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>               <int> <dbl> <chr>         <chr>       <chr>           <dbl> <chr>  <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 *dyplr* 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 *dyplr*

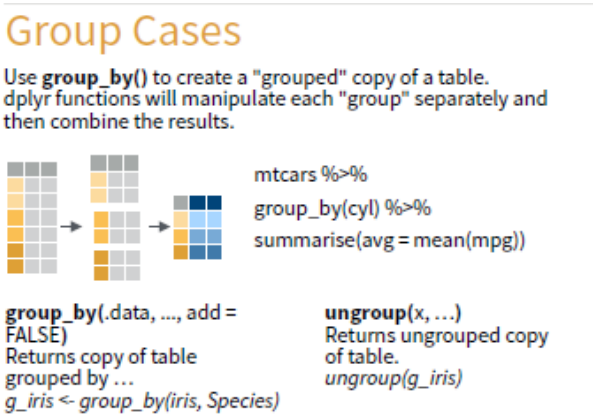
```
> library(tidyverse)
```

```
> 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.




```
> 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>               <int> <dbl> <chr>         <chr>       <chr>          <dbl> <chr>  <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
>
```

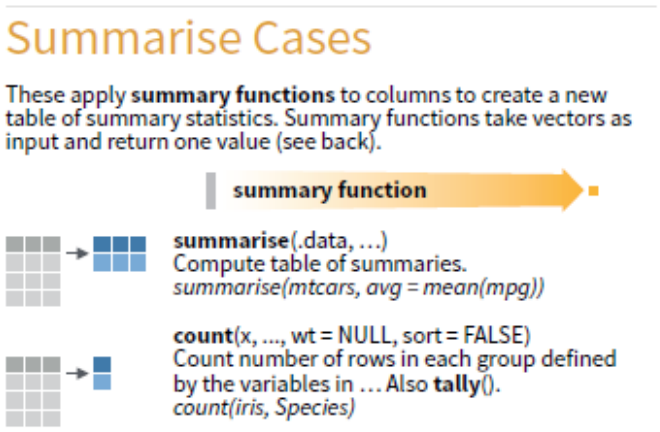
Es pot veure que el resultat ens dona el nombre de grups en la segona línia del missatge de sortida. Si per cada grup volem aplicar algun càlcul, cal que definim funcions d'agregat associades (count(), mean(), min(), max(), sum()), com en el llenguatge SQL de Bases de Dades.

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



Primer agrupem i després comptem:

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

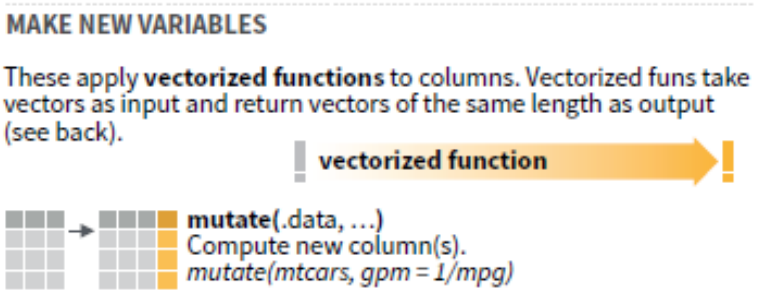
Nota: Abans de fer aquest exercici veieu què passa si fem la mitjana d'una variable que conté valors NA. Per això proveu:

```
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 especifiquem 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()`.



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/mean(height,na.rm=TRUE)) #simplificant-ho
```

