# Curso completo de machine learning con R de cero a experto

[Curso completo de machine learning con R de cero a experto 1](#__RefHeading___Toc76_1571306296)

[Varios 2](#__RefHeading___Toc78_1571306296)

[Tema 1: Preparando los datos para el análisis 2](#__RefHeading___Toc80_1571306296)

[El fichero CSV y la función read.csv 3](#__RefHeading___Toc82_1571306296)

[Leyendo datos de un XML 6](#__RefHeading___Toc84_1571306296)

[Leyendo datos de un JSON 7](#__RefHeading___Toc109_4065685023)

[Ficheros de ancho fijo 7](#__RefHeading___Toc111_4065685023)

[Creando ficheros Rdata y rds 8](#__RefHeading___Toc132_1444951313)

[Eliminar datos sin valor con na.omit 9](#__RefHeading___Toc134_1444951313)

[Evitando duplicaciones de entradas 11](#__RefHeading___Toc168_4294363079)

## Varios

Comunidad:

<https://discord.gg/Gq5NX6a>

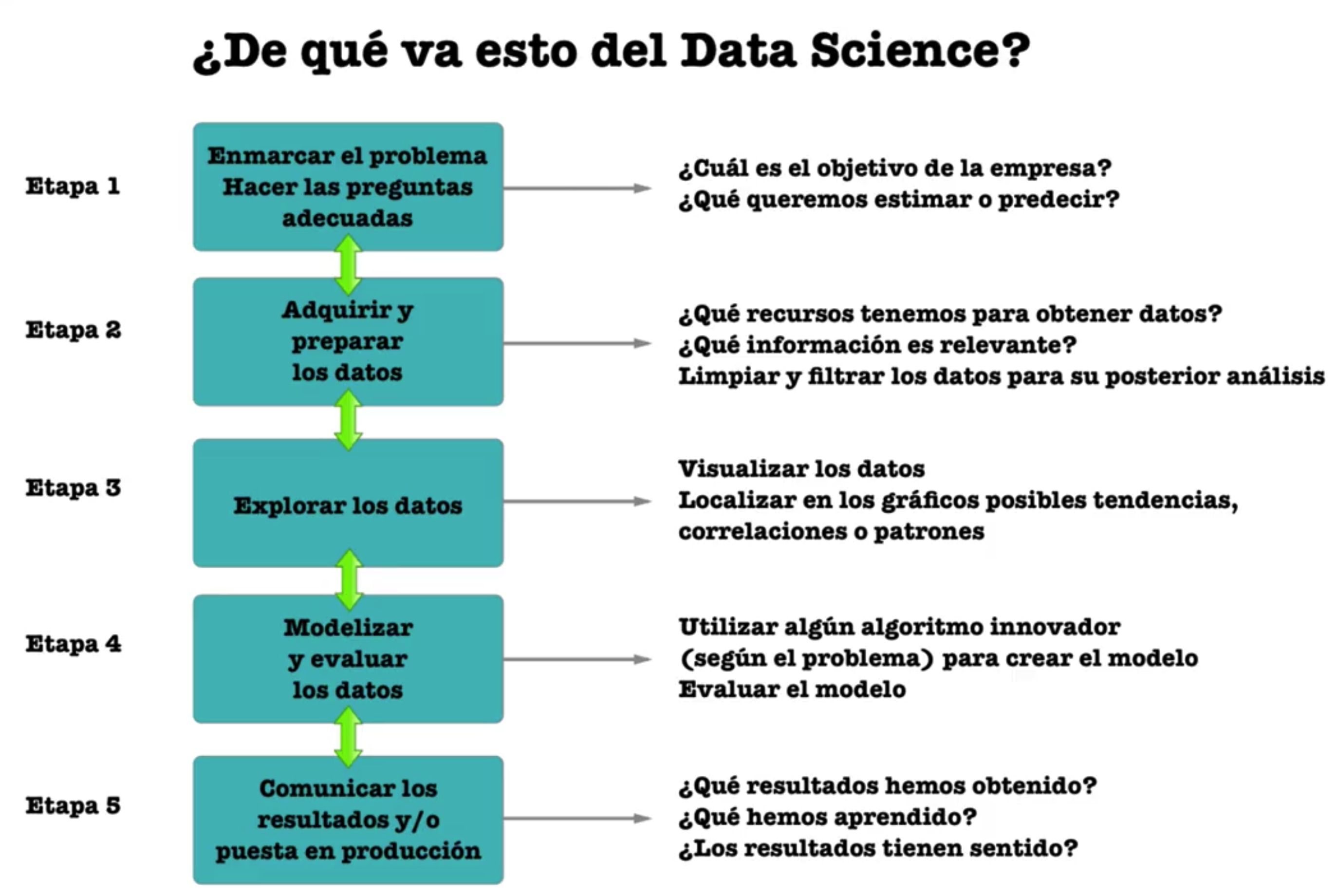
Cheatsheets:

