Introducción al Análisis Exploratorio de Datos (EDA) en R

Módulo 1

2024-03-16

R y RStudio

El entorno R

R es un entorno de programación para el análisis de datos y gráficos (R Core Team, 2000). Algunas características de R son las siguientes:

- Permite el almacenamiento y la manipulación de datos.
- Incluye una amplia colección integrada de herramientas para el análisis de datos.
- Dispone de un lenuaje de programación interpretado, simple y efectivo que incluye condicionales, ciclos, funciones recursivas, etc. (Muchas de las funciones suministradas en el Sistema están escritas en lenguaje R).
- La funcionalidad de R consiste en paquetes modulares.

¿Por qué R?

R tiene las siguientes ventajas:

- R es un software libre
- R es multiplataforma
- R tiene una **sofisticada capacidad para hacer gráficos**, incluyendo paquetes gráficos especializados.
- R tiene librerías que cubre un amplio rango de la metodologías de la estadística y las matemáticas (series temporales, optimización matemática, inferencia estadística, etc.)
- Existe una comunidad activa que ha promovido el incremento en su número de usuarios (The R Project, R Contributor Site, RStudio Community y R Bloggers).

R y RStudio

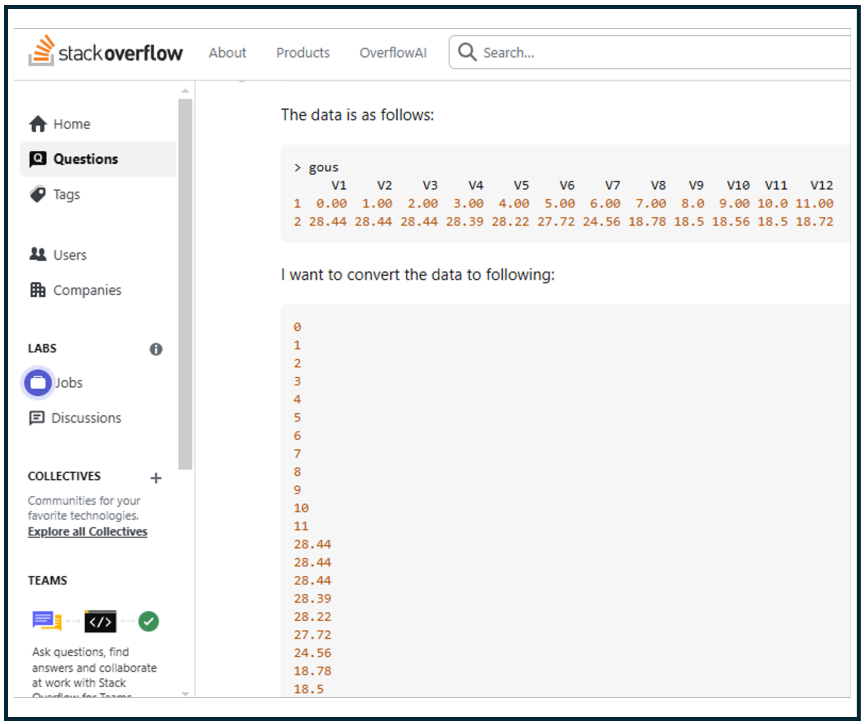
RStudio es un **IDE (Integrated Development Environment)** para el lenguaje de programación R.

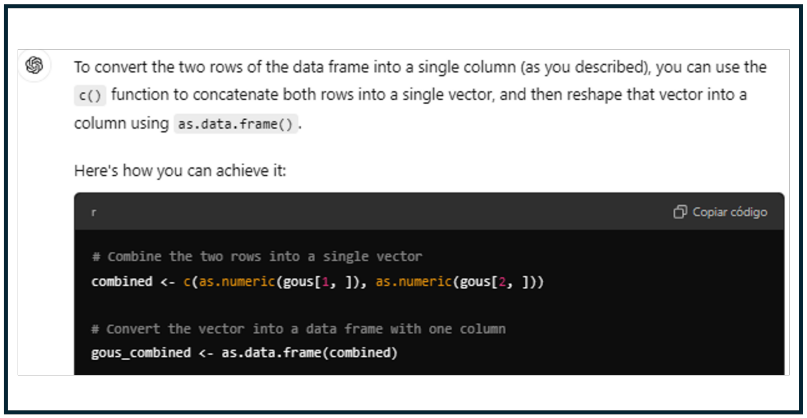
Para instalar R: https://cran.r-project.org/

