# 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.

# Elementos básicos de 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).

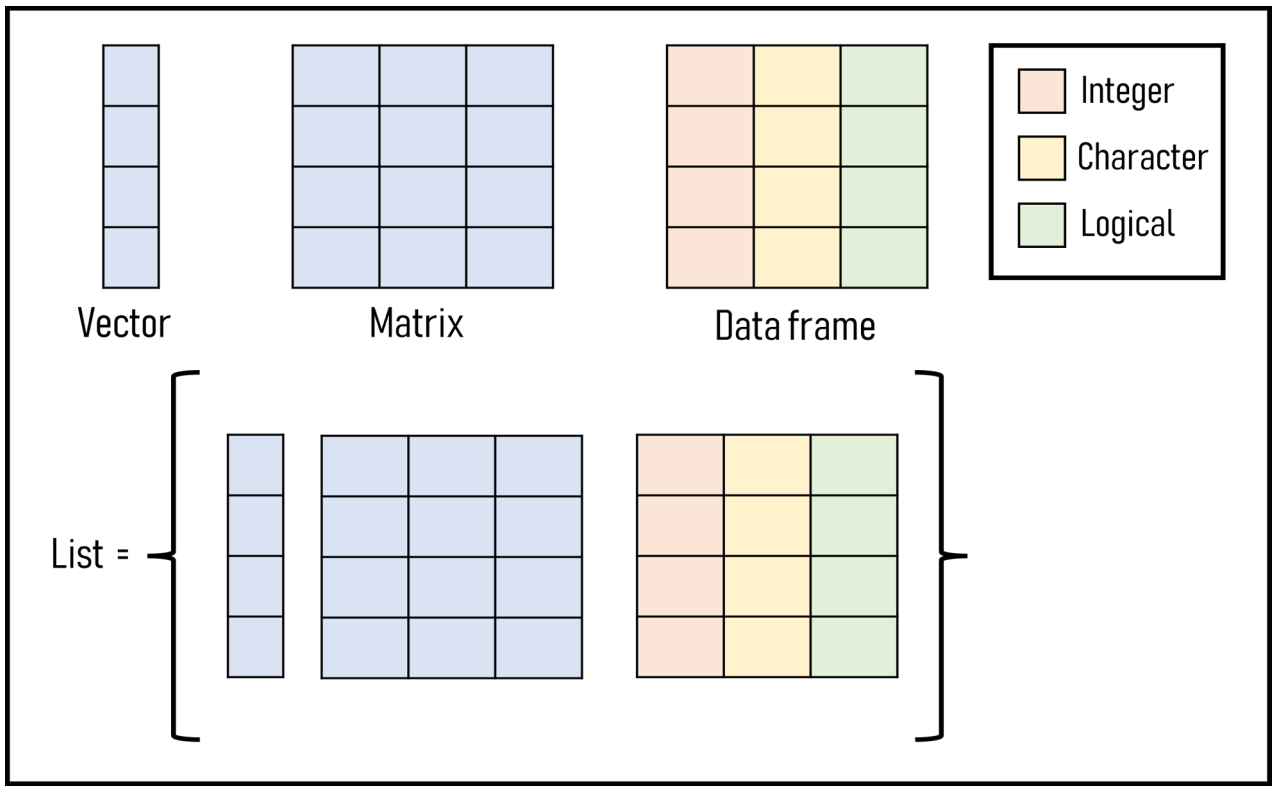


**Figure 1:** Estructuras principales de datos en R

# Tipos de vectores

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

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

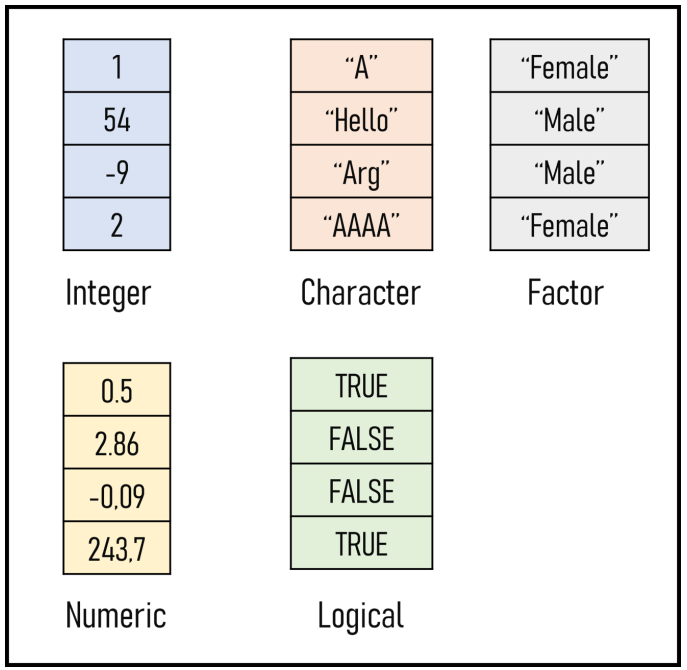


