Manipular datos en R

Ejercicios

Julio 2019

Introducción

En los ejercicios de la segunda sesión vamos a trabajar sobre la manipulación de datos en `R`:

1. **Abre el script** `sesion2_ejercicios.R`. **Limpia** el espacio de datos ejecutando `rm(list = ls())`. Con este comando eliminarás del espacio de trabajo todos los datos (objetos) que estén disponibles evitando posibles confusiones[1].
2. **Carga los paquetes** que necesitas para realizar la práctica, ejecutando las líneas de `library()`. En caso de que alguno de ellos no esté instalado, instaláló utilizando `install.packages("package")`[2].
3. En la carpeta **data** encontrarás los ficheros de datos que vas a utilizar en esta práctica.

[1] Puedes usar el atajo `Ctrl + Enter` para ejecutar una línea de código en `RStudio`

[2] Usa comillas (" ") a la hora de instalar los paquetes con la función `install.packages()`

A. Cargar means `read` & `glipmse`

A.1 Lo primero que vas a hacer es **cargar** un archivo de tipo `.csv` y almacenarlo como `my_csv`. En la carpeta `data` hay un archivo con el nombre `ine_valladolid.csv` que contiene los códigos y nombres de municipios de la provincia. Utiliza la función `read_csv2()`. Explora los datos. ¿Qué problemas identificas?

`read_csv2(file, col_names = TRUE, locale)` Devuelve un data frame a partir de un archivo `.csv` separado por punto y coma (;).

`str(x)` Devuelve la estructura de `x`.

`glimpse(x)` Devuelve un resumen del data frame.

A.2 Vuelve a **cargar los datos del ejercicio anterior**. En esta ocasión utiliza el argumento `locale` para especificar el `encoding`. Prueba con `Latin1`. Una vez cargados los datos, convierte la variable `CMUN` a numérica. Comprueba que los datos han sido cargados correctamente.

A.3 Ahora vas a **cargar** la base de datos `cis_oct17.sav` utilizando la función `read_spss()` del paquete `haven`. Almacena el objeto como `my_spss` y explora la estructura de los datos. ¿De qué tipo es la variable `tamuni`? ¿Y la variable `sexo`?

`read_spss(file, user_na = FALSE)` Devuelve un data frame de tipo *labelled* a partir de un archivo `.sav`.

A.4 Convierte las columnas de tipo `labelled` en factores con la función `as_factor()` y alamacena el resultado como `my_spss_factor`. Explora la estructura de los datos. Fíjate en la variable `edad`, ¿qué tiene de particular?

`as_factor(x)` Devuelve el objeto convertido en factor.

A.5 Convierte la varaible `edad` en una de tipo numérico y comprueba el resultado de la transformación.

B. Seleccionar variables

`summary(x)` Devuelve un resumen estadístico del objeto `x`.

`select(.data, ...)` Devuelve un data frame en el que se han seleccionado las variables especificadas en `...`.

`ends_with(match)` Dentro de `select` devuelve un data frame con las columnas terminadas en `match`.

B.1 Primero, **explora** el data frame utilizando las funciones `summary()` y `glimpse()`.

B.2 Selecciona las **variables** `edad` y `estud` del conjunto de datos `my_spss_factor`. No guardes el cambio, pero imprime el resultado.

B.3 Ahora **selecciona** las variables cuyos nombres terminan por la letra **d**. Utiliza para ello la función auxiliar `ends_with()`. No guardes el resultado.

B.4 Crea un **nuevo data frame** en el que figuren todas las variables presentes en `my_spss_factor` **excepto** `ocupa`. LLama al nuevo data frame `cis`.

C. Ordenar y filtrar casos

`arrange(.data, ...)` Devuelve un data frame ordenado por las variables especificadas en `...`.

`desc(x)` En `arrange` determina el sentido descendente del orden por la variable `x`.

`filter(.data, ...)` Devuelve un data frame en el que se han filtrado los casos según las condiciones especificadas en `...`.

`group_split(.data, ...)` Devuelve una lista con data frames filtrados según los niveles de un factor.

