4. Fundamentos de R

4. Fundamentos de R

4.1 Instalación

R es un lenguaje adecuado para análisis estadístico y diseño avanzado de gráficos. R se puede descargar desde la página http://www.r-project.org/ Lo primero será seleccionar el 'mirror' más adecuado para que la instalación sea rápida. En nuestro caso es http://cran.es.r-project.org/

La principal característica de R es que permite programar todos los pasos asociados a los procesos estadísticos y a la elaboración de gráficos lo que facilita un control total al usuario. Esto hace que el sistema sea muy potente pero a la vez hace que el usuario deba ser muy cuidadoso al elaborar sus 'scripts' ya que cualquier cosa que se no se indique expresamente no será ejecutada por el programa. El programa R puede instalarse desde estén enlace: http://www.r-project.org/ Para acceder a manuales de R con las últimas novedades es conveniente visitar este enlace: http://cran.r-project.org/manuals.html Además hay muchos enlaces con ayudas y ejemplos de programas (en algunos casos es necesario tener una base sólida antes de comprender los programas que se muestran). A continuación se expone una lista de enlace de ayudas y ejemplos:

- Quick R: http://www.statmethods.net/
- R Programming:http://manuals.bioinformatics.ucr.edu/home/programming-in-r

Además debemos instalar el programa RStudio que es un IDE (*Integrated Development Environment)* que permite programar en R. RStudio es un programa *open source* que se puede descargar gratis de la siguiente dirección: http://www.rstudio.com/ Hay otros IDE (como Eclipse: http://www.eclipse.org/ RkWard: http://rkward.sourceforge.net/ o RCommander: http://www.rcommander.com/) que pueden resultar de interés pero nosotros por su sencillez nos centraremos en RStudio. Además de instalar R y RStudio, conviene instalar un editor de textos avanzado como Notepad ++ que permite generar códigos para programas informáticos de forma sencilla. Notepad ++ puede descargarse de forma gratuita de http://notepad-plus-plus.org/

Una vez hemos descargado el ejecutable (.exe) de R desde el servidor de CRAN, lo que tenemos que hacer es ejecutar el programa (si ya tenemos instalado R, lo mejor es desinstalarlo antes de instalar la nueva versión). En la instalación podremos seleccionar el idioma, la carpeta donde queremos que se instale y el tipo de instalación (recomiendo que se

instale la de usuario, es decir la estándar que recomienda el programa) Cuando ya tengamos R instalado podemos proceder a instalar RStudio que nos facilitará el uso de R ya que dispone de ispone de una consola y un editor de texto con características muy útiles como como el resaltado de sintaxis del código (un sistema de colores que también podemos ver en Notepad++), un sistema para depurar errores, función de autocompletado de código con sugerencias de como terminar la instrucción que se está escribiendo, un sistema para instalar paquetes (sin teclear las instrucciones), un visor de gráficos que permite exportarlos en diferentes formatos ... En este vídeo podéis como instalar los dos programas (R y RStudio):

https://web.microsoftstream.com/embed/video/d002f1ee-414f-4867-83bc-00913c88d6e5?autoplay=false&showinfo=true

En esta parte del curso vamos a utilizar el manual de análisis de datos selvícolas (Bravo et al, 2015) que tenemos publicado en el repositorio documental de la Universidad de Valladolid. En este vídeo podéis ver como explorar este repositorio:

https://web.microsoftstream.com/embed/video/23a50457-d160-4646-84b9-7f3306027751?autoplay=false&showinfo=true

Todos los datos que se necesitan para seguir este manual están disponibles en este ENLACE.

