Práctica 2: Introducción a R

Hoy trabajaremos con el fichero cis3145t.sav. Se trata de un fichero del estudio poselectoral de las elecciones de junio de 2016. Todos los ficheros que vamos a utilizar en las prácticas los puedes encontrar en: https://github.com/pablocal/EMASdataviz

En esta práctica introduciremos algunos aspectos básicos de R. Las soluciones al ejercicio las tienes en la carpeta de Ejercicios resueltos.

1. En primer lugar, abre el fichero de código Pracitica2.R . Carga los paquetes que necesitas para realizar la práctica, ejecutando las líneas de **`library`**(). En caso de que alguno de ellos no esté instalado, quita la # de delante del comando **`install.packages`**(), y ejecuta esa línea del código.

   TIP! Para ejecutar una línea de código, posiciónate sobre la línea y pulsa **`Ctrl + Enter`**

2. Una vez cargadas las librerías, limpia el espacio de datos utilizando **`rm(list=ls())`**, con esto quitarás del espacio de trabajo todos los datos que estén abiertos, y así evitarás confusiones. Después utiliza la función **`setwd()`** para establecer tu carpeta de trabajo, donde has copiado los datos "cis3145t.sav". Recuerda que las barras para indicar la ruta deben ser "/", y que la ruta a la carpeta debe ir entre comillas " ".

   Ejemplo:
   `setwd("C:/Mis Documentos/Curso R")`

3. Crea dos vectores (**`x`**, **`y`**), ambos de tres elementos. El primero (2, 4, 6), y el segundo (3, 4, 7). Completa la sintaxis para el vector **`y`**.

4. Ahora que ya tienes los vectores, puedes utilizarlos como datos, y operar con ellos. Estos son los principales operadores que puedes utilizar en R:

Operator	Description	Operator	Description
+	addition	<	less than
-	subtraction	<=	less than or equal to
*	multiplication	>	greater than
/	division	>=	greater than or equal to
^ or **	exponentiation	==	exactly equal to
x %% y	modulus (x mod y) 5%%2 is 1	!=	not equal to
x %/% y	integer division 5%/%2 is 2	!x	Not x
		x | y	x OR y
		x & y	x AND y
		isTRUE(x)	test if X is TRUE

En primer lugar calcula la media de cada vector, para ello usa la función **`mean()`**, y almacena los resultados como **`meanx`** y **`meany`** respectivamente. Como ves, has creado dos nuevos vectores, que ahora aparecen en el panel superior de la derecha de R-studio. Ahora suma los vectores **`x`** e **`y`**, y almacena el resultado como **`sumxy`**, para ello completa sintaxis. Imprime **`x`**, **`y`**, y **`sumxy`**:

x	y	sumxy
2	3	5
4	4	8
6	7	13

Como ves, R ha sumado el primer elemento de `x` con el primer elemento de **`y`**, para almacenarlo en **`sumxy`**, y así sucesivamente. Ahora, multiplica el vector **`x`** por **`meanx`** y almacénalo como **`multx`**, ¿qué ha ocurrido?

Pero además de operaciones aritméticas, también se pueden hacer operaciones lógicas. Por ejemplo, para saber si la media del vector **x** es igual que la media del vector **y**, escribe en el código:

```
meanx == meany
```

Al ejecutar esta línea, R devolverá **TRUE** si es verdadero, y **FALSE** si es falso.

5. Recuerda que los vectores almacenan elementos del mismo tipo, numéricos o de caracteres, mientras que las matrices están formadas por elementos numéricos exclusivamente. Los conjuntos de datos que solemos utilizar en Ciencias Sociales se identifican en R como data frame, en el que las columnas son variables, y las filas casos. Una función útil de R es que a partir de vectores se puede crear un data frame.

   Imagina que el vector **x** que creamos en el paso 3 es la variable edad, y que el vector **y** es la variable número de veces que el niño ha sido recetado con antibióticos. Completando el código, construye el data frame y asígnalo al objeto **mydata**. Lo que hemos hecho es combinar los dos vectores, x e y, como si fueran dos columnas utilizando la función **cbind**().

   Posteriormente **mydata** se ha convertido en un data frame utilizando la función **as.data.frame**(). El data frame **mydata** contiene dos variables **x** e **y**. A partir de ahora para utilizar cualquier variable habrá que llamarla utilizando **mydata$x** o **mydata$y** (nota que se usa el **$** para seleccionar elementos dentro de otro elemento). Ahora calcula la media de la variable **x** del data frame **mydata**. No es necesario que la asignes a un objeto, se imprimirá en pantalla (recuerda utilizar $).

