Master en Big Data. Fundamentos Matemáticos del Análisis de Datos (FMAD).

Tarea 1

García Vázquez, Carlos

Curso 2021-22. Última actualización: 2021-09-16

# Preliminares

Cargamos en memoria los paquetes que van a ser necesarios durante la ejecución de la práctica.

library(tidyverse) # Ya tiene incorporadas dplyr y ggplot  
library(gridExtra)  
#Este tercero, lo necesitaremos para acceder a los datos necesarios en el segundo apartado   
#del último ejercicio  
library(nycflights13)

# Instrucciones preliminares

* Empieza abriendo el proyecto de RStudio correspondiente a tu repositorio personal de la asignatura.
* En todas las tareas tendrás que repetir un proceso como el descrito en la sección *Repite los pasos Creando un fichero Rmarkdown para esta práctica* de la *Práctica00*. Puedes releer la sección *Practicando la entrega de las Tareas* de esa misma práctica para recordar el procedimiento de entrega.

# Ejercicio 1. Análisis exploratorio de un conjunto de datos y operaciones con dplyr.

* Vamos a utilizar el conjunto de datos contenido en el fichero (es un enlace):  
  [cholesterol.csv](https://gist.githubusercontent.com/fsansegundo/ee991e53e1a571dd34034c42b5516eae/raw/2206455b5772e90c5a2a24a3f42a84408fd1d1c5/cholesterol.csv)  
  Los datos proceden de un estudio realizado en la *University of Virginia School of Medicine* que investiga la prevalencia de la obesidad, la diabetes y otros factores de riesgo cardiovascular. Se puede encontrar más información sobre el fichero en este enlace:  
  <https://biostat.app.vumc.org/wiki/pub/Main/DataSets/diabetes.html>
* Carga el conjunto de datos en un data.frame de R llamado chlstrl.

#Cargamos los datos requeridos  
chlstrl=read\_csv("./data/cholesterol.csv")

## Rows: 403 Columns: 7

## -- Column specification --------------------------------------------------------  
## Delimiter: ","  
## chr (1): gender  
## dbl (6): chol, age, height, weight, waist, hip

##   
## i Use `spec()` to retrieve the full column specification for this data.  
## i Specify the column types or set `show\_col\_types = FALSE` to quiet this message.

* Empezaremos por información básica sobre el conjunto de datos. Cuántas observaciones contiene, cuáles son las variables y de qué tipos,…

#Obtenemos una primera visualización de la estructura de datos que contiene la tabla  
head(chlstrl)

## # A tibble: 6 x 7  
## chol age gender height weight waist hip  
## <dbl> <dbl> <chr> <dbl> <dbl> <dbl> <dbl>  
## 1 203 46 female 62 121 29 38  
## 2 165 29 female 64 218 46 48  
## 3 228 58 female 61 256 49 57  
## 4 78 67 male 67 119 33 38  
## 5 249 64 male 68 183 44 41  
## 6 248 34 male 71 190 36 42

Analizamos el número de variables totales a manejar, los tipos, número de registros…

#Número de registros con el que trabajaremos  
nrow(chlstrl)

## [1] 403

#Número de variables con las que trabajaremos  
ncol(chlstrl)

## [1] 7

#Resumen con información básica adicional, como podría ser la tipología de datos que vamos a manejar  
str(chlstrl)

## spec\_tbl\_df [403 x 7] (S3: spec\_tbl\_df/tbl\_df/tbl/data.frame)  
## $ chol : num [1:403] 203 165 228 78 249 248 195 227 177 263 ...  
## $ age : num [1:403] 46 29 58 67 64 34 30 37 45 55 ...  
## $ gender: chr [1:403] "female" "female" "female" "male" ...  
## $ height: num [1:403] 62 64 61 67 68 71 69 59 69 63 ...  
## $ weight: num [1:403] 121 218 256 119 183 190 191 170 166 202 ...  
## $ waist : num [1:403] 29 46 49 33 44 36 46 34 34 45 ...  
## $ hip : num [1:403] 38 48 57 38 41 42 49 39 40 50 ...  
## - attr(\*, "spec")=  
## .. cols(  
## .. chol = col\_double(),  
## .. age = col\_double(),  
## .. gender = col\_character(),  
## .. height = col\_double(),  
## .. weight = col\_double(),  
## .. waist = col\_double(),  
## .. hip = col\_double()  
## .. )  
## - attr(\*, "problems")=<externalptr>

* Asegúrate de comprobar si hay datos ausentes y localízalos en la tabla.