4.2 Aspectos básicos

Instrucciones básicas

R es un potente programa que mediante comandos nos permite hacer potentes análisis estadísticos, gráficos profesionales y mucho más. Es importante tener en cuenta que R solo nos devuelve lo que le pedimos por tanto hay que ser cuidadoso para solicitar todo lo que precisemos. A continuación se presentan algunas instrucciones básicas que nos ayudarán a programar de forma eficaz:

1. # significa que lo que sigue es un comentario y que el programa no lo considera comando. Por tanto, # es muy útil para comentar el programa y entender qué se está haciendo. Hay comentar todo lo que sea posible porque si no se hace es fácil que dentro de un tiempo no entendamos el porqué y para qué de las instrucciones del programa de R que hayamos realizado
2. ?*** Nos conecta a internet y nos busca la ayuda sobre el comando (***) que hayamos escrito
3. Al describir las rutas se debe poner / en lugar de \ y escribirlas siempre entre " "
4. <- significa =
5. R es 'case sensitive' por lo que debemos estar atentos a las mayúsculas y minúsculas así 'DATA' no es lo mismo que 'data' ni que 'Data'

6. Es frecuente que algunos de los programas que escribamos necesiten instalar alguna librería para poder acceder a las funciones que deseamos ejecutar. Las librerias (o paquetes) deben instalarse (si no se ha hecho en algún momento anterior) y llamarse cada vez que corramos el programa)

Aunque no tiene porque ser así vamos a considerar que por defecto cada vez que actualicemos R (instalemos una nueva versión) habrá que cargar de nuevo las librerías. No hay que porque hacerlo inmediatamente; podemos hacerlo según surja la necesidad de utilizar cada una de las librerías.

Quick-R tiene multitud de ejemplos para principiantes. Aprovecha para darte una vuelta por

esta web que te ayudará a largo de nuestro trabajo con R.

Comencemos con RStudio

A continuación vamos a conocer la estructura básica de Rstudio y algunas utilidades básicas.

https://web.microsoftstream.com/embed/video/d151b306-62c9-49f7-bb62-af670d0eace7?autoplay=false&showinfo=true

Felipe Bravo. *Comenzando con RStudio* (CC BY)

Trabajar con las librerías (paquetes)

Como ya hemos comentado, algunos de los programas que nos interesará hacer necesitan instalar alguna librería o paquete. Para instalar la librería deberemos usar una orden denominada INSTALL (para instalar la librería) y dentro del programa REQUIRE o LIBRARY (para llamarla). La instalación de los paquetes solo hay que hacerla una vez (salvo si reinstalamos R y perdemos las librerías) pero en cada programa hay que cargarlas.

Así, para instalar la librería 'plyr' (por ejemplo) debemos escribir lo siguiente:

```
install.packages('plyr')
```

para llamarla debemos escribir

```
require('plyr')
```

o

```
library('plyr')
```

El *script* para instalar y utilizar una librería será entonces:

```
install.packages('plyr')
require('plyr')
```

En resumen,

- En muchos casos hay que instalar y llamar (requerir) librerías para realizar determinados procedimientos en R.
- Solo hay que instalar las librerías la primera vez (aunque cuando se actualiza la versión de R hay que volver a instalar las librerías) pero hay que llamarlas cada vez que se ejecute el programa.
- Todas las librerías disponibles en R pueden consultarse en http://cran.r-project.org/web/packages/

Importar datos a R

Lo normal es tener una base de datos asociada (BD) que puede contener por ejemplo: datos de inventario, pesadas de una balanza,... y que pude estar en en Excel, en formato delimitados por coma,.... Para ello, lo primero que tenemos que hacer es importar los datos a R. Es aconsejable disponer de un directorio de trabajo que nos permita referir todos los archivos a abrir o guardar a una misma ruta. Por ejemplo, creemos una carpeta en C:\ que se llame datos (en este paso cada usuario puede crear la carpeta donde crea más oportuno). En esta carpeta tendremos todas las bases de datos asociadas a los ejercicios de este manual. Los nombres de las variables no deben contener espacios y es aconsejable que solo tengan caracteres alfanuméricos (letras y números)

Recordad que el texto en verde puede copiarse y pegarse en Rstudio para correrlo directamente.

El primer paso será crear el espacio de trabajo asociado a esa carpeta. Es decir si no se especifica otra cosa, R entenderá que los archivos que se importen o exporten estarán referidos a la carpeta de trabajo. La instrucción que utilizaremos es setwd (de "set working directory" o establecer directorio de trabajo):

```
setwd('C:/datosR')
```

Como ya se comentó antes hay ser estricto con la nomenclatura (R es 'case sensitive) y utilizar comillas y slash hacia la derecha (/) asi como llamar a la carpeta (y la ruta) exactamente igual que en el ordenador. En lugar de *datosR* se debe poner a ruta exacta donde queremos trabajar en nuestro ordenador. La ruta que pongamos ya debe estar creada en nuestro ordenador antes de ejecutar esta instrucción.