6. Pero muchas de las variables que usamos en ciencias sociales no son numéricas, sino códigos numéricos que indican categorías nominales. Por ejemplo, en sexo, 1 es “Mujer”, y 2 es “Hombre”. Ahora completa la sintaxis para crear el vector **sex**, que contiene tres elementos “masc”, “fem”, “masc”.

7. Ahora crea el data frame **mydata2** uniendo **mydata**, con el nuevo vector **sex**. Esta operación la puedes realizar utilizando la fución **cbind**(), como ya hiciste en el punto 5. El resultado final debe ser un nuevo data frame (**mydata2**) que contiene tres variables. Para asegurarnos que está bien construido imprime los datos en la pantalla ejecutando **mydata2**.

8. Cuatro funciones que pueden ser útiles son **class**(), **levels**(), **as.factor**(), **as.numeric**(). La función **class**() devuelve el tipo de objeto, como en el siguiente ejemplo:

```
> voto <- c("PSOE", "PP", "Podemos")
> class(voto)
[1] "character"
```

   Averigua qué clase de objeto es el vector **sex**, y la variable **sex** dentro de **mydata2**:

   En el caso de un factor, **levels**() es una función que te devuelve los diferentes niveles de ese factor. Para probar esta función, utiliza **levels**() para comprobar los valores del factor sex del conjunto de datos **mydata2**.

En ocasiones es necesario que factores pasen a ser elementos numéricos, para ello se utiliza la función **as.numeric**(). Para experimentar cambia la variable **mydata2$sex**, que es de tipo factor, a numérica. Acuérdate de guardar el resultado como la misma variable, para que la transformación se guarde.

Por el contrario, también se puede convertir una variable numérica en factor, para ello se utiliza la función **as.factor**(). Prueba a convertir la variable que acabas de transformar en numérica (**mydata2$sex**) en un factor de nuevo. Para comprobar la transformación utiliza las funciones **class**() y **levels**(), ¿qué ha ocurrido?

El factor sex fue convertido a numérico, con lo cual los niveles fueron borradao, al volverlo a convertir en factor comprobamos que los niveles son '1' y '2', en vez de 'masc' y 'fem'. Para volver a los niveles originales puedes utilizar la función levels, asignando los niveles deseados:

```
levels(mydata2$sex) <- c('masc', 'fem')
```

9. En la segunda parte de la práctica vamos a trabajar con un data frame real para ver diferentes formas de transformar los datos que puedes necesitar antes de realizar los gráficos. En primer lugar hay que abrir los datos, como la base de datos original está en formato SPSS utiliza la función **read.spss**() del paquete foreign. El argumento **to.data.frame** de **read.spss**() es igual a **TRUE**, esto indica que queremos que R guarde el conjunto de datos como un dataframe. Los datos han sido asignados al objeto **d**.

10. Lo primero al abrir el conjunto de datos es explorar su contenido. Para ello puedes utilizar las funciones **str**(), **view_df**(), **head**(), **tail**(), **summary**(). La función **str**() te devueve la estructura del data frame. Por su parte, **head**() y **tail**() ofrecen una visión de las primeras y últimas observaciones del conjunto de datos. La función **summary**() hace un resumen de las variables según sean numéricas o factores. Utiliza estos cinco comandos para explorar el data frame **d**.

11. Una vez conocido el conjunto de datos, céntrate en las variables que vamos a utilizar. Por un lado, los indicadores de confianza **confparl**, **confpart**, y **confjudic**, por otro las variables **voto15** y **voto16**. Las primeras son variables numéricas referidas al nivel de confianza de los ciudadanos, medido en una escala del 1 al 10, en el parlamento, los partidos, y el poder judicial. Las variables **voto15** y **voto16** se refieren al recuerdo de voto en las elecciones de 2015 y 2016. Para explorar las variables de forma separada puede utilizar tablas: **table**() o **frq**() son dos opciones diferentes. Prueba estas dos funciones que crean tablas de frecuencia para conocer las variables nombradas arriba. Para variables numéricas como confparl, confpart o confjudic, también puedes utilizar la función **descr**() que genera un resumen de estadísticos descriptivos.

