Primeros Pasos en R

Clase 2: Introducción a R

Ana Alvarado, María Constanza Prado y Riva Quiroga

DATA SCIENCE UC
FACULTAD DE MATEMÁTICAS

FACULTAD DE MATEMÁTICAS
PONTIFICIA UNIVERSIDAD
CATÓLICA DE CHILE

Contenido del curso

Módulo 1: Introducción a R.

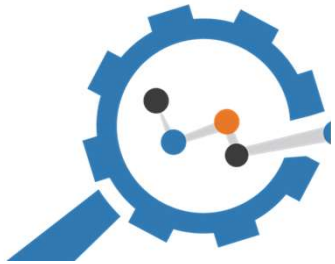
- Descripción software R.
- Diferencia entre R y R Studio.
- Interfaz (consola, editor, menú).
- Lenguaje de programación en R.
- Instalación de paquetes.

Módulo 2: Operaciones básicas en R.

- Importar bases de datos.
- Crear, poblar y grabar bases de datos.
- Concatenar / separar archivos (registros y variables) de Estructuras Estadísticas
- Transformar/re codificar variables.
- Creación de objetos.
- Creación de funciones.

Módulo 3: Análisis exploratorio y descriptivo en R.

- Calcular estadísticas de resumen.
- Generar tablas.
- Generar gráficos.



La clase de hoy:

Parte I

- Objetos en R.
  - Asignaciones
  - Tipos de objetos
  - Cómo nombrar a los objetos.
- Tipos de Objetos
- Paquetes en R
- ¿Errores? Como corregirlos y dónde buscar ayuda.

Parte II

- Operadores lógicos en R
- Manipulación de Objetos
- Importación de archivos
- Selección de objetos
- Comparación de las diferentes formas de programar en R