En este tipo de ejercicios, resulta fundamental tener en cuenta los datos ausentes con los que contamos.

#Sacamos el número de datos ausentes en toda la tabla, detectando de esta forma su presencia  
table(is.na(chlstrl)) #Tener en cuenta los TRUE

##   
## FALSE TRUE   
## 2810 11

#Los localizamos en el DataFrame, analizando las posiciones que ocupan  
head(is.na(chlstrl)) #Imprimimos solo el head

## chol age gender height weight waist hip  
## [1,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [2,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [3,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [4,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [5,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [6,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

#Hacemos la cuenta de entre los 403 registros, de cuales son los que contienen algún valor nulo  
table(complete.cases(chlstrl)) #Tener en cuenta los FALSE

##   
## FALSE TRUE   
## 9 394

#Con la siguiente instrucción, localizamos las filas con algún dato ausente  
head(complete.cases(chlstrl)) # Imprimimos solo el head

## [1] TRUE TRUE TRUE TRUE TRUE TRUE

#Finalmente, con el summary, obtenemos el número de nulos en cada una de las variables contempladas   
#en la tabla  
summary(chlstrl)

## chol age gender height   
## Min. : 78.0 Min. :19.00 Length:403 Min. :52.00   
## 1st Qu.:179.0 1st Qu.:34.00 Class :character 1st Qu.:63.00   
## Median :204.0 Median :45.00 Mode :character Median :66.00   
## Mean :207.8 Mean :46.85 Mean :66.02   
## 3rd Qu.:230.0 3rd Qu.:60.00 3rd Qu.:69.00   
## Max. :443.0 Max. :92.00 Max. :76.00   
## NA's :1 NA's :5   
## weight waist hip   
## Min. : 99.0 Min. :26.0 Min. :30.00   
## 1st Qu.:151.0 1st Qu.:33.0 1st Qu.:39.00   
## Median :172.5 Median :37.0 Median :42.00   
## Mean :177.6 Mean :37.9 Mean :43.04   
## 3rd Qu.:200.0 3rd Qu.:41.0 3rd Qu.:46.00   
## Max. :325.0 Max. :56.0 Max. :64.00   
## NA's :1 NA's :2 NA's :2

Una vez disponemos de la información suficiente sobre la presencia de datos ausentes en la tabla, tenemos que decidir cómo gestionar su presencia.

En este caso, se ha decidido, que en las funciones en las que se considere necesario para su correcta utilización, se incorpore el parámetro opcional na.rm=TRUE para ignorar los nulos y que no pasen inadvertidos.

* El análisis exploratorio (numérico y gráfico) debe cubrir todos los tipos de variable de la tabla. Es decir, que al menos debes estudiar una variable por cada tipo de variable presente en la tabla. El análisis debe contener, al menos:
  + Para las variables cuantitativas (continuas o discretas).  
    Resumen numérico básico.  
    Gráficas (las adecuadas, a ser posible más de un tipo de gráfico).

De entre las variables cuantitativas y continuas, seleccionamos la variable ‘chol’ para trabajarla y estudiarla

RESUMEN NUMÉRICO DE LA VARIABLE

#Sacamos el mínimo, máximo,media,mediana, cuartiles y datos ausentes de la variable seleccionada  
summary(chlstrl$chol)

## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's   
## 78.0 179.0 204.0 207.8 230.0 443.0 1

Como ya sabemos, la media se suele ver muy influenciada por la presencia de valores atípicos, al contrario que la mediana, que por su construcción, no se ve tan afectada por estos. De esta forma, la comparación entre ambas nos puede dar información determinante acerca de la presencia de este tipo de valores en la muestra, que si no se tratan correctamente, pueden llevar a interpretaciones erróneas de los datos.

#Calculamos el recorrido intercuartílico  
IQR(chlstrl$chol, na.rm = TRUE)

## [1] 51

#Los valores atípicos  
unname(quantile(chlstrl$chol, probs = c(1/4, 3/4),na.rm=TRUE)   
 + c(-1, 1) \* 1.5 \* IQR(chlstrl$chol, na.rm = TRUE))

## [1] 102.5 306.5

Con estas 2 últimas funciones, tenemos un mayor conocimiento de la dispersión de la variable a estudiar. Y concretamente, con la función unname(), conocemos el rango (102.5-306.5) a partir del cual podremos distinguir los valores atípicos de la muestra.

