

# Master en Big Data. Fundamentos matemáticos del análisis de datos.

## Sesión 2: Tipos de Variables y Análisis Exploratorio

Fernando San Segundo

Curso 2020-21. Última actualización: 2020-09-08



- 1 Trabajando con ficheros de datos.
- 2 Tipos de Variables.
- 3 Variables cuantitativas discretas.
- 4 Variables cuantitativas continuas,

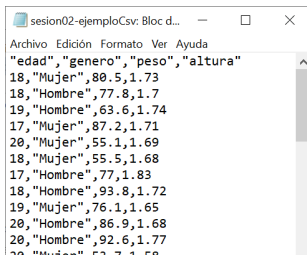
## Section 1

Trabajando con ficheros de datos.

- En la primera sesión hemos usado tablas de datos incorporadas en R (o en librerías). Pero para nuestro trabajo necesitaremos normalmente importar datos procedentes de fuentes externas. Hoy aprenderemos a usar datos almacenados en:
  - ▶ ficheros de texto
  - ▶ ficheros Excel
  - ▶ ficheros de otros programas estadísticos (SAS, SPSS, etc.)
  - ▶ ficheros RData propios de R Vamos a ver como leer estos ficheros para usar los datos en R y también veremos como guardar datos desde R en algunos de esos formatos.
- En otro momento del curso hablaremos de formas alternativas de acceder a datos no almacenados en ficheros (URLs, APIs, bases de datos tipo SQL, Web Scrapping, etc.)

# Ficheros de tipo csv

- El nombre csv proviene de *comma separated values*, valores separados por comas, aunque vamos a ver enseguida que no hay que tomarse el nombre al pie de la letra.
- Un fichero csv es un fichero de *texto plano* que contiene una tabla de datos. Cada fila del fichero contiene una fila de la tabla y, dentro de esa fila, los elementos correspondientes a cada columna de la tabla se separan mediante comas o espacios o tabuladores, etc. La siguiente figura muestra uno de esos ficheros abierto en el *Bloc de Notas* de Windows y la tabla correspondiente (se muestran las primeras filas).



```
sesion02-ejemploCsv: Bloc d...
Archivo Edición Formato Ver Ayuda
"edad","genero","peso","altura"
18,"Mujer",80.5,1.73
18,"Hombre",77.8,1.7
19,"Hombre",63.6,1.74
17,"Mujer",87.2,1.71
20,"Mujer",55.1,1.69
18,"Mujer",55.5,1.68
17,"Hombre",77,1.83
18,"Hombre",93.8,1.72
19,"Mujer",76.1,1.65
20,"Hombre",86.9,1.68
20,"Hombre",92.6,1.77
20,"Mujer",53.7,1.59
```

Fichero csv



edad	genero	peso	altura
18	Mujer	80.5	1.73
18	Hombre	77.8	1.70
19	Hombre	63.6	1.74
17	Mujer	87.2	1.71
20	Mujer	55.1	1.69
18	Mujer	55.5	1.68
17	Hombre	77.0	1.83
18	Hombre	93.8	1.72
19	Mujer	76.1	1.65
20	Hombre	86.9	1.68
20	Hombre	92.6	1.77

la correspondiente tabla

## Ficheros csv con R.

- Vamos a empezar descargando uno de estos ficheros, llamado `movies.csv` que contiene datos sobre las [películas más taquilleras entre 2007 y 2011](#).
- Recuerda que debes indicarle a RStudio el *Directorio de Trabajo* y que el fichero descargado debe estar almacenado en la subcarpeta *datos* de ese directorio de trabajo.
- Empieza abriendo ese fichero en un editor de texto (tipo *Bloc de Notas*) para hacer una exploración preliminar.
- Para abrir ese fichero con R vamos a empezar usando una función del sistema básico de R:

```
movies = read.csv(file = "../datos/movies.csv", header = TRUE)
```

- Enseguida repetiremos esta operación con funciones del tidyverse. Pero hemos usado `read.table` para insistir en que el resultado de este comando es un `data.frame` de R. Las opciones de la función son:
  - ▶ *file*: el nombre y directorio del fichero relativo (a la carpeta de trabajo).
  - ▶ *header*: que puede ser `TRUE` o `FALSE`, para indicar si la primera fila del csv contiene los nombres de las variables.

