Universidad de Alcalá

Programa de Doctorado

Ambiente de trabajo y funciones

Manejo y visualización de datos en R

Paloma Ruiz Benito, Verónica Cruz Alonso,
Julen Astigarraga Urcelay

Enero - 2022

Contenido

Día 2

Parte I.- Introducción a R

- Entornos de programación, introducción a R y RStudio.
- Tipos de variables y datos, operaciones aritméticas y lógicas, creación de vectores, matrices, listas y tablas. Selección de datos.

Parte II.- Manejo de datos

- Flujo y funciones en la gestión de bases de datos.
- Recomendaciones para la estructura de las bases de datos y creación de código de programación.
- Introducción a la gestión de datos. Recomendaciones para su generación.
- Estructuras de programación: condicionales, bucles y funciones.

Parte III.- Visualización de datos

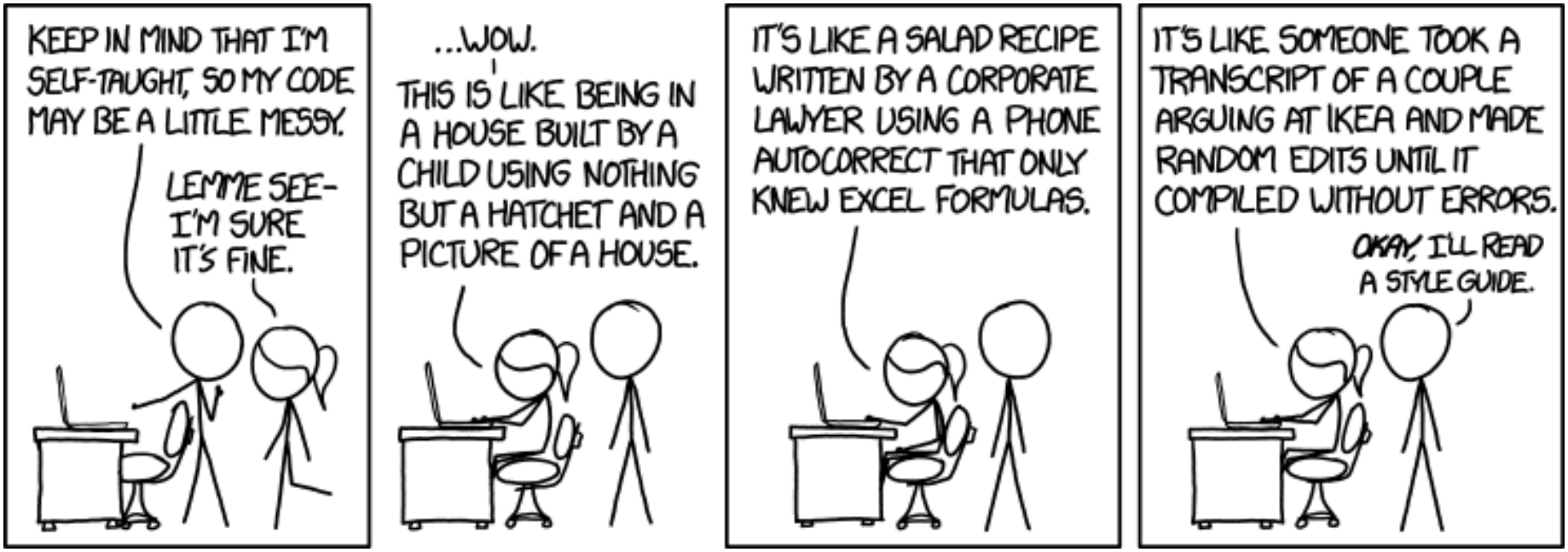
- Funciones básicas e introducción a ggplot para la visualización de datos.
- Generación de gráficos unidimensionales: histograma, dispersión, gráfico de cajas y bigotes, etc.
- Generación de gráficos bidimensionales: dispersión, boxplot, gráficos de barras, etc.
- Ejemplos y prácticas de visualización de gráficos en mapas.

Parte IV.- Trabajo reproducible

- Trabajo reproducible.
- Introducción a git y github.
- Rmarkdown.

