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

Tarea 1

Suero, Jesús

Curso 2021-22. Última actualización: 2021-09-17

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 0

- Si no has hecho los *Ejercicios* de la *Práctica00* (págs. 12 y 13) hazlos ahora y añádelos a esta tarea. Si ya los has hecho y entregado a través de GitHub no hace falta que hagas nada.

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
  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`.

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.4     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.1     v forcats 0.5.1
```

```
## -- Conflicts ---------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
chlstrl = read_csv("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,...

```
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>
```

```
glimpse(chlstrl)
```

```
## Rows: 403
## Columns: 7
## $ chol   <dbl> 203, 165, 228, 78, 249, 248, 195, 227, 177, 263, 242, 215, 238,~
## $ age    <dbl> 46, 29, 58, 67, 64, 34, 30, 37, 45, 55, 60, 38, 27, 40, 36, 33,~
## $ gender <chr> "female", "female", "female", "male", "male", "male", "male", "~
## $ height <dbl> 62, 64, 61, 67, 68, 71, 69, 59, 69, 63, 65, 58, 60, 59, 69, 65,~
## $ weight <dbl> 121, 218, 256, 119, 183, 190, 191, 170, 166, 202, 156, 195, 170~
## $ waist  <dbl> 29, 46, 49, 33, 44, 36, 46, 34, 34, 45, 39, 42, 35, 37, 36, 37,~
## $ hip    <dbl> 38, 48, 57, 38, 41, 42, 49, 39, 40, 50, 45, 50, 41, 43, 40, 41,~
```

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

```
any(is.na(chlstrl))
```

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

```
apply(is.na(chlstrl),2,which)
```

```
## $chol
## [1] 28
##
## $age
## integer(0)
##
## $gender
## integer(0)
##
## $height
## [1]  64  87 196 232 318
##
## $weight
## [1] 162
##
## $waist
## [1] 337 394
##
## $hip
## [1] 337 394
```

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

Comenzamos por la variable chol, variable cuantitativa a priori discreta

```
table(chlstrl$chol)
```

```
##
##  78 115 118 122 128 129 132 134 135 136 138 140 142 143 144 145 146 147 148 149
##   1   1   1   1   1   1   2   2   2   1   2   1   1   2   1   2   2   2   1   2
## 150 151 152 155 156 157 158 159 160 162 163 164 165 166 168 169 170 171 172 173
##   1   2   1   2   1   3   3   4   5   2   2   4   3   1   3   3   4   3   3   5
## 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193
##   5   1   2   2   2  11   3   5   4   3   4   3   2   1   4   4   4   5   5   5
```

```
## 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213
##   7   3   4   2   3   6   1   3   2   7   9   3   5   5   2   5   1   3   4   3
## 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233
##   4   7   4   5   6   7   2   2   1   3   4   3   4   4   5   2   2   3   2   1
## 234 235 236 237 238 239 240 241 242 243 244 245 246 248 249 251 252 254 255 260
##   3   4   2   4   1   2   2   2   4   2   3   2   1   1   3   2   1   3   4   2
## 261 262 263 265 266 267 268 269 270 271 273 277 279 281 283 284 289 292 293 296
##   1   2   2   1   1   1   2   2   1   2   1   2   1   1   2   1   2   1   3   2
## 298 300 301 302 305 306 307 318 322 337 342 347 404 443
##   1   2   1   1   1   1   1   1   1   1   1   1   1   1
```

La tabla anterior no ofrece mucha información, así que trataremos a chol como una variable discreta Para ello dividimos los valores que puede tomar chol en 10:

```
cholPorNiveles = cut(chlstrl$chol, breaks = 10)
```

Tabla de frecuencia absoluta:

```
table(cholPorNiveles)
```

```
## cholPorNiveles
## (77.6,114]  (114,151]  (151,188]  (188,224]  (224,260]  (260,297]  (297,334]
##          1         31        100        150         74         31         10
##  (334,370]  (370,406]  (406,443]
##          3          1          1
```

Tabla de frecuencia relativa:

```
signif(prop.table(table(cholPorNiveles)), 2)
```