Veremos más adelante otras opciones importantes de esta función y funciones similares.

## Repaso de operaciones con data.frames.

- Recuerda que puedes seleccionar por filas con instrucciones como:

```
movies[7, ]
```

```
##      Film      Genre Lead.Studio Audience.score..  
## 7 WALL-E Animation      Disney              89  
## Profitability Rotten.Tomatoes.. Worldwide.Gross Year  
## 7      2.896019              96      $521.28 2008
```

- Y por columnas de forma similar o también por nombre de variable usando \$:

```
tail(movies$Year, 20) # se muestran las 20 últimas
```

```
## [1] 2011 2009 2010 2008 2009 2007 2010 2011 2011 2009 2008  
## [12] 2008 2007 2010 2011 2007 2009 2011 2008 2009
```

- Recuerda asimismo que puedes seleccionar por condiciones. Por ejemplo para ver el género de las películas de 2010 con:

```
movies$Genre[movies$Year == 2010]
```

También sabemos usar dplyr para seleccionar:

```
library(tidyverse)
```

```
movies %>%
```

```
  filter(Year == 2010) %>%
```

```
  select(Genre) %>%
```

```
  .[1:20, ]      # ¿Qué hace esta última operación?
```

```
## [1] "Comedy"      "Comedy"      "Comedy"      "Comedy"  
## [5] "Comedy"      "Animation"   "Comedy"      "Comedy"  
## [9] "Comedy"      "Drama"       "Comedy"      "Comedy"  
## [13] "Comedy"      "Comedy"      "Comedy"      "Action"  
## [17] "Comedy"      "Comedy"      "Comedy"      "Drama"
```

- **Ejercicio:** ¿Cuál es la película más taquillera? ¿Cuál es el género de esa película?

## Usando readr para leer y escribir ficheros csv.

- La librería `readr`, que forma parte del `tidyverse`, incluye la función `read_csv`, que es muy fácil de usar y más rápida que `read.table` para ficheros grandes. Explora esta tabla como hemos hecho con la primera versión.

```
movies2 = read_csv("../datos/movies.csv")
```

- Ejercicio:** Ejecuta `str(movies2)`. ¿Qué tipo de objeto usa el `tidyverse` para almacenar tablas en lugar del `data.frame` del R básico?
- También puedes usar `readr` para crear ficheros csv a partir de una tabla (por ejemplo un `data.frame`) en R. El siguiente código genera primero una tabla con tres variables A, B y C y a continuación guarda esa tabla a un fichero csv. Asegúrate de abrir el fichero resultante en un editor de texto para ver el resultado.

```
set.seed(2019)
datos =
  data.frame(A = sample(1:100, 10), B = sample(LETTERS, 10), C = rnorm(10))
head(datos, 2)
write_csv(datos, path = "../datos/sesion02-guardarCsv.csv")
```

```
##      A B      C
## 1 25 N -0.1114757
## 2 42 Q -2.3553230
```

- Las funciones `write.table` y `write.csv` de R funcionan de manera parecida. Veremos algún ejemplo de uso más adelante.



- Las hojas de cálculo y en particular Excel son una herramienta muy utilizada. Por eso no es infrecuente encontrarse con ficheros de datos que se han almacenado en alguno de los formatos propios de diferentes versiones de Excel.
- Descarga para usar como ejemplo [este fichero](#) en formato xls, que contiene datos sobre accidentes ferroviarios ocurridos en 2010 en los Estados Unidos. Puedes encontrar más detalles sobre el fichero en [este documento auxiliar](#).
- Para leer esos datos vamos a usar la librería `readxl` (del tidyverse) de esta forma

```
library(readxl)
accidentes = read_excel("../datos/train_acc_2010.xls")
```

Puedes leer más sobre `readxl` [aquí](#).

- **Ejercicio:** exporta esta tabla de R a un fichero en formato csv llamado `accidentes.csv`.