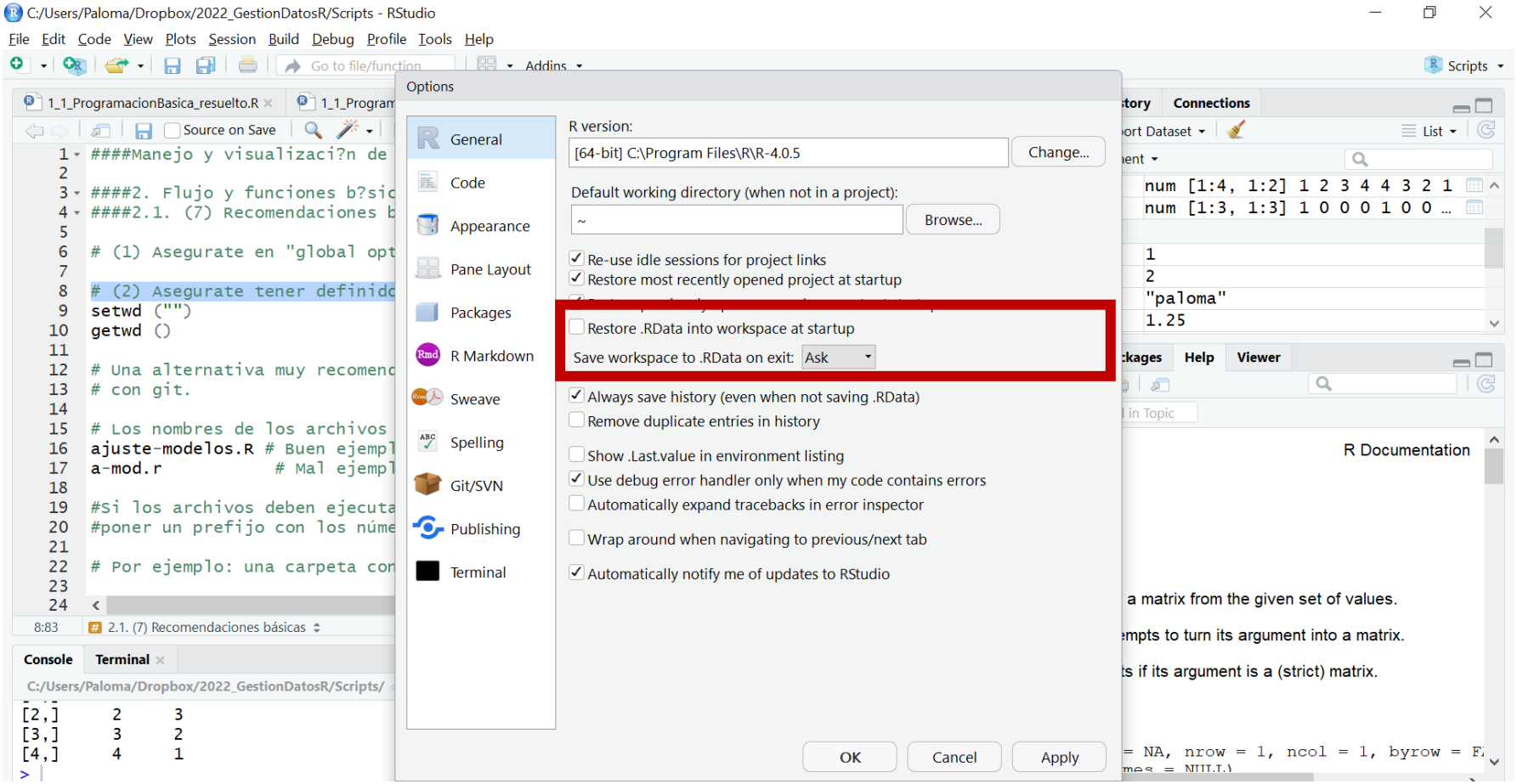
Flujo y funciones en la gestion de datos

```
# Recomendaciones
# Un ejemplo completo
# Funciones
```


KEEP IN MIND THAT I'M SELF-TAUGHT, SO MY CODE MAY BE A LITTLE MESSY.
LEMME SEE- I'M SURE IT'S FINE.
...WOW.
THIS IS LIKE BEING IN A HOUSE BUILT BY A CHILD USING NOTHING BUT A HATCHET AND A PICTURE OF A HOUSE.
IT'S LIKE A SALAD RECIPE WRITTEN BY A CORPORATE LAWYER USING A PHONE AUTOCORRECT THAT ONLY KNEW EXCEL FORMULAS.
IT'S LIKE SOMEONE TOOK A TRANSCRIPT OF A COUPLE ARGUING AT IKEA AND MADE RANDOM EDITS UNTIL IT COMPILED WITHOUT ERRORS.
OKAY, I'LL READ A STYLE GUIDE.