Consultas y comentarios clase anterior

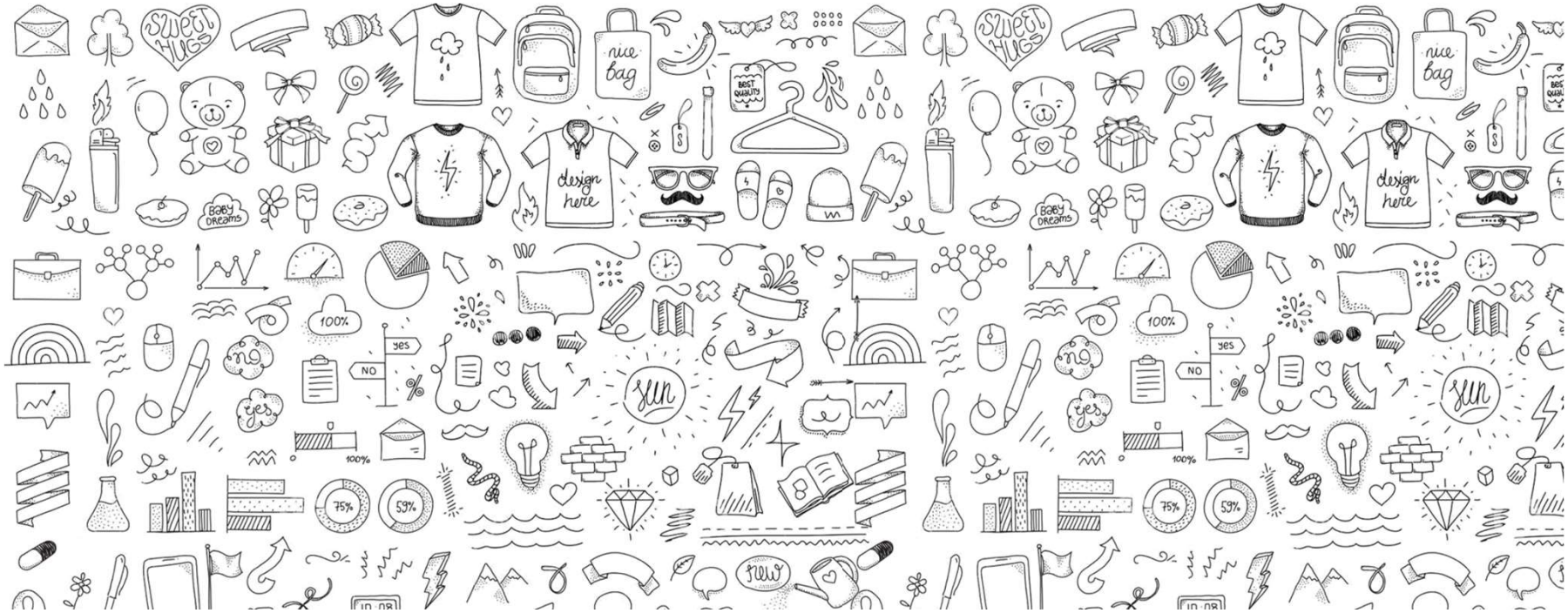
Consultas y comentarios clase anterior

- ¿Qué programa utilizar?, por que instalamos dos software R y RStudio, pero ¿uno es complemento del otro?, ¿por que instalamos dos programas si los dos tienen la misma función?
- Datos gratuitos para practicar como fuente de datos para R, ideal que sean masivos o grandes
- Repasar el tema del directorio.
- Es posible que nos compartan las presentaciones antes de tener la clase.

Objetos en R

Objetos

En R todas las funciones y herramientas son realizadas sobre objetos.



Objetos

Un objeto es una colección de información indexada, bajo un nombre previamente definido diferenciando mayúsculas de minúsculas.

Para asignar o crear un objeto, utilizaremos el operador de asignación: **<-**

```
nombre <- valor
```

También podemos usar ->

Alt + -

```
valor -> nombre
```

Cada asignación puede ir en una fila diferente, o bien seguido de ";".

El comando **ls()** muestra todos los objetos creados.

Objetos en R

Por ejemplo:

```
num <- 521
521 -> num
```

Para crear objetos no se recomienda usar = (aunque funciona también) porque puede causar confusiones.

Ahora podemos aplicar funciones a estos objetos:

```
num <- 521
sqrt(num)
```

y si deseamos, podemos guardar ese resultado en otro objeto:

```
res <- sqrt(num)
```

Objetos en R

Si el objeto ya existe R lo va a sobre escribir.

Si tenemos:

```
a <- 5
b <- a
a <- 4
```

¿Cuál es el valor de a?

Objetos en R

Si el objeto ya existe R lo va a sobre escribir.

Si tenemos:

```
a <- 5
b <- a
a <- 4
```

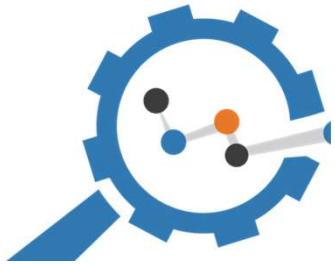
¿Cuál es el valor de a?

¿Cuál es el valor de b?

Objetos en R

R recordará todos los objetos creados, hasta el momento en que uno cierra la consola, o bien, hasta que se le pida eliminar información.

El comando **`rm(OBJETO)`** elimina cierto objeto, mientras que **`rm(list=ls())`** elimina toda la memoria.



Objetos más usados en Data Science

- Escalar
- Vector
- Data frame
- Lista

Objetos más usados en Data Science

- Escalar
- Vector
- Data frame
- Lista

Se considera un escalar al objeto que posee un valor fijo como una constante.

```
> x <- 5
> x
[1] 5
```

Objetos más usados en Data Science

- Escalar
- Vector
- Data frame
- Lista

Se considera vector a un conjunto de escalares, que posean la misma naturaleza.

En términos matemáticos, es una matriz con una dimensión.

En términos estadísticos, lo denominaremos variable.

Para crear un vector se utiliza el comando
c(OBJETO 1, OBJETO2, …).

```
> num <- c(3,4,5,6,7)
> num
[1] 3 4 5 6 7
```

Objetos más usados en Data Science

- Escalar
- Vector
- Data frame
- Lista

Un vector puede tener números o letras (caracteres). En este último caso, se debe considerar comillas, de lo contrario R pensará que es otro objeto.

En R hay 6 tipos de vectores, en Data Science es común usar estos 4 :

- numeric (números reales)
- integer (números enteros)
- character (letras)
- logical (verdadero/falso)

```
> dias <- c("L", "M", "W","J","V")
> dias
[1] "L" "M" "W" "J" "V"
```

Ahora...

¿Qué entrega la siguiente secuencia de comandos?

```
num <- c(3,4,5,6,7)
dias <- c("L", "M", "W","J","V")
numdias <-c(num,dias)
numdias
```

¿Qué tipo de vector es `numdias`?

