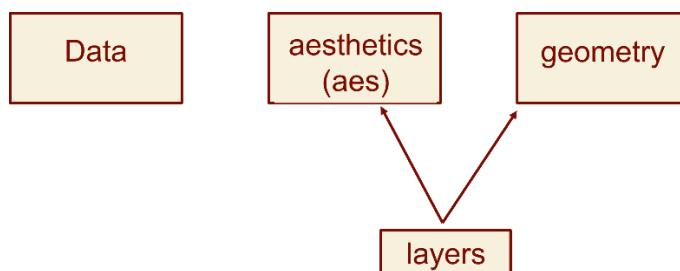


SEMINARI 1. *R / ggplot*. Introducció (Respostes)

1. OBJECTIUS

Aquest seminari serveix per familiaritzar-se amb l'ús de *ggplot2* i els seus passos successius.



Si no l'heu instal·lat encara, instal·leu i carregueu la llibreria tidyverse.

```
> install.packages("tidyverse")
> library(tidyverse)
```

NOTA: En aquest seminari, l'únic objectiu és familiaritzar-se amb l'ús i l'estructura de *ggplot* i veure les diferents “point shapes” segons el tipus de dades que tenim. Treballarem només amb `geom_point()`. Per tant, les visualitzacions que farem en aquest seminari NO seran les més adequades pel tipus de dades, però això ho anirem veient amb els següents seminaris, on, un cop ja familiaritzats amb les eines, sí que farem un especial èmfasis en aquest segon aspecte.

2. PART 1. Com és el nostre dataset? Quin tipus de variables hi tenim?

El conjunt de dades *mtcars* conté informació de 32 cotxes. És un conjunt de dades petit que conté una varietat de variables contínues i categòriques i ens permetrà familiaritzar-nos amb *ggplot2*. Podeu utilitzar `str()` per explorar la estructura d'aquest dataset.

Si obriu R de nou, primer de tot recordeu que heu de tornar a carregar la llibreria tidyverse.

```
> library (tidyverse)
```

Podeu utilitzar `str()` per explorar la estructura d'aquest dataset.

```
> str(mtcars)
'data.frame': 32 obs. of 11 variables:
 $ mpg : num 21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
 $ cyl : num 6 6 4 6 8 6 8 4 4 6 ...
 $ disp: num 160 160 108 258 360 ...
 $ hp : num 110 110 93 110 175 105 245 62 95 123 ...
 $ drat: num 3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
 $ wt : num 2.62 2.88 2.32 3.21 3.44 ...
 $ qsec: num 16.5 17 18.6 19.4 17 ...
 $ vs : num 0 0 1 1 0 1 0 1 1 1 ...
 $ am : num 1 1 1 0 0 0 0 0 0 0 ...
 $ gear: num 4 4 4 3 3 3 3 4 4 4 ...
 $ carb: num 4 4 1 1 2 1 4 2 2 4 ...
```

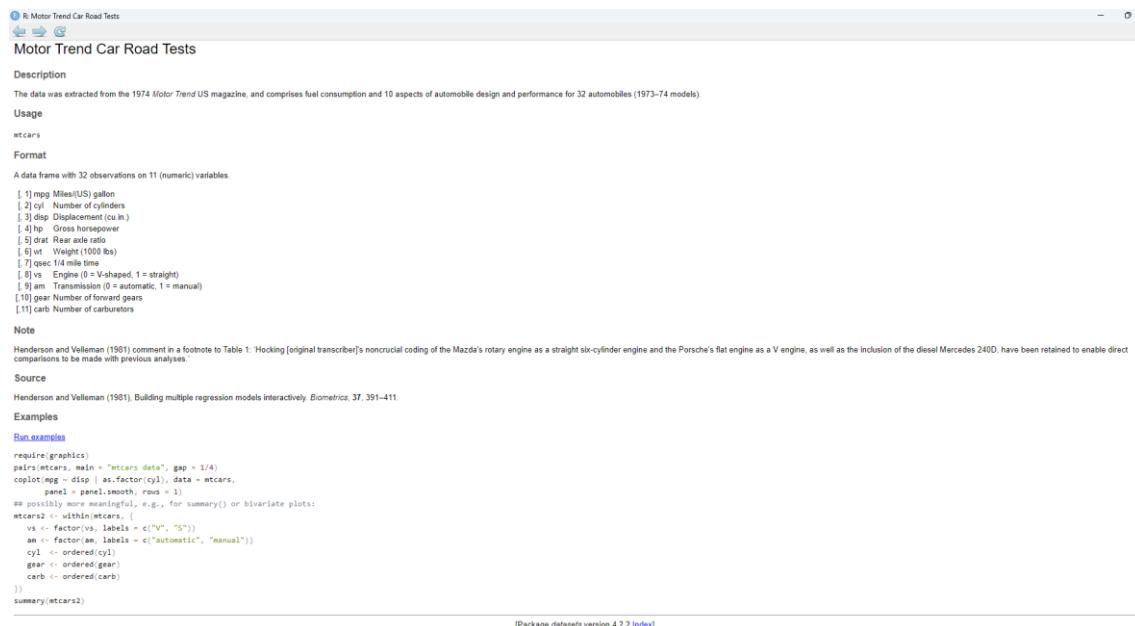
I per saber la definició de cada variable utilitzeu

```
> ?mtcars
```

O :

```
> help (mtcars)
```

Se us obrirà una pantalla nova:



R: Motor Trend Car Road Tests

Motor Trend Car Road Tests

Description

The data was extracted from the 1974 Motor Trend US magazine, and comprises fuel consumption and 10 aspects of automobile design and performance for 32 automobiles (1973-74 models).

Usage

```
mtcars
```

Format

Data frame with 32 observations on 11 (numeric) variables.

[1] mpg Miles/(US) gallon
[2] cyl Number of cylinders
[3] disp Displacement (cu.in.)
[4] hp Gross horsepower
[5] drat Rear axle ratio
[6] wt Weight (1000 lbs)
[7] qsec 1/4 mile time
[8] vs Engine (0 = V-shaped, 1 = straight)
[9] am Transmission (0 = automatic, 1 = manual)
[10] gear Number of forward gears
[11] carb Number of carburetors

Note

Henderson and Velleman (1981) comment in a footnote to Table 1: Hocking [original transcriber]'s noncrucial coding of the Mazda's rotary engine as a straight six-cylinder engine and the Porsche's flat engine as a V engine, as well as the inclusion of the diesel Mercedes 240D, have been retained to enable direct comparisons to be made with previous analyses.'

Source

Henderson and Velleman (1981), Building multiple regression models interactively. *Biometrika*, 68, 391-411.

Examples

Run examples

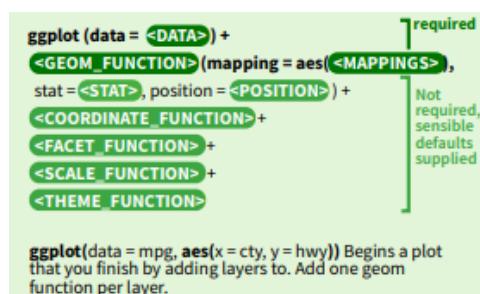
```
require(graphics)
pairwise.mtcars, main = "mtcars data", gap = 1/4)
coplot(mpg ~ disp | as.factor(cyl), data = mtcars,
       panel = panel.smooth, rows = 1)
## possibly more meaningful, e.g., for summary() or bivariate plots:
mtcars2 <- within(mtcars, {
  vs <- factor(vs, labels = c("V", "S"))
  am <- factor(am, labels = c("automatic", "manual"))
  cyl <- ordered(cyl)
  gear <- ordered(gear)
  carb <- ordered(carb)
})
summary(mtcars2)
```

[Package datasets version 4.2.2 [index](#)]

EXERCICIS:

1.- Utilitzeu *ggplot* per dibuixar una gràfica on l'eix x correspongi a la variable 'cyl' (cilindres) i l'eix y a la variable 'mpg' (km de galó). Utilitzeu *geom_point()*.

Hem vist a classe que



Per tant posarem:

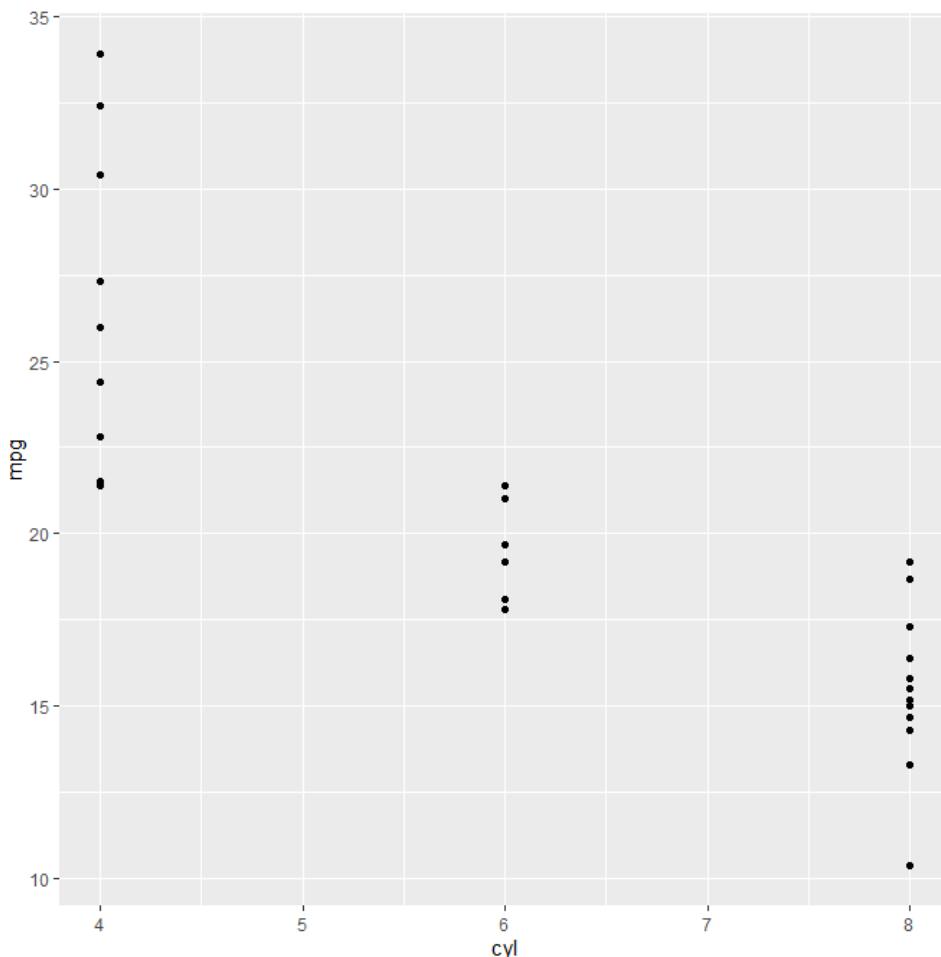
```
> ggplot(data=mtcars, aes(x=cyl, y=mpg))+geom_point()
```

O per simplificar:

```
> ggplot(mtcars, aes(cyl, mpg))+geom_point()
```

```
> ggplot(mtcars) + aes(cyl, mpg)+geom_point()
```

Noteu que al utilitzar `geom_point()`, el `ggplot` tracta la variable 'cyl' com una variable continua. Tenim una gràfica, on dona la impressió que existeix algun automòbil de 5 o 7 cilindres quan no és així.



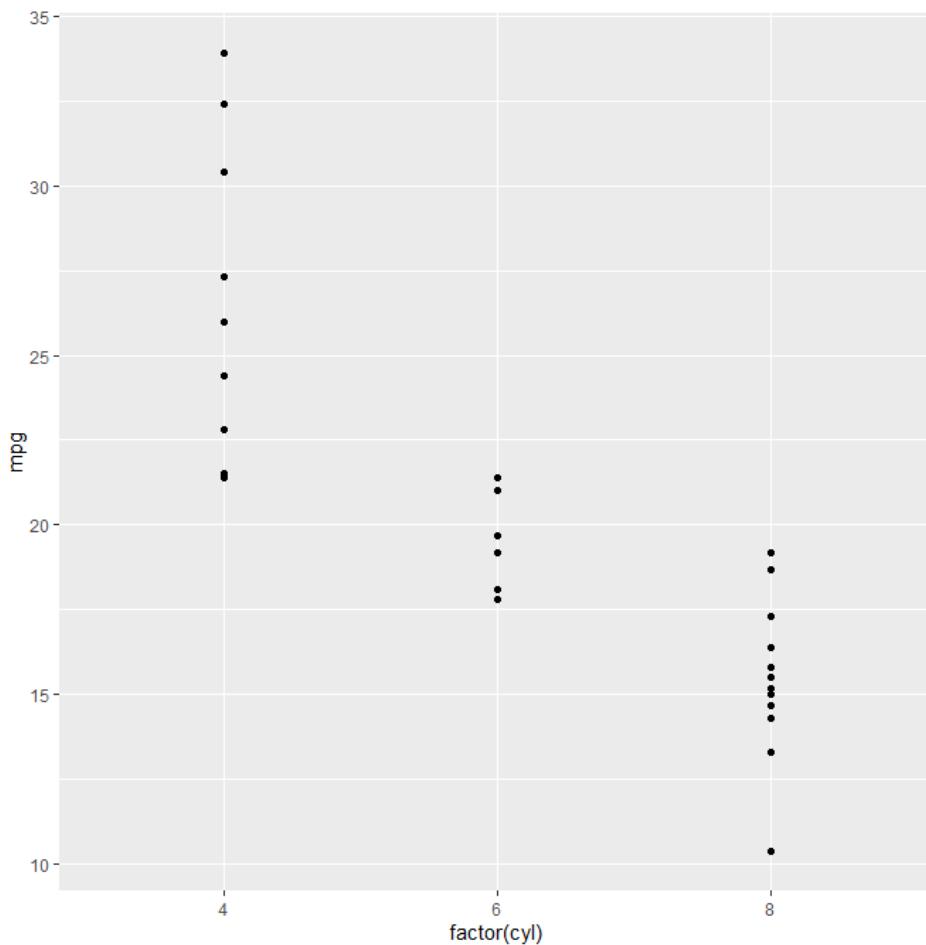
2.- Utilizeu la funció `ggplot`, però ara categoritzeu la variable 'cyl' ordinal. Per això utilitzeu la funció `factor`. Quina informació podeu extreure'n d'aquesta gràfica?

NOTA: Primer escriviu `?factor` per a que R us digui com especificar que 'cyl' és una variable ordinal que ens està diferenciant en tres grups/nivells de cotxes (els que tenen 4, 6 o 8 cilindres). És el que abans quan veiem els tipus de variables em anomenat **factors**.

Només hem de canviar `cyl` per `factor(cyl)`

```
> ggplot(mtcars, aes(factor(cyl), mpg))+geom_point()
```

```
> ggplot(mtcars)+aes(factor(cyl), mpg)+geom_point()
```



Ara l'eix x no conté valors com 5 o 7 indicant una certa continuïtat errònia de la variable, sinó només els valors que estaven al data set.

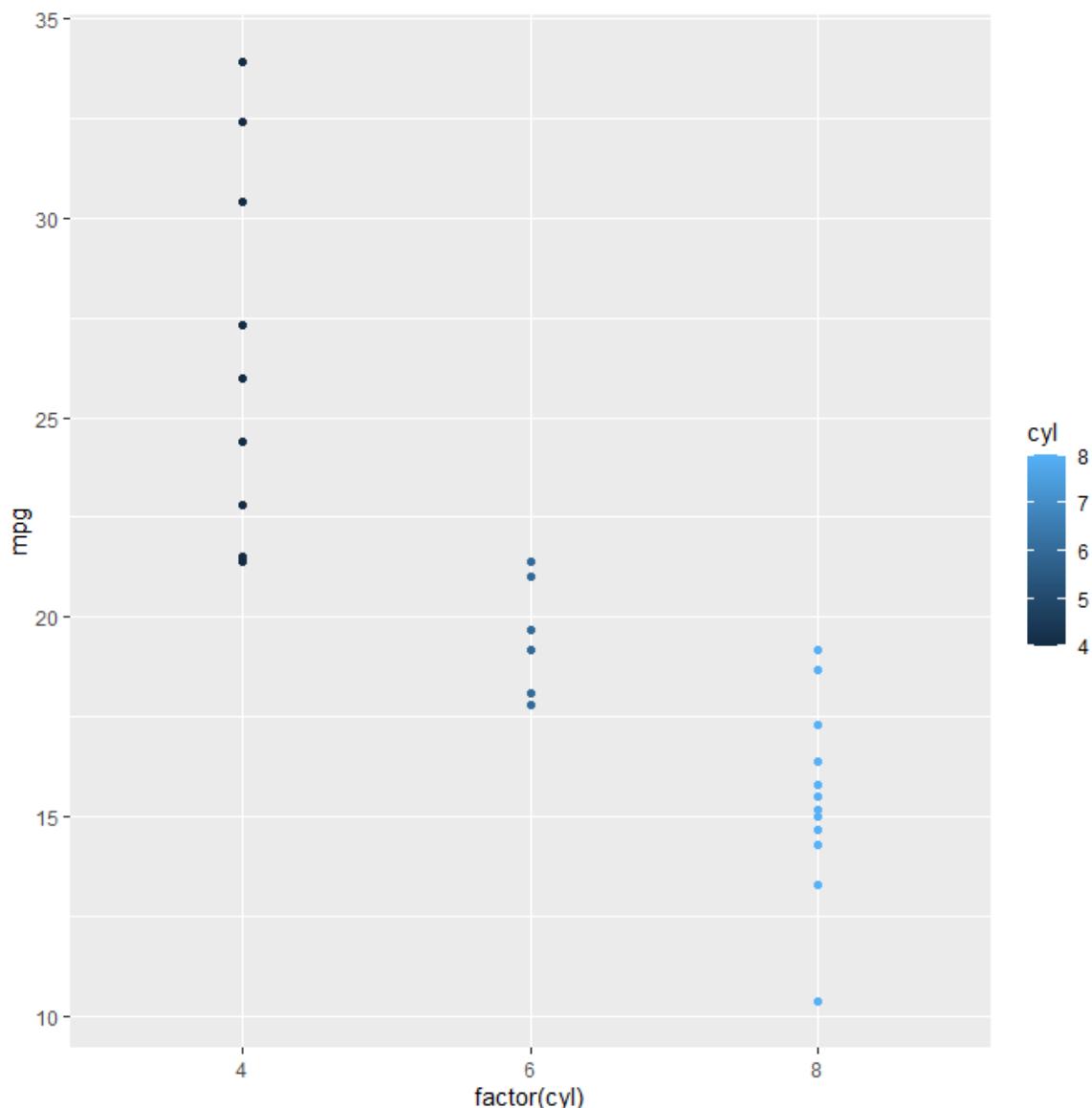
Veiem que els cotxes amb més cilindres són els que menys consumeixen, mentre que els cotxes amb més cilindres són els que més consumeixen.

3.- Afegiu un color segons els cilindres que tingui el cotxe. Ens aporta alguna informació nova? Per què?

El color en ggplot s'afegeix fent un mapeig de la propietat estètica color. Dins d'aes(), afegim un argument color igual a la variable 'cyl'. Recordeu que 'cyl' és a la base una variable numèrica i si no diem el contrari, R la tractarà com a variable continua.

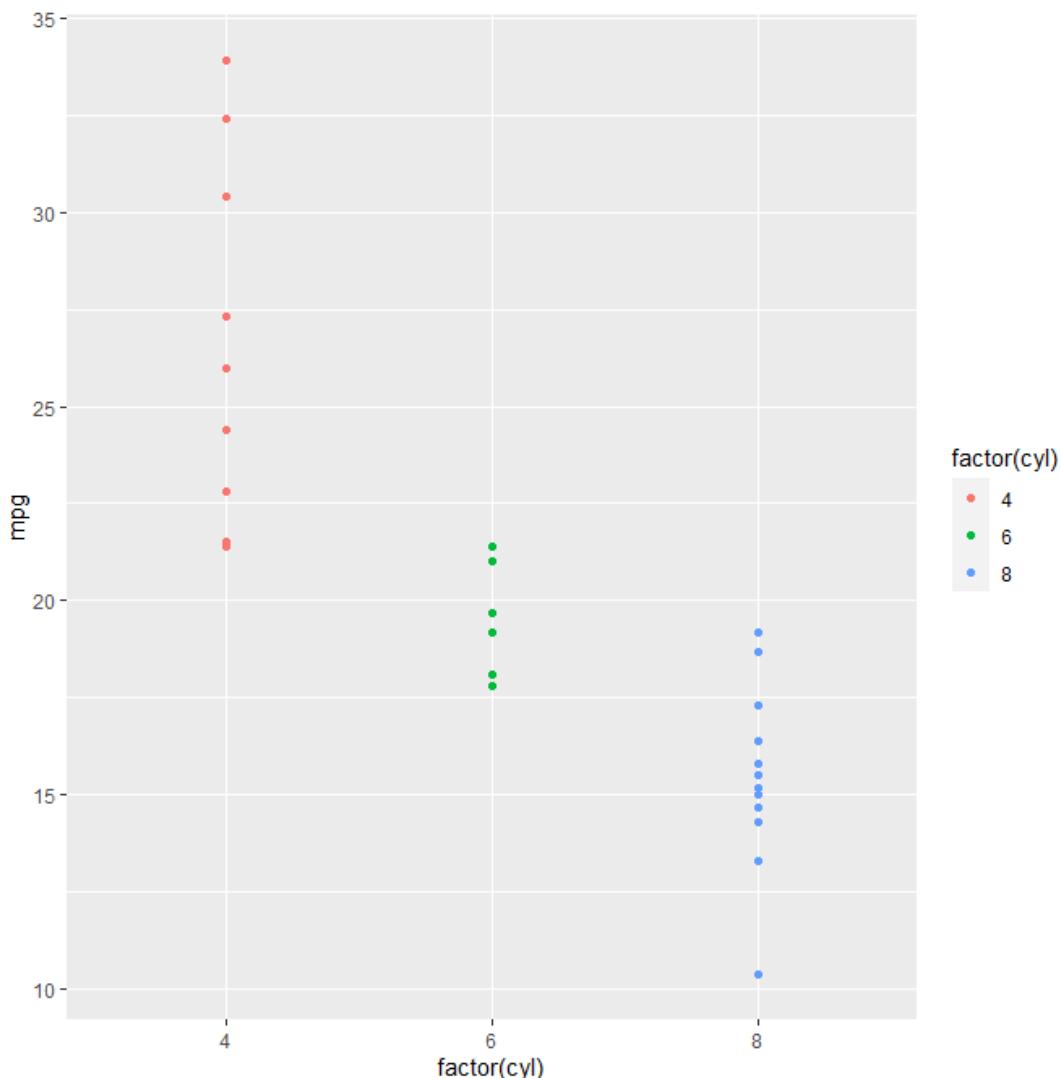
Si fem:

```
>ggplot(mtcars, aes(factor(cyl), mpg, color=cyl))+geom_point()
```



L'escala de color mostra 'cyl' com una variable continua altre cop. Per tant, especificarem que la variable 'cyl' és un factor com abans (la categoritzem). Això ens permetrà donar un color als cotxes que utilitzen 4 cilindres, diferent del color dels que n'utilitzen 6 i dels que n'utilitzen 8.

```
>ggplot(mtcars, aes(factor(cyl), mpg, color=factor(cyl)))+geom_point()
> ggplot(mtcars)+aes(factor(cyl), mpg, color=factor(cyl))+geom_point()
```



Ara bé, el color no ens ha afegit cap informació nova respecte la gràfica de l'exercici 2 on els grups ja quedaven diferenciats en l'eix x.

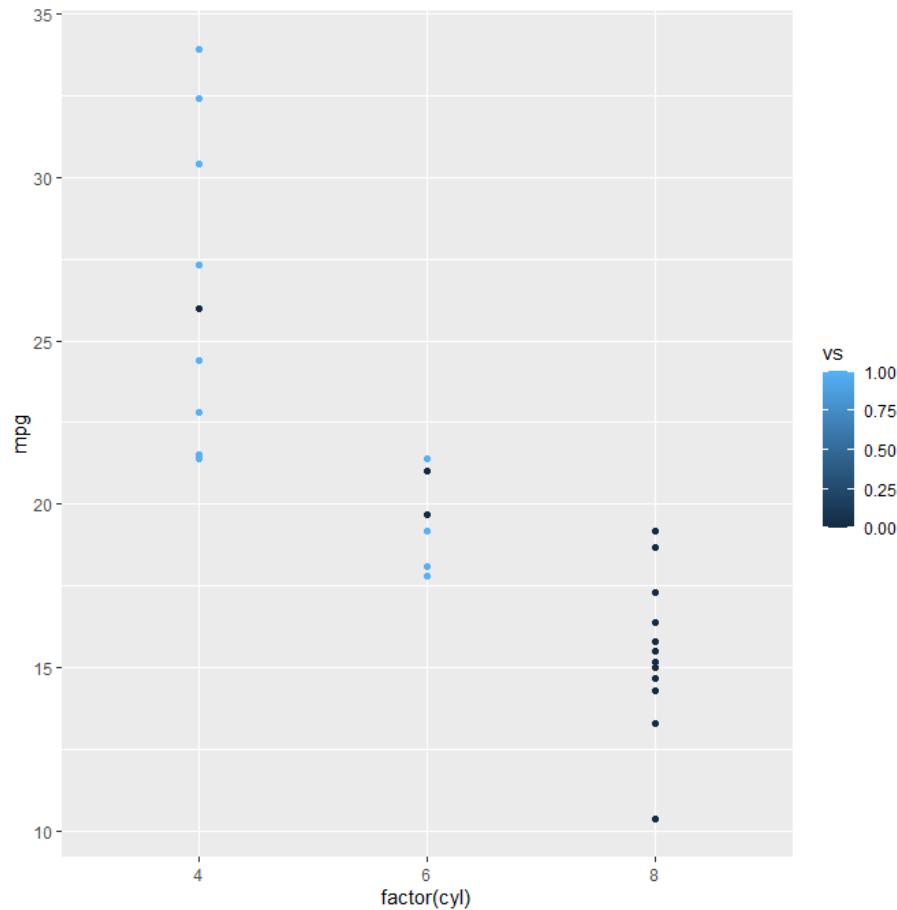
4.- Seguint amb la mateixa gràfica (on l'eix x corresponguia a la variable 'cyl' i l'eix y a la variable 'mpg'). Afegiu ara un color al motor del cotxe (engine), per això primer recordeu mirar com és la variable 'vs' i feu els ajustos necessaris. Un cop tenim la gràfica, ens aporta alguna informació nova respecte la gràfica de l'exercici 2? Per què? Quina és aquesta informació?

Utilitzeu `?scale_x_discrete` , `?scale_x_continuous` i `?scale_color_discrete` per posar el nom als eixos i a la llegenda de colors amb `scale`

Tot i que no estem mostrant la variable engine (vs), de la mateixa manera que en l'apartat anterior, dins d'aes(), podem afegir un argument color igual a la variable vs

Si no mirem com és la variable vs :

```
> ggplot(mtcars, aes(factor(cyl), mpg, color=vs))+geom_point()
> ggplot(mtcars) + aes(factor(cyl), mpg, color=vs)+geom_point()
```



Sembla que 'vs' prengui valors en una escala de 0 a 1. Però fent ?mtcars veiem que 'vs' té dos valors només que diferencien entre dos grups/nivells segons la forma del motor:

```
> ?mtcars
```

```
Files Plots Packages Help Viewer
Refresh Help Top
R: Motor Trend Car Road Tests ▾ Find in Topic



## Usage



```
mtcars
```



## Format



A data frame with 32 observations on 11 (numeric) variables.



```
[, 1] mpg Miles/(US) gallon
[, 2] cyl Number of cylinders
[, 3] disp Displacement (cu.in.)
[, 4] hp Gross horsepower
[, 5] drat Rear axle ratio
[, 6] wt Weight (1000 lbs)
[, 7] qsec 1/4 mile time
[, 8] vs Engine (0 = V-shaped, 1 = straight)
[, 9] am Transmission (0 = automatic, 1 = manual)
[,10] gear Number of forward gears
[,11] carb Number of carburetors
```

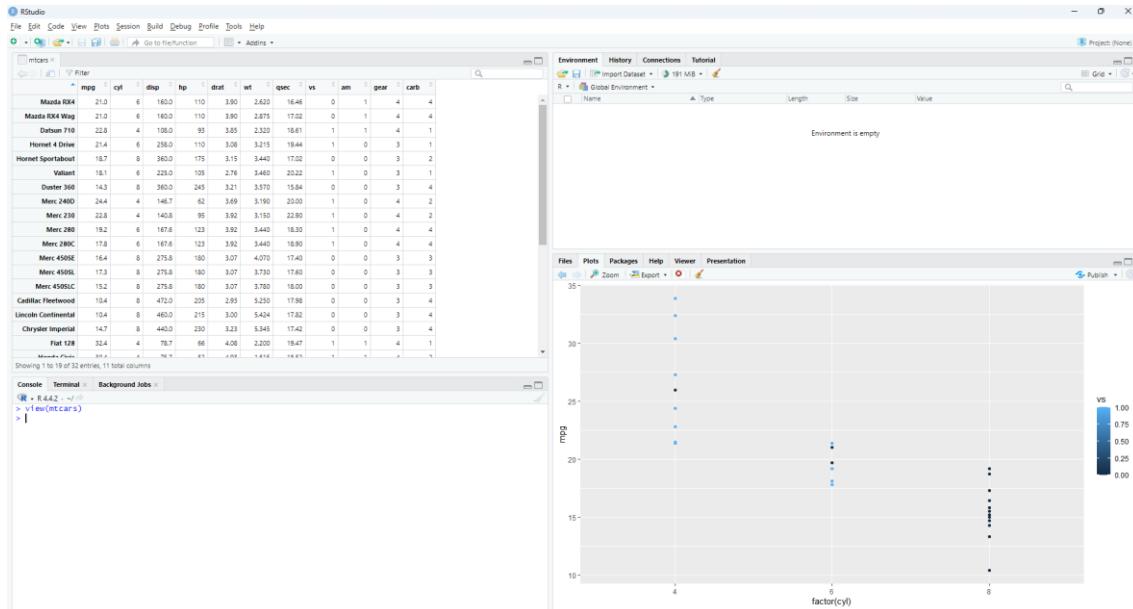


## Note


```

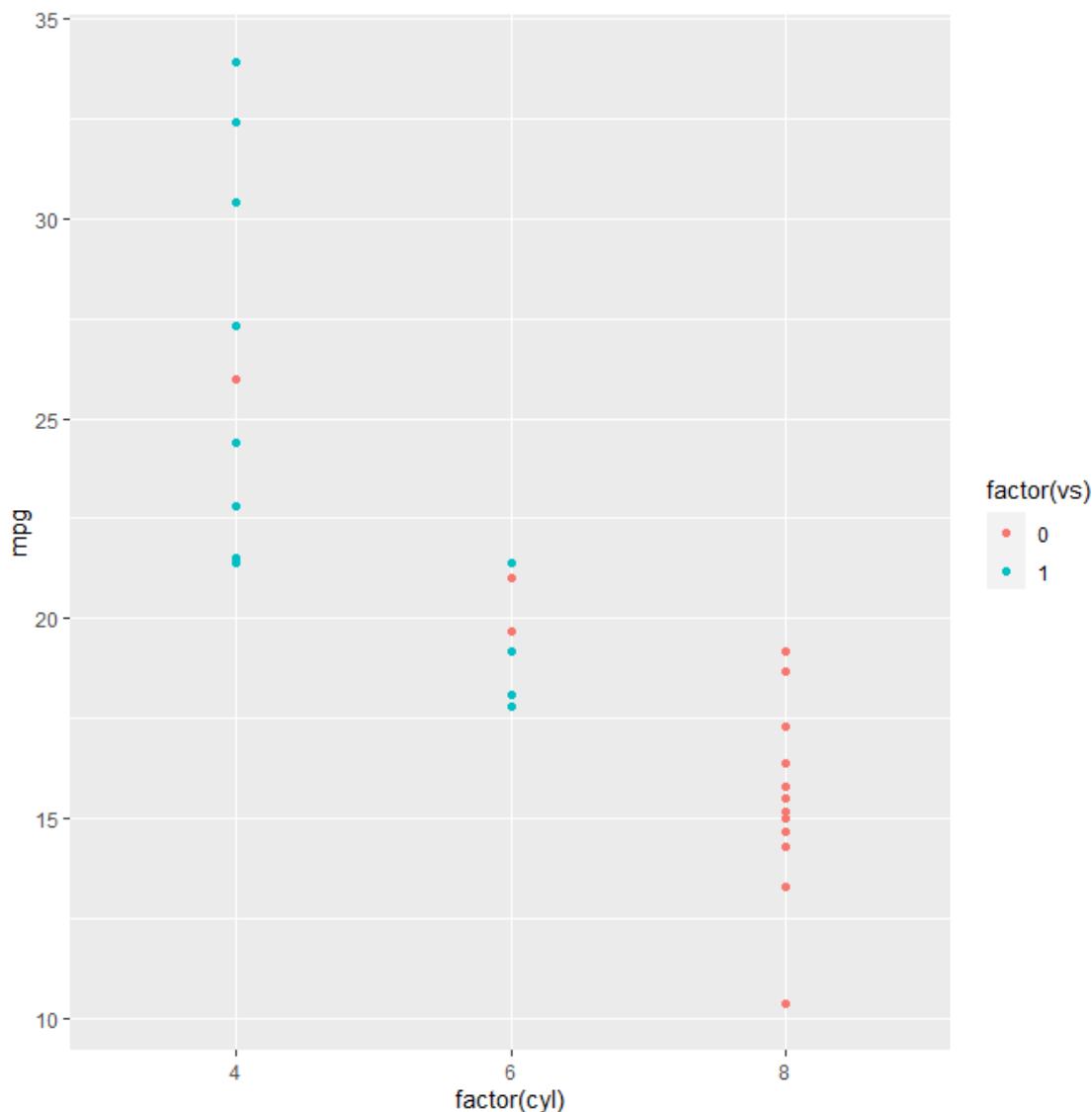
També podem fer us de 'view' per re-assegurar-nos tot visualitzant les dades:

```
> view(mtcars)
```



Per tant 'vs' s'ha de categoritzar també utilitzant factor (o convertir en la variable factor de R):

```
> ggplot(mtcars, aes(factor(cyl), mpg, color=factor(vs)))+geom_point()
```



Posem llegenda als nous colors de la variable factor 'vs':

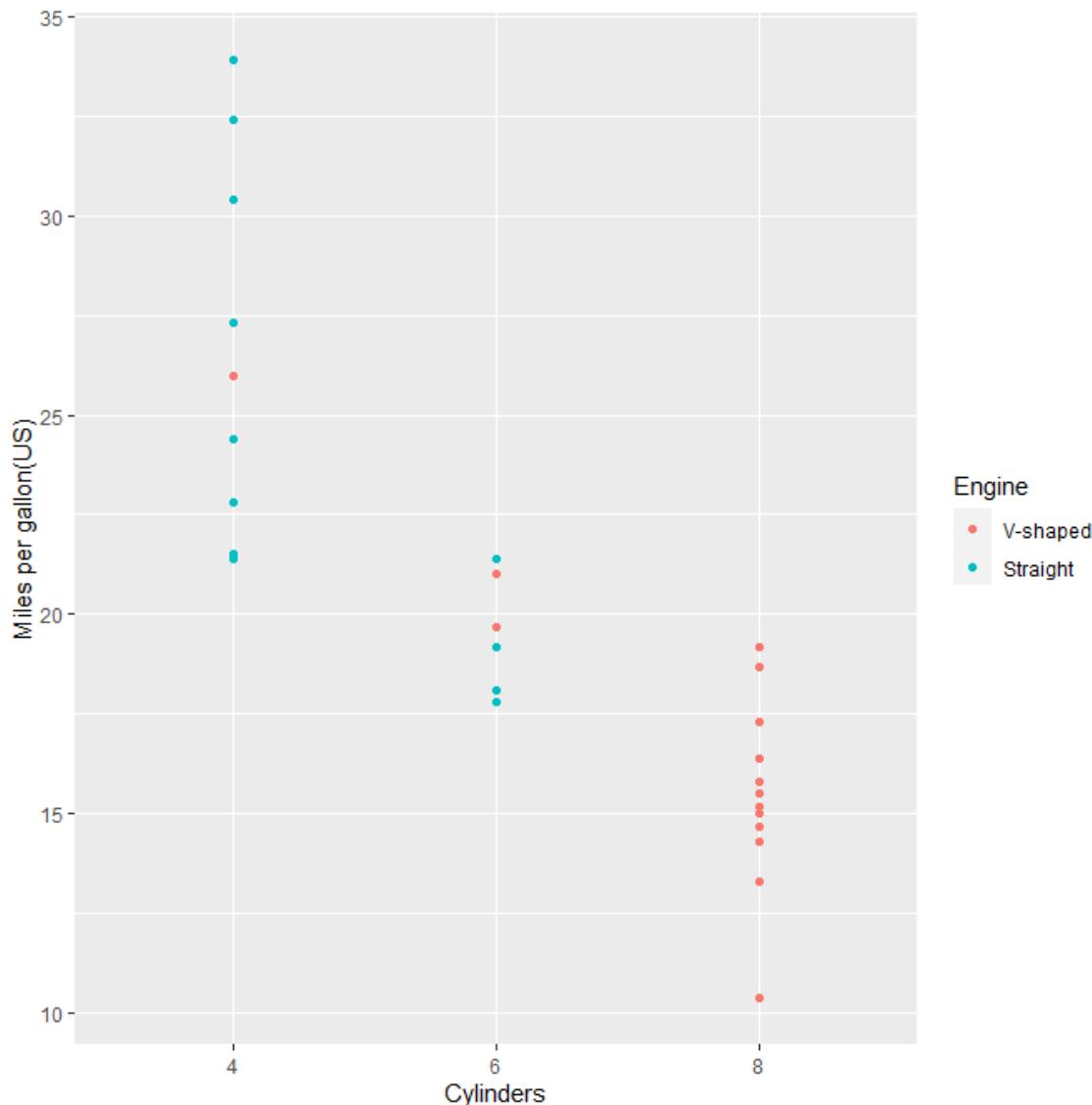
```
> ggplot(mtcars, aes(factor(cyl), mpg, color=factor(vs)))+geom_point()+
  scale_color_discrete("Engine", labels = c("V-shaped", "Straight"))
```

I posant llegenda als eixos també (COMPTE, l'eix x és discret):

```
> ggplot(mtcars, aes(factor(cyl), mpg, color=factor(vs)))+geom_point()+
  scale_x_discrete("Cylinders") + scale_y_continuous ("Miles per
  gallon(US)")+scale_color_discrete("Engine", labels = c("V-
  shaped", "Straight"))
```

O el que és el mateix:

```
> ggplot(mtcars, aes(factor(cyl), mpg, color=factor(vs)))+geom_point()+
  scale_x_discrete("Cylinders") + scale_y_continuous ("Miles per
  gallon(US)")+scale_colour_hue("Engine", labels = c("V-
  shaped", "Straight"))
```



Aquí el color sí que ens aporta informació, doncs gràcies al color podem afegir en el mateix gràfic de l'exercici 2 la informació d'un nou factor (nova variable discreta amb dos valors segons la forma del motor). Veiem per exemple que tots els cotxes de 8 cilindres tenen el motor en forma de V.