#Desviación típica  
sd(chlstrl$chol, na.rm = TRUE)

## [1] 44.44556

#Varianza  
var(chlstrl$chol, na.rm = TRUE)

## [1] 1975.408

Por último, hemos calculado tanto la varianza como la desviación típica de la variable, para tener aún más información acerca de la dispersión y variabilidad que tienen lo datos respecto a su media.

De esta forma, podemos completar el estudio y tener un resumen global a nivel numérico del comportamiento de la variable ‘chol’

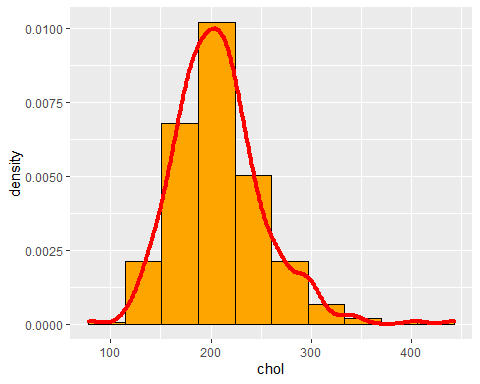
GRÁFICAS

1. Histograma

#FORMA 1  
  
#Hacemos los cortes para realizar 10 intervalos, omitiendo los valores nulos de la variable  
cortes = seq(min(chlstrl$chol,na.rm=TRUE), max(chlstrl$chol,na.rm=TRUE), length.out = 11)  
#Graficamos  
ggplot(chlstrl, aes(x = chol)) +   
 geom\_histogram(aes(y=stat(density)),   
 breaks=cortes, fill = "orange", color="black") +   
 geom\_density(color="red", size=1.5)

## Warning: Removed 1 rows containing non-finite values (stat\_bin).

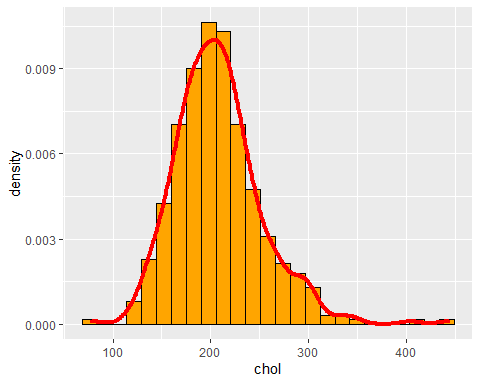
## Warning: Removed 1 rows containing non-finite values (stat\_density).



#FORMA 2  
  
#En este caso, especificamos el número de intervalos con bins  
#Lo dividiremos en 25 intervalos, en este caso sin especificar de forma explicita los intervalos  
ggplot(chlstrl, aes(x = chol)) +   
 geom\_histogram(aes(y=stat(density)),   
 bins=25, fill = "orange", color="black") +   
 geom\_density(color="red", size=1.5)

## Warning: Removed 1 rows containing non-finite values (stat\_bin).

## Warning: Removed 1 rows containing non-finite values (stat\_density).

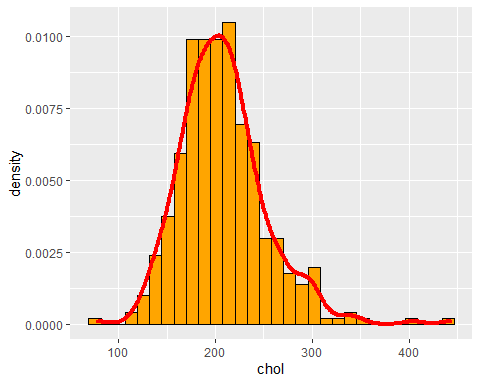


#FORMA 3  
  
#En este último caso, simplemente dejamos que R defina los intervalos  
ggplot(chlstrl, aes(x = chol)) +   
 geom\_histogram(aes(y=stat(density))   
 , fill = "orange", color="black") +   
 geom\_density(color="red", size=1.5)

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.

## Warning: Removed 1 rows containing non-finite values (stat\_bin).

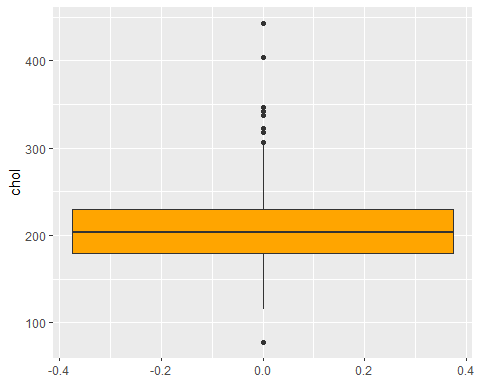
## Warning: Removed 1 rows containing non-finite values (stat\_density).