Objetos más usados en Data Science

- Escalar
- Vector
- Data frame
- Lista

Un data frame es un conjunto de vectores de diversas naturalezas.
Cada vector tiene el mismo número de elementos.

Para crear un data frame usamos el comando:

`data.frame()`

```
df <- data.frame(num, dias)
df
  num dias
1   3    L
2   4    M
3   5    W
4   6    J
5   7    V
```

Objetos más usados en Data Science

- Escalar
- Vector
- Data frame
- Lista

Una lista es un conjunto es un conjunto de objetos de cualquier naturaleza.

Para crear una lista usamos el comando:

```
list()
```

```
lista <-list(num, dias, df)
lista
```

Recomendaciones

Al nombrar los objetos

- Utilizar un nombre que tenga alguna relación con los datos que contiene el objeto
- Evitar caracteres especiales, como ñ, tildes o espacios
- Para separar palabras se puede utilizar un guión bajo (**`proyeccion_enero`**) o mayúscula inicial (**`ProyeccionEnero`**). Lo importante es ser consistente en la opción elegida.

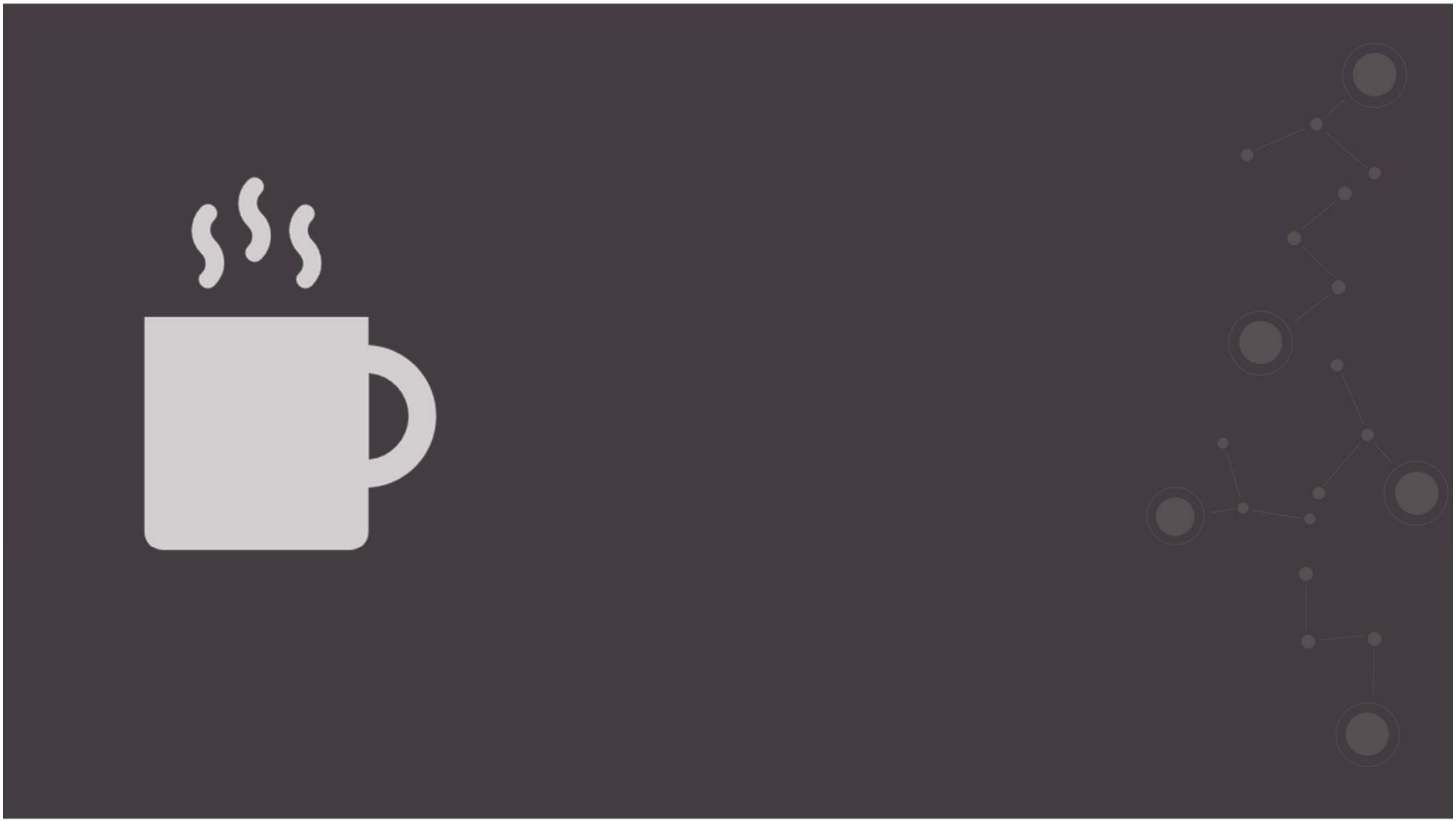
Recuerda que R es sensible a mayúsculas y minúsculas.