```
> starwars%>%select(name,height,species)%>% mutate(height/mean(height,na.rm=TRU
E))
# 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))
```

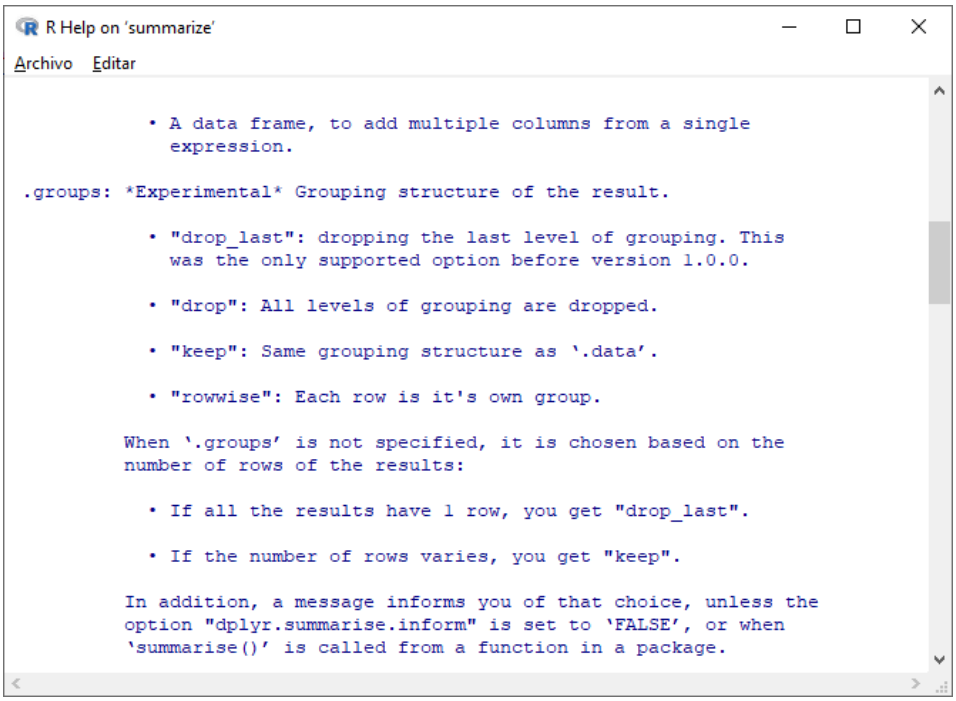
```
>  starwars %>% filter(gender == "masculine") %>% group_by(species) %>%summarize(mean(height, na.rm = TRUE))
`summarise()` ungrouping output (override with `.groups` argument)
# A tibble: 33 x 2
   species    `mean(height, na.rm = TRUE)`
   <chr>                             <dbl>
 1 Aleena                             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`



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 dyplr, 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:

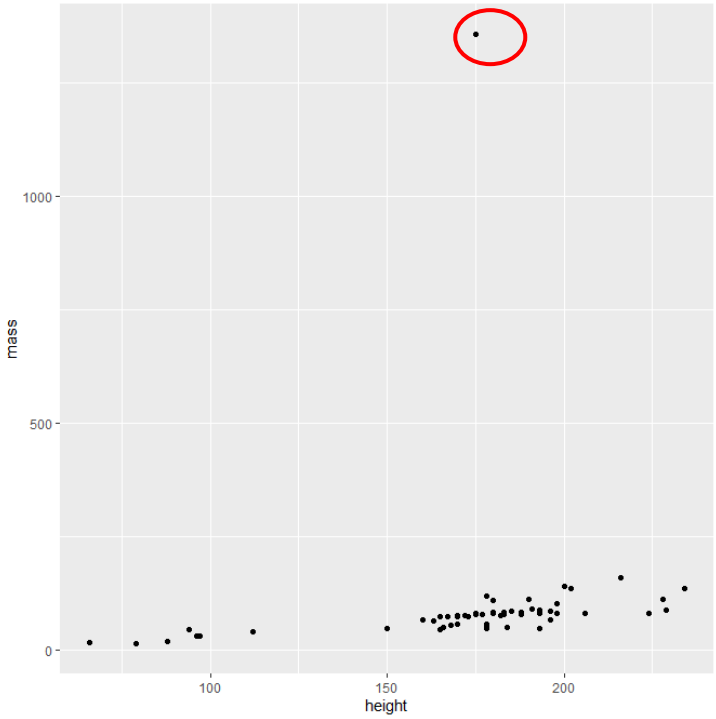
```
> ggplot(starwars)+aes(height,mass)+geom_point()
> #or:
> ggplot(starwars, aes(height,mass))+geom_point()
Warning message:
Removed 28 rows containing missing values (geom_point).
```

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 `filtre()`:

```
> 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()
```