 Mediante el histograma se puede apreciar la distribución de la variable a estudiar, que es asimétrica a la derecha y se asemeja a la normal.

1. Boxplot

#Boxplot normal  
#Podemos ver como los valores atípicos se grafican individualmente  
ggplot(chlstrl) +   
 geom\_boxplot(mapping = aes(y = chol), fill="orange")

## Warning: Removed 1 rows containing non-finite values (stat\_boxplot).

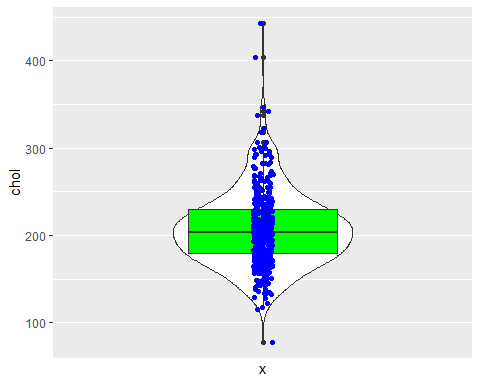


#Violinplot  
ggplot(chlstrl) +   
 geom\_violin(mapping = aes(x=0, y = chol)) +  
 scale\_x\_discrete(breaks = c()) +  
 geom\_boxplot(mapping = aes(y = chol), fill="green") +  
 geom\_jitter(aes(x=0, y = chol),   
 position = position\_jitter(w=0.05, h= 0), col="blue")

## Warning: Removed 1 rows containing non-finite values (stat\_ydensity).

## Warning: Removed 1 rows containing non-finite values (stat\_boxplot).

## Warning: Removed 1 rows containing missing values (geom\_point).



El Boxplot y Violinplot representados, también nos permiten hacernos una idea de la distribución de la variable, además de identificar los valores atípicos.

* Variables categóricas (factores).  
  Tablas de frecuencia (absolutas y relativas).  
  Gráficas (diagrama de barras).

Como variable categórica haremos uso de ‘gender’, que es la única de este tipo que podemos encontrar en el DataFrame

INFORMACIÓN DE LA VARIABLE

summary(chlstrl$gender)

## Length Class Mode   
## 403 character character

TABLAS DE FRECUENCIA

#table(chlstrl$gender) con R básico  
chlstrl %>%  
 count(gender) #Con dplyr

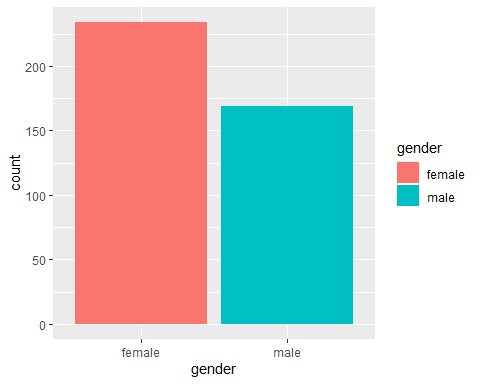
## # A tibble: 2 x 2  
## gender n  
## <chr> <int>  
## 1 female 234  
## 2 male 169

#prop.table(table(chlstrl$gender)) con R básico  
chlstrl %>%   
 count(gender) %>%  
 mutate(gender, relFreq = prop.table(n), n=NULL) #Con dplyr

## # A tibble: 2 x 2  
## gender relFreq  
## <chr> <dbl>  
## 1 female 0.581  
## 2 male 0.419

GRÁFICO DE BARRAS

ggplot(chlstrl) +   
 geom\_bar(aes(x=gender,fill=gender))



* Los valores de height y weight están en pulgadas (inches) y libras (pounds) respectivamente. Una libra son 0.454kg y una pulgada son 0.0254m. Usa dplyr para convertir esas columnas a metros y kilogramos respectivamente. Las nuevas columnas deben llamarse igual que las originales.

#Cambiamos las unidades de los valores asociados a las variables 'height' y 'weight'   
#sobreescribiendo chlstrl  
(chlstrl=chlstrl %>%   
 mutate(height=height\*0.0254,weight=weight\*0.454))

## # A tibble: 403 x 7  
## chol age gender height weight waist hip  
## <dbl> <dbl> <chr> <dbl> <dbl> <dbl> <dbl>  
## 1 203 46 female 1.57 54.9 29 38  
## 2 165 29 female 1.63 99.0 46 48  
## 3 228 58 female 1.55 116. 49 57  
## 4 78 67 male 1.70 54.0 33 38  
## 5 249 64 male 1.73 83.1 44 41  
## 6 248 34 male 1.80 86.3 36 42  
## 7 195 30 male 1.75 86.7 46 49  
## 8 227 37 male 1.50 77.2 34 39  
## 9 177 45 male 1.75 75.4 34 40  
## 10 263 55 female 1.60 91.7 45 50  
## # ... with 393 more rows

* Ahora usa esos valores de height y weight para añadir una nueva columna llamada BMI, definida mediante:
* (se divide por el cuadrado de la altura).

