

Introducción a

¿Qué es ?

- Programa de código libre
- Dedicado a la estadística pero no solamente ...
- Entorno: programa y lenguaje de programación

¿Qué es ?

- Funciona en modo consola
- Se interactúa escribiendo comandos
- Funciones existentes
- Posibilidad de crear nuevas funciones

Preparación

- `c:\Mis Documentos\stats`
- `c:\Mis Documentos\stats\data`
- `c:\Mis Documentos\stats\docs`
- `c:\Mis Documentos\stats\scripts`

Empezamos

- ¡Abren R!
- Prompt: >
- q()
- Paréntesis, aún si estan vacios
- Consejo: ¡cerrar paréntesis inmediatamente!

Cerramos

- Save workspace image? [y/n/c]:
- Espacio de trabajo = variables que usamos en esta sesión dentro de R
- Cerrar la ventana = N

Uso de scripts

- En el espacio de trabajo, no hay los comandos que tipeamos
- Definir la carpeta de trabajo:
- `setwd("c:\\Mis Documentos\\stats")`
- Para guardar comandos, se usa un script
- Abrir editor de texto
(geany/cream/smultron/textwrangler...)
- Guardar el documento como:
- `c:\\Mis Documentos\\stats\\scripts\\script.R`

Comentarios

- R ignora los #
- ¡Comentarios, comentarios, comentarios!

Ayuda dentro de

- `help(q)`
- `help(quit)`
- `?q`
- `help.start(browser="firefox")`
- `help.search()`
- `??quit`

Ayuda fuera de

(¡Google es tu amigo!)

- <http://cran.r-project.org/>
- <http://crantastic.org/>
- www.rseek.org/
- STOP

R es una calculadora

- $3+5$
- $2*4$
- $3-8$
- $2.1-6*2.5$
- $3*2-5*(2-4)/6.02$
- 2^2

R es una calculadora científica

- `sqrt(4)`
- `abs(-4)`
- `log(4)`
- `sin(0)`
- `exp(1)`
- `round(3.1415,2)`
- Algunas constantes: `pi`

Creación de variables

- Crear una variable: `x <- 3.14`
- Ver el valor de un variable: `x`
- Usar una variable: `y <- 2*x`
- Mostrar todas las variables: `ls()`
- Borrar una variable: `rm(x)`

Noción de objetos

- Vector: colección de elementos de mismo tipo
- Crear un vector: `c()`
- `x <- c(1,2,3,-5,0)`
- `y <- c("a","b","c","AA")`

Otras maneras de crear vectores

- `rep(2.34, 5)`
- `2:7`
- `6:1`
- `seq(from=1, to=2, by=0.25)`
- `seq(from=1, to=2, length=7)`

Otro objetos

- Generalización en 2 dimensiones del vector: `matrix()`
- Generalización en n dimensiones del vector: `array()`
- Para variables cualitativas: `factor()`
- Lo más común con datos experimentales: `data.frame()`
- Contenedor general de datos: `list()`

Operaciones sobre los vectores

Ejercicio:

- Crean un vector x , un vector y de mismo tamaño que x , y un vector z de tamaño diferente. ¿Qué hacen los comandos siguientes?
- $x+1$
- $x*2$
- $\log(x)$
- $x>2$
- $x+y$
- $x*y$
- $x+z$
- $x*z$

Acceso a los elementos de un vector

- Indices y operador []
- `x<-c(1,3,-2,0,1.3,-6.43)`
- `x[1]`
- `x[3]`
- `x[-4]`
- `x[c(1,3)]`
- `x[c(-4,-5,-1)]`
- `x[c(TRUE,FALSE,TRUE,TRUE,FALSE,FALSE)]`
- `x[c(T,F)]`
- `x[x<1]`

Un lenguaje para la estadística

- `sum(x)`
- `length(x)`
- `mean(x)`
- `sd(x)`
- `median(x)`
- `var(x)`
- `summary(x)`

