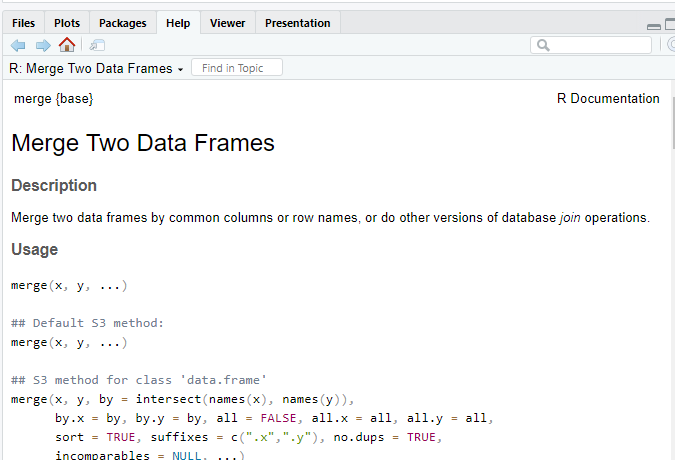
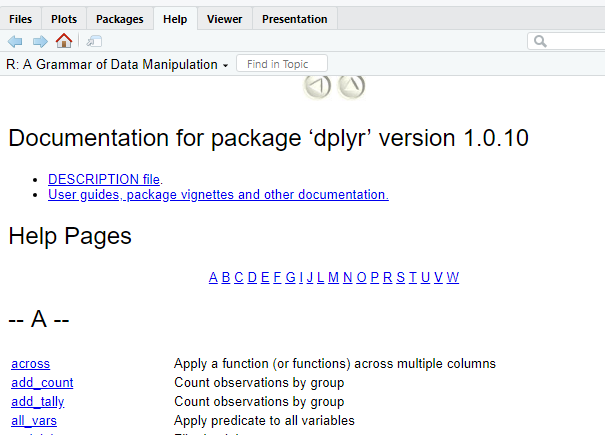


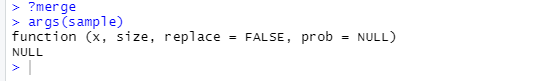
**? Nos permite solicitar ayuda sobre una función**



**Para solicitar ayuda sobre una librería utilizamos help().**



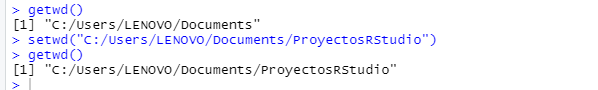
**Para conocer los argumentos de una función utilizamos args().**