NOTA: Ara bé, com hem dit al principi, avui hem usat `geom_point()` per simplicitat i per tal de familiaritzar-nos amb ggplot, però en el seminari 2 veurem (amb més profunditat) que *quan dues de les tres variables són qualitatives* (com és el cas de cilindres i motor), *hi ha altres tipus de gràfiques que ens aporten més informació*. Anem-hi pensant amb el que fem a classe de teoria.

D'altra banda, aquest exercici estava fet una mica 'manualment'. Podríeu fer un segon *dataframe* on tot estigués categoritzat i fer directament el dibuix

```
>mtcars2 <- within(mtcars, {
  vs <- factor(vs, labels = c("V", "S"))
  cyl <- factor(cyl)
})
```

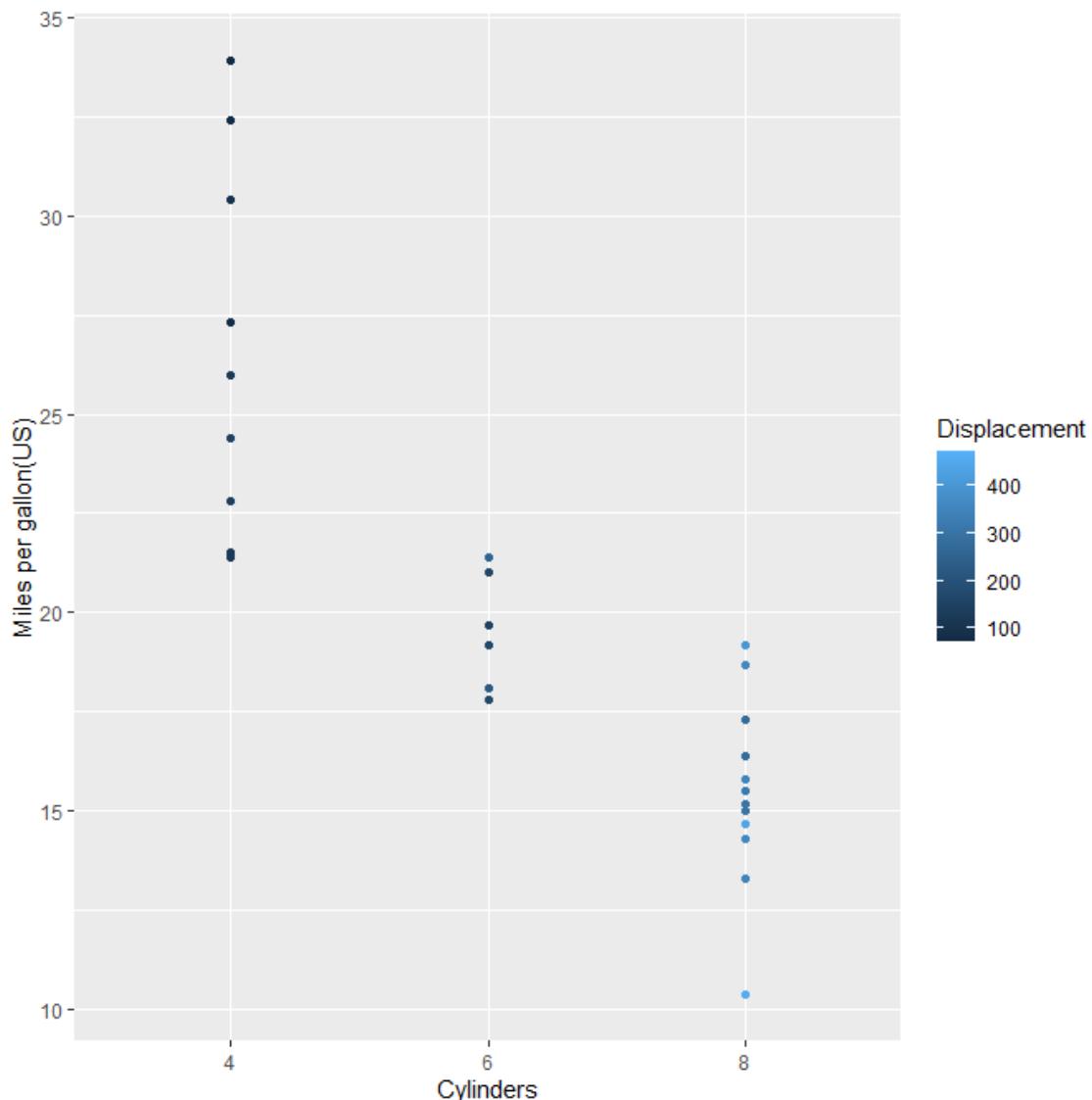
```
> ggplot(mtcars2, aes(cyl, mpg, color=vs))+geom_point()
```

A l'haver fet l'assignació `vs <- factor(vs, labels = c("V", "S"))` al nou dataframe ja no hem de pensar a què es refereixen els valors 0 o 1 de la variable vs.

5.- Afegiu ara un color a la variable *Displacement* de cada cotxe i poseu les llegendes adients. És fàcil de veure el que ens aporta aquesta nova informació? Per què? Podeu millorar la visualització de la gràfica d'una manera simple? Quina informació diríeu que en podeu extreure al veure les dades gràficament?

Tot es pot fer molt semblant als apartats anteriors, tenint en compte però que a l'hora de posar el títol a l'escala de color em d'especificar que la variable desplaçament és contínua:

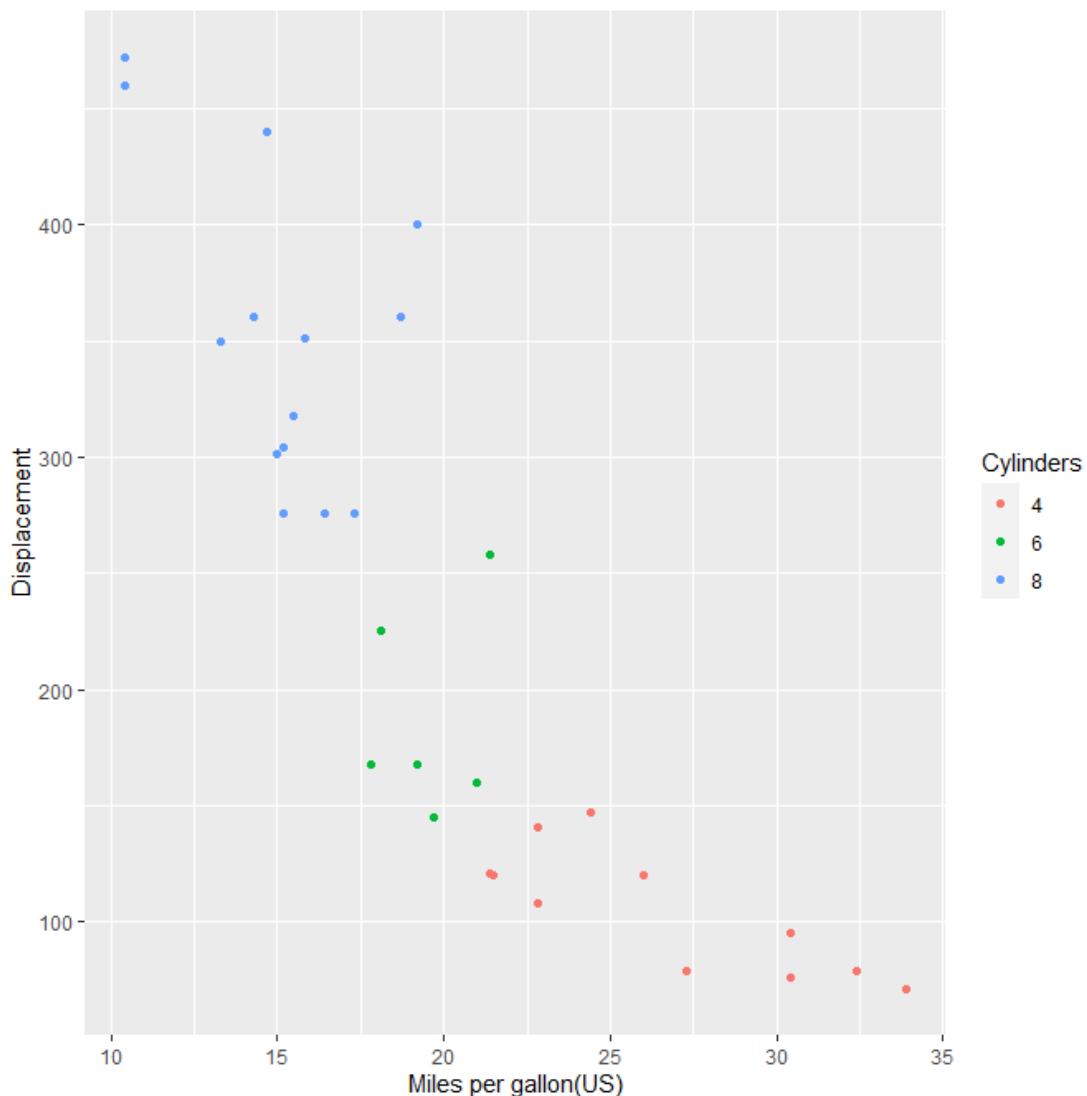
```
> ggplot(mtcars, aes(factor(cyl), mpg, color=disp))+geom_point()+
  scale_x_discrete("Cylinders") + scale_y_continuous("Miles per
gallon(US)") + scale_color_continuous("Displacement")
```



NOTA: Aquí la variable que afegim és contínua i l'escala de color ha de ser contínua també. Però la veritat es que al visualitzar una escala contínua per aquestes dades, ens costa diferenciar el valor desplaçament en cada punt. Sembla que el color en un gràfic de punts com aquest és més indicat per diferenciar entre variables discretes com en l'exercici 4, però veureu més d'això més endavant (a teoria – en la classe de color- i als seminaris).

Quan volem visualitzar dues variables quantitatives contínues i una variable qualitativa discreta com és aquest cas, **és més adient que l'eix x i y continguin la informació de les variables** contínues i el color s'afegeixi com a variable discreta (on la nostra percepció visual funcionarà millor). Per exemple, podrem extreure més informació dibuixant la següent gràfica que no pas l'anterior:

```
> ggplot(mtcars, aes(mpg, disp, color=factor(cyl)))+geom_point()+
  scale_x_continuous("Miles per gallon(US)")+
  scale_y_continuous("Displacement")+scale_color_discrete("Cylinders")
```



Aquí es veu clarament que quan més cilindres té un cotxe (més volum combinat per tant), aquest fa un desplaçament major amb menys consum. En canvi, quants menys cilindres té el cotxe, aquest necessita un major consum per molt menys desplaçament.

6.- Seguint amb la mateixa gràfica (on l'eix x correspongui a la variable 'cyl' i l'eix y a la variable 'mpg') de l'exercici 4. Intenteu utilitzar *shape* en *aes()* per posar una forma segons cada desplaçament. Què creieu que passa? Podeu utilitzar *shape* amb alguna variable? Quina per exemple?

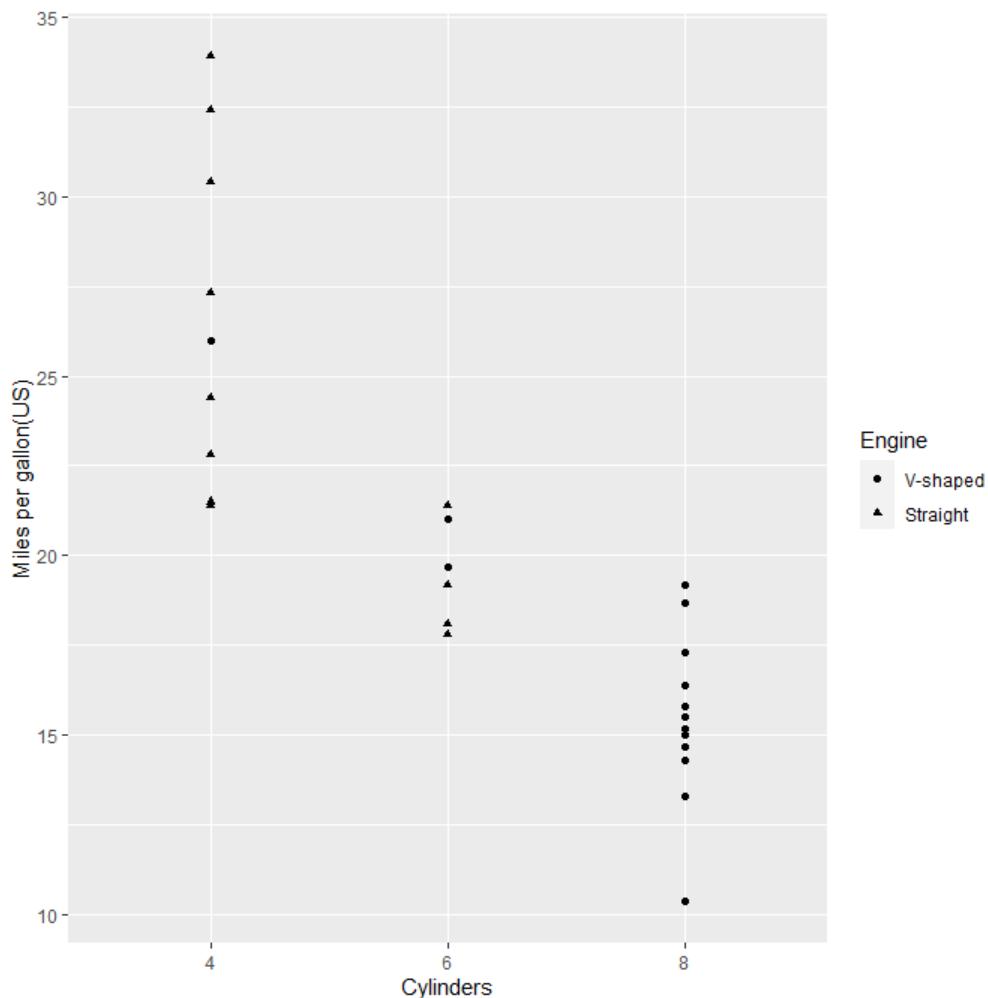
Provem amb la variable disp:

```
> ggplot(mtcars, aes(factor(cyl), mpg, shape=disp))+geom_point()
```

Obtenim un error de R, doncs *shape* només té sentit amb variables discretes i la variable disp (desplaçament) és contínua. Per tant, necessitem utilitzar *shape* amb una variable discreta.

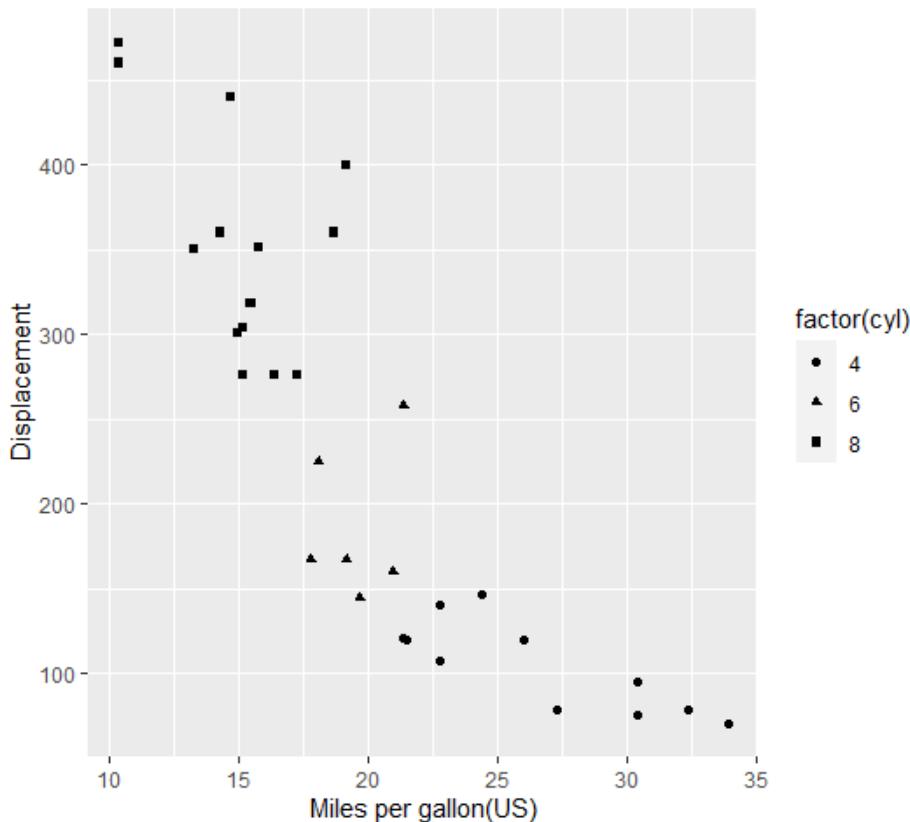
Aquí teniu un exemple com el de l'exercici 4 (tot i que ja hem introduït que *geom_point()* no era la millor forma de visualitzar una variable contínua, versus dues qualitatives) :

```
> ggplot(mtcars,aes(factor(cyl),mpg,shape=factor(vs)))+ geom_point()+
  scale_x_discrete("Cylinders")+scale_y_continuous("Miles per
  gallon(US)")+ scale_shape_discrete("Engine", labels = c("V-
  shaped","Straight"))
```



EXTRA: Tot i que no se us demana a l'enunciat, si hi penseu, un exemple útil per utilitzar *shape* seria reproduir l'exercici anterior canviant el *color* per *shape*. Per exemple, en el cas que volguéssim incloure el gràfic de l'exercici anterior en un document en blanc i negre:

```
> ggplot(mtcars, aes(mpg, disp, shape=factor(cyl)))+geom_point()+
  scale_x_continuous("Miles per gallon(US)")+
  scale_y_continuous("Displacement")+scale_color_discrete("Cylinders")
```



!! RESUM: Hem vist la importància de saber com és cada variable del dataset per tal de visualitzar-les. El mapeig `aes()` de les propietats estètiques s'anomena “escalatge” i depèn del tipus de variable. El mapeig de les variables discretes es realitza a escales diferents que el de les variables contínues. Per tant avui hem vist:

aes	Discreta	Contínua
Color (color)	Arco iris de colors	Gradient de colors
Forma (shape)	Diferent formes	NO APLICA

També podem experimentar amb altres *aesthetics* i veure per exemple que:

aes	Discreta	Contínua
Talla (size)	Escala discreta de talles	Mapeig lineal entre àrea i el valor
Transparència (alpha)	NO APLICA	Mapeig lineal a la transparència

!! Però també hem vist, gràcies a la comanda `factor()`, que quan una variable numèrica ordinal o lògica ens està diferenciant entre grups/nivells, podem categoritzar-la mitjançant `factor`. Això ens permet veure molt millor la informació que aquestes variables aporten.

Ara que hem vist com d'important és primer familiaritzar-nos amb el nostre dataframe i el tipus de variables que hi tenim per tal d'extreure'n la màxima informació de forma visual, veiem què volem mostrar a la part 2 del seminari.

3. PART 2. Què volem mostrar? Per què és important visualitzar les dades?

Anem a utilitzar el conjunt de dades anscombe, del que ja us han parlat a la classe de teoria. Escrivint **anscombe** a R podeu visualitzar-lo. També podeu explorar la seva estructura amb `str(anscombe)`. O accedir a la seva ajuda escrivint `?anscombe`

```
> str(anscombe)
'data.frame':   11 obs. of  8 variables:
 $ x1: num  10 8 13 9 11 14 6 4 12 7 ...
 $ x2: num  10 8 13 9 11 14 6 4 12 7 ...
 $ x3: num  10 8 13 9 11 14 6 4 12 7 ...
 $ x4: num  8 8 8 8 8 8 19 8 8 ...
 $ y1: num  8.04 6.95 7.58 8.81 8.33 ...
 $ y2: num  9.14 8.14 8.74 8.77 9.26 8.1 6.13 3.1 9.13 7.26 ...
 $ y3: num  7.46 6.77 12.74 7.11 7.81 ...
 $ y4: num  6.58 5.76 7.71 8.84 8.47 7.04 5.25 12.5 5.56 7.91 ...
> |
```

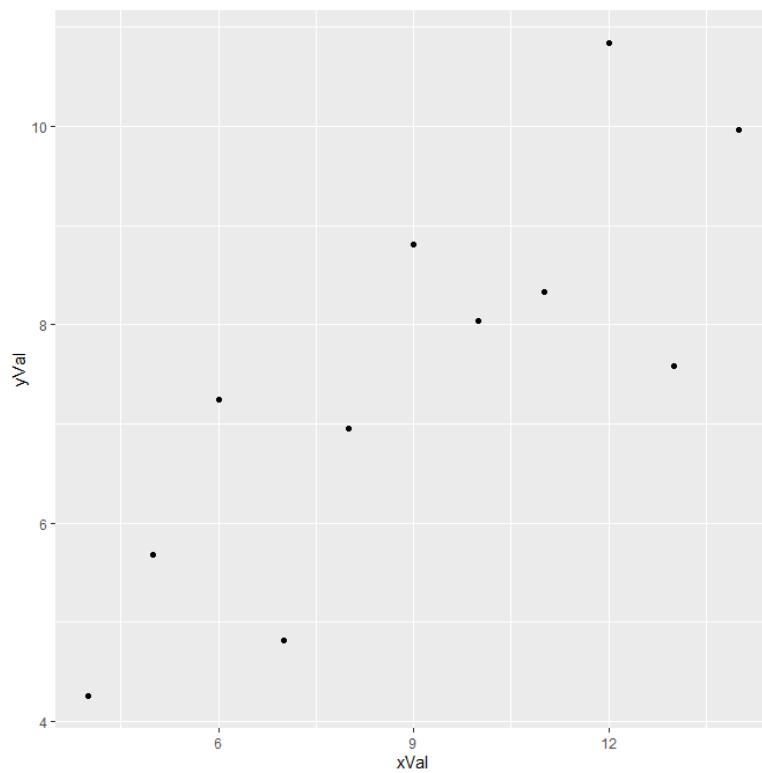
Primer construirem 4 grups de datasets. Els anomenarem **g1data**, ..., **g4data**. I cada un contindrà els valors (x_i, y_i) , amb $i=1..4$. Després en veure'm les seves respectives mitjanes i desviació estàndards, omplint la taula següent. Tot seguit plotejarem cada grup **g1data**, ..., **g4data** per separat utilitzant `ggplot()` amb `geom_point()`. Què està passant?

	<code>mean(gidata\$xVal)</code>	<code>mean(gidata\$yVal)</code>	<code>sd(gidata\$xVal)</code>	<code>sd(gidata\$yVal)</code>
g1data	9	7.500909	3.316625	2.031568
g2data	9	7.500909	3.316625	2.031657
g3data	9	7.500909	3.316625	2.030424
g4data	9	7.500909	3.316625	2.030579

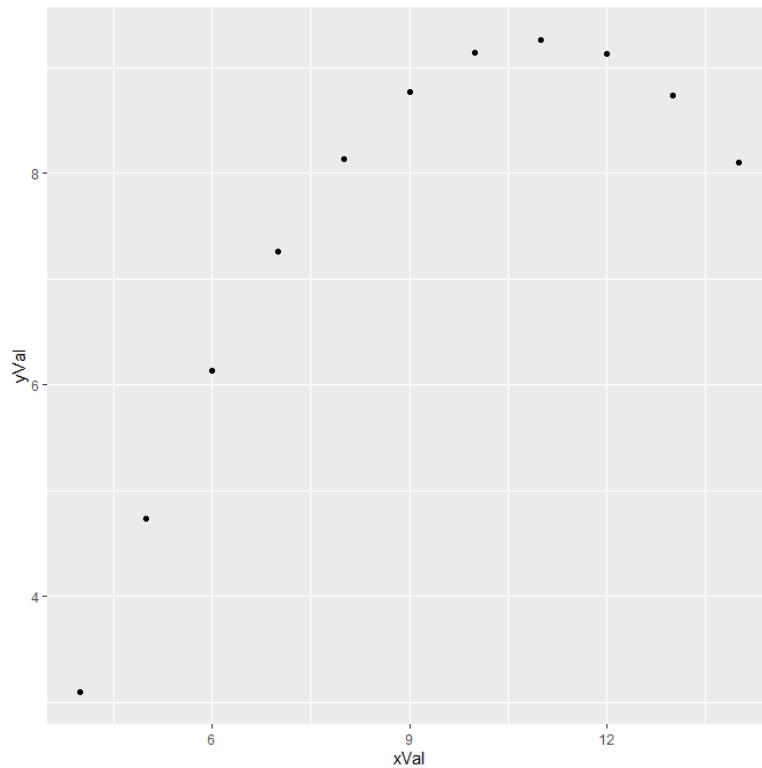
NOTA: `g1data=with(anscombe,data.frame(xVal=c(x1),yVal=c(y1)))` us crearà el primer grup. Per fer les respectives mitjanes farem servir les comandes que hem posat en cada columna de la taula anterior on $i=1,2,3,4$ respectivament:

```
> mean(g1data$xVal)
> mean(g1data$yVal)
> sd(g1data$xVal)
> sd(g1data$yVal)
```

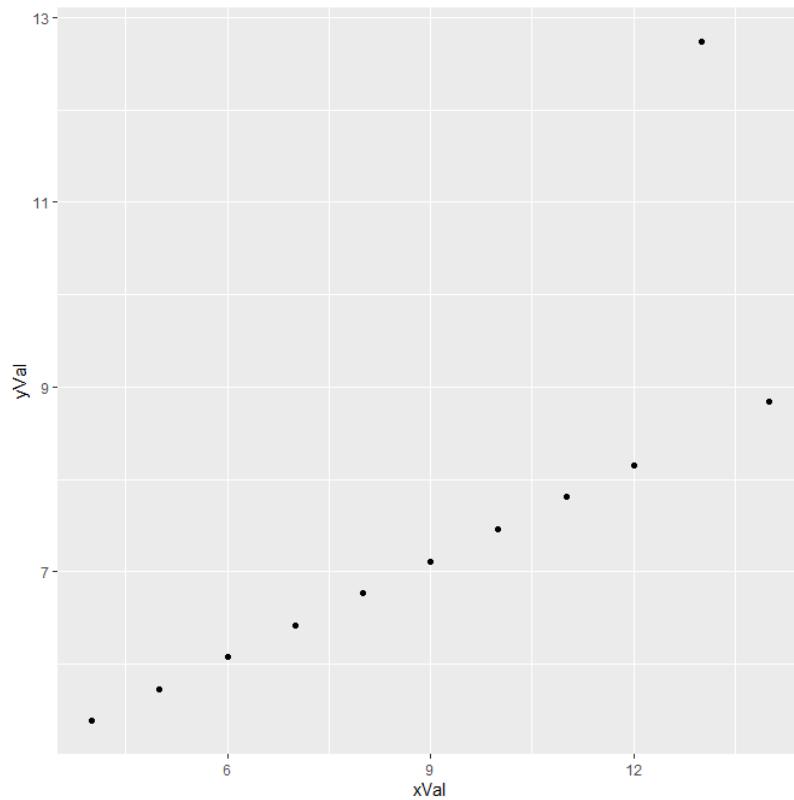
```
> g1data=with(anscombe,data.frame(xVal=c(x1),yVal=c(y1)))
> ggplot(g1data, aes(xVal, yVal))+geom_point()
```



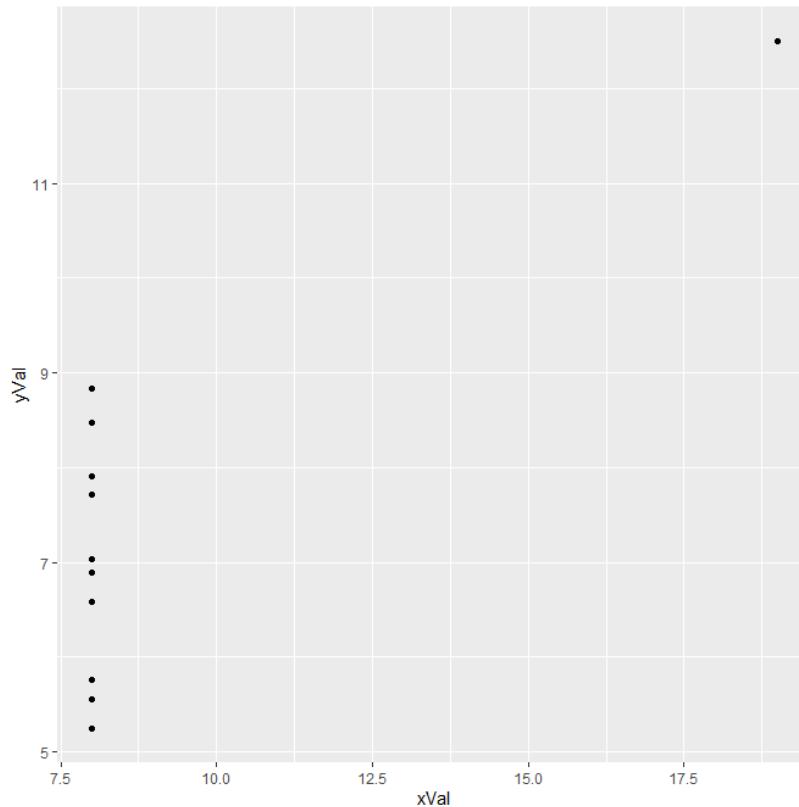
```
> g2data=with(anscombe,data.frame(xVal=c(x2),yVal=c(y2)))
> ggplot(g2data, aes(xVal, yVal))+geom_point()
```



```
> g3data=with(anscombe,data.frame(xVal=c(x3),yVal=c(y3)))  
> ggplot(g3data, aes(xVal, yVal))+geom_point()
```



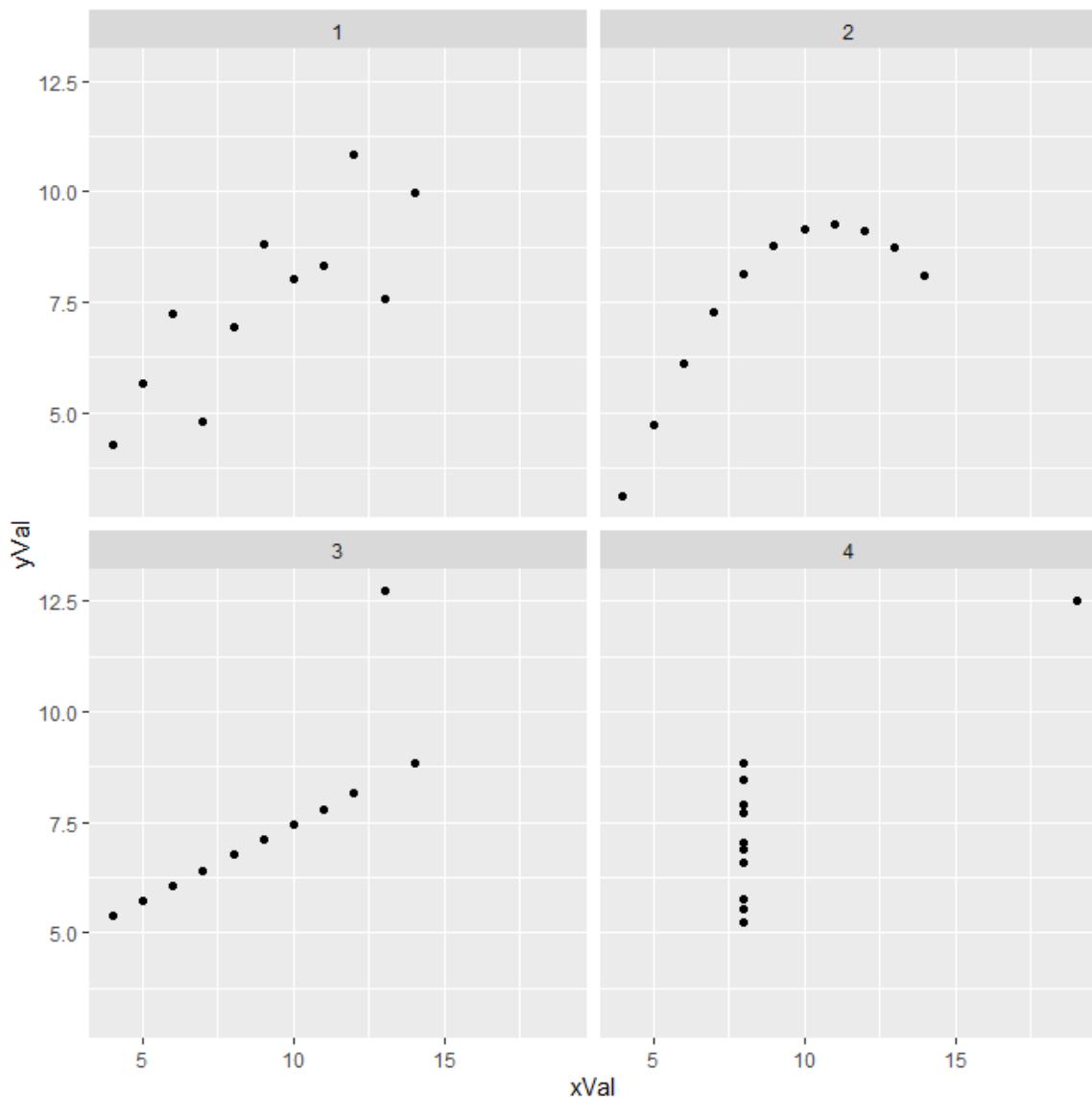
```
> g4data=with(anscombe,data.frame(xVal=c(x4),yVal=c(y4)))  
> ggplot(g4data, aes(xVal, yVal))+geom_point()
```



!!RESUM: El dataset ‘anscombe’ conté quatre conjunts de dades que tenen la mateixa mitjana i la mateixa desviació estàndard (per x i per y) però al visualitzar-les tenen una aparença molt diferent. Per això és important visualitzar les dades que tenim. Si només haguéssim mirat la mitjana o la desviació estàndard dels datasets que tenim, haguéssim assumit que els quatre datasets són iguals. La distribució dels punts ens mostra que no. En el pròxim seminari veure’m com mostrar distribucions.

EXTRA: D’una manera més avançada, que encara no hem vist, per crear els grups g_1, \dots, g_4 ((x_i, y_i) , amb $i=1, \dots, 4$) d’una manera menys manual i visualitzar-los en un mateix gràfic, escriure’m en R:

```
> anscombeData=with(anscombe,data.frame(xVal=c(x1,x2,x3,x4),
yVal=c(y1,y2,y3,y4), anscombeGroup=gl(4,nrow(anscombe))))
> ggplot(anscombeData,aes(x=xVal,y=yVal,group=anscombeGroup))+geom_point() + facet_wrap(~anscombeGroup)
```



SEMINARI 2. Comparacions i Distribucions (Respostes)

1. OBJECTIUS

La part 1 d'aquest seminari introduceix les geometries de *ggplot* que ens permeten visualitzar comparacions de dades categòriques (nominals i ordinals). La part 2, introduceix les geometries de *ggplot* que ens permeten visualitzar distribucions. A més, el seminari té com objectiu també, introduir algunes eines que ens permetran fer una mínima edició dels nostres gràfics.

2. PART 1. Comparacions mitjançant els diagrames de barres

Continuem amb el conjunt de dades *mtcars* del seminari 1 que conté informació de 32 cotxes. Com el conjunt de dades és petit, poseu `view(mtcars)` per veure el *dataframe* en forma de taula. La funció `view()` ens permet obtenir informació sobre el conjunt de dades de manera complementaria a les dues maneres que vam veure l'altre dia: l'ajuda `?mtcars` o la funció mostrant l'estructura, `str(mtcars)`.

Si obriu R de nou, primer de tot recordeu que heu de tornar a carregar la llibreria tidyverse: `library(tidyverse)`.

(Els solucionaris contenen una possible solució, però podria haver més d'una solució)

EXERCICIS:

1.- Veient una sola gràfica, volem saber quants cotxes hi ha de tres grups/nivells diferents segons el nombre de cilindres (4 cilindres, 6 cilindres i 8 cilindres). Feu un gràfic de barres on l'alçada de les barres sigui proporcional al nombre de cotxes de cada grup/nivell. Quina informació us dona la gràfica?

- Poseu la etiqueta als eixos utilitzant `xlab()` i `ylab()`
- Pinteu les barres segons el nombre de cilindres utilitzant `fill` dins `aes()`. Ens dona alguna informació extra pintar-les? Per què?
- Poseu títol i etiquetes a la llegenda que us ha sortit anteriorment per defecte
- Feu les barres més estretes especificant la `width` de les barres. Nota: `width` actua com argument de `geom_bar()`

Hem vist les comandes `geom_bar()` i `geom_col()` a la primera part de la classe. Fixeu-vos que volem que l'alçada del nostre gràfic sigui proporcional al nombre de cotxes d'una categoria, per tant utilitzem `geom_bar()`.