<https://www.rstudio.com/resources/cheatsheets/>

Repositorio github:

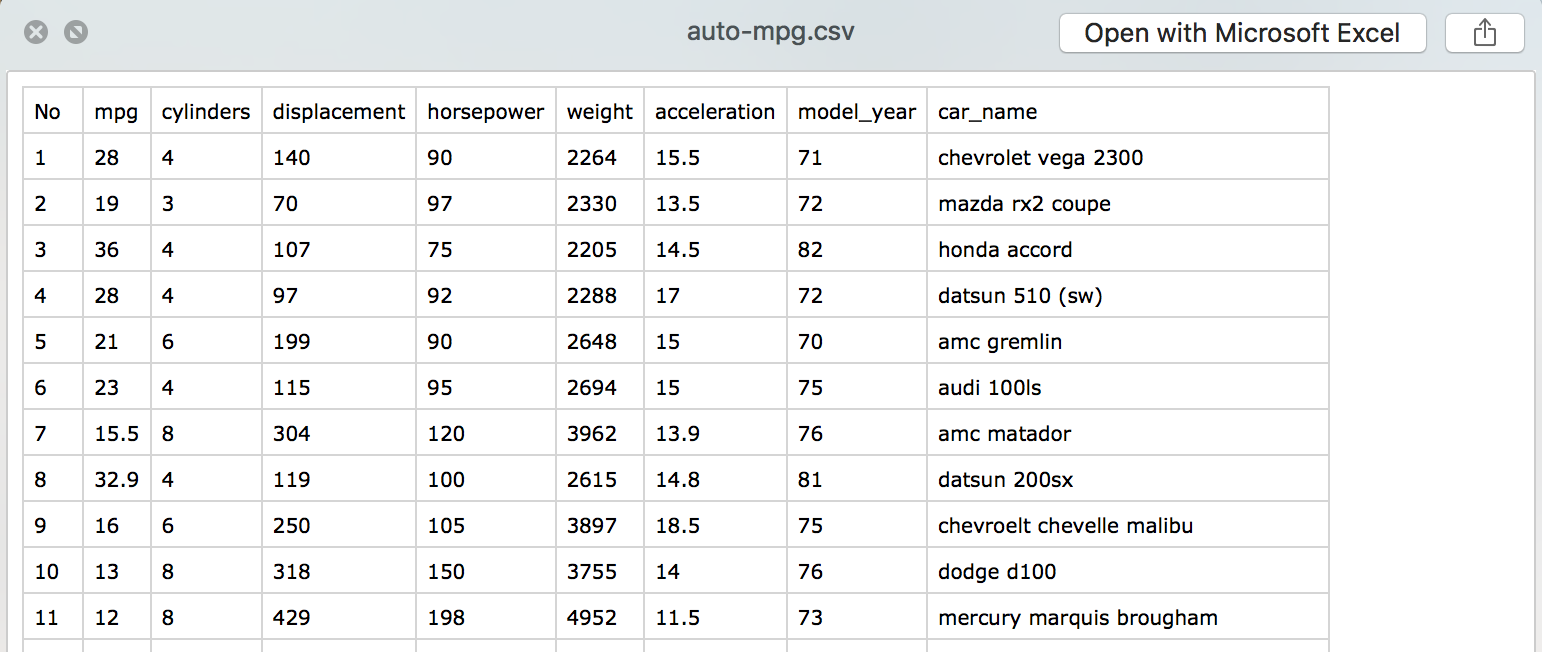
<https://github.com/joanby/r-course>

## Tema 1: Preparando los datos para el análisis

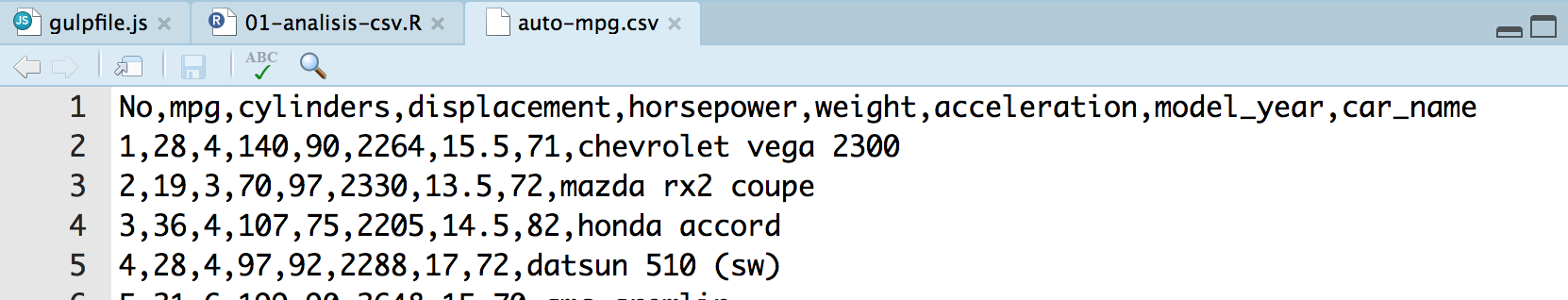


### El fichero CSV y la función read.csv

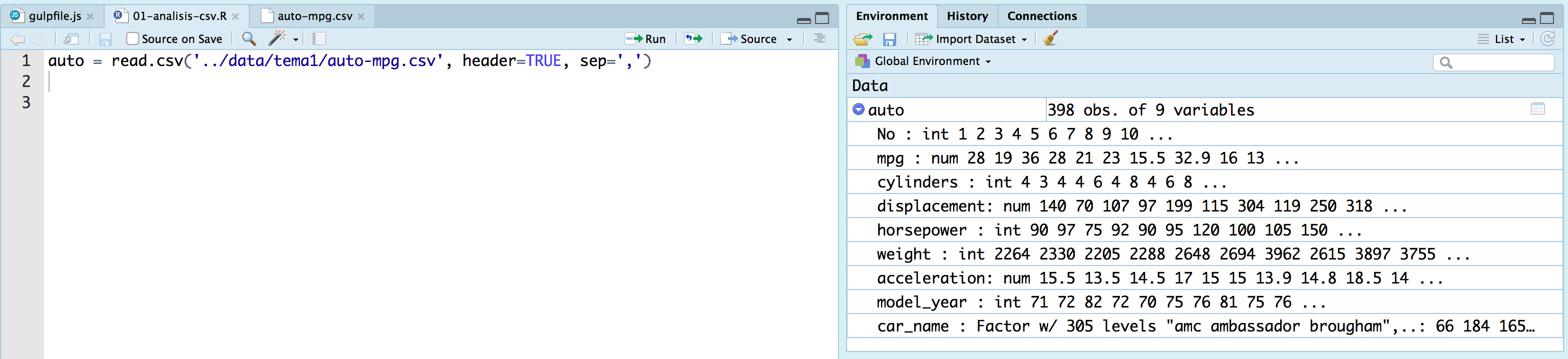
Vamos a trabajar con el fichero auto-mpg.csv de la carpeta “data” del “tema1”. En Mac, podemos pulsar la barra espaciadora sobre el nombre del fichero, para ver su contenido:

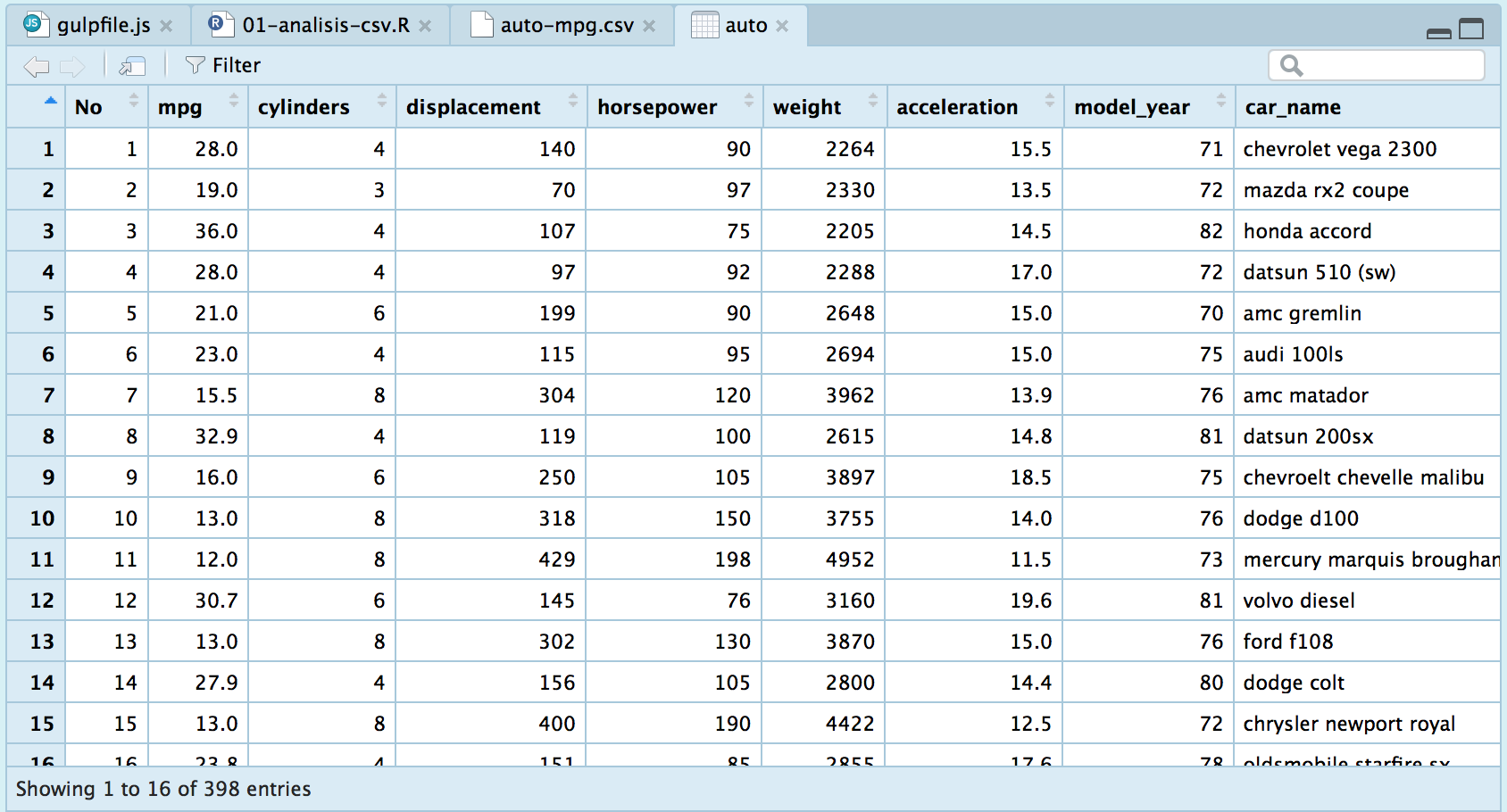


La representación que nos da del csv R-Studio no es muy bonita:



Pero podemos crear un script que lo lea, ejecutarlo y luego, en la pestaña “Environment”, podremos acceder a la variable donde lo hemos cargado, y pulsar sobre ella para visualizarlo:





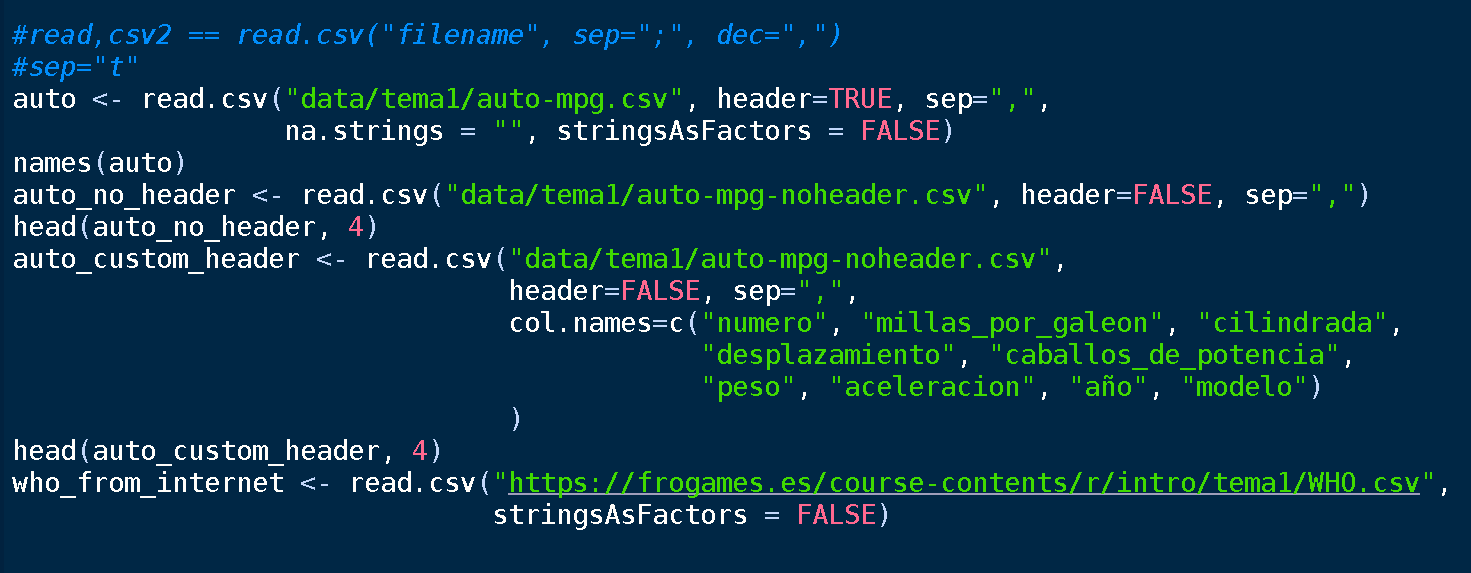
Con “head(dataset)” mostramos las primeras líneas de los valores cargados.

Con “names(dataset)” mostramos los nombres de las columnas.

Podemos crear nuestros propios nombres de columnas con “col.names”.

read.csv2 utiliza por defecto “;” como separador de campos, y “,” como separador decimal (dec). Si queremos usar el tabulador como separador de campos, haremos sep=”\t”.

auto-mpg.csv.R:



Los valores numéricos en blanco se tratan como “NA” (not available).

Los valores categóricos los interpretará en blanco, directamente. Pero podemos utilizar “na.strings”, asignando el valor que queramos.

Se recomienda usar “stringsAsFactors = FALSE” para que trate los strings como cadenas de caracteres.

### Leyendo datos de un XML

Necesitamos instalar el paquete “XML”.

Al convertir el xml en un dataframe, normalmente tendremos que aplicarle una transposición.

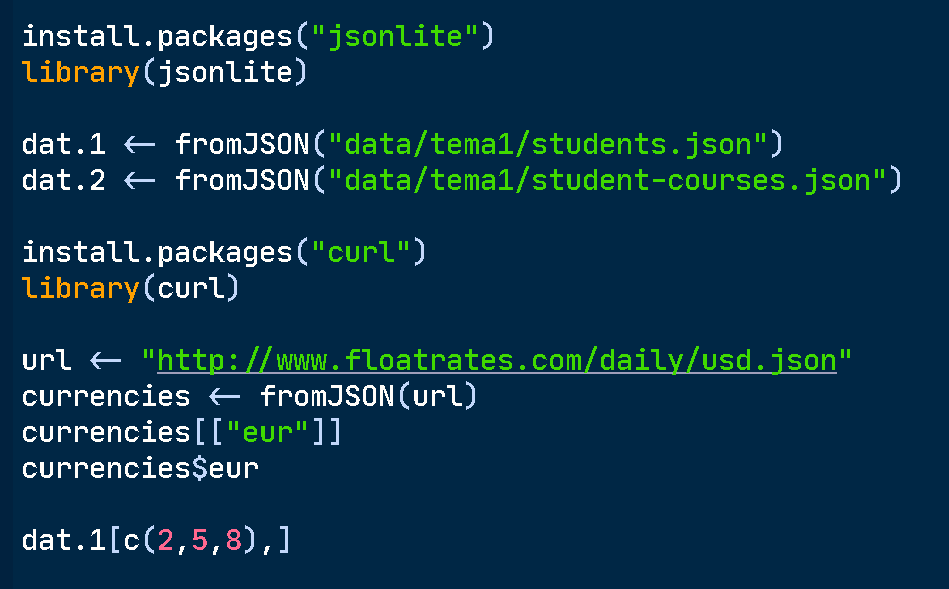
También podemos extraer las tablas de una página HTML.