#Añadimos la nueva columna especificada  
(chlstrl=chlstrl %>%   
 mutate(BMI=weight/(height)^2))

## # A tibble: 403 x 8  
## chol age gender height weight waist hip BMI  
## <dbl> <dbl> <chr> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 203 46 female 1.57 54.9 29 38 22.2  
## 2 165 29 female 1.63 99.0 46 48 37.5  
## 3 228 58 female 1.55 116. 49 57 48.4  
## 4 78 67 male 1.70 54.0 33 38 18.7  
## 5 249 64 male 1.73 83.1 44 41 27.8  
## 6 248 34 male 1.80 86.3 36 42 26.5  
## 7 195 30 male 1.75 86.7 46 49 28.2  
## 8 227 37 male 1.50 77.2 34 39 34.4  
## 9 177 45 male 1.75 75.4 34 40 24.5  
## 10 263 55 female 1.60 91.7 45 50 35.8  
## # ... with 393 more rows

* Crea una nueva columna llamada ageGroup dividiendo la edad en los siguientes tres niveles:

#Añadimos la nueva columna especificada  
(chlstrl=chlstrl %>%   
 mutate(ageGroup=cut(age,breaks=c(10,40,70,100))))

## # A tibble: 403 x 9  
## chol age gender height weight waist hip BMI ageGroup  
## <dbl> <dbl> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <fct>   
## 1 203 46 female 1.57 54.9 29 38 22.2 (40,70]   
## 2 165 29 female 1.63 99.0 46 48 37.5 (10,40]   
## 3 228 58 female 1.55 116. 49 57 48.4 (40,70]   
## 4 78 67 male 1.70 54.0 33 38 18.7 (40,70]   
## 5 249 64 male 1.73 83.1 44 41 27.8 (40,70]   
## 6 248 34 male 1.80 86.3 36 42 26.5 (10,40]   
## 7 195 30 male 1.75 86.7 46 49 28.2 (10,40]   
## 8 227 37 male 1.50 77.2 34 39 34.4 (10,40]   
## 9 177 45 male 1.75 75.4 34 40 24.5 (40,70]   
## 10 263 55 female 1.60 91.7 45 50 35.8 (40,70]   
## # ... with 393 more rows

* Usando dplyr calcula cuántas observaciones hay en cada nivel de ageGroup (indicación: usa group\_by). Ahora, usando aquellas observaciones que corresponden a mujeres, ¿cuál es la media del nivel de colesterol y de BMI en cada uno de esos grupos de edad?

#Numero de observaciones para cada nivel de ageGroup  
chlstrl %>%   
 count(ageGroup)

## # A tibble: 3 x 2  
## ageGroup n  
## <fct> <int>  
## 1 (10,40] 160  
## 2 (40,70] 207  
## 3 (70,100] 36

#De entre las mujeres, media del nivel de colesterol y de BMI en cada uno de esos grupos de edad  
chlstrl %>%   
 filter(gender=='female') %>%  
 group\_by(ageGroup) %>%  
 summarise(mediaCol=mean(chol,na.rm=TRUE),mediaBMI=mean(BMI,na.rm=TRUE))

## # A tibble: 3 x 3  
## ageGroup mediaCol mediaBMI  
## <fct> <dbl> <dbl>  
## 1 (10,40] 189. 30.5  
## 2 (40,70] 221. 30.3  
## 3 (70,100] 230. 29.4

# Ejercicio 2: Funciones de R.

* Crea una función de R llamada cambiosSigno que dado un vector x de números enteros no nulos, como
* -12, -19, 9, -13, -14, -17, 8, -19, -14,
* calcule cuántos cambios de signo ha habido. Es decir, cuántas veces el signo de un elemento es distinto del signo del elemento previo. Por ejemplo, en el vector anterior hay 4 cambios de signo (en las posiciones 3, 4, 7 y 8).