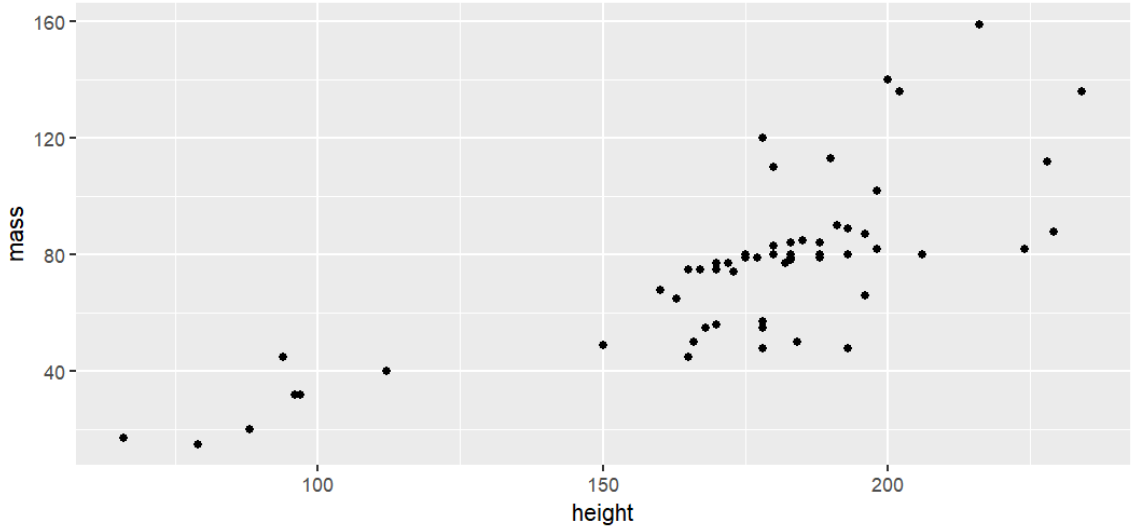
Si no som fans (o ens trobem davant un altre dataset desconegut per nosaltres) ens podem quedar amb els personatges que no estan per sobre un cert valor d'umbralització o *threshold*:

```
> 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()
```

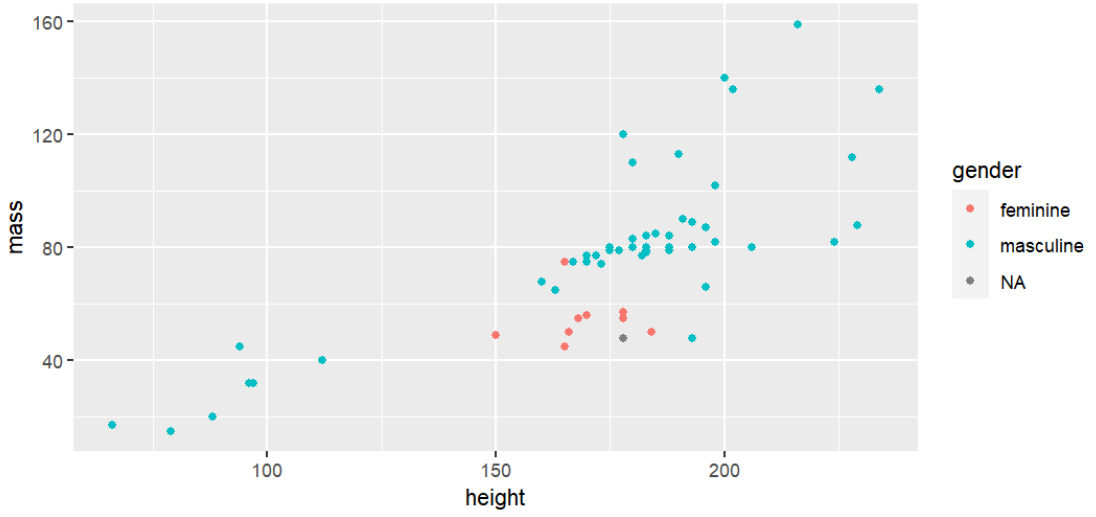
En aquest cas, el gràfic que ens surt ara és:



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

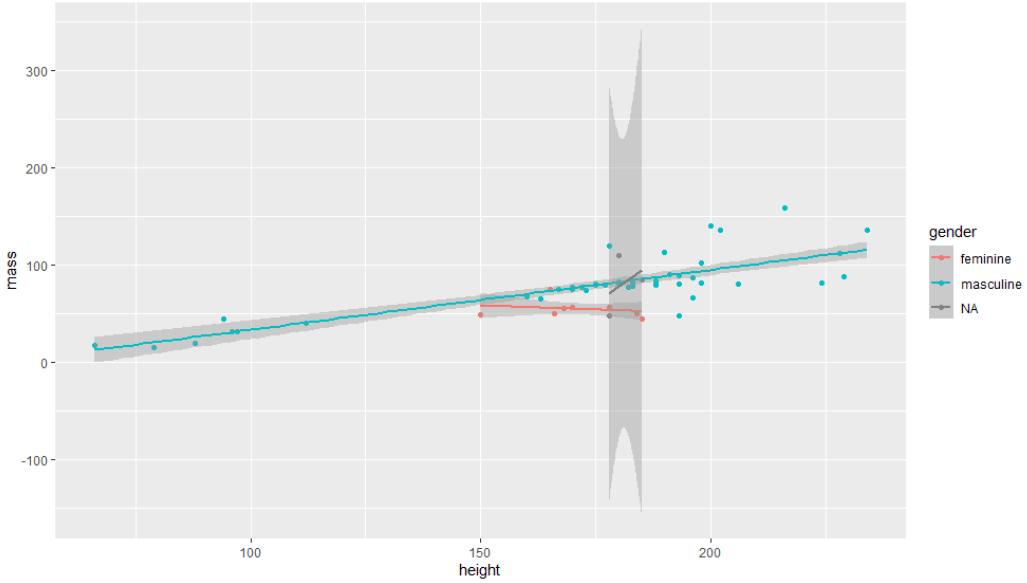
```
> ggplot(starwars3, aes(height,mass,color=gender))+geom_point()
```



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.

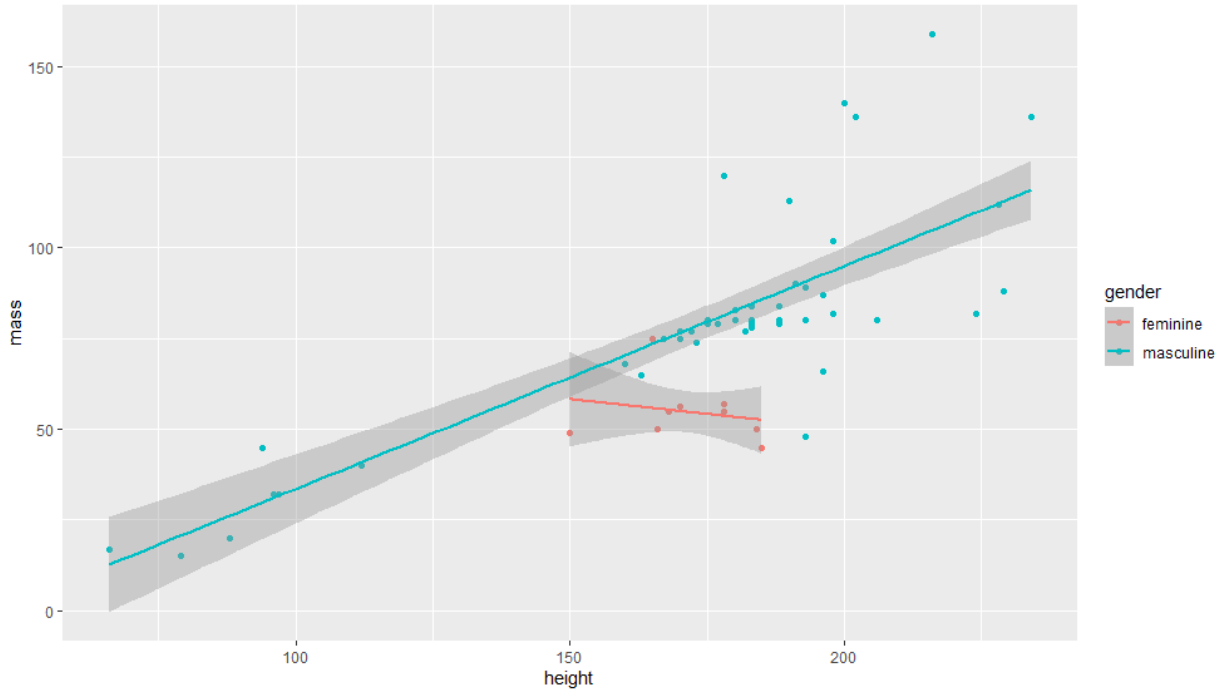
```
>ggplot(starwars3, aes(height,mass,color=gender))+geom_point()+geom_sm
ooth(method="lm", level=0.9)
```



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 especies 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.