En l'exercici 1 del seminari 1, ja vam veure que si no indicàvem que 'cyl' era una variable 'factor', l'eix x on representàvem el nombre de cilindres ens enganyava mostrant una variable contínua. Aquí ens passarà el mateix si fem:

```
> ggplot(mtcars, aes(cyl)) + geom_bar()
```

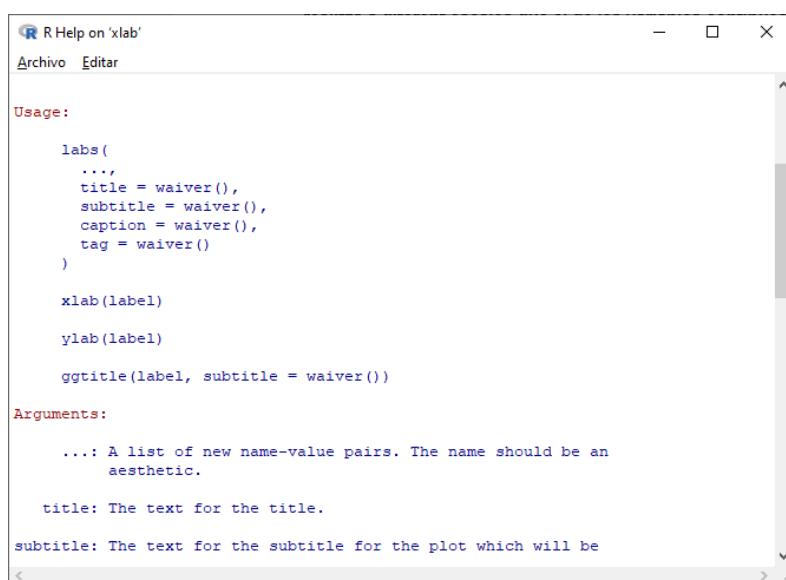
Per tant veiem que hem d'utilitzar *factor* per especificar que 'cyl' és un factor i que no ens passi això. A més avui hem vist que si volem emfatitzar que és una variable categòrica ordinal, com és el cas de 'cyl' podem utilitzar *ordered*. Per simplicitat, fem com vam veure en el seminari 1, i creem ja un dataframe mtcars2 que contingui les variables de mtcars on 'cyl' sigui ara una variable categòrica ordinal:

```
>mtcars2 <- within(mtcars, {
  cyl <- ordered(cyl)
})
> ggplot(mtcars2, aes(cyl)) + geom_bar() #o també:
> ggplot(mtcars2)+ aes(cyl) + geom_bar()
```

a) No hem utilitzat encara xlab i ylab. Demaneu ajuda:

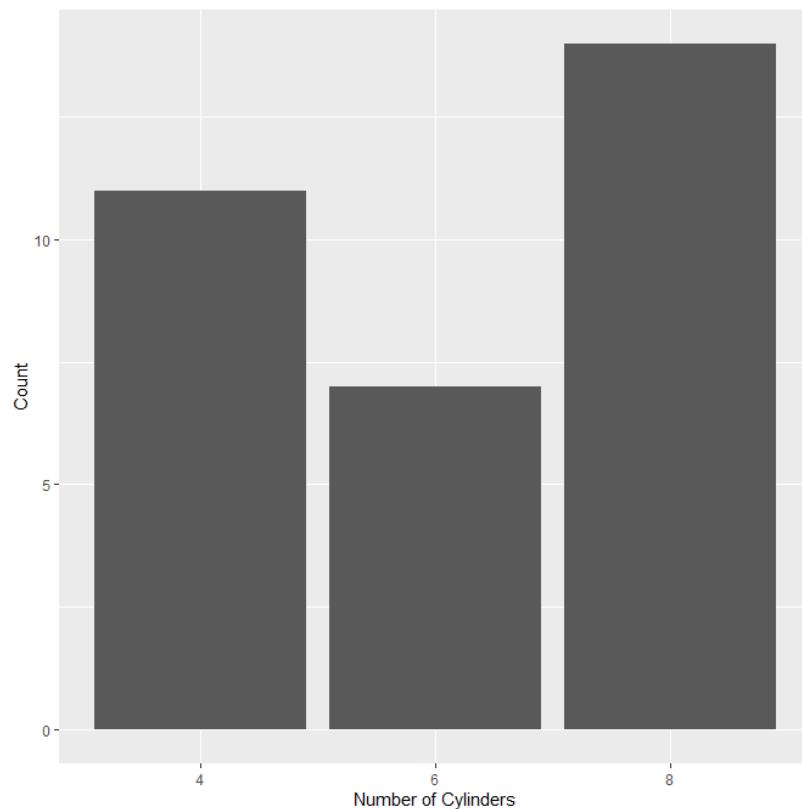
```
>?xlab
```

L'ajuda ens mostra com s'utilitza xlab (ylab s'utilitza igual):



Assignem a una variable *g* el que teníem i sumem les etiquetes dels eixos:

```
> g<- ggplot(mtcars2)+ aes(cyl) + geom_bar()
> g+xlab("Number of Cylinders")+ ylab("Count")
```



Aquesta gràfica, ens indica d'una manera prou visual que el grup amb més cotxes és el grup dels cotxes que tenen 8 cilindres. De fet en tenim aproximadament el doble que de 6 cilindres, el grup del qual en tenim menys.

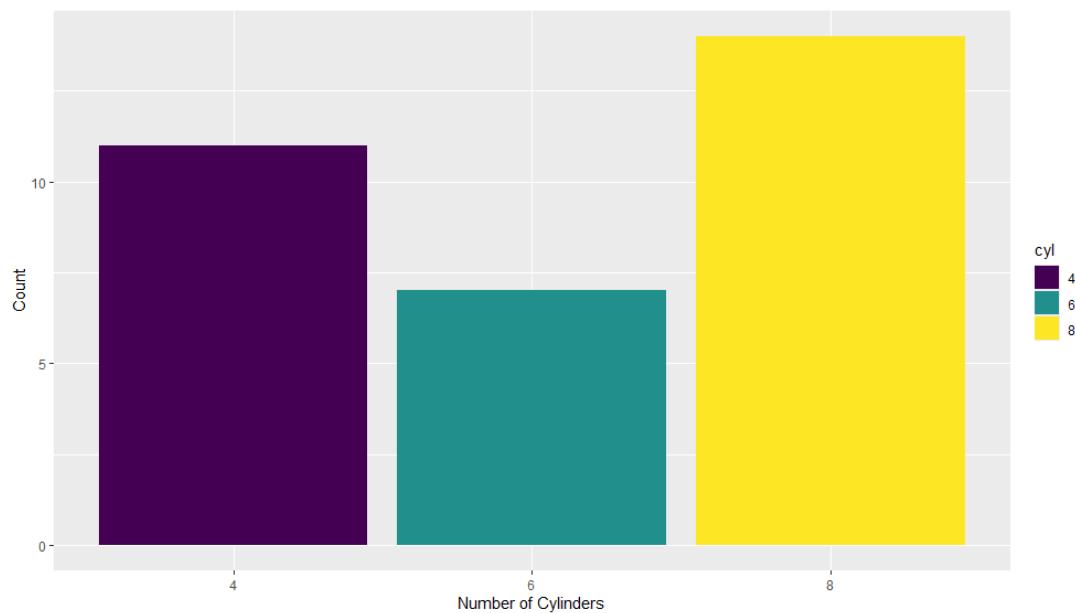
NOTA: També podríem fer-ho com vam veure l'altre dia amb `scale`, però ho em fet amb `label` per l'enunciat, si ho féssim seria:

```
> g+ scale_x_discrete("Number of  
Cylinders") + scale_y_continuous("Count")  
  
> g # per mostrar
```

Finalment, també podríem utilitzar `labs`. Veieu:
<https://ggplot2.tidyverse.org/reference/labs.html>

b) Ja vam veure en el 1er seminari que per pintar una gràfica segons un grup de variables necessitàvem fer un mapeig del color en `aes()`, on especificaríem el color segons el grup de variables que volíem. Aquí ens diuen que utilitzem `fill` enllot de `color`, però ho fem de la mateixa manera (en `aes()`). Nota: Recordeu utilitzar factor amb 'cyl' per a que l'escala de color no sigui contínua.

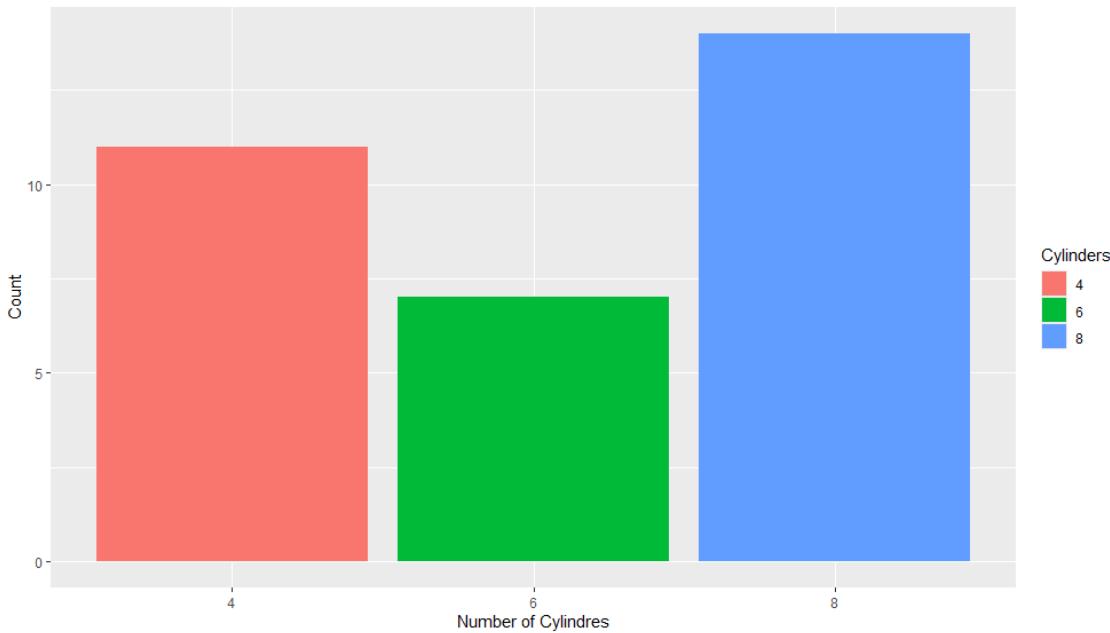
```
> b<-ggplot(mtcars2, aes(cyl, fill=cyl)) + geom_bar() + xlab("Number  
of Cylinders") + ylab("Count")  
  
> b
```



El color NO ens aporta cap informació nova. L'eix x ja ens estava dividint les dades en els tres grups que volíem.

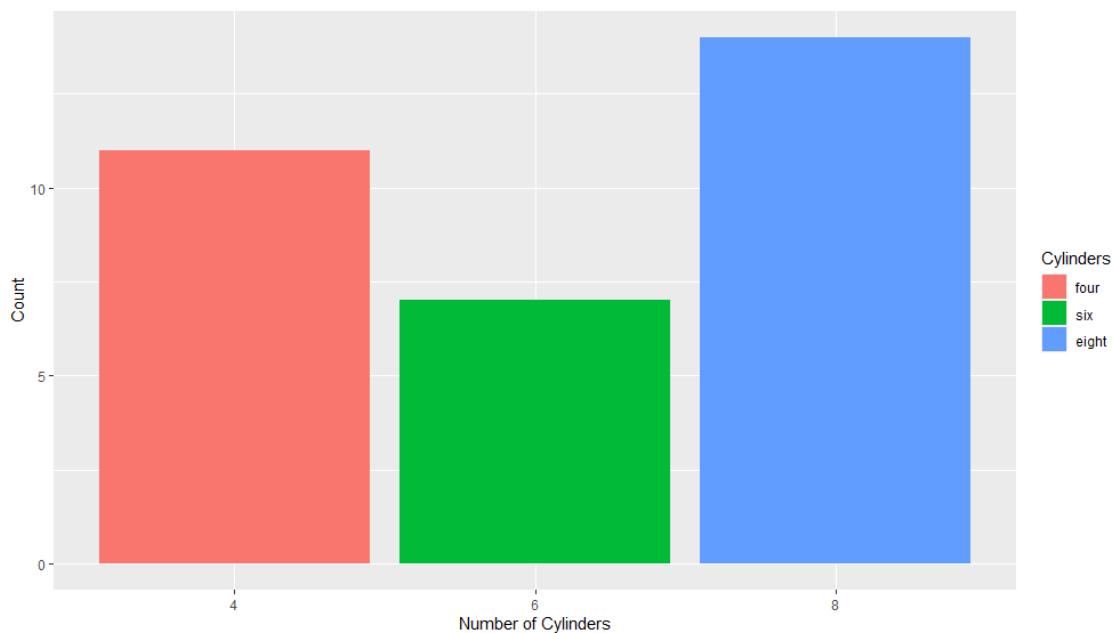
c) Podem posar un títol i etiquetes a la llegenda com l'altre dia, amb `scale_fill_discrete`

```
> b<-ggplot(mtcars2, aes(cyl, fill=cyl)) + geom_bar()
+scale_x_discrete("Number of Cylindres") + scale_y_continuous("Count")
> b+scale_fill_discrete("Cylinders")
```



També podem posar etiquetes personalitzades a la llegenda:

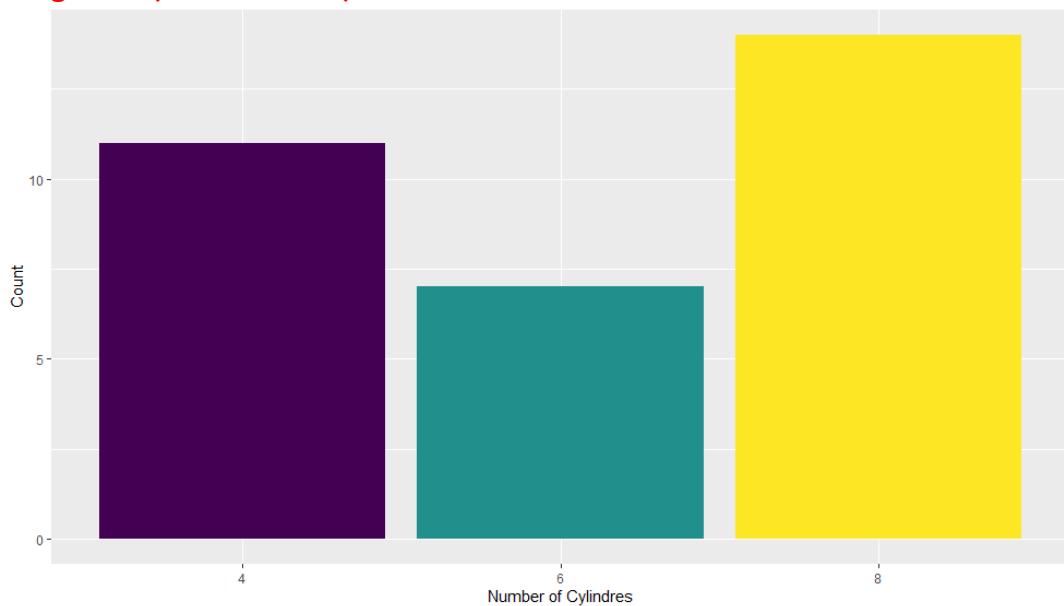
```
> b+scale_fill_discrete("Cylinders", labels=c('four','six','eight'))
```



NOTA: També podríem utilitzar `labs`. Veieu l'enllaç que ja citàvem en l'apartat (a):
<https://ggplot2.tidyverse.org/reference/labs.html>

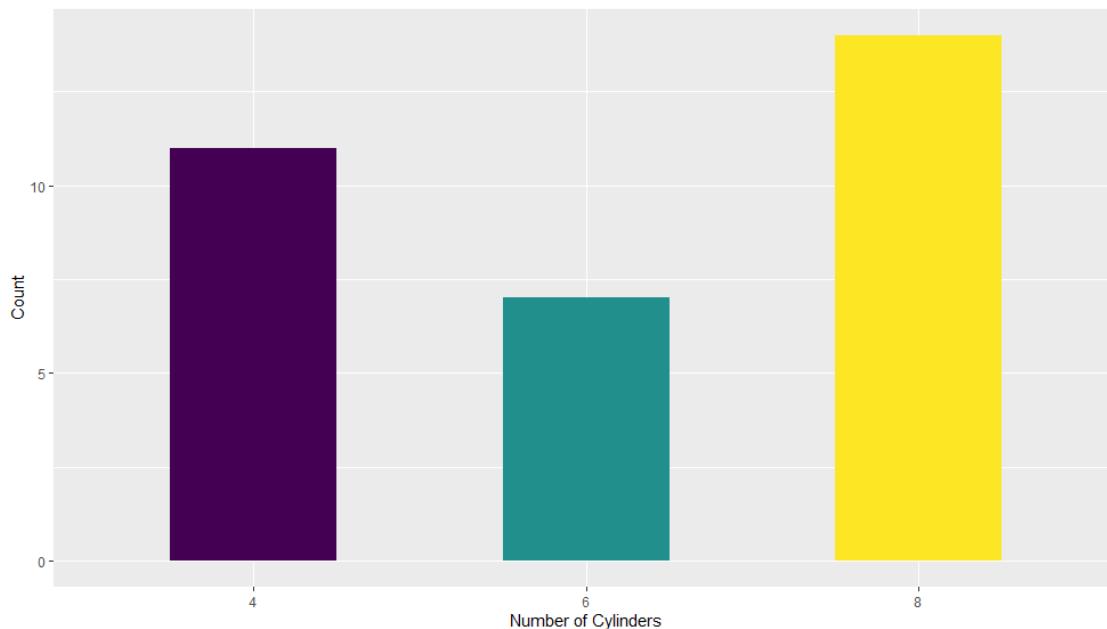
NOTA2: De fet si hi pensem les etiquetes de la llegenda NO ens donen informació extra pel que podríem treure-la posant:

```
>b+guides(fill="none")
```



d) Exemple amb `width=0.5`

```
> ggplot(mtcars2, aes(cyl, fill=cyl)) + geom_bar(width=0.5) +
  xlab("Number of Cylinders") + ylab("Count") +guides(fill="none")
```



Ens permet fer barres més amples o estretes segons la mida posem. L'amplada de les barres ha de ser proporcional a l'eix x.

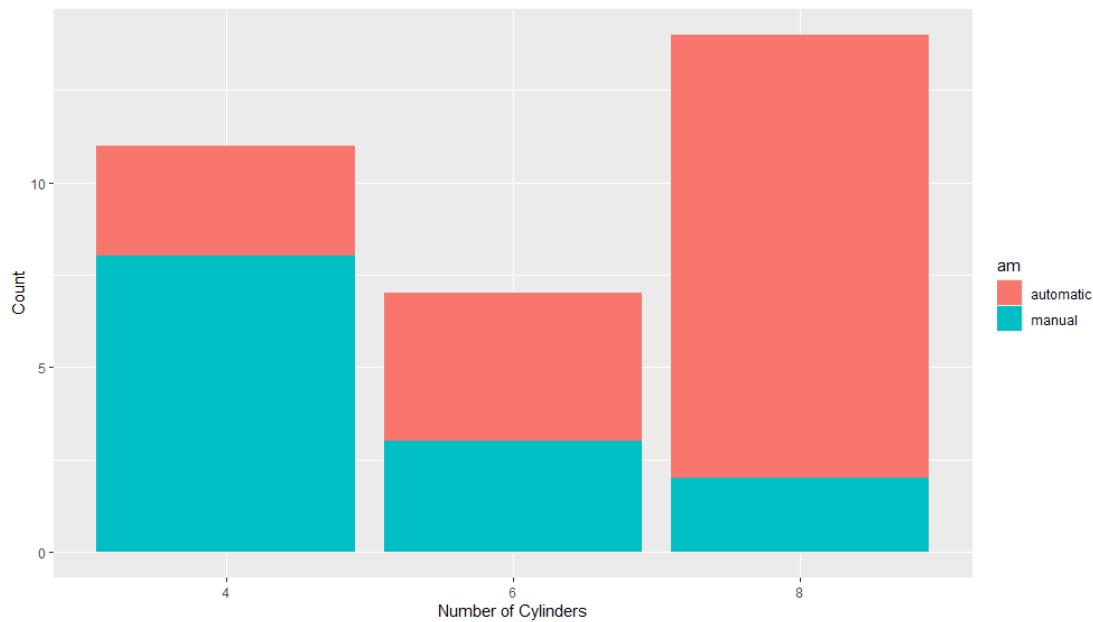
NOTA: Si haguéssim provat de fer una amplada de 1.5, les barres s'haurien sobreexposat i R ens tornaria un *warning* que ens ho indicaria.

2.- Feu un gràfic on pugueu veure la proporció de vehicles amb transmissió manual versus automàtica segons el nombre de cilindres. Què observeu?

Utilitzeu l'argument *dodge* de manera que cada tipus de transmissió aparegui en l'eix x, una al costat de l'altra d'acord amb el nombre de cilindres. Per fer això últim especificarem l'argument *posició* en `geom_bar(position="dodge")`. Observeu més coses?

Com sempre, abans de fer la gràfica, primer mirem com són les variables 'cyl' i 'am'. Pregunteu-vos, per exemple: són contínues/discretes?, divideixen les dades en subgrups?, es poden categoritzar amb *factor* o no?. La variable 'cyl' ja hem vist que es pot categoritzar utilitzant *factor/ordered* segons la connotació que volem donar. D'altra banda, 'am' és una variable lògica que només pren dos valors (0 o 1), segons transmissió sigui automàtica (0) o manual (1). Per tant també es pot categoritzar amb *factor*.

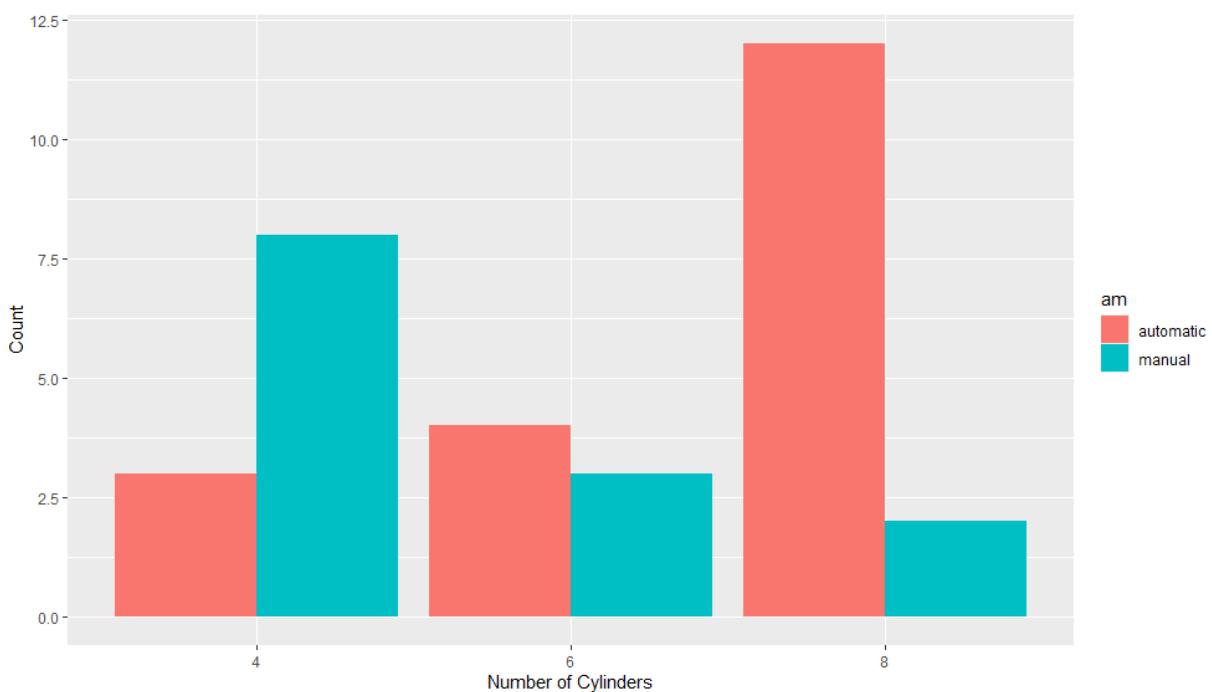
```
>mtcars2 <- within(mtcars2, {
  am <- factor(am, labels = c("automatic", "manual"))
})
> ggplot(mtcars2, aes(cyl, fill=am))+ geom_bar()+xlab("Number of
Cylinders")+ylab("Count")
```



Sembla que la major part dels cotxes de 4 cilindres tenen una transmissió manual, en el cas de 6 cilindres ja tenim algun cotxe més amb transmissió automàtica que manual i en el cas de 8 cilindres, encara més. També veiem globalment que tenim pocs cotxes amb 6 cilindres, i dels que més en tenim són dels de 8 cilindres.

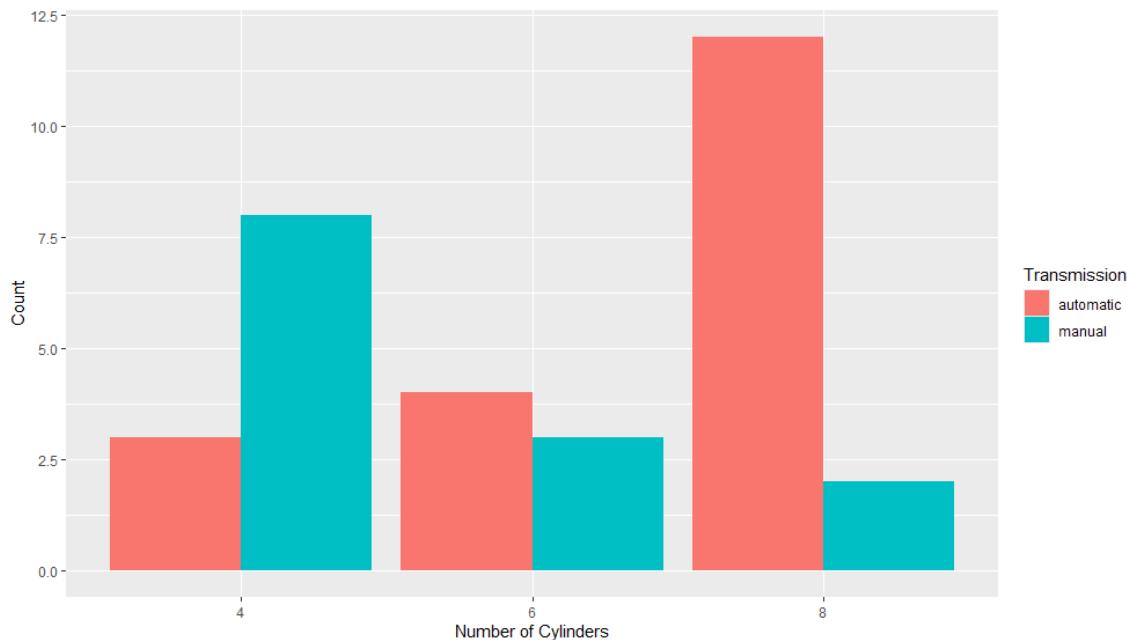
Però, aquesta gràfica no ens mostra la informació que volem d'una forma simple. Fem el que ens diu l'enunciat utilitzant l'argument posició a veure si ens ajuda. Per això, especificarem, com ens diu l'enunciat, `geom_bar(position="dodge")`:

```
> g<-ggplot(mtcars2, aes(cyl, fill=am))+  
  geom_bar(position="dodge")+xlab("Number of Cylinders")+ylab("Count")  
  
> g
```



I podem posar títol a la llegenda per fer-ho més fàcil d'interpretar:

```
> g+scale_fill_discrete("Transmission")
```



Com ens deia l'enunciat: la transmissió apareix en l'eix x, una al costat de l'altra d'acord amb el nombre de cilindres de cada cotxe. La gràfica és molt més clara que abans. Ara comparar les proporcions entre cotxes (automàtics o manuals) per un cert nombre de cilindres es pot fer d'una manera molt més clara, i sense perdre la informació global. *Efectivament veiem que la transmissió automàtica (en vermell) creix notablement a mesura que augmentem els cilindres (i exponencialment quan passem de '6' a '8' cilindres). A la vegada, la transmissió manual (en blau), decreix notablement quan els cilindres dels cotxes augmenten. En el cas dels cotxes amb 8 cilindres (columnes més a la dreta), per exemple, la quantitat de cotxes automàtics (vermell) és més de 4 vegades que la de manuals (blau).*

NOTA: Podeu veure més arguments en
https://ggplot2.tidyverse.org/reference/geom_bar.html

Ara que hem vist com comparar algunes proporcions mitjançant diagrames de barres, veiem les distribucions a la part 2 del seminari.

3. PART 2. Distribucions

EXERCICIS:

1.- Mostreu la distribució de la variable 'hp'. Quina informació en podeu extreure?

Avui hem vist just dues maneres de mostrar distribucions, via histogrames, polígons de freqüència i diagrames de caixes (boxplots). Hem vist que els histogrames i els polígons de freqüències eren útils quan volíem mostrar la distribució d'una funció contínua com és el cas de 'hp'.

Si feu:

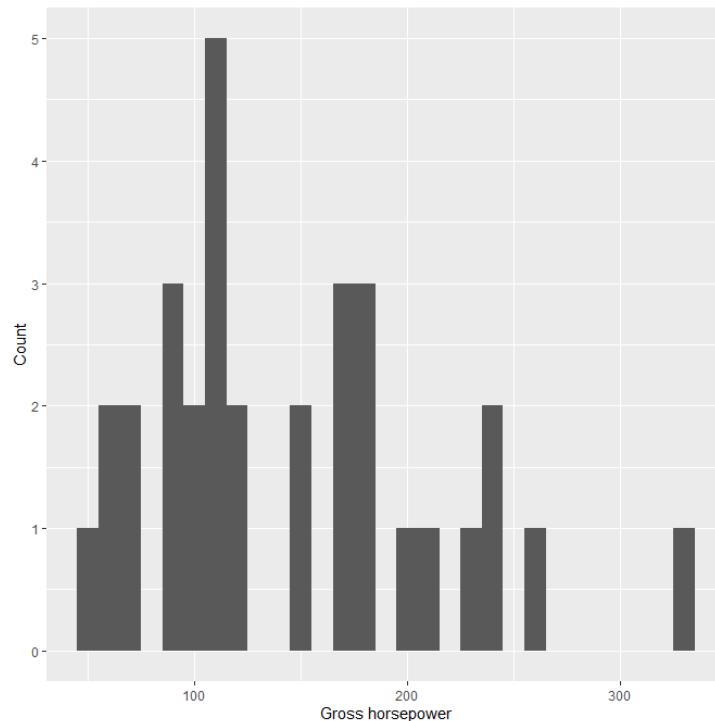
```
> ggplot(mtcars,aes(hp))+geom_histogram()+xlab("Gross Horsepower")+
  ylab("Count")
```

ATENCIÓ: R ens retorna un error:

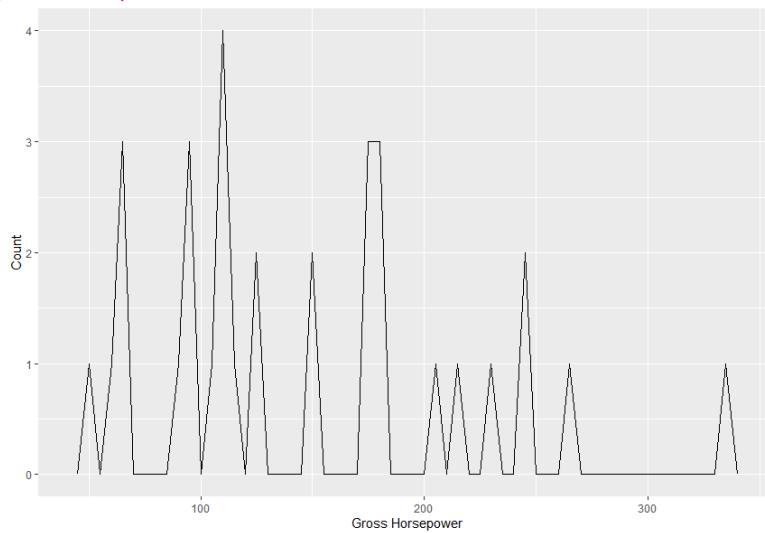
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

Com dèiem abans hem d'especificar l'argument *binwidth*. Exemple:

```
> ggplot(mtcars, aes(hp))+geom_histogram(binwidth=10)+xlab("Gross Horsepower") +ylab("Count")
```



```
> ggplot(mtcars, aes(hp))+geom_freqpoly(binwidth=5)+xlab("Gross Horsepower") +ylab("Count")
```



Sembla que tenim poques dades, veiem que hi ha 5 elements amb una potència bruta de 100 o cap al voltant de 300. Però en general, tenim quelcom dispers

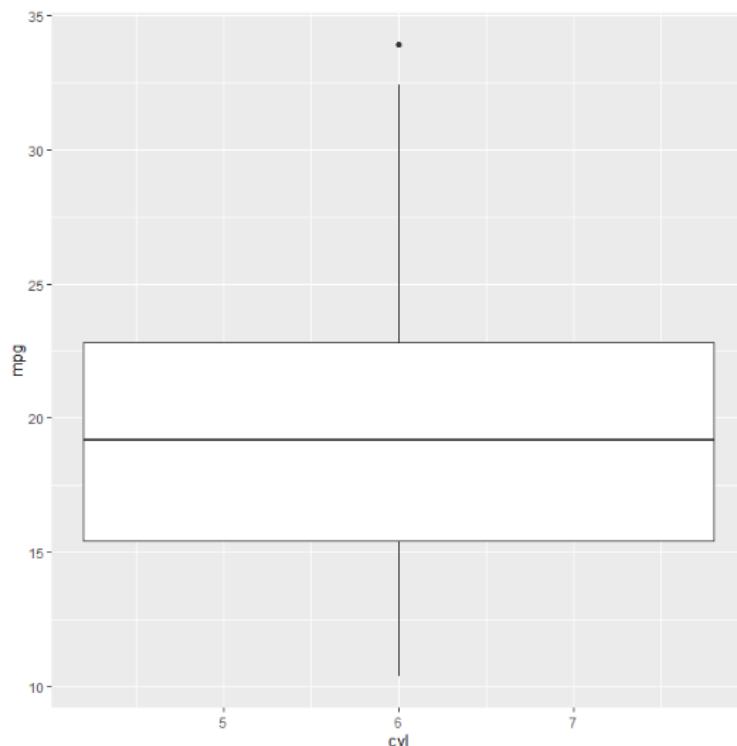
2.- Feu ús de geom_boxplot() per fer un gràfic en que l'eix de les x correspongui a la variable 'cyl' (cilindres) i l'eix y a la variable 'mpg' (milles de galó). Un cop

tingueu el gràfic, compareu aquesta gràfica amb la resultant de l'exercici 2 del seminari 1 (Quina informació tenim aquí que no teníem abans?). A més:

- Poseu la llegenda dels eixos utilitzant `xlab()` i `ylab()`
- Gireu els eixos de coordenades, fent que els boxplots quedin en horitzontal utilitzant la comanda `coord_flip()`
- Compareu la gràfica resultant amb la gràfica de l'exercici 2 seminari 1.

Si fem directament:

```
> ggplot(mtcars,aes(x=cyl,y=mpg)) + geom_boxplot()
```



R ens retorna aquesta gràfica i un avis:

Warning message:

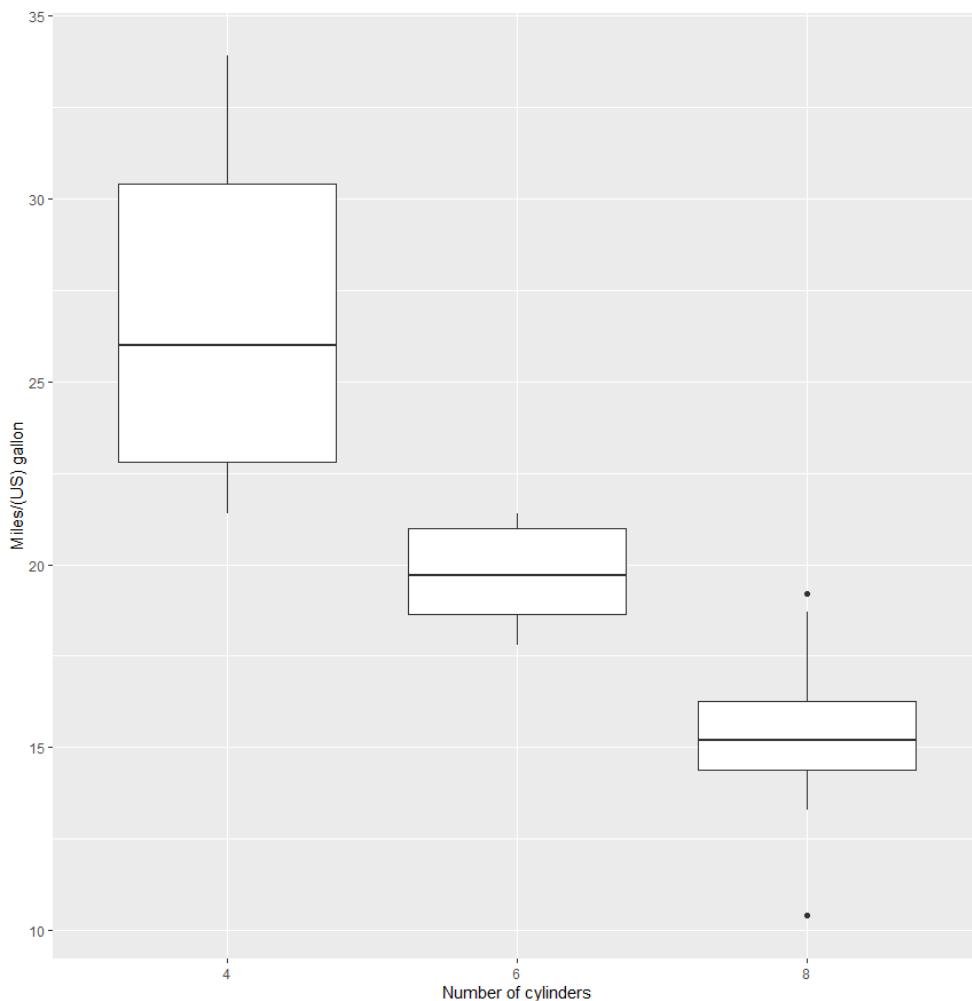
Continuous x aesthetic -- did you forget aes(group=...)?