02-analisis-xml.R:



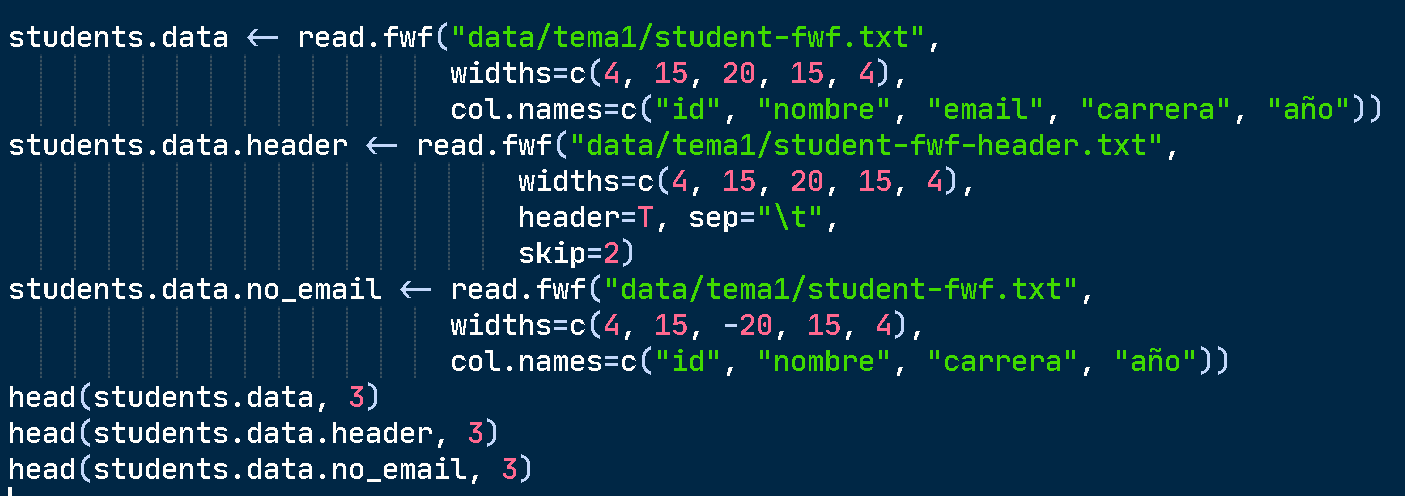
### Leyendo datos de un JSON

03-analisis-json.R:



### Ficheros de ancho fijo

04-analisis-fwf.R:



### Creando ficheros Rdata y rds

Existen varios dataframes en el core. Con la función “data()” podemos cargar el que deseemos.