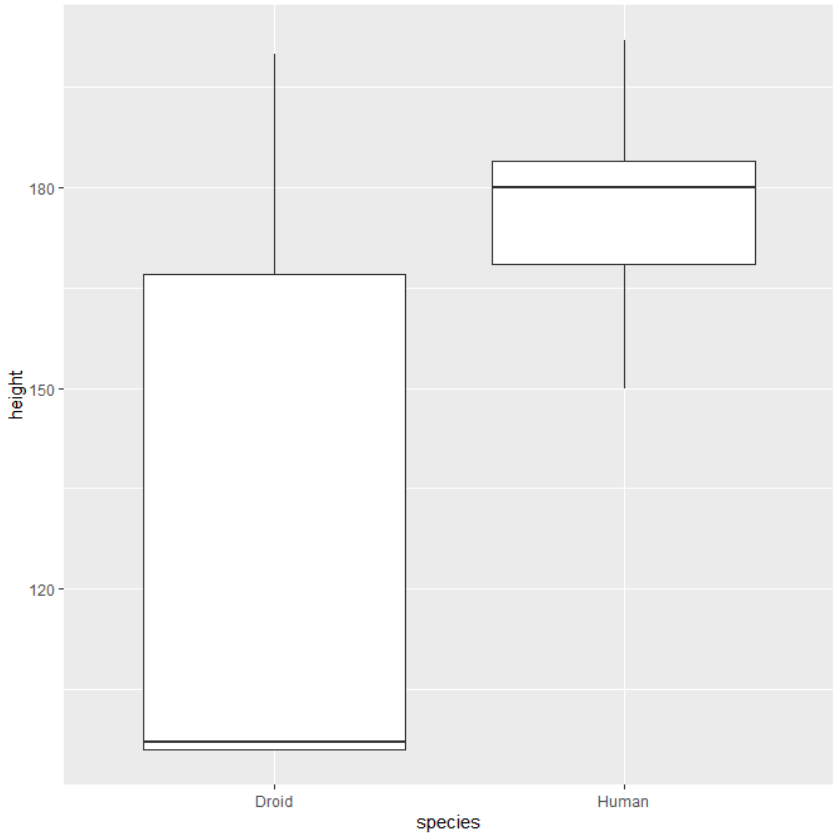
a) Si fem per exemple el cas que ens diu dels humans i robots:

```
> 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 especies:

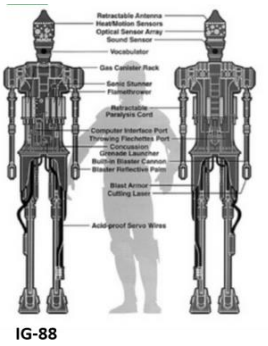
```
> GrupXspecie <-starwars2%>%group_by(species)%>%count()
> GrupXspecie

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

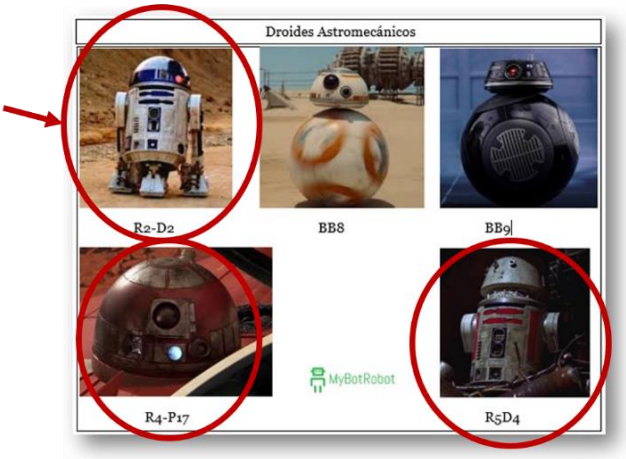
Tenim només 5 robots, per tant el boxplot amb tant poques dades ens pot enganyar (les alçades de 5 personatges difícilment seguiran una distribució normal o una distribució coneguda).

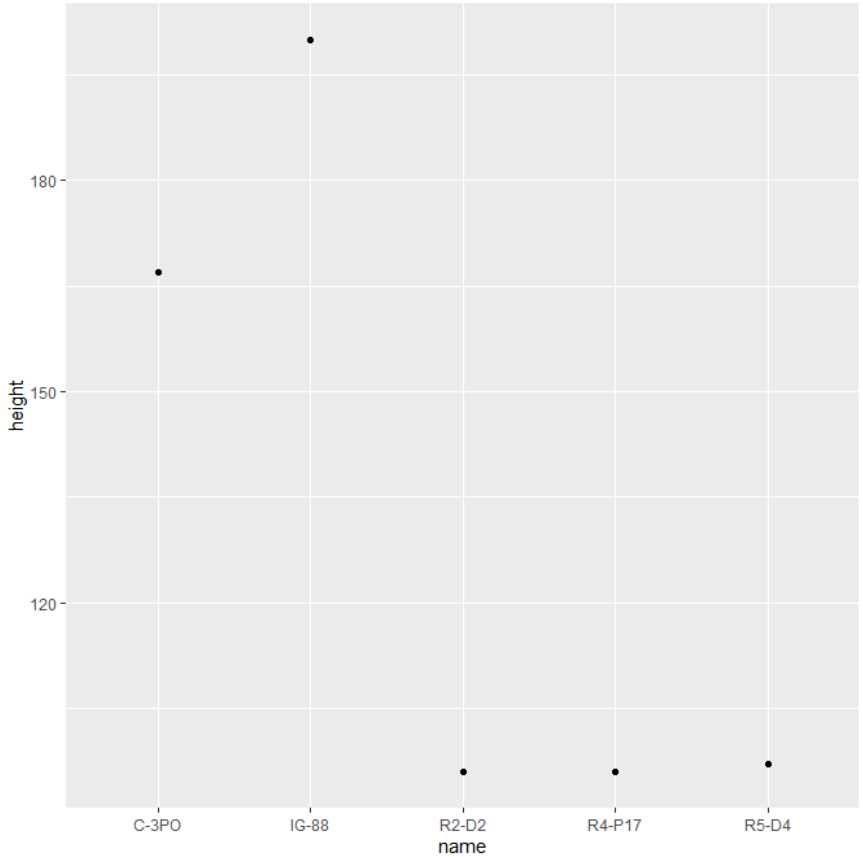
c) Al fer el *scatter plot*:

```
> 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)

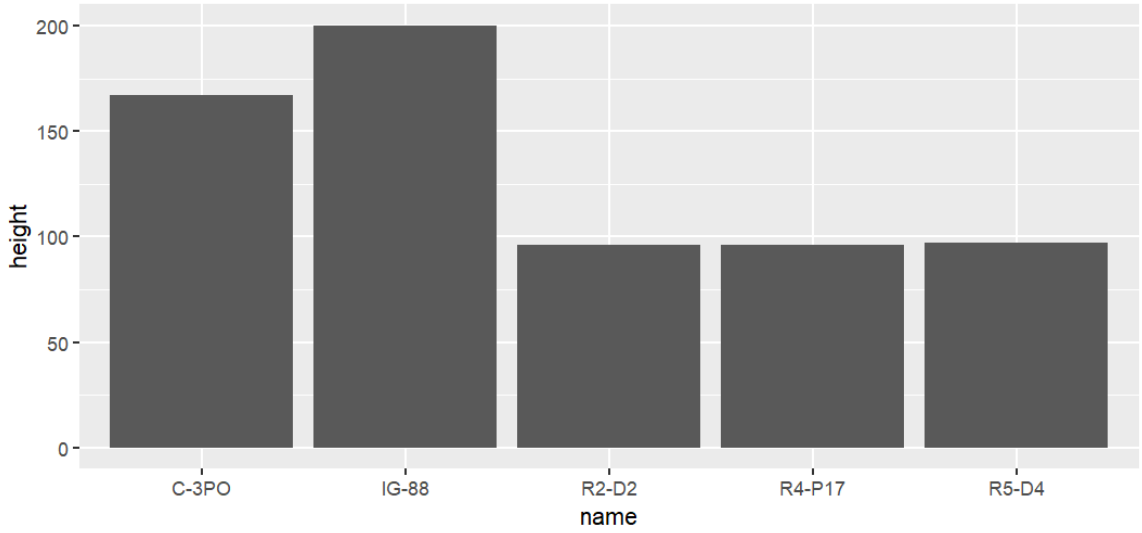






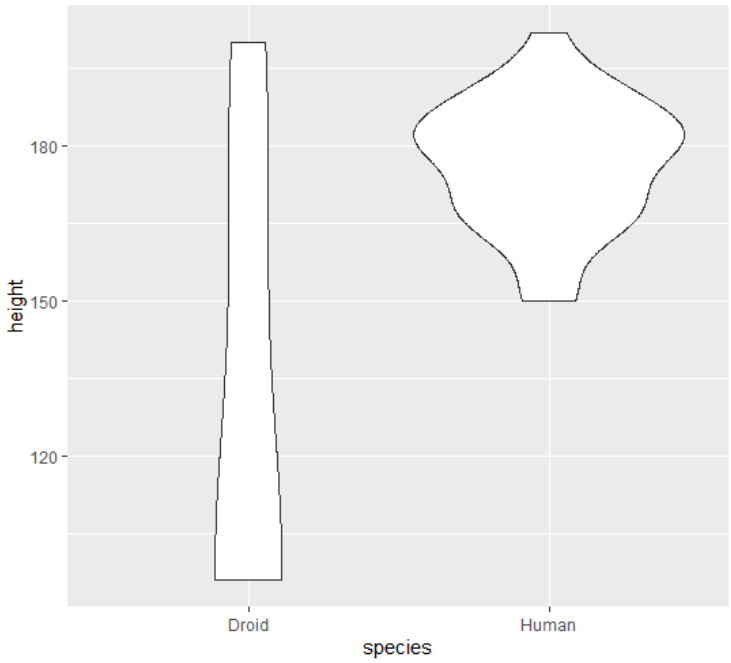
El *scatter plot* aquí ens mostra realment la diferencia d'alçades que veiem en les figures dels personatges.

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:

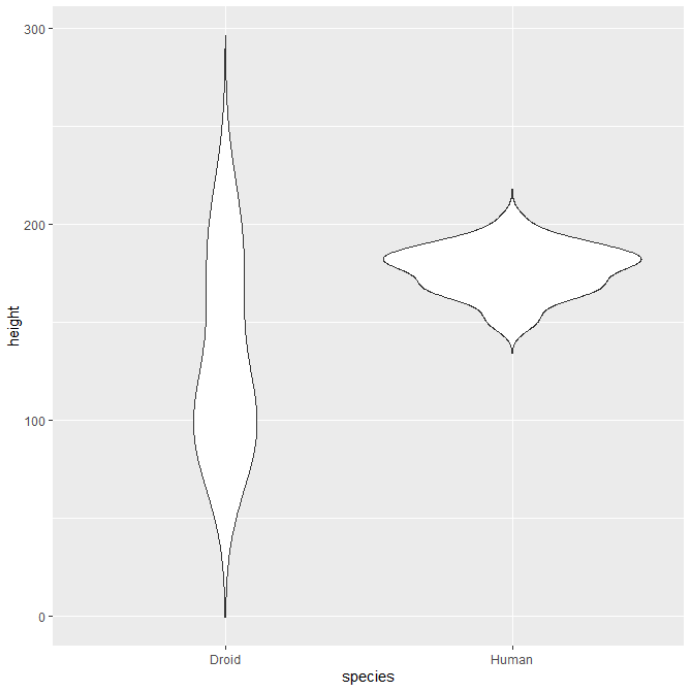
```
>  starwars2<-starwars  %>%filter(species=="Human"|species==  "Droid")
%>%filter(height!="NA")
> ggplot(starwars2, aes(species, height))+geom_violin()
```



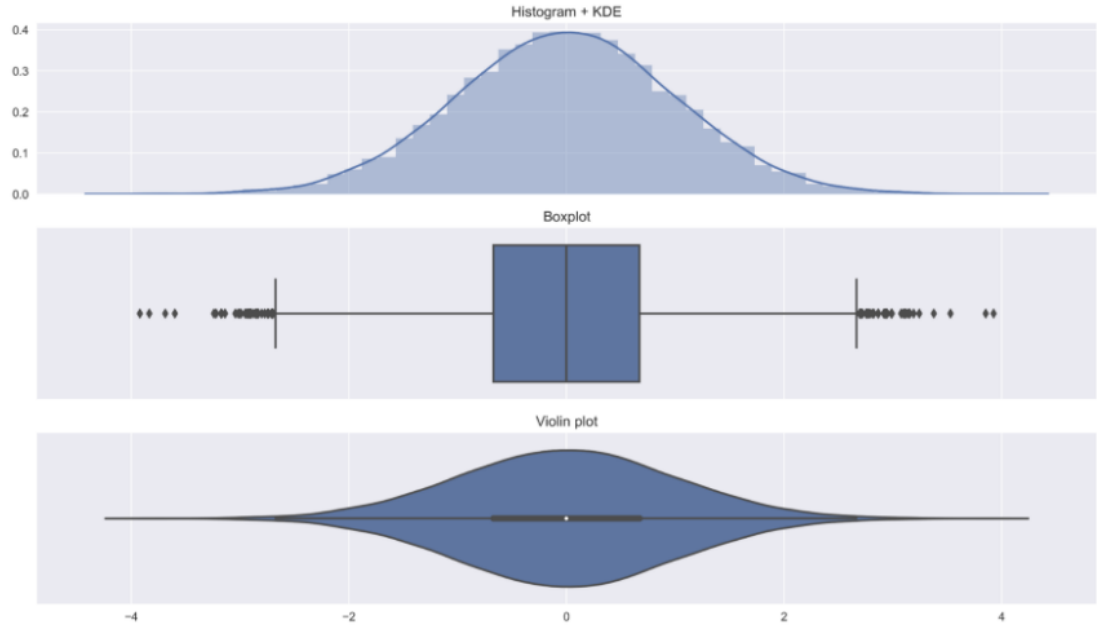
Si no volem que ens talli les cues dels violí segons les dades, fem `?geom_violin` on veiem entre d'altres opcions:

```
  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 (tidyr)* & gràfiques avançades

En aquesta tercera part del seminari utilitzarem el dataframe iris. Utilitzarem algunes funcions de la llibreria tidyr 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.Lenght**, **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	5.4	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 *tidyr* 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(columnes) menys species

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

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 especifiquem 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:

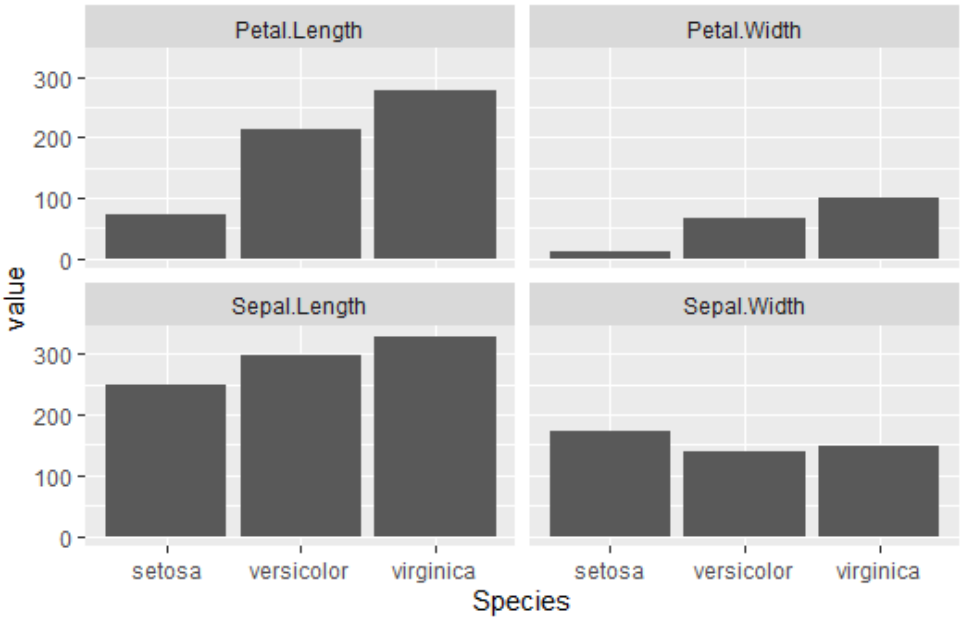
- Volem un subgrafic per cada mètrica per tant necessitarem fer ús del canal: `facet_wrap(~ metric)`
- A més, volem un gràfic que en cada una de les 4 finestres referent a la mètrica ens compari els valors d'aquesta per diferent espècies. I vam veure que `geom_bar` i `geom_col` servien per comparar valors.
- 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