Curso de introducción al software estadístico R.
Introducción al entorno R: instalación y uso básico. Librerías.

Profesores: Carlos Pérez González y Roberto Dorta Guerra

Bienvenidos a R

Contenidos

1. Introducción al entorno R: instalación y uso básico. Librerías.
2. Conceptos básicos del lenguaje R: instrucciones sencillas, scripts y funciones sencillas.
3. Introducción a la interfaz Rstudio.
4. Gráficos en R: el comando plot() y librerías específicas.
5. Librería RCommander y técnicas de análisis estadístico.

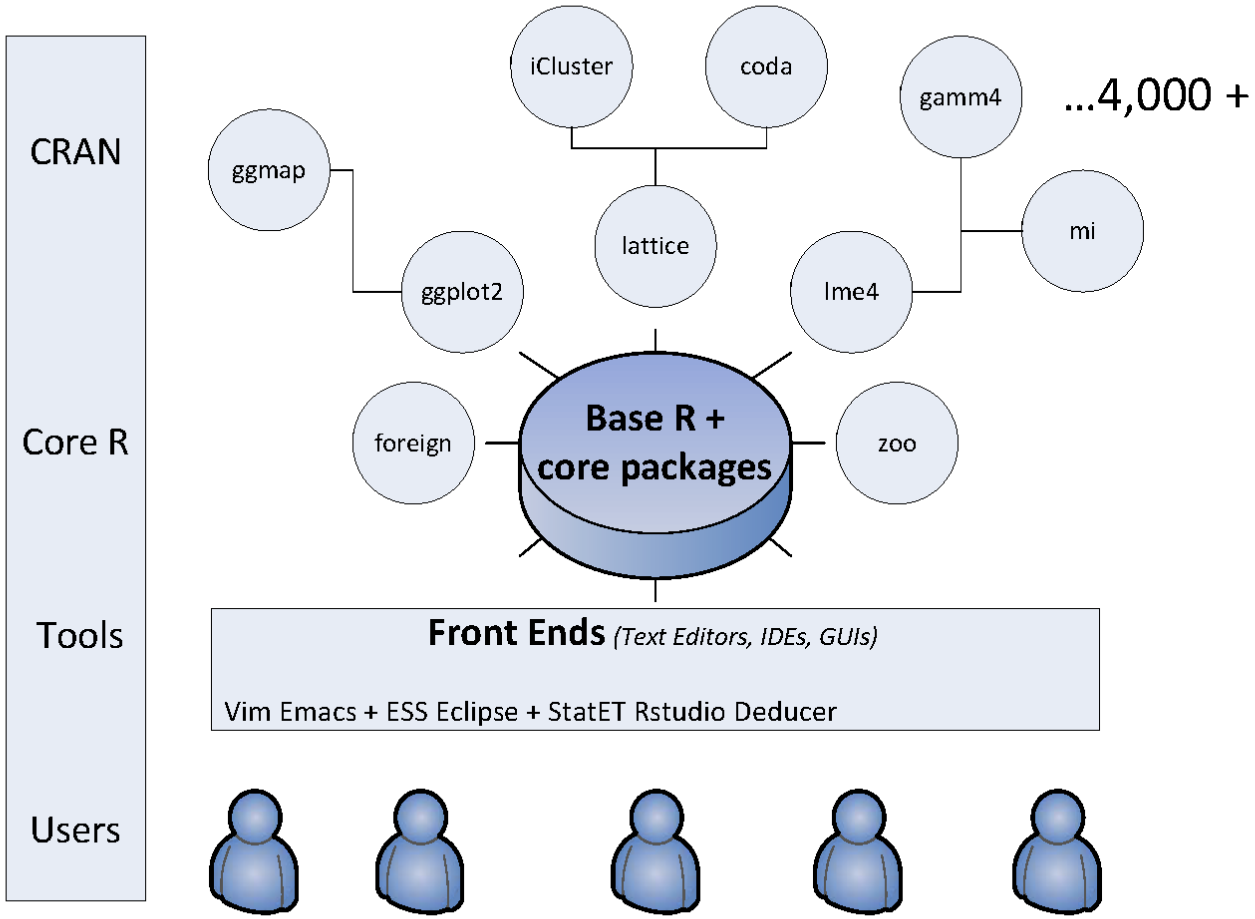
Introducción al entorno R: instalación y uso básico. Librerías.