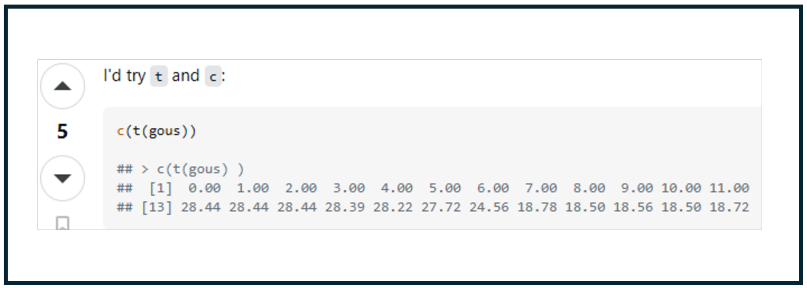
Para instalar RStudio: http://www.rstudio.com/download

RStudio facilita el trabajo con R a través de una interfaz que es común a Windows, Mac OS y Linux.

R y herramientas de IA







Otros recursos. . .

Posit Community

stackoverflow

StackExchange

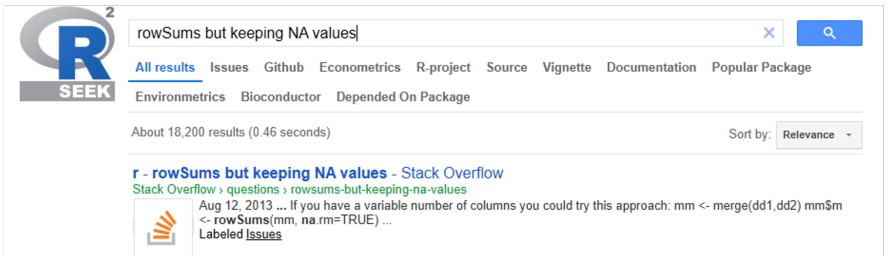
R-bloggers

StackExchange [r] Data Science

StackExchange Search on Cross Validated... CrossValidated

RPubs by RStudio

GitHub



Estructuras de datos en R

Estructuras de datos en R

En lo sucesivo, nos concentraremos en las estructuras de datos representadas en la **Figura 1**. (La figura omite los arreglos, `array()`, que es el resultado de aumenta la dimensión de una matriz).