****

**Obtener directorio de trabajo**

****

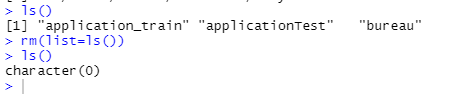
**Establecer directorio de trabajo**

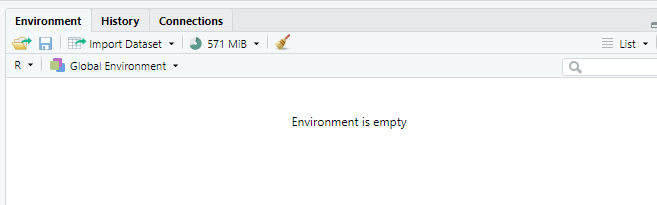
****

**Comprobar y crear directorio**

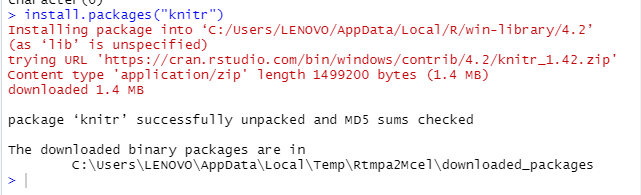
****

**Listar y limpiar nuestro espacio de trabajo**

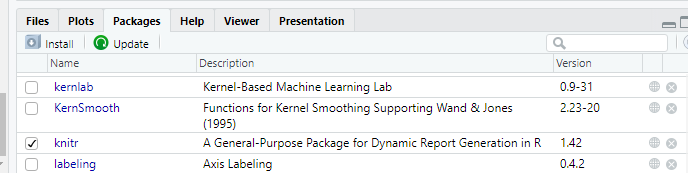
****

****

**Instalar y cargar paquetes de R**

****

**Vemos la librería cargada en:**

****

**Para utilizar la librería, la cargamos (habrá que hacerlo en cada sesión):**

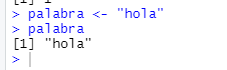
****

**Asignación + Impresión en pantalla**

****

**O Autoprinting**

****

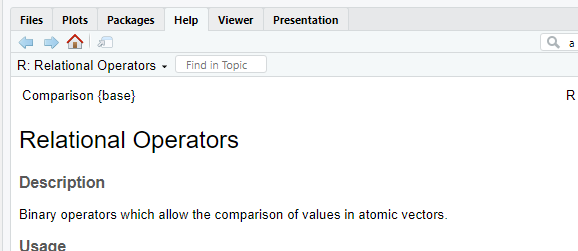
****

**Creando secuencia de enteros**

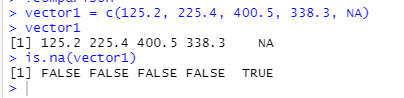
****

**Tests lógicos en R**

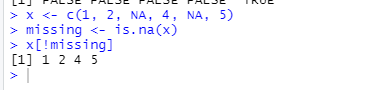
****

****

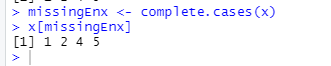
**Missing values**

****

**Eliminar NA**

****

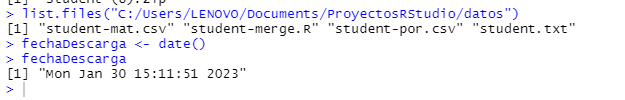
**Complete.cases**

****

**NaN: Missing values en operaciones matemáticas. Los valores NaN son a su vez NA (pero no al contrario)**

**Descarga student**

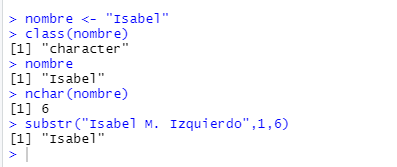
****

****

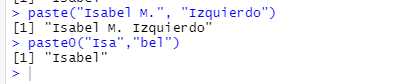
**Tipos de datos en R**

**class() y unclass()**

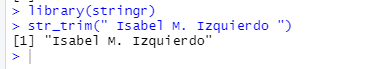
**Cadenas de caracteres, nchar(), substr()**

****

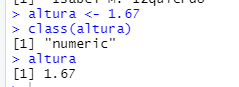
Cadenas de caracteres: paste(), paste0()

****

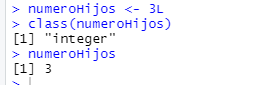
**Librería {stringr}**

****

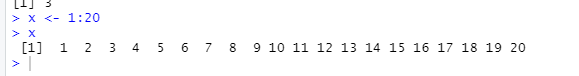
**Numéricos**

****

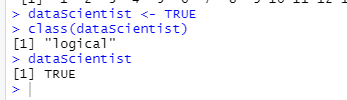
**Enteros**

****

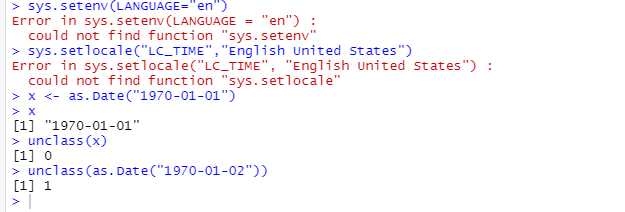
**Operador “:” para crear secuencia de enteros**

****

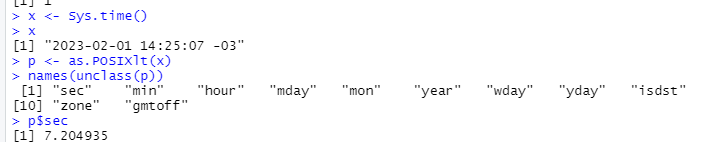
**Logicos**

****

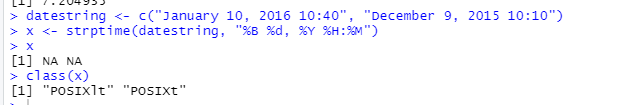
**Fechas y Horas. Las Fechas/Horas son almacenadas internamente como número de días/segundos desde 01/01/1970. clase Date**

****

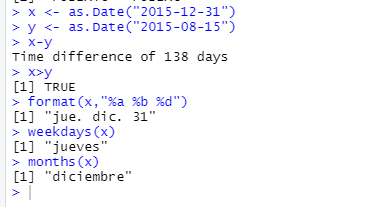
**Para Horas, las clases POSIXct y POSIXlt**

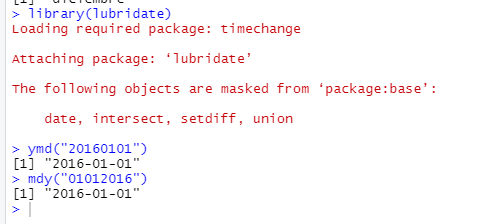
****

**función strptime() damos formato a las fechas (?strptime para más info**

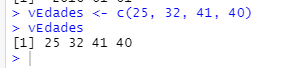
****

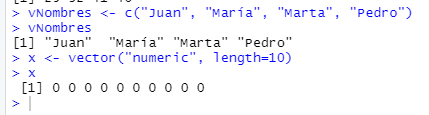
**Suma, resta, comparaciones y formato de fechas y horas**