Una vez que ya tenemos definido el directorio de trabajo podemos comenzar a importar los datos. R es muy flexible y permite importar datos de diferentes formatos. Sin embargo, para trabajar con los casos de curso se aconseja que los datos se guarden formato csv delimitado por comas.

Los bases de datos en formato csv se importan mediante la siguiente instrucción:

```
data0<-read.csv('dendro.csv', sep=';', dec=',', header=TRUE)
```

Salvo que se indique otra cosa los datos con los que se va a trabajar en el curso se pueden descargar de este enlace: http://sostenible.palencia.uva.es/sites/default/files/manuales/datos_0.zip descomprime el archivo zip que se descarga de la ruta y coloca el archivo con los datos (*dendro.csv*) en tu directorio de trabajo.

En esta orden le estamos indicando que lea los datos llamados dendro, que nos separe los datos mediante (;), que los decimales los tiene con separación de comas (,) y que la primera fila contiene el encabezado de las variables (header=TRUE). Para comprobar que los datos importados lo han sido de forma correcta, utilizamos la orden

```
head(data0)
```

La instrucción *head* nos muestra los 6 primeros registros de la base de datos. Si queremos ver más tendremos que escribir head(data0, n) donde n es el número de registros que queramos

visualizar.

Antes de probar a hacer esto en R, vamos a conocer un poco el formato csv que es un formato abierto en el que las observaciones están separadas por comas (csv: *comma-separated values*) CSV es un formato el que las columnas (observaciones) están separadas por comas mientrás que los registros (observaciones) están delimitadas por un salto de línea. Es importante resaltar que cuando usamos la coma como separador decimal, el separador de registros es el punto y coma.

Así la instrucción para importar datos separados por comas es la siguiente

```
data0<-read.csv('dendro.csv')
```

mientrás que si el separador es el punto y como escribiremos lo siguiente

```
data0<-read.csv2('dendro.csv')
```

o como vimos antes

```
data0<-read.csv('dendro.csv', sep=';', dec=',', header=TRUE)
```

En este vídeo podemos ver cómo hacerlo

https://web.microsoftstream.com/embed/video/7269fb91-fe0a-4361-a873-89f5ed6e8bf5?autoplay=false&showinfo=true

Felipe Bravo. *importar datos csv* (Dominio público)

Si los datos estuvieran en formato txt , separado por tabulaciones, las instrucciones anteriores serían:

```
data0<-read.table('dendro.txt', sep='\t', dec=',',header=TRUE)

head(data0)
```

Se puede indicar otro formato de datos, otra separación de los datos y otra separación de los

decimales pueden tener otras opciones, como

```
data<-read.table("DATOS1.txt", sep="\t", dec=".", header=TRUE)
```

Finalmente, se pueden leer datos directamente desde la web indicando el formato adecuado. Por ejemplo en el caso de datos en formato txt que estén en un servidor web la instrucción sería la siguiente:

```
data0 <- read.table('http://sostenible.palencia.uva.es/sites/default/files/manuales/datos1.txt', sep="\t", dec=".", header=TRUE)
```

y si están en otro formato bastaría con adaptar la extensión del archivo.

Las instrucciones anteriores pueden abreviarse creando un objeto que corresponda con la dirección web donde se encuentren los datos. Asi tendremos:

```
Web <- ('http://sostenible.palencia.uva.es/sites/default/files/manuales/datos1.txt')

data0 <- read.table('Web', sep='\t', dec='.', header=TRUE)
```

Se pueden importar archivos de excel (.xlx o .xlsx) con las siguientes instrucciones:

```
library(xlsx)

data0 <- read.xlsx("c:/dendro.xlsx", sheetName = "hoja1")
```

Habrás oído últimamente (o quizá no) que los datos de la pandemia covid19 se cuelgan en la web en formato pdf y que eso es una locura para cualquier analista de datos (sea científico o trabaje como experto o ilustrador para un periódico). En este enlace puedes ver cómo hacerlo (el poder de R viene de estas cosas): https://analisisydecision.es/seguimiento-del-coronavirus-en-espana-por-comunidad-autonoma-extraer-informacion-de-un-pdf-con-r/ (no te preocupes si ahora no lo entiendes, cuando acabes el curso debieras poder hacerlo)