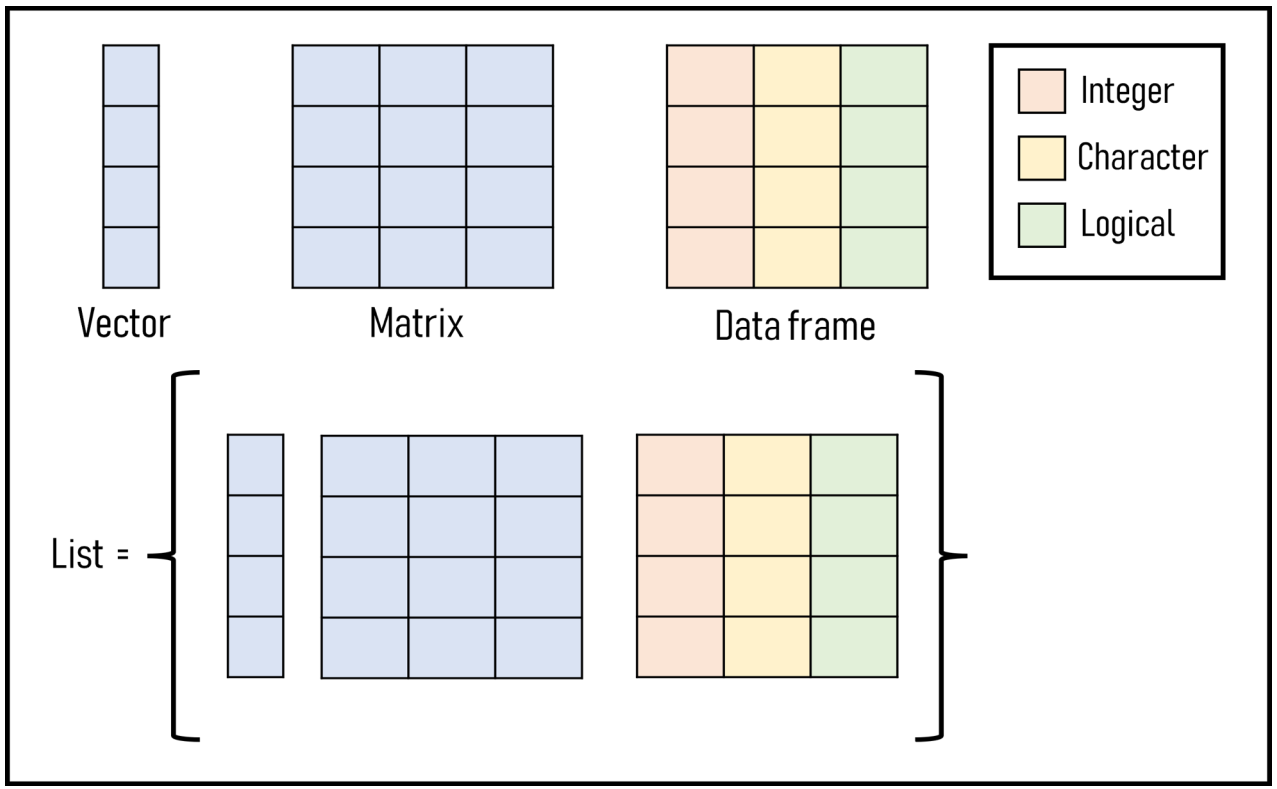


Figure 1: Estructuras principales de datos en R

Tipos de vectores

La función `class()` permite recuperar alguna de las siguientes clases

- Cadena de caracteres
- Numérico (*double*)
- Entera
- Lógico

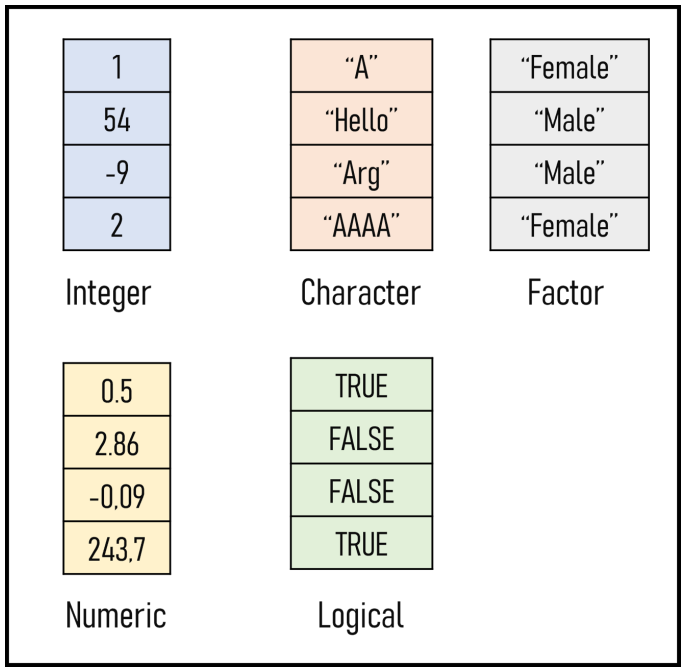


Figure 2: Clases de vectores

Vectores

El **operador de asignación** <- es usado para asignar un nombre al objeto. Para definir vectores:

```
v1 <- c("a", "b", "c")
v1
```

```
## [1] "a" "b" "c"
```

Para seleccionar el elemento *i*:

```
v1[3]
```

```
## [1] "c"
```

Recuerde que

- `length()` muestra la longitud
- `typeof()` muestra el tipo de vector
- `names()` nombra los elementos
- R usa **ejecución por elementos**

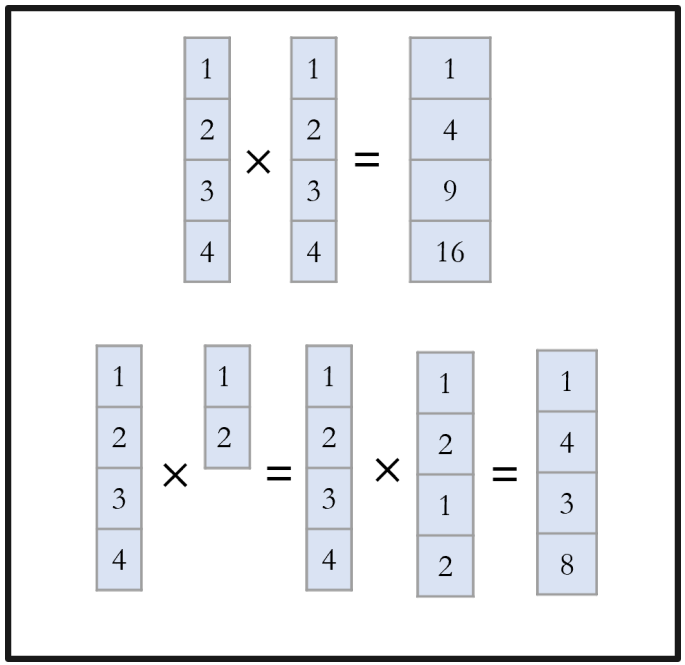


Figure 3: Ejemplos de ejecución por elementos

Matrices

Una matriz se define usando `matrix()`. El parámetro `byrow` determina si las entradas son completadas por filas (`TRUE`) o columnas (`FALSE`). Así

```
m1 <- matrix(c(1,2,3,5), nrow = 2, ncol = 2, byrow = T)
m1
```

```
##      [,1] [,2]
## [1,]    1    2
## [2,]    3    5
```

Las entradas son seleccionadas con `[i, j]`:

```
m1[2,2] # Fila 2 y columna 2
m1[2,]  # Fila 2
m1[ ,2] # Columna 2
```

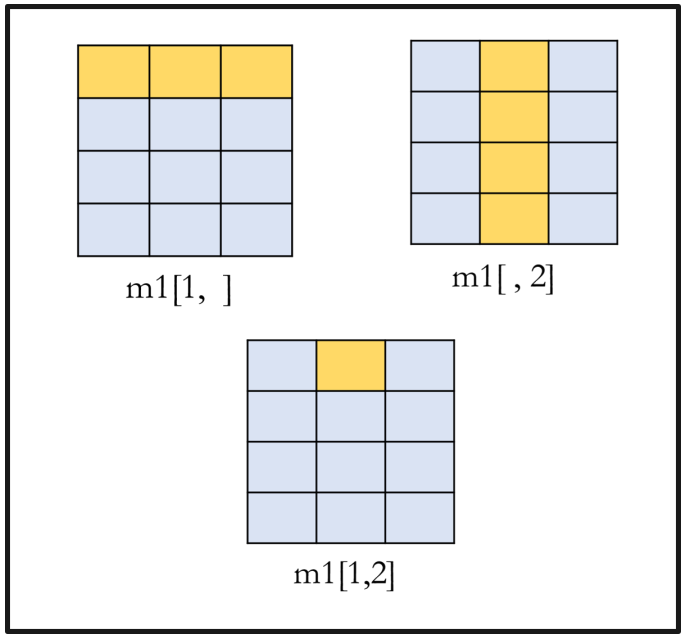


Figure 4: Uso de índices en matrices

Funciones importantes: `rbind` (permite agregar filas), `cbind` (permite agregar columnas, `dim` (proporciona las dimensiones), `diag` extrae elementos de la diagonal, `nrow` y `ncol()` (proporciona el no. de filas y columnas, respectivamente)

Nota: operaciones con matrices

A diferencia de la ejecución por elemento a*b, el producto de matrices se obtiene de

```
m1 %*% t(m1) # m1 por su traspuesta
```

```
##      [,1] [,2]
## [1,]    5   13
## [2,]   13   34
```

La **Figura 5** muestra otras operaciones útiles usando matrices.

Operación	Descripción
dim(m1)	Dimensión de m1 (n x m)
m1 + m2	Suma
m1 – m2	Resta
t(m1)	Traspuesta de M1
2*m1	Multiplicación por escalar
m1 %*% m2	Multiplicación de matrices
det(m1)	Determinante de m1
solve(m1)	Inversa de m1
diag(m1)	Diagonal de m1

Figure 5: Operaciones con matrices

Data frames

Los data frames son estructuras rectangulares de datos que pueden contener objetos de diferente tipo (cadena, numéricos, lógicos, etc.). Creamos un data frame con tres columnas (*id*, *sexo* y *edad*):

```
id = 1:4
sexo = factor(c("male", "male", "female", "female"))
edad = c(15, 26, 43, 56)
df = data.frame(id, sexo, edad)
df
```

```
##   id   sexo edad
## 1  1   male   15
## 2  2   male   26
## 3  3 female   43
## 4  4 female   56
```

Al igual que una matriz, se seleccionan sus entradas usando `[i,j]`. También:

```
df$edad              # seleccionar variable edad
df$edad[1]           # primer elemento de edad
df[c("id", "edad")]  # seleccionar las variables id y edad
```

Para verificar los nombres de las variables usamos `colnames()`.