## Ficheros de otros programas estadísticos.

- Aunque existen muchos otros programas estadísticos, aquí solo vamos a ver como se usa la librería `haven` del `tidyverse` para importar en R ficheros de datos de SPSS, Stata y SAS. Si necesitas importar datos almacenados en un formato propio de otro programa lo mejor es buscar en Internet algo como *import from ... to R*. Recuerda empezar cargando [otra librería del tidyverse](#).

```
library(haven)
```

- También recomendamos consultar el libro (Boehmke 2016), especialmente la Parte IV para completar la introducción a la importación de datos que hemos visto aquí.

## Fichero SAV de SPSS

- Descarga el fichero `CH10_Planet_distances_and_y.SAV` a la carpeta datos desde [este enlace](#) y ábrelo con:

```
library(haven)
planetas = read_spss("../datos/CH10_Planet_distances_and_y.SAV")
head(planetas, 3) # Veamos las tres primeras filas.
```

```
## # A tibble: 3 x 4
##   Planet   PositionNumber Distancefromsunmillionmiles Lengthofyearearthyears
##   <chr>         <dbl>             <dbl>                 <dbl>
## 1 Mercury         1              36                 0.24
## 2 Venus           2              67                 0.61
## 3 Earth          3              93                 1
```

## Ficheros sas7bdat de SAS y dta de Stata

- Usa [este enlace](#) para descargar el fichero transport.sas7bdat a la carpeta datos y ábrelo con:

```
transport = read_sas("../datos/transport.sas7bdat")
head(transport, 3)
```

```
## # A tibble: 3 x 4
##   AUTOTIME BUSTIME DTIME  AUTO
##   <dbl>    <dbl> <dbl> <dbl>
## 1    52.9      4.40 -48.5     0
## 2     4.10     28.5  24.4     0
## 3     4.10     86.9  82.8     1
```

- Procede de forma análoga con [este fichero](#) llamado auto2.dta en formato de Stata

```
auto2 = read_dta("../datos/auto2.dta")
head(auto2, 3)
```

```
## # A tibble: 3 x 13
##   make      price    mpg rep78 headroom trunk  weight  length  turn displacement gear_ratio    foreign weightsq
##   <chr>    <dbl> <dbl> <dbl>    <dbl> <dbl>   <dbl>   <dbl>   <dbl>    <dbl>    <dbl> <dbl> <dbl>
## 1 AMC Concord  4099    22     3      2.5    11   2930   186    40      121     3.58 0 [Domestic]  8584900
## 2 AMC Pacer    4749    17     3      3      11   3350   173    40      258     2.53 0 [Domestic] 11222500
## 3 AMC Spirit  3799    22    NA     3      12   2640   168    35      121     3.08 0 [Domestic]  6969600
```

- Guarda ambos ficheros en formato csv en la carpeta datos, los usaremos después como ejemplos.

- Como ves todos los casos se gestionan de forma muy parecida. En ejemplos posteriores veremos otras situaciones; como tratar por ejemplo con ficheros comprimidos tipo zip.

## Ficheros RData.

- R también posee su propio formato de almacenamiento de objetos, usando ficheros tipo RData. Estos ficheros pueden contener varias tablas de datos, variables y otros objetos de R. Por ejemplo podemos guardar la tabla de accidentes ferroviarios y la de planetas que hemos usado antes mediante:

```
# Ficheros RData.  
save("accidentes", "planetas", file = "../datos/accidentes_planetas.RData")
```

- Fíjate en que hemos añadido la extensión RData manualmente, porque R no lo hace por defecto. Ahora vamos a eliminar por ejemplo la tabla planetas con:

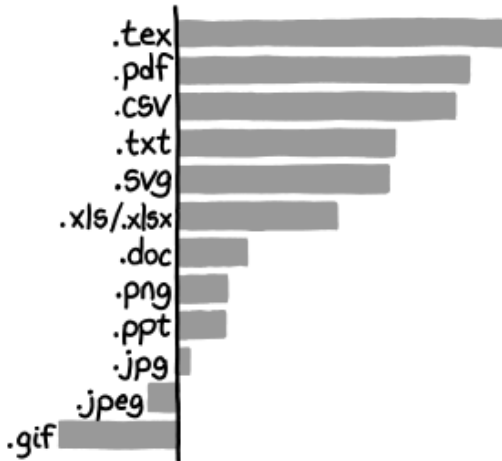
```
rm(planetas)
```

Comprueba mirando el panel de entorno que en efecto la tabla ha desaparecido y si intentas usarla R lanzará un mensaje de error. Y ahora para recuperar esos datos usa:

```
load(file = "../datos/accidentes_planetas.RData")  
head(planetas, 3)
```

```
## # A tibble: 3 x 4  
##   Planet PositionNumber Distancefromsunmillionmiles Lengthofyearearthyears  
##   <chr>          <dbl>                <dbl>                <dbl>  
## 1 Mercury          1                  36                  0.24  
## 2 Venus            2                  67                  0.61  
## 3 Earth            3                  93                   1
```

## TRUSTWORTHINESS OF INFORMATION BY FILE EXTENSION



## Section 2

### Tipos de Variables.

- Los tablas de datos que hemos leído en los ficheros de la sección previa contienen variables de distintos tipos: números enteros, con decimales, fechas, variables binarias de tipo sí/no, hombre/mujer, ubicaciones, etc. Existen muchos tipos de datos distintos, que permiten distintas operaciones con ellos.
- En las próximas secciones vamos a conocer las categorías básicas de datos y las formas más adecuadas de describirlos. Como ejemplos iniciales vamos a usar la tabla `mpg` contenida en la librería `tidyverse` y también una tabla con datos relativos a un estudio sobre enfermedades coronarias llevado a cabo en Framingham (UK). Puedes descargar el fichero csv desde [este enlace](#) y leer más detalles sobre el estudio [aquí](#).
- **Ejercicio:** lee el fichero a una tabla de R llamada `fhs` (de Framingham Heart Study). Explora esa tabla con las funciones `str` y `glimpse`. Piensa en qué tipo de información contiene cada variable de la tabla. Lee también la documentación sobre `mpg` en [este enlace](#).

	male	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol	sysBP	diaBP	BMI	heartRate	glucose	TenYearCHC
1	1	39	4	0	0	0	0	0	0	195	106.0	70.0	26.97	80	77	0
2	0	46	2	0	0	0	0	0	0	250	121.0	81.0	28.73	95	76	0
3	1	48	1	1	20	0	0	0	0	245	127.5	80.0	25.34	75	70	0
4	0	61	3	1	30	0	0	1	0	225	150.0	95.0	28.58	65	103	1
5	0	46	3	1	23	0	0	0	0	285	130.0	84.0	23.10	85	85	0
6	0	43	2	0	0	0	0	1	0	228	180.0	110.0	30.30	77	99	0
7	0	63	1	0	0	0	0	0	0	205	138.0	71.0	33.11	60	85	1
8	0	45	2	1	20	0	0	0	0	313	100.0	71.0	21.68	79	78	0
9	1	52	1	0	0	0	0	1	0	260	141.5	89.0	26.36	76	79	0
10	1	43	1	1	30	0	0	1	0	225	162.0	107.0	23.61	93	88	0
11	0	50	1	0	0	0	0	0	0	254	133.0	76.0	22.91	75	76	0
12	0	43	2	0	0	0	0	0	0	247	131.0	88.0	27.64	72	61	0
13	1	46	1	1	15	0	0	1	0	294	142.0	94.0	26.31	98	64	0

- Los datos que han ido apareciendo en nuestros ejemplos se pueden clasificar en:
  - ▶ **Datos Cuantitativos (Numéricos):** que a su vez se dividen en **discretos** y **continuos**.
  - ▶ **Datos Cualitativos (Factores):** que pueden ser o no ordenados.
- Esta es la clasificación tradicional en muchos cursos de introducción a la Estadística y enseguida vamos a ver ejemplos para entender la diferencia entre estos tipos de datos, Pero queremos subrayar que existen muchos tipos de datos estructurados de uso frecuente que superan esta clasificación tradicional (fechas, imágenes, ficheros de audio o vídeo).
- Primero vamos a aprender a analizar variables individuales, por separado, antes de preguntarnos por las relaciones entre ellas.