4 recomendaciones básicas

(1) En "global options" no tengas marcado "Restore Rdata..."



4 recomendaciones básicas

(2) Genera un proyecto organizado:

Carpetas para los diferentes grupos de archivos:
Por ejemplo: “1-archivos”, “2-scripts”, “3-resultados”

Nombres de los archivos deben tener un significado y en el caso de scripts terminar en “.R”.

```
ajuste-modelos.R
```

Si los scripts deben ejecutarse en secuencia está bien numerarlos.

```
# Por ejemplo: una carpeta de "2-scipts" con los archivos:
# 0-descarga.R, 1-exploracion.R, 2-Análisis.R
```

4 recomendaciones básicas

(3) Mantén tu código comprensible y fácil de leer:

(3.1) Documenta tu código (reproducible)

(3.2) Usa nombres memorables y correctos para tus variables

Object names

> "There are only two hard things in Computer Science: cache invalidation and naming things."
>
> — Phil Karlton

- No uses caracteres especiales o palabras reservadas.
- Usa minúsculas separadas por "_"
- se recomienda reservar los "." para objetos tipo S3
- Para variables usa nombres y para funciones verbos.
- Evita el uso de nombres comunes

(3.3) Espaciado

4 recomendaciones básicas

(3) Estructura tu código:

Ejemplo:

(4.1) Establecer el directorio de trabajo
(4.2) Llamar a librerias (o instalarlas si no estan instaladas)
y asegurate de posteriormente citarlos

```
library("dplyr")
citation("dplyr")
```

(4.3) Leer datos
(4.4) Gestión de archivos
(4.5) Analizar los datos
(4.6) Obtener los resultados que deseas y ¡guardarlos!
(archivo, mapa)

recomendaciones básicas

```
miproyecto
|- README          # información general
|- DESCRIPTION     # metadatos y dependencias del paquete
|- NAMESPACE       # generado automáticamente
|- makefile        # script maestro que ejecuta el análisis
|- data/           # datos depurados
|- data-raw/       # datos brutos
|- R/              # definición de funciones
|- man/            # documentación de las funciones
|- tests/          # tests de las funciones
|- analysis/       # ficheros Rmarkdown, figuras, etc.
```

Rodríguez-Sánchez et al. (2016) **Ciencia Reproducible: qué, por qué, cómo**. Ecosistemas 25: 83-92.

2 - Ambiente de trabajo y funciones

- **Directorio de trabajo**
  `setwd(), getwd()`

- **Crear vectores y data frames**
  `rep(), sample(), runif()`
  `data.frame()`

- **Gestion de datos**
  `colSums(), rowSums(), tapply(), dplyr()`

- **Escribir tablas**
  `write.table(), write.csv()`

- **Gráficos básicos**
  `plot()`

2 - Ambiente de trabajo y funciones

- **Si queréis tener un buen vocabulario en R:**

`http://adv-r.had.co.nz/Vocabulary.html`

Ejemplo competo

(1) establece el **directorio de trabajo**

```
> getwd()
> setwd("nuevo directorio")
```

(2) llama a **librerías** (e instálalas si no están instaladas)

```
> install.packages(c("plyr", "ggplot2"),
dep = T)
> library("plyr")
> citation(plyr)
```

¡¡NO OS OLVIDEIS DE CITAR R y paquetes!!

