R

¿Qué es R?

R es un lenguaje y un ambiente para cómputo estadístico, es un proyecto GNU similar al lenguaje y ambiente S, desarrollado por los laboratorios Bell.

Huh ¿?

R es un lenguaje que nos ayuda a leer, transformar, graficar, modelar y entender datos.

¿Por qué R?

- R está enfocado al manejo modelado y graficación de información.
- Miles de paquetes, con los últimos desarrollos en estadística.
- Manejo de series de tiempo, bases de datos, datos geoespaciales.
- Conecta muy bien.

Bases del lenguaje.

Rstudio

Variables

Los objetos y funciones se almacenan con el operador `<-`

In [1]:

```
x <- 2
y <- 3
```

Variables

Podemos ver lo que tienen las variables al teclear el nombre:

In [2]:

```
x
y
```

2

3

Vectores

Un vector es un conjunto de cosas, las cosas contenidas tienen que ser del mismo tipo, e.g. todos números, todos caracteres.

In [7]:

```
v <- c(1, 2, 3, 42)
v1 <- c(v,'a')
v
v1
```

1 · 2 · 3 · 42

'1' · '2' · '3' · '42' · 'a'

Vectores

Se pueden generar de distintas maneras:

In [8]:

```
v <- 1:10
v <- seq(1,10)
v <- seq(1,20,by=0.2)
v
```

1 · 1.2 · 1.4 · 1.6 · 1.8 · 2 · 2.2 · 2.4 · 2.6 · 2.8 · 3 · 3.2 · 3.4 · 3.6 · 3.8 · 4 · 4.2 · 4.4 · 4.6 · 4.8 · 5 ·
5.2 · 5.4 · 5.6 · 5.8 · 6 · 6.2 · 6.4 · 6.6 · 6.8 · 7 · 7.2 · 7.4 · 7.6 · 7.8 · 8 · 8.2 · 8.4 · 8.6 · 8.8 · 9 · 9.2 ·
9.4 · 9.6 · 9.8 · 10 · 10.2 · 10.4 · 10.6 · 10.8 · 11 · 11.2 · 11.4 · 11.6 · 11.8 · 12 · 12.2 · 12.4 · 12.6 · 12.8 ·
13 · 13.2 · 13.4 · 13.6 · 13.8 · 14 · 14.2 · 14.4 · 14.6 · 14.8 · 15 · 15.2 · 15.4 · 15.6 · 15.8 · 16 · 16.2 · 16.4 ·
16.6 · 16.8 · 17 · 17.2 · 17.4 · 17.6 · 17.8 · 18 · 18.2 · 18.4 · 18.6 · 18.8 · 19 · 19.2 · 19.4 · 19.6 · 19.8 · 20

Vectores

También se pueden crear vectiores repitiendo algún otro vecrtor :).

In [9]:

```
v <- rep(1:20,5)
v
```

1 · 2 · 3 · 4 · 5 · 6 · 7 · 8 · 9 · 10 · 11 · 12 · 13 · 14 · 15 · 16 · 17 · 18 · 19 · 20 · 1 · 2 · 3 · 4 · 5 ·
6 · 7 · 8 · 9 · 10 · 11 · 12 · 13 · 14 · 15 · 16 · 17 · 18 · 19 · 20 · 1 · 2 · 3 · 4 · 5 · 6 · 7 · 8 · 9 · 10 ·
11 · 12 · 13 · 14 · 15 · 16 · 17 · 18 · 19 · 20 · 1 · 2 · 3 · 4 · 5 · 6 · 7 · 8 · 9 · 10 · 11 · 12 · 13 · 14 ·
15 · 16 · 17 · 18 · 19 · 20 · 1 · 2 · 3 · 4 · 5 · 6 · 7 · 8 · 9 · 10 · 11 · 12 · 13 · 14 · 15 · 16 · 17 · 18 ·
19 · 20

Vectores

Podemos acceder a determinados valores del vector por su posición. (Ojo, los índices en R empiezan en 1.)

In [10]:

```
v[3]
```

3

Vectores

Se pueden hacer operaciones sobre vectores:

In [11]:

```
(v*2+42)^3
```

85184 · 97336 · 110592 · 125000 · 140608 · 157464 · 175616 · 195112 · 216000 · 238328 · 262144 · 287496 ·
314432 · 343000 · 373248 · 405224 · 438976 · 474552 · 512000 · 551368 · 85184 · 97336 · 110592 · 125000 ·
140608 · 157464 · 175616 · 195112 · 216000 · 238328 · 262144 · 287496 · 314432 · 343000 · 373248 · 405224 ·
438976 · 474552 · 512000 · 551368 · 85184 · 97336 · 110592 · 125000 · 140608 · 157464 · 175616 · 195112 ·
216000 · 238328 · 262144 · 287496 · 314432 · 343000 · 373248 · 405224 · 438976 · 474552 · 512000 · 551368 ·
85184 · 97336 · 110592 · 125000 · 140608 · 157464 · 175616 · 195112 · 216000 · 238328 · 262144 · 287496 ·
314432 · 343000 · 373248 · 405224 · 438976 · 474552 · 512000 · 551368 · 85184 · 97336 · 110592 · 125000 ·
140608 · 157464 · 175616 · 195112 · 216000 · 238328 · 262144 · 287496 · 314432 · 343000 · 373248 · 405224 ·
438976 · 474552 · 512000 · 551368

A estas se les llama operaciones *vectorizadas*

Vectores

Podemos esta operación vectorizada nos dice los elementos positivos del vector

In [12]:

```
x <- c(1, -3, -2, 4, -10, 2, 3, 90)
x > 0
```

TRUE · FALSE · FALSE · TRUE · FALSE · TRUE · TRUE · TRUE

Vectores

Mejor aún, podemos obtener los vectores!