**Figure 2:** Clases de vectores

# 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**

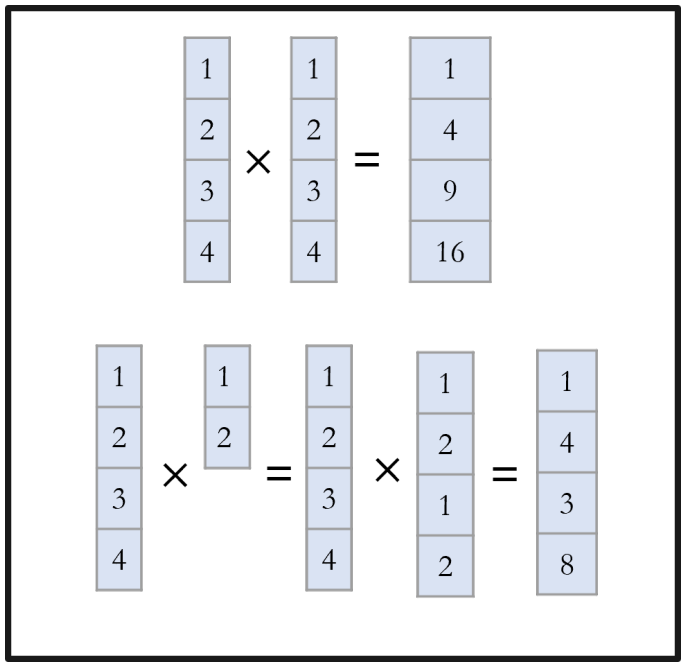


**Figure 3:** Ejemplos de ejecución por elementos

# 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
```

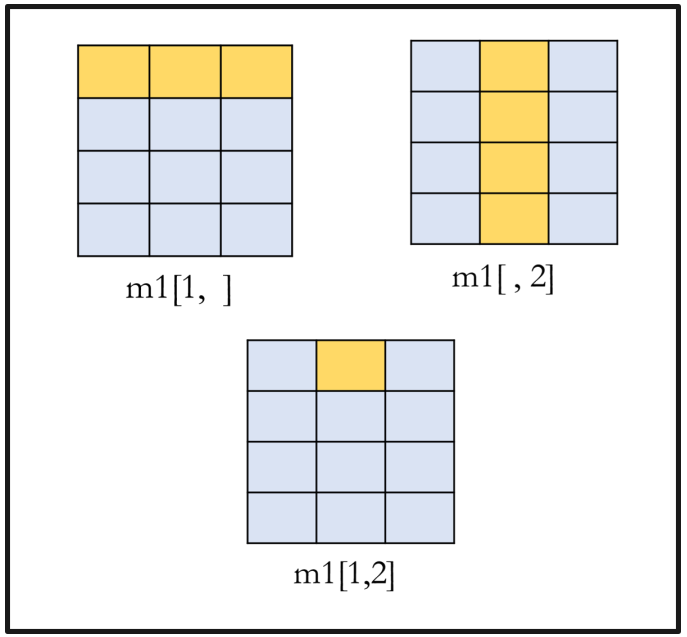


**Figure 4:** Uso de índices en matrices

**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
```