Ejemplo competo

(3) lee los **datos,** COMPRUEBA, prepara los archivos

```
> mydata<- data.frame(
    ID = c(10,20,30,40,50,60,70,80,90,100),
    items = c("libro", "libro", "bolígrafo", "manual","estuche",
    "manual", "libro", "manual", "estuche","estuche"),
    store = c(TRUE,FALSE,TRUE,FALSE,
    TRUE,FALSE,TRUE,FALSE,FALSE,FALSE),
    price = c(2.5,8,10,7,8,10,30,1,10,7))
> write.csv(mydata,”data_T11.csv”)
> mydata<-read.csv(”data_T11.csv”)
> mydata<-read.table(”data_T11.csv”, header=T, sep=“,”)
#mydata<-read.table(”clipboard”, header=T,sep=“\t”)
```

Ejemplo competo

(4) **Analiza** los datos: tapply y plot

```
summary(mydata)
```

```
       ID                items       store          price      
 Min.   : 10.0   bolígrafo:1   Mode :logical   Min.   : 1.00  
 1st Qu.: 32.5   estuche  :3   FALSE:6         1st Qu.: 7.00  
 Median : 55.0   libro    :3   TRUE :4         Median : 8.00  
 Mean   : 55.0   manual   :3                   Mean   : 9.35  
 3rd Qu.: 77.5                                 3rd Qu.:10.00  
 Max.   :100.0                                 Max.   :30.00  
```

Ejemplo competo

(5) Obtiene los resultados que deseas y sálvalo!
(archivo, mapa)

tapply() – agrupa los datos de un vector de acuerdo a una variable especificada y les aplica una función

```
tapply(mydata$price, mydataframe$items, FUN = mean)
tapply(mydata$price, mydataframe$items, FUN = sd)
tapply(mydata$price, mydataframe$items, FUN = range)
tapply(mydata$price, mydataframe$items, FUN =
quantile)
```

Ejemplo competo

(5) Obtiene los resultados que deseas y sálvalo!
(archivo, mapa)

aggregate() - Agrupa los datos de un arreglo de datos de acuerdo a una variable especificada y les aplica una función. Los resultados vienen dados en forma de arreglo de datos con las mismas variables de entrada, pero con tantas filas (casos) como niveles del factor utilizados para agrupar los datos.

```
> aggregate(mydata[,c("ID","price")],
list(mydataframe$items), FUN = mean)
    Group.1       ID     price
1 bolígrafo 30.00000 10.000000
2   estuche 80.00000  8.333333
3     libro 33.33333 13.500000
4    manual 60.00000  6.000000
```

Ejemplo competo

(5) Obtiene los resultados que deseas y sálvalo!
(archivo, mapa)

ddply() – resume diferentes funciones para diferentes factores

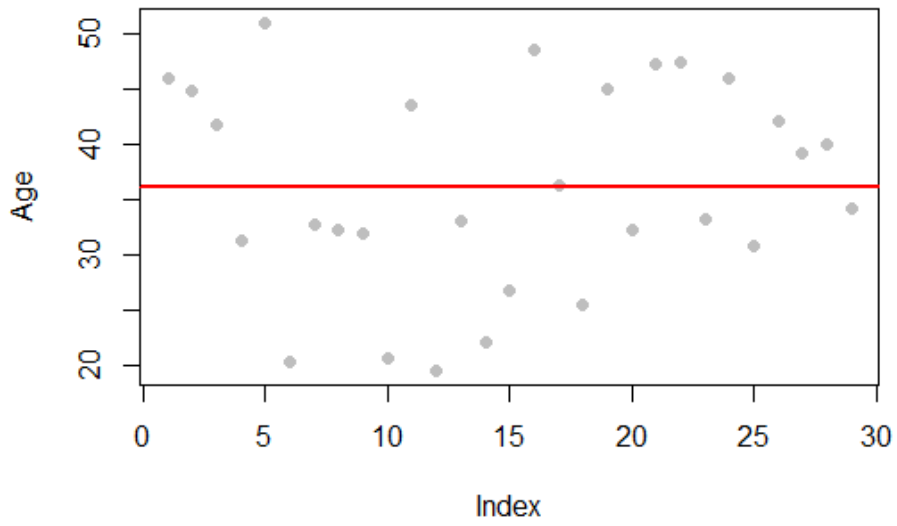
```
dfx <- data.frame(
  group = c(rep('A', 8), rep('B', 15), rep('C', 6)),
  sex = sample(c("M", "F"), size = 29, replace = TRUE),
  age = runif(n = 29, min = 18, max = 54),
  alt = dnorm(28, mean=160,sd=0.7)
)


ddply(dfx, .(group, sex), summarize,
      mean = mean(age),
      sd = sd(age))
```