In [13]:

```
x[x > 0]
```

1 · 4 · 2 · 3 · 90

Factores

Cuando uno quiere analizar datos, regularmente necesita variables que funcionan como etiquetas. Estas etiquetas pueden ser contadas, pero no sumadas. A este tipo de variables se les conoce como variables *categóricas* y en `R` se llaman **factores**.

Factores

Es posible crearlas a partir de un `vector`

In [14]:

```
f <- c("M", "F", "M", "M", "F", "M")
f <- factor(f)
f
```

M · F · M · M · F · M

▶ **Levels**:

Y pueden tener una descripción larga (`label`)

In [15]:

```
f <- factor(c("M", "F", "M", "M", "F", "M", "F"),
            levels=c("M", "F"),
            labels=c("Hombre", "Mujer"))
f
```

Hombre · Mujer · Hombre · Hombre · Mujer · Hombre · Mujer

▶ **Levels**:

Factores

Hagámos una copia de `f`

In [16]:

```
g <- f
g
f
```

Hombre · Mujer · Hombre · Hombre · Mujer · Hombre · Mujer

▶ **Levels**:

Hombre · Mujer · Hombre · Hombre · Mujer · Hombre · Mujer

▶ **Levels**:

Factores

Los `levels` se pueden extraer con la siguiente función:

In [17]:

```
levels(g)
```

'Hombre' · 'Mujer'

Factores

¿Qué pasa si quiero agregar un nuevo elemento al final del factor?

In [21]:

```
g[length(g)+1] <- "NR"
g
```

```
Warning message in `[<-.factor`(`*tmp*`, length(g) + 1, value = "NR"):
“invalid factor level, NA generated”
```

Hombre · Mujer · Hombre · Hombre · Mujer · Hombre · Mujer · <NA> · <NA> · <NA> · <NA>

▶ **Levels**:

Mmmm, como el *nivel* `NR` no estaba definido, lo agrega como un valor inexistente (`NA`).

Factores

In [22]:

```
g <- factor(c("a", "b", "a", "a", "b", "b", "b"),
            levels=c("a", "b", "c"))
g
```

a · b · a · a · b · b · b

▶ **Levels**:

In [23]:

```
g[length(g)+1] <- "c"
g
```

a · b · a · a · b · b · b · c

▶ **Levels**:

Factores

Una tabla con los conteos por nivel se puede obtener fácilmente

In [24]:

```
table(g)
```

```
g
a b c
3 4 1
```

Factores

In [25]:

```
a <- factor(c("estudiante", "profesor", "estudiante",
              "profesor", "estudiante", "estudiante",
              "estudiante"))
a
```

estudiante · profesor · estudiante · profesor · estudiante · estudiante · estudiante

▶ **Levels**:

Es posible mezclar los dos factores en una tabla

In [26]:

```
t <- table(a,f)
t
```

```
            f
a            Hombre Mujer
  estudiante      3     2
  profesor        1     1
```

Nota como se le está asignando el género por renglón.

Factores

Y una tabla de *proporciones* nos da (obviamente) la proporción

In [27]:

```
prop.table(t, 2)
```

```
            f
a               Hombre     Mujer
  estudiante 0.7500000 0.6666667
  profesor   0.2500000 0.3333333
```

Data frames

Data frames

- Cuando se piensa en análisis de datos, usualmente se tiene en mente una estructura de observaciones y características de las mismas, en una especie de "tabla", la característica común es que son objetos rectangulares (de dos dimensiones).
- `R` proveé una abstración para los datos en formato rectangular llamado `data.frame`

Data Frames

In [28]:

```
df <- data.frame(
         var.1=c('A', 'B', 'C', 'A'),
         var.2=c('h', 'a', 'u', 'p'),
         var.3=c(1, 2, 3, 4.5))
df
```

A data.frame: 4 × 3

var.1 <fct>	var.2 <fct>	var.3 <dbl>
A	h	1.0
B	a	2.0
C	u	3.0
A	p	4.5

Data frames

La extracción ahora debe de hacerse en dos dimensiones (renglones, columnas)

In [30]:

```
df[3,1]
```

C

▶ **Levels**:

Extraer un renglón

In [31]:

```
df[3,]
```

A data.frame: 1 × 3

	var.1	var.2	var.3
	<fct>	<fct>	<dbl>
3	C	u	3

Data frames

Extraer una columna

In [32]:

```
df[,3]
```

1 · 2 · 3 · 4.5

Se puede utilizar el nombre de la columna para extraer la columna completa

In [33]:

```
df$var.3
df['var.3']
```

1 · 2 · 3 · 4.5

A data.frame: 4 × 1

var.3
<dbl>
1.0
2.0
3.0
4.5

Data frames

La extracción "mágica" se preserva también en los =data.frame=s

In [34]:

```
df[df$var.3 > 2,]
```

A data.frame: 2 × 3

	var.1	var.2	var.3
	<fct>	<fct>	<dbl>
3	C	u	3.0
4	A	p	4.5

In [35]:

```
df[df$var.1 == "A", "var.3"]
```

1 · 4.5

Data frames

Así como los vectores (y sus derivados) tenían longitud, los `data.frame` tiene **tamaño** y es bidimensional

In [36]:

```
dim(df)
```

4 · 3

In [37]:

```
nrow(df)
ncol(df)
```

4

3

In [38]:

```
names(df)
```

'var.1' · 'var.2' · 'var.3'