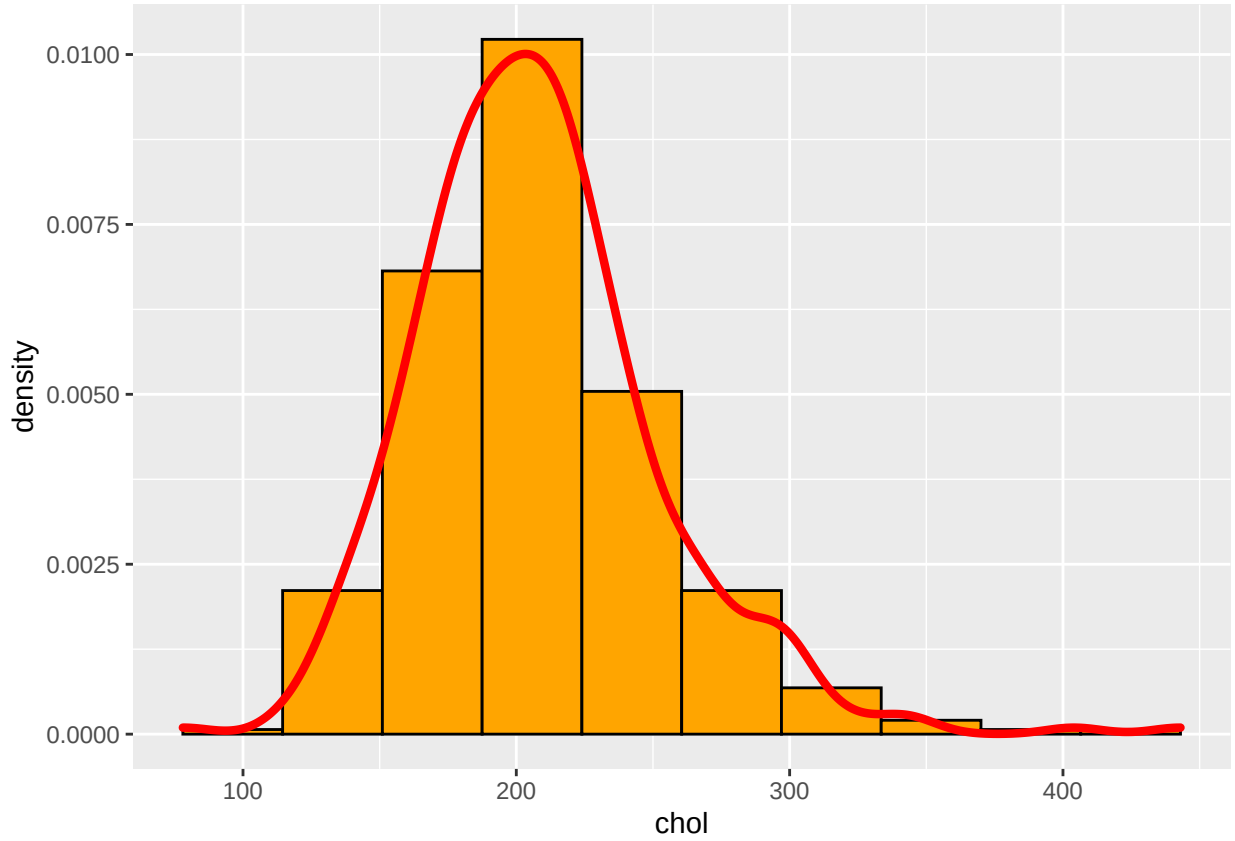
```
## cholPorNiveles
## (77.6,114]  (114,151]  (151,188]  (188,224]  (224,260]  (260,297]  (297,334]
##     0.0025     0.0770     0.2500     0.3700     0.1800     0.0770     0.0250
##  (334,370]  (370,406]  (406,443]
##     0.0075     0.0025     0.0025
```

Histograma + curva de densidad:

```
cortes = seq(min(na.omit(chlstrl$chol)), max(na.omit(chlstrl$chol)), length.out = 11)
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).
```



Ahora vamos a analizar la variable categórica gender: Tabla de frecuencia absoluta:

```
table(chlstrl$gender)
```

```
##
## female   male
##    234    169
```

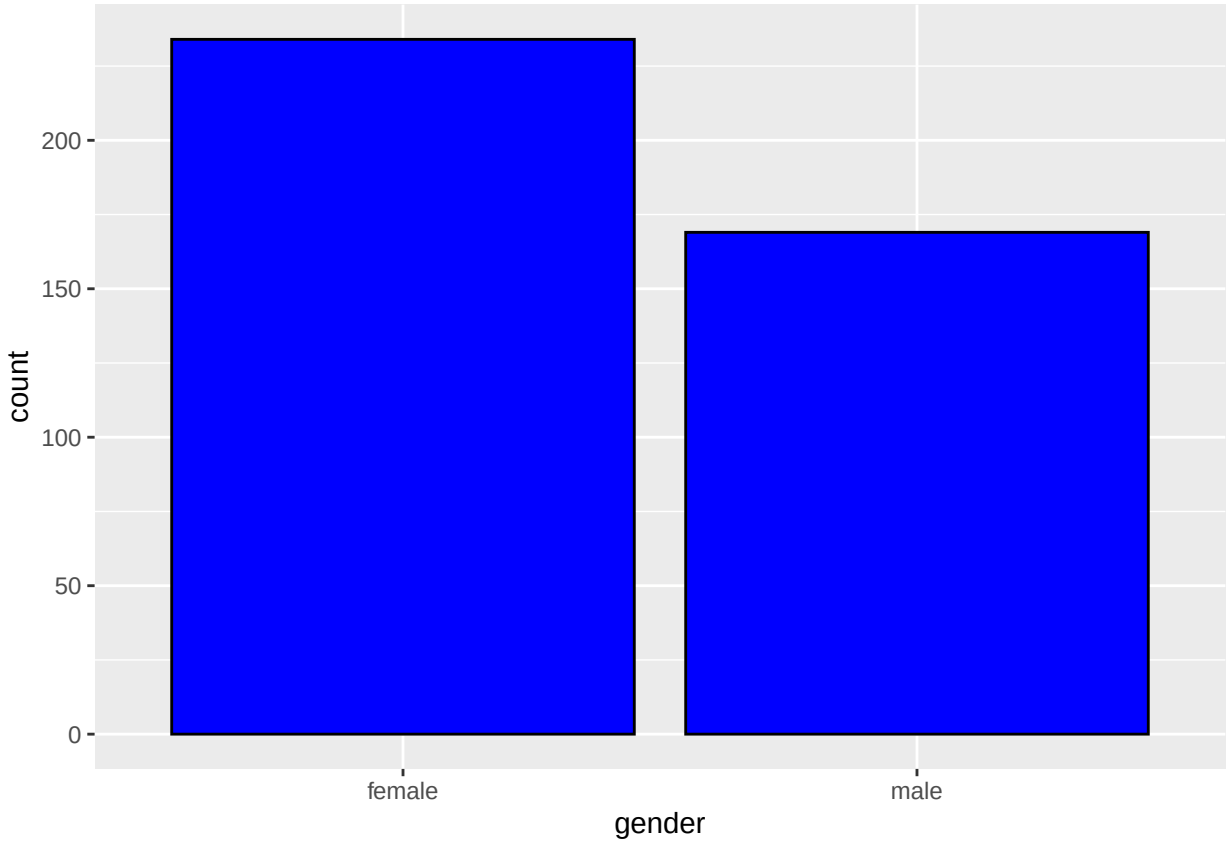
Tabla de frecuencia relativa:

```
signif(prop.table(table(chlstrl$gender)), 2)
```

```
##
## female   male
##   0.58   0.42
```