****

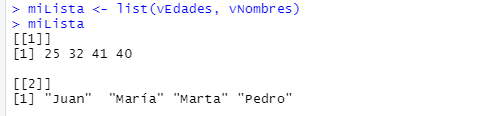
****

**Vectores. Conjunto de valores de la misma clase. Podemos crearlos con c() o con vector()**

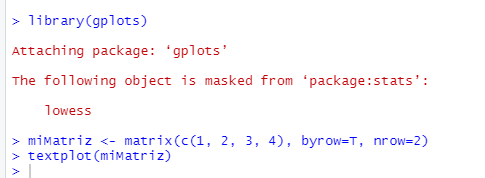
****

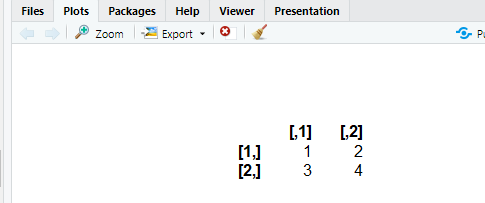
****

**Listas. Vector de valores que pueden ser de distintas clases**

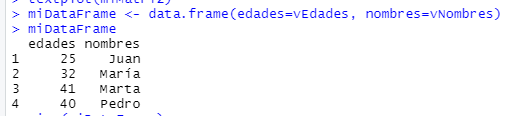
****

**Matrices. Vectores con múltiples dimesiones**

****

****

**DataFrames. Múltiples vectores de posibles diferentes clases, de la misma longitud**

****

**Librería {data.table}. Un objeto data.table acepta las mismas funciones que un data.frame, pero es mucho más rápido, especialmente en subsetting, merges y actualizaciones**

**En un DF(dataframe), se puede seleccionar columnas de este modo:**

nombre.df [nombre.df$campo.df == "valor, c("columna1", "columna2", "columna3")]

* 1. **Para hacer esto mismo en un DT(datatable)**:

nombre.dt [nombre.dt$campo.dt == "valor, c("columna1", "columna2", "columna3"), with = F]

**Pero es más habitual trabajar con list en los DT**:

nombre.dt[nombre.dt$campo.dt == "valor",list(columna1,columna2,columna3)]

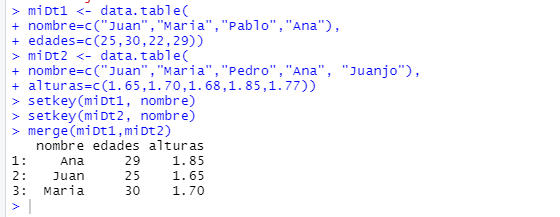
**La ventaja de la lista es que puedo cambiar los nombres de las columnas al mismo tiempo:**

nombre.dt[nombre.dt$campo.dt == "valor", list(new\_name\_col1=columna1,new\_name\_col2=columna2,new\_name\_col3=columna3)]

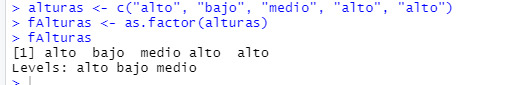
merge()

**library**(data.table)

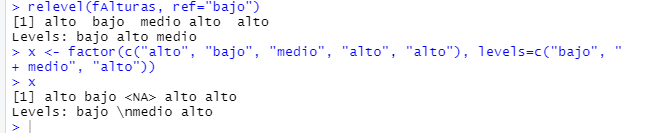
establecer la clave de nuestros objetos DT con la función setkey(),

****

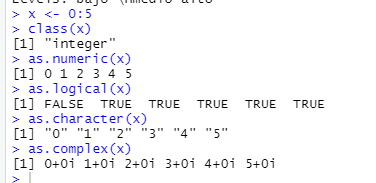
**Factores. Variables cualitativas**

****

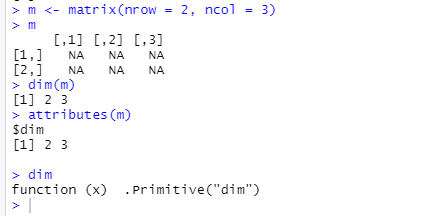
**Reordenar los niveles de un factor**

****

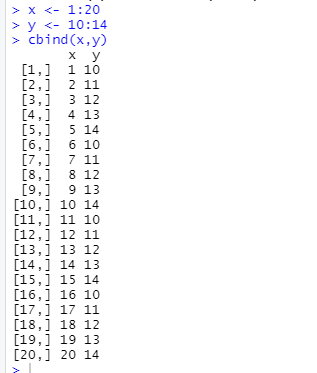
**as. Fuerzan a un objeto a convertirse de una clase, a otra clase**