Com hem vist a classe, quan tenim dues variables, els boxplots són útils quan volem observar la distribució de dues variables, on una de les variables és discreta i l'altra variable és contínua. Ja hem vist en la Part 1 d'aquest seminari, i en el seminari 1, com fer que aquesta variable sigui tractada de forma discreta, **categoritzant-la amb factor/ordered**. Per tant usem directament mtcars2 (o veiem en la Part 1 com crear-lo):

```
> ggplot(mtcars2,aes(x=cyl,y=mpg)) + geom_boxplot()
```

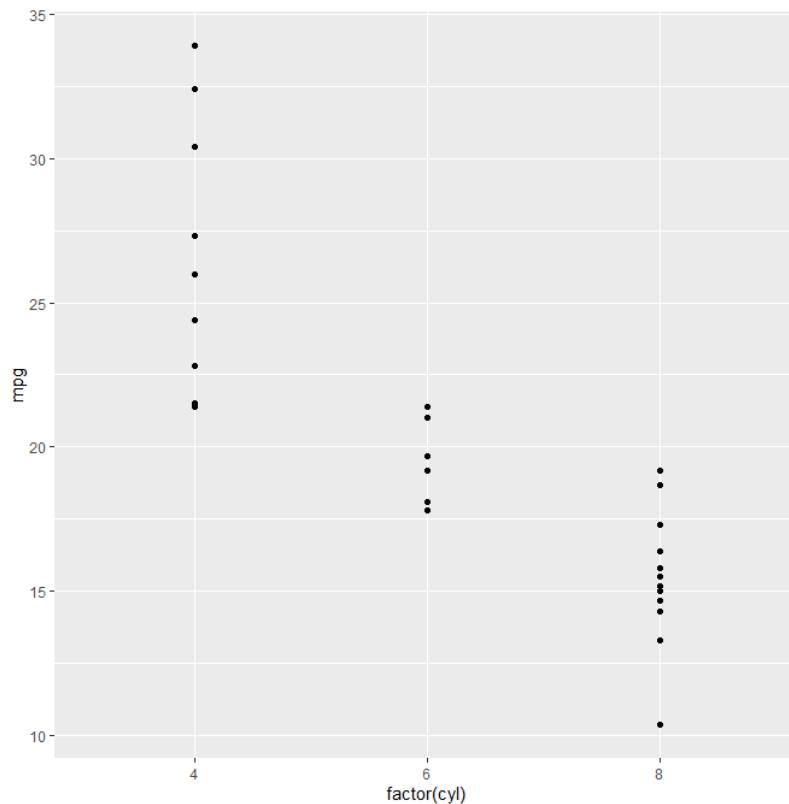
- Com en l'exercici 1 de la part 1, fem us de `xlab` i `ylab` per posar noms als eixos de la gràfica

```
> ggplot(mtcars2,aes(x=cyl,y=mpg))+ geom_boxplot() +
  xlab('Cylinders') + ylab('Miles/(US) gallon')
```



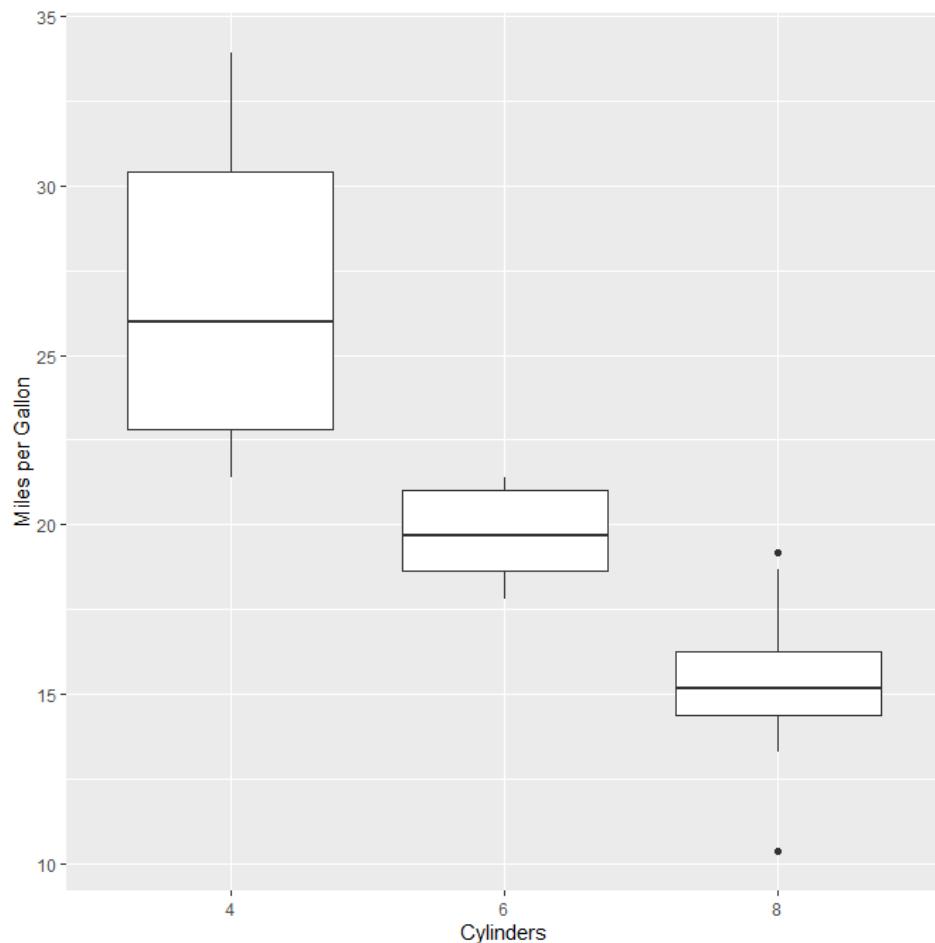
PRIMER FEM l'apartat c

c) Si comparem amb la gràfica de l'exercici 2 del seminari 1 (a sota), tot i que ja podíem intuir alguna informació preliminar sobre la distribució de les nostres dades en la gràfica de punts (scatter plot), no era tant evident com en el boxplot que acabem de fer (i en perdíem informació). Això es deu al fet que la variable cilindres és qualitativa. Així com en el cas de variables quantitatives els scatter plots ens donen molta informació, no ens en donen tanta per a variables qualitatives.



Gràfica de l'exercici 2 seminari 1

OBSERVACIONS que ens aporta aquest nou gràfic: Ara podem veure per exemple ràpidament que: i) la mediana de milles de galó dels cotxes amb 8 cilindres és al voltant de 15, mentre que la de 6 cilindres és al voltant de 20, i puja fins més de 25mpg en el cas dels cotxes amb 4 cilindres (el qual tindria sentit); ii) tenim outliers en el cas dels cotxes amb 8 cilindres (representats amb dos punts fora del boxplot); iii) els quartils Q1 i Q3 i per tant la distribució (és molt més amplia en termes de 'mpg'), varia més en el cas dels cotxes de 4 cilindres, per exemple; iv) la desviació, casi és nul·la pels cotxes de sis cilindres per exemple (gairebé no tenim bigotis).



b) Com sempre que no hem utilitzat prèviament un comandament fem servir **?coord_flip** per a que l'ajuda de R ens digui com podem utilitzar-lo

R Help on 'coord_flip'

Archivo Editar

limits, then those data points may show up in places such as the axes, the legend, the plot title, or the plot margins.

Examples:

```
# Very useful for creating boxplots, and other interval
# geoms in the horizontal instead of vertical position.

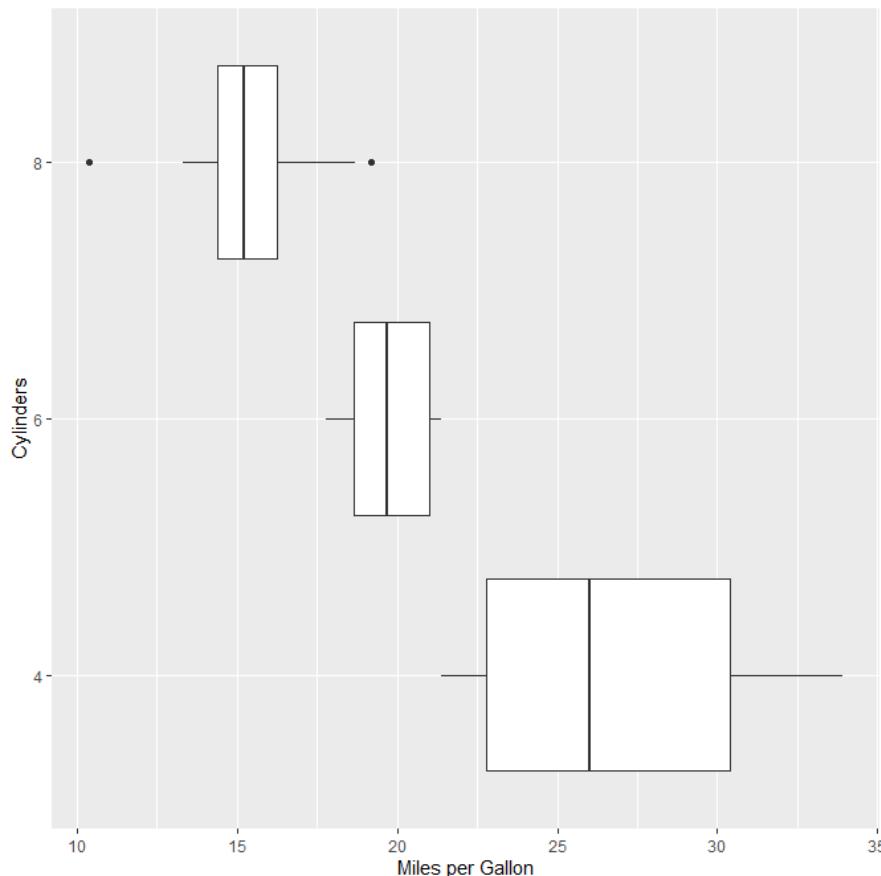
ggplot(diamonds, aes(cut, price)) +
  geom_boxplot() +
  coord_flip()

h <- ggplot(diamonds, aes(carat)) +
  geom_histogram()
h
h + coord_flip()
h + coord_flip() + scale_x_reverse()

# You can also use it to flip line and area plots:
df <- data.frame(x = 1:5, y = (1:5) ^ 2)
ggplot(df, aes(x, y)) +
  geom_area()
last_plot() + coord_flip()
```

Farem:

```
> ggplot(mtcars2,aes(x=cyl,y=mpg)) + geom_boxplot() +
  xlab('Cylinders') + ylab('Miles per Gallon')+coord_flip()
```



Però, com abans, per no haver de repetir cada vegada la línia de codi, podem assignar la nostra gràfica a una variable i afegir comandes a fer mesura, obtenint el mateix resultat. Escriure'm:

```
> my_boxplot <- ggplot(mtcars2,aes(x=cyl,y=mpg))+geom_boxplot()
+ xlab('Cylinders')+ylab('Miles per Gallon')
> my_boxplot+coord_flip()
```

Escolliu en cada cas la manera de treballar que us sigui més còmoda.

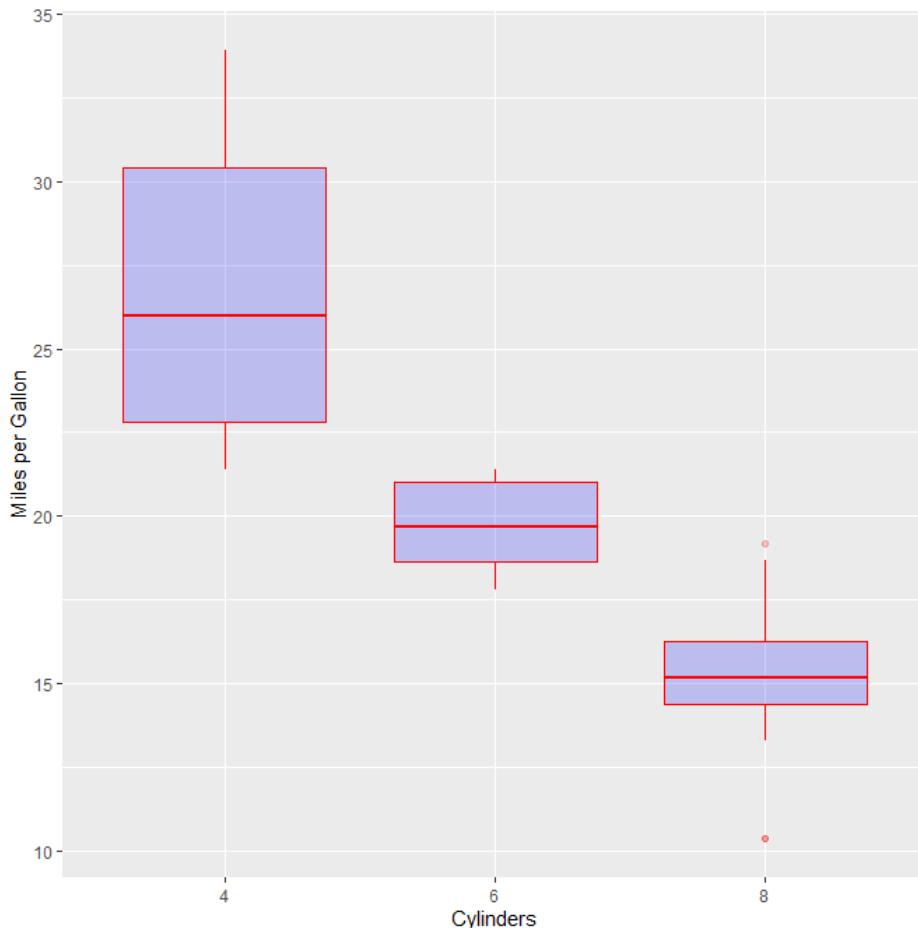
d) Pinteu els boxplots de tres maneres diferents:

d1) Escollint un únic color amb fill, color i alpha per als tres grups:

```
geom_boxplot(color="red", fill="blue", alpha=0.2)
```

Fixeu-vos en el resultat de canviar els tres paràmetres *color*, *fill*, *alpha*. Què fa cadascun?

```
> ggplot(mtcars2,aes(x=cyl,y=mpg)) + geom_boxplot(color="red",
fill="blue", alpha=0.2) + xlab('Cylinders') + ylab('Miles per Gallon')
```



Color canvia el color referent als contorns del boxplot, *fill* el color de dins, i *alpha* ens varia l'opacitat del color amb el que pintem.

!REMARCA: Fixem-nos que estem utilitzant els arguments de `geom_boxplot` (com hem fet abans amb els arguments `width` o `position` en `geom_bar()`). No és que estem fent un mapeig del color per cada grup com fèiem amb `aes()` en el seminari 1 o abans en l'exercici 1 de la part 1.

d2) De manera que el color es trobi en el contorn del boxplot per a cada grup (cotxes amb els mateixos cilindres)

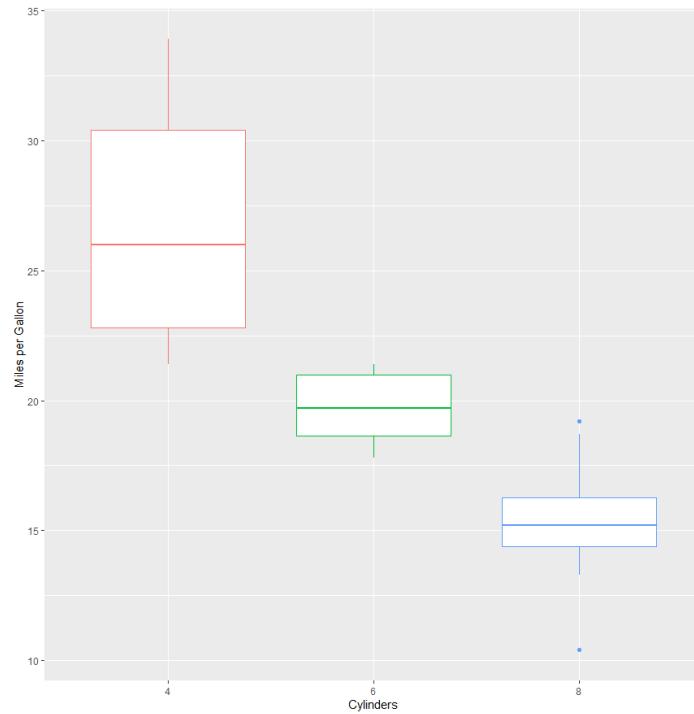
d3) De manera que el color es trobi en l'interior del boxplot per a cada grup (cotxes amb els mateixos cilindres)

Els canvis fets en d2) i d3) ens aporten alguna informació nova?

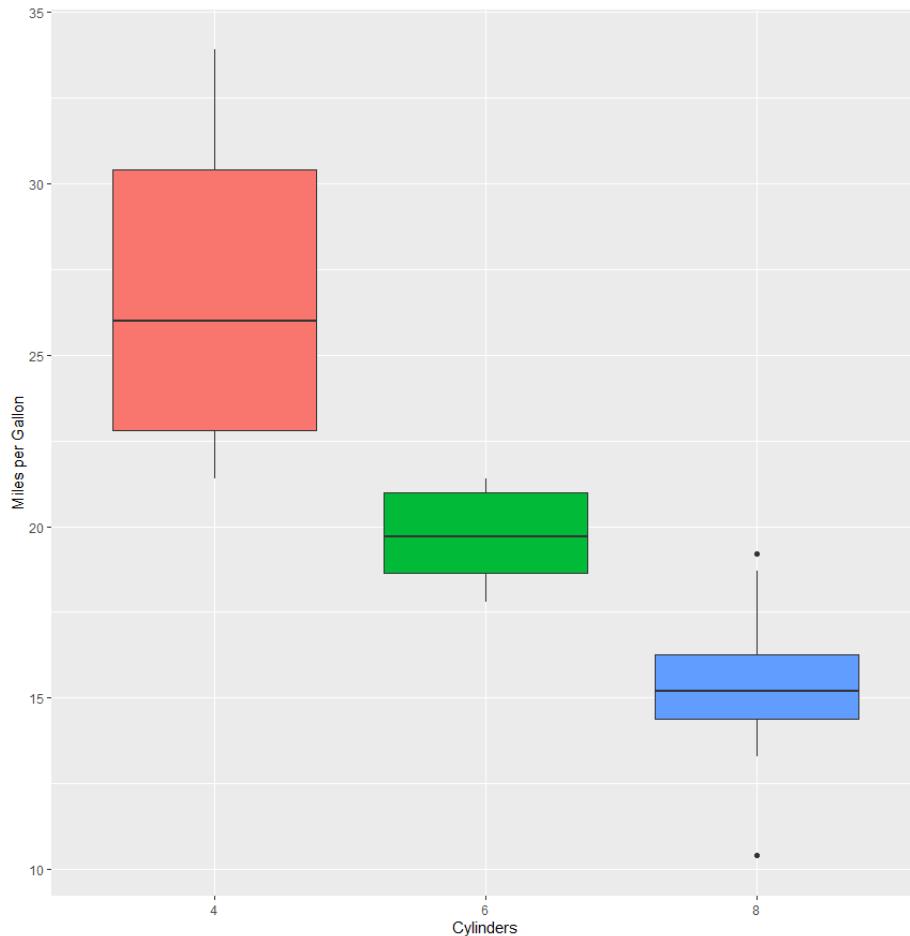
NOTA: Per d2) i d3) al pintar diferenciant per grups recordeu que afegim la informació dins d'aes() com ja havíem fet en el seminari 1.

Seguim el que ens diu la nota, especificant dins `aes()` la variable que volem agrupar, tal i com fèiem en el seminari 1.

```
d2) > ggplot(mtcars2,aes(x=cyl,y=mpg, color=cyl)) + geom_boxplot()
+   xlab('Cylinders')      +   ylab('Miles      per      Gallon')+ 
guides(color="none")
```



```
d3) > ggplot(mtcars2,aes(x=cyl, y=mpg, fill=cyl)) + geom_boxplot()
+   xlab('Cylinders') + ylab('Miles per Gallon') +
guides(fill="none")
```

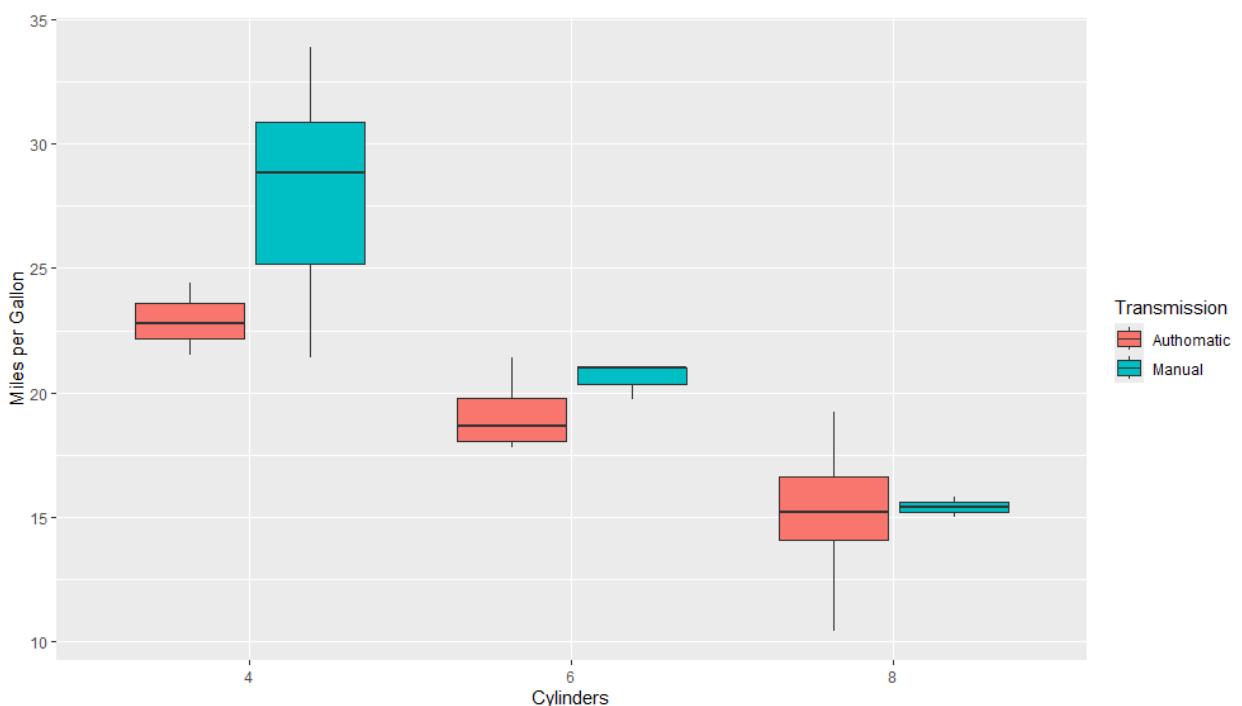


De la mateixa manera que havíem vist en el seminari 1, *pintar els boxplots segons el grup del nombre de cilindres al que pertanyen, no ens aporta cap informació nova respecte el mateix boxplot sense color.*

El mapeig de color per grups via `aes()`, ens podria aportar informació si afegim en el mateix gràfic el color als grups generats per una variable discreta com pot ser el tipus transmissió del cotxe (`factor(am)`) o la forma del motor (`factor(vs)`). (Si ho féssim amb `vs`, recordem que hauríem de categoritzar-la com hem fet amb `am`).

Exemple:

```
> ggplot(mtcars2,aes(x=cyl, y=mpg, fill=am)) + geom_boxplot() +
  xlab('Cylinders') + ylab('Miles per Gallon')+
  scale_fill_discrete("Transmission")
```



I com sempre, per facilitar la visualització posem títol a la llegenda.

3.- Mostreu en un mateix gràfic un polígon de freqüències i un histograma del temps en quarts de milla (qsec). Afegiu títol i etiquetes als eixos. Poseu un color per cada geometria. Què es mostra? NOTA: Ajusteu els bins.

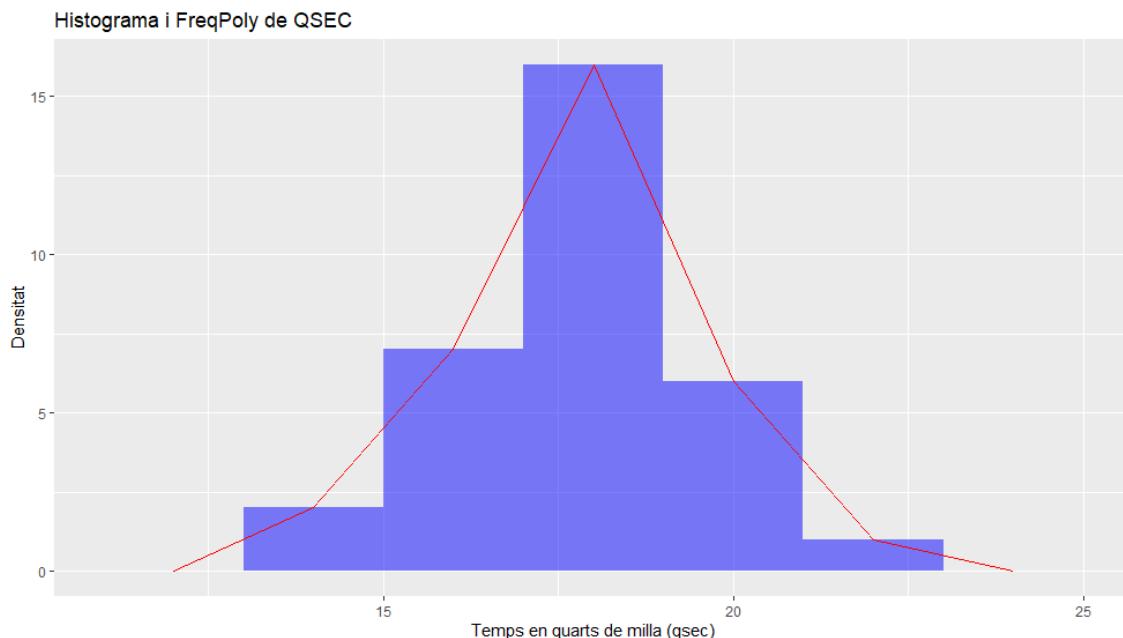
Un cop tingueu el gràfic afegiu la capa `theme_minimal()`, què us fa?

Ja hem vist que podem anant sumant capes, podem usar colors i canals alpha (que ens donen transparència, si volem).

Aquí l'exemple està fet usant `labs` per posar etiquetes als eixos i títol (per veure noves opcions). Podeu veure com s'utilitzen fent `?labs`.

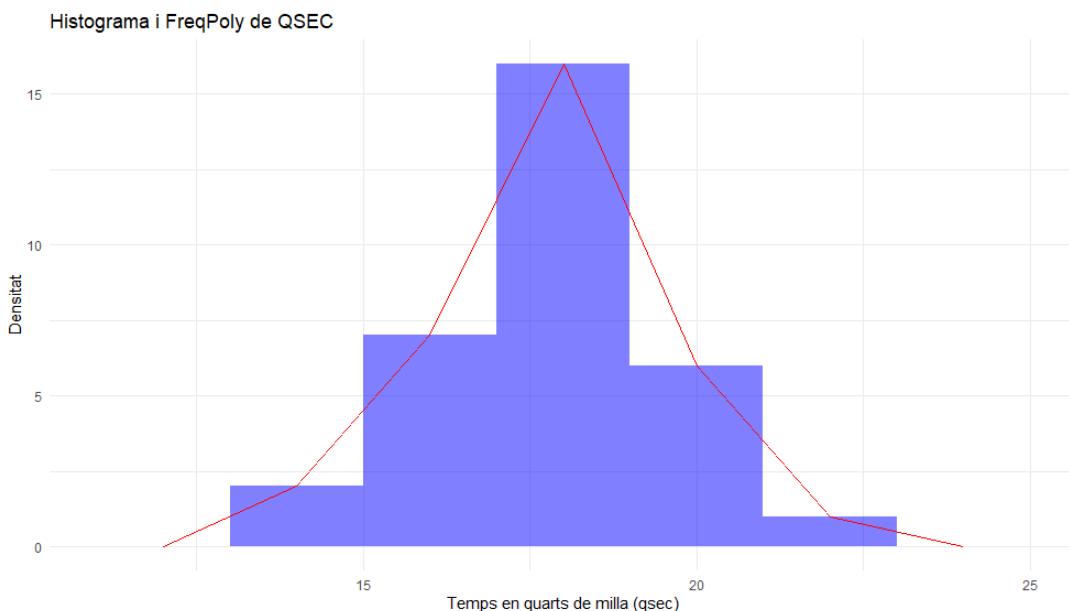
```
>p <- ggplot(mtcars2, aes(x = qsec)) +geom_histogram(binwidth = 2,
fill = "blue", alpha = 0.5) +geom_freqpoly(binwidth = 2, color =
```

```
"red") +labs(title = "Histograma i FreqPoly de QSEC", x = "Temps en quarts de milla (qsec)",y = "Densitat")
```



Veiem que tot i baixar el binwidth a 2, hi ha poques dades. El que podem dir és que l'histograma mostra una moda clara per exemple al voltant de 17-18 qsec, per exemple.

```
>p + theme_minimal()
```



La funció `theme_minimal()` en ggplot2 aplica un tema visual minimalist a el gràfic.
Concretament:

- Elimina els fons de color i els marcs ombrejats.
- Usa una graella lleugera i línies fines per fer el gràfic més net.
- Evita elements innecessaris, fent que el gràfic sigui més lleigble i elegant.

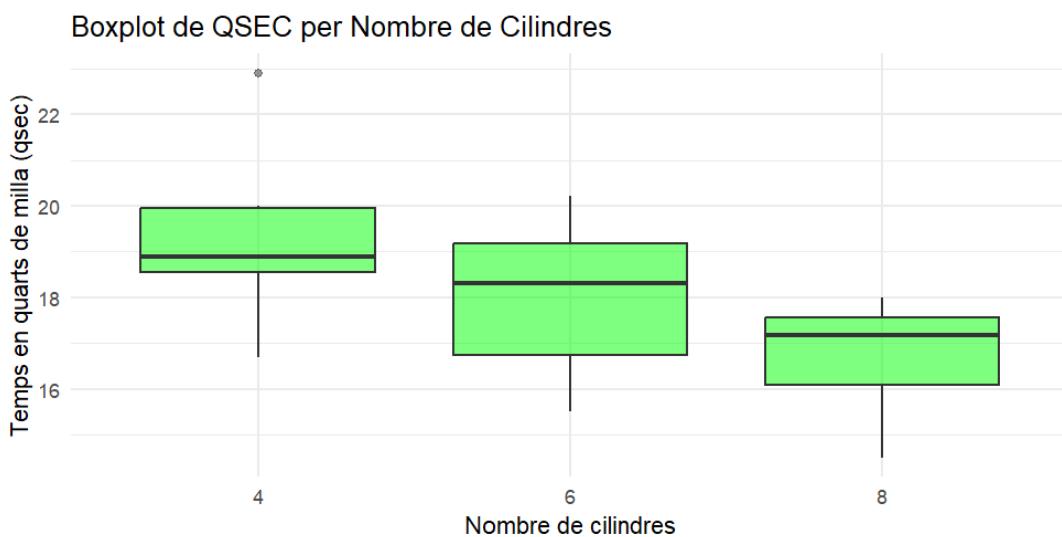
Si vols provar altres estils, també hi ha `theme_classic()`, `theme_light()`, `theme_dark()`, entre d'altres.

4.- Com depèn la distribució del temps en quarts de milla (qsec) segons la cilindrada? Feu una imatge que permeti visualitzar-ho. Imagineu que heu de posar aquesta gràfica en una web corporativa de tons verds, poseu-hi un color verd per tal que estèticament quedí més en el context de la pàgina web.

Ara tenim una variable contínua i una variable discreta categòrica amb 3 categories, o fem tres histogrames, o si ho volem posar en un gràfic tot fem un boxplot.

Ja hem vist com categoritzar la variable ordinal cyl (si no s'ha fet veure PART1). També hem vist en la PART 2 com pintar els boxplots. Per tant, ajuntant tot el que hem après podríem posar, per exemple:

```
>ggplot(mtcars2, aes(x = cyl, y = qsec)) + geom_boxplot(fill = "green", alpha = 0.5) + labs(title = "Boxplot de QSEC per Nombre de Cilindres", x = "Nombre de cilindres", y = "Temps en quarts de milla (qsec)") + theme_minimal()
```



Veiem que tenim només un outlier pel cas de cotxes de 4 cilindres, i les distribucions per cada cilindrada varien bastant. Pel cas dels cotxes de 4 cilindres podem dir que el temps en qsec té una distribució asimètrica negativa, pel cas de 6 cilindres hi ha una mínima asimetria positiva, tot i que està prou centrada, i, finalment, pel cas de 8 cilindres hi ha una clara asimetria positiva

SEMINARI 3.1 Gràfiques Exploratòries (Respostes)

1. OBJECTIUS

Aquest seminari introduceix l'ús de gràfiques exploratòries.

2. PART 1. Diagrames de barres o diagrames de sectors circulars

Com sempre, si obriu R de nou, primer de tot recordeu que heu de tornar a carregar la llibreria tidyverse.

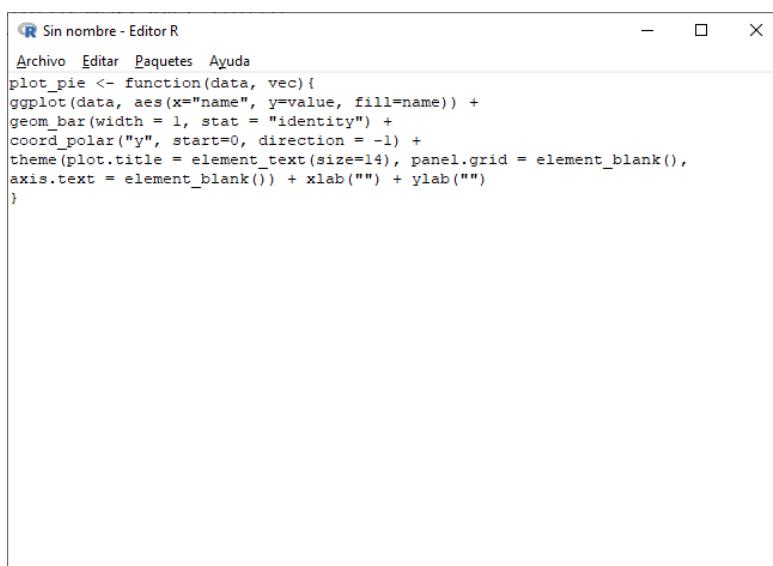
En aquesta primera part del seminari compararem l'ús de `pie()` i `ggplot` per crear un diagrama de sectors. Després compararem diagrames de sectors circulars amb diagrames de barres

1.- Diagrames de sectors circulars amb `ggplot`

a) Feu un script en R que contingui la funció `plot_pie` creada amb `ggplot` següent, i guardeu-la amb el nom `plot_pie.R`:

```
plot_pie <- function(data, vec){  
  ggplot(data, aes(x="name", y=value, fill=name)) +  
    geom_bar(width = 1, stat = "identity") +  
    coord_polar("y", start=0, direction = -1) +  
    theme(plot.title = element_text(size=14), panel.grid =  
      element_blank(),  
      axis.text = element_blank()) + xlab("") + ylab("")  
}
```

Fem [Archivo-> Nuevo script](#) i se'ns obre una pantalla on copiem el text



The screenshot shows the RStudio code editor window titled "Sin nombre - Editor R". The menu bar includes "Archivo", "Editar", "Paquetes", and "Ayuda". The code area contains the following R function:

```
plot_pie <- function(data, vec){  
  ggplot(data, aes(x="name", y=value, fill=name)) +  
    geom_bar(width = 1, stat = "identity") +  
    coord_polar("y", start=0, direction = -1) +  
    theme(plot.title = element_text(size=14), panel.grid = element_blank(),  
      axis.text = element_blank()) + xlab("") + ylab("")  
}
```

Guardem l'arxiu posant com a nom (per exemple) `plot_pie.R` (Recordreu triar en quin directori el guardeu, ja que després haureu de saber el directori per tal de fer-ne us)

b) Creeu tres datasets

```
data1 <- data.frame( name=letters[1:5], value=c(17,18,20,22,24) )
data2 <- data.frame( name=letters[1:5], value=c(20,18,21,20,20) )
data3 <- data.frame( name=letters[1:5], value=c(24,23,21,19,18) )
```



R version 4.0.3 (2020-10-10) -- "Bunny-Wunnies Freak Out"
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R es un software libre y viene sin GARANTIA ALGUNA.
Usted puede redistribuirlo bajo ciertas circunstancias.
Escriba 'license()' o 'licence()' para detalles de distribucion.

R es un proyecto colaborativo con muchos contribuyentes.
Escriba 'contributors()' para obtener más información y
'citation()' para saber cómo citar R o paquetes de R en publicaciones.

Escriba 'demo()' para demostraciones, 'help()' para el sistema on-line de ayuda,
o 'help.start()' para abrir el sistema de ayuda HTML con su navegador.
Escriba 'q()' para salir de R.

[Previously saved workspace restored]

> library (tidyverse)
-- Attaching packages --
v ggplot2 3.3.2 v purrr 0.3.4
v tibble 3.0.4 v dplyr 1.0.2
v tidyverse 1.1.2 v stringr 1.4.0
v readr 1.4.0 vforcats 0.5.0
-- Conflicts --
x dplyr::filter() masks stats::filter()
x dplyr::lag() masks stats::lag()
> data1 <- data.frame(name=letters[1:5], value=c(17,18,20,22,24))
> data2 <- data.frame(name=letters[1:5], value=c(20,18,21,20,20))
> data3 <- data.frame(name=letters[1:5], value=c(24,23,21,19,18))
> |

c) Utilitzant el script que heu creat, dibuixeu els següents gràfics de sectors i guardeu-los en fitxers .png:

- Un pie chart on *data* sigui el primer dataset creat (*data1*) i vec sigui **c(10,35,55,75,93)**
- Un pie chart on *data* sigui el segon dataset creat (*data2*) i vec sigui **c(10,35,53,75,93)**
- Un pie chart on *data* sigui el tercer dataset creat (*data3*) i vec sigui **c(10,29,50,75,93)**

Com heu vist a les diapositives primer hem de cridar el script

```
> source("<name_path>/plot_pie.R") # Recordeu especificar en <name_path>
l'adreça/directori on heu guardat el script
```

Ara només us cal cridar la funció *plot_pie()* com ens indica cada subapartat. Però a més volem guardar els gràfics amb extensió .png. Per tant, com heu vist a les diapositives, per cada subapartat primer crideu a la funció *png()*, després crideu *plot_pie()* i després tanqueu i guardeu amb *dev.off()*. Exemple:

```
> png("<name_path>/pie_data1.png",width=800,height=800) # Actualitzar
path i nom
> plot_pie(data1, c(10,35,55,75,93))
> dev.off()
```

Tindreu tres fitxers .png:

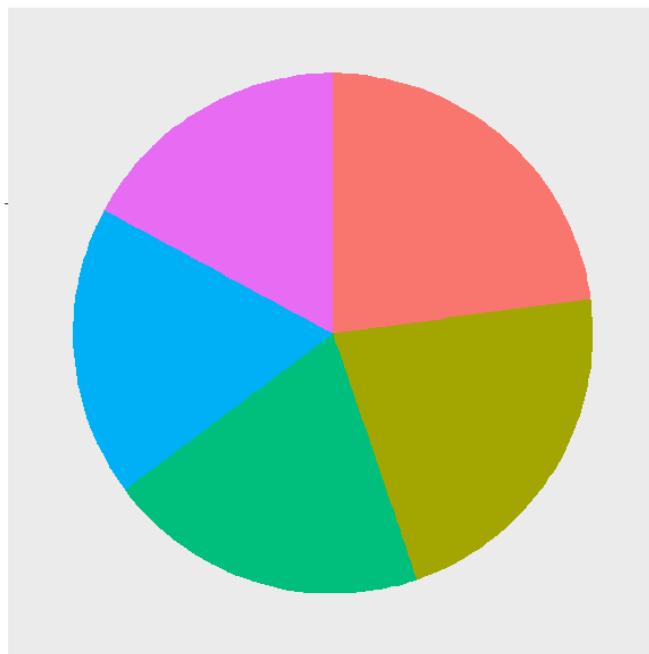
pie_data1.png



pie_data2.png



pie_data3.png



2.- Mostreu la mateixa informació que se us demanava en l'exercici 1.c, però ara utilitzeu un diagrama de barres. Compareu un a un amb el respectiu diagrama de sector circular que heu guardat en l'exercici 1. Si tinguéssiu que triar entre utilitzar diagrames de barres o diagrames de sectors circulars, què utilitzaríeu, per què?

