



Universidad del
Rosario

| Educación
Continua

Módulo 2
DIPLOMADO EN CIENCIA DE DATOS



Universidad del
Rosario

Educación
Continua

Introducción a R y análisis estadístico

En este módulo:

Un primer contacto con R

Análisis estadístico de datos

Más sobre R



Un primer contacto con R

- R es un software libre y de código abierto (free and open-source) para análisis estadísticos
- Creado por Ross Ihaka y Robert Gentleman de la Universidad de Auckland (Nueva Zelanda), basado en el lenguaje S
- Actualmente es manejado por R-Project for Statistical Computing: <http://www.r-project.org/>
- Es compatible con Mac, Windows y Linux

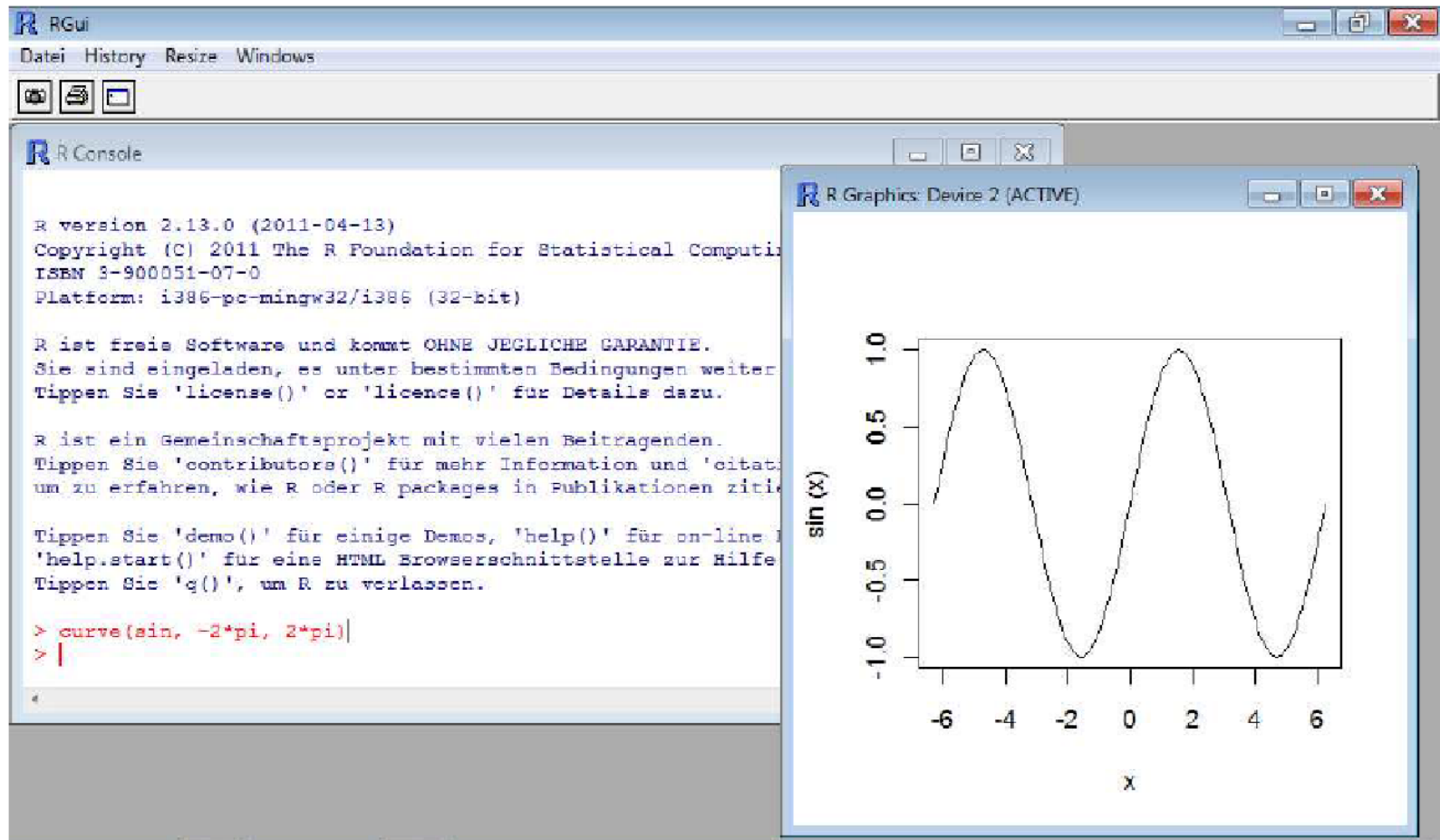


Un primer contacto con R

- Tiene un amplio número de recursos y actualizaciones en tiempo real
 - CRAN dispone de mas de 3500 librerías
 - <http://rseek.org/>
 - The R Journal <https://journal.r-project.org/>
- “Poderoso” y amigable para la realización de gráficas y mapas
- Compatible con otro tipo de lenguajes (traductores disponibles) para Matlab, S, SAS, Julia, ...



Consola de R



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



Rstudio IDE

- RStudio IDE es una plataforma de programación (integrated development environment) para R
- RStudio hace más amigable la interacción con R
- Se encuentra disponible en <http://www.rstudio.com/>
- Es compatible con Mac, Windows y Linux



Consola Rstudio

```
R version 3.0.0 (2013-04-03) -- "Masked Marvel"  
Copyright (C) 2013 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.  
  
> getwd()  
[1] "H:/MyData/RFiles"  
> 5*5  
[1] 25  
> A <- matrix(c(1,2,3,4,5,6,7,8), nrow=4, ncol=2)  
> A  
      [,1] [,2]  
[1,]    1    5  
[2,]    2    6  
[3,]    3    7  
[4,]    4    8  
> B <- matrix(c(1,2,3,4,5,6,7,8), nrow=4, ncol=2, byrow=TRUE)  
> B  
      [,1] [,2]  
[1,]    1    2  
[2,]    3    4  
[3,]    5    6  
[4,]    7    8  
>
```