cambioSigno=function(x=sample(c(-1, 1), 9, replace = TRUE) \* sample(1:20, 9, replace = TRUE)) {  
   
 sol=list()  
 cont=0  
 for (i in 1:(length(x)-1)) {  
 if ((x[i]>0 && x[i+1]<0) | (x[i]<0 && x[i+1]>0)){  
 cont=cont+1  
 }  
 }  
 sol$vector=x  
 sol$cambSigno=cont  
 returnValue(sol)  
}  
cambioSigno(x)

## $vector  
## [1] -12 -19 9 -13 -14 -17 8 -19 -14  
##   
## $cambSigno  
## [1] 4

#Si no introduces nada  
cambioSigno()

## $vector  
## [1] 5 -1 14 -7 6 -6 -17 -16 6  
##   
## $cambSigno  
## [1] 6

* Modifica la función para que devuelva como resultado las posiciones donde hay cambios de signo. Llama cambiosSignoPos(x) a esa otra función. Por ejemplo, para el vector anterior el resultado de esta función sería

cambiosSignoPos=function(x=sample(c(-1, 1), 9, replace = TRUE) \* sample(1:20, 9, replace = TRUE)) {  
 sol=list()  
 vectorPos=c()  
 for (i in 1:(length(x)-1)) {  
 if ((x[i]>0 && x[i+1]<0) | (x[i]<0 && x[i+1]>0)){  
 vectorPos=c(vectorPos,i+1)  
 }  
 }  
 sol$vector=x  
 sol$posiciones=vectorPos  
 returnValue(sol)  
}  
cambiosSignoPos(x)

## $vector  
## [1] -12 -19 9 -13 -14 -17 8 -19 -14  
##   
## $posiciones  
## [1] 3 4 7 8

#Si no introduces nada  
cambioSigno()

## $vector  
## [1] -14 17 -16 -2 17 -13 1 9 -12  
##   
## $cambSigno  
## [1] 6

Podemos ver como ambas funcionan , independientemente de si se les introduce un vector como argumento o no. Concretamente, en el caso de no introducir ningun vector, se crea uno por defecto.

# Ejercicio 3. R4DS.

Es recomendable que esta semana del curso hagas al menos una lectura somera de los Capítulos 1 a 5 de [R for Data Science (R4DS), de H. Wickham](https://r4ds.had.co.nz/index.html), con énfasis especial en los Capítulos 3 y 5 (los capítulos 1, 2 y 4 son muy breves). Los siguientes apartados pretenden motivar esa lectura y por eso mismo pueden resultar un poco más laboriosos.

* Haz el [ejercicio 6 de la Sección 3.6.1 de R4DS](https://r4ds.had.co.nz/data-visualisation.html#exercises-3).

#Vemos la estructura de la tabla con los datos que se van a trabajar  
head(mpg)

## # A tibble: 6 x 11  
## manufacturer model displ year cyl trans drv cty hwy fl class   
## <chr> <chr> <dbl> <int> <int> <chr> <chr> <int> <int> <chr> <chr>   
## 1 audi a4 1.8 1999 4 auto(l5) f 18 29 p compa~  
## 2 audi a4 1.8 1999 4 manual(m5) f 21 29 p compa~  
## 3 audi a4 2 2008 4 manual(m6) f 20 31 p compa~  
## 4 audi a4 2 2008 4 auto(av) f 21 30 p compa~  
## 5 audi a4 2.8 1999 6 auto(l5) f 16 26 p compa~  
## 6 audi a4 2.8 1999 6 manual(m5) f 18 26 p compa~

Gráfico 1

g1=ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +   
 geom\_point() +   
 geom\_smooth(se=FALSE)

Gráfico 2

g2=ggplot(data = mpg,aes(x = displ, y = hwy)) +  
 geom\_point() +   
 geom\_smooth(aes(group = drv),se=FALSE)

Gráfico 3

g3=ggplot(data = mpg,aes(x = displ, y = hwy,color=drv)) +  
 geom\_point() +   
 geom\_smooth(se=FALSE)

Gráfico 4

g4=ggplot(data = mpg,aes(x = displ, y = hwy)) +  
 geom\_point(aes(color=drv)) +   
 geom\_smooth(se=FALSE)

Gráfico 5

g5=ggplot(data = mpg,aes(x = displ, y = hwy)) +  
 geom\_point(aes(color=drv)) +   
 geom\_smooth(aes(linetype=drv),se=FALSE)

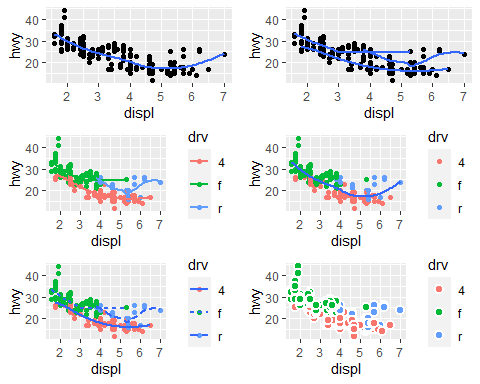
Gráfico 6

g6=ggplot(data = mpg) +   
 geom\_point(aes(x = displ, y = hwy, fill=drv), shape = 21,color="white",size= 2.5, stroke = 1.5)

Mezclamos los 6 gráficos en una única representación

grid.arrange(g1, g2, g3, g4, g5, g6, nrow = 3)

## `geom\_smooth()` using method = 'loess' and formula 'y ~ x'  
## `geom\_smooth()` using method = 'loess' and formula 'y ~ x'  
## `geom\_smooth()` using method = 'loess' and formula 'y ~ x'  
## `geom\_smooth()` using method = 'loess' and formula 'y ~ x'  
## `geom\_smooth()` using method = 'loess' and formula 'y ~ x'



* Haz el [ejercicio 1 de la Sección 5.2.4 de R4DS](https://r4ds.had.co.nz/transform.html#exercises-8).

#Vemos la estructura de la tabla con los datos que se van a trabajar  
head(flights)

## # A tibble: 6 x 19  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time sched\_arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int> <int>  
## 1 2013 1 1 517 515 2 830 819  
## 2 2013 1 1 533 529 4 850 830  
## 3 2013 1 1 542 540 2 923 850  
## 4 2013 1 1 544 545 -1 1004 1022  
## 5 2013 1 1 554 600 -6 812 837  
## 6 2013 1 1 554 558 -4 740 728  
## # ... with 11 more variables: arr\_delay <dbl>, carrier <chr>, flight <int>,  
## # tailnum <chr>, origin <chr>, dest <chr>, air\_time <dbl>, distance <dbl>,  
## # hour <dbl>, minute <dbl>, time\_hour <dttm>

Find all flights that:

Had an arrival delay of two or more hours

flights %>%  
 filter(arr\_delay>=2\*60)

## # A tibble: 10,200 x 19  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time sched\_arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int> <int>  
## 1 2013 1 1 811 630 101 1047 830  
## 2 2013 1 1 848 1835 853 1001 1950  
## 3 2013 1 1 957 733 144 1056 853  
## 4 2013 1 1 1114 900 134 1447 1222  
## 5 2013 1 1 1505 1310 115 1638 1431  
## 6 2013 1 1 1525 1340 105 1831 1626  
## 7 2013 1 1 1549 1445 64 1912 1656  
## 8 2013 1 1 1558 1359 119 1718 1515  
## 9 2013 1 1 1732 1630 62 2028 1825  
## 10 2013 1 1 1803 1620 103 2008 1750  
## # ... with 10,190 more rows, and 11 more variables: arr\_delay <dbl>,  
## # carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,  
## # air\_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time\_hour <dttm>

Flew to Houston (IAH or HOU)

flights %>%  
 filter(dest %in% c('HOU','IAH'))

## # A tibble: 9,313 x 19  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time sched\_arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int> <int>  
## 1 2013 1 1 517 515 2 830 819  
## 2 2013 1 1 533 529 4 850 830  
## 3 2013 1 1 623 627 -4 933 932  
## 4 2013 1 1 728 732 -4 1041 1038  
## 5 2013 1 1 739 739 0 1104 1038  
## 6 2013 1 1 908 908 0 1228 1219  
## 7 2013 1 1 1028 1026 2 1350 1339  
## 8 2013 1 1 1044 1045 -1 1352 1351  
## 9 2013 1 1 1114 900 134 1447 1222  
## 10 2013 1 1 1205 1200 5 1503 1505  
## # ... with 9,303 more rows, and 11 more variables: arr\_delay <dbl>,  
## # carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,  
## # air\_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time\_hour <dttm>

Were operated by United, American, or Delta

flights %>%  
 filter(carrier %in% c('UA','AA','DL'))