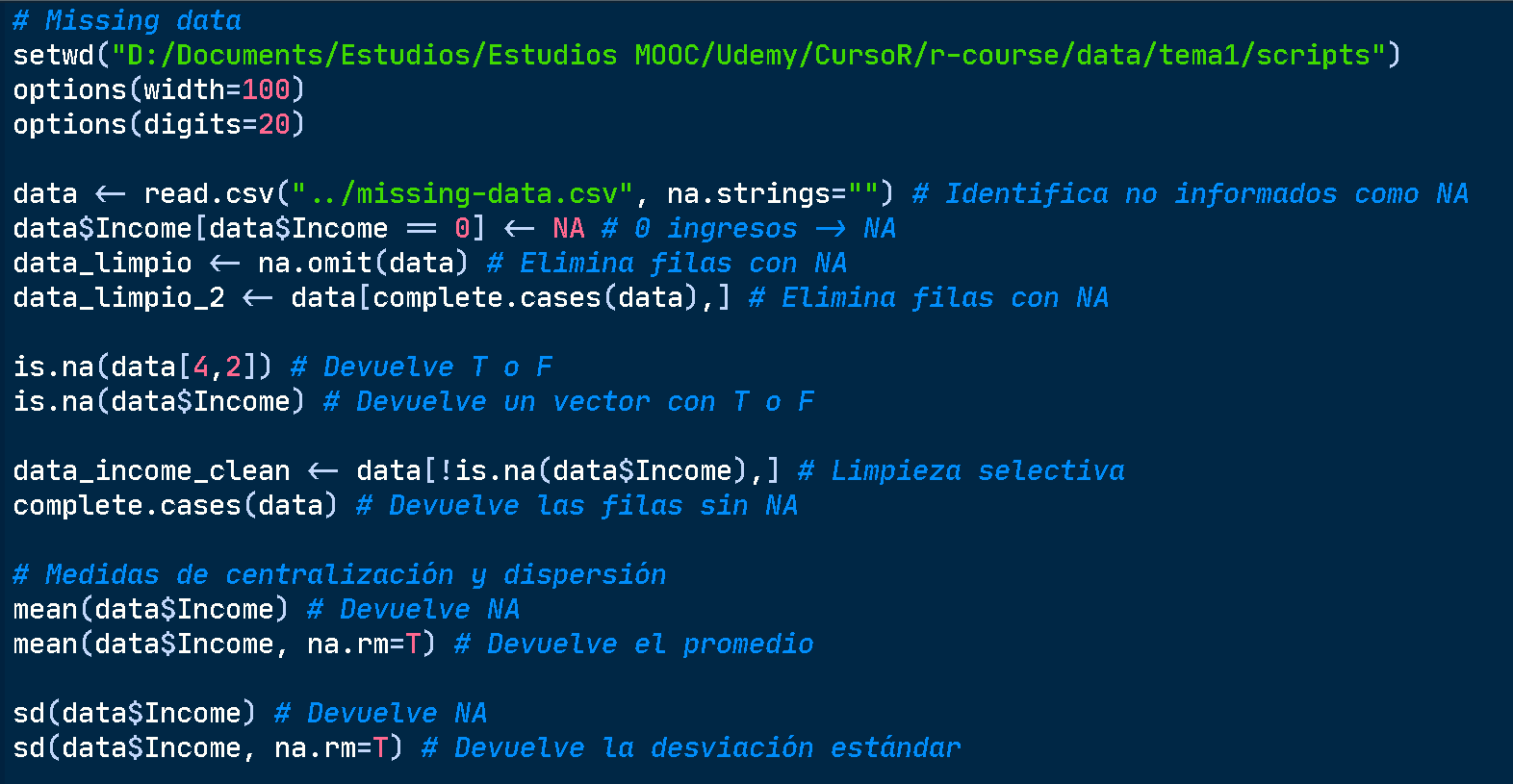
05-analisis-rdata.R:



### Eliminar datos sin valor con na.omit