Trabajar con (nuevas) variables

En la base de datos dendro.txt (que debemos importar previamente) podemos generar nuevas

variables mediante las siguientes instrucciones que combinan datos que están en el archivo Libro1a.csv (que también debemos importar previamente):

dendro$suma <- Libro1a$dato2 + Libro1a$dato3

dendro$media <- (Libro1a$dato2 + Libro1a$dato3)/2

O mediante la siguiente expresión para generar una nueva variable en Libro1a:

```
attach(Libro1a)
Libro1a$suma <- dato2 + dato3
Libro1a$media <- (dato2 + dato3)/2
detach(Libro1a)
```

Para crear nuevas variables (o para programar funciones) en R se pueden utilizar, entre otros los siguientes operadores:

Operador	Descripción
Aritméticos	
+	Suma
-	Resta
*	Multiplicación
/	División

^ o **	Exponente

Lógicos

<	Menor que
<=	Menor o igual a
>	Mayor que
>=	Mayor o igual a
==	Exactamente igual a
!=	No igual a
x | y	x o y (alternativamente)
x & y	x e y (adicionalmente)
isTRUE(x)	Comprueba si x es verdadero

En este vídeo podemos ver algunos aspectos básicos de la creación de nuevas variables:

https://web.microsoftstream.com/embed/video/d6e46822-0ae6-4e4f-8c4b-08a57250f363?autoplay=false&showinfo=true

Felipe Bravo. *Crear nuevas variables* (GNU/GPL)

Mezclar bases de datos

A partir de dos bases de datos que contienen al menos una variable en común se puede generar una nueva base de datos que contenga los datos de las dos anteriores. Con las bases de datos Libro1a.csv y Libro1b.csv (que debemos importar previamente) que tienen en común la variable 'dato1' podemos generar una nueva base de datos que contendrá todos los datos de las dos bases de datos anteriores. Para ello utilizaremos la siguiente rutina:

```
setwd('C:/datosR')

datos1 <- read.table("Libro1a.csv", sep = ";",header = TRUE)

datos2 <- read.table("Libro1b.csv", sep = ";",header = TRUE)

datafinal <- merge(datos1, datos2)

head(datafinal, 10)
```

Si la variable por la que se quiere mezclar las bases de datos tuviese un nombre diferente en cada una de ellas habría que emplear la siguiente estructura:

```
datafinal <-

merge(datos1, datos2, by.datos1='nombre1', by.datos2='nombre2')
```

4.3 Análisis descriptivo

El objetivo de este apartado es aprender a realizar un análisis descriptivo básico de un conjunto de datos. En primer lugar importaremos los datos (debemos recordar que # significa que es un comentario que no será ejecutado por el ordenador)

```
setwd('C:/datosR')

data<-read.csv('dendro.csv',sep=';',dec=',',header=TRUE)
```

A continuación vamos a calcular las principales medidas descriptivas de un conjunto de datos. En este ejemplo no hace falta llamar a ningún paquete (recordad que # indica que es un comentario)

```
#media de cada columna

colMeans(data, na.rm=TRUE)

#para medias por filas

rowMeans(data, na.rm=TRUE)

#si queremos sumar los datos de las columnas o las filas

rowSums(data, na.rm=TRUE)

colSums(data, na.rm=TRUE)

# el comando dim te da el número de filas y de columnas

dim(data)

#el comando str te da información sobre el tipo de caracteres

#que tiene cada columna y la tabla como en “diagrama de hojas"
```

```
    str(data)


#la mediana se puede conseguir así, donde tienes que
#especificar la variable y se aplica para cada una de las columnas
    median(data$d1, na.rm=TRUE)


#o con el comando sapply , que es para aplicar en una tabla
#una función determinada
    sapply(data, na.rm=TRUE,median)
#otras variables descriptivas se pueden calcular también con
#sapply como la desviación típica, la varianza y el rango
    #Desviación típica
    sapply(data, na.rm=TRUE,sd)
    #Varianza
    sapply (data, na.rm=TRUE, var)
    #Rango
    sapply (data, na.rm=TRUE, range)


#summary() calcula el mínimo, el máxim, la media, la mediana, el primer y el tercer cuartil.
    summary(data, na.rm=TRUE)
```

Con esto ya hemos podido realizar nuestro primer análisis descriptivo de datos.