- Una *variable cuantitativa* (discreta o continua) es una variable que toma valores numéricos que *además* se han medido en alguna escala que permite interpretarlos y hacer operaciones aritméticas (sumas, productos, etc) con ellos.
- Una variable cuantitativa es *discreta* si se mide en una escala de unidades enteras (paso a paso, los valores se miden *contando*). Y la variable *continua* si la escala de medida se puede dividir arbitrariamente (se usan valores decimales). Pero como veremos en ejemplos, la división discreto/continuo también es sutil y se refiere en realidad a la forma en la que *usamos* la variable.
- Podría pensarse entonces que las variables cuantitativas son las variables numéricas y las cualitativas las no numéricas. La diferencia es, en realidad, un poco más sutil. Una variable es *cualitativa (nominal)* cuando *solo* se utiliza para establecer categorías, para *clasificar*. Podemos *representar* los valores de una de estas variables con números, pero el valor numérico concreto es arbitrario, es una *etiqueta*. Las variables cualitativas nominales también se denominan *factores*
- **Ejercicio:** Examina las variables `cty`, `disp`, `class` y `cyl` de la tabla `mpg`. ¿De qué tipo crees que es cada variable?

## Section 3

Variables cuantitativas discretas.

## Tablas de frecuencia absolutas y relativas.

- La variable `cty` de `mpg` el número de millas por galón que el coche recorre en ciclo urbano. Fíjate en que los valores son un número entero de millas. En principio no hay nada que impida dar esos valores con decimales. Pero no es eso lo que se *ha decidido* hacer aquí, sino que se trata como una variable discreta.
- El primer paso con una variable discreta como esta es obtener una tabla de frecuencias (absolutas), que nos dirá qué valores toma la variable y cuántas veces toma cada valor. Usando `table`

```
table(mpg$cty)
```

```
9 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 28 29 33 35  
5 20  8 21 19 24 19 16 26 20 11 23  4  3  5  2  3  2  1  1  1
```

- También se puede usar la función `count` de `dplyr` así (se omite el resultado):

```
mpg %>%  
  count(cty)
```

## Tabla de frecuencias relativas.

- A menudo, y especialmente cuando se usan para comparaciones, nos interesan más saber la fracción del total que corresponde a cada uno de los valores de una variable discreta. Cuando esas fracciones se expresan como *tanto por uno* obtenemos las frecuencias relativas, que es fácil convertir en porcentajes.
- Para obtener una tabla de frecuencias relativas usando R básico hacemos (hemos usado la función `signif` para controlar el número de cifras significativas y mejorar la presentación):

```
signif(prop.table(table(mpg$cty)), 2)
```

```
##
##      9      11      12      13      14      15      16      17      18      19
## 0.0210 0.0850 0.0340 0.0900 0.0810 0.1000 0.0810 0.0680 0.1100 0.0850
##     20     21     22     23     24     25     26     28     29     33
## 0.0470 0.0980 0.0170 0.0130 0.0210 0.0085 0.0130 0.0085 0.0043 0.0043
##      35
## 0.0043
```

- Se puede usar `dplyr` pero en este caso es más complicado que con el R básico.

```
mpg %>%
  count(cty) %>%
  mutate(cty, relFreq = prop.table(n), n=NULL)
# NULL aquí elimina la columna n
```

## Propiedades de las frecuencias relativas.

- Las frecuencias relativas suman siempre 1,

```
sum(prop.table(table(mpg$cty)))
```

```
## [1] 1
```

- Además las frecuencias relativas están relacionadas con la idea de *probabilidad empírica*. Es decir, si elegimos aleatoriamente un valor de la variable `cty` y repetimos esa elección muchas veces, la probabilidad de cada uno de los distintos valores es la frecuencia relativa que hemos calculado.

## Frecuencias acumuladas.

- Las *frecuencias acumuladas* se usan con variables discretas para responder a la pregunta “¿cuántos valores hay que sean menores o iguales que ...?” En R se obtienen con:

```
cumsum(table(mpg$cty))
```

```
##    9  11  12  13  14  15  16  17  18  19  20  21  22  23  24
##    5  25  33  54  73  97 116 132 158 178 189 212 216 219 224
##   25  26  28  29  33  35
##  226 229 231 232 233 234
```

que nos dice, por ejemplo, que en la tabla hay 116 valores menores o iguales que 16.