Diagrama de barras:

```
ggplot(chlstrl) +
  geom_bar(mapping = aes(x = gender), fill='blue', color='black')
```



- 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.

```
library(dplyr)
```

```
chlstrl %>%
  mutate(height = height * 0.0254) %>%
  mutate(weight = weight * 0.454) %>%
  head(10)
```

```
## # A tibble: 10 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
```

- Ahora usa esos valores de `height` y `weight` para añadir una nueva columna llamada BMI, definida mediante:

$$BMI = \frac{weight}{height^2}$$

```
chlstrl %>%
  mutate(BMI = weight/(height^2)) %>%
  head(10)
```

```
## # A tibble: 10 x 8
##     chol   age gender height weight waist   hip    BMI
##    <dbl> <dbl> <chr>   <dbl>  <dbl> <dbl> <dbl>  <dbl>
##  1   203    46 female     62    121    29    38 0.0315
##  2   165    29 female     64    218    46    48 0.0532
##  3   228    58 female     61    256    49    57 0.0688
##  4    78    67 male       67    119    33    38 0.0265
##  5   249    64 male       68    183    44    41 0.0396
##  6   248    34 male       71    190    36    42 0.0377
##  7   195    30 male       69    191    46    49 0.0401
##  8   227    37 male       59    170    34    39 0.0488
##  9   177    45 male       69    166    34    40 0.0349
## 10   263    55 female     63    202    45    50 0.0509
```

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

  `(10,40], (40,70], (70,100]`

```
chlstrl %>%
  mutate(ageGroup = cut(age, breaks = c(10, 40, 70, 100))) %>%
  head(10)
```

```
## # A tibble: 10 x 8
##     chol   age gender height weight waist   hip ageGroup
##    <dbl> <dbl> <chr>   <dbl>  <dbl> <dbl> <dbl> <fct>
##  1   203    46 female     62    121    29    38 (40,70]
##  2   165    29 female     64    218    46    48 (10,40]
##  3   228    58 female     61    256    49    57 (40,70]
##  4    78    67 male       67    119    33    38 (40,70]
##  5   249    64 male       68    183    44    41 (40,70]
##  6   248    34 male       71    190    36    42 (10,40]
##  7   195    30 male       69    191    46    49 (10,40]
##  8   227    37 male       59    170    34    39 (10,40]
##  9   177    45 male       69    166    34    40 (40,70]
## 10   263    55 female     63    202    45    50 (40,70]
```

- Usando `dplyr` calcula cuántas observaciones hay en cada nivel de `ageGroup`.

```
chlstrl %>%
  mutate(ageGroup = cut(age, breaks = c(10, 40, 70, 100))) %>%
  group_by(ageGroup) %>%
  summarise(observacionesPorNivel=n())
```

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

- 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?

```
chlstrl %>%
  mutate(ageGroup = cut(age, breaks = c(10, 40, 70, 100))) %>%
  mutate(BMI = weight/(height^2)) %>%
  filter(gender=='female') %>%
  group_by(ageGroup) %>%
  summarise(observacionesPorNivel=n(), cholMean = mean(chol), BMIMean = mean(na.omit(BMI)))
```

```
## # A tibble: 3 x 4
##   ageGroup observacionesPorNivel cholMean BMIMean
##   <fct>                    <int>    <dbl>   <dbl>
## 1 (10,40]                     97     189.  0.0433
## 2 (40,70]                    117     221.  0.0430
## 3 (70,100]                    20     230.  0.0417
```

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).

```
genPasswd = function(x){
  c = 0
  for (i in 1:(length(x)-1)){
    if((x[i]<0 && x[i+1]>=0) || (x[i]>=0 && x[i+1]<0))
      c = c + 1
  }
  return(c)
}
genPasswd(x)
```

```
## [1] 4
```

```
(x1 = sample(-50:50, 20, replace = TRUE))
```

```
##  [1]  43  21  16  33  -4  18  16 -15 -46  44  36 -26  41 -50  -5  12 -44  41  19
## [20]  41
```

```
genPasswd(x1)
```

```
## [1] 10
```

- 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 [1] 3 4 7 8

```
cambiosSignoPos = function(x){
  v = c()
  for (i in 1:(length(x)-1)){
    if((x[i]<0 && x[i+1]>=0) || (x[i]>=0 && x[i+1]<0))
      v = c(v, i+1)
  }
  return(v)
}
cambiosSignoPos(x)
```

```
## [1] 3 4 7 8
```

```
cambiosSignoPos(x1)
```

```
##  [1]  5  6  8 10 12 13 14 16 17 18
```

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, 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.

Recreate the R code necessary to generate the following graphs: Gráfico 1:

```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_point() +
  geom_smooth(se=FALSE)
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

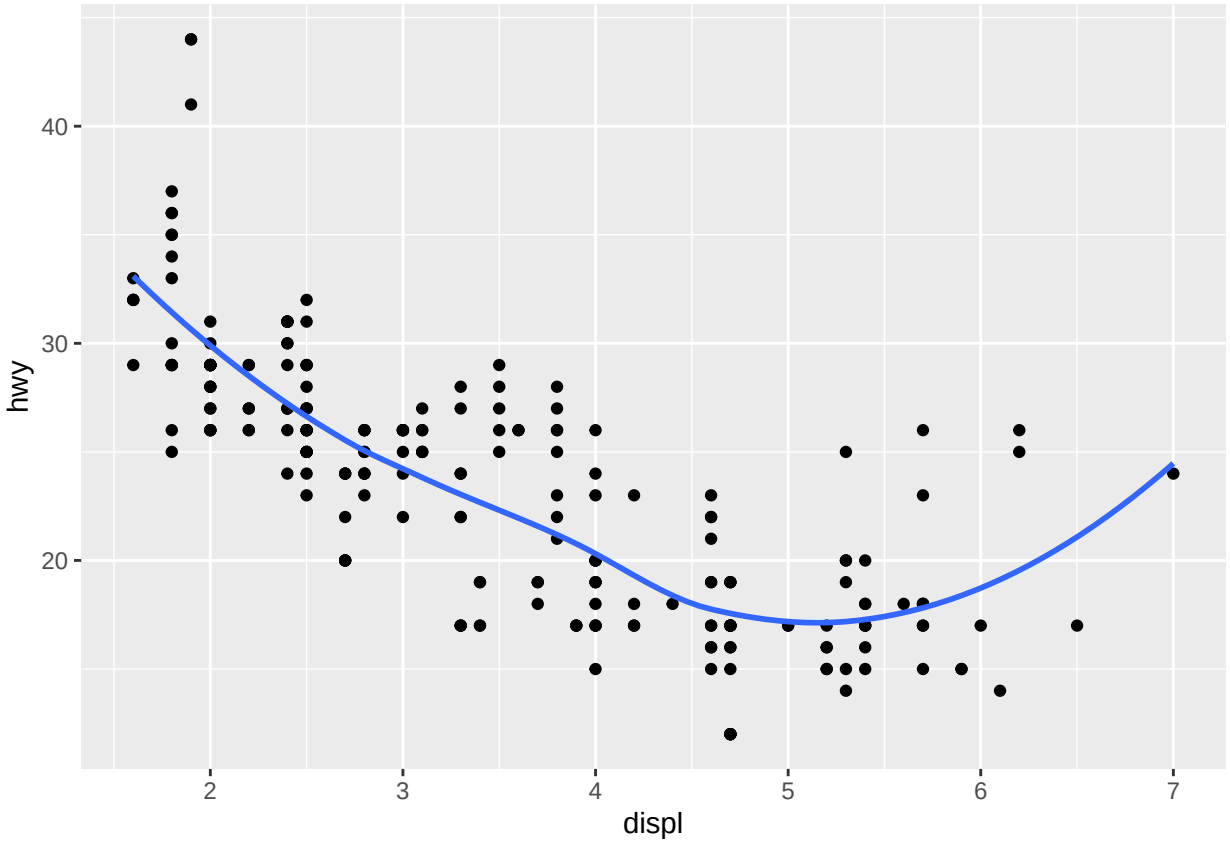


Gráfico 2:

```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_point() +
  geom_smooth(se=FALSE, mapping=aes(group=drv))
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

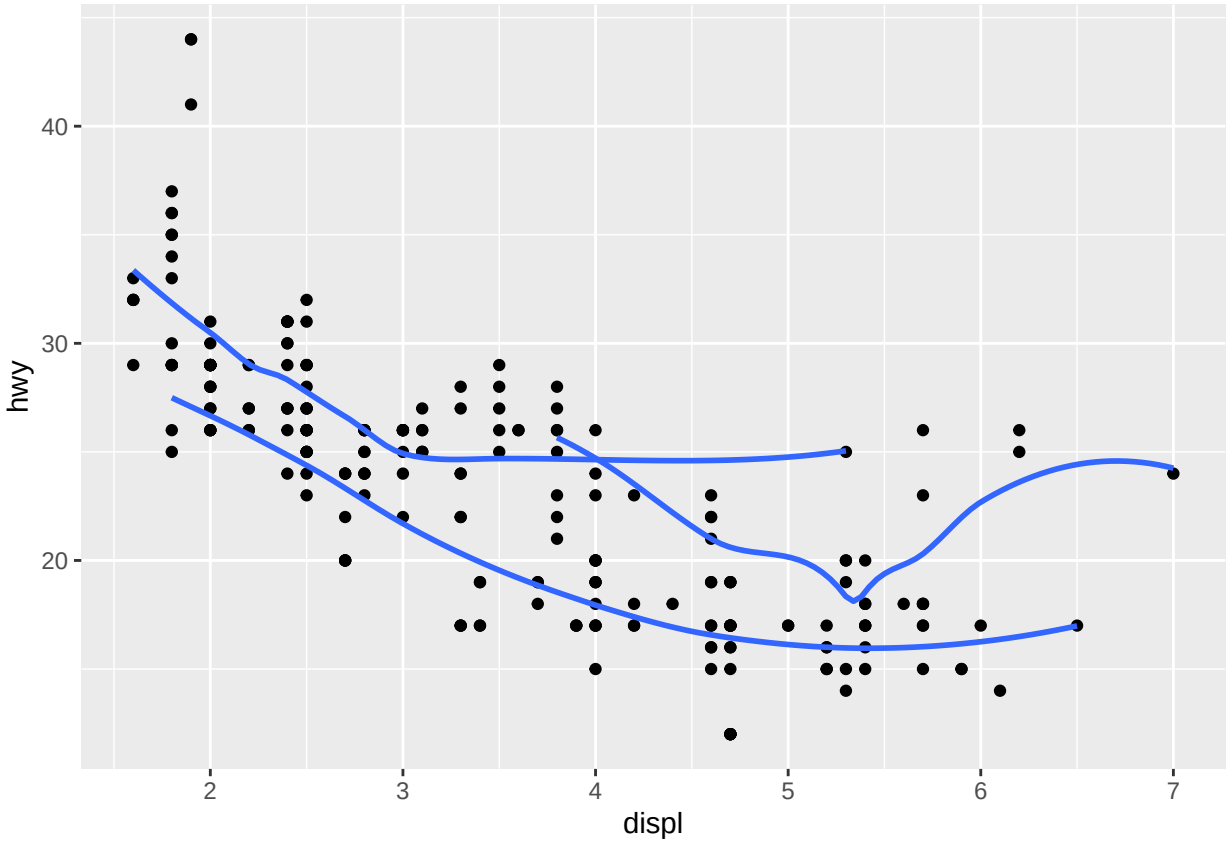


Gráfico 3:

```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy, color=drv)) +
  geom_point() +
  geom_smooth(se=FALSE, mapping=aes(group=drv))
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

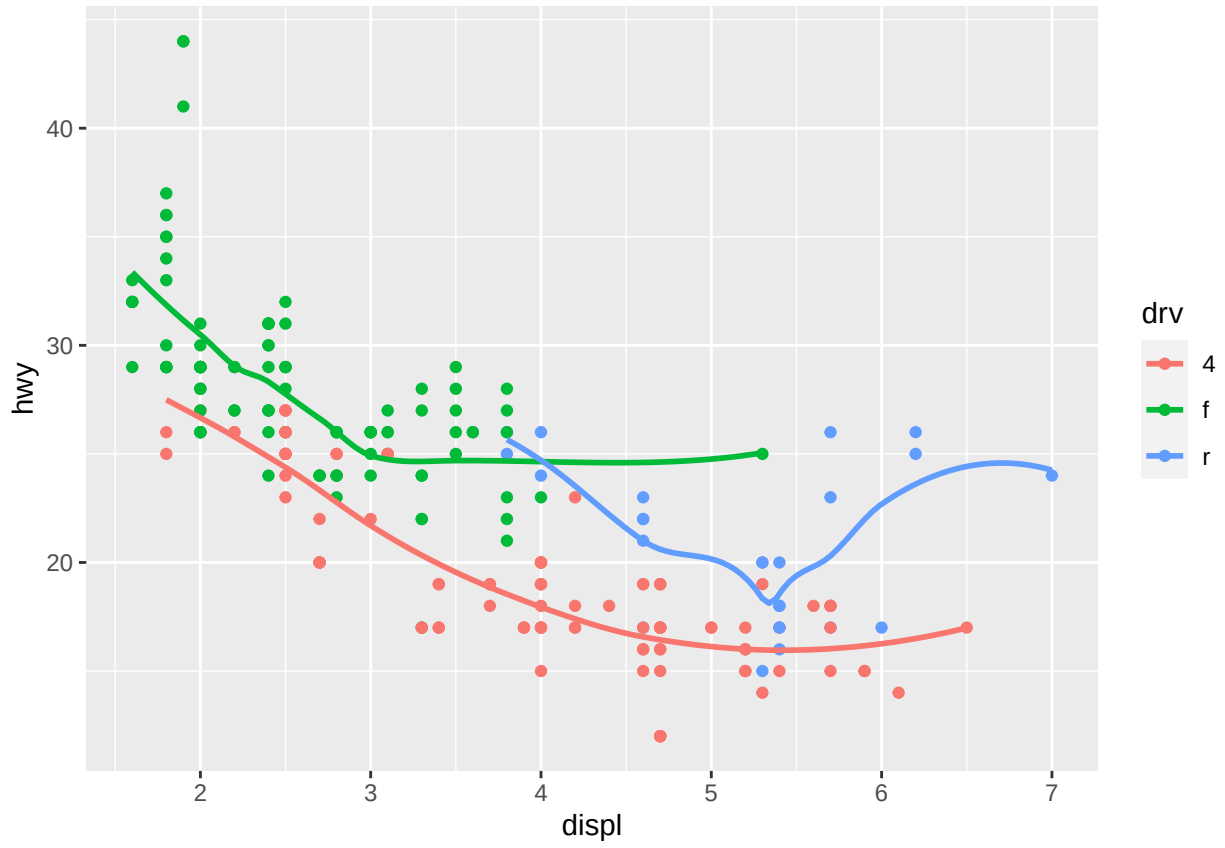


Gráfico 4:

```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_point(mapping=aes(color=drv)) +
  geom_smooth(se=FALSE)
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