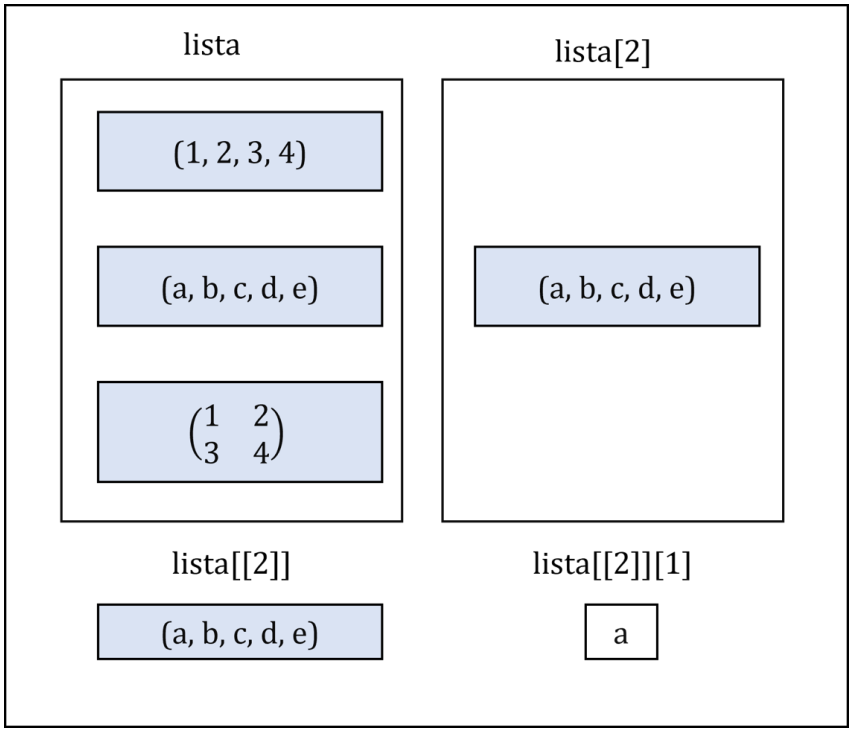


**Figure 6:** Uso de índices en listas

# 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. Crear una matriz partir de tres vectores (una cadena y dos numéricos). Convertir la matriz en data frame y seleccionar el elemento (2, 3)
2. Crear una función que, si la entrada es un vector numérico, la salida sea la media, la mediana, el percentil 25 y 75. (Para la salida use una lista).

# Herramientas para la manipulación de datos

# Paquetes en R

Los paquetes en R son colecciones de funciones, datos y documentación cuyo objetivo es extender las capacidades básicas de R. **CRAN** (The Comprehensive R Archive Network) es una red de servidores que almacenan versiones de R, así como librerías en R que cumplen las políticas del repositorio (CRAN, 2022).

Para instalar paquetes del repositorio **CRAN**:

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

Después de instalar el paquete, se debe cargar la librería:

```
library(dplyr)
```

Para encontrar la documentación del paquete:

```
help(dplyr)
```

# Tidyverse

**Tidyverse** es un conjunto de librerías en R diseñadas para el análisis de datos (importar, transforma, visualizar y modelar con datos) (Wickham, 2019).

Nos concentraremos en las siguientes librerías:

- dplyr
- ggplot2
- forcats*


**Figure 8:** Librerías en Tidyverse

# Importar datos

El primer paso es definir el directorio de trabajo:

```
setwd("path")
```

Nos concentraremos en funciones para importar los siguientes formatos de datos

| Formato | Formato específico | Función | Paquete |
|---|---|---|---|
| Texto o tabulares | CSV | read_csv() | readr |
| | Otros formatos de texto | read_delim() | readr |
| Formatos de otros programas | Excel | read_excel() | readxl |
| | SPSS | read_sav() | haven |
| | STATA | read_dta() | haven |
| | SAS | read_sas() | haven |
| Formatos propios de R | .rda | load() | base |
| | .rds | readRDS() | base |

# Pipe (%>%)

La tubería de comando o *pipeline* (%>%) es una herramienta utilizada para el encadenamiento de funciones. El operador nos permite escribir una secuencia de operaciones

Una secuencia en su **forma estándar** sigue la forma

```
dataset_2 <- dplyr::filter(dataset, attend > 15 & attend != 20)
```

En **forma encadenada**:

```
dataset_2 <- dataset %>% dplyr::filter(attend > 15 & attend != 20)
```

El siguiente atajo es útil: 

# Dplyr

El paquete **dplyr** proporciona una sintaxis para la manipulación de datos. (El operador %>% pertenece a la sintaxis de dplyr). Nos concentraremos en las siguientes funciones:

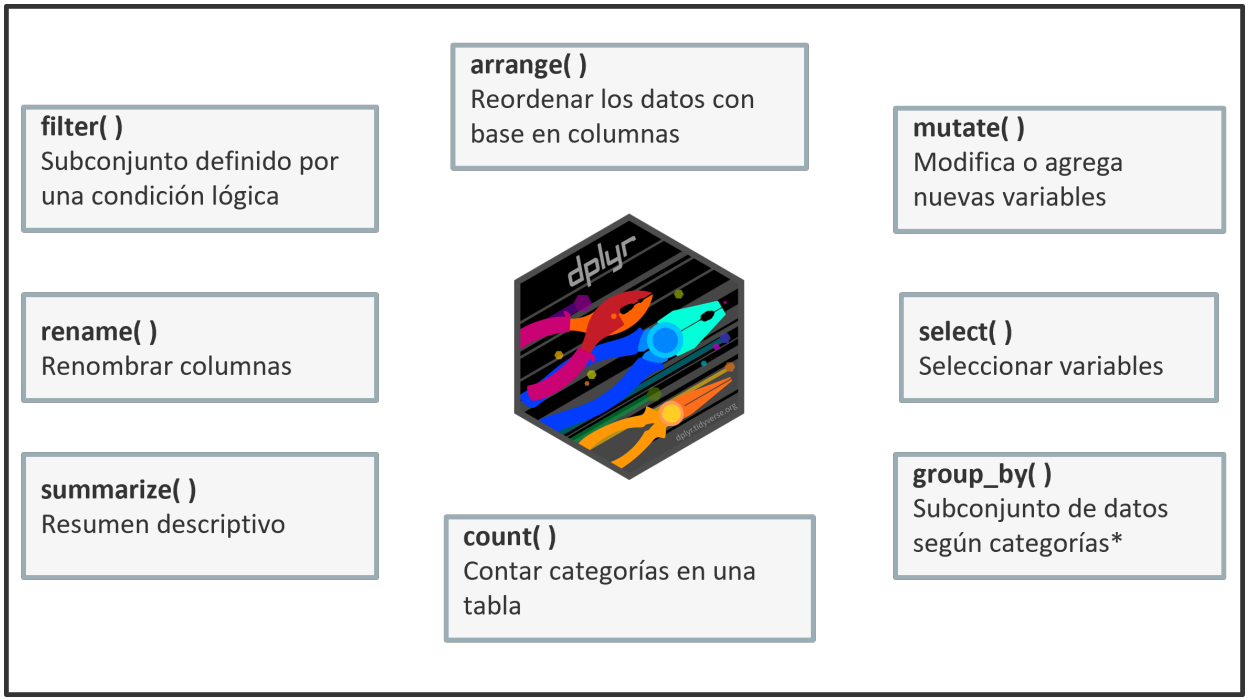


**Figure 9:** Algunas funciones en el paquete dplyr

# Resumen por grupo

Usando las funciones `summarize()` y `group_by()`, obtenemos un resumen descriptivo de la base de datos diferenciado según una o más variables de control. Por ejemplo:

```
# Resumen general
table_1 <- new_dataset %>% filter(Int_attend == "Group 4")
%>%   summarize(MeanAttend = mean(attend),SdAttend = sd(attend))
```

```
# Resumen diferenciado
table_2 <- new_dataset %>% group_by(Int_attend) %>%
  summarize(MeanAttend = mean(attend), SdAttend = sd(attend))
```

La **Figura 10** muestra el funcionamiento de `summarize()` y `group_by()`.

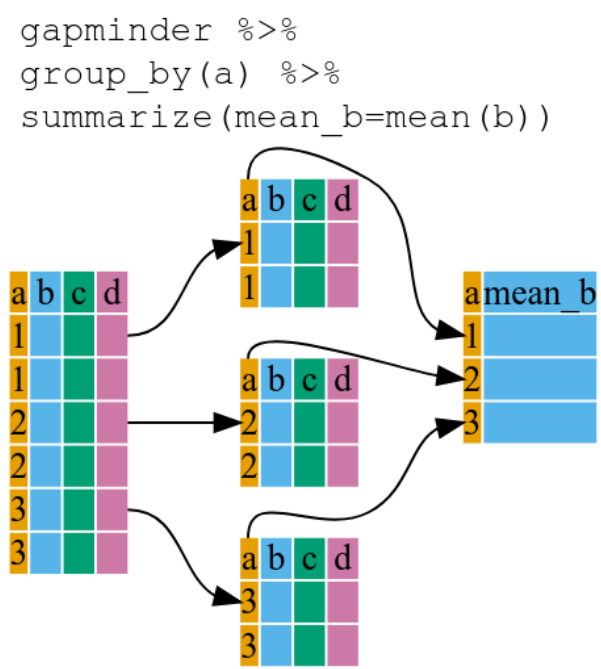


**Figure 10:** Caption for the picture.

# ggplot2

El paquete **ggplot2** proporciona un sistema coherente para visualizar datos y crear gráficos. La versatilidad de **ggplot2** radica en el uso de la Gramática de Gráficos (*Grammar of Graphics*).

```
ggplot(dataset, aes()) + geometría + faceta + opciones
```

donde:

1. *dataset* es un data frame
2. Las características del mapa **aes()** describe los ejes $(x, y)$, el color exterior (**color** o **colour**), el color interior (**fill**), la forma de los puntos (**shape**), el tipo de línea (**linetype**) y el tamaño (**size**)
3. Los objetos geométricos (**geometría**) determinan el tipo de gráfico:
   - Puntos (*geom_point*)
   - Líneas (*geom_lines*)
   - Histogramas (*geom_histogram*)
   - Boxplot (*geom_boxplot*)
4. La **faceta** permite dividir un gráfico en múltiples gráficos de acuerdo con grupos

# Ejercicios

1. Crear un gráfico que el ingreso promedio de acuerdo con la edad en años para las personas ocupados. Presente la gráfica diferenciada por ciudades.
2. Filtra la base de datos de personas ocupadas para el grupo de mujeres con ingresos de \$2.000.000 – \$5.000.000. Muestre un gráfico sobre el nivel de educación en esta población.

# Recursos alternativos

# Recursos alternativos

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

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

- 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


- Wickham, H. (2016) GGplot2. Elegant Graphics for Data Analysis. Springer
- 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.