Nota:
Existen otros tipos de objetos como:
- Matrices
- Arrays

Ahora...

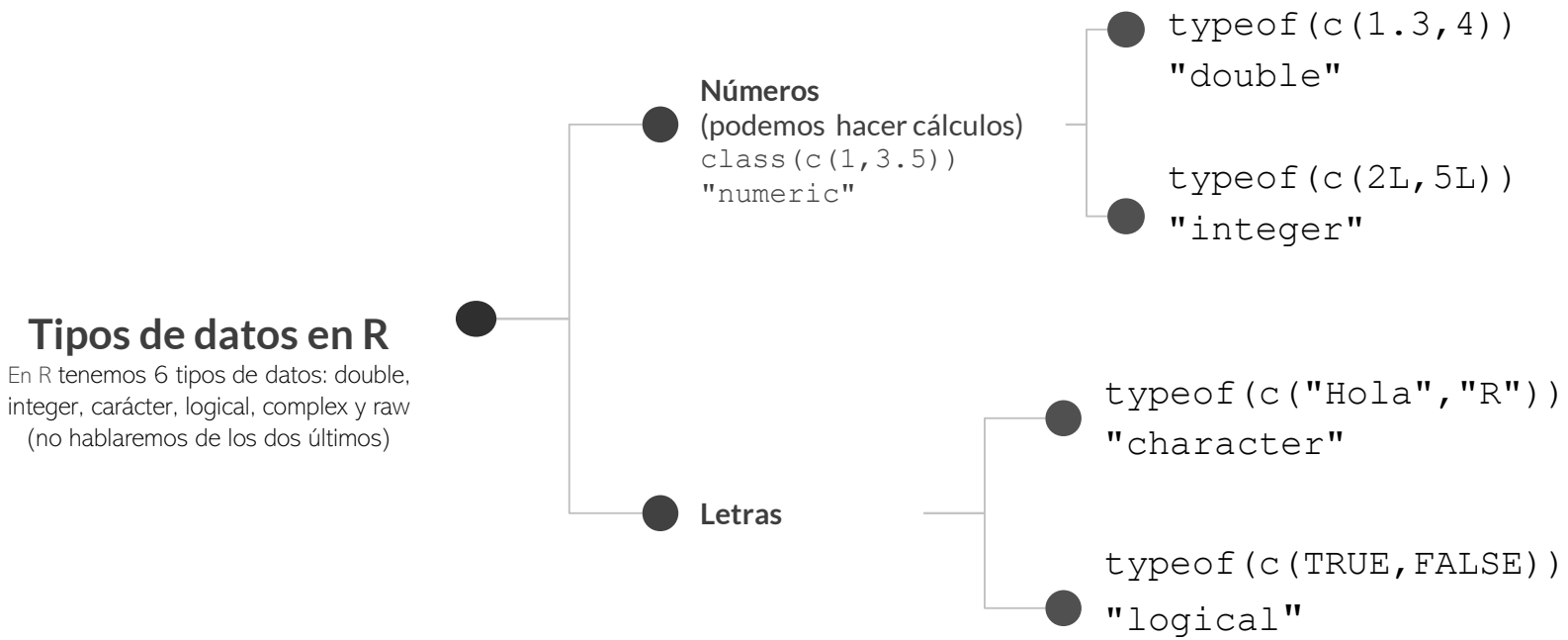
¿Qué pasa al ejecutar el siguiente código?

```
polera <- c(254,203,182,50)

mean(Polera)

sum(poleras)
```