12. Lo normal antes de realizar gráficos es que se requieran transformaciones de los datos. Recodificar quiere decir que a una serie de valores viejos le vamos asignar valores nuevos. Por ejemplo, en caso de **voto16** vamos a crear **voto16r** para juntar a podemos con las confluencias, así como aglutinar a todos los partidos pequeños en la categoría "otros". Para ello se utiliza este código:

```
d$voto16r <- recode(d$voto16, "PP"="PP", "PSOE"="PSOE",
              "Cs"="Cs", "ECPodem"="Podemos", "Podemos"="Podemos",
              "Compromis"="Podemos", "En Marea" = "Podemos",
             "NoVoto"="No voto", "NC"="NC", "NR"="NR",
             .default="Otros")
```

En el código **d$voto16r** es el nombre de la nueva variable. La función **recode**() primero necesita el nombre de la variable a recodificar (**d$voto16**), posteriormente, entre comillas dobles (" ") aparecen los

cambios a realizar. Por ejemplo, "PP" = "PP". Además, .default='Otros' indica que todos los valores que no han sido nombrados se recodificarán como otros.

Ejecuta la recodificación de **voto16** en **voto16r**, y comprueba el resultado utilizando **frq**() para generar una tabla de frecuencias de la nueva variable (**voto16r**). A continuación, prueba a hacer una recodificación similar de **voto15** en **voto15r**.

13. Compilar una nueva variable es otra de las transformaciones más habituales. Por ejemplo, si hay varios indicadores de un fenómeno, se pueden sumar para crear un índice. Estas operaciones son generalmente de carácter aritmético, pero pueden incluir elementos lógicos. Ahora vas a crear un índice de confianza en instituciones públicas (**confpub**) que sea la suma de **confparl**, **confpart** y **confjudic**. Antes de empezar a recodificar vuelve a echar un vistazo a las tres variables utilizando **frq**(). Las variables **confparl**, **confpart** y **confjudic** tenían valores 98 y 99 (no sabe y no contesta). Estos valores han sido tenidos en cuenta por R, y los ha sumado para generar **confpub**. Lo ideal sería que, si en alguna de las variables anteriores hay un 98 o 99, el valor en **confpub** sea perdido (excluido para el análisis), que en R es "NA". Para ello hemos hecho una modificación lógica de **confpub**, de forma que si para cada individuo un valor de **confparl**, **confpart**, o **confjudic** es mayor que 97, entonces **confpub** será igual a NA. A continuación, crea la nueva variable, que llamarás **confpub**. La sintaxis está ya lista, solo tienes que ejecutarla.

    Después de ejecutar esta modificación, vuelve ejecuta las funciones **descr**(), **head**() e **hist**() para comprobar el que el cambio ha funcionado.

14. Otra operación recurrente es la de seleccionar casos. En ocasiones solo nos interesa un subgrupo de la muestra para el análisis, y lo que tenemos que hacer es filtrar esos datos para trabajar solo con la submuestra. En R se puede utilizar la función **filter**(). Con esta función se seleccionan una serie de casos que cumplen una regla lógica, por ejemplo:

```
d2 <- filter(d, voto16r == "PSOE" | voto16r == "PP")
```

    Este código crea un nuevo conjunto de datos (**d2**) en el que solo están presentes los casos que en **voto16r**, del data frame **d**, votaron al PP o PSOE. Puedes comprobarlo ejecutando esta sintaxis y haciendo una tabla de frecuencias de **d2$voto16r** (nota que el conjunto de datos ahora es d2). ¿Cuál será el resultado de seleccionar el siguiente subconjunto?

```
d3 <- filter(d, voto16r == "PSOE" | voto16r == "PP" & confpub>20)
```

    Para concluir con este punto crea tu propio criterio y haz una subselección del data frame **d**, asignándolo a **d4**. Haz comprobaciones para asegurarte que el conjunto **d4** contiene las observaciones esperadas.

15. Hay ocasiones en las que se necesitan seleccionar variables en vez de casos, sobre todo cuando partimos de grandes conjuntos de datos. Para ello puedes utilizar la función **select()**. En el siguiente caso, **d5** es un data frame nuevo creado a partir de **d**, en el que se contienen las variables **cuest** y **voto16r**.

```
d5 <- select(d, cuest, voto16r)
```

    Crea un data frame (**d6**) en el que además de **cuest** y **voto16r** contenga todas las variables **conf*** con las que hemos trabajado hoy. Utiliza las funciones que ya conoces para comprobar que **d6** contiene las variables deseadas.

16. Con esto hemos terminado una rápida introducción a R y el manejo básico de datos. Si ere nuevo en R, puedes terminar la práctica haciendo una lista de los comandos más útiles que has aprendido hoy, ya que los necesitarás para completar las actividades del resto del curso.