¿Qué es R (y qué es lo que no es)? (I)

- R es un entorno de libre acceso para computación estadística y gráficos así como para programación. Tiene una serie de características que lo hacen bastante asequible cuando empezamos a trabajar por primera vez con él.
  - Utiliza un lenguaje muy bien desarrollado pero simple.
  - Su funcionalidad es la de un intérprete de comandos (se ejecutan las instrucciones a medida que las introducimos) y la de un lenguaje de programación (se puede realizar un script o conjunto de instrucciones a modo de programa).
  - La instalación por primera vez proporciona un entorno básico de R casi sin ninguna dificultad.
  - Permite un rápido desarrollo de nuevas herramientas para resolver problemas.
  - Estos desarrollos se distribuyen en forma de librerías que cualquier usuario puede descargar e instalar para personalizar su propio entorno de R.

- R no es un lenguaje de programación que permita realizar aplicaciones ejecutables. Sin embargo, se pueden programar interfaces gráficas para facilitar el uso de los programas a los usuarios pero es preciso que estos tengan instalado R.

¿Qué es R (y qué es lo que no es)? (y II)



¿En qué me puede ayudar?

- Tenemos miedo a utilizar esto y luego venga otra cosa que lo sustituya. . . .
- Sin embargo, si lo pensamos tranquilamente, muchas de las cosas que necesitamos hacer habitualmente (gráficas, problemas de estadística, etc..) no necesitan de un software complicado (SPSS, SAS, etc..).
- Podemos utilizar R para resolver de forma más o menos fácil (depende de lo que hayamos entendido en clase de estadística) muchos de estos problemas. A medida que uno desarrolle trabajos de más dificultad (trabajos, proyectos fin de grado o máster, investigación, etc..) R nos será de mayor utilidad.

¿Por donde empezamos? R Base y librerías o paquetes. Rstudio

- En principio, se comienza descargando el paquete básico de R (Base R)
  - Ir a www.r-project.org(http://www.r-project.org) y descarga la versión de RStudio para tu OS.
- El paquete base viene con un conjunto de herramientas básicas de gestión de datos, análisis y funciones gráficas.
- A medida que uno necesita más herramientas, se pueden descargar e instalar entre las +4000 librerías que hay en CRAN las que puedan sernos de ayuda.
  - Una librería que utilizaremos en este curso es Rcmdr (RCommander). La instalaremos desde el terminal de comandos de R.
- R se puede utilizar básicamente con lo que tenemos en la instalación base (terminal de comandos) pero podemos instalar, de forma adicional, algún otro entorno gráfico que facilite su uso (Rstudio).
  - Ir a www.rstudio.org(http://www.rstudio.org) y descarga la versión de RStudio para tu OS.

Ya tengo esto instalado. . . .

- Empezaremos adquiriendo conocimientos básicos de R.
  - Se trata de iniciarse en el manejo de R (comienza como exploreR para llegar a ser un useR). Hay excelentes tutoriales en internet para mejorar aprendizaje.
    - Por ejemplo, codeschool tiene este tutorial online R para ensayar con R solo con el navegador, sin necesidad de instalar nada.
    - Hay muchos recursos como tutorial auto-aprendizaje o UCL wiki.
- Aunque la curva de aprendizaje de R puede ser pronunciada, el soporte de la comunidad de usuarios es muy grande.
  - La comunidad es muy activa en foros, páginas web,. . .
  - Buscar en Google una duda siempre retorna algo útil.
  - Familiarízate con stackoverflow
  - Si conoces otro software, hay bastante ayuda para el paso a R
    - R para usuarios Stata
    - R para usuarios Matlab

Conceptos básicos del lenguaje R: instrucciones sencillas, scripts y funciones sencillas.

La consola de R. Operaciones elementales

- El código R puede ser introducido directamente en la consola de comandos o mediante un script. Por ejemplo, se pueden realizar operaciones aritméticas elementales

```
# operaciones básicas
7 + 3
7 * 3
7/3
7%%3
7/0
```

- Observar que los comandos están separados con un salto de línea (o también con ;). Sin embargo, si el comando comienza con el caracter # esa línea no se ejecuta (p.e., apropiado para añadir comentarios explicativo)

- Todo en R (variables, vectores, funciones, gráficos, fórmulas, . . .) son objetos.