Per cada dataset fem:

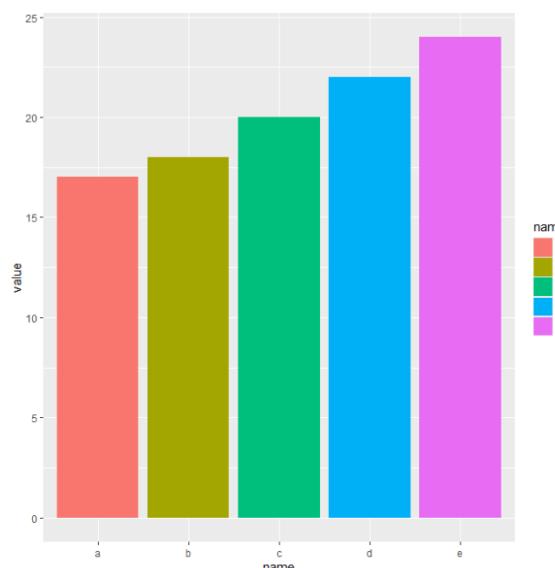
```
> ggplot(data1, aes(x=name, y=value, fill=name))+ geom_bar( stat = "identity")
```

o el que és el mateix en aquest cas on stat= "identity":

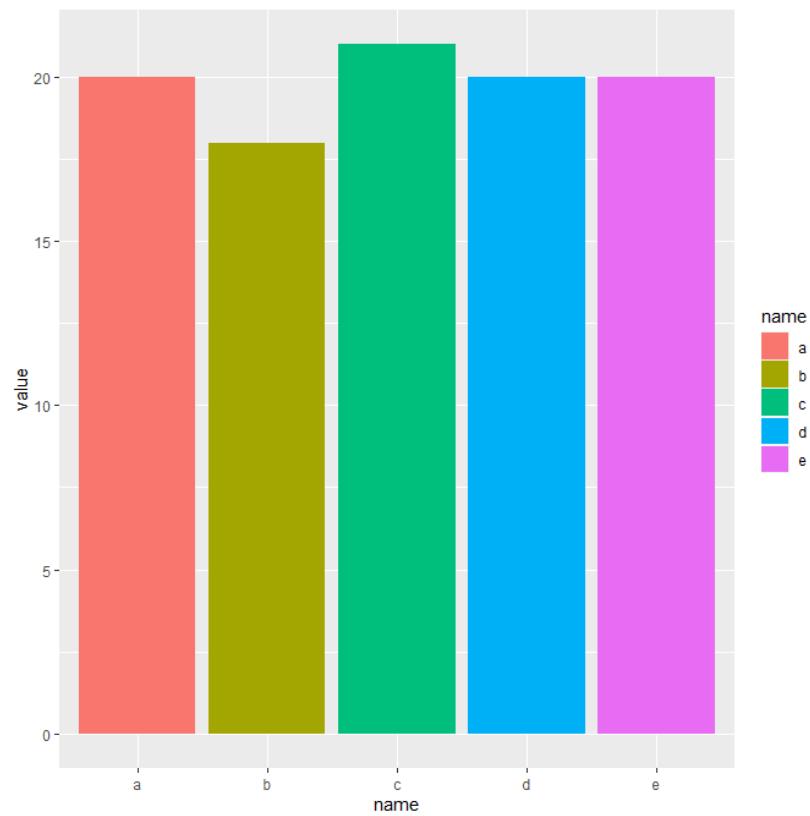
```
> ggplot(data1, aes(x=name, y=value, fill=name))+ geom_col()
```

Obtindrem les gràfiques següents:

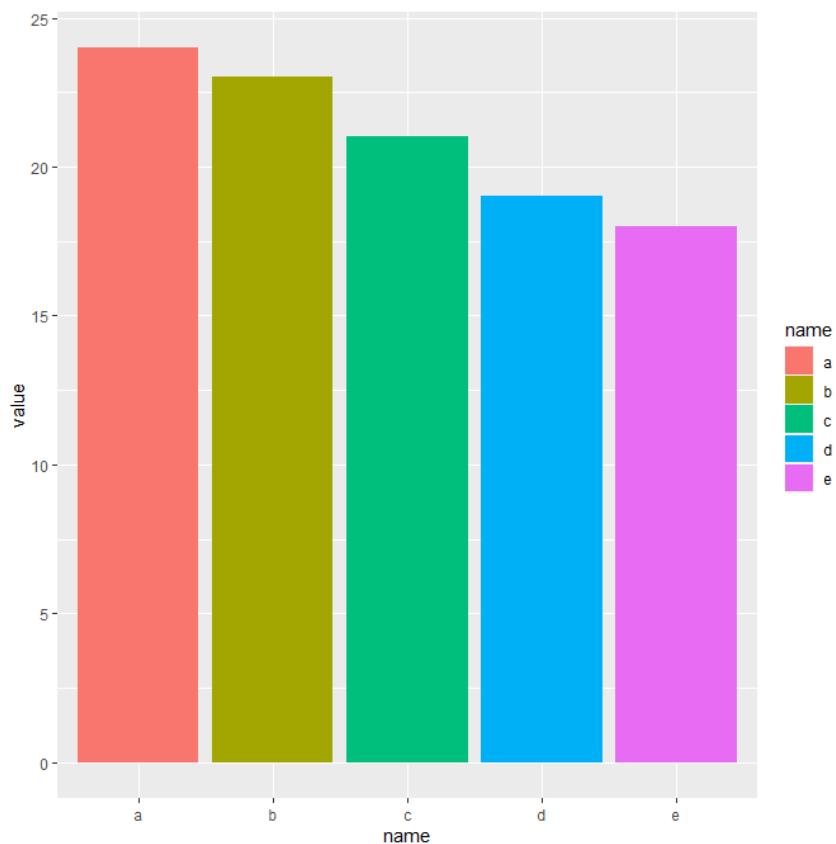
- Per data1



- Pel data2



- Pel data3



En el diagrama de barres veiem diferències significatives que no es veien en els diagrames de sectors circulars.

! L'ull humà és més sensible a diferenciar alçades de barres que no pas a diferenciar entre angles. Només utilitzem diagrames de sectors circulars quan les diferències proporcionals siguin molt grans. En un exemple com el d'aquest exercici, clarament els diagrames de barres ens ajuden més a mostrar la informació que tenim en les nostres dades.

3. PART 2. Distribucions & comparacions

En aquesta segona part del seminari anem a explorar un nou **dataframe** i fer algunes gràfiques tot practicant amb comandes que ja hem vist. També farem us de la nova llibreria **forcats** que hem vist avui

Starwars és un **dataframe** on cada fila és una observació i cada columna és una variable

```
> starwars
# A tibble: 87 x 14
   name      height  mass hair_color skin_color eye_color birth_year sex gender homeworld species films vehicles starships
   <chr>     <int> <dbl> <chr>    <chr>    <dbl> <chr>    <chr> <chr> <list>   <list>   <list>
1 Luke Skywalker 172     77 blond   fair       19 male   masculine Tatooine Human <chr [5]> <chr [2]> <chr [2]>
2 C-3PO          167     75 <NA>    gold      yellow  112 none   masculine Tatooine Droid <chr [6]> <chr [0]> <chr [0]>
3 R2-D2          96      32 <NA>   white, blue red    33 none   masculine Naboo Droid <chr [7]> <chr [0]> <chr [0]>
4 Darth Vader   202    136 none   white      yellow  41.9 male   masculine Tatooine Human <chr [4]> <chr [0]> <chr [1]>
5 Leia Organa   150     49 brown   light     brown   19 female feminine Alderaan Human <chr [5]> <chr [1]> <chr [0]>
6 Owen Lars     178     120 brown, grey light     blue   52 male   masculine Tatooine Human <chr [3]> <chr [0]> <chr [0]>
7 Beru Whitesun lars 165     75 brown   light     blue   47 female feminine Tatooine Human <chr [3]> <chr [0]> <chr [0]>
8 R5-D4          97      32 <NA>   white, red red    NA none   masculine Tatooine Droid <chr [1]> <chr [0]> <chr [0]>
9 Biggs Darklighter 183     84 black   light     brown   24 male   masculine Tatooine Human <chr [1]> <chr [0]> <chr [1]>
10 Obi-Wan Kenobi 182     77 auburn, white fair   blue-gray 57 male   masculine Stewjon Human <chr [6]> <chr [1]> <chr [5]>
# ... with 77 more rows
> |
```

Si volem saber quantes files i columnes tenim, fem:

```
> glimpse(starwars)
```

NOTA: En molts dels exercicis R ens tornarà un *warning*:

Warning message:

```
Removed 6 rows containing non-finite values (stat_density).
```

Veurem més endavant, quan veiem data massatge com treure aquestes files de forma que NO ens surtin *warnings*.

1.- Mostra la distribució de les alçades dels personatges i indica quina és la moda.

a) Quin tipus de gràfica és més evident de les que hem vist?

Height és una variable numèrica contínua. Vam veure que l'histograma era una bona manera de mostrar distribucions d'una variable contínua.

Si fem:

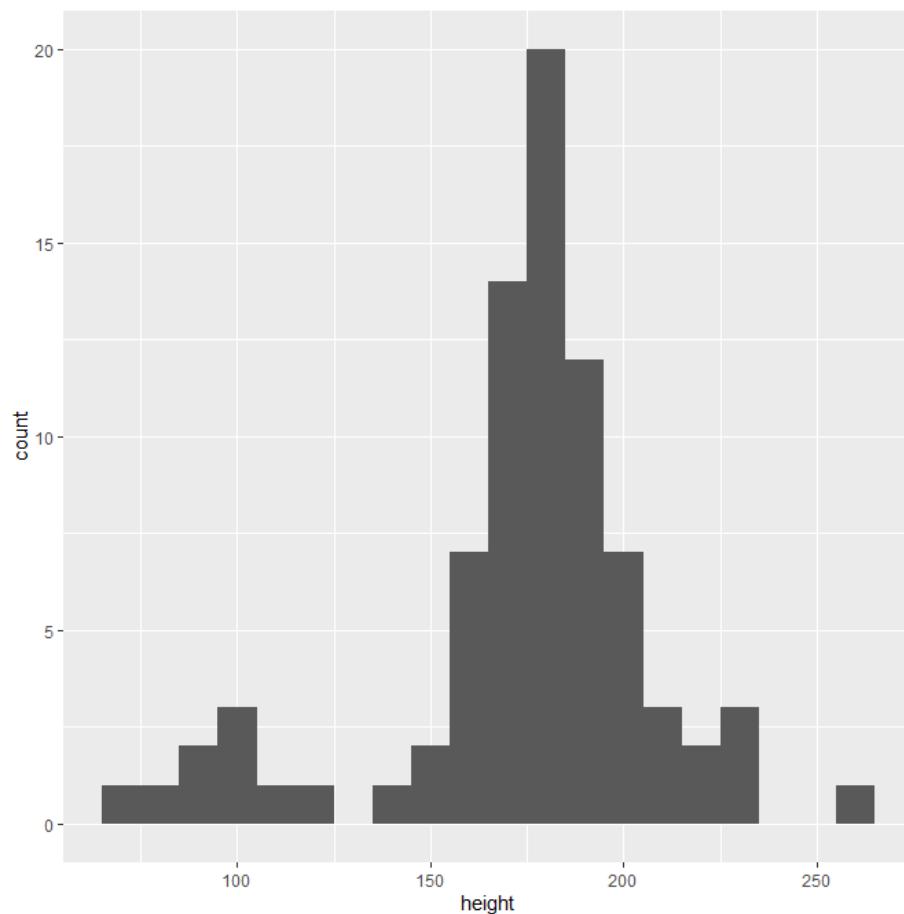
```
> ggplot(starwars,aes(x=height)) +geom_histogram()
```

Ens diu efectivament:

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

Provem amb *binwidth*=10 per exemple (o altre que creguem adients):

```
> ggplot(starwars,aes(x=height))+geom_histogram(binwidth=10)
```

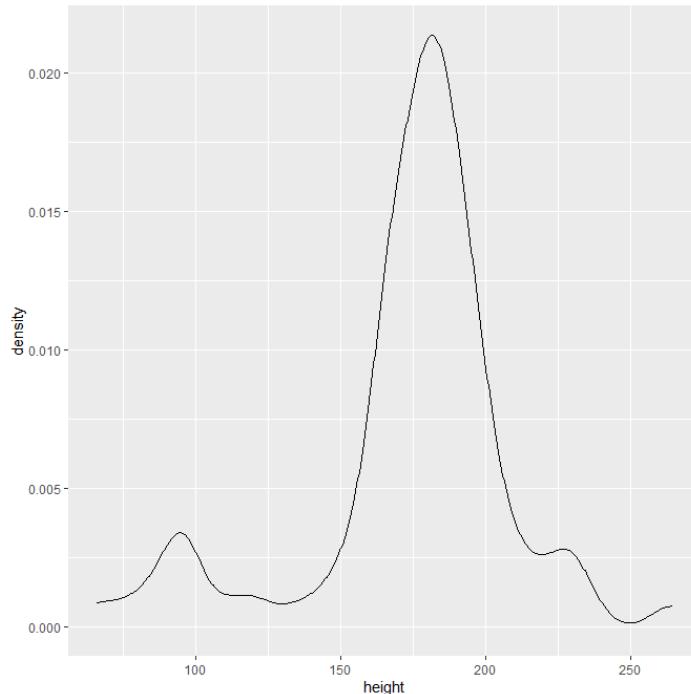


En quant a la moda, veiem que la majoria dels personatges tenen alçades entre 160-190 cm, podria ser degut a que en les primeres pel·lícules de Star Wars no hi havia tants efectes especials, i alguns personatges eren humans disfressats. Segur que podeu intuir en quina part de l'histograma estava algun personatge com en Yoda i/o algun altre personatge.



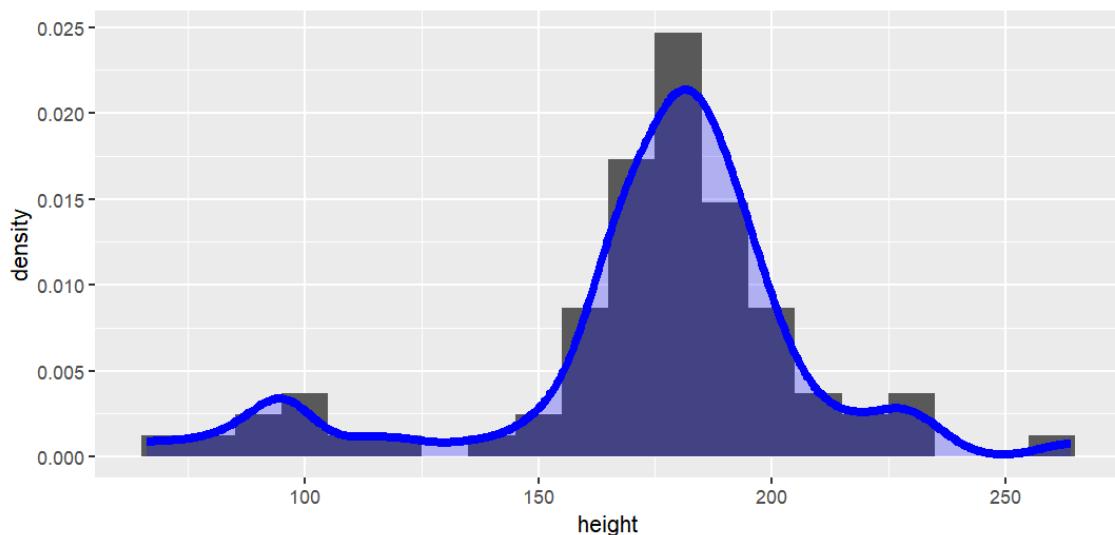
b) Ara proba la `geom_density()` i explica quina informació et dona

```
> ggplot(starwars,aes(x=height)) +geom_density()
Warning message:
Removed 6 rows containing non-finite values (stat_density).
```



Aquí semblaria que hi ha personatges amb alçades entre 230-240 cm mentre en l'histograma veiem que no n'hi ha.

c) Una altra opció és adjuntar ambdues gràfiques en una fent servir una transparència. Per això podeu posar `aes(y=..density..)` en el `geom_histogram`.



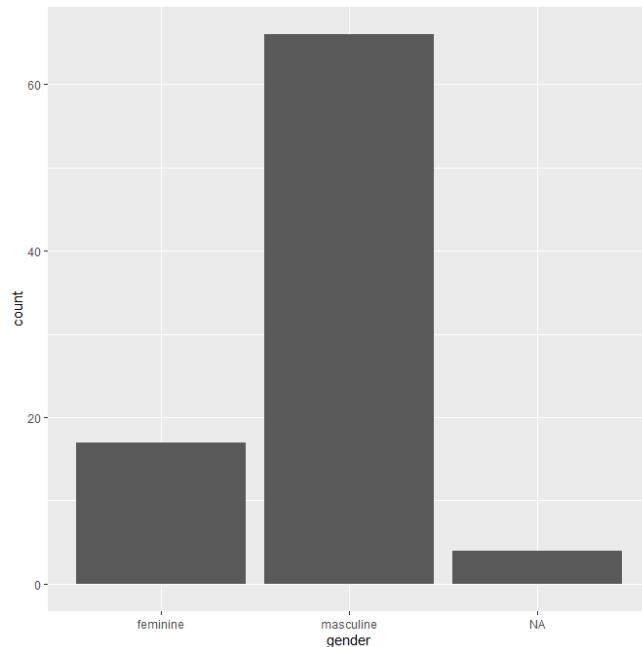
```
>ggplot(starwars,aes(x=height))+geom_histogram(binwidth=10, aes(y=..density..))+geom_density(lwd = 2, colour = 'blue', fill = 'blue', alpha = 0.25)
```

Nota: `lwd` només marca el gruix de la línia de `geom_density`

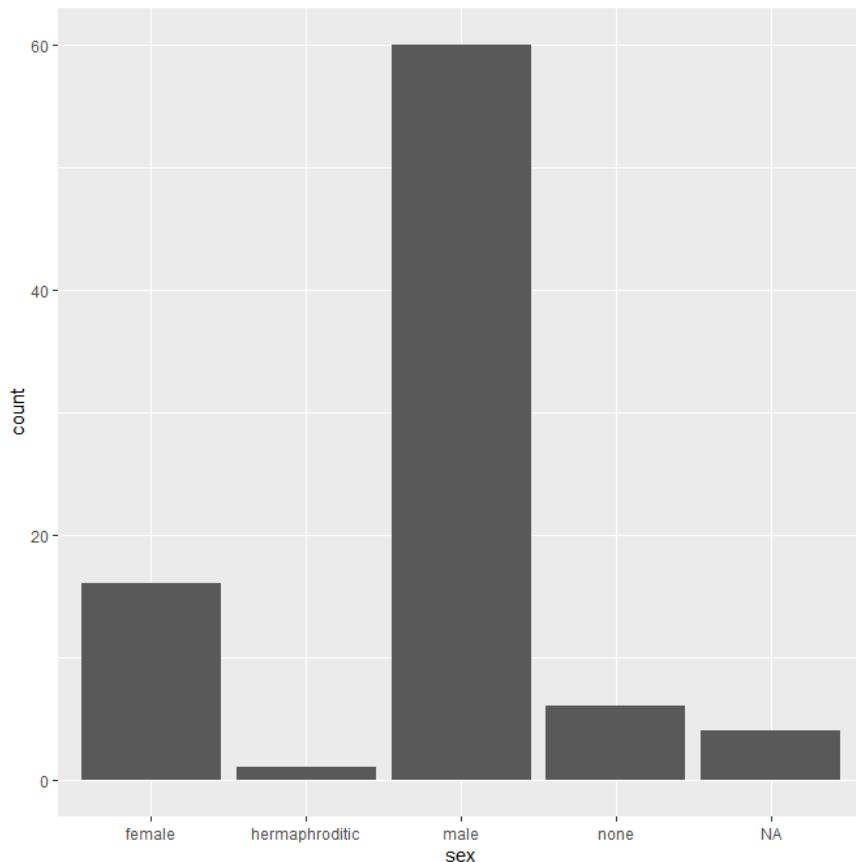
2.- Mostra la proporció dels diferents gèneres dels personatges de starwars

Gender és una variable discreta qualitativa. Volem comparar quants personatges hi ha de cada gènere, per tant sembla adient fer un diagrama de barres

```
>ggplot(starwars,aes(x=gender))+geom_bar()
```



```
>ggplot(starwars,aes(x=sex))+geom_bar()
```



Hi ha menys actors amb sexe masculí que amb gènere masculí. El gènere si feu **?starwars** veureu que es refereix al rol que adapten els personatges. Personatges com R2-D2 no tenen sexe però adapten un gènere masculí.

3.- Mostra la distribució de les alçades versus el gènere

a) Quin gènere té un alçada mitjana més alta?

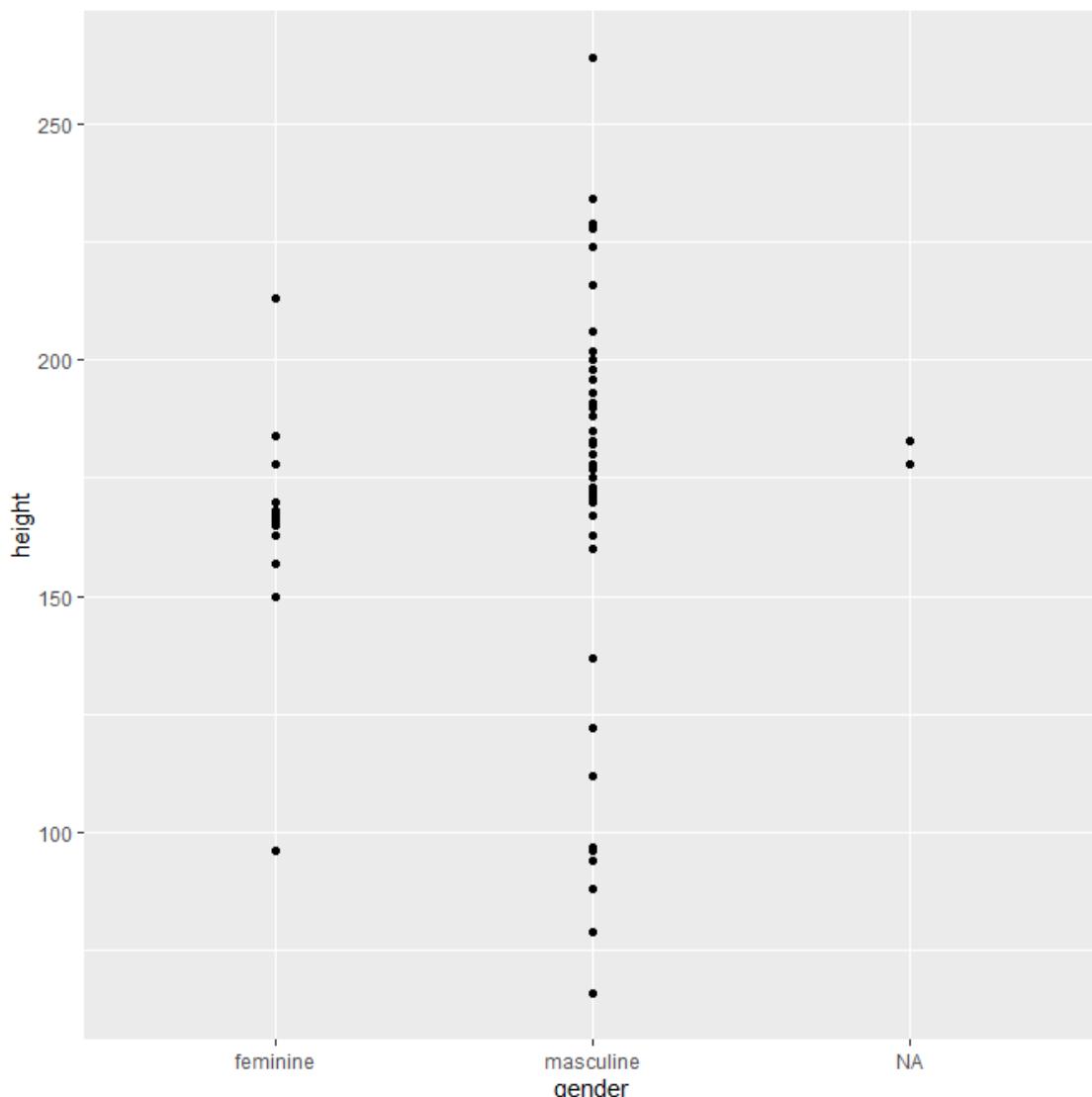
b) De quin gènere és el personatge més alt?

height és una variable numèrica contínua. gender és una variable discreta-categòrica. Si fem un scatter plot, on l'eix x representi el gènere i l'eix y l'alçada, la visualització no ens dona gaire informació:

```
> ggplot(starwars,aes(x=gender,y=height))+geom_point()
```

Warning message:

Removed 6 rows containing missing values (geom_point).

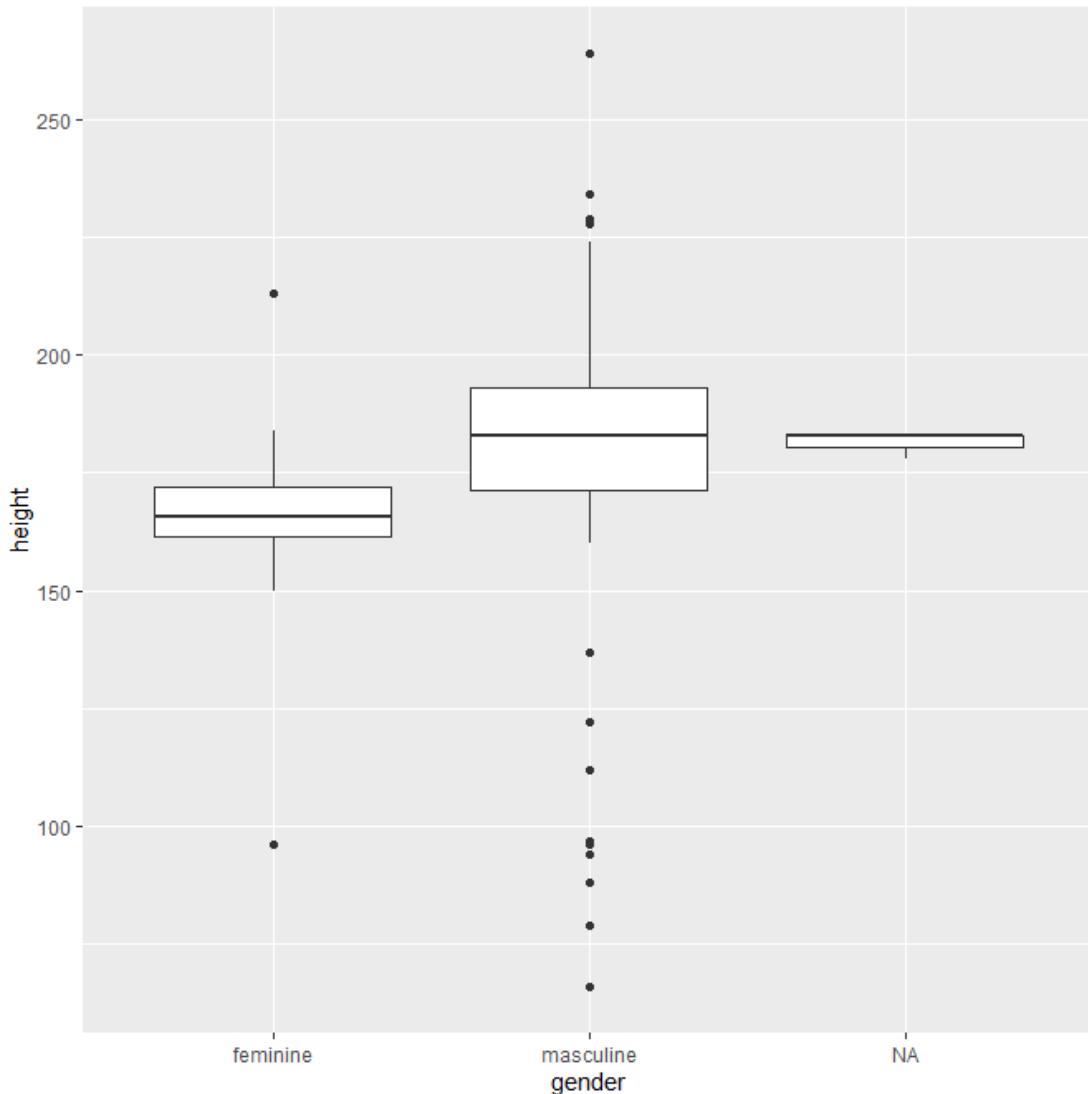


Si fem un scatter plot, utilitzant `geom_point()`, contestar la pregunta b és fàcil (també ho era pels seguidors de Star Wars o els que ja heu vist la imatge que us he posat en la resposta de l'exercici 1). El personatge més alt és de gènere masculí. Però contestant

l'apartat a ens equivocaríem segurament si no féssim un diagrama de caixes (boxplot). A més el boxplot ens dóna molta més informació sobre la distribució de les alçades versus el gènere.

Per tant:

```
> ggplot(starwars,aes(x=gender,y=height))+geom_boxplot()
Warning message:
Removed 6 rows containing non-finite values (stat_boxplot).
```



I curiosament la mitjana del gènere masculí i la de NA és la mateixa (cosa que no es veia en el scatterplot)

4.- Mostra la proporció dels diferents colors d'ulls dels personatges de starwars

eye_color és una variable discreta. Volem comparar quants personatges hi ha de cada color d'ulls, per tant sembla adient fer un diagrama de barres. Ara bé si ho fem:

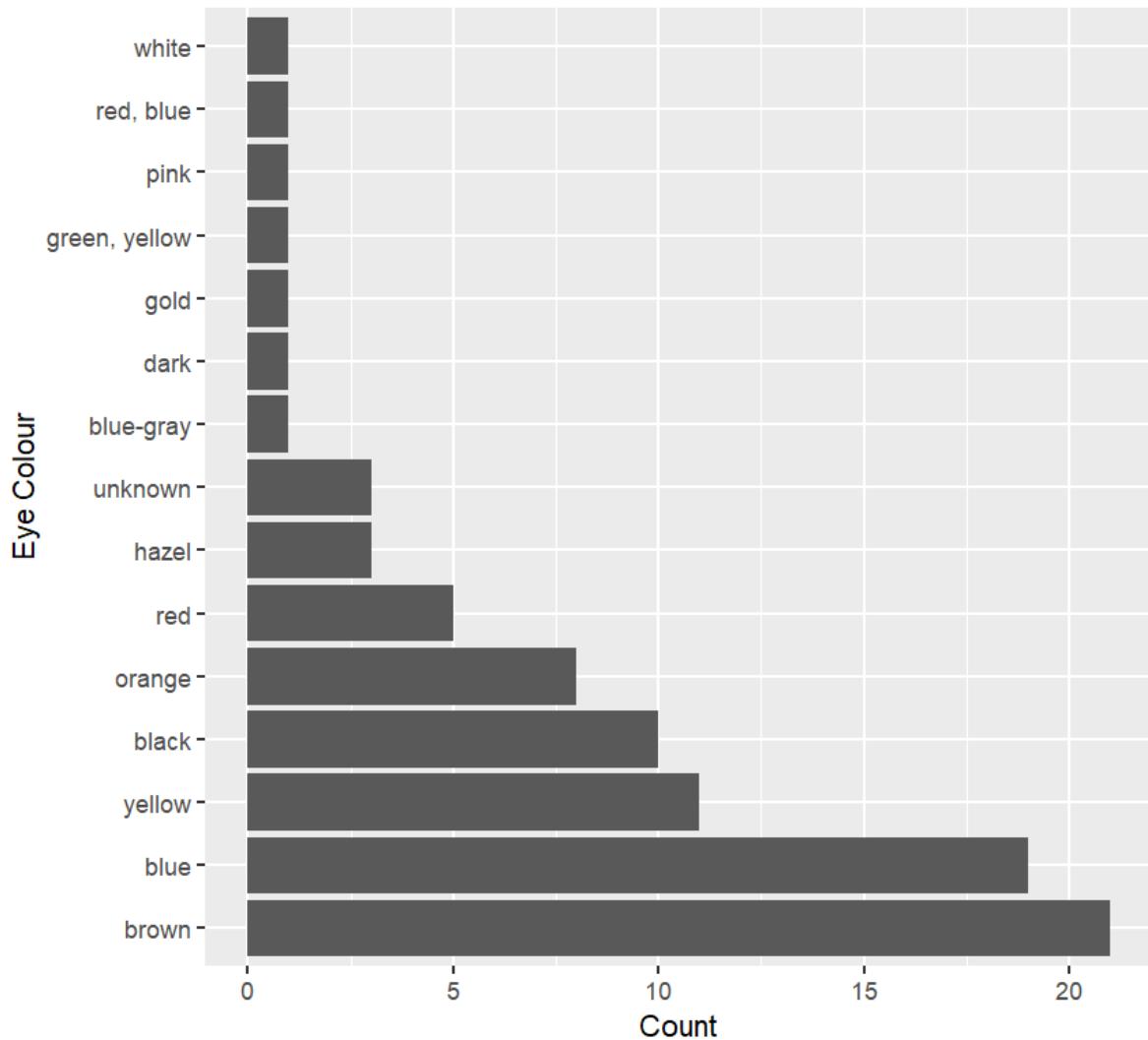
```
> ggplot(starwars, mapping = aes(x = eye_color)) + geom_bar()
```

Veure'm que se'ns amunteguen els noms de les variables de color d'ulls a l'eix x.

Sembla adient fer un `coord_flip()`, com heu vist en les diapositives, i reordenar el diagrama de barres per freqüències amb la llibreria `forcats` que heu vist avui. **NOTA:** És com l'exemple que heu vist a les diapositives del color de cabell (que també podeu refer).

```
> library(forcats) #si heu carregat tidyverse ja hi és, si sol heu
carregat ggplot2 l'haurieu de carregar.

> ggplot(starwars, aes(x = fct_infreq(eye_color))) +
geom_bar() + labs(x='Eye Colour', y='Count') + coord_flip()
```



Els dos colors majoritaris són el marró seguit del blau. Un color minoritari podria ser el rosa.

NOTA: Com sempre, aquest solucionari presenta una solució, no vol dir que per alguns casos, no hi hagi més solucions adequades.

SEMINARI 3.2. *Data Massaging* (Respostes)

1. OBJECTIUS

Aquest seminari introduceix al *Data Massaging*.

2. PART 1. *Data Massaging*

En aquesta primera part del seminari anem a fer alguns canvis en el dataset *starwars* que ja vam explorar en el seminari 3 de la llibreria tidyverse. Si heu obert R de nou, recordeu carregar la llibreria tidyverse.

Starwars és un dataset on cada fila és una observació i cada columna és una variable

```
> starwars
# A tibble: 87 x 14
   name        height  mass hair_color skin_color eye_color birth_year sex   gender homeworld species films vehicles starships
   <chr>      <int> <dbl> <chr>     <chr>      <chr>      <dbl> <chr>    <chr> <chr> <list>   <list>   <list>
 1 Luke Skywalker    172    77 blond    fair       blue      19 male   masculine Tatooine Human  <chr [5]> <chr [2]> <chr [2]>
 2 C-3PO             167    75 <NA>     gold      yellow     112 none   masculine Tatooine Droid  <chr [6]> <chr [0]> <chr [0]>
 3 R2-D2              96     32 <NA>     white, blue red     33 none   masculine Tatooine Droid  <chr [7]> <chr [0]> <chr [0]>
 4 Darth Vader      202    136 none     white     yellow     41.9 male   masculine Tatooine Human  <chr [4]> <chr [0]> <chr [1]>
 5 Leia Organa       150     49 brown    light      brown     19 female feminine Alderaan Human  <chr [5]> <chr [1]> <chr [0]>
 6 Owen Lars          178    120 brown, grey light      blue      52 male   masculine Tatooine Human  <chr [3]> <chr [0]> <chr [0]>
 7 Beru Whitesun lars 165     75 brown    light      blue      47 female feminine Tatooine Human  <chr [3]> <chr [0]> <chr [0]>
 8 R5-D4              97     32 <NA>     white, red red     NA none   masculine Tatooine Droid  <chr [1]> <chr [0]> <chr [0]>
 9 Biggs Darklighter  183     84 black    light      brown     24 male   masculine Tatooine Human  <chr [1]> <chr [0]> <chr [1]>
10 Obi-Wan Kenobi     182    77 auburn, white fair    blue-gray    57 male   masculine Stewjon Human  <chr [6]> <chr [1]> <chr [5]>
# ... with 77 more rows
> |
```

Primer ens familiaritzarem amb algunes funcions de la llibreria *dplyr* com:

- **select:** Seleccionar columnes
- **filter:** Filtrar
- **arrange:** Ordenar un conjunt de dades
- **group_by:** Agrupar per alguna variable
- **summarize/summarise:** Especificar algunes funciones d'agregats:
 - **n()** : comptar;
 - **sum()** : sumar variables numèriques;
 - **mean()** : la mitjana de variables numèriques entre altres
- **mutate:** modificar, transformar o agregar variables del conjunt de dades
- **pipes %>%:** combinar operacions

EXERCICIS:

En primer lloc, carreguem la llibreria *dplyr*

```
> library(tidyverse)
> library(dplyr)
```

1.- Seleccioneu només el nom i gènere del conjunt de dades *starwars*

Primer cridem les llibreries i després fem us de la funció `select()`, ja que estem seleccionant variables/columnes de la taula.



select(.data, ...)
Extract columns as a table. Also **select_if()**.
`select(iris, Sepal.Length, Species)`

Use these helpers with select (),
e.g. `select(iris, starts_with("Sepal"))`

<code>contains(match)</code>	<code>num_range(prefix, range)</code>	; e.g. <code>mpg:cyl</code>
<code>ends_with(match)</code>	<code>one_of(...)</code>	-; e.g. <code>-Species</code>
<code>matches(match)</code>	<code>starts_with(match)</code>	

```
> select(starwars, name, gender)
# A tibble: 87 x 2
  name           gender
  <chr>          <chr>
1 Luke Skywalker masculine
2 C-3PO          masculine
3 R2-D2          masculine
4 Darth Vader   masculine
5 Leia Organa   feminine
6 Owen Lars     masculine
7 Beru Whitesun lars feminine
8 R5-D4          masculine
9 Biggs Darklighter masculine
10 Obi-Wan Kenobi masculine
# ... with 77 more rows
> |
```

2.- Seleccioneu els personatges que són humans. Després seleccioneu els que no

En la columna *species*, trobem algunes files que són “Human”. Per tant, seleccionem les observacions/files amb la funció *filter* com hem vist a classe

Manipulate Cases

EXTRACT CASES

Row functions return a subset of rows as a new table.



filter(.data, ...) Extract rows that meet logical criteria. `filter(iris, Sepal.Length > 7)`

```
> filter(starwars, species=="Human")
> #or also:
> starwars %>% filter(species=="Human")
```

Al primer seminari vam veure que les cadenes de caràcters podien escriure's entre “Human” o ‘Human’.

```
> starwars %>% filter(species=="Human")
# A tibble: 35 x 14
  name     height mass hair_color   skin_color eye_color birth_year sex gender homeworld species films vehicles starships
  <chr>    <int> <dbl> <chr>      <chr>      <chr>      <dbl> <chr>  <chr>  <chr>  <list>  <list>  <list>
1 Luke Skywalker    172    77 blond     fair       blue        19 male   masculine Tatooine Human  <chr [5]> <chr [2]> <chr [2]>
2 Darth Vader       202    136 none      white      yellow      41.9 male   masculine Tatooine Human  <chr [4]> <chr [0]> <chr [1]>
3 Leia Organa        150     49 brown    light      brown       19 female feminine Alderaan Human  <chr [5]> <chr [1]> <chr [0]>
4 Owen Lars          178    120 brown, grey light      blue        52 male   masculine Tatooine Human  <chr [3]> <chr [0]> <chr [0]>
5 Beru Whitesun lars 165     75 brown    light      blue        47 female feminine Tatooine Human  <chr [3]> <chr [0]> <chr [0]>
6 Biggs Darklighter  183     84 black    light      brown       24 male   masculine Tatooine Human  <chr [1]> <chr [0]> <chr [1]>
7 Obi-Wan Kenobi    182     77 auburn, white fair      blue-gray    57 male   masculine Stewjon Human  <chr [6]> <chr [1]> <chr [5]>
8 Anakin Skywalker    188     84 blond    fair       blue        41.9 male   masculine Tatooine Human  <chr [3]> <chr [2]> <chr [3]>
9 Wilhuff Tarkin     180     NA auburn, grey fair       blue        64 male   masculine Eriadu Human  <chr [2]> <chr [0]> <chr [0]>
10 Han Solo          180     80 brown    fair       brown       29 male   masculine Corellia Human  <chr [4]> <chr [0]> <chr [2]>
# ... with 25 more rows
> |
```

Per veure els que *no*, utilitzem la negació en R !