Workspace History

Data	
A	4x2 double matrix
B	4x2 double matrix

Files Plots Packages Help

New Folder Delete Rename More

H: > MyData > RFiles

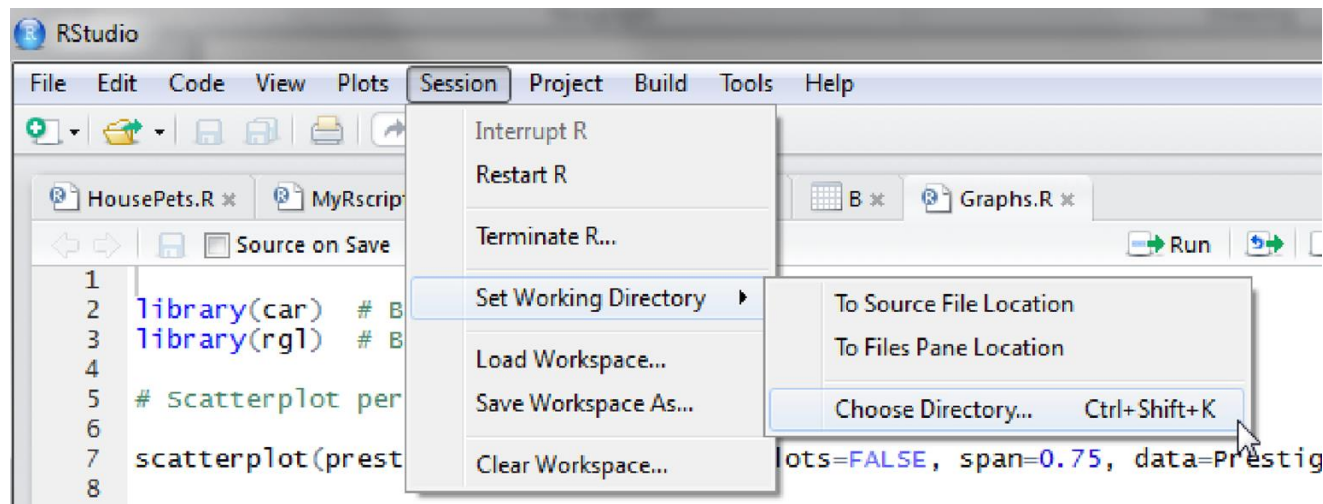
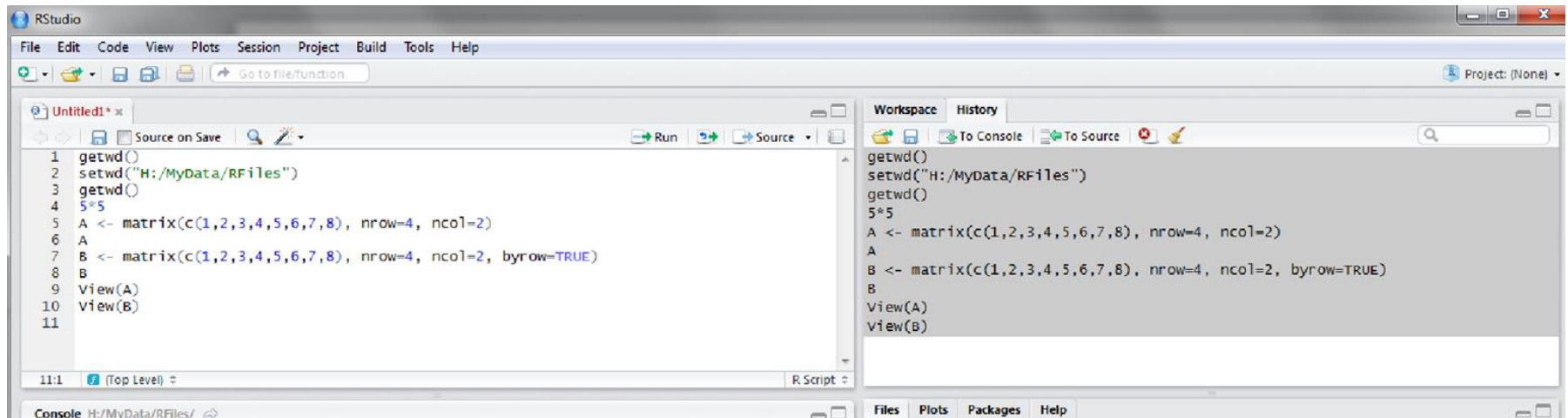
Name	Size	Modified
..		
.Rhistory	34 bytes	Aug 23, 2013, 1:26 PM



Universidad del
Rosario

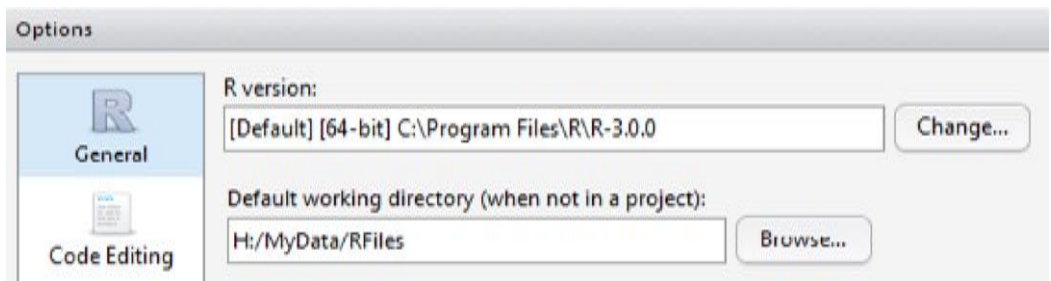
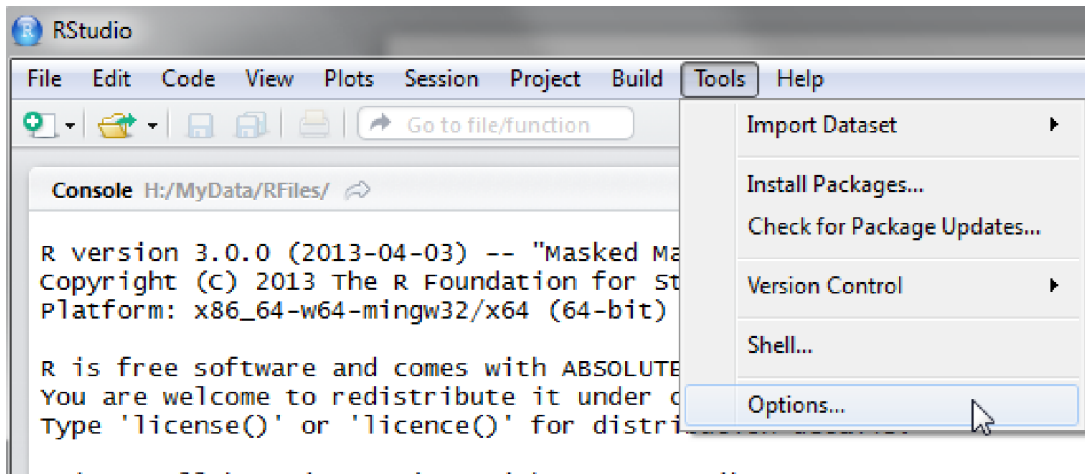
Educación
Continua

Consola Rstudio



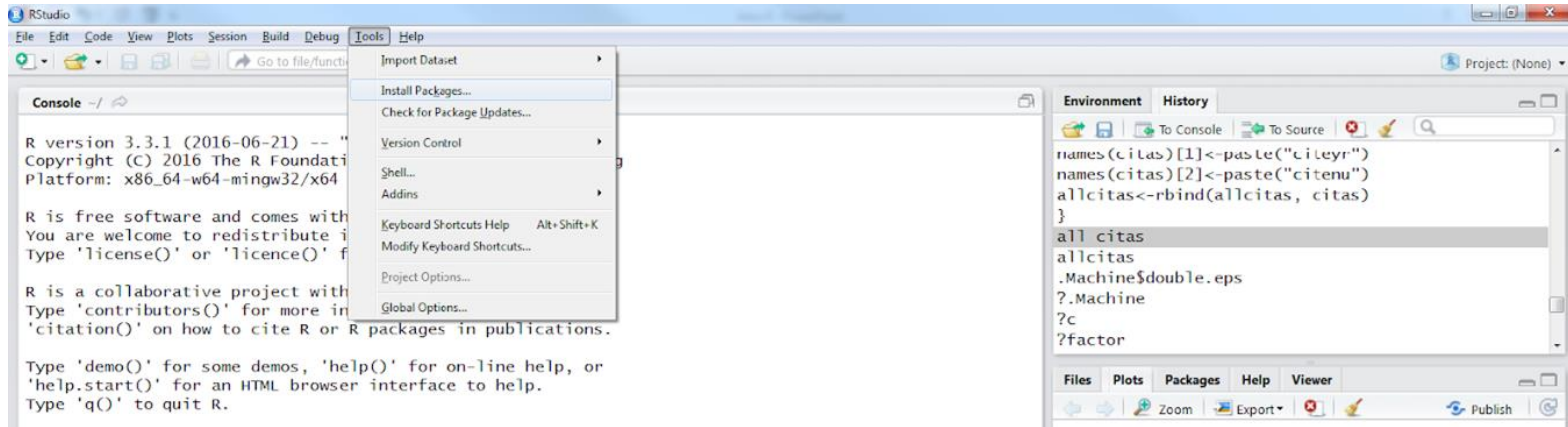


Consola Rstudio





Consola Rstudio



Instalemos el paquete rgl, es útil para gráficas en 3D



Conociendo el ambiente R

Nota: Para correr en Rstudio, se utiliza la combinación de teclas Ctr+Enter. Desde la consola basta con Enter