La consola de R. Operaciones elementales (II)

- El objeto más elemental es una variable. Para crear una variable (x) debemos asignar un valor (número, palabra, . . .) mediante `<-` or `=` . Se recomienda `<-` .

```
x <- 3
```

- Tengamos en cuenta que R es sensible a mayúsculas: `x` es diferente de `X`
- Una reasignación sobreescribe el contenido de la variable

```
x <- "ahora x es una cadena."
x
x <- c(3, 4, 5, 6, 7, 8)  # c() combina valores en un vector.
x
```

- Acceder a elementos de un vector `x` con `x[n]`

```
x
x[3:6]  # elementos 3,4,5,6
x[c(1, 3, 6)]  # elts 1,3 y 6
x[-c(1, 3, 6)]  # todos elts excepto 1,3 y 6
```

Ejercicio 1

- crear un vector x con valores 0, 3, 6, 9, 12, 15, 18
- crear un vector y con las palabras {“mis”,“números”,“favoritos”,“son”}
- crear un vector z con todos los elementos de y los 2 últimos elementos de x
- ejecutar `typeof(x)`, `typeof(y)` y `typeof(z)`
- crear un vector w con valores el logaritmo de cada elemento de x

Vectores

- Veamos algunos ejemplos con vectores

```
x <- c(1:3)
x + 1  #suma 1 a cada elemento del vector
x/2  #divide por 2 cada elemento del vector
length(x)  # número de elementos del vector
max(x)
min(x)
sum(x)  #suma todos los elementos del vector
log(x)  #aplica la función log() a cada elemento del vector
```

- Observar que estamos utilizando las primeras funciones de R (length, max, min, . . .). Si se desea obtener la ayuda en particular de una función debemos escribir ? seguido del nombre de la función

```
`?`(sum)
`?`(mean)
```

Vectores (II)

- Ahora, veamos como son las operaciones entre vectores

```
x <- c(1:3)
y <- 4:6  # también se pueden generar secuencias numéricas sin usar c()
x + y

x <- 1:4
x + y  # R da un warning si se suman de diferente longitud

x <- 1:3
y <- 4:6
x * y  # producto elemento a elemento
```

Matrices

- Veamos como se crea una matriz en R:

```
m1 <- matrix(data = 1:9, nrow = 3, ncol = 3, byrow = TRUE)
m1

m1[1, 2]
m1[, 2]
m1[1, ]
m1[c(1, 3)]
```

- Seleccionemos partes de una matriz

```
dim(m1)
m1[1, 2]
m1[2, ]  # fila 2, todas columnas
m1[, 1]  # todas las filas, columna 1
m1[c(1, 3), c(2, 3)]  # filas 1,3 y columnas 2,3
```

- Algunos ejemplos de operaciones con matrices

```
m1 + m1
m1 * m1  # de nuevo, producto elemento a elemento
```

Otras operaciones de interés con vectores y matrices

- Al contrario que con los vectores, ciertas operaciones con matrices no son posibles

```
m2 <- matrix(data = 1:12, nrow = 4, ncol = 3, byrow = FALSE)
m1 + m2   # error
m1 * m2   # error
```

- ¿Y las operaciones usuales de vectores (producto escalar) y matrices (prod. matricial)? Se utiliza el operador %*%

```
x %*% y

m2 %*% m1
m1 %*% t(m2)
m1 %*% m2
```

- Para hallar el determinante y la inversa de una matriz

```
m3 <- matrix(c(1, 2, 3, 4), nrow = 2, ncol = 2, byrow = TRUE)
det(m3)   #determinante
solve(m3)   #inversa
```

Ejercicio 2

- crear una matriz A de 2x2 con la primera fila 2,3 y la segunda 6,7
- crear una matriz b de 2x1 con los elementos 1,5
- resolver la ecuación Ax=b

Operaciones lógicas (comparaciones)

- Veamos como se hacen comparaciones en R

```
x <- 1
y <- 3
x == y  # comparar variables
x < y
x > y

x <- c(1, 2, 3, 4)
y <- c(1, 3, 2, 4)
x == y  # comparar vectores
x < y
x > y
```

- Veamos como obtener los elementos de un vector que satisfacen una condición

```
x[x > y]  # elementos de x que son mayores que los corresp. de y
y[x > y]  # elementos de y que son menores que los corresp. de x
```

Ejercicio 3

- Tenemos un vector x con los elementos 1,2,..,9,10 que identifican a 10 personas (vamos a simularlo con sample(1:10,10))
- Tenemos otro vector y con el peso de estas 10 personas (vamos a simularlo con rnorm(10,75,10))
- Hallar el peso de las personas con id. menor o igual a 5
- Hallar las personas con peso menor de 74 kg (determinar la persona y su peso).