Ejemplo competo

(5) Obtén los resultados que deseas y sálvalo!
(archivo, mapa)

plot() – Función de alto nivel a la que se le pueden añadir funciones de bajo nivel

```
plot(dfx$age, pch=16, col="gray", ylab="Age")
abline(h=mean(dfx$age), col="red", lwd=2)
```



Ejemplo competo

```
tiff("MiPrimeraFigura.tif",res=400, width = 120,
     height = 100, units = "mm")
plot(dfx$age, pch=16, col="gray", ylab="Age")
abline(h=mean(dfx$age), col="red", lwd=2)
dev.off()

plot(dfx$age~dfx$alt, pch=16,col="gray",
     ylab="Age", xlab="Height")

plot(dfx$age~dfx$sex, pch=16,col="gray",
     ylab="Age", xlab="Sex")

plot(dfx$group~dfx$sex, pch=16,col="gray",
     ylab="Group", xlab="Sex")
```

Función	**Utilidad**
points (x,y)	Agrega puntos sobre el gráfico (o argumento type en plot)
lines(x,y)	Como el anterior con líneas
mtext(text, side = 3, …)	Agrega el texto y en el margen especificado
abline(a, b) abline(h = y) abline(v = x) abline(lm.obj)	Dibuja una línea de pendiente “b” e intercepto “a”. Línea horizontal en el valor de y. Línea vertical en el valor de x. Dibuja el ajuste de la línea de regresion

2 - Ambiente de trabajo y funciones

#Funciones

Las funciones son objetos en si mismas. Puedes trabajar con las funciones de la misma manera que trabajas con otro tipo de objetos.

- ¿Qué compone una función?
- Alcance léxico
- Papel de las funciones
- Argumentos
- Resultados

Every operation is a function call

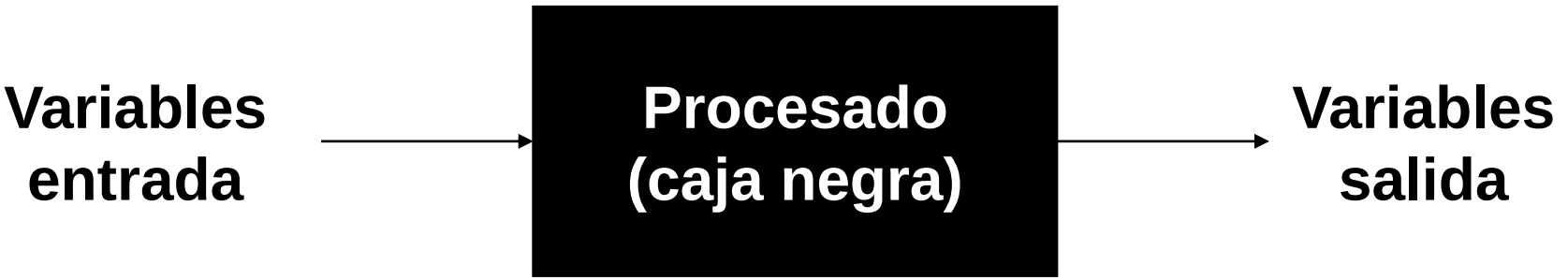
> “To understand computations in R, two slogans are helpful:
>
> - Everything that exists is an object.
> - Everything that happens is a function call."
>
> — John Chambers