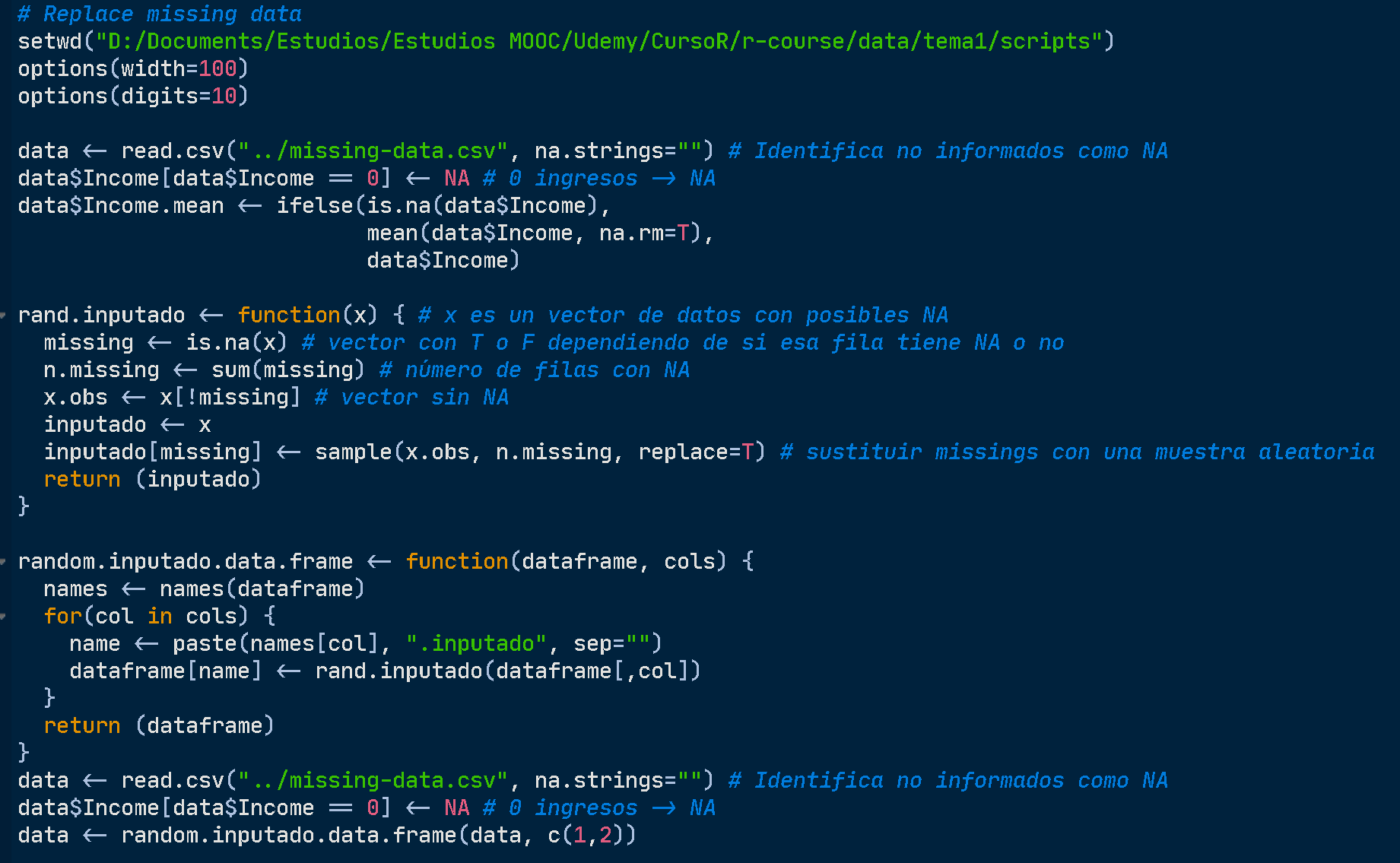
Podemos encontrar dataframes con columnas no informadas en algunas filas.