Listas

Las listas son estructuras heterogéneas de datos. Aunque son estructuras unidimensionales, las listas permiten almacenar objetos de distinta clase (vectores, matrices, data frames, otras listas). Así:

```
lista <- list(vector1 = v1,
              matriz = m1, dataframe = df)
lista
```

```
## $vector1
## [1] "a" "b" "c"
##
## $matriz
##      [,1] [,2]
## [1,]    1    2
## [2,]    3    5
##
## $dataframe
##   id    sexo edad
## 1  1    male   15
## 2  2    male   26
## 3  3  female   43
## 4  4  female   56
```

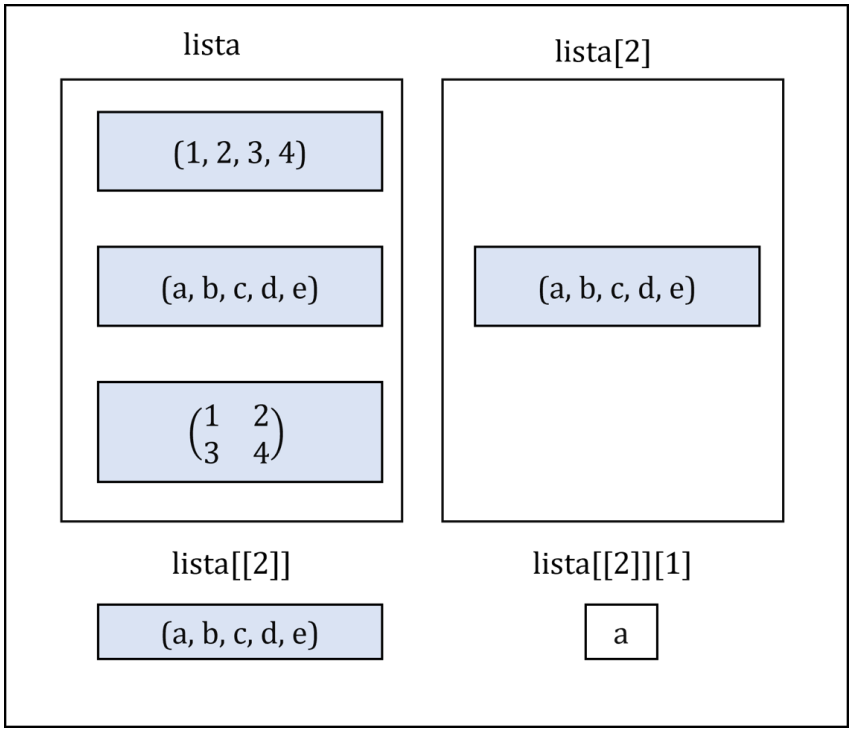
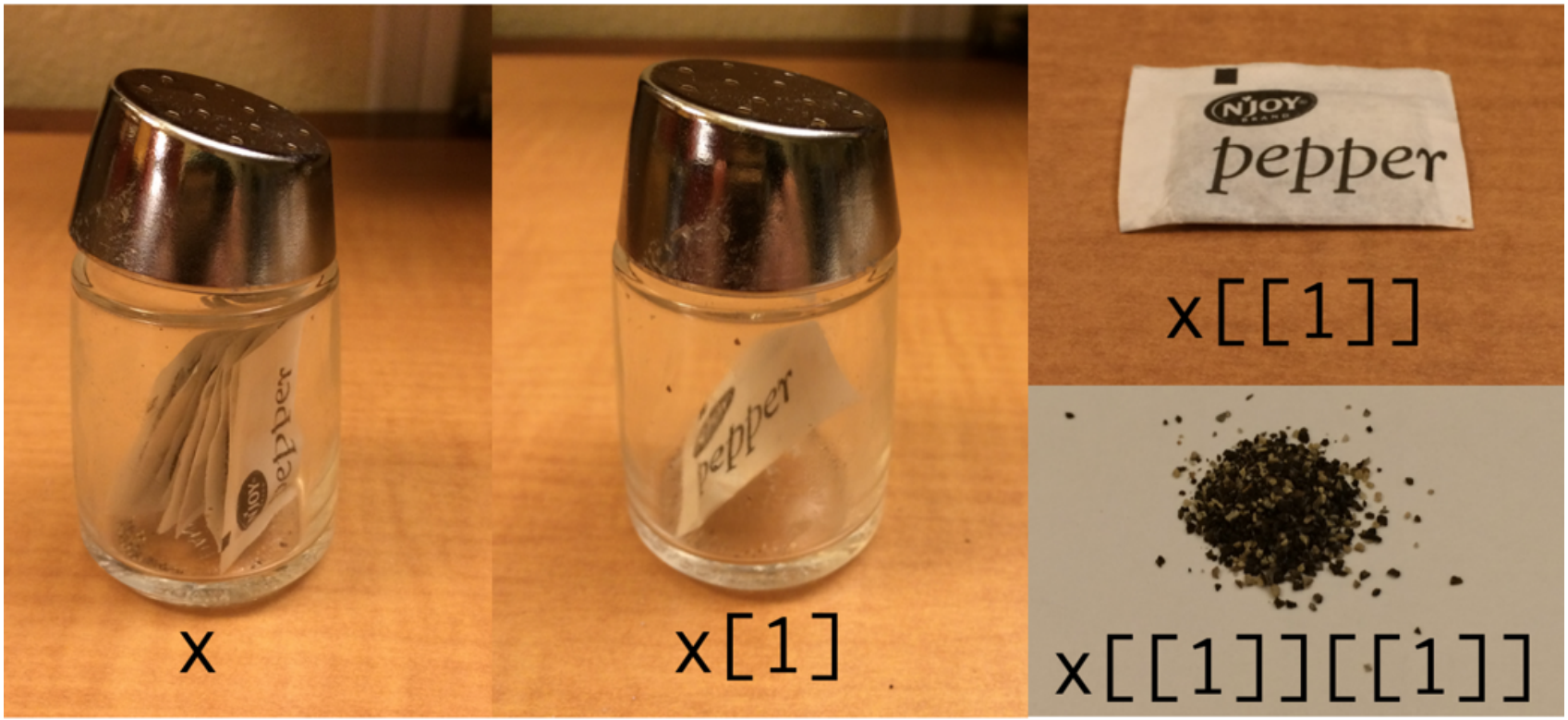