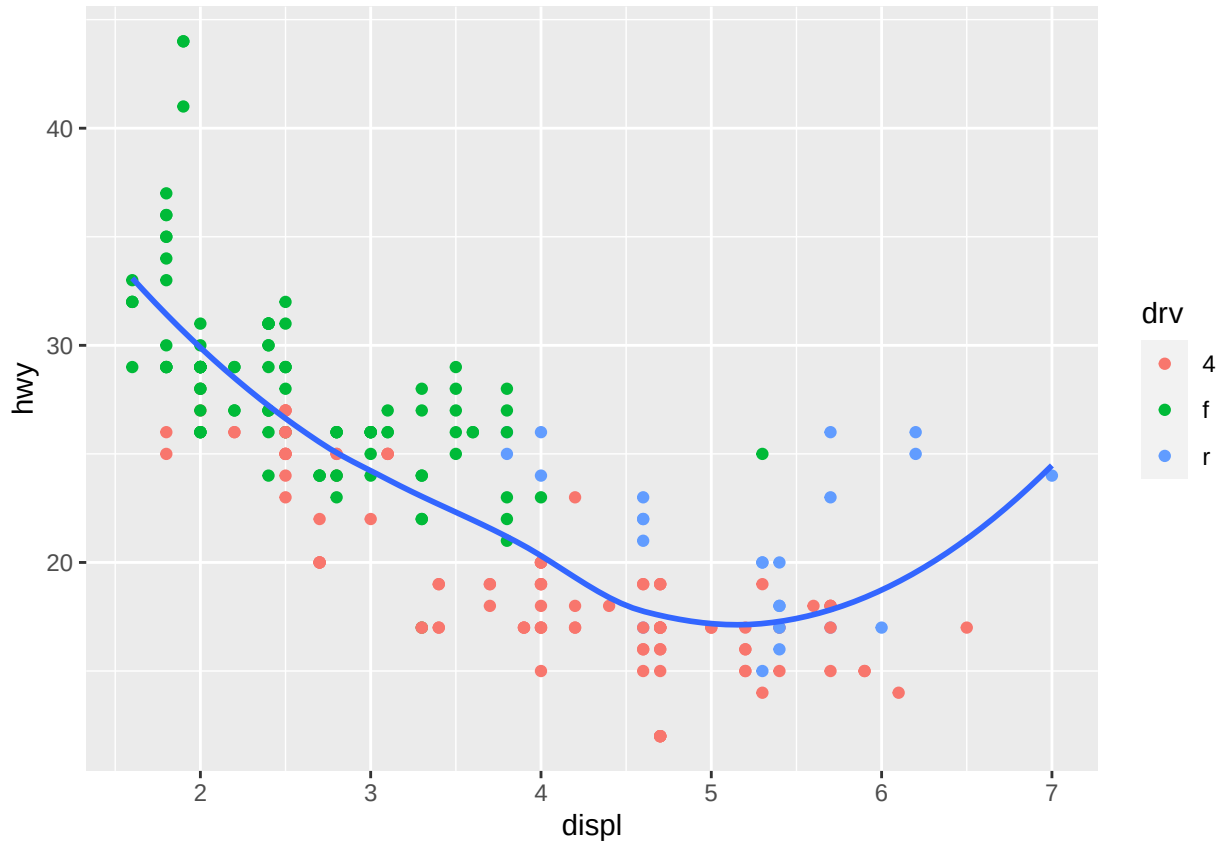


Gráfico 5:

```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy, color=drv)) +
  geom_point() +
  geom_smooth(se=FALSE, mapping=aes(group=drv, linetype=drv))
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

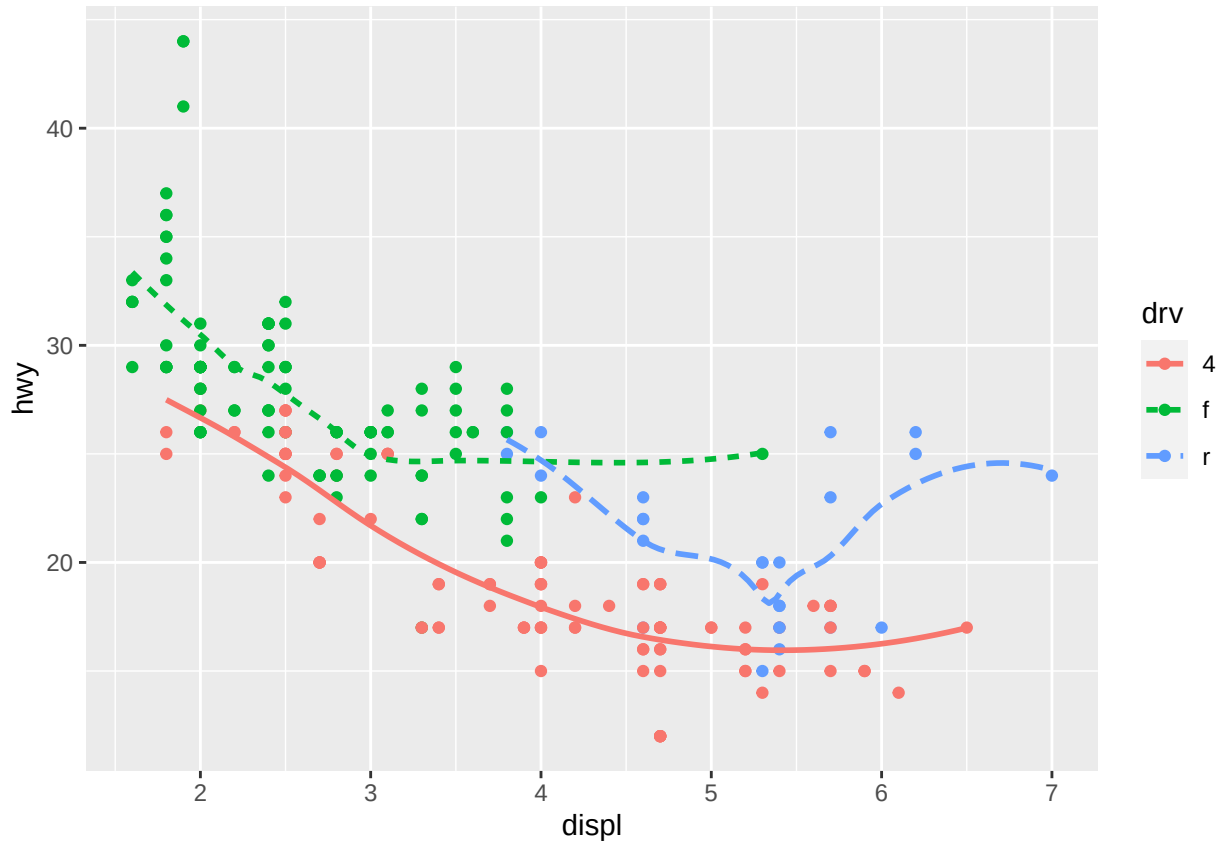
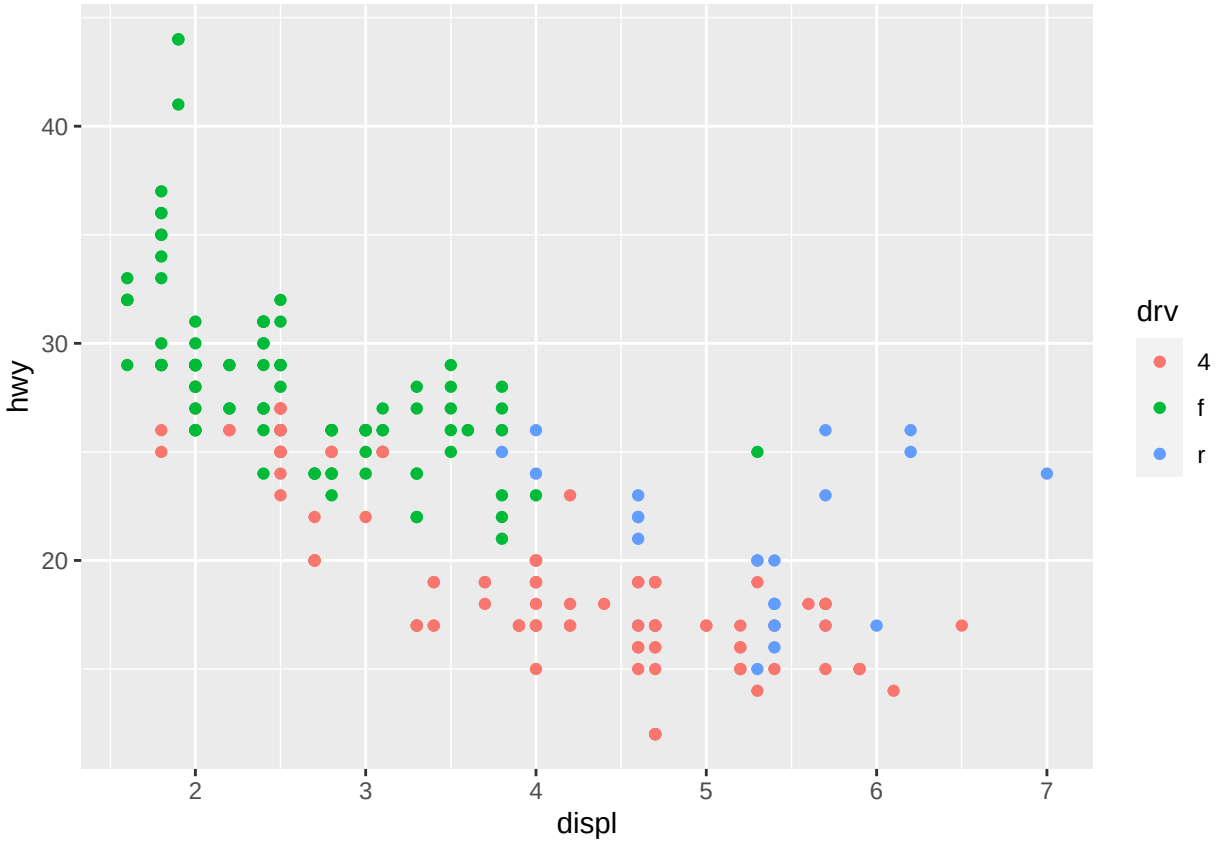


Gráfico 6:

```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_point(mapping=aes(color=drv))
```



- Haz el ejercicio 1 de la Sección 5.2.4 de R4DS.

```
library(nycflights13)
view(flights)
```

Find all flights that: - Had an arrival delay of two or more hours

```
flights %>%
  filter(arr_delay >= 120) %>%
  select(time_hour, flight, origin, dest, arr_delay)
```

```
## # A tibble: 10,200 x 5
##    time_hour           flight origin dest  arr_delay
##    <dttm>               <int> <chr>  <chr>     <dbl>
##  1 2013-01-01 06:00:00   4576 LGA    CLT         137
##  2 2013-01-01 18:00:00   3944 JFK    BWI         851
##  3 2013-01-01 07:00:00    856 EWR    BOS         123
##  4 2013-01-01 09:00:00   1086 LGA    IAH         145
##  5 2013-01-01 13:00:00   4497 EWR    RIC         127
##  6 2013-01-01 13:00:00    525 EWR    MCO         125
##  7 2013-01-01 14:00:00   4181 EWR    MCI         136
##  8 2013-01-01 13:00:00   5712 JFK    IAD         123
##  9 2013-01-01 16:00:00   4092 EWR    DAY         123
## 10 2013-01-01 16:00:00   4622 LGA    BNA         138
## # ... with 10,190 more rows
```

- Flew to Houston (IAH or HOU)

```
flights %>%
  filter(dest == "IAH" | dest == "HOU" ) %>%
  select(time_hour, flight, origin, dest)
```

```
## # A tibble: 9,313 x 4
##    time_hour           flight origin dest
##    <dttm>               <int> <chr>  <chr>
##  1 2013-01-01 05:00:00   1545 EWR    IAH
##  2 2013-01-01 05:00:00   1714 LGA    IAH
##  3 2013-01-01 06:00:00    496 LGA    IAH
##  4 2013-01-01 07:00:00    473 LGA    IAH
##  5 2013-01-01 07:00:00   1479 EWR    IAH
##  6 2013-01-01 09:00:00   1220 EWR    IAH
##  7 2013-01-01 10:00:00   1004 LGA    IAH
##  8 2013-01-01 10:00:00    455 EWR    IAH
##  9 2013-01-01 09:00:00   1086 LGA    IAH
## 10 2013-01-01 12:00:00   1461 EWR    IAH
## # ... with 9,303 more rows
```

- Were operated by United, American, or Delta