06-missing-data.R:



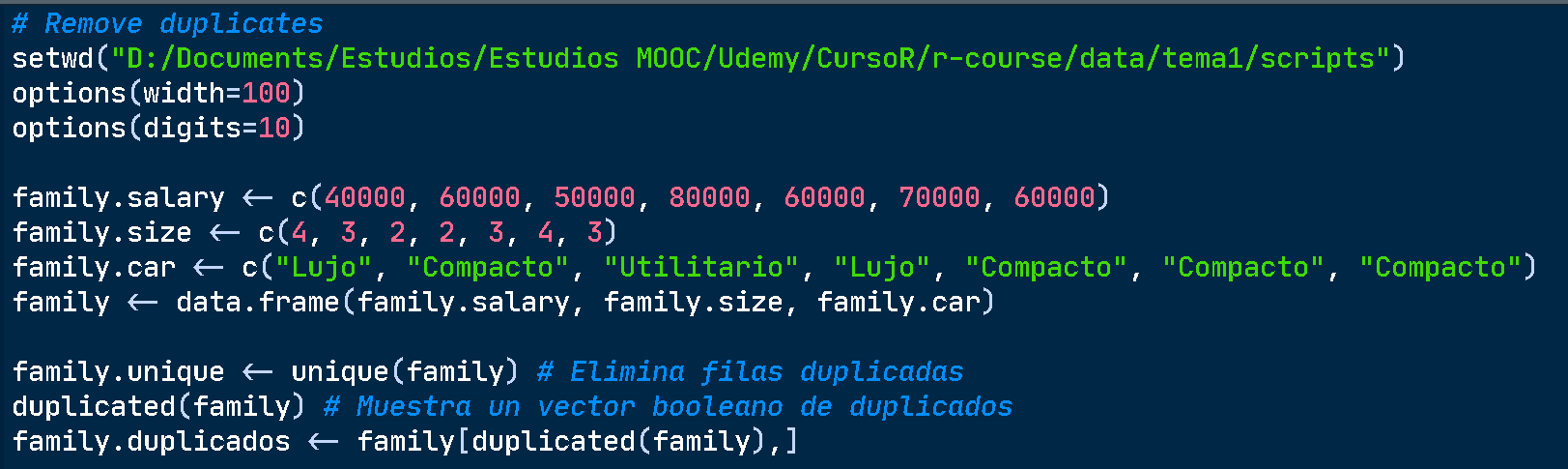
Reemplazar los NA con la media, o asignar muestras aleatorias.

07-replace-missing.R:



### Evitando duplicaciones de entradas

08-remove-duplicates.R:



### Reescalado lineal de datos

09-reescalado.R



### Normalización de datos

Normalizaremos en base a la distribución normal.

10-normalizacion.R: