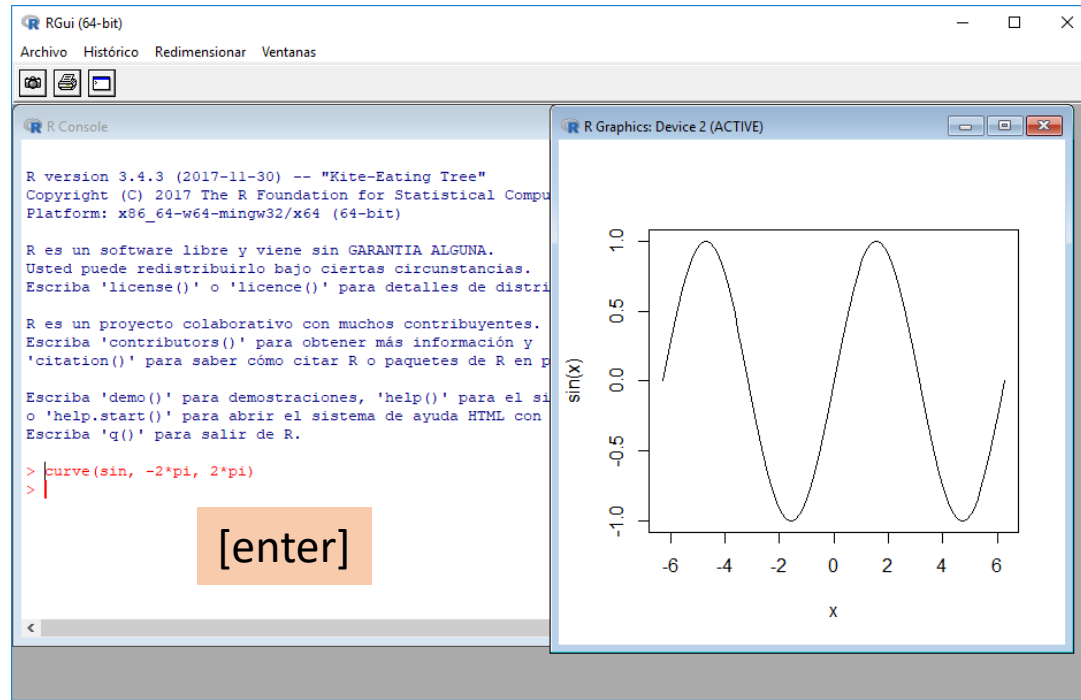
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