4.4 Elaboración de gráficos

En este apartado vamos a conocer los comandos de R para poder realizar un análisis gráfico de un conjunto de datos. Hay paquetes como *ggplot2* que son muy potentes y permiten hacer gráficos mucho más complejos y vistosos pero para poder familiarizarnos con la estructura básica vamos a utilizar solo el paquete *lattice*.

En primer lugar instalaremos la librería lattice (en R Console)

```
#Este paso requiere instalar la librería 'lattice':

    install.packages('lattice')

#La llamamos para que se pueda ejecutar

    require('lattice')
```

Ahora pasamos a leer los datos

```
    setwd('C:/datosR')

    data<-read.csv('dendro.csv',sep=';',dec=',',header=TRUE)
```

A continuación haremos varios gráficos (nube de puntos, un boxplot, un scatter.smooth, un histograma y un contraste de pares entre variables).

Tipo de gráfico	Fución en R

Nube de puntos, líneas	plot()
Boxplot	boxplot()
Scatter somooth	scatter.smooth()
Histograma	histogram()
Contraste entre pares de variables	pairs()

Las funciones están definidas por distintos argumentos. Por ejemplo, en el caso de plot() algunos argumentos pueden ser col,main,type,sub,xlab,ylab

el argumento col define el Color de la gráfica

el argumento main define el titulo de la gráfica

el argumento type define el tipo de grafica a hacer

"p" para una nube de puntos

"l" para un gráfico de líneas

"b" para una gráfica de puntos y líneas

"c" para obtener una grafica de tipo "b" pero sin puntos

"h" para líneas verticales tipo histograma

"s" para gráficos de barras

"n" te indica el sistema de referencia del gráfico, pero sin gráfico

el argumento sub define un subtítulo debajo del título de los ejes x o y.

los argumentos xlab e ylab definen los títulos de cada uno de los ejes

Por ejemplo, si queremos hacer un gráfico exploratorio que compare los dos diámetros recogidos en el muestreo escribiremos lo siguiente (OJO: data$d1 indica a R que debe usar la variable d1 del conjunto de datos data)

Una nube puntos

```
plot(data$d1 $daa
d2,col="red",main="Gráfico de los dos diámetros recogidos en el muestreo",
    ylab="d2(mm)", xlab="d1(mm)",type="p")
```

Visualiza un diagrama de cajas de la variable Htotal por clases de d1

```
boxplot(d1~Htotal,data, ylab = "Htotal (m)", xlab= "d1")
```

(d1~Htotal,data,...) es otra forma de indicar a R que variables de qué variables queremos usar

Para hacer un gráfico exploratorio añadiendo una línea de tendencia suavizada con los puntos en rojo escribiremos

```
scatter.smooth(data$d $daa
Htotal, ylab = "Htotal (m)", xlab= "d2 (mm)", col="red")
```

Para visualizar los datos (Htotal) en un histograma por clases de d2 la instrucción es

```
histogram(~Htotal | d2, data)
```

Si queremos visualiza los datos por pares de variables en una matriz de gráficos escribiremos

```
pairs(~d1+d2+Htotal+Hcopa,data, main="Matriz de Scatterplots")
```

El potencial para hacer R con sus diferentes paquetes es quizá lo que más llama la atención a un principiante. Aunque no lo necesitaremos para el curso te animo a ver cómo se hacen gráficos más complicados e n estas páginas web:

- La galería de gráficos por defecto: https://www.r-graph-gallery.com/
- Uso de paleta de colores: https://datavizpyr.com/color-boxplots-with-r-colorbrewer/
- Cómo hacen los gráficos en la BBC: https://bbc.github.io/rcookbook/

pero conviene recordar que los gráficos también se miente por lo que te recomiendo leer este artículo: https://theconversation.com/3-questions-to-ask-yourself-next-time-you-see-a-graph-chart-or-map-141348