Logical and boolean operators to use with filter()

<	<=	is.na()	%in%		xor()
>	>=	is.na()	!	&	

See `?base::Logic` and `?Comparison` for help.

```
> filter(starwars, !species=="Human")

> #or:

> starwars %>% filter(!species=="Human") #alternatively

> #or:

> starwars %>% filter(species!="Human")

> #or:

> filter(starwars, species!="Human")

> starwars %>% filter(!species=="Human")
# A tibble: 48 x 14
  name     height mass hair_color skin_color   eye_color birth_year sex gender homeworld species films vehicles starships
  <chr>    <int> <dbl> <chr>      <chr>      <chr>      <dbl> <chr>  <chr>  <chr>  <list>  <list>  <list>
1 C-3PO      167    75 <NA>     gold       yellow      112 none   masculine Tatooine Droid  <chr [6]> <chr [0]> <chr [0]>
2 R2-D2       96     32 <NA>     white, blue red        33 none   masculine Naboo Droid  <chr [7]> <chr [0]> <chr [0]>
3 R5-D4       97     32 <NA>     white, red red        NA none   masculine Tatooine Droid  <chr [1]> <chr [0]> <chr [0]>
4 Chewbacca   228    112 brown   unknown    blue        200 male   masculine Kashyyyk Wookiee <chr [5]> <chr [1]> <chr [2]>
5 Greedo      173    74 <NA>     green      black       44 male   masculine Rodia Rodian <chr [1]> <chr [0]> <chr [0]>
6 Jabba the Hutt 175  1358 <NA>   green-tan, brown orange    600 hermaphroditic masculine Nal Hutta Hutt <chr [3]> <chr [0]> <chr [0]>
7 Yoda         66     17 white    green      brown       896 male   masculine <NA> Yoda's species <chr [5]> <chr [0]> <chr [0]>
8 IG-88        200    140 none    metal      red        15 none   masculine <NA> Droid  <chr [1]> <chr [0]> <chr [0]>
9 Bossk        190    113 none    green      red        53 male   masculine Trandoshan Trandoshan <chr [1]> <chr [0]> <chr [0]>
10 Ackbar      180     83 none   brown mottle orange    41 male   masculine Mon Cala Mon Calamari <chr [2]> <chr [0]> <chr [0]>
# ... with 38 more rows
> |
```

3.- Seleccioneu només el nom i gènere dels personatges amb gènere femení

- Pas a pas:** Assigneu primer a una variable temporal (per exemple de nom `dadesub`) la selecció feta en l'exercici 1. Després seleccioneu del subconjunt `dadesub` les observacions/files que corresponen al gènere femení i assigneu el subconjunt resultant a una altra variable (per exemple de nom `femenins`)
- Feu ús de les pipes** que hem vist a classe, per tal de combinar les dues operacions fetes en l'apartat a. Assigneu el subconjunt resultant a una variable (per exemple de nom `feminisub`) i verifiqueu que correspon al mateix subconjunt assignat a `femenins` en l'apartat a). NOTA: Sempre que puguem utilitzarem **pipes** com en aquest apartat, és a dir, no crearem variables temporals extres com em fet en l'apartat a).

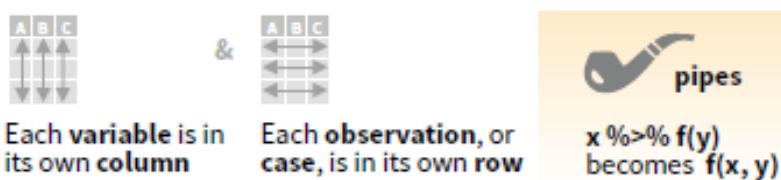
a) Assignem a la variable temporal `dadesub` el subconjunt de dades resultant de la selecció feta en l'exercici 1. Després seleccionem les observacions/files amb la funció `filter` com hem vist a l'exercici 2. Fixem-nos que `filter` selecciona files/observacions que satisfan un cert criteri lògic (en aquest cas seria com pensar `if gender=='feminine'`)

```
> dadesub <- select(starwars, name, gender)
> femenins <- filter(dadesub, gender=='feminine')
> femenins

> femenins <- filter(dadesub, gender=='feminine')
> femenins
# A tibble: 17 x 2
  name      gender
  <chr>    <chr>
1 Leia Organa  feminine
2 Beru Whitesun Lars  feminine
3 Mon Mothma  feminine
4 Shmi Skywalker  feminine
5 Ayla Secura  feminine
6 Adi Gallia  feminine
7 Cordé  feminine
8 Luminara Unduli  feminine
9 Barriss Offee  feminine
10 Dormé  feminine
11 Zam Wesell  feminine
12 Taun We  feminine
13 Jocasta Nu  feminine
14 R4-P17  feminine
15 Shaak Ti  feminine
16 Rey  feminine
17 Padmé Amidala  feminine
> |
```

b. Hem vist que podíem seleccionar variables (columnes) i observacions (o files) fent us de *pipes* seguint:

dplyr functions work with pipes and expect tidy data. In tidy data:



Per tant combinem la selecció i el filtratge utilitzant *pipes* de la següent manera:

```
> feminisub <- starwars %>% filter(gender=='feminine')
%>%select(name,gender)
> feminisub
```

En aquest cas, podem intercanviar el filtratge i la selecció obtenint el mateix resultat:

```
> starwars %>%select(name,gender)%>%filter(gender=='feminine')
```

```
> feminisub <- starwars %>% filter(gender=="feminine") %>% select(name, gender)
> feminisub
# A tibble: 17 x 2
  name      gender
  <chr>    <chr>
1 Leia Organa  feminine
2 Beru Whitesun lars  feminine
3 Mon Mothma  feminine
4 Shmi Skywalker  feminine
5 Ayla Secura  feminine
6 Adi Gallia  feminine
7 Cordé      feminine
8 Luminara Undulí  feminine
9 Barriss Offee  feminine
10 Dormé     feminine
11 Zam Wesell  feminine
12 Taun We   feminine
13 Jocasta Nu  feminine
14 R4-P17    feminine
15 Shaak Ti  feminine
16 Rey       feminine
17 Padmé Amidala  feminine
> |
```

4.- Ordeneu els noms dels personatges femenins en ordre descendente. Podríeu fer-ho sense fer us de les variables temporals creades en exercici anterior (és a dir, sense fer ús del subgrup)?

Hem vist :

ARRANGE CASES



`arrange(.data, ...)` Order rows by values of a column or columns (low to high), use with `desc()` to order from high to low.
`arrange(mtcars, mpg)`
`arrange(mtcars, desc(mpg))`

Per tant:

```
> arrange(feminisub,desc(name)) #partint del subgrup creat
```

```
> arrange(feminisub,desc(name))
# A tibble: 17 x 2
  name      gender
  <chr>    <chr>
1 Zam Wesell  feminine
2 Taun We   feminine
3 Shmi Skywalker  feminine
4 Shaak Ti  feminine
5 Rey       feminine
6 R4-P17    feminine
7 Padmé Amidala  feminine
8 Mon Mothma  feminine
9 Luminara Undulí  feminine
10 Leia Organa  feminine
11 Jocasta Nu  feminine
12 Dormé     feminine
13 Cordé      feminine
14 Beru Whitesun lars  feminine
15 Barriss Offee  feminine
16 Ayla Secura  feminine
17 Adi Gallia  feminine
> |
```

Per tal de no fer us del subgrup `feminisub`, altre cop hauríem d'utilitzar les *pipes*:

```
> ordered_fem <- starwars %>%filter(gender=="feminine")
%>%select(name,gender) %>%arrange(desc(name))
> ordered_fem

> ordered_fem <- starwars %>% filter(gender=="feminine") %>%select(name,gender) %>%arrange(desc(name))
> ordered_fem
# A tibble: 17 x 2
  name           gender
  <chr>          <chr>
1 Zam Wesell    feminine
2 Taun We       feminine
3 Shmi Skywalker feminine
4 Shaak Ti      feminine
5 Rey            feminine
6 R4-P17         feminine
7 Padmé Amidala feminine
8 Mon Mothma    feminine
9 Luminara Unduli feminine
10 Leia Organa   feminine
11 Jocasta Nu    feminine
12 Dormé          feminine
13 Cordé          feminine
14 Beru Whitesun lars feminine
15 Barriss Offee feminine
16 Ayla Secura   feminine
17 Adi Gallia    feminine
> |
```

Judit Chamorro Servent

Bellaterra, Març 2025

SEMINARI 4. *Data Massaging & Advanced Systems I*

(Respostes)

1. OBJECTIUS

Aquest seminari continua amb el *Data Massaging* que vam començar en el seminari 3 i la seva importància a l'hora de visualitzar les dades. A més començarem a familiaritzar-nos amb alguns gràfics més avançats com els multi-panel, i veure'm com mostrar algunes incertituds.

2. PART 1. *Data Massaging (Dplyr)*

En aquesta primera part del seminari anem a fer alguns canvis en el dataset *starwars* que ja vam explorar en el seminari 3 de la llibreria tidyverse. Si heu obert R de nou, recordeu carregar la llibreria tidyverse.

Starwars és un dataframe ordenat on cada fila és una observació i cada columna és una variable

```
> starwars
# A tibble: 87 x 14
   name    height  mass hair_color skin_color eye_color birth_year sex gender homeworld species films vehicles starships
   <chr>     <dbl> <dbl> <chr>       <chr>       <chr>      <dbl> <chr> <chr>       <chr> <list>   <list>   <list>
1 Luke Skywalker     172     77 blond    fair        blue        19 male   masculine Tatooine Human <chr [5]> <chr [2]> <chr [2]>
2 C-3PO              167     75 <NA>       gold        yellow      112 none   masculine Tatooine Droid <chr [6]> <chr [0]> <chr [0]>
3 R2-D2              96      32 <NA>       white, blue red       33 none   masculine Naboo Droid <chr [7]> <chr [0]> <chr [0]>
4 Darth Vader        202    136 none       white       yellow      41.9 male   masculine Tatooine Human <chr [4]> <chr [0]> <chr [1]>
5 Leia Organa         150     49 brown    light       brown       19 female feminine Alderaan Human <chr [5]> <chr [1]> <chr [0]>
6 Owen Lars           178    120 brown, grey light       blue       52 male   masculine Tatooine Human <chr [3]> <chr [0]> <chr [0]>
7 Beru Whitesun lars 165     75 brown    light       blue       47 female feminine Tatooine Human <chr [3]> <chr [0]> <chr [0]>
8 R5-D4              97      32 <NA>       white, red red       NA none   masculine Tatooine Droid <chr [1]> <chr [0]> <chr [0]>
9 Biggs Darklighter   183     84 black    light       brown      24 male   masculine Tatooine Human <chr [1]> <chr [0]> <chr [1]>
10 Obi-Wan Kenobi    182     77 auburn, white fair      blue-gray      57 male   masculine Stewjon Human <chr [6]> <chr [1]> <chr [5]>
# ... with 77 more rows
> |
```

Primer ens familiaritzarem amb algunes funcions de la llibreria *dplyr* com:

- **select**: Seleccionar columnes
- **filter**: Filtrar
- **arrange**: Ordenar un conjunt de dades
- **group_by**: Agrupar per alguna variable
- **summarize/summarise**: Especificar algunes funciones d'agregats:
 - **n()** : comptar;
 - **sum()** : sumar variables numèriques;
 - **mean()** : la mitjana de variables numèriques entre altres
- **mutate**: modificar, transformar o agregar variables del conjunt de dades
- **pipes %>%**: combinar operacions

EXERCICIS:

En primer lloc, carreguem la llibreria *dplyr*

```
> library(tidyverse)
> library(dplyr)
```

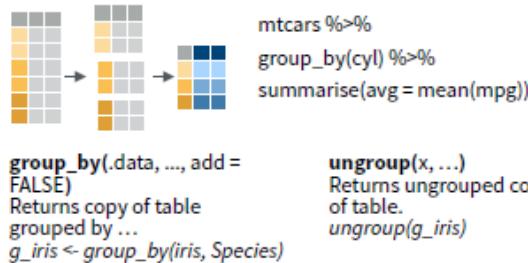
1.- Agrupaments

a) Agrupeu els personatges per gènere

Utilitzarem la funció group_by. Aquesta funció és molt semblant al GROUP_BY del llenguatge SQL de BBDD.

Group Cases

Use `group_by()` to create a "grouped" copy of a table.
`dplyr` functions will manipulate each "group" separately and then combine the results.



```
> GrupXgenere <- group_by(starwars, gender)
> GrupXgenere
```

```
> GrupXgenere <- group_by(starwars, gender)
> GrupXgenere
# A tibble: 87 x 14
# Groups:   gender [3]
  name      height  mass hair_color   skin_color eye_color birth_year sex   gender homeworld species films vehicles starships
  <chr>     <dbl> <dbl> <chr>       <chr>      <chr>        <dbl> <chr> <chr>    <chr> <chr> <list> <list>
1 Luke Skywalker    172    77 blond      fair       blue          19 male   masculine Tatooine Human  <chr [5]> <chr [2]> <chr [2]>
2 C-3PO             167    75 <NA>      gold       yellow        112 none  masculine Tatooine Droid  <chr [6]> <chr [0]> <chr [0]>
3 R2-D2              96     32 <NA>      white, blue red          33 none  masculine Naboo  Droid  <chr [7]> <chr [0]> <chr [0]>
4 Darth Vader       202    136 none      white       yellow        41.9 male  masculine Tatooine Human  <chr [4]> <chr [0]> <chr [1]>
5 Leia Organa        150    49 brown      light      brown         19 female feminine Alderaan Human  <chr [5]> <chr [1]> <chr [0]>
6 Owen Lars           178    120 brown, grey light      blue          52 male   masculine Tatooine Human  <chr [3]> <chr [0]> <chr [0]>
7 Beru Whitesun lars 165     75 brown      light      blue          47 female feminine Tatooine Human  <chr [3]> <chr [0]> <chr [0]>
8 R5-D4              97     32 <NA>      white, red red          NA none  masculine Tatooine Droid  <chr [1]> <chr [0]> <chr [0]>
9 Biggs Darklighter   183    84 black      light      brown         24 male   masculine Tatooine Human  <chr [1]> <chr [0]> <chr [1]>
10 Obi-Wan Kenobi     182    77 auburn, white fair      blue-gray      57 male   masculine Stewjon Human  <chr [6]> <chr [1]> <chr [5]>
# ... with 77 more rows
> |
```

Es pot veure que el resultat ens dona el nombre de grups en la segona línia del missatge de sortida. Si per cada grup volem aplicar algun càcul, cal que definim funcions d'agregat associades (`count()`, `mean()`, `min()`, `max()`, `sum()`), com en el llenguatge SQL de Bases de Dades.

b) Compteu quants personatges hi ha de cada grup utilitzant les *pipes*.

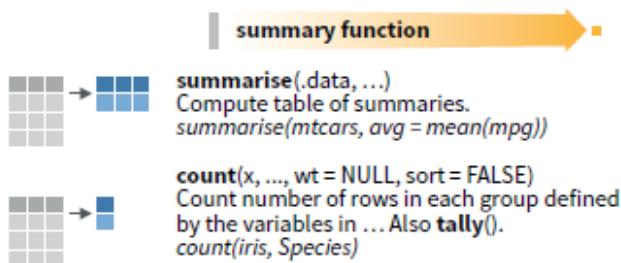
Nota: Una possible solució per aquest cas senzill de l'apartat *b* és:

```
> GrupXgenere <- starwars %>% count(gender)
```

Però se us demana fer el mateix fent ús de la mateixa comanda que heu fet servir a l'apartat anterior annexada amb *pipes* amb una altra comanda

Summarise Cases

These apply **summary functions** to columns to create a new table of summary statistics. Summary functions take vectors as input and return one value (see back).



Primer agrupem i després comptem:

```
> GrupXgenere <- starwars %>%group_by(gender) %>%count()
0 equivalentment amb summarize(n()):
> GrupXgenere <- starwars %>%group_by(gender) %>%summarize(n())
```

Però així veieu la idea de com utilitzar el `group_by()` també.

```
> GrupXgenere

> GrupXgenere <-starwars%>%group_by(gender)%>%count()
> GrupXgenere
# A tibble: 3 x 2
# Groups:   gender [3]
  gender     n
  <chr>    <int>
1 feminine    17
2 masculine   66
3 <NA>        4
> |
```

2.- Creeu una nova columna que inclogui l'alçada normalitzada per la mitjana global dels personatges. Mostreu dita columna junt amb el nom del personatge, la seva alçada i la espècie a la que pertany. És a dir:

```
# A tibble: 87 x 4
  name           height species height_norm
  <chr>          <int> <chr>      <dbl>
1 Luke Skywalker    172 Human       0.986
2 C-3PO              167 Droid       0.958
3 R2-D2              96 Droid       0.551
4 Darth Vader        202 Human       1.16 
5 Leia Organa         150 Human       0.860
6 Owen Lars            178 Human       1.02 
7 Beru Whitesun lars   165 Human       0.946
8 R5-D4                97 Droid       0.556
9 Biggs Darklighter    183 Human       1.05 
10 Obi-Wan Kenobi       182 Human       1.04
# ... with 77 more rows
> |
```

Nota: Abans de fer aquest exercici veieu què passa si fem la mitjana d'una variable que conté valors NA. Per això proveu:

```
x <- c(1,2,NA,3)
mean(x) # què us retorna?
mean(x, na.rm=TRUE) # I ara?
```

Si comencem fent el que ens diu la nota: Com x conté un valor NA, si no especificuem que no el tingui compte al fer la mitjana (o el que és el mateix, que només tingui en compte na.rm=TRUE), la mitjana ens retornarà NA com resultat.

Això ja ens està indicant que alguna alçada pot tenir un valor NA i haurem de tenir-ho en compte a l'hora de fer la mitjana.

Per a fer l'exercici: Hem vist que per crear noves columnes es podia fer servir la funció `mutate()`.

MAKE NEW VARIABLES

These apply **vectorized functions** to columns. Vectorized funs take vectors as input and return vectors of the same length as output (see back).

vectorized function

 **mutate(.data, ...)**
Compute new column(s).
`mutate(mtcars, gpm = 1/mpg)`

L'exercici ens demana a més, mostrar aquesta nova columna junt a les columnes `name`, `height` i `species` (les seleccionem amb `select`, doncs). Per tant, hem de fer:

```
> starwars %>% select(name, height, species) %>%
  mutate(height_norm = height / mean(height, na.rm = TRUE))

> starwars %>% select(name, height, species) %>%
  mutate(height_norm = height / mean(height, na.rm = TRUE)) # simplificant-ho

> starwars %>% select(name, height, species) %>% mutate(height_norm = height / mean(height, na.rm = TRUE))
#>
#> # A tibble: 87 x 4
#>   name           height species `height/mean(height, na.rm = TRUE)`
#>   <chr>          <int> <chr>   <dbl>
#> 1 Luke Skywalker    172 Human     0.986
#> 2 C-3PO              167 Droid     0.958
#> 3 R2-D2              96 Droid     0.551
#> 4 Darth Vader       202 Human     1.16
#> 5 Leia Organa        150 Human     0.860
#> 6 Owen Lars          178 Human     1.02
#> 7 Beru Whitesun Lars 165 Human     0.946
#> 8 R5-D4              97 Droid     0.556
#> 9 Biggs Darklighter  183 Human     1.05
#> 10 Obi-Wan Kenobi    182 Human     1.04
#> # ... with 77 more rows
> |
```

3.- Agrupeu els personatges de gènere masculí segons la seva espècie especificant quants n'hi ha de cada espècie. Feu el mateix però enlloc d'especificar 'quants', especifiqueu la mitjana de l'alçada de cada espècie

El gènere hem vist que formava part de les observacions o files del dataset i seleccionem les observacions/files amb la funció `filter()`

Agrupem amb `group_by()` i les especificacions les fem amb `summarize()`

Per tant:

> #Quants

```
> starwars %>%filter(gender=='masculine') %>%group_by(species)
%>%summarize(n())
```

```
> starwars %>% filter(gender == "masculine") %>% group_by(species) %>%summarize(n())
`summarise()` ungrouping output (override with `groups` argument)
# A tibble: 33 x 2
  species   `n()`
  <chr>     <int>
1 Aleena      1
2 Besalisk    1
3 Cerean      1
4 Chagrian    1
5 Droid       5
6 Dug         1
7 Ewok         1
8 Geonosian   1
9 Gungan       3
10 Human      26
# ... with 23 more rows
```

> #Mitjana de l'alçada de cada espècie

```
> starwars %>% filter(gender == "masculine") %>% group_by(species)
%>%summarize(mean(height, na.rm = TRUE))
```

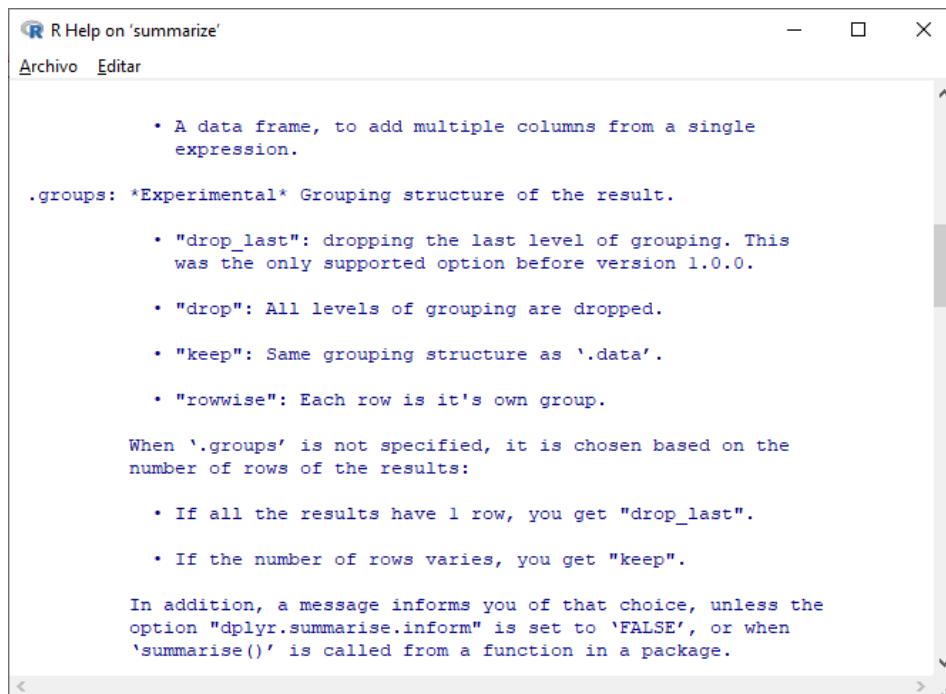
```
> starwars %>% filter(gender == "masculine") %>% group_by(species) %>%summarize(mean(height, na.rm = TRUE))
`summarise()` ungrouping output (override with `groups` argument)
# A tibble: 33 x 2
  species   `mean(height, na.rm = TRUE)`
  <chr>     <dbl>
1 Aleena      79
2 Besalisk    198
3 Cerean      198
4 Chagrian    196
5 Droid       140
6 Dug         112
7 Ewok         88
8 Geonosian   183
9 Gungan      209.
10 Human      182.
# ... with 23 more rows
> |
```

! En cas de sortir-vos el missatge ‘`summarise()` ungrouping...’ és “amigable”. De fet en algunes versions de R, ja no us sortirà:

```
> starwars %>% filter(gender == "masculine") %>% group_by(species) %>%summarize(mean(height, na.rm = TRUE))
# A tibble: 33 x 2
  species   `mean(height, na.rm = TRUE)`
  * <chr>     <dbl>
1 Aleena      79
2 Besalisk    198
3 Cerean      198
4 Chagrian    196
5 Droid       140
6 Dug         112
7 Ewok         88
8 Geonosian   183
9 Gungan      209.
10 Human      182.
# ... with 23 more rows
> |
```

En cas de sortir dit missatge, simplement ens diu que s'està desgropant , és a dir, quan hi ha un sol group_by, elimina aquesta agrupació després del summarize

No és greu, però si fem > ?summarize



R Help on 'summarize'

Archivo **Editar**

- A data frame, to add multiple columns from a single expression.

.groups: *Experimental* Grouping structure of the result.

- "drop_last": dropping the last level of grouping. This was the only supported option before version 1.0.0.
- "drop": All levels of grouping are dropped.
- "keep": Same grouping structure as '.data'.
- "rowwise": Each row is it's own group.

When '.groups' is not specified, it is chosen based on the number of rows of the results:

- If all the results have 1 row, you get "drop_last".
- If the number of rows varies, you get "keep".

In addition, a message informs you of that choice, unless the option "dplyr.summarise.inform" is set to 'FALSE', or when 'summarise()' is called from a function in a package.

Ens diu que si canviem els .groups a la funció summarize(), no rebem el missatge perquè s'eliminen els atributs del grup.

```
> starwars %>% filter(gender == 'masculine') %>% group_by(species)
%>%summarize(mean(height, na.rm = TRUE), .groups='drop')

> starwars %>% filter(gender == "masculine") %>% group_by(species) %>%summarize(mean(height, na.rm = TRUE), .groups='drop')
# A tibble: 33 x 2
  species `mean(height, na.rm = TRUE)`
  <chr>          <dbl>
1 Aleena        79
2 Besalisk      198
3 Cerean         198
4 Chagrian      196
5 Droid          140
6 Dug            112
7 Ewok           88
8 Geonosian     183
9 Gungan         209.
10 Human         182.
# ... with 23 more rows
> |
```

Finalment, remarcar-vos que per les altres funcions d'agregat (min(), max(), sum()) caldria afegir el mateix paràmetre:

```
starwars%>%group_by(gender)%>%summarize(fagr=min(height, na.rm=TRUE))
starwars%>%group_by(gender)%>%summarize(fagr=max(height, na.rm=TRUE))
starwars%>%group_by(gender)%>%summarize(fagr=sum(height, na.rm=TRUE))
```

Ara que ja ens hem familiaritzat amb algunes de les funcions de la llibreria dplyr, veiem com ens ajuda *data massaging* a la visualització de dades.

3. PART 2. Visualització de dades després de *data massaging*

En aquesta segona part del seminari anem a veure dos exemples de l'aplicació del *data massaging* a la visualització de dades.

EXERCICIS:

1.- Dibuixeu un *scatter plot* on l'eix x correspongui a l'alçada i l'eix y correspongui a la massa dels personatges. Intenteu que no us surti cap *warning*. Què us permet veure aquesta gràfica? La massa és més gran o més petita a mesura que l'alçada creix? Hi ha algun personatge que tingui una massa/alçada molt gran/petita respecte els altres i us dificulti la resposta? Traieu-lo si és el cas, torneu a dibuixar el gràfic sense aquest personatge i refeu el raonament.

Finalment, afegiu al mateix gràfic la informació referent al gènere del personatge (feminine, masculine, NA). En traieu informació nova? Quina per exemple?

Proveu de fer una regressió lineal (com vam veure a la classe de teoria 5) i ajusteu el vostre interval de confiança al 90%.

Si fem directament:

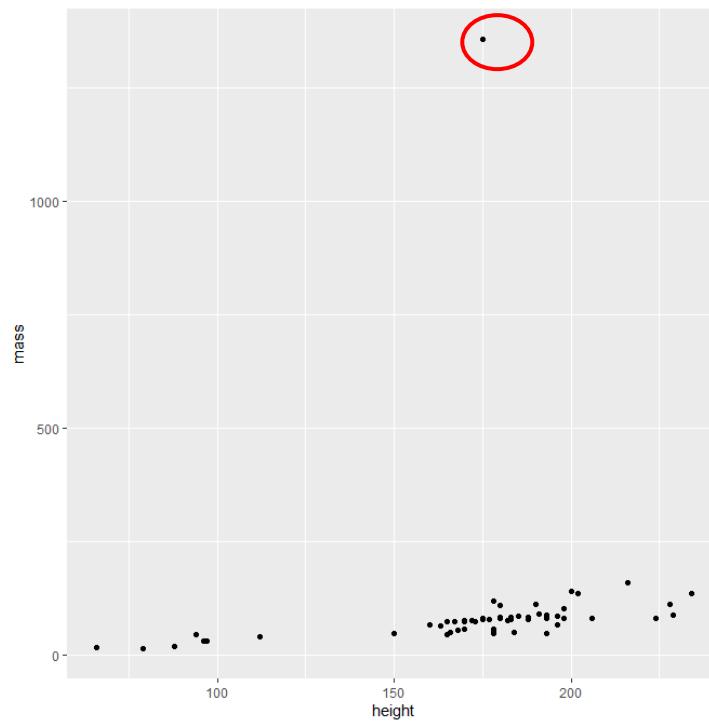
```
> ggplot(starwars)+aes(height,mass)+geom_point()  
> #or:  
> ggplot(starwars, aes(height,mass))+geom_point()  
Warning message:  
Removed 28 rows containing missing values (geom_point).
```

Per treure el *warning* hem de treure doncs els personatges que tenen un valor NA en la columna height o mass. Al treure personatges estem traient files, per tant usem *filter()*:

```
> starwars2 <-starwars %>% filter(height!="NA")%>%filter(mass!="NA")  
> ggplot(starwars2, aes(height,mass))+geom_point()
```

O:

```
> starwars2 <-starwars %>% filter(height!="NA", mass!="NA")  
> ggplot(starwars2, aes(height,mass))+geom_point()
```



Clarament en el cercle vermell tenim un *outlier* (un personatge amb una massa molt gran per la seva alçada) que ens dificulta veure la tendència de la massa respecte l'alçada. De fet podríem inclús respondre erròniament que tot i que a mesura que augmenta l'alçada, la massa augmenta, aquest creixement no és significatiu.

Si som fans de starwars sabem el nom del personatge que té una massa molt superior als altres (tot i que la seva alçada no és la més alta), per tant serà fàcil treure'l. Una pista:



```
> starwars3<-filter(starwars2, !name=="Jabba Desilijic Tiure")
> ggplot(starwars3, aes(height,mass))+geom_point()
```

O bé:

```
> starwars3<-filter(starwars2, name!="Jabba Desilijic Tiure")
> ggplot(starwars3, aes(height,mass))+geom_point()
```

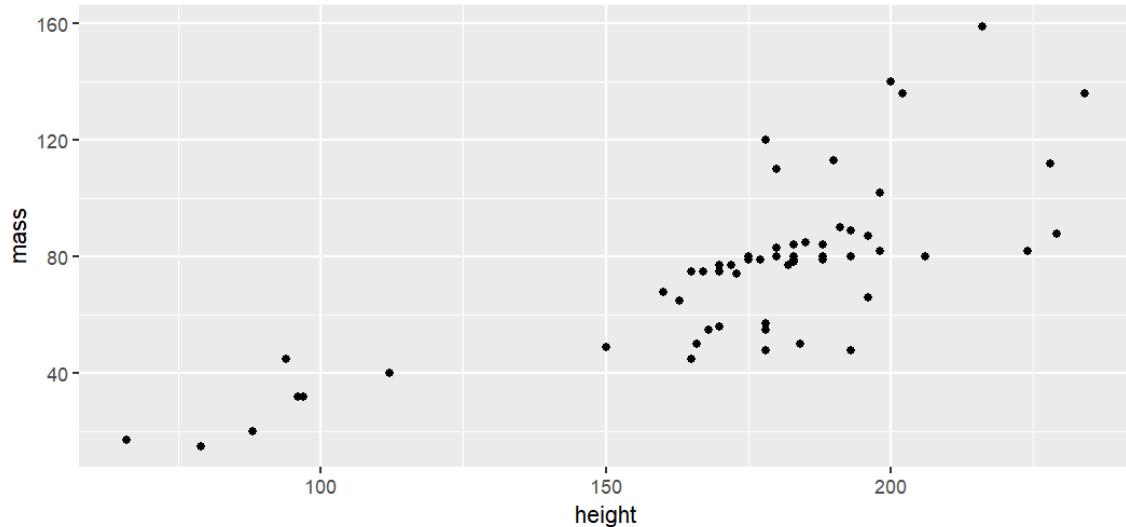
Si no som fans (o ens trobem davant un altre dataset desconegut per nosaltres) ens podem quedar amb els personatges que no estan per sobre un cert valor d'umbralització o *threshold*:

```
> starwars3 <-filter(starwars2, !mass>1000)
```

o dit d'una altra manera, que estan per sota un cert *threshold*:

```
> starwars3 <-filter(starwars2, mass<1000)
> ggplot(starwars3, aes(height,mass))+geom_point()
```

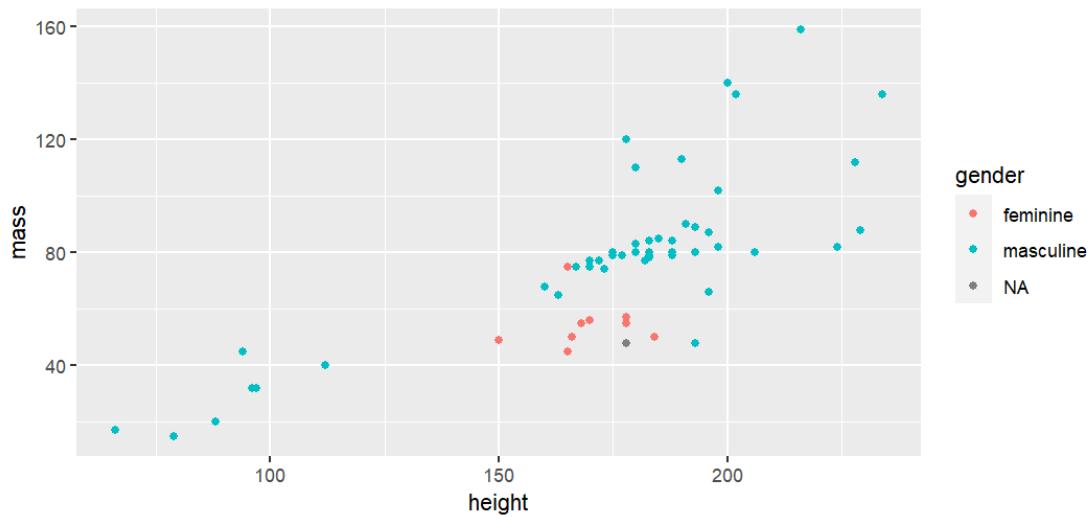
En aquest cas, el gràfic que ens surt ara és:



La nova gràfica, mostra molt millor que la massa corporal creix al créixer l'alçada. A l'esquerra tindríem majorment els robots o en Yoda, són petits i no pesen gaire. Entre 170cm-190cm , es trobarien majorment (tret d'algunes excepcions) els humans i la seva massa corporal està majorment entre 70-90 kg. A partir de 200cm no tenim personatges per sota els 80kg.

Si ara volem afegir el gènere, primer veiem que gènere és una variable que pren tres valors: feminine, masculine i NA. Per tant necessitem posar-li un color a cada gènere

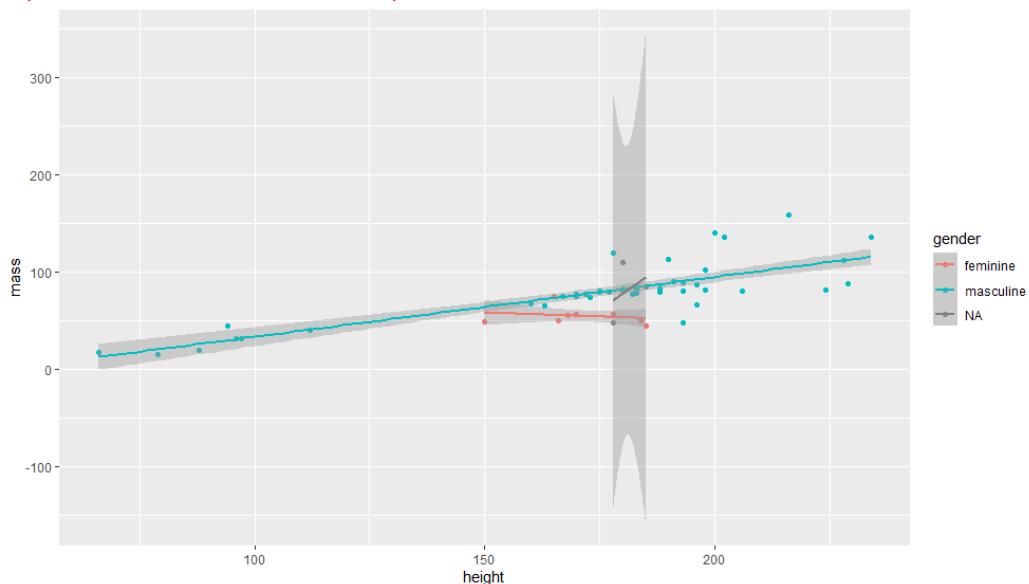
```
> ggplot(starwars3, aes(height,mass,color=gender))+geom_point()
```



Se n'ha parlat molt a la premsa, que la majoria dels personatges de starwars eren masculins o tenien un rol masculí (gènere del dataset). De fet aquí veiem clarament que hi ha pocs personatges de gènere femení. Podem intuir també que la seva relació alçada-massa no és alta.

Podem utilitzar el canal `geom_smooth()`. Per defecte utilitza el mètode “loess” i una regió de confiança del 95%. Provarem amb un interval del 90% i un mètode lineal com diu l'exercici, tot seguint les indicacions de la classe de teoria 5.