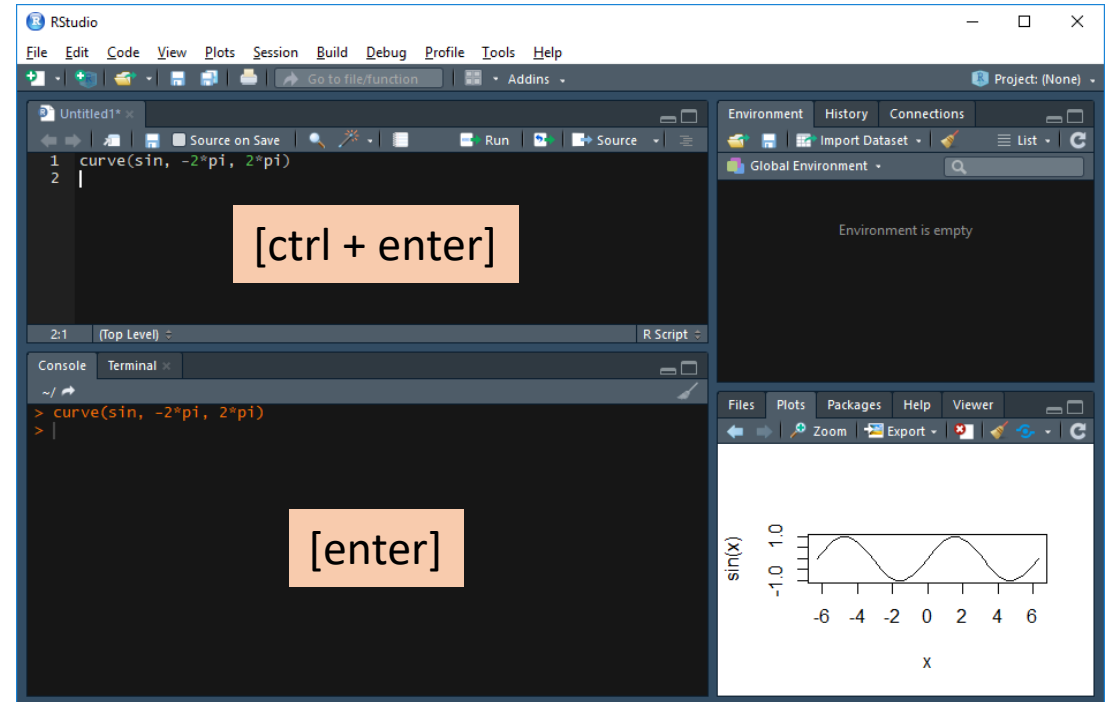
> |
```

Inicializando R

R



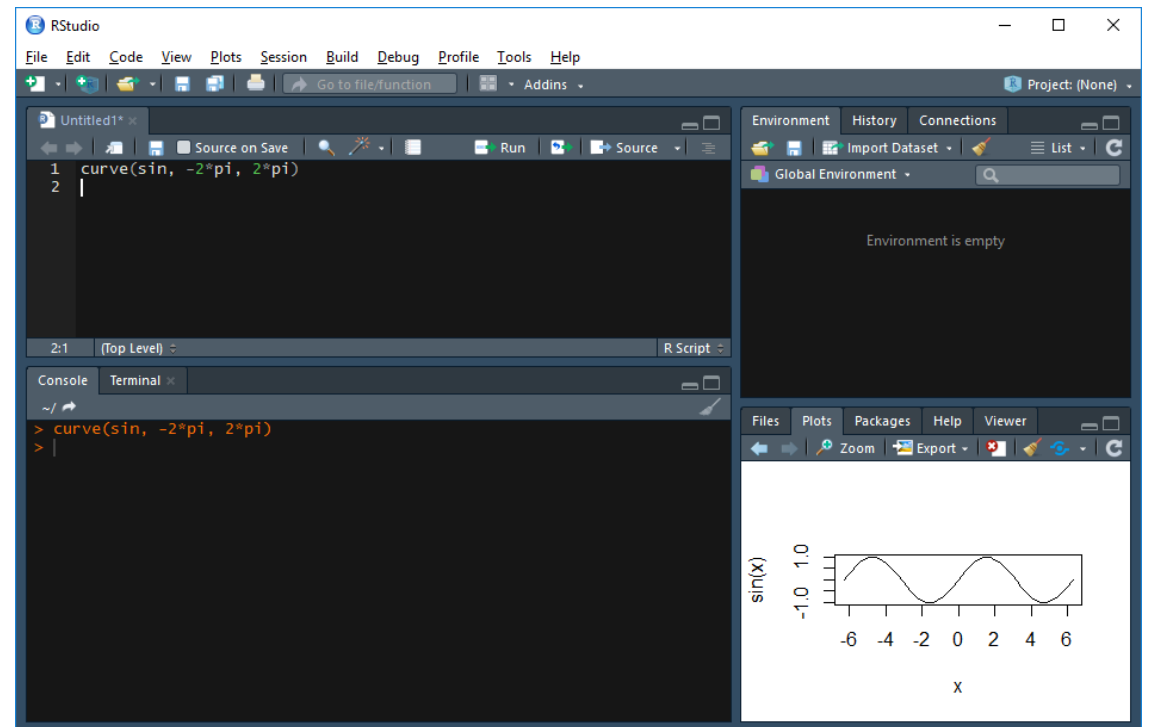
RStudio



Inicializando R

RStudio es un entorno de desarrollo integrado (IDE) para R.

Incluye una consola, editor de resaltado de sintaxis que admite la ejecución directa de código, así como herramientas para gráficos, historial, depuración y gestión del espacio de trabajo.



Componentes básicos de R

Ingresando expresiones:

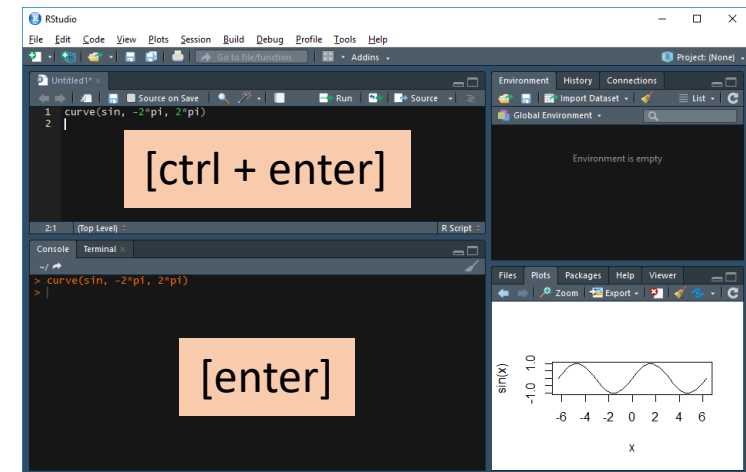
Al frente del prompt (>) se escriben expresiones