Tipos de Datos en R

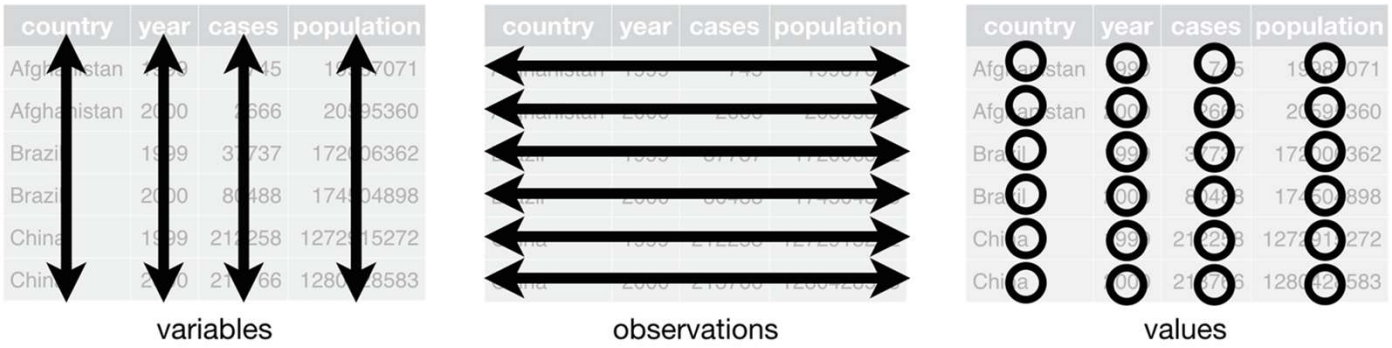
Tipos de datos en R

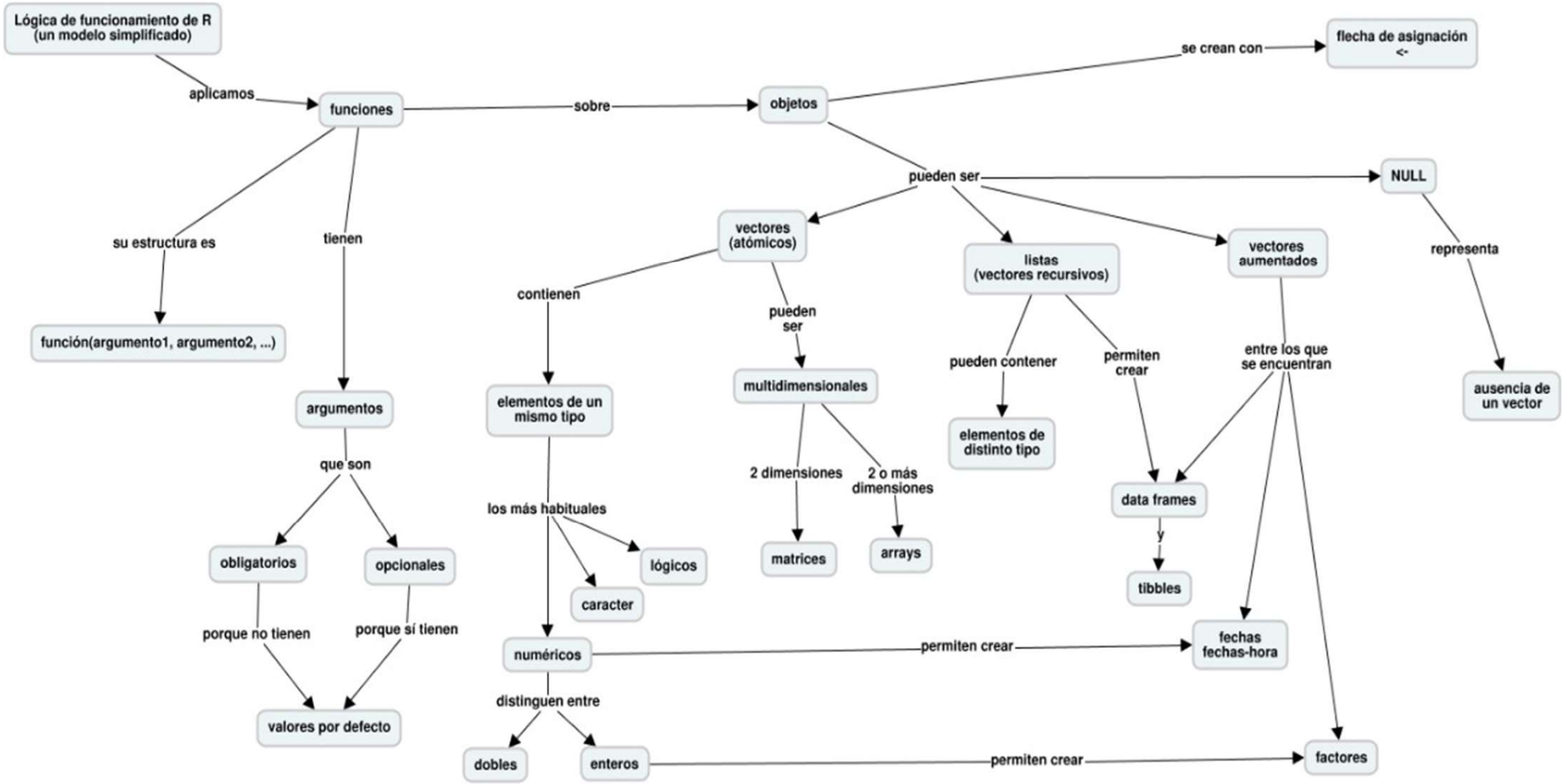


También podemos encontrarnos con:

- factores (tipos de datos para referirnos a relaciones cualitativas: bueno o malo. (Pueden ser números o letras)
- fechas.

Nota: Usualmente vamos a trabajar con datos ordenados:





Esquema por: Riva Quiroga

Paquetes en R

Paquetes

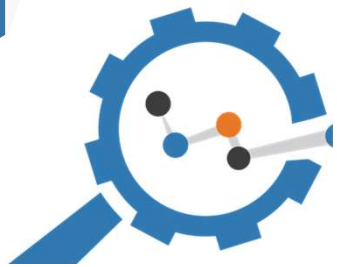
Recuerde…

R utiliza librerías como conjunto de funciones, que denominaremos paquetes o *packages*.

Son colecciones de funciones, datos y documentación.

Se instalan solo una vez en nuestro equipo (pero se deben actualizar cada cierto tiempo).

Son algunos de los paquetes más conocidos por su utilidad.



Paquetes

Cuando iniciamos R se carga solo un conjunto de funciones datos y documentación que se conoce como R Base.

Para extender las posibilidades de R Base tenemos que instalar otros paquetes.

La función **`search()`** permite ver los *paquetes* actuales en funcionamiento.

Para revisar si existen actualizaciones, puedes ir al menú

Tools > Check for Package Updates.

Aparecerá la lista de paquetes que tienen actualizaciones disponibles y te dará la opción de instalarlas.

Instalar paquetes

CRAN

Comprehensive R Archive Network

Es el repositorio oficial de paquetes de R. Para estar acá los paquetes deben ser aprobados.

```
install.packages("nombrepaquete")