Un lenguaje para la estadística

Ejercicio:

- Crean un vector x de tamaño > 10
- Sin usar `mean()` ni `var()`, calculan la media y la varianza de x . Guardan los resultados en las variables *media.x* y *varianza.x*
- Comparan con los resultados de `mean(x)` y `var(x)`

Importación de datos en R

- Lo más flexible: `read.table()`
- Variante: `read.csv()`
- ¡Necesita camino completo!
- Para importar datos de Excel: convertir en *.csv desde Excel y leer el archivo *.csv desde R

Pinzones de Darwin

- `?read.table`
- `picos <- read.table("beaksize.csv", header=TRUE, sep="\t", dec=",")`
- Nombre del archivo
- El archivo tiene un membrete
- Separador de columnas
- Separador de decimales
- Datos del archivo en variable *picos*

Objeto *data.frame*

- Hoja de cálculo para almanecer datos
- Tabla de 2 dimensiones
- Con nombres de columnas, nombres de líneas ...
- Líneas = individuos
- Columnas = variables

Atributos de *picos*

- `dim(picos)`
- `nrow(picos)`
- `ncol(picos)`
- `rownames(picos)`
- `colnames(picos)`

Acceder a las variables de *picos*

- Con los índices: `picos[, 1]`
- Con los nombres de la variables: `picos[, "beaksize"]`
- Con el \$: `picos$beaksize`

Introducción a los gráficos

- `plot(1:10)`
- `plot(picos)`
- Algunos parámetros (ver `?par`):

`type` = tipo de línea

`main` = título del gráfico

`xlab` = título para eje x

`ylab` = título para eje y

`col` = color

...

Vector y factor

- Variable continua: vector
- Variable discreta: factor
- $\text{factor} = \text{vector} + \text{lista de valores posibles}$
- `as.factor()` transforma un vector en factor

picos: gráfico

- `plot(x=picos[, "survival"], y=picos[, "beaksize"],
xlab="Sobreviviencia", ylab="Largo del pico",
col="blue", main="Sobreviviencia de los pinzones")`

Ejercicio

- Chequen `picos$survival`. ¿De qué tipo es?
- ¿La representación gráfica es correcta?
- ¿Como cambiarla?

picos: gráfico

- `plot(x=picos[, "survival"], y=picos[, "beaksizel"],
xlab="Sobreviviencia", ylab="Largo del pico",
col="blue", main="Sobreviviencia de los pinzones")`

Ejercicio

- Chequen `picos$survival`. ¿De qué tipo es?
- ¿La representación gráfica es correcta?
- ¿Como cambiarla?

picos: gráfico

- `plot(x=picos[, "survival"], y=picos[, "beaksize"],
xlab="Sobreviviencia", ylab="Largo del pico",
col="blue", main="Sobreviviencia de los pinzones")`

Ejercicio

- Chequen `picos$survival`. ¿De qué tipo es?
- ¿La representación gráfica es correcta?
- ¿Como cambiarla?

picos: gráfico

- `plot(x=picos[, "survival"], y=picos[, "beaksize"],
xlab="Sobreviviencia", ylab="Largo del pico",
col="blue", main="Sobreviviencia de los pinzones")`

Ejercicio

- Chequen `picos$survival`. ¿De qué tipo es?
- ¿La representación gráfica es correcta?
- ¿Como cambiarla?