```
> curve(sin, -2*pi, 2*pi)
```

```
> demo(graphics)
```

```
> demo(persp)
```

```
> demo(image)
```



Las expresiones se ejecutan desde el editor con `[ctrl + enter]` y desde la consola con `[enter]`

Componentes básicos de R

Instalar paquetes

```
> install.packages("readr")  
> install.packages(c("dplyr", "ggplot2"))
```

Cerrar R

```
> q()
```

Operador de asignación (<-)

```
> x <- 1  
> print(x)  
> x  
> msg <- "Big Data CO 2018"  
> msg
```

Componentes básicos de R

Cómo nombrar variables

```
i_use_snake_case  
otherPeopleUseCamelCase  
some.people.use.periods
```

Notas:

R Reconoce mayúsculas de minúsculas

```
> r_rocks <- 2 ^ 3  
> R_rock  
> r_rocks
```

R sobrescribe variables

```
> x <- 2 ^ 3  
> x <- 10  
> x
```


Componentes básicos de R

Expresiones incompletas

```
> x <-
```

Evaluación de expresiones (autoprint)

```
> 2 + 2
```

```
> x <- 5
```

```
> x
```

```
> print(x)
```

Componentes básicos de R

Ver objetos almacenados en memoria

```
> ls
```

Ver contenido directorio actual

```
> dir()
```

Ver la ruta del directorio de trabajo

```
> getwd()
```

Cambiar dirección de directorio de trabajo

```
> setwd("home/user")
```

Componentes básicos de R

Borrar un objeto de la memoria

```
> rm("objeto")
```

Borrar todos los objetos cargados en memoria

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

Hacer comentarios...

```
> # Este es un comentario
```

Ejecutar Script

```
> source("Script.R")
```

Componentes básicos de R

Llamar funciones

```
> nombre_funcion(arg1 = val1, arg2 = val2, ...)
```

Función secuencia

```
> seq(from = 1 , to = 10)
```

```
> seq(1, 10)
```

Notas:

Las funciones pueden escribirse en una única línea de comando

```
> 2+3; 3-2; 5*2; 15/3; 2^5; sin(1)
```

Ayuda de la función: [F1]

Componentes básicos de R

Exponentes

> 3^2

> 3^(1/2)

Constantes importantes

> pi

> exp(1)

Objetos de R

R tiene cinco clases de objetos básicos o "atómicos":

- character
- numérico (números reales)
- entero
- complejo
- lógico (verdadero / falso)

Nota: Se puede conocer la clase del objeto `x` utilizando la función `class()`

```
> class(x)
```

Funciones de R

R es un lenguaje basado en funciones (como otros software estadísticos, e.g., Stata, Eviews, etc), y por tanto cada función debe estar identificada (nombrada) de manera única

Las funciones de R suelen estar definidas para una clase particular de objetos

Funciones de R

Función `c()` crea vectores de objetos concatenándolos juntos

El tipo más básico de objeto R es un vector. Un vector solo puede contener objetos de la misma clase.

```
> x <- c(0.5, 0.6) ## numérico
> x <- c(TRUE, FALSE) ## lógico
> x <- c(T, F) ## lógico
> x <- c("a", "b", "c") ## caracter
> x <- 9:29 ## entero
> x <- c(1+0i, 2+4i) ## complejo
```

Nota: Se puede conocer la clase del objeto `x` utilizando la función `class()`

```
> class(x)
```


Funciones de R

Suma y media

```
> sum(c(1, 3, -2))  
> mean(c(1, 3, -2))
```

Varianza y desviación estandar

```
> var(c(1, 3, -2))  
> sd(c(1, 3, -2))
```

Minimo y máximo

```
> min(c(1, 3, -2))  
> max(c(1, 3, -2))
```

Nota:

Se puede ahorrar sintaxis definiendo, e.g., `x <- c(1, 3, -2)`

Funciones de R

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

Correlación y covarianza