- Demos

```
> demo()  
> demo(graphics)  
> demo(persp)  
> demo(image)
```



Conociendo el ambiente R

- Instalación de paquetes

```
> install.packages("pkg1")  
> install.packages(c("pkg1",  
"pkg2"))
```

- Cerrar R

```
> q()
```

- Definir un objeto

```
> objeto<-...  
> x<-2
```



Conociendo el ambiente R

- Ver objetos
 `> ls()`
- Ver contenido directorio actual
 `> dir()`
- Ver dirección actual de directorio
 `> getwd()`
- Cambiar dirección actual de directorio
 `> setwd("/home/user")`



Conociendo el ambiente R

- Sintaxis general
 - > objeto<-función(argumentos)
- Ayuda
 - > ?función
 - > help(función)
- Cargar Librería
 - > library("nombre")
- Lista de funciones en una librería
 - > library(help="nombre")



Conociendo el ambiente R

- Ejecutar Script auxiliar

```
> source("Script.R")
```
- Borrar un objeto

```
> rm("objeto")
```
- Borrar todos los objetos en "Environment"

```
> rm(list=ls())
```
- Hacer comentarios...

```
> # Hola!!!!
```



Cálculos básicos

- Suma y resta

- > $3+2$

- > $3-2$

- Multiplicación y división

- > $3*2$

- > $3/2$



Cálculos básicos

- Exponentes
 - > 3^2
 - > $3^{(1/2)}$
- Constantes importantes
 - > π
 - > $\exp(1)$

Nota: Las funciones pueden escribirse en una única línea de comando

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



Valores especiales en R

- Infinito

- > Inf

- > 1+Inf

- Nivel de precisión (Machine epsilon)

- > .Machine\$double.eps

- > ?Machine

- > 0>.Machine\$double.eps



Manipulación de variables

- Definir valores particulares dos variables, e.g., x , y :
 - > $x < -3$ ($x = 3$)
 - > $y < -2$ ($y = 2$)
- Cálculos a partir de x , y
 - > $x + y$
 - > $x * y$

Nota: R es case-sensitive, es decir distingue entre x y X .

Nota: R sobre-escribe cuando se asigna un nuevo valor a un objeto ya creado.



Funciones en R

R es un lenguaje que emplea funciones (como otros software, e.g., Stata, Eviews, Python, etc.), donde cada función esta identificada (nombrada) de manera única

Una función básica permite combinar elementos (números, caracteres, cadenas de texto -strings-) en un único objeto o **lista** (elementos del mismo tipo)

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

```
> c("a", "a", "b", "b", "a")
```



Funciones básicas

- 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: Lo anterior se puede simplificar definiendo un objeto, e.g., `x<-c(1, 3, -2)`.



Funciones básicas

- Definir objetos
 - > `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)`



Otras funciones importantes

- Sucesiones 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 y ordenamiento

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

- Producto de vectores elemento a elemento

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



Programación orientada a objetos

R está basado en la manipulación de objetos que están asociados a una clase (“class”) que indica el tipo de objeto que representa

Las funciones de R están definidas para una clase particular de objetos

Por ejemplo, no se pueden sumar ni restar cadenas de texto



Clases de objetos

- Numéricos

```
> x<-c(1,3,4,6,8)
> x
> class(x)
```

- Enteros

```
> x<-c(1L,3L)
> x
> class(x)
```

- Caracteres

```
> x<-c("a","a","b")
> x
> class(x)
```



Clases de objetos

- Factores o categorías

```
> x<-factor(c("a","a","b"))  
> ?factor  
> x  
> class(x)
```