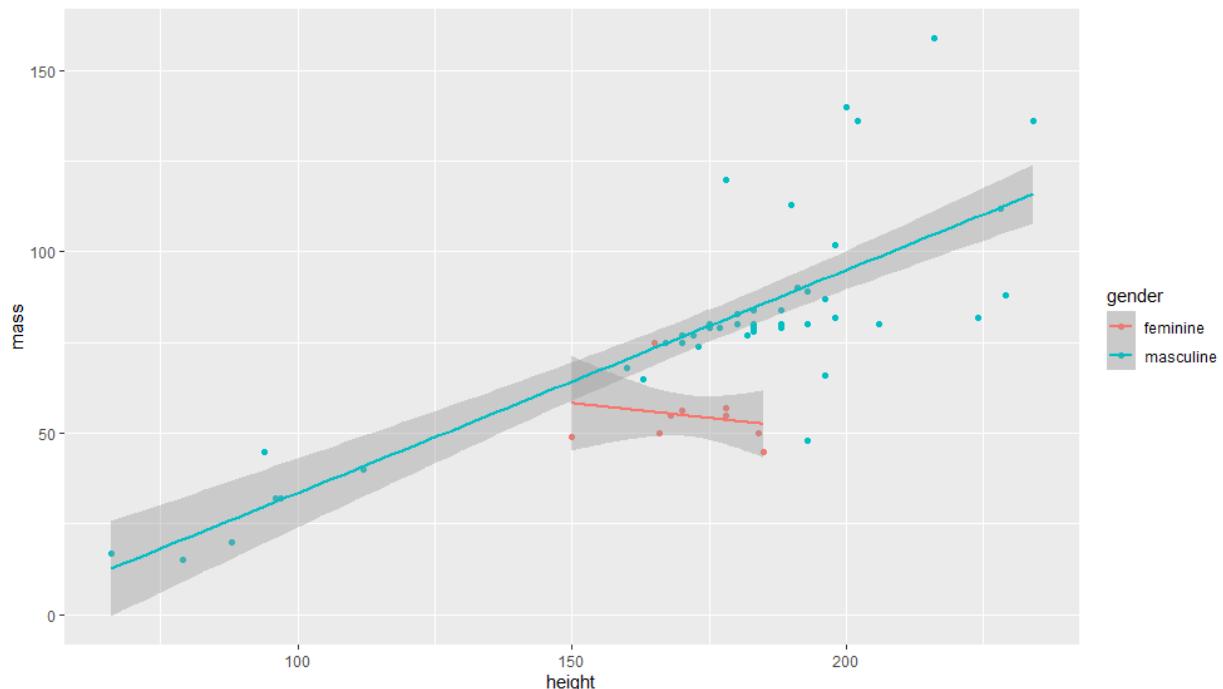
```
>ggplot(starwars3, aes(height,mass,color=gender))+geom_point()+geom_smooth(method="lm", level=0.9)
```



Veiem que quan fem una regressió lineal, pel gènere masculí a partir d'1.75 m comencen a no estar dins del nostre interval de confiança del 90%. En canvi el gènere femení sí que exceptuant un personatge, la resta estan més dins de l'interval de confiança del 90%

Podem treure els NA fent prèviament:

```
>starwars3 <-filter(starwars3, gender != 'NA')
```



2.- Intenteu veure la distribució de l'alçada per dues o tres espècies que trieu lliurement.

- a) Si no són les que heu triat, feu-ho pel cas humans i robots i traieu-ne conclusions.
- b) Compteu quants personatges hi ha de cada espècie. Us dona alguna informació de com interpretar l'apartat a?
- c) Ara mostreu amb un *scatter plot* on es mostri el nom i l'alçada dels robots. Quines conclusions en traieu? Representeu amb un gràfic de barres, fent servir el que vam veure en l'últim seminari, que us permeti comparar les proporcions d'alçades pels diferents robots.
- d) Proveu de fer un diagrama de violí com els que heu vist a teoria per mostrar el mateix que en l'apartat a. Utilitzeu `geom_violin()`.

Les espècies són variables discretes mentre que l'alçada és una variable contínua, per tant de les gràfiques que em vist fins ara per mostrar distribucions, sembla que els diagrames de caixa (boxplots) són adients per mostrar-nos les distribucions de l'alçada segons les espècies.

L'enunciat ens està dient que triem dues o tres espècies, per tant ens està demanant que filtrem la informació de dues o tres espècies.

a) Si fem per exemple el cas que ens diu dels humans i robots:

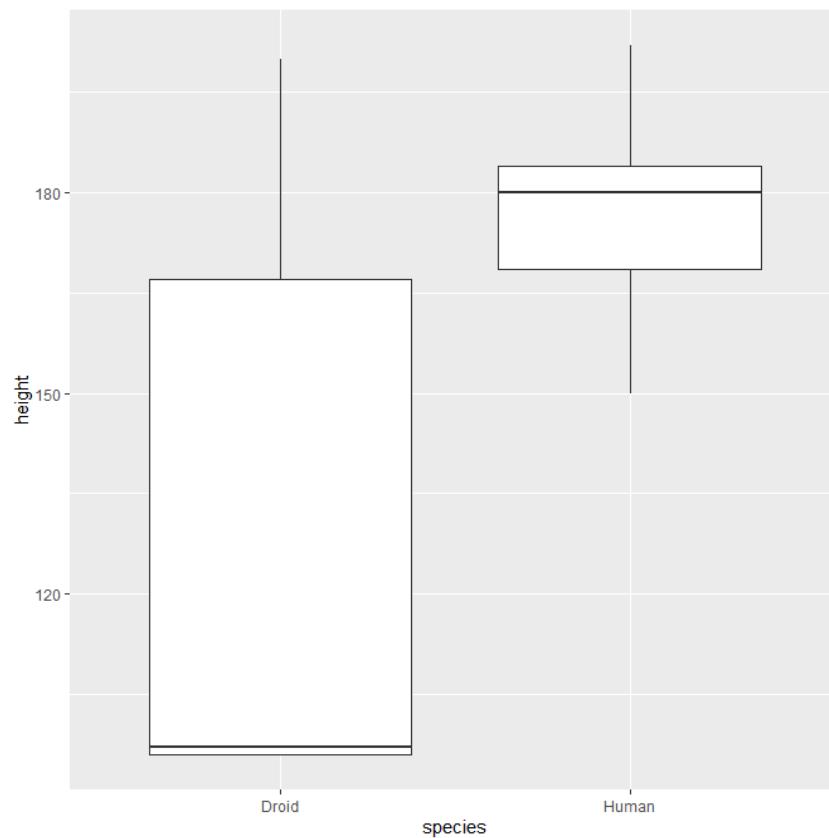
```
> starwars2<-starwars %>%filter(species == "Human" | species == "Droid")
> ggplot(starwars2, aes(species, height))+geom_boxplot()
```

Warning message:

Removed 5 rows containing non-finite values (stat_boxplot).

Per treure el *warning* podem fer:

```
> starwars2<-starwars %>%filter(species=="Human" | species=="Droid")
%>%filter(height!="NA")
> ggplot(starwars2, aes(species, height))+geom_boxplot()
```



La distribució de l'alçada dels humans tot i que és més variada per sota la seva respectiva mediana (que la dels robots), és tota ella menys variada que en el cas dels robots. En el cas dels robots el boxplot ens diu que la distribució de les alçades dels robots està majoritàriament per sobre les medianes, de fet la màxima alçada està molt per sobre la mediana. De fet en aquest segon cas, el mínim està molt a prop de la mediana. El boxplot dels robots ens està indicant ja alguna cosa (veure apartat c).

b) Com hem fet abans en la Part 1 del seminari, utilitzarem les funcions `group_by()` i `count()` per comptar el número de personatges de cadascuna de les dues espècies:

```
> GrupXspecie <- starwars2 %>% group_by(species) %>% count()
> GrupXspecie
```

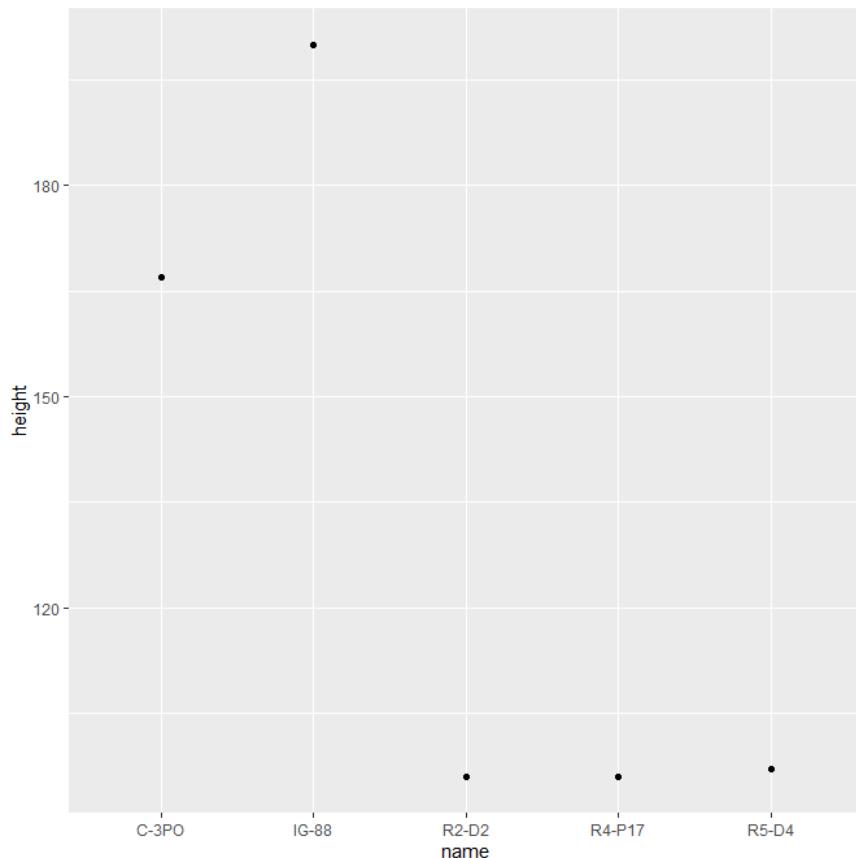
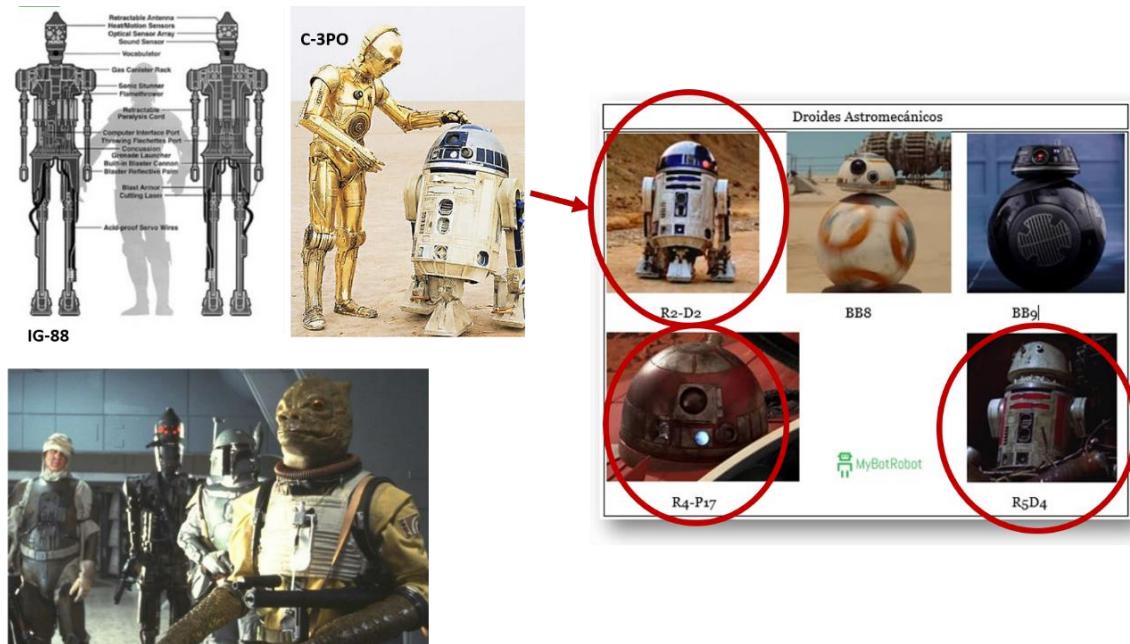
```
# A tibble: 2 x 2
# Groups:   species [2]
  species     n
  <chr>    <int>
1 Droid      5
2 Human     30
```

Tenim només 5 robots, per tant el boxplot amb tant poques dades ens pot enganyar (les alçades de 5 personatges difícilment seguiran una distribució normal o una distribució coneguda).

c) Al fer el *scatter plot*:

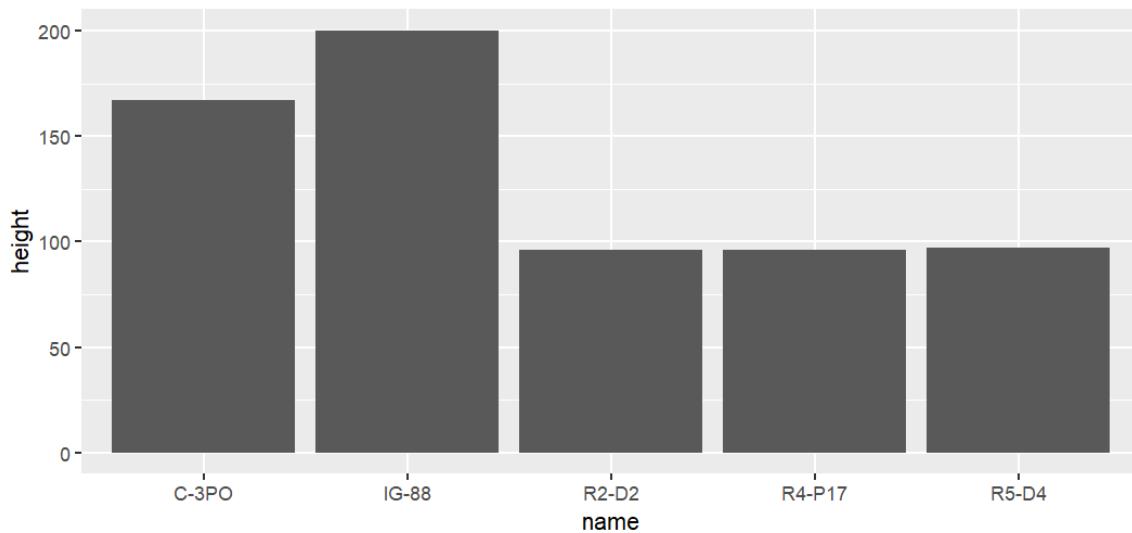
```
> starwars2 <- starwars %>% filter(species=="Droid")
%>% filter(height!="NA")
> ggplot(starwars2, aes(name, height)) + geom_point()
```

Dels 5 robots que teníem en el dataset, tres d'ells tenen una alçada semblant (R2-D2, R4-P17, R5D4), mentre que els altres dos tenen una alçada més elevada (C-3PO al voltant de la mediana de l'alçada humana i el IG-88 molt més alt). Al només comptar amb aquests 5 personatges el boxplot ens pot enganyar (de fet en l'apartat (a) ja hem vist alguna cosa)



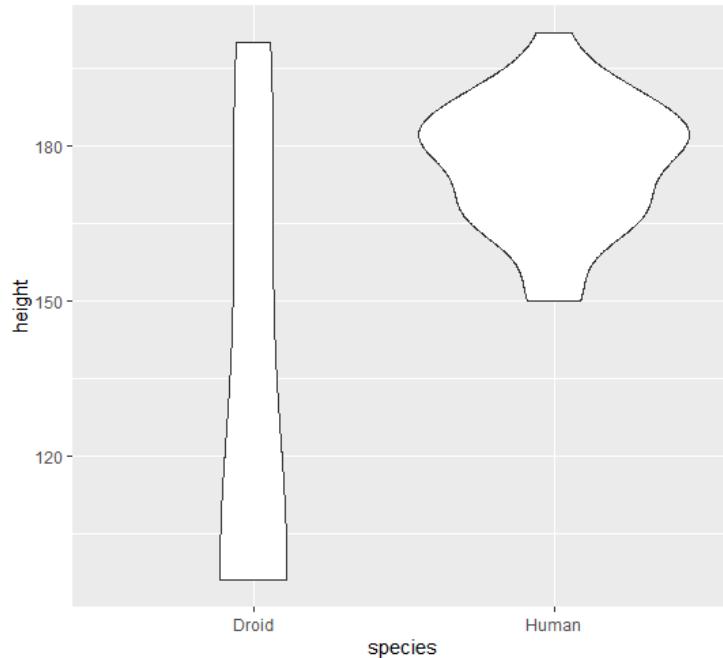
El *scatter plot* aquí ens mostra realment la diferencia d'alçades que veiem en les figures dels personatges.

Per fer el gràfic de barres vam veure el `geom_bar()` i el `geom_col()` al seminari 1. Com volem veure els valors de les dades de la variable “height” com alçades de les barres, fem ús de `geom_col()`.



d) Si fem un diagrama de violí:

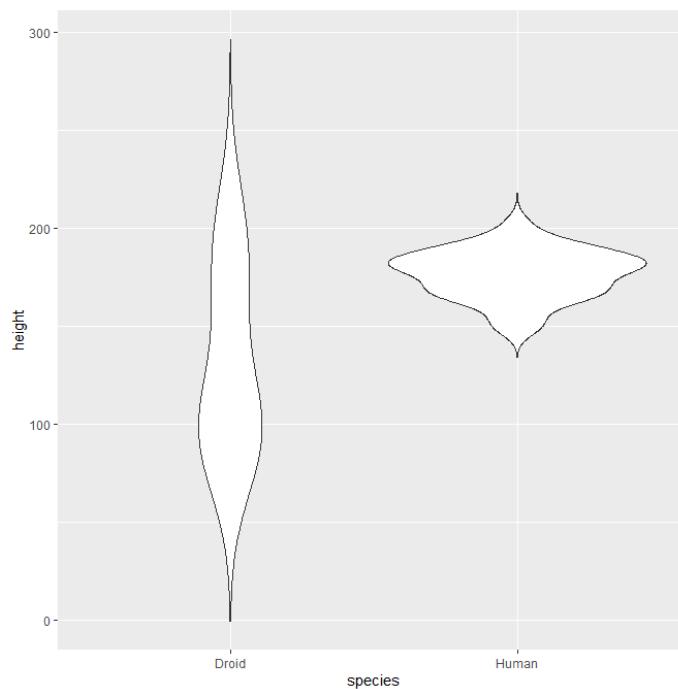
```
> starwars2<-starwars  %>%filter(species=="Human" | species=="Droid")
%>%filter(height!="NA")
> ggplot(starwars2, aes(species, height))+geom_violin()
```



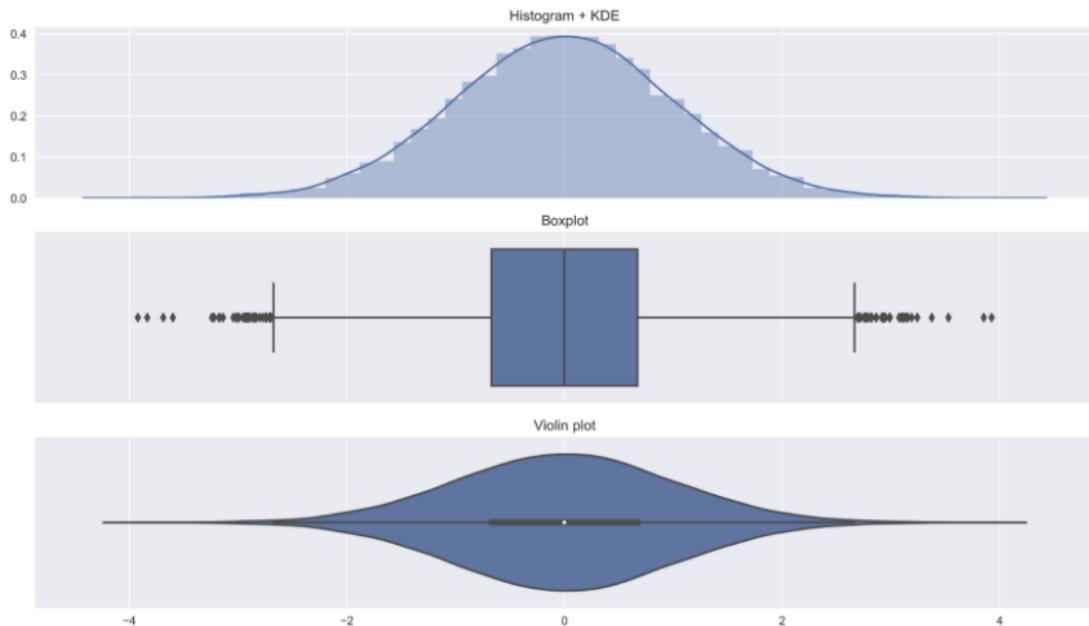
Si no volem que ens talli les cues dels violí segons les dades, fem `?geom_violin` on veiem entre d'altres opcions:

`trim: If 'TRUE' (default), trim the tails of the violins to the range of the data. If 'FALSE', don't trim the tails.`

```
> ggplot(starwars2, aes(species, height))+geom_violin(trim='FALSE')
```



El diagrama de violí ens enganya menys que el *boxplot* (tot i que el dataset contínua sent reduït per aquest tipus de gràfic). Com heu vist a classe, el violí us mostra més la distribució de les dades. Per posar un exemple, si la distribució fos normal, comparant *boxplots* i violins tindríem que el contorn del violí tindria forma de distribució normal :



4. PART 3. *Data massaging (tidy)* & gràfiques avançades

En aquesta tercera part del seminari utilitzarem el dataframe iris. Utilitzarem algunes funcions de la llibreria tidy per reformular el nostre dataframe i acabarem fent un multi-panel de figures (small multiples).

Iris proporciona les mesures (en cm) de les variables longitud i amplada dels sèpals i dels pètals respectivament per 50 flors de cadascuna de les 3 espècies d'Iris (150 en total). Les espècies d'iris són: la Versicolor, la Virginica i la Setosa.



Iris és un dataframe amb 150 casos (files) i 5 variables (columnes) anomenades: **Sepal.Length**, **Sepal.Width**, **Petal.Length**, **Petal.Width** i **Species**. Els valors de les variables referents a les respectives longituds i amplades (mètriques) estan en centímetres.

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5.0	3.4	1.5	0.2	setosa
9	4.4	2.9	1.4	0.2	setosa
10	4.9	3.1	1.5	0.1	setosa
11	5.1	3.7	1.5	0.2	setosa

Showing 1 to 11 of 150 entries, 5 total columns

Concretament ens familiaritzarem amb algunes funcions de la llibreria *tidyverse* que vam veure dimarts a classe, com:

- Gather: to convert wide format dataframe to long format.
- Spread: to convert long format dataframe to wide format.
- Separate: to separate two variables being placed into the same column.
- Unite: to combine multiple columns into a single column.
- Drop_na, fill & replace_na: to handle missing values.
- pipes %>%: to combine operations.

EXERCICIS:

1.- Partint del dataframe iris

a) Feu un nou dataframe de format llarg que tingui una columna ‘metric’ (amb les 4 mètriques) una columna ‘value’ (amb el valor en ‘cm’ de cada respectiva mètrica) i ordenat per *Species*. El dataframe resultant ha de tenir aquesta forma:

	Species	metric	value
1	setosa	Sepal.Length	5.1
2	setosa	Sepal.Length	4.9
3	setosa	Sepal.Length	4.7
4	setosa	Sepal.Length	4.6
5	setosa	Sepal.Length	5.0
6	setosa	Sepal.Length	5.4
7	setosa	Sepal.Length	4.6
8	setosa	Sepal.Length	5.0
9	setosa	Sepal.Length	4.4

Showing 1 to 9 of 600 entries, 3 total columns

La funció *gather* ens convertia un dataframe en format llarg. El dataframe original estava format per 150 observacions (files) i 5 variables (columnes). De les 5 columnes quatre d'elles eren mètriques, exceptuant la variable *Species*. Per tant, no farem cap canvi en la columna *Species*. En canvi, el valor de les altres quatre columnes el posarem en la nova variable *value* (nova columna), associant-lo amb el nom de la mètrica que correspon (columna *metric*):

```
gather(data, key, value, ..., na.rm = FALSE,
       convert = FALSE, factor_key = FALSE)
```

gather() moves column names into a **key** column, gathering the column values into a single **value** column.

table4a

country	1999	2000
A	0.7K	2K
B	37K	80K
C	212K	213K

→

country	year	cases
A	1999	0.7K
B	1999	37K
C	1999	212K
A	2000	2K
B	2000	80K
C	2000	213K

key value

```
gather(table4a, `1999`, `2000`,
      key = "year", value = "cases")
```

Fixeu-vos que en l'exemple de les transparències prèvies, l'ordre (de les columnes a crear o existents) no ens influïa:

```
> df
# A tibble: 3 x 3
  country `1999` `2000`
  <chr>    <chr>   <chr>
1 A        0.7K    2K
2 B        37K     80K
3 C        212K    213K
> gather(df, '1999','2000', key="year",value="cases")
# A tibble: 6 x 3
  country year   cases
  <chr>    <chr>   <chr>
1 A        1999   0.7K
2 B        1999   37K
3 C        1999   212K
4 A        2000   2K
5 B        2000   80K
6 C        2000   213K
> gather(df, key="year",value="cases", '1999','2000')
# A tibble: 6 x 3
  country year   cases
  <chr>    <chr>   <chr>
1 A        1999   0.7K
2 B        1999   37K
3 C        1999   212K
4 A        2000   2K
5 B        2000   80K
6 C        2000   213K
> |
```

Fem:

```
#iris_long<-gather(iris, metric, value, Sepal.Length, Sepal.Width,
Petal.Length, Petal.Width)

>iris_long<-gather(iris, metric, value, -Species)      #totes les
variables(columnnes) menys species

>view(iris_long) #useu view per visualitzar el nou dataframe i comprovar
que heu fet el que volieu
```

Podríem canviar l'ordre de les variables existents/noves? i.e., posant mètric i value al final del gather?

```
> iris_long<-gather(iris, Sepal.Length, Sepal.Width, Petal.Length,
Petal.Width, key='metric',value='value')
```

!! Funcionaria canviar l'ordre si especificuem qui és la *key* i qui és el *value* del nou dataframe amb format long

Però no ens funcionaria si canviem l'ordre sense especificar el *key* i *value*:

```
> iris_long<-gather(iris, Sepal.Length,Sepal.width, Petal.Length, Petal.Width, metric,value)
Error: Can't subset columns that don't exist.
x Column `metric` doesn't exist.
Run `rlang::last_error()` to see where the error occurred.
>
```

NOTA: Per no recordar si van primer les variables existents o les noves, simplement guardem l'ordre que preferim, especificant clarament qui seran la *key* i el *value* en el nou dataframe.

b) Un cop el tingueu feu us del canal que hem vist a classe `facet` per fer una figura *multi-panel* (és a dir, una figura amb 4 finestres/panels, on cada finestra correspongui a una mètrica). Per cada mètrica, en cada panel, es compararà el valor (de la mètrica corresponent) per cada espècie.

Referent al gràfic:

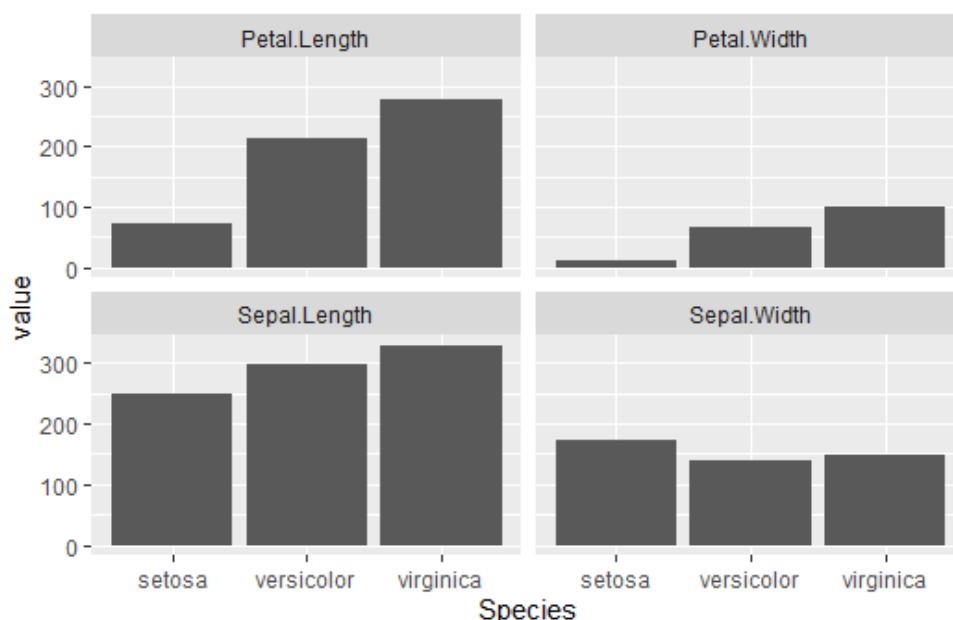
- Volem un subgrafic per cada mètrica per tant necessitarem fer ús del canal: `facet_wrap(~ metric)`
- A més, volem un gràfic que en cada una de les 4 finestres referent a la mètrica ens compari els valors d'aquesta per diferent espècies. I vam veure que `geom_bar` i `geom_col` servien per comparar valors.
- Finalment en cada finestra volem comparar valor d'espècies del nou dataframe `iris_long`, per tant sabem que hem de posar `ggplot(iris_long)+aes(Species, value)`

Si ho unim tot:

```
>ggplot(iris_long)+aes(Species, value)+geom_col()+facet_wrap(~ metric)
```

O alternativament:

```
>ggplot(iris_long)+aes(Species,
value)+geom_bar(stat='identity')+facet_wrap(~ metric)
```



2.- Combineu la columna Petal.Width amb Petal.Length en una columna Petal, de manera que la columna Petal contingui el valor de l'amplada seguit de la longitud del pètal separats per un guió baix, exemple:

	Sepal.Length	Sepal.Width	Petal	Species
1	5.1	3.5	0.2_1.4	setosa
2	4.9	3.0	0.2_1.4	setosa
3	4.7	3.2	0.2_1.3	setosa
4	4.6	3.1	0.2_1.5	setosa
5	5.0	3.6	0.2_1.4	setosa

```
>Petal_unit<- iris %>% unite(Petal,Petal.Width, Petal.Length,sep="_")  
>view(Petal_unit) #per visualitzar
```

Judit Chamorro Servent

Bellaterra, Març 2025

SEMINARI 5. *Processament de dades i gràfiques avançades II*

1. OBJECTIUS

En aquest seminari, seguirem amb gràfiques exploratòries, i introduirem la visualització d'incertituds amb ggplot. A més, veure'm l'aplicabilitat de les tibbles. Finalment, veurem l'aplicabilitat amb R de tècniques per mostrar múltiples variables.

2. PART 1. *Advanced Systems I*

En aquesta primera part del seminari anem a veure algunes eines de les que vam veure en la teoria de processat de dades per mostrar la incertitud i error. Per això utilitzarem el dataframe *iris*.

Iris proporciona les mesures (en cm) de les variables longitud i amplada dels sèpals i dels pètals respectivament per 50 flors de cadascuna de les 3 espècies d'Iris (150 en total). Les espècies d'iris són: la Versicolor, la Virginica i la Setosa.



Iris és un dataframe amb 150 casos (files) i 5 variables (columnes) anomenades: **Sepal.Length**, **Sepal.Width**, **Petal.Length**, **Petal.Width** i **Species**. Els valors de les variables referents a les respectives longituds i amplades (mètriques) estan en centímetres.

EXERCICIS:

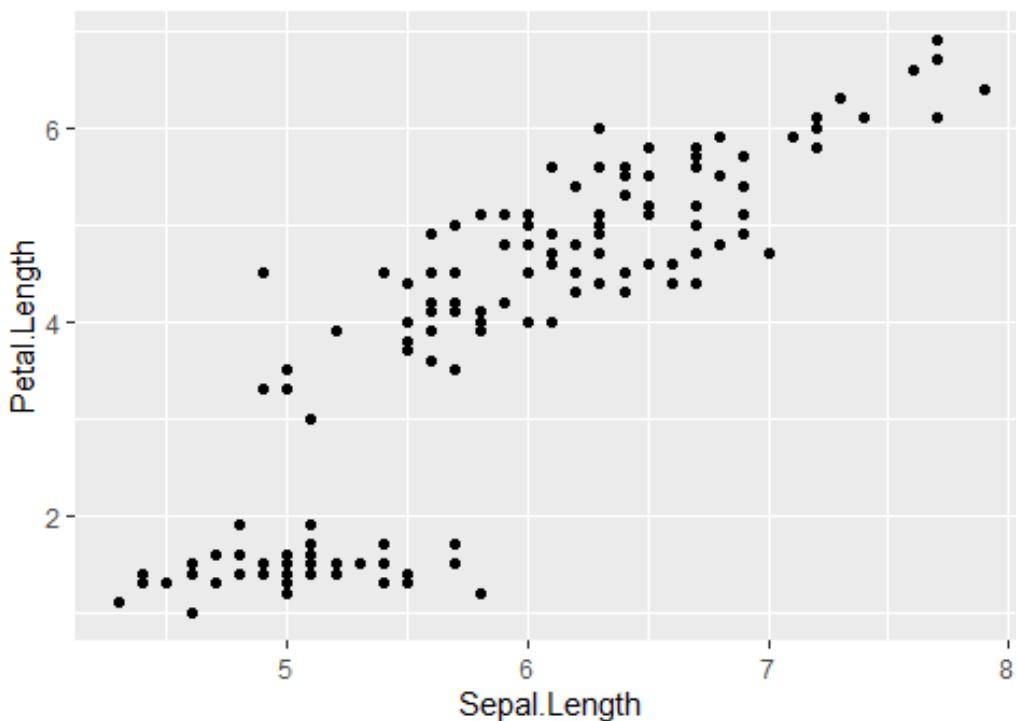
1.- Suposem que volem veure la correlació entre dues variables **Sepal.Length** i **Petal.Length**.

a) Comenceu fent una gràfica de punts/"scatterplot". Trobeu algun patró entre ambdues variables.

#No repetirem en cada exercici les diferents formes d'escriure, però sent el primer, dues possibles formes són:

```
>ggplot(iris, aes(Sepal.Length, Petal.Length)) + geom_point()
```

```
>ggplot(iris)+aes(Sepal.Length, Petal.Length) + geom_point()
```

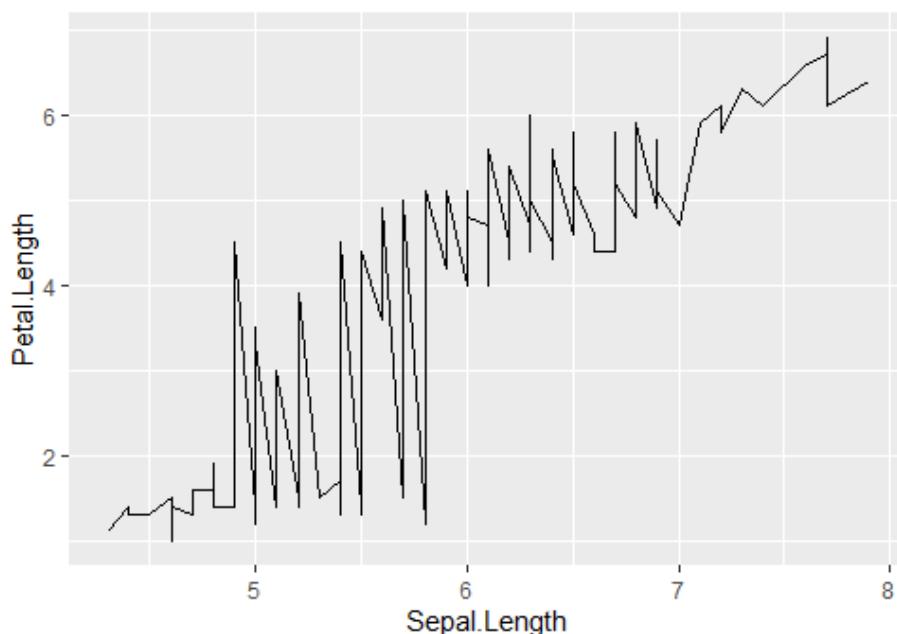


Veiem que per valors de Sepal.Length petits (grans) tenim valors petits (grans) de Petal.Length. Però a més hi a partir d'una Petal.Length més gran que 3cm, tenim que les Sepal.Length guarden una relació molt diferent a quan aquesta és menor que 3cm.

b) Proveu d'utilitzar línies (utilitzant geom_line()). Creieu que és una bona opció en aquest cas?

Només ens demanen canviar la nostra geometria de punts per una geometria de línia:

```
>ggplot(iris)+aes(Sepal.Length, Petal.Length) + geom_line()
```



En aquest cas utilitzar línies no és una bona opció, de fet ens introduiria error. Sembla que hi ha flors amb una longitud de pètal entre 2-4 cm, mentre que la figura de l'apartat (a) ens mostrava clarament que no.

c) Agregueu un canal que afegeixi al gràfic de punts de l'apartat (a), un ombrejat basat en càlculs estadístics de funcions de suavitzat (canal ‘stat/geom_smooth()’). Proveu de fer una regressió i ajusteu el vostre interval de confiança al 90% (NOTA: mireu quin dels tres mètodes s'ajusta millor a les vostres dades: *linear model* (“lm”), *generalized linear model* (“glm”) i *local regression fitting* “loess”.

- `lm` s'utilitza per ajustar models lineals, incloent-hi els multivariants.
- Si la relació fos lineal però distorsionada per la presència d' *outliers* en les dades utilitzaríem “`r1m`” (model lineal robust) per minimitzar la influència dels *outliers* en l'estimació de la relació.
- Si la relació fos no lineal però suau, podríem utilitzar tant “`loess`” com “`gam`”. El mètode “`loess`” es basa en l'allissament local lineal i pot gestionar *outliers*. En canvi “`gam`” permet diferents tipus d'allissament.
- El mètode `glm` seria útil en situacions en què la variable de resultat es tractaria com una variable binària.
- Les funcions `lm` i `r1m` també poden acomodar relacions no lineals de forma paramètrica (per exemple, quadràtica, cúbica...), tot i que hauríeu de fer servir una especificació de fórmula. Exemple: `geom_smooth(method="lm", formula = y ~ x + I(x^2))` per a una relació quadràtica estimada amb el mètode `lm`.

Recordeu que el canal `geom_smooth()`, per defecte utilitza el mètode “`loess`” amb fórmula '`y ~ x'` i una regió de confiança del 95%

`geom_smooth()`, per defecte:

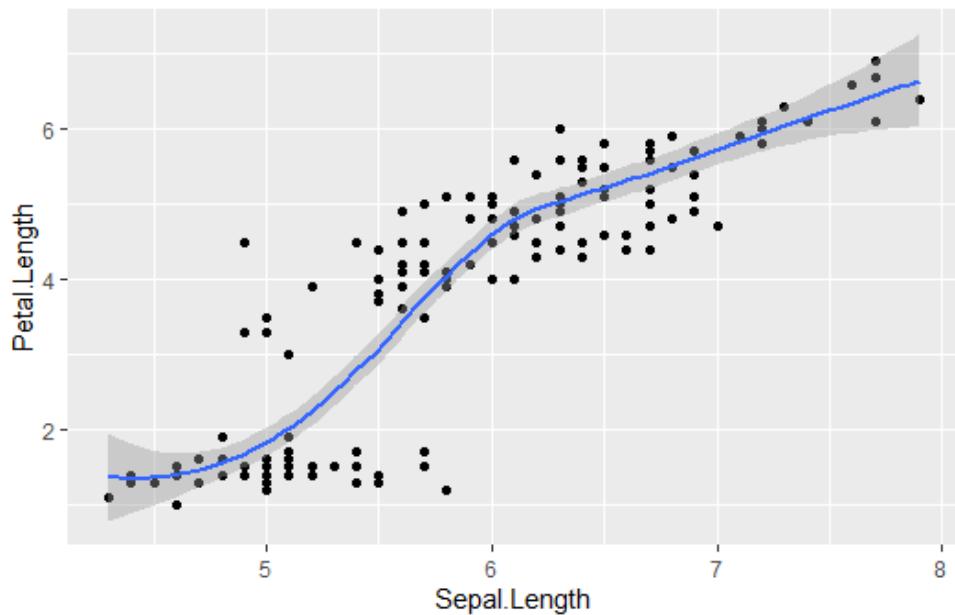
```
>ggplot(iris, aes(Sepal.Length, Petal.Length)) + geom_point() + geom_smooth()
```

Provarem els tres mètodes amb una regió de confiança del 90%, com ens demana el problema:

```
>ggplot(iris, aes(Sepal.Length, Petal.Length)) + geom_point() + geom_smooth(method="lm", level=0.90)
```

```
>ggplot(iris, aes(Sepal.Length, Petal.Length)) + geom_point() + geom_smooth(method="glm", level=0.90)
```

```
> ggplot(iris, aes(Sepal.Length, Petal.Length)) + geom_point() + geom_smooth(level=0.90) #al no especificar mètode per defecte agafa el “loess” i veient els gràfics resultants és el que millor ens ajusta el nostre dataset
```



En el cas de `glm` i `lm` no observem diferències. `glm` funciona millor que `lm` sol quan hi ha una variable resultat que funciona com a binària.