```
> cor(x, y)  
> cov(x, y)
```

Combinación de columnas y filas

```
> cbind(x, y)  
> rbind(x, y)
```

Funciones de R

Secuencias de números

```
> c(1:4) ; 3*c(1:4)
> seq(-5, 5, by=0.2)
> seq(length=51, from=-5, by=0.2)
```

Raíz cuadrada

```
> sqrt(c(1:4))
```

Mediana

```
> x <- c(8.97, 10.06, 9.29, 7.44, 9.48)
> median(x)
```

Producto de vectores elemento a elemento

```
> c(1:4) * c(4:1)
```

Transformación de objetos

numérico y carácter

```
> x <- c(1, 3, -2)
```

```
> is.numeric(x)
```

```
> as.character(x)
```

```
> x<-c("1", "3", "-2")
```

```
> is.character(x)
```

```
> as.numeric(x)
```

Nota: Estas transformaciones también pueden aplicarse a grandes volúmenes de datos.

Transformación de objetos

Prueba 1

```
> x <- c(1, "a", -2)
```

```
> class(x)
```

Prueba 2

```
> x <- 1:10 < 5 #Conjunto de valores lógicos
```

```
> class(x)
```

```
> is.character(x)
```

```
> as.numeric(x)
```

Otros objetos de R

Matrices

Son vectores con atributos de dimension (número de filas y columnas)

```
> m <- matrix(nrow = 2, ncol = 3)
```

```
> m
```

```
> dim(m)
```

```
> attributes(m)
```

Nota:

`attributes()` muestra los atributos de un objeto

Otros objetos de R

Llenado de matrices

```
> m <- matrix(1:6, nrow = 2, ncol = 3)
> m
```

Agregandole la dimension a un vector

```
> m <- 1:10
> m
> dim(m) <- c(2, 5)
> m
```

Combinación de columnas y filas

```
> x <- c(1,3,4,6,8)
> y <- c(2,3,5,7,9)
> cbind(x,y)
> rbind(x,y)
```

Otros objetos de R

Listas

Las listas son un tipo especial de vector que puede contener elementos de diferentes clases.

```
> x <- list(1, "a", TRUE, 1 + 4i)
```

```
> x
```


Otros objetos de R

Factores

Los factores se usan para representar datos categóricos y pueden estar desordenados u ordenados

Es como un vector de enteros donde cada entero tiene una etiqueta

Importantes en modelos estadísticos y son usados especialmente por funciones de modelos como `lm()` y `glm()`

```
> x <- factor(c("yes", "yes", "no", "yes", "no"))
```

Otros objetos de R

```
> x <- factor(c("yes", "yes", "no", "yes", "no"))  
> x  
> table(x)
```

¿Cómo R representa cada uno de los factores?

```
> unclass(x)
```

Otros objetos de R

Base de datos (data frame)

```
> x <- c(1, 3, -2)
> y <- c("a", "a", "b")
> z <- data.frame(x, y)
> z
> class(z)
> attributes(z)
```

Funciones de R

- Suma y media

```
> sum(c(1, 3, -2))
```

```
> mean(c(1, 3, -2))
```

- Varianza y desviación estandar

```
> var(c(1, 3, -2))
```

```
> sd(c(1, 3, -2))
```

- Mínimo y máximo

```
> min(c(1, 3, -2))
```

```
> max(c(1, 3, -2))
```

Nota:

Se puede ahorrar sintaxis definiendo, e.g., `x <- c(1, 3, -2)`

Funciones adaptables a objetos

Existen funciones que varían sus resultados dependiendo del tipo de objeto al que se aplica

Imprimir o mostrar

```
> x<-c(1, 3, -2)
> y<-c("a", "a", "b")
> print(x)
> print(y)
> print(paste("el primer elemento de      x      es", x[1]))
```

Nota: `paste` es una función muy útil para la presentación de resultados y el estudio basado en textos.

Funciones adaptables a objetos

Resumen

```
> x<-c(1, 3, -2)
> y<-factor(c("a", "a", "b"))
> summary(x)
> summary(y)
```

Rango

```
> range(x)
> range(y)
# R informa de manera explícita errores al aplicar
funciones que no se adaptan a una clase particular de
objeto
```

Operadores lógicos

Operador	Descripción
<	menor que
<=	menor o igual que
>	mayor que
>=	mayor o igual que
==	exactamente igual a
!=	diferente a
!x	no x
x y	x O y
x & y	x Y y
isTRUE(x)	prueba si x es cierto

Operadores lógicos

Ejemplo 1:

```
> x <- y <- 5  
> x < y  
> x <= y  
> x > y  
> x == y  
> x != y
```

Ejemplo 2:

```
> x <- 5 ; y <- 10  
> (x < 10 | y < 10)  
> (x < 10 & y < 10)
```


Combinando objetos

Adición de listas

```
> x<-1:3
```

```
> y<-101:103
```

```
> c(x, y)
```

Combinación por columnas y filas

```
> z<-cbind(x, y)
```

```
> w<-rbind(z, z)
```

Datos disponibles en R

<https://stat.ethz.ch/R-manual/R-devel/library/datasets/html/00Index.html>

```
> data(austres)
> ?austres
> View(austres)
> summary(austres)
> austres[1]
> austres[1:5]
```

Otras bases de datos disponibles:

<https://vincentarelbundock.github.io/Rdatasets/datasets.html>

Datos disponibles en R

```
> data(mtcars)
> ?mtcars
> View(mtcars)
> summary(mtcars)
> names(mtcars)
# Ver nombre de las columnas
> mtcars$mpg[1:10]
# Invocar una columna particular
> mtcars[mtcars$am==1]
# Cuál es el error?
> mtcars[mtcars$am==1,]
> mtcars[mtcars$am==1,2:5]
> length(mtcars)
```

Datos disponibles en R

```
> x<-mtcars[mtcars$am==1,]  
# Crear sub dataset  
> x  
> x$h cyl<-x$cyl>4  
# Crear nueva variable  
> x$h cyl<-as.numeric(x$h cyl)  
> x$h mpg<-as.numeric(x$mpg>mean(x$mpg))  
> x$h mpg<-NULL  
# Borra una variable de la base de datos
```

Funciones sobre bases de datos

Dimensión

```
> length(mtcars)
> length(mtcars$hp)
> dim(mtcars)
> rownames(mtcars)
> colnames(mtcars)
```

Indices (posiciones) y ordenamiento

```
> sort(mtcars$hp)
# Ordena los valores de la variable indicada
> order(mtcars$hp)
# Encuentra los índices o posiciones del vector ordenado
```

Crear funciones

```
> f <- function() {  
  }  
  
> ## Las funciones tienen su propia clase  
  
> class(f)  
[1] "function"  
  
> ## Ejecutar la función  
  
> f()  
  
NULL
```

Crear funciones

```
> f <- function() {  
    cat("Hello, world!\n")  
}  
  
> f()
```

Crear funciones

```
> f <- function(num) {  
  for(i in seq_len(num)) {  
    cat("Hello, world!\n")  
  }  
}  
  
> f(3)
```


Crear funciones

```
f <- function(num) {  
  hello <- "Hello, world!\n"  
  for(i in seq_len(num)) {  
    cat(hello)  
  }  
  chars <- nchar(hello) * num  
  chars  
}  
  
> meaningoflife <- f(3)  
> print(meaningoflife)
```

Estructuras de control

- If/else

```
if (...) {  
    "operaciones"  
} else {  
    "operaciones"  
}
```

- If/else anidado

```
if (...) {  
    "operaciones"  
} else if (...) {  
    "operaciones"  
} else {  
    "operaciones"  
}
```

Estructuras de control

Ejemplo 1:

```
>x<-10
>if (x>5) {
  x<-x/2
  y<-2*x
} else {
  x<-2*2
  y<-x
}
>x
>y
```

Ejemplo 2:

```
>x<-4
>if (x>5) {
  x<-x/2
  y<-2*x
} else {
  x<-2*2
  y<-x
}
>x
>y
```

Estructuras de control

- loops

Sintaxis general

```
for(i in I) {  
  "operaciones"  
}
```

Ejemplo 1

```
> for(i in letters[1:5]) {  
  print(i)  
}
```

Ejemplo 2

```
>x<-11:15  
>x  
> for(i in 1:5) {  
  x[i]=x[i]+1  
}
```

Estructuras de control

- **while**

Sintaxis general

```
while (...) {  
    "operaciones"  
}
```

Nota: Las operaciones se llevan a cabo hasta que la condición en (...) no se satisface.

Estructuras de control

Ejemplo 1:

```
>x<-80
>iter<-0
>while(x<100) {
  x<-x+sqrt(x)/10
  iter=iter+1
}
>x
>iter
```

Ejemplo 2:

```
>x<-80
>iter<-0
>while(x<100 & iter<20) {
  x<-x+sqrt(x)/10
  iter=iter+1
}
>x
>iter
```