```

GitHub

Es una plataforma para desarrollo de software.

Muchos paquetes solo se comparten por acá.

En Github también podemos acceder a la versión en desarrollo de paquetes que están en CRAN.

```
install.packages("remotes")
remotes::install_github("usuario
/paquete")
```

Usar funciones y datos de paquetes

Para usar las funciones y datos contenidos en un paquete tenemos que "activarlo" en nuestra sesión de R:

```
library(nombrepaquete)
```

Luego de esto, aparecerán disponibles las funciones de ese paquete.

Hay tantos paquetes, que a veces puede existir conflicto con los nombres. Podemos especificar de qué paquete viene una función con:

```
nombrepaquete::nombrefuncion
```

Ahora..

Instalemos el paquete ggplot2

```
install.packages("ggplot2")
```

Luego activemos el paquete ggplot2

```
library(ggplot2)
```

Notar que no requiere " "

Ahora estamos listos para usar todas las funciones gráficas del paquete ggplot2

En las próximas clases aprenderemos a realizar gráficos en ggplot

Ejemplo:

```
ggplot(data=iris, aes(Sepal.Length, Petal.Length, color=Species)) +
geom_point()
```

Errores frecuentes

¿Errores?

La mayoría de los errores que cometemos son por problemas de tipeo:

- escribimos mal el nombre de una función u objeto
- nos falta cerrar un paréntesis
- nos falta una coma.

En caso de que falte un paréntesis o una coma, el editor de RStudio nos lo advertirá.