Estructuras de datos. La instrucción data.frame()

- Un conjunto de datos estándar se representa en R como un data.frame
- Es una estructura similar a `matrix`
- Las filas son las observaciones y las columnas son las variables (numéricas o categóricas)

```
datos <- data.frame(aula = rep(1:3, each = 2), sexo = rep(1:2, 3), horas = rnorm(6,
    8, 2))
dim(datos)  # hay 6 observaciones y 3 variables
datos
```

- NOTA: Todas las columnas son de la misma longitud (un data.frame es rectangular, es como una hoja de cálculo)

- Como hemos dicho, un data.frame es similar a una matriz. Por tanto, para acceder a sus elementos lo hacemos de forma parecida.

```
datos[1, ]  # observación 1
datos[, 2]  # variable 2
datos[1:3, 1:2]  # ?
```

Estructuras de datos. La instrucción data.frame() (II)

- A las columnas podemos acceder utilizando su nombre

```
datos[, "sexo"]
datos[, c("aula", "sexo")]
```

aunque la más habitual es con el operador `$`:

```
datos$sexo  # la columna sexo (como vector de valores)
datos$sexo[1:3]  # los 3 primeros elementos de sexo
datos$sexo[c(1, 4, 6)]  # ?
```

- Podemos hacer operaciones vectoriales con las variables de un data.frame

```
x <- datos$sexo + datos$horas
datos$dias <- datos$horas/24
```

Filtrar conjuntos de registros en un data.frame

- Supongamos que queremos filtrar aquellas filas del dataset 'datos' que verifican una condición determinada (p.e., las filas tales que 'aula' sea igual a 2 y las filas con 'sexo' igual a 2).

```
datos_filt1 <- subset(datos, aula == 2)  # filas del aula 2
datos_filt1
# Otra forma:
cond <- datos$aula == 2
datos_filt1 <- datos[cond, ]  # filas del aula 2
datos_filt1

datos_filt2 <- subset(datos, sexo == 1)  # filas del sexo 1
datos_filt2
# Otra forma:
cond <- datos$sexo == 1
datos_filt2 <- datos[cond, ]  # filas del sexo 1
datos_filt2
```

Filtrar conjuntos de registros en un data.frame

- Si queremos combinar condiciones utilizamos los operadores `&` (AND) y `|` (OR). Por ejemplo:

```
datos_filt3 <- subset(datos, aula == 1 & sexo == 2)  # filas con aula=1 y sexo=2
# Otra forma:
cond <- (datos$aula == 1 & datos$sexo == 2)
datos_filt3 <- datos[cond, ]  # filas con aula=1 y sexo=2


datos_filt4 <- subset(datos, aula == 1 | sexo == 2)  # filas con aula=1 o sexo=2
# Otra forma:
cond <- (datos$aula == 1 | datos$sexo == 2)
datos_filt4 <- datos[cond, ]  # filas con aula=1 o sexo=2
datos_filt4
```

Ejercicio 4

```
mascotas <- data.frame(
animal=rep(c("gato","perro","mono"),each=2),
sexo=rep(c("M","H"),3),
peso=rnorm(6,15,5)
)
```

- Filtro por una variable: Filtrar las 'mascotas' de tipo 'gato'. ¿Cuántos animales hay?
- Suma agrupada: Hallar la suma de todos los pesos de cada animal (animal='perro', animal='gato',animal='mono'). ¿Cuál es el máximo de las sumas agrupadas por animal?
- Media de una variable por grupos: Hallar el peso medio de cada animal.

Resúmenes por filas y columnas

- Se pueden sumar los valores de las columnas o las filas con las siguientes instrucciones

```
colSums(datos)
rowSums(datos)  #sólo funciona si todas las variables son numéricas
```

- Sin embargo, a veces queremos sumar los valores de una variable agrupados de acuerdo a una variable categórica. Por ejemplo, sumar todas las horas para cada aula o cada sexo en 'datos'.

```
by(datos[, "horas"], datos[, "aula"], sum)  # suma las horas para cada aula
by(datos[, 3], datos[, 1], sum)  # cuidado con esta forma que puede dar lugar a errores
```

- En R puede haber múltiples instrucciones para sumar por grupos.

```
aggregate(horas ~ aula, sum, data = datos)  # utilizando una fórmula
```

Ejercicio 5

- Suma agrupada: Hallar la suma de todos los pesos de cada animal (animal=‘perro’, animal=‘gato’,animal=‘mono’).
- Media de una variable por grupos: Hallar el peso medio de cada animal.