Figure 6: Uso de índices en listas

Indexando listas



Condicionales y funciones

Condicionales

Usando condicionales, una operación es ejecutada si la condición se cumple (`TRUE`); de otro modo (`ELSE`), establece la operación que es ejecutada si la condición no se cumple (`FALSE`). La estructura general es la siguiente:

```
if (condition) {
  # Ejecutado cuando la condición es verdadera
} else {
  # Ejecutado cuando la condición es falsa
}
```

Operator	Description
$x > y$	Greater than
$x >= y$	Greater than or equal to
$x == y$	Exactly equal to
$!x$	Not
$x != y$	Not equal to
$x \mid y$	OR
$x \& y$	AND
$x \ \%in\% \ y$	In the set

Figure 7: Descripción de operadores

Funciones

En general, la estructura de una función es la siguiente:

```
function(arg_1 = x1, arg_2 = x2, ..., arg_n = xn)
```

A partir de `function()`, se pueden crear funciones con base en la siguiente estructura:

```
my_function = function(args){
  #statement
  #statement
  #statemente
  return(y)
}
```

Cuando se usan funciones definidas en una librería, se pueden conocer los detalles de los argumentos y la salida de la funciones con los comandos `??`y `help()`.

Ejercicio

1. Usando la base de datos *cars*, cree un vector que contenga los valores de la variable *speed*. Calcule la media y la desviación estándar sobre el vector (funciones 'mean()' y 'sd()').
2. Crear una función que, si la entrada es un vector numérico, la salida sea la media, la desviación estándar, la mediana, el percentil 25 y 75. (Para la salida use una lista).

Recursos alternativos

Recursos alternativos

- La librería `swirl` proporciona un tutorial sobre elementos básicos en R

```
install.packages("swirl")
library (swirl)
swirl()

# Para cerrar el tutorial:
bye()
```

- Data wrangling with dplyr and tidyr (Cheat Sheet): Recurso 1.2
- Visualización de datos usando ggplot2 (Guía Rápida): Recurso 1.3
- Factors with forcats (Cheat Sheet): Recurso 1.4
  \end{itemize}

Bibliografía de consulta

Bibliografía de consulta

- Grolemund, G. (2014). Hands-On Programming with R. O'Reilly Media: Sebastopol, CA.
- Schutt, R. & O'Neil, C. (2014). Doing Data Science. O'Reilly Media: Sebastopol, CA.
- Wickham & Grolemund, G. (2016). R for Data Science: Import, Tidy, Transform, Visualize, and Model Data. O'Reilly Media: Sebastopol, CA.