A veces se generan problemas porque olvidamos correr una línea de código o porque sobreescribimos un objeto. En esos casos, lo mejor es reiniciar R y volver a ejecutar el código desde el principio.

Recuerda que equivocarse es normal. Puedes googlear un error si no sabes como resolverlo (o no lo entiendes). Es muy probable que a alguien más ya le haya sucedido.

Encuentra el error en el código:

1. `install.packages(ggplot2)`

2. `librari(ggplot)`

3. `c(2.4, 5, 7, 8, 2,)`

4. `nombres <- c(Constanza,  "Riva", "Ana")`

5. `edad <- c(34,67,20,45,10)`
   `sum(Edad)`

   `maen(edad)`

Encuentra el error en el código:

1. **install.packages(ggplot2)**
   R: install.packages("ggplot2")

2. **librari(ggplot)**
   R: library(ggplot2)

3. **c(2.4, 5, 7, 8, 2,)**
   R: c(2.4, 5, 7, 8, 2)

4. **nombres <- c(Constanza, "Riva", "Ana")**
   R: nombres <- c("Constanza", "Riva", "Ana")

5. **edad <- c(34,67,20,45,10)**
   sum(Edad)
   R: sum(edad)

   maen(edad)
   R: mean(edad)

Generación de Objetos

Generación de objetos

`n1:n2`

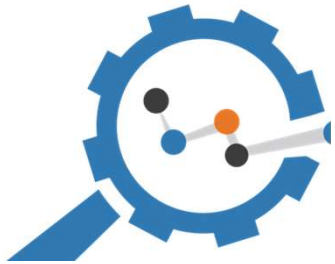
Crea datos enteros de n1 hasta n2.

`seq(from, to, by, length.out)`

Crea secuencias desde un punto inicial from y un punto final to, con argumentos para agregar la distancia entre los dos valores (by), o la cantidad total de números entre ellos (length.out).

`rep(x,n)`

Replica el objeto x un total de n veces.



Operadores lógicos

Operadores lógicos

En R tenemos también la opción de usar operadores lógicos:

<	**menor que**
>	**mayor que**
<=	**menor o igual que**
>=	**mayor o igual que**
==	**igualdad**
!=	**diferencia**
x & y	**intersección**
x | y	**unión**

Por ejemplo:

Podemos comparar dos vectores:

```
num <- c(7, 3, 23, 8, 2.5)
num < 3
```

El resultado es un vector de valores lógicos:

```
FALSE FALSE FALSE FALSE  TRUE
```

¿Diferentes formas de programar en R?

¿R base y el Tidyverse?

Los tres tipos de sintaxis más usados en R son:

1. Sintaxis de $ (R base)
2. Sintaxis de fórmula
3. Sintaxis del Tidyverse

En el archivo *cheatsheet_sintaxis_R.pdf* (traducido por Riva) pueden ver varios ejemplos y comparaciones.

La sintaxis de $ se enfoca más en el uso de subsetting con [] y la lógica de programación es de adentro hacia afuera.

La sintaxis de fórmula se basa en el uso de

~ (virgulilla)

La sintaxis del Tidyverse se ha popularizado en los últimos años porque permite leer y programar de forma más natural (izquierda a derecha) y tiene un ecosistema de paquetes que trabajan juntos y facilitan mucho el trabajo. Uno de sus elementos es "el pipe" %>%.

"Si bien usualmente se intenta enseñar R en el marco de un tipo de sintaxis, la mayoría de las personas programa usando una combinación de ellas. "

Algunos ejemplos adicionales:

Vamos a utilizar el set de datos **`mtautos`** disponible en el paquete datos.

Referencias y material complementario

R para Ciencia de Datos: https://es.r4ds.hadley.nz/ (libro en línea, en español)

RStudio cheatsheets: https://rstudio.com/resources/cheatsheets/ ("torpedos"; en la parte inferior de la página hay versiones en español disponibles)

Tidyverse https://www.tidyverse.org/ (documentación, ejemplos de los paquetes del Tidyverse).

Para practicar en casa:
RStudio Primers: https://rstudio.cloud/learn/primers (ejercicios interactivos; en inglés)

Comunidades:
En twitter: #rstats #rstatsES #datosdemiércoles #tidytuesday #RLadies y pueden seguir a @R4DS_es , @R4DScommunity