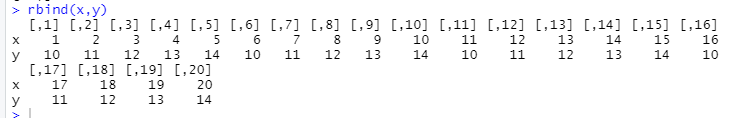
****

**dim() y attributes() El atributo "dimensión" es un vector de longitud dos (nrow, ncol)**

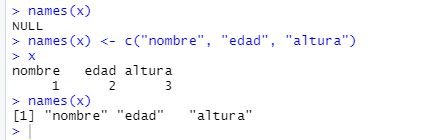
****

**cbind (Column binding) y rbind (row binding)**

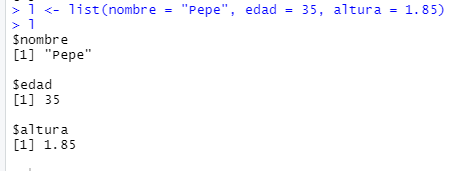
****

****

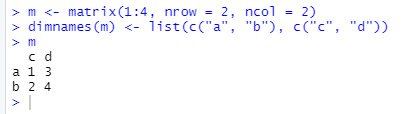
**names()**

****

**Nombres para listas**

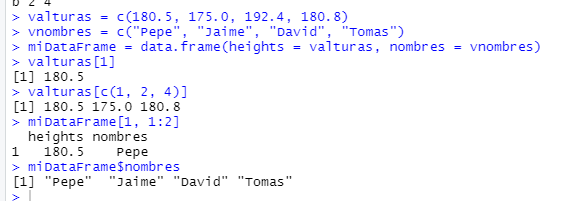
****

**Nombres para Matrices**

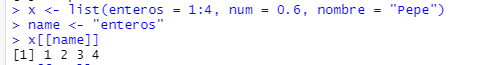
****

**Subsetting Se conoce como “subsetting” a las operaciones que sirven para extraer subconjuntos de objetos R.**

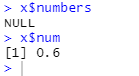
**Operador [**

****

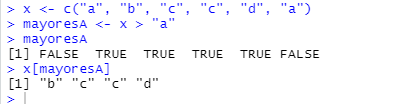
**Operador [[ Util en modelos, cuando queremos que una variable tome como nombre, cada nombre de las columnas de un data.frame, o de los elementos de una lista**

****

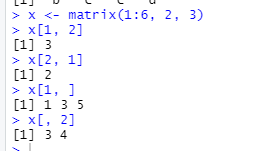
**Operador $**

****

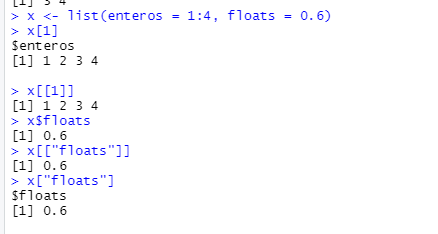
**Logical subsetting**

****

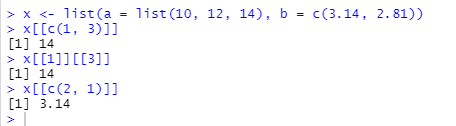
**Subsetting matrices**

****

**Subsetting listas**

****

**Subsetting elementos anidados de una lista**

****

**Distribuciones en R Las funciones de Distrubución de Probabilidad, tienen funciones asociadas en R, precedidas por:**

**• d: Densidad**

**• r: Random number generation**

**• p: Distribución acumulativa**

**• q: Quantile function**

**\*dbeta,dbinom, dcauchy, dchisq, dexp, df, dgamma, dgeom, dhyper, dlogis, dlnorm, dnbinom, dpois, dt, dunif, dweibull**

**• dnorm: Evaluar la Densidad de Probabilidad Normal )dada una media/desviación estñandar) en un punto o vector de puntos**

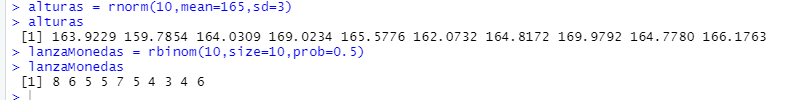
**• rbeta, rbinom, rcauchy, rchisq, rexp, rf, rgamma, rgeom, rhyper, rlogis, rlnorm, rnbinom, rt, runif, rweibull**

**• rpois: Generación de variables aleatorias Poisson, dado un ratio**

**• rnorm: Generación variables aleatorias Normales, dada una media y desviación estándar**

**pnorm**: Evalúa la Función de Distribución acumulativa para una Distribución Normal

**Ejemplos:**

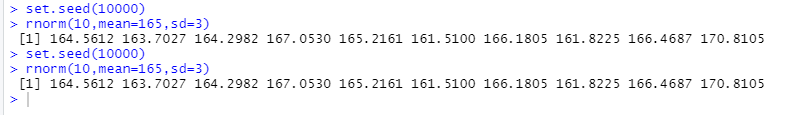
****

**Semillas en R Establecer una semilla (sed.seed()) asegura poder reproducir los resultados de procesos aleatorios en R**

****

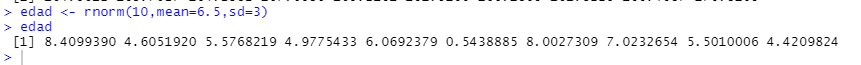
****

**Ejecutando 2 veces rnorm() con los mismos parámetros, pero sin haber establecido una semilla, nos ha dado resultados diferentes. En cambio, si establecemos una semilla, los resultados son idénticos**