- Matrix

```
> x<-c(1,3,-2)  
> y<-c(2,1,6)  
> z<-cbind(x,y)  
> z  
> class(z)
```

Clases de objetos

- Base de datos (Data frame)

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

```
> y<-c("a", "a", "b")
```

```
> z<-data.frame(x,y)
```

```
> z
```

```
> class(z)
```



Transformación de objetos

- Numérico vs carácter

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

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

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

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

```
> is.character(y)
```

```
> as.numeric(y)
```

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



Transformación de objetos

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

```
> class(x)
```

```
# en este caso los datos se conforman  
como carácter
```



Operadores lógicos

- Los operadores lógicos son aquellos que arrojan como resultado “verdadero” o “falso”
- Son un tipo de objeto útil para determinar qué funciones o líneas de comando se deben ejecutar de acuerdo con la satisfacción de una condición



Principales operadores lógicos

Operador	Función
<	Menor que
>	Mayor que
<=	Menor o igual que
>=	Mayor o igual que
==	Igual a
!=	Diferente a
!x	“no” x
x y	x o y
x & y	x y y



Uso de 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)
```


Tipo lógico

```
> x <- 1:10 < 5 ; x  
# Conjunto de valores lógicos  
> class(x)  
> is.character(x)  
> is.numeric(x)  
> as.numeric(x)
```



Tipo lógico

```
> !x
```

```
# Negación de x
```

```
> which(x)
```

```
# Muestra los índices (posiciones) en  
los que x toma el valor "True"
```

Tipo lógico

- Algunas operaciones de matrices

```
> y <- c(TRUE, TRUE, FALSE, FALSE)
```

```
> as.numeric(y)
```

```
> y & TRUE
```

```
> y & FALSE
```

```
> !y
```

```
> y | FALSE
```



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



Vectores y matrices

- Vectores

```
> x <- 1:10; names(x) <- letters[1:10]; x
> x[1:5]
> x[c(2,3,5)]
> x[c("b", "d")]
# Invoca elementos particulares

> y <- rep(0,3)
# Repetir valores en un vector
> x<-array(1:60, c(5,4,3))
# Arreglo matricial
```



Vectores y matrices

- Matrices

```
> x <- matrix(1:30, 3, 10)
> class(x)
> x[1:2,]
> x[,1:2]
> dim(x)
> nrow(x); ncol(x)
> as.vector(x)
# Vectorización
```

Nota: Una matrix puede contruirse con cualquier tipo de objetos, pero todos los elementos deben ser del mismo tipo



Vectores y matrices

- Algunas operaciones de matrices

```
> x <- matrix(1:30, 3, 10)
> x+1; x*2; x/3; exp(x)
# Operaciones elemento a elemento
> y <- 1:3
> x*y
# Multiplicación elemento a elemento
> z <- 1:10
> x%*%z
# Multiplicación matricial
```




Vectores y matrices

- Algunas operaciones de matrices

```
> x <- matrix(1:30, 3, 10)
```

```
> is.matrix(x)
```

```
> t(x)
```

```
# Transposición de matrices
```

```
> diag(x)
```

```
# Diagonal de x
```

```
> diag(diag(x))
```

```
> solve(x)
```

```
# Matriz diagonal
```

```
> x%*%solve(x)
```



Vectores y matrices

- Algunas operaciones de matrices

```
> y <- matrix(runif(9), 3, 3)
```

```
> solve(y)
```

```
> y %*% solve(y)
```

```
> solve(y, rep(1, 3))
```

```
> y %*% solve(y, rep(1, 3))
```

Vectores y matrices

- Algunas operaciones de matrices