## Section 4

Variables cuantitativas continuas,

## Discreto vs continuo.

- Las tablas de frecuencias por valores no son útiles cuando hay muchos valores distintos. La tabla de frecuencias de `cty` ya era un poco excesiva. Pero si tratamos de calcular una tabla de frecuencia para la variable `age` de la tabla `fhs`

```
table(fhs$totChol)
```

puedes comprobar que la tabla que se obtiene no es una representación útil de la información.

- En muchos ejemplos como este las diferencias entre valores consecutivos no son relevantes. Las preguntas relevantes pasan a ser las que se refieren a intervalos de valores. Para agrupar los valores en intervalos en R podemos usar la función `cut`.

```
cholLevels = cut(fhs$totChol, breaks = 10)  
head(cholLevels)
```

```
## [1] (166,225] (225,284] (225,284] (225,284] (284,343]  
## [6] (225,284]  
## 10 Levels: (106,166] (166,225] (225,284] ... (637,697]
```

- La respuesta de R nos indica que ha dividido el *recorrido* de la variable (de mínimo a máximo) en 10 intervalos semiabiertos de igual longitud. El primero incluye los valores entre 106 y 166, hasta el último que incluye los valores de 637 a 697.
- La variable `cholLevels` que hemos fabricado es un *factor ordenado*, Veremos más ejemplos cuando aprendamos más sobre factores.

- Con las variables puramente continuas no suele haber demasiado dudas a la hora de reconocerlas. Pero con las variables discretas el problema puede ser más complicado, porque depende esencialmente del número de valores distintos que tome la variable. Al final, en muchos casos, tratar a una variable como discreta o continua es decisión de quien realiza el análisis.
- Por ejemplo, la tabla de frecuencias de la variable agrupada `cholLevels` es mucho más informativa que la de la variable original `totChol`. Eso nos indica que seguramente es mejor tratar a `cholLevels` como una variable continua aunque sus valores sean enteros,

```
table(cholLevels)
```

```
## cholLevels
## (106,166] (166,225] (225,284] (284,343] (343,402] (402,460]
##      164      1555      1898       503        60         7
## (460,519] (519,578] (578,637] (637,697]
##           1           0           1           1
```

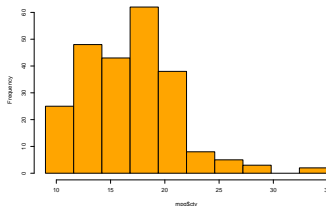
- En cualquier caso si una variable cuantitativa discreta solo toma cinco o menos valores en general es beneficioso pensar en ella como un *factor ordenado*, que discutiremos más adelante.



# Histogramas con R básico

- Una forma común de representar gráficamente la tabla de frecuencias una variable discreta que tome más de cinco valores distintos es mediante un *histograma*, que es un diagrama de barras. Con R básico:

```
cortes = seq(min(mpg$cty), max(mpg$cty), length.out = 11)
hist(mpg$cty, breaks = cortes, col="orange", main="")
```

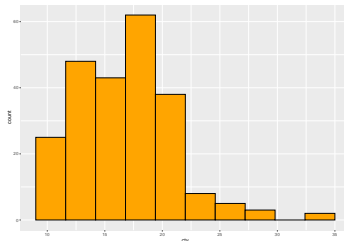


- Fíjate en que el eje horizontal contiene los valores de la variable mientras que el eje vertical muestra las frecuencias. Hemos usado la opción `breaks` combinada con `seq` para elegir los puntos de corte entre intervalos.
- Ejercicio.** Ejecuta `hist(mpg$cyl)`. ¿Por qué ocurre esto?