## # A tibble: 139,504 x 19  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time sched\_arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int> <int>  
## 1 2013 1 1 517 515 2 830 819  
## 2 2013 1 1 533 529 4 850 830  
## 3 2013 1 1 542 540 2 923 850  
## 4 2013 1 1 554 600 -6 812 837  
## 5 2013 1 1 554 558 -4 740 728  
## 6 2013 1 1 558 600 -2 753 745  
## 7 2013 1 1 558 600 -2 924 917  
## 8 2013 1 1 558 600 -2 923 937  
## 9 2013 1 1 559 600 -1 941 910  
## 10 2013 1 1 559 600 -1 854 902  
## # ... with 139,494 more rows, and 11 more variables: arr\_delay <dbl>,  
## # carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,  
## # air\_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time\_hour <dttm>

Departed in summer (July, August, and September)

flights %>%  
 filter(month %in% c(7:9))

## # A tibble: 86,326 x 19  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time sched\_arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int> <int>  
## 1 2013 7 1 1 2029 212 236 2359  
## 2 2013 7 1 2 2359 3 344 344  
## 3 2013 7 1 29 2245 104 151 1  
## 4 2013 7 1 43 2130 193 322 14  
## 5 2013 7 1 44 2150 174 300 100  
## 6 2013 7 1 46 2051 235 304 2358  
## 7 2013 7 1 48 2001 287 308 2305  
## 8 2013 7 1 58 2155 183 335 43  
## 9 2013 7 1 100 2146 194 327 30  
## 10 2013 7 1 100 2245 135 337 135  
## # ... with 86,316 more rows, and 11 more variables: arr\_delay <dbl>,  
## # carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,  
## # air\_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time\_hour <dttm>

Arrived more than two hours late, but didn’t leave late

flights %>%  
 filter(dep\_delay<=0,arr\_delay>2\*60)

## # A tibble: 29 x 19  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time sched\_arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int> <int>  
## 1 2013 1 27 1419 1420 -1 1754 1550  
## 2 2013 10 7 1350 1350 0 1736 1526  
## 3 2013 10 7 1357 1359 -2 1858 1654  
## 4 2013 10 16 657 700 -3 1258 1056  
## 5 2013 11 1 658 700 -2 1329 1015  
## 6 2013 3 18 1844 1847 -3 39 2219  
## 7 2013 4 17 1635 1640 -5 2049 1845  
## 8 2013 4 18 558 600 -2 1149 850  
## 9 2013 4 18 655 700 -5 1213 950  
## 10 2013 5 22 1827 1830 -3 2217 2010  
## # ... with 19 more rows, and 11 more variables: arr\_delay <dbl>, carrier <chr>,  
## # flight <int>, tailnum <chr>, origin <chr>, dest <chr>, air\_time <dbl>,  
## # distance <dbl>, hour <dbl>, minute <dbl>, time\_hour <dttm>

Were delayed by at least an hour, but made up over 30 minutes in flight

flights %>%  
 filter(dep\_delay>=60,(arr\_delay-dep\_delay)<(-30))

## # A tibble: 1,844 x 19  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time sched\_arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int> <int>  
## 1 2013 1 1 2205 1720 285 46 2040  
## 2 2013 1 1 2326 2130 116 131 18  
## 3 2013 1 3 1503 1221 162 1803 1555  
## 4 2013 1 3 1839 1700 99 2056 1950  
## 5 2013 1 3 1850 1745 65 2148 2120  
## 6 2013 1 3 1941 1759 102 2246 2139  
## 7 2013 1 3 1950 1845 65 2228 2227  
## 8 2013 1 3 2015 1915 60 2135 2111  
## 9 2013 1 3 2257 2000 177 45 2224  
## 10 2013 1 4 1917 1700 137 2135 1950  
## # ... with 1,834 more rows, and 11 more variables: arr\_delay <dbl>,  
## # carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,  
## # air\_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time\_hour <dttm>

Departed between midnight and 6am (inclusive)

flights %>%  
 filter(dep\_time ==2400 | dep\_time<=600)

## # A tibble: 9,373 x 19  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time sched\_arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int> <int>  
## 1 2013 1 1 517 515 2 830 819  
## 2 2013 1 1 533 529 4 850 830  
## 3 2013 1 1 542 540 2 923 850  
## 4 2013 1 1 544 545 -1 1004 1022  
## 5 2013 1 1 554 600 -6 812 837  
## 6 2013 1 1 554 558 -4 740 728  
## 7 2013 1 1 555 600 -5 913 854  
## 8 2013 1 1 557 600 -3 709 723  
## 9 2013 1 1 557 600 -3 838 846  
## 10 2013 1 1 558 600 -2 753 745  
## # ... with 9,363 more rows, and 11 more variables: arr\_delay <dbl>,  
## # carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,  
## # air\_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time\_hour <dttm>