```
> det(x)
```

```
# Determinante de la matriz
```

```
> x <- matrix(c(13, -4, 2, -4, 11, -2, 2, -2, 8), 3,  
3,  
byrow=TRUE)
```

```
> eigen(x)
```

```
> ev <- eigen(x)
```

```
> vectors <- ev$vectors
```

```
# Valores propios y vectores propios
```

Ver en qué consiste "value" dentro de una función: e.g., ?eigen

```
> x <- matrix(rexp(100, rate=.1), ncol=10)
```

```
# Matrices de números aleatorios siguiendo  
distribución exponencial
```

```
> x <- matrix(runif(100), ncol=10)
```



Concatenando objetos

- Concatenación de listas

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

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

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

- Combinación por columnas y filas

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

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



Listas

Las listas contienen objetos de diferente tipo que pueden ser referenciados como `$"nombre"` o `[["numero"]]`. Dichas listas también pueden anidarse

```
> x<-list(One=11:15, Two=c("a", "b",  
    "c"), Three=(1:4)); x
```

```
> y<-list(x=x, Four=1:3); y
```

```
> x$One; y$x$One
```

```
> y$Four; y[[2]]
```



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 Datasets

- **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



Funciones sobre Datasets

- Indices (posiciones) y ordenamiento

```
> mtcars2 <- mtcars[order(mtcars$mpg), ]  
# Crea una nueva base de datos ordenado por mpg
```

```
> mtcars2 <- mtcars[order(-mtcars$mpg), ]  
# Crea una nueva base de datos ordenada,  
decreciente, por mpg
```



Funciones sobre Datasets

- Aplicar función a múltiples columnas

```
> mtcars[1:3,1:4] # como obtenemos la media  
de cada  
columna o fila?
```

```
> mean(mtcars[1:3,1:4])  
> mean(mtcars$mpg)  
> apply(mtcars[1:3,1:4],1, mean)  
> apply(mtcars[1:3,1:4],2, mean)
```

Nota: `apply` permite aplicar una importante serie de funciones de manera recursiva

```
> apply(mtcars[1:3,1:4],2, summary)  
> apply(mtcars[1:3,1:4],1:2, function(x) x/2)
```



Funciones sobre Datasets

- Generación de estadísticas agregadas

```
> aggregate(mtcars[,1:4], by=list(mtcars$cyl),  
FUN=mean)
```

```
> aggregate(mtcars[,1:4], by=list(mtcars$cyl),  
FUN=summary)
```

- Muestreo aleatorio de un dataset

```
> mtcars[sample(1:length(mtcars[,1]),30),]  
# submuestra de tamaño 30 sin reemplazamiento
```

```
> mtcars[sample(1:length(mtcars[,1]),30,  
replace=TRUE),]  
# submuestra de tamaño 30 con reemplazamiento
```



Ejercicio

Escribir un Script que permita realizar las siguientes operaciones:

- Generar un vector columna de tamaño 50 que distribuya normal con media 10 y varianza 5.
- Generar un vector columna de tamaño 20 con números consecutivos desde -5 con saltos de 1.5
- Combinar los 2 vectores anteriores en un vector columna
- Generar un vector columna que siga una distribución uniforme y pueda combinarse como una columna adicional del objeto anterior
- Del objeto combinado de 2 columnas, extraer la parte que corresponden a valores entre 0,2 – 0,4 y 0,6 -0,8 de la columna 2
- Imprimir una frase que presente la media y desviación estándar de la primera columna



Ejercicio

```
> x<-rnorm(50, 10, sqrt(5))
> y<-seq(length=20, from=-5, by=1.5)

> z<-c(x,y)
# La función rbind es adaptable a objetos:
> class(x); class(y)
> rbind(x,y)
> rbind(as.matrix(x),as.matrix(y))

> w<-cbind(z, runif(70))

> w<-w[((w[,2]>0.2 & w[,2]<0.4) | (w[,2]>0.6 & w[,2]<0.8)),]
> print(paste("La media de la columna 1 es",          mean(w[,1]),
"y la desviación estandar es",          sd(w[,1]))))
```