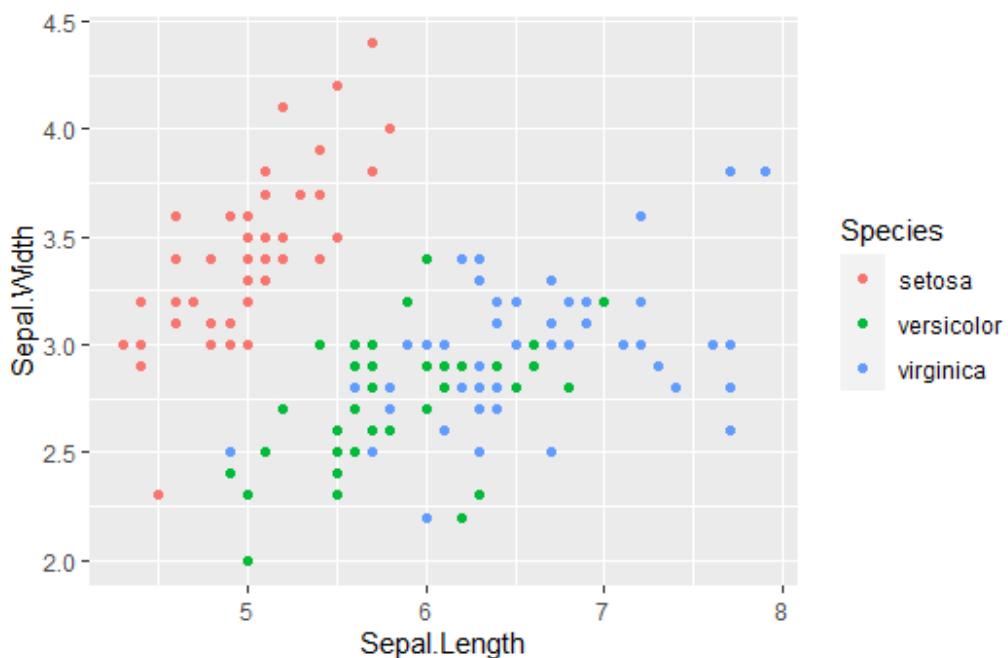
Nota: El `rml` no funciona per algunes versions de R.

2.- Seguint amb la correlació entre dues variables, en aquest cas Sepal.Length i Sepal.Width,

a) Pinteu els punts del scatter plot amb diferents colors segons les espècies.

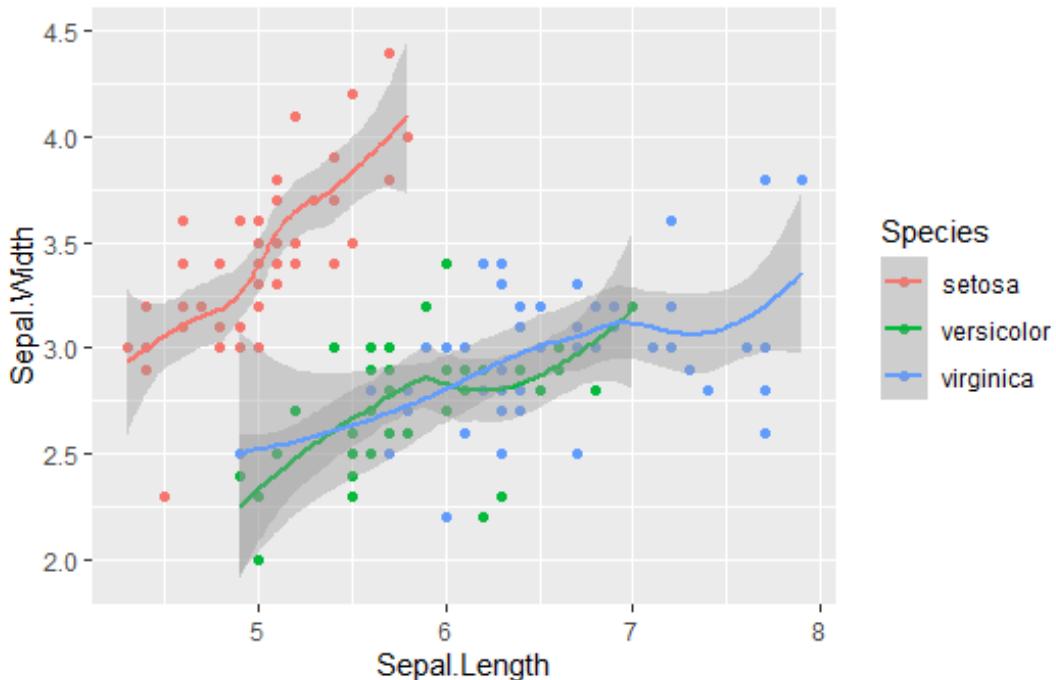
Dins d'aes fem un mapeig de color segons les 'species'

```
>ggplot(iris, aes(Sepal.Length, Sepal.Width, color=Species)) +geom_point()
```



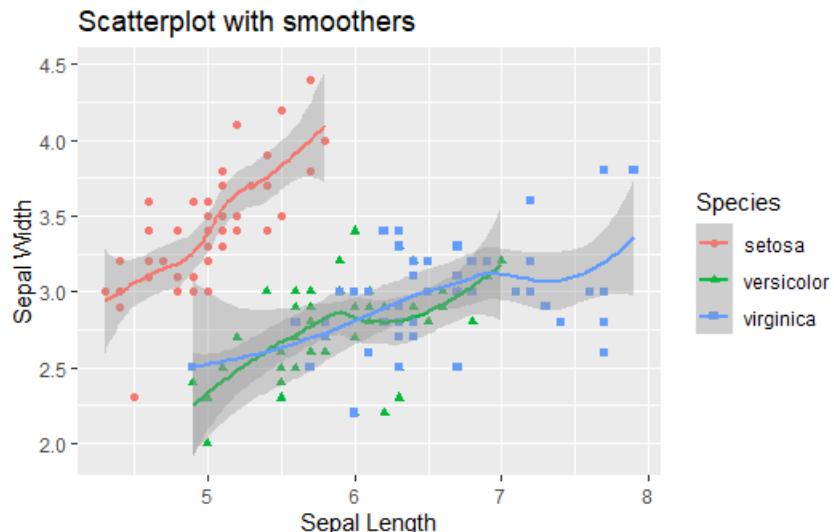
b) Com en l'exercici anterior, agregueu un canal que afegeixi al gràfic de punts de l'apartat (a), un ombrejat basat en càlculs estadístics de funcions de suavitzat (canal 'stat/geom_smooth()').

```
>ggplot(iris, aes(Sepal.Length, Sepal.Width, color=Species)) +geom_point()+geom_smooth()
```



c) Afegiu una forma (*shape*) al tipus d'espècie. Poseu un títol i un nom adient als eixos

```
> ggplot(iris, aes(Sepal.Length, Sepal.Width, color=Species)) +geom_point(aes(shape=Species))  
+geom_smooth()+ xlab("Sepal Length") + ylab("Sepal Width") + ggtitle("Scatterplot with  
smoothers")
```



Donats els colors (per defecte) de les espècies *versicolor* i *virginica*, i la seva similitud en alguns patrons entre l'amplada i longitud del sèpal (width-length), utilitzar la forma ens ajuda.

d) Fent ús dels gràfics multipanells (*facets*) feu un gràfic de 3 files que contingui la mateixa informació que la figura anterior però on cada espècie es vegi separadament.

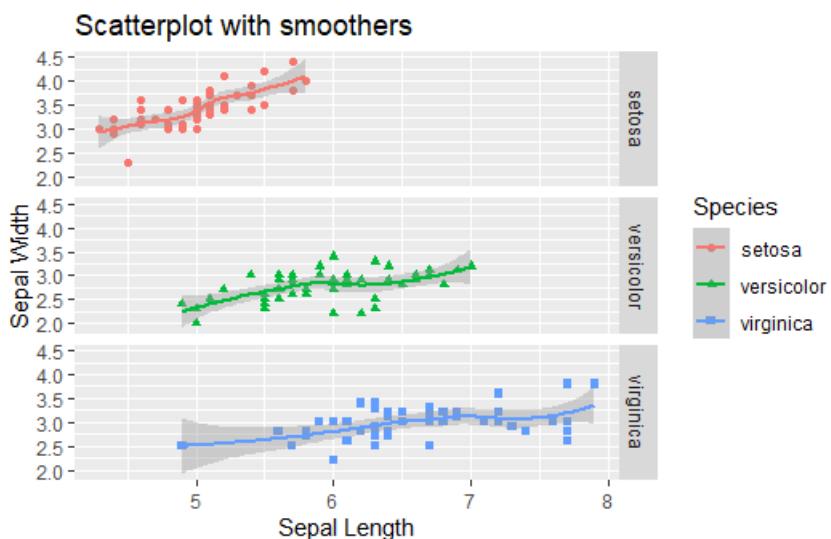
Podeu assignar en una variable la comanda de l'exercici anterior i sumar

```
>Ex2c<-ggplot(iris, aes(Sepal.Length, Sepal.Width, color=Species))
+geom_point(aes(shape=Species)) +geom_smooth() + xlab("Sepal Length") + ylab("Sepal
Width") + gtitle("Scatterplot with smoothers")
```

I podem afegir un canal *facet_wrap* d'una sola columna o un *facet_grid* on especifiquem que volem les Species per files

```
>Ex2c+facet_wrap(~ Species, ncol=1)
```

```
>Ex2c+ facet_grid(Species ~ .)
```

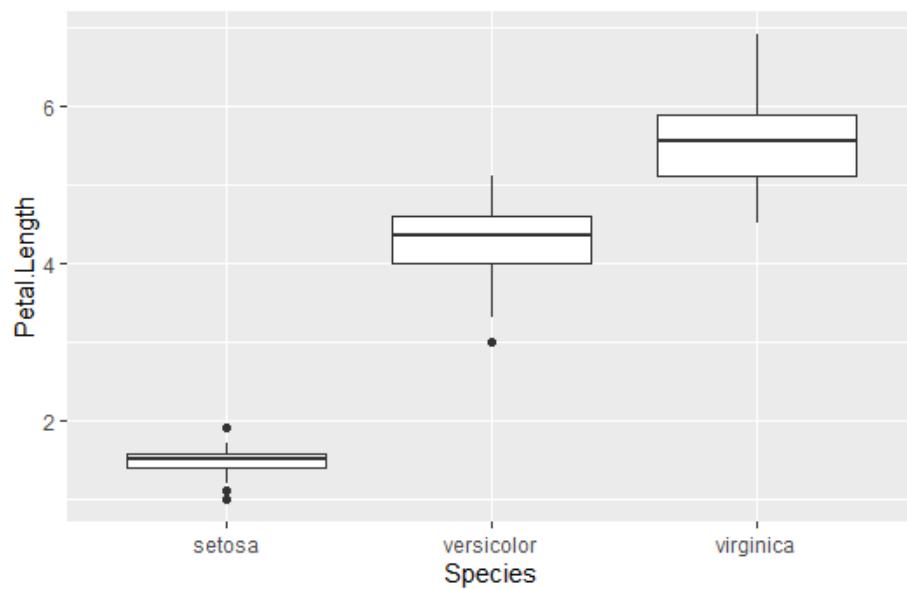


3.- Voleu conèixer alguns aspectes de la distribució de la longitud del sèpal segons l'espècie. Responeu en els següents apartats, què utilitzaríeu: un boxplot o un diagrama de violins. Feu ambdós gràfics i raoneu la resposta

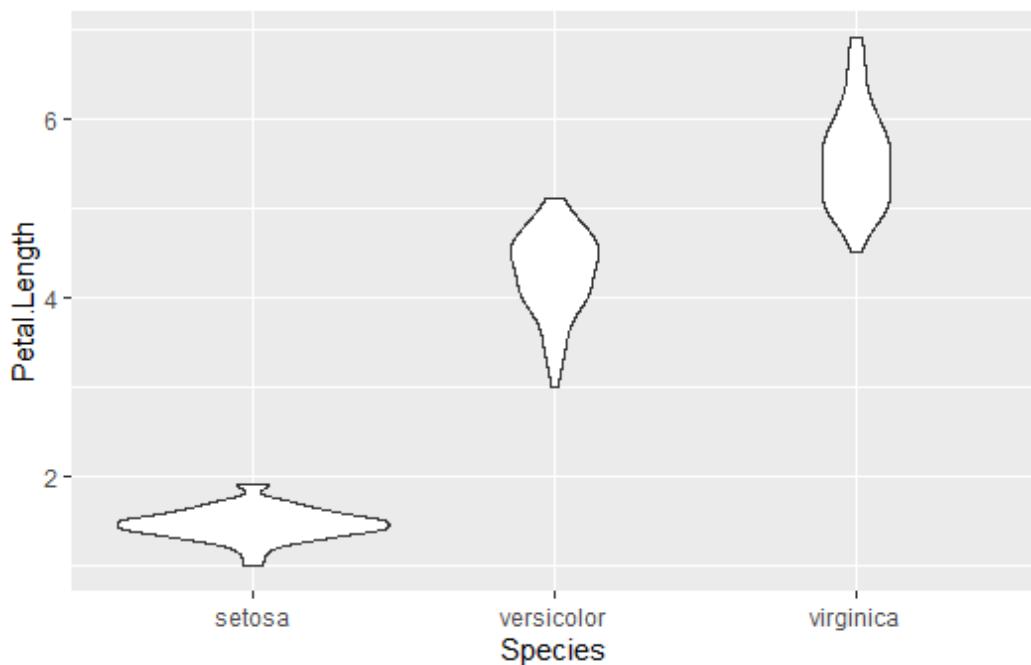
- Us interessa majorment conèixer les medianes i els *outliers* que teniu en cada espècie segons la longitud
- Us interessa majorment conèixer la distribució de la longitud del sèpal per cada espècie.
- En les visualitzacions de l'exercici 3.b, podeu afegir algunes mesures estadístiques fent us del canal *stat_summary*. Exemple: `stat_summary(fun=median, geom="point", color="red")` #fun és la funció estadística que volem afegir, podeu testejar la mitjana amb *mean*

Fem primer ambdós:

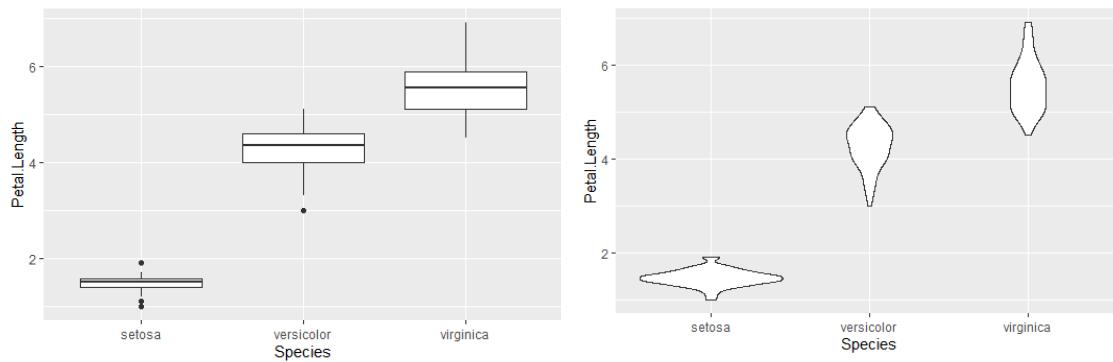
```
>ggplot(iris, aes(Species, Petal.Length)) +geom_boxplot()
```



```
>ggplot(iris, aes(Species, Petal.Length)) +geom_violin()
```



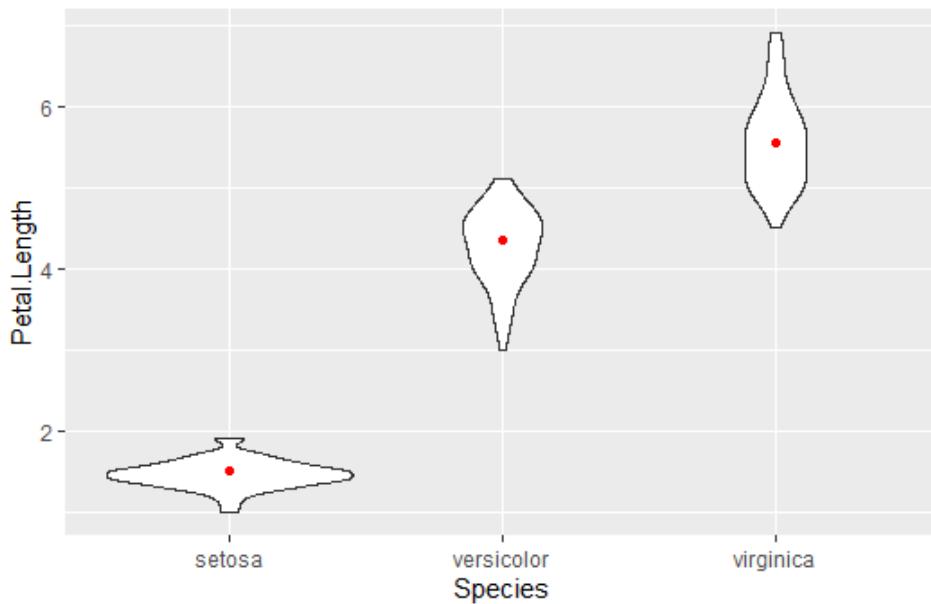
Si veiem ambdós un al costat de l'altre:



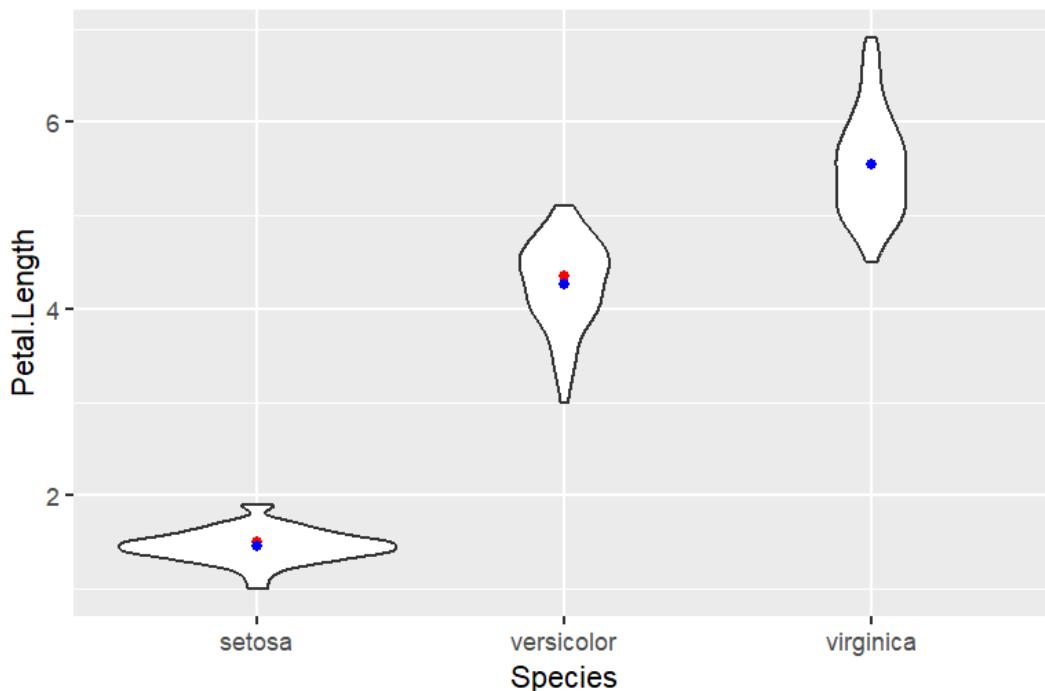
Els boxplots ens mostra molt clarament les medianes i outliers de la longitud dels pètals per cada espècie, i els violins la distribució d'aquests.

c) Podríem afegir algunes mesures estadístiques fent us del canal `stat_summary`. Per exemple afegim un punt vermell que ens marqui la mediana en cada violí:

```
>ggplot(iris, aes(Species, Petal.Length)) +geom_violin() +stat_summary(fun=median, geom="point", color="red") #fun és la funció estadística que volem afegir, podeu testejar la mitjana amb mean
```



```
>ggplot(iris, aes(Species, Petal.Length)) +geom_violin() +stat_summary(fun=median, geom="point", color="red") +stat_summary(fun=mean, geom="point", color="blue")
```



Veiem que la mitjana i mediana difereixen per la setosa i versicolor però no per la virginica.

3. PART 2. Data *tidying*

En aquesta segona part del seminari, anem a veure algunes eines de la llibreria *tidyverse* que ens permeten lidiar amb valors que hem perdut en un *dataframe*. Farem servir un *dataframe* basat en un *dataframe* de R que ja vam utilitzar en un seminari anterior: *mtcars*, i que conté 32 observacions i 11 variables sobre cotxes. Ara bé, en el nou *dataframe* hem perdut dades.

Farem servir les següents funcions:

Handle Missing Values

`drop_na(data, ...)`

Drop rows containing NA's in ... columns.

x	
x1	x2
A	1
B	NA
C	NA
D	3
E	NA

`drop_na(x, x2)`

`fill(data, ..., .direction = c("down", "up"))`

Fill in NA's in ... columns with most recent non-NA values.

x	
x1	x2
A	1
B	NA
C	NA
D	3
E	NA

`fill(x, x2)`

`replace_na(data, replace = list(), ...)`

Replace NA's by column.

x	
x1	x2
A	1
B	NA
C	NA
D	3
E	NA

`replace_na(x, list(x2 = 2))`

EXERCICIS:

- 1.- Primer de tot llegiu el *dataframe* que se us proporciona `df.csv`, fent us de `read.csv`. Assigneu-lo a una variable de nom `df_csv`. Verifiqueu que l'heu llegit bé

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
1	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
2	21.0	6	160.0	NA	3.90	2.875	17.02	0	1	4	4
3	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
4	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
5	18.7	NA	360.0	175	3.15	3.440	17.02	0	0	NA	2
6	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
7	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
8	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
9	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
10	NA	6	167.6	123	3.92	3.440	18.30	1	0	4	4
11	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4	4

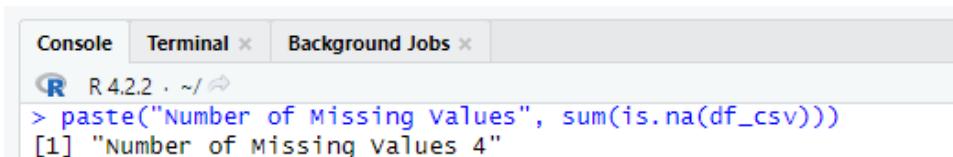
Showing 1 to 12 of 32 entries, 11 total columns

a) Digueu quant valors NA hi ha tot familiaritzant-vos amb la comanda:

```
> paste("Number of Missing Values", sum(is.na(df_csv)))
```

Carreguem la llibreria dplyr, llegim el *dataframe* i després fem ús de la comanda

```
> library (dplyr)
> df_csv <- read.csv("Directory/df.csv")
```



```
Console Terminal × Background Jobs ×
R 4.2.2 · ~/ ...
> paste("Number of Missing values", sum(is.na(df_csv)))
[1] "Number of Missing values 4"
```

Ens diu que tenim 4 valors NA.

b) Elimineu les files que contenen NA's en les columnes tot creant un nou *dataframe* de nom df_no_na. Verifiqueu que no hi ha NA's. Quantes observacions i columnes teniu respecte al *dataframe* inicial?

```
> df_no_na <- drop_na(df_csv)
> paste("Number of Missing Values", sum(is.na(df_no_na)))
```

Ens ha de retornar el missatge "Number of Missing Values 0".

A més si fem ús de la comanda str, podem observar que el dataframe inicial tenia 32 observacions i 11 columnes, mentre que el nou dataframe té 29 observacions. Drop-na ens ha eliminat les files que tenien un NA en alguna columna:

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
1	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
2	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
3	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
4	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
5	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
6	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
7	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
8	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4	4
9	16.4	8	275.8	180	3.07	4.070	17.40	0	0	3	3
10	17.3	8	275.8	180	3.07	3.730	17.60	0	0	3	3
11	15.2	8	275.8	180	3.07	3.780	18.00	0	0	3	3

Showing 1 to 12 of 29 entries, 11 total columns

```
> paste("Number of Missing values", sum(is.na(df_no_na)))
[1] "Number of Missing Values 0"
> str(df_csv)
'data.frame': 32 obs. of 11 variables:
 $ mpg : num 21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 NA ...
 $ cyl : int 6 6 4 6 NA 6 8 4 4 6 ...
 $ disp: num 160 160 108 258 360 ...
 $ hp : int 110 NA 93 110 175 105 245 62 95 123 ...
 $ drat: num 3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
 $ wt : num 2.62 2.88 2.32 3.21 3.44 ...
 $ qsec: num 16.5 17 18.6 19.4 17 ...
 $ vs : int 0 0 1 1 0 1 0 1 1 1 ...
 $ am : int 1 1 1 0 0 0 0 0 0 0 ...
 $ gear: int 4 4 4 3 NA 3 3 4 4 4 ...
 $ carb: int 4 4 1 1 2 1 4 2 2 4 ...
> str(df_no_na)
'data.frame': 29 obs. of 11 variables:
 $ mpg : num 21 22.8 21.4 18.1 14.3 24.4 22.8 17.8 16.4 17.3 ...
 $ cyl : int 6 4 6 6 8 4 4 6 8 8 ...
 $ disp: num 160 108 258 225 360 ...
 $ hp : int 110 93 110 105 245 62 95 123 180 180 ...
 $ drat: num 3.9 3.85 3.08 2.76 3.21 3.69 3.92 3.92 3.07 3.07 ...
 $ wt : num 2.62 2.32 3.21 3.46 3.57 ...
 $ qsec: num 16.5 18.6 19.4 20.2 15.8 ...
 $ vs : int 0 1 1 0 1 1 0 0 ...
 $ am : int 1 1 0 0 0 0 0 0 0 ...
 $ gear: int 4 4 3 3 3 4 4 4 3 3 ...
 $ carb: int 4 1 1 1 4 2 2 4 3 3 ...
> |
```

- c) Enlloc d'eliminar les files que contenen NA's en les columnes 'mpg' o 'hp', completeu els valor que falten a les columnes on hi ha NA. En el cas de la variable 'mpg' feu-ho amb l'entrada següent. En el cas de la variable 'hp' feu-ho amb l'entrada anterior.

```
> df_na_filled <- df_csv %>% fill(mpg,.direction =
"down")%>%fill(hp,.direction="up")
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
1	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
2	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
3	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
4	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
5	18.7	NA	360.0	175	3.15	3.440	17.02	0	0	NA	2
6	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
7	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
8	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
9	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
10	17.8	6	167.6	123	3.92	3.440	18.30	1	0	4	4
11	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4	4

Showing 1 to 12 of 32 entries, 11 total columns

- d) Enlloc d'eliminar les files que contenen NA's en les columnes que queden 'cyl' o 'gear', reemplaçeu-les per un 0 fent servir `mutate_all(replace_na,0)`

```
> df_na_filled <-df_na_filled%>%mutate_all(replace_na,0)
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
1	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
2	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
3	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
4	21.4	6	256.0	110	3.08	3.215	19.44	1	0	3	1
5	18.7	6	360.0	175	3.15	3.440	17.02	0	0	3	2
6	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
7	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
8	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
9	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
10	17.8	6	167.6	123	3.92	3.440	18.30	1	0	4	4
11	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4	4

4. PART 3. Sistemes Avançats.

Dataframe: simpsons_episodes.csv. Conjunt de dades amb els detalls d'aproximadament 600 episodis dels Simpson.

EXERCICIS:

1.- Feu una gràfica que permeti veure la distribució del nombre de visualitzacions ('views') de la primera temporada ('season'). Expliqueu l'elecció del gràfic i les conclusions que podeu extreure'n.

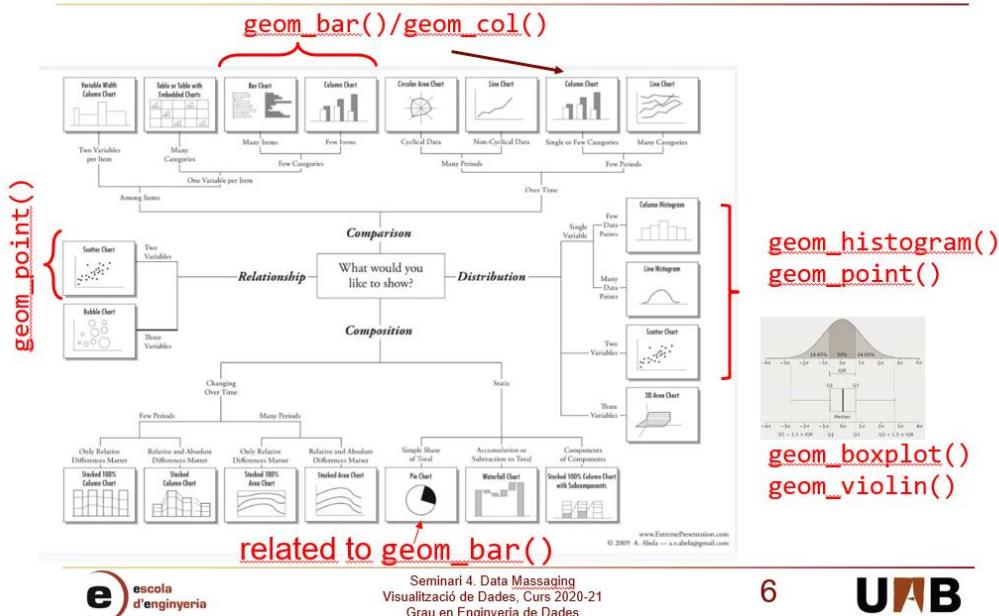
Llegim el fitxer

```
>simpsons_episodes <- read_csv('data/simpsons_episodes.csv') #o via el Environment
```

Primer necessitem filtrar la primera temporada. Sembla que no hi ha nans ni valors 'nulls' en aquesta (pel que fa a les visualitzacions), per tant podem prosseguir.

En quant al gràfic, tenim una variable numèrica continua "views" i se'ns demana una distribució. Hem vist varies vegades a classe (teòrica amb Guillermo i en seminaris) quines gràfiques es podien usar per mostrar una distribució:

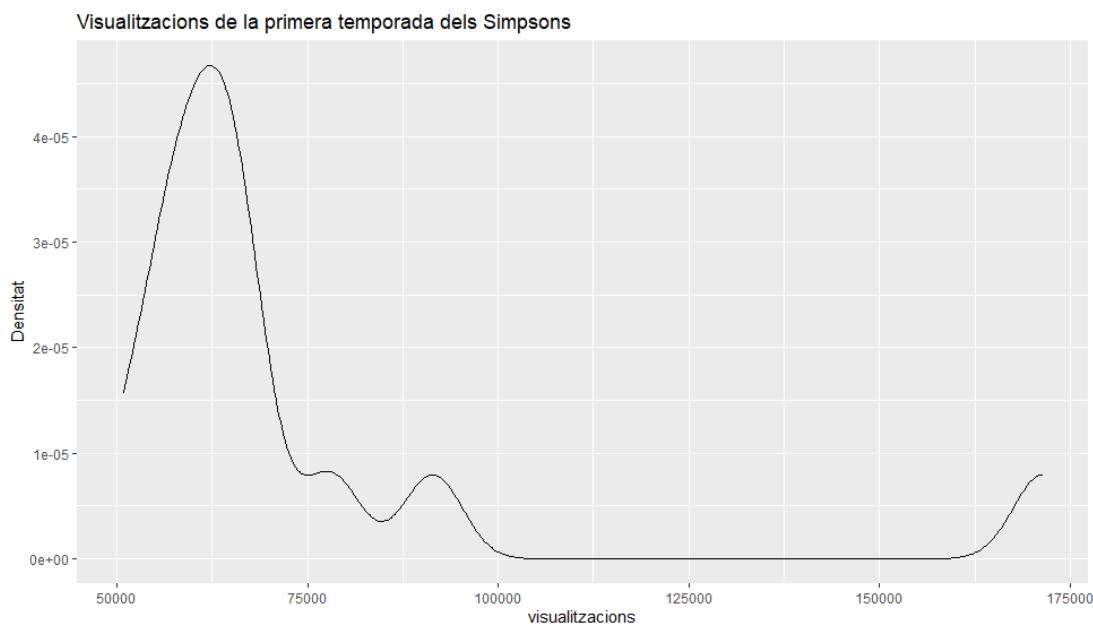
1.3. Quick summary: R tools (geom_)



6

Un cop identificades els gràfic òptims per mostrar distribucions (*column histogram*, *line histogram*, *scatter chart*, *3D area chart*, *boxplots*, *diagrama de violins*, per exemple), només hem de buscar quin/s d'ells serveix/en per la distribució d'una variable contínua, com és “views”. Tenim que és un gràfic de densitats i/o histograma. Per tant, fem aquest tipus de gràfic:

```
>simpsons_episodes %>% filter(season==1) %>% drop_na(views) %>% ggplot(aes(views)) + geom_density() + xlab('visualitzacions') + ylab('Densitat') + ggtitle('Visualitzacions de la primera temporada dels Simpsons')
```



2.- Feu una gràfica que permeti veure la distribució de les 10 primeres temporades ('season') respecte al nombre de visualitzacions ('views'). Expliqueu l'elecció del gràfic i les conclusions que podeu extreure'n.

Primer necessitem filtrar les 10 primeres temporades. Sembla que no hi ha nans ni valors 'nulls' en aquestes (pel que fa a les visualitzacions), per tant podem prosseguir.

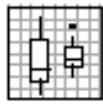
En quant al gràfic, tenim una variable numèrica continua "views" i una variable 'season' que podem fer discreta categòrica amb factor i se'ns demana una distribució. Tornem a mirar la diapositiva citada en l'exercici anterior

Un cop identificades els gràfic òptims per mostrar distribucions (*column histogram, line histogram, scatter chart, 3D area chart, boxplots, diagrama de violins, per exemple*), només hem de buscar quins d'ells serveixen pel nostre tipus de variables, fent servir el xuletari:

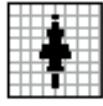
Discrete X, Continuous Y
`f <- ggplot(mpg, aes(class, hwy))`



`f + geom_bar(stat = "identity")`
`x, y, alpha, color, fill, linetype, size, weight`



`f + geom_boxplot()`
`x, y, lower, middle, upper, ymax, ymin, alpha, color, fill, group, linetype, shape, size, weight`



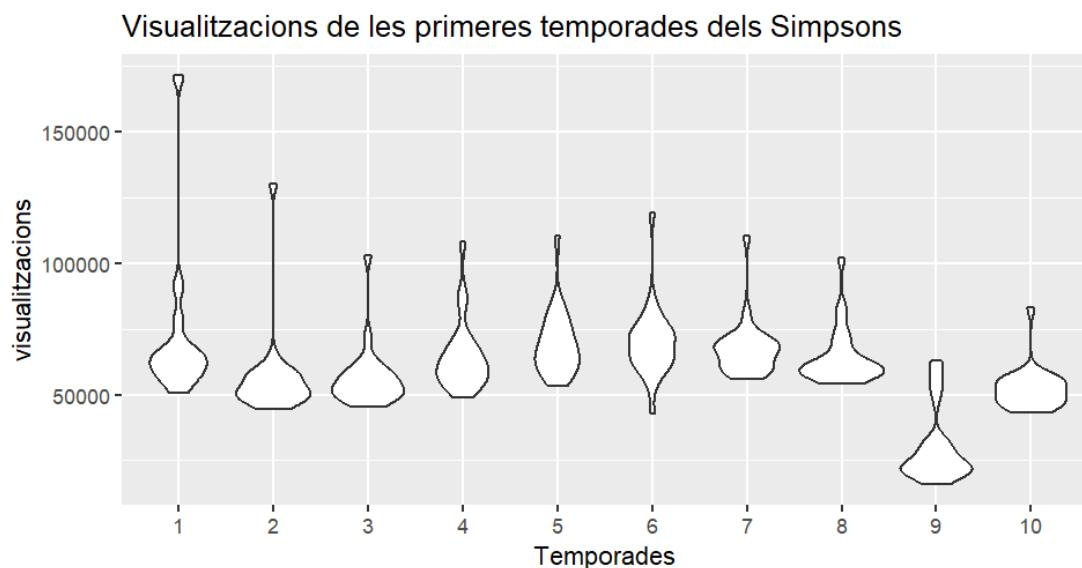
`f + geom_dotplot(binaxis = "y", stackdir = "center")`
`x, y, alpha, color, fill, group`



`f + geom_violin(scale = "area")`
`x, y, alpha, color, fill, group, linetype, size, weight`

Podem doncs fer per exemple, un `geom_boxplot()` o un `geom_violin()`. Ara bé, l'enunciat s'interessa per la distribució, no ens demana ni outliers, ni quartils, ni medianes, per tant ambdós són òptims, i si volem per exemple saber la 'forma' de la distribució `geom_violin()` pot ser bona elecció:

```
>simpsons%>%filter(season<=10)%>%drop_na(views)%>%ggplot(aes(x=factor(season), y=views)) +geom_violin() +theme(legend.position='none')+xlab('Temporades')+ylab('visualitzacions')+ggtitle('Visualitzacions de les primeres temporades dels Simpsons')
```



El gràfic ens mostra clarament una davallada de les visualitzacions durant la temporada 9, o que en la temporada 1 va haver algun episodi que va tenir més de 150000 visualitzacions, tot i que la majoria dels episodis van tenir unes 60000 visualitzacions. A més, els diagrames de violí de les tres primeres sessions tenen cues més llargues, indicant que alguns dels episodis tenien puntualment més visualitzacions que la resta de la mateixa temporada.

Judit Chamorro Servent
Bellaterra, Març 2025