****

**Muestras en R**

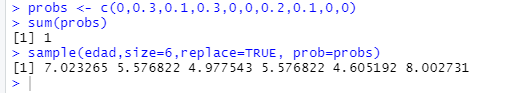
**Muestras aleatorias. Suponemos que de un objeto “edad” de 10 elementos, queremos generar una muestra aleatoria de 6 elementos.**

****

**Utilizamos sample() con reemplazo**

****

**Utilizamos sample() con reemplazo, fijando unas probabilidades**

****

**Utilizamos sample() sin reemplazo**

****

**Subsampling con porcentaje para train y test. Cuando para poder entrenar y después evaluar un modelo, dividimos nuestro dataset en train y test, prefijando por ejemplo una proporción 70/30, estamos empleando también el concepto de subsampling**

**tenemos en un objeto de nombre “dataset”, y queremos que el subconjunto de datos de entrenamiento sea el 70%, y el de test el 30%.**

**Análisis exploratorio con R**

**Proceso de Análisis de datos**

1.Análisis exploratorio

2.Entendimiento de los datos. Para ello realizaremos un Análisis descriptivo de los datos, estudiamos los “missing values”, los “outliers”8, etc. para cada variable; Visualizando posibles relaciones o patrones en los datos: Box-plot9 para variables numéricas, histogramas para variables categóricas, etc., matrices de dispersión, mosaicos

Recolección de los datos

Calidad de los datos

Estadística descriptiva

3.Entendimiento del Negocio

**Leer datos en R**

read.table()

Con este comando leemos un fichero de nuestro sistema de archivos local, en forma tabulada, almacenándolo en un data.frame, que tendrá como “observaciones” (filas) las líneas del fichero, y como “variables” (columnas), los campos del fichero.

Argumentos principales:

\* file: Nombre de fichero o conexiónç

\* header: Lógico, indica si el ficheo tiene o no cabecera

\* row\_names: Vector de nombres para las filas, o "NULL"

\* sep: String, separador de columnas

\* colClasses: Vector de caracteres, que indica la clase de cada columna

\* quote: quote="", para evitar errores si tenemos " en valores de los datos

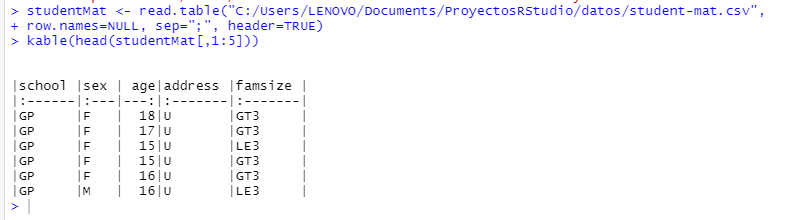
\* comment.char: String que indica el carácter usado para comentarios

\* na.strings: caracter que se debe interpretar como missing value

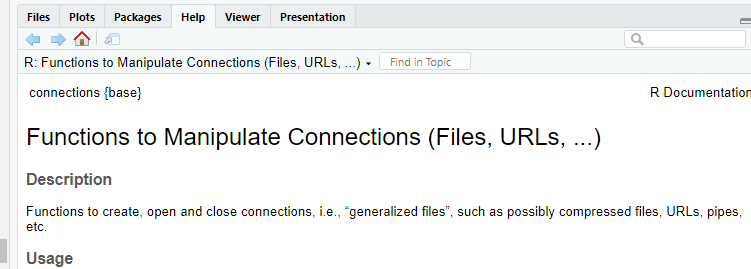
\* stringsAsFactors: Lógico, indica si las variables alfanuméricas deben considerarse factores

\* skip: Número de líneas a saltarse, contando desde el principio

\* nrows: Número de filas. Para ficheros grandes, hará que la lectura sea más rápida

****

****

****

Las más utilizadas:

\* file

\* url

\* gzfile

\* bzfil

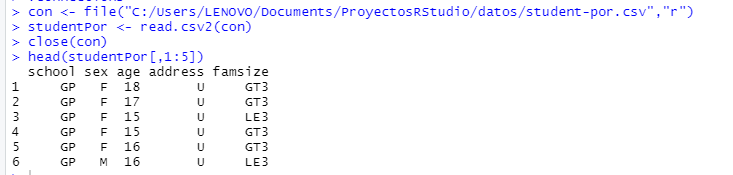
**read.csv() y read.csv2()**

Sirven para leer desde R ficheros CSV, aunque en realidad admiten más separadores que la coma:

• sep="," por defecto en read.csv

• sep=";" por defecto en read.csv2

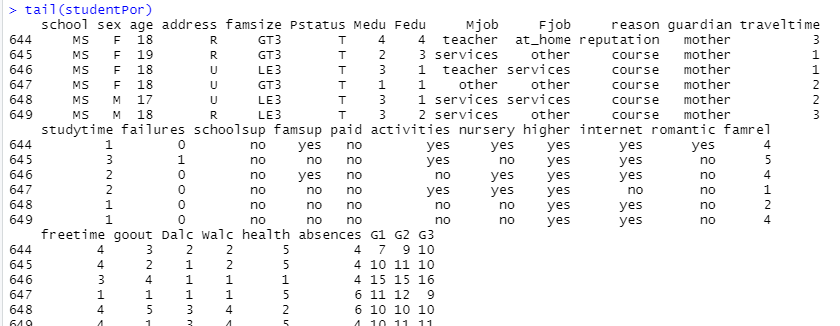
• header=TRUE por defecto en ambos

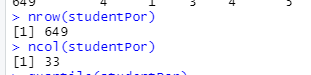
****

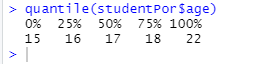
**Librería {RMySQL}**

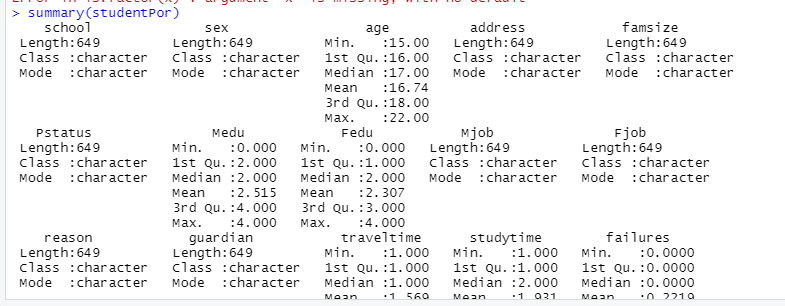
**Esta librería se utiliza para importar datos de MySQL.**

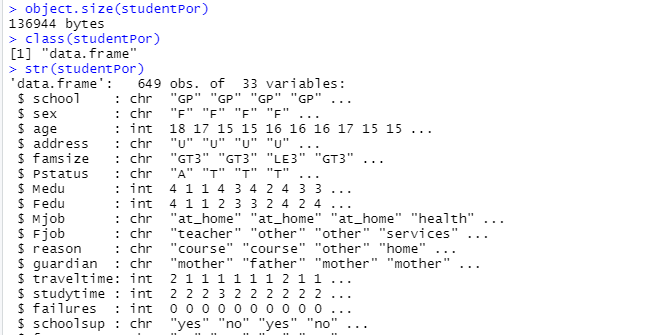
**Empezaremos nuestro análisis exploratorio con funciones que podemos clasificar como “summarizing”:**