# Histogramas con ggplot. Número de intervalos.

- O usando ggplot y los mismos puntos de corte:

```
# Histograma con ggplot2 (los mismos cortes)
ggplot(data = mpg) +
  geom_histogram(mapping = aes(cty), breaks = cortes,
                 fill = "orange", color="black")
```

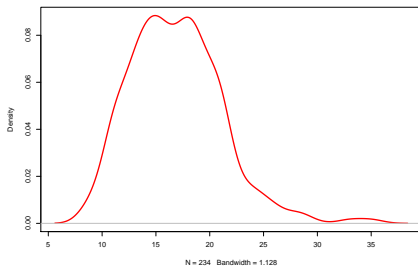


- ¿Cuántos intervalos se deben usar en la construcción de un histograma? No hay una regla fija. Aunque R y el resto de programas utilizan diversos algoritmos para determinar ese número, lo cierto es que la respuesta depende mucho de los datos concretos con los que trabajamos. Por eso normalmente es necesario experimentar un poco con diversos valores. En cualquier caso es *muy desaconsejable* utilizar menos de cinco intervalos (o más que  $\sqrt{n}$ , siendo  $n$  el número de datos).

## Curvas de densidad.

- La *curva de densidad* es un tipo de diagrama alternativo al histograma. Por ejemplo, para los datos de `cty` que venimos usando se obtiene con:

```
plot(density(mpg$cty), col="red", main="", lwd = 3)
```



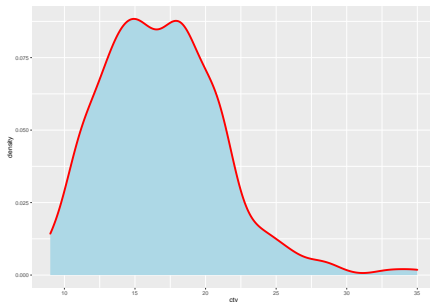
De nuevo el eje horizontal contiene los valores de la variable y la altura de la curva indica la frecuencia de cada valor. La opción `lwd` controla el grosor de la curva.

- Ejercicio:** Usando los datos de `auto2` dibuja la curva de densidad de cada una de las variables `length`, `price`, `displacement` y `rep78`.

# Curvas de densidad con ggplot.

- Para obtener las curvas de densidad en ggplot usaremos algo como:

```
ggplot(mpg) +  
  geom_density(mapping = aes(cty), color="red", fill="lightblue", size=1.5)
```

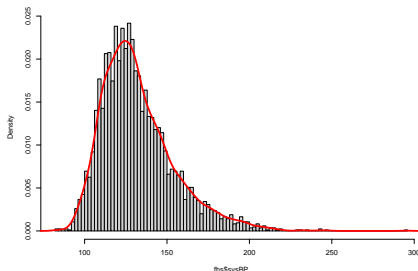


- **¿Curvas de densidad o histogramas?** ¡Ambos! En general, cuando analicemos un conjunto de datos es mejor empezar en la fase de exploración dibujando muchos gráficos. Al presentar nuestras conclusiones seleccionaremos aquellos que ilustren mejor la historia que queremos contar.

## Relación entre curvas de densidad e histogramas.

- En muestras de tamaño grande y usando una partición fina en subintervalos la curva de densidad se ajusta bastante a la forma o perfil del histograma como ilustra este ejemplo. Esa *forma* es lo que llamaremos **distribución** de la variable.

```
hist(x = fhs$sysBP, breaks=150, probability = TRUE, main="")  
lines(density(fhs$sysBP), col="red", lwd=4)
```



Este fenómeno es una manifestación más de esa separación borrosa que existe entre las variables discretas con muchos valores (el histograma es una representación discreta) y las variables continuas (la curva de densidad es una representación continua).

- Ejercicio:** ¿por qué usamos `probability = TRUE` al dibujar el histograma?

## Enlaces

- [Código de esta sesión](#)
- [Cookbook for R](#)
- [Página web de ggplot2](#), que contiene el [resumen \(chuleta\)](#) elaborado por RStudio.
- [Resumen sobre importación de datos a R \(chuleta\)](#) elaborado por RStudio.
- Web del libro [PostData](#) y los tutoriales asociados. Para esta sesión se recomienda el Capítulo 2.

## Bibliografía

Boehmke, B. C. (2016). *Data Wrangling with R* (p. 508). Springer.  
<https://doi.org/10.1007/978-3-319-45599-0>