picos: gráfico

- `plot(x=as.factor(picos[, "survival"]), y=picos[, "beaksize"], xlab="Sobreviviencia", ylab="Largo del pico", col="blue", main="Sobreviviencia de los pinzones")`
- Para transformar en factor de manera permanente:
- `picos[, "survival"] <- as.factor(picos[, "survival"])`

picos: gráfico

- `plot(x=as.factor(picos[, "survival"]), y=picos[, "beaksize"], xlab="Sobreviviencia", ylab="Largo del pico", col="blue", main="Sobreviviencia de los pinzones")`
- Para transformar en factor de manera permanente:
- `picos[, "survival"] <- as.factor(picos[, "survival"])`

picos: gráfico

- `plot(x=as.factor(picos[, "survival"]), y=picos[, "beaksize"], xlab="Sobreviviencia", ylab="Largo del pico", col="blue", main="Sobreviviencia de los pinzones")`
- Para transformar en factor de manera permanente:
- `picos[, "survival"] <- as.factor(picos[, "survival"])`

picos: gráfico

- `plot(x=as.factor(picos[, "survival"]), y=picos[, "beaksize"], xlab="Sobreviviencia", ylab="Largo del pico", col="blue", main="Sobreviviencia de los pinzones")`
- Para transformar en factor de manera permanente:
- `picos[, "survival"] <- as.factor(picos[, "survival"])`

Más sobre los gráficos

- `plot()` es una función genérica
- Su comportamiento cambia en función de los datos
- Vean `?plot` y `?plot.default`

Histograma

- `hist(picos[, "beaksize"])`

- Parámetros:

`freq`: ¿Números o porcentajes?

`breaks`: Número de clases

`plain, labels`: ¿Etiquetar las barras?

- `hist(picos[, "beaksize"], breaks=20, plain, labels=TRUE, col="lightgrey", xlab="Largo del pico", ylab="Numero de observaciones", main="Pinzones de Darwin : largo del pico")`

Histograma

- `hist(picos[, "beaksize"])`
- Parámetros:
 - `freq`: ¿Números o porcentajes?
 - `breaks`: Número de clases
 - `plain, labels`: ¿Etiquetar las barras?
- `hist(picos[, "beaksize"], breaks=20, plain, labels=TRUE, col="lightgrey", xlab="Largo del pico", ylab="Numero de observaciones", main="Pinzones de Darwin : largo del pico")`

Histograma

- `hist(picos[, "beaksize"])`
- Parámetros:
 - `freq`: ¿Números o porcentajes?
 - `breaks`: Número de clases
 - `plain, labels`: ¿Etiquetar las barras?
- `hist(picos[, "beaksize"], breaks=20, plain, labels=TRUE, col="lightgrey", xlab="Largo del pico", ylab="Numero de observaciones", main="Pinzones de Darwin : largo del pico")`

Un poco más sobre los gráficos

- `example(plot)`
- `example(plot.default)`
- `example(hist)`
- `demo(graphics)`
- `demo(persp)`
- `library(lattice); demo(lattice)`
- `library(plotrix); demo(plotrix)`

Concurso de belleza

Ejercicio

- Trazan $f(x) = \sin(x)$ con $x \in [-2\pi, 2\pi]$ ¡la más linda posible!

Creación de nuevas funciones

- `nombre <- function(arg1,arg2,...){expresión}`
- `fun <- function(x){x^2}`
- `fun2 <- function(a, b, c = 4, d = FALSE){...}`
- `fun <- function(x){y <- x^2; return(y) }`

Control de ejecución

- `if(condición){ expresión }`
- `if(condición){ expresión else }`
- `for(variable en secuencia){ expresión }`
- `while(condición){ expresión }`
- Para más información, ver [?Control](#)

¡Te toca a ti!

Ejercicio

- Escribir una función *freq* que restituye la frecuencia de los números dados en argumento
- **Pista:** chequeen `?sum`

¡Te toca a ti!

Ejercicio

- Dados $N=40$ y $M=100$
- Usen `runif()` y `matrix()` para crear una matriz aleatoria de dimensiones $N \times M$
- Escriben una nueva función que calcula la media de cada columna.
- **Pista:** Hay por lo menos 3 maneras diferentes (+un bonus)
- **Pista:** chequeen `for` en `?Control`, `?rep`, `?matmult`, `?split`, `?sapply`