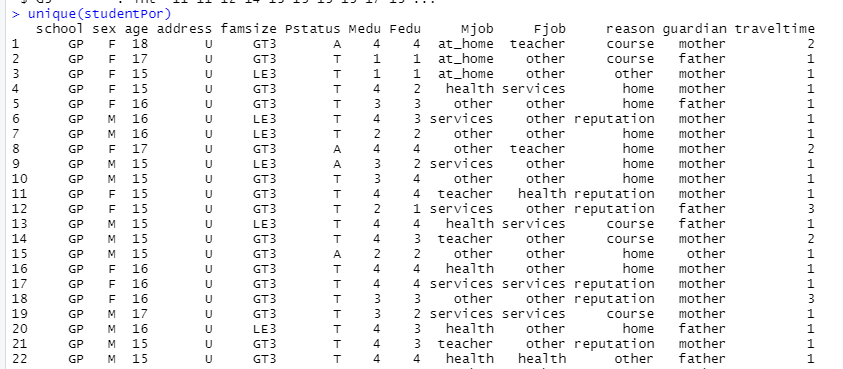
****

****

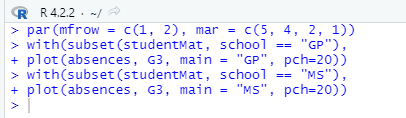
****

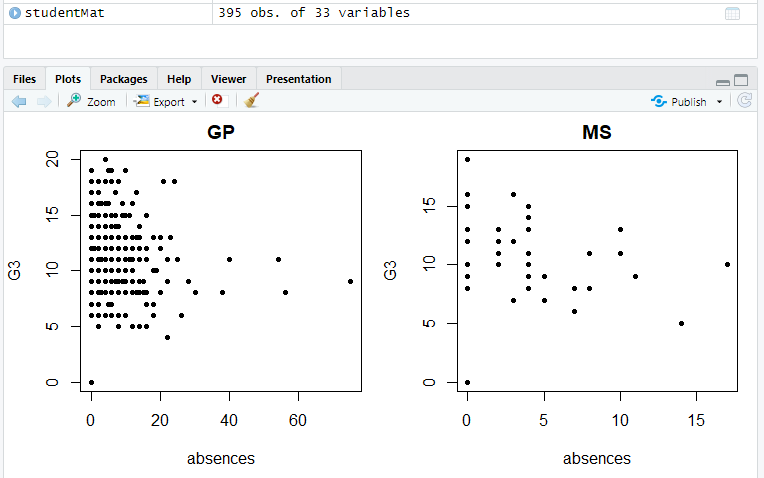
****

****

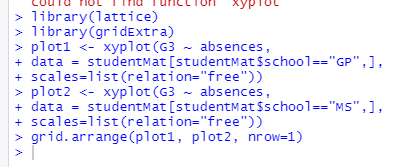
****

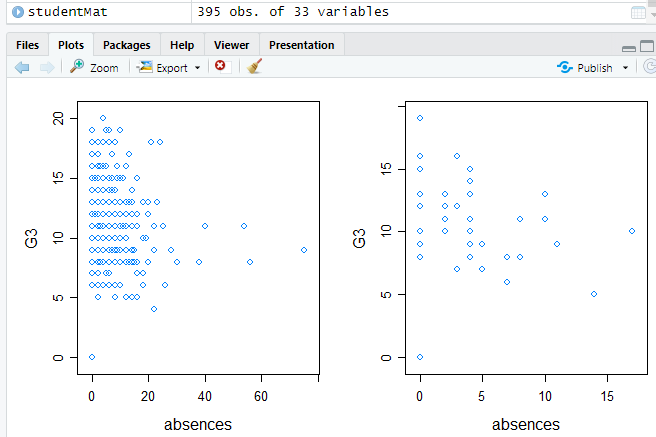
* 1. graphics >> lattice >> ggplot2. Un ejemplo de gráfico básico con {graphics}:

****

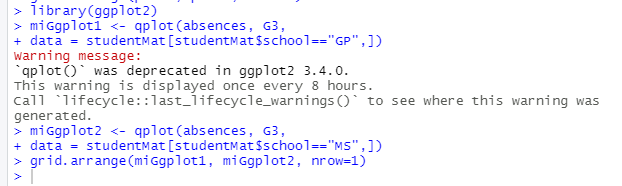
****

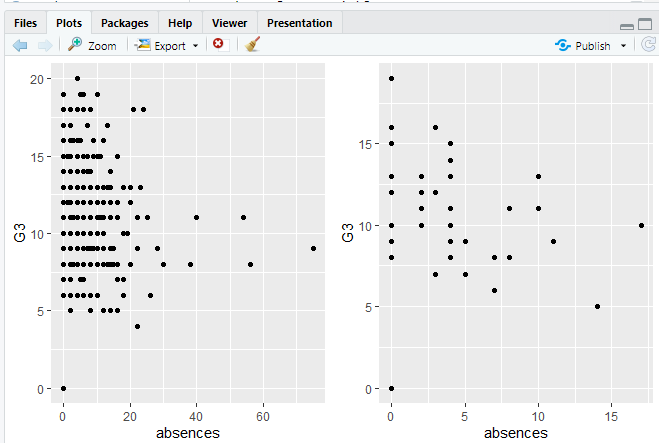
**El mismo ejemplo con {lattice}:**

****

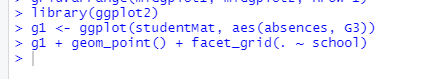
****

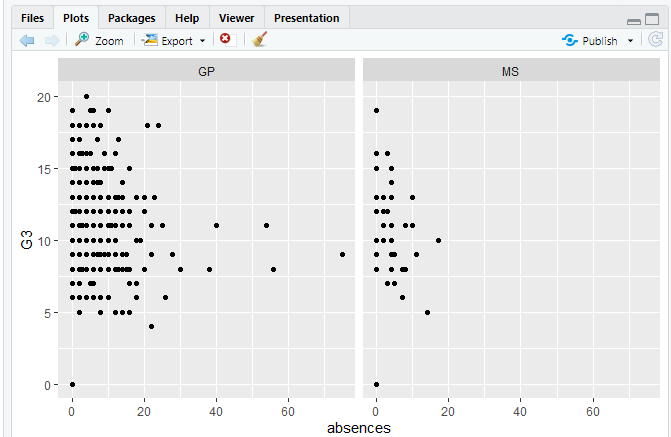
**El mismo ejemplo con qplot de {ggplot2}**

****

****

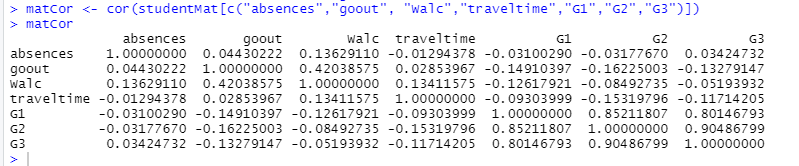
**El mismo ejemplo que en el apartado anterior, pero con ggplot de {ggplot2}**

****

****

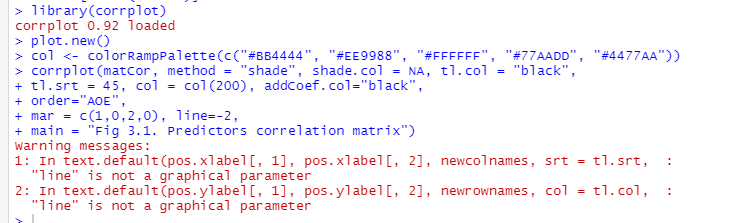
**Matrices de correlación. Para ver lo relacionadas que están unas variables con otras, y también con el target, podemos calcular y dibujar una matriz de correlación.**