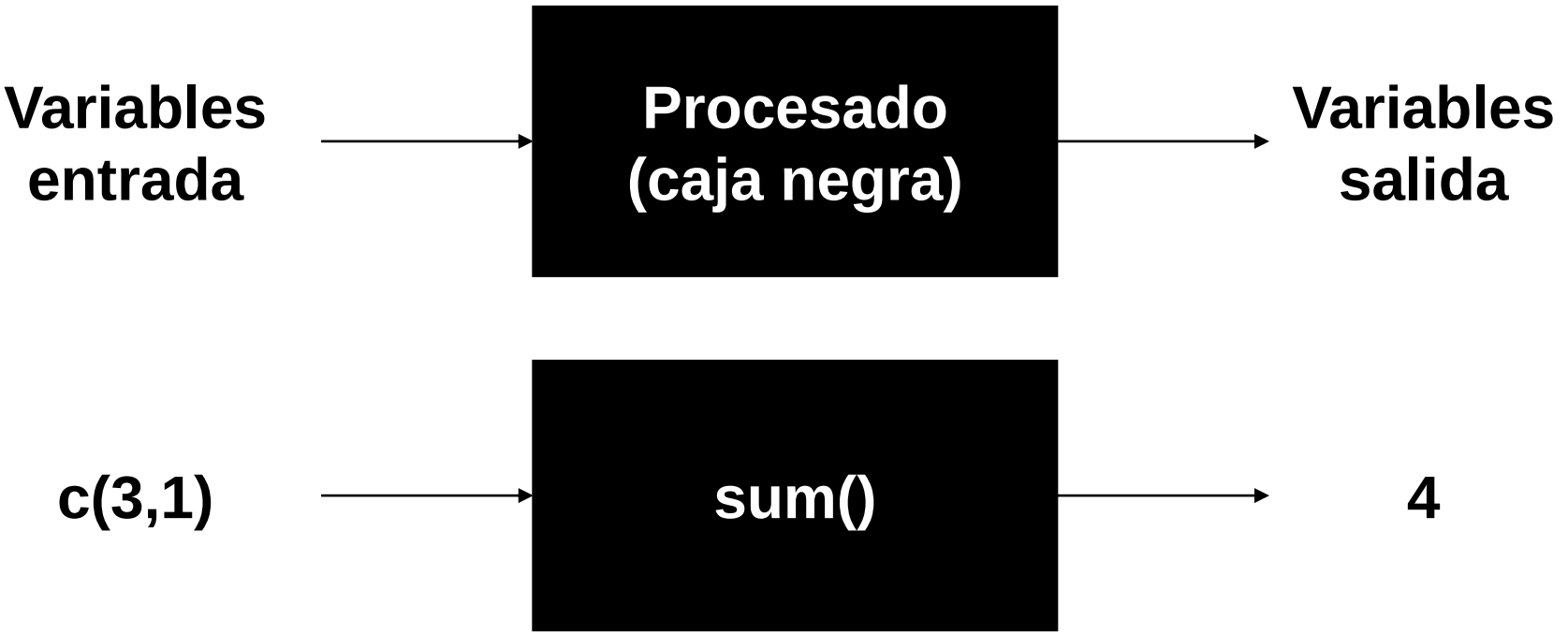
2 - Ambiente de trabajo y funciones

#Funciones

Las funciones son objetos en si mismas. Puedes trabajar con las funciones de la misma manera que trabajas con otro tipo de objetos.

- ¿Qué compone una función?
- Alcance léxico
- Papel de las funciones
- Argumentos
- Resultados


Variables
entrada
Procesado
(caja negra)
Variables
salida


Variables
entrada
Procesado
(caja negra)
Variables
salida
c(3,1)
sum()
4

```
> 1 + 2
3
> sumNum <- function(a, b) {a + b}
> sumNum(a = 1, b = 2)
3
```

```
> valores <- c(1,2)
> sum(valores)
> help(sum)
> ?sum
```

sum {base} R Documentati

Sum of Vector Elements

Description

`sum` returns the sum of all the values present in its arguments.

Usage

```
sum(..., na.rm = FALSE)
```

Arguments

`...`	numeric or complex or logical vectors.
`na.rm`	logical. Should missing values (including `NaN`) be removed?

Details

This is a generic function: methods can be defined for it directly or via the `Summary` group generic For this to work properly, the arguments `...` should be unnamed, and dispatch is on the first argument.

If `na.rm` is `FALSE` an `NA` or `NaN` value in any of the arguments will cause a value of `NA` or `NaN` to l returned, otherwise `NA` and `NaN` values are ignored.

Logical true values are regarded as one, false values as zero. For historical reasons, `NULL` is accepted and treated as if it were `integer(0)`.

Loss of accuracy can occur when summing values of different signs: this can even occur for sufficiently long integer inputs if the partial sums would cause integer overflow. Where possible extended-precision accumulators are used, typically well supported with C99 and newer, but possi platform-dependent.

Value

The sum. If all of the [illegible] arguments are of type integer or logical, then the sum is [illegible]


useR!