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 múltiple 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.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.

EXERCICIS:

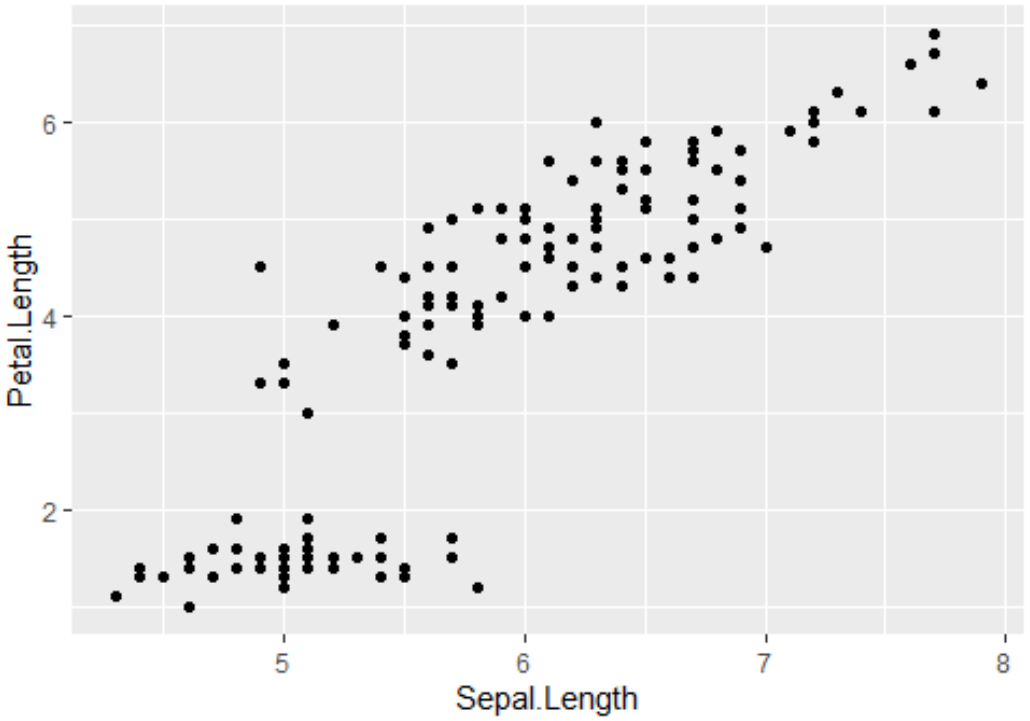
1.- Suposem que volem veure la correlació entre dues variables Sepal.Length i Petal.Length.

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()
```

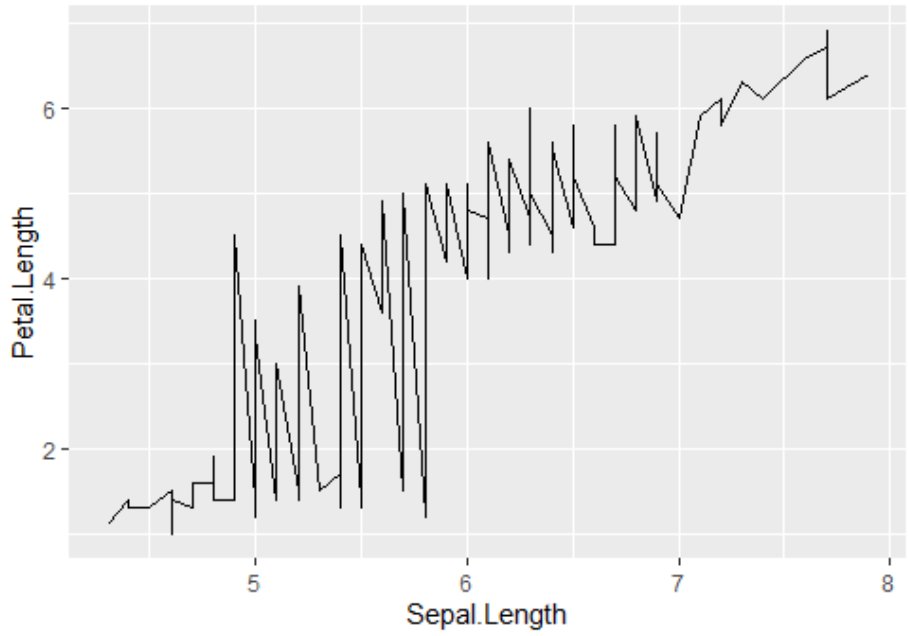


Veiem que per valors de Sepal.Length petits (grans) tenim valors petits (grans) de Petal.Lenght. Però a més hi a partir d'una Petal.Lenght més gran que 3cm, tenim que les Sepal.Length guarden una relació molt diferent a quan aquesta és menor que 3cm.

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 presencia d' *outliers* en les dades utilitzaríem "`rlm`" (model lineal robust) per minimitzar la influencia 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 rlm 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()
```