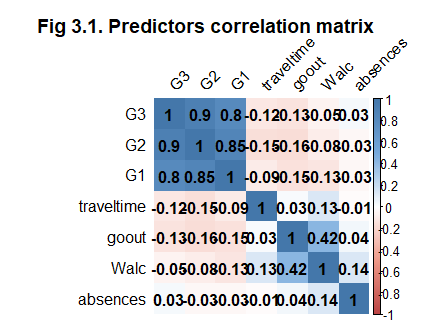
**Utilizaremos en R la función cor() que aplica a variables numéricas**

****

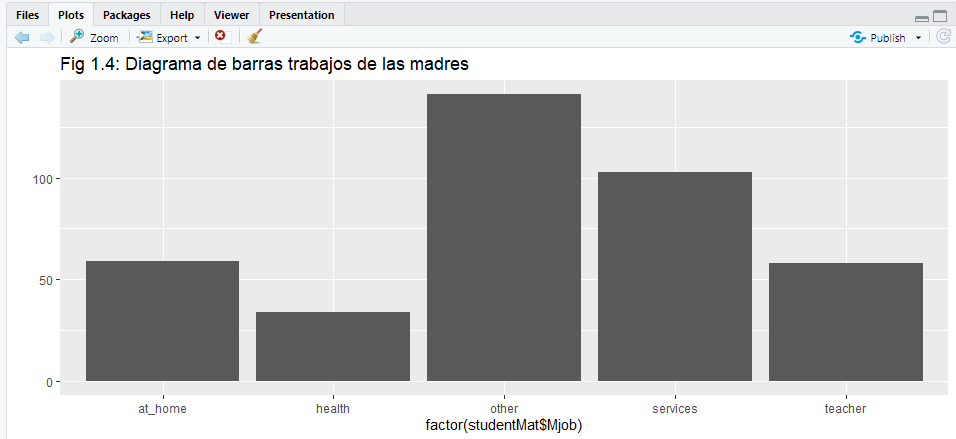
Si en el resultado de **cor()** hay NAs

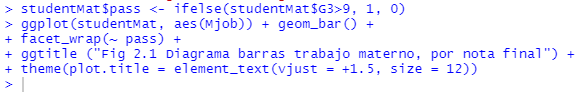
Y para dibujar matrices de correlación hay varios métodos. Recomendamos *corrplot* de la librería {corrplot}:

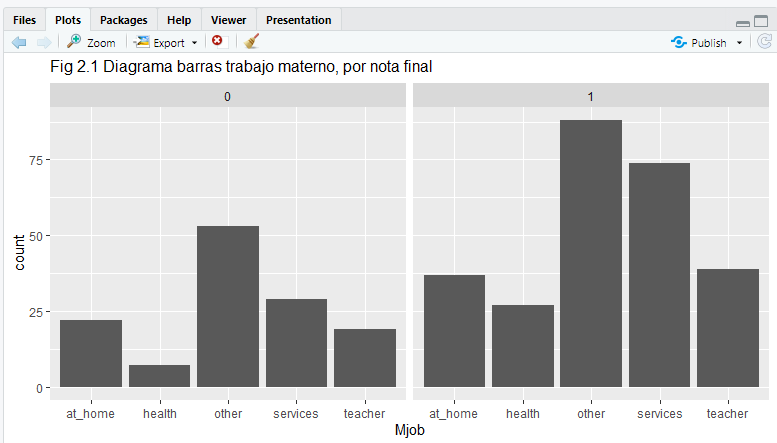
****

****

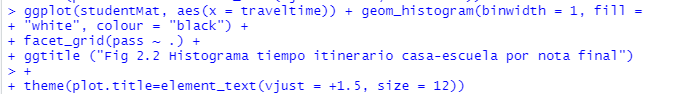
****

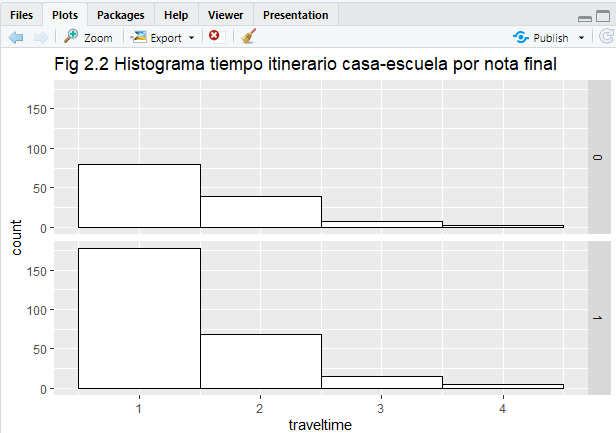
****

****

****

*# Para las variables continuas, podemos ver su distribución en relación al target, con histogramas. Por ejemplo*

****

****