```
flights %>%
  filter(carrier == "UA" | carrier == "AA" | carrier == "DL")%>%
  select(time_hour, flight, origin, dest, carrier)
```

```
## # A tibble: 139,504 x 5
##    time_hour           flight origin dest  carrier
##    <dttm>               <int> <chr>  <chr> <chr>
##  1 2013-01-01 05:00:00   1545 EWR    IAH   UA
##  2 2013-01-01 05:00:00   1714 LGA    IAH   UA
##  3 2013-01-01 05:00:00   1141 JFK    MIA   AA
##  4 2013-01-01 06:00:00    461 LGA    ATL   DL
##  5 2013-01-01 05:00:00   1696 EWR    ORD   UA
##  6 2013-01-01 06:00:00    301 LGA    ORD   AA
##  7 2013-01-01 06:00:00    194 JFK    LAX   UA
##  8 2013-01-01 06:00:00   1124 EWR    SFO   UA
##  9 2013-01-01 06:00:00    707 LGA    DFW   AA
## 10 2013-01-01 06:00:00   1187 EWR    LAS   UA
## # ... with 139,494 more rows
```

- Departed in summer (July, August, and September)

```
flights %>%
  filter(month == 7 | month == 8 | month == 9) %>%
  select(time_hour, flight, origin, dest, month)
```

```
## # A tibble: 86,326 x 5
##    time_hour           flight origin dest  month
##    <dttm>               <int> <chr>  <chr> <int>
```

```
##  1 2013-07-01 20:00:00    915 JFK    SFO       7
##  2 2013-07-01 23:00:00   1503 JFK    SJU       7
##  3 2013-07-01 22:00:00    234 JFK    BTV       7
##  4 2013-07-01 21:00:00   1371 LGA    FLL       7
##  5 2013-07-01 21:00:00    185 JFK    LAX       7
##  6 2013-07-01 20:00:00    165 JFK    PDX       7
##  7 2013-07-01 20:00:00    415 JFK    LAX       7
##  8 2013-07-01 21:00:00    425 JFK    TPA       7
##  9 2013-07-01 21:00:00   1183 JFK    MCO       7
## 10 2013-07-01 22:00:00    623 JFK    LAX       7
## # ... with 86,316 more rows
```

- Arrived more than two hours late, but didn't leave late

```
flights %>%
  filter(arr_delay >= 120 & dep_delay <= 0) %>%
  select(time_hour, flight, origin, dest, dep_delay, arr_delay)
```

```
## # A tibble: 29 x 6
##    time_hour           flight origin dest  dep_delay arr_delay
##    <dttm>               <int> <chr>  <chr>     <dbl>     <dbl>
##  1 2013-01-27 14:00:00   3728 EWR    ORD          -1       124
##  2 2013-10-07 13:00:00   5181 LGA    MSN           0       130
##  3 2013-10-07 13:00:00   1151 LGA    DFW          -2       124
##  4 2013-10-16 07:00:00      3 JFK    SJU          -3       122
##  5 2013-11-01 07:00:00    399 JFK    LAX          -2       194
##  6 2013-03-18 18:00:00    389 JFK    SFO          -3       140
##  7 2013-04-17 16:00:00   4540 LGA    DTW          -5       124
##  8 2013-04-18 06:00:00    707 LGA    DFW          -2       179
##  9 2013-04-18 07:00:00   2083 EWR    DFW          -5       143
## 10 2013-05-22 18:00:00   4674 LGA    CLE          -3       127
## # ... with 19 more rows
```

- 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) %>%
  select(time_hour, flight, origin, dest, dep_delay, arr_delay)
```

```
## # A tibble: 1,819 x 6
##    time_hour           flight origin dest  dep_delay arr_delay
##    <dttm>               <int> <chr>  <chr>     <dbl>     <dbl>
##  1 2013-01-01 17:00:00   1999 EWR    MIA         285       246
##  2 2013-01-01 21:00:00    199 JFK    LAS         116        73
##  3 2013-01-03 12:00:00    551 EWR    SFO         162       128
##  4 2013-01-03 17:00:00    575 JFK    EGE          99        66
##  5 2013-01-03 17:00:00    177 JFK    SFO          65        28
##  6 2013-01-03 17:00:00    979 EWR    PHX         102        67
##  7 2013-01-03 18:00:00     91 JFK    OAK          65         1
##  8 2013-01-03 20:00:00   3439 JFK    CVG         177       141
##  9 2013-01-04 17:00:00    575 JFK    EGE         137       105
## 10 2013-01-04 17:00:00    177 JFK    SFO         145        97
## # ... with 1,809 more rows
```

- Departed between midnight and 6am (inclusive)

```
flights %>%
  filter(hour >= 0 & hour <= 6) %>%
  select(time_hour, flight, origin, dest, hour)
```

```
## # A tibble: 27,905 x 5
##    time_hour           flight origin dest   hour
##    <dttm>               <int> <chr>  <chr> <dbl>
##  1 2013-01-01 05:00:00   1545 EWR    IAH       5
##  2 2013-01-01 05:00:00   1714 LGA    IAH       5
##  3 2013-01-01 05:00:00   1141 JFK    MIA       5
##  4 2013-01-01 05:00:00    725 JFK    BQN       5
##  5 2013-01-01 06:00:00    461 LGA    ATL       6
##  6 2013-01-01 05:00:00   1696 EWR    ORD       5
##  7 2013-01-01 06:00:00    507 EWR    FLL       6
##  8 2013-01-01 06:00:00   5708 LGA    IAD       6
##  9 2013-01-01 06:00:00     79 JFK    MCO       6
## 10 2013-01-01 06:00:00    301 LGA    ORD       6
## # ... with 27,895 more rows
```