C.1 Ordena el data frame `cis` por las variables `region` (ascendente) y `edad` (descendente). Comprueba el orden del data frame utilizando la función `View()`. Cuando hagas una transformación **no olvides volver guardar el objedo** para no perder el cambio en el data frame.

C.2 Selecciona una **submuestra de los casos** del data frame `cis` que contenga los casos de `Andalucía`, `Madrid (Comunidad de)` y `Cataluña`. Para realizar el filtrado intenta usar el operador `%in%`, que sirve para hacer selecciones múltiples y así evitar el uso reiterado del operador lógico `|`. Guarda el data frame resultante con el mismo nombre.

C.3 Ahora crea una lista de **data frames segmentados** a partir de la variable `region` utilizando para ello la función `group_split()`. Guarda la lista de data frames como `cis_split`. Comprueba la estructura de `cis_split`.

C.4 Realiza una **tabla** de la variable `idv` correspondiente a `Andalucía` a partir del objeto `cis_split`. Para realizar la tabla puedes utilizar la función `table()`.

D. Transformar variables y renombrar

`mutate(.data, ...)` Devuelve un data frame con variables transformadas.

D.1 Crea una variable que sea la **suma** de la variable `edad` en `cis` y 50. Llama a la nueva variable `edad_rec1`. Comprueba la nueva

variable con la función `head()` mostrando solo las variables `edad` y `edad_rec1`. Para hacer la selección de las variables usa `select()`.

D.2 Ahora crea una variable (`edad_rec2`) que sea la **suma de edad y un número aleatorio** entre 1 y 50 **diferente para cada caso**. Para generar el número aleatorio utiliza la función `runif()` con un mínimo de 1 y un máximo de 50. Comprueba la transformación de la variable con `head()`.

`recode(x, ..., .default, .missing)` Devuelve un vector recodificado.

D.3 Recodifica la variable `idv` del data frame `cis` en `idv_rec` de forma que tenga seis categorías: `PSOE`, `PP`, `UP`, `Cs`, `Otros` y `No vota`.

1. Utiliza la función `levels()` para **conocer los niveles** de la variable original (`idv`)
2. Los valores `N.C.` tansfórmalos en **perdidos** (`NA`)
3. Utiliza la función `mutate()` en combinación con `recode()`
4. Para simplificar el trabajo utiliza el argumento `.default` de la función `recode()`, que establece la categoría por defecto a la que pertecen los casos en la variable recodificada

¿De qué **tipo** es el vector `idv_rec`?

`case_when(...)` Determina una serie `...` de condiciones y acciones separadas por `~`.

D.4 Crea la variable `edad_group` con **4 grupos de edad** `18-29`, `30-44`, `45-64`, `65+`. Para ello utiliza la función auxiliar `case_when()`. Haz una tabla para comprobar la transformación de la variable.

D.5 Crea una nueva variable (`idv_sexo`) que sea una **combinación** de `idv_rec` y `sexo` para los casos de `PSOE` y `PP`, mientras que el resto de los casos será especificado como `"Otros"`. Fíjate en la siguiente tabla:

idv_rec	sexo	idv_sexo
PSOE	Hombre	PSOE Hombre
PSOE	Mujer	PSOE Mujer
PP	Hombre	PP Hombre
PP	Mujer	PP Mujer
Otros	Indiferente	Otros

`ifelse(test, yes, no)` Devuelve un objeto modificado según una condición.

D.6 Ahora vas a crear una **nueva variable edad** (`edad_rec3`) en la que los menores de 25 años tengan un valor perdido (`NA`) y el resto su valor orignal en la variable `edad`. Para ello utiliza la función `ifelse()`. Comprueba la transformación.

`rename(.data, ...)` Devuelve un data frame con los nombres de las columnas modificados según `...`.

D.7 Cambia el **nombre** de la variable `edad_rec3` a `edad_na`. Para ello utiliza la función `rename()`. Utiliza la función `colnames()` para comprobar que el nombre de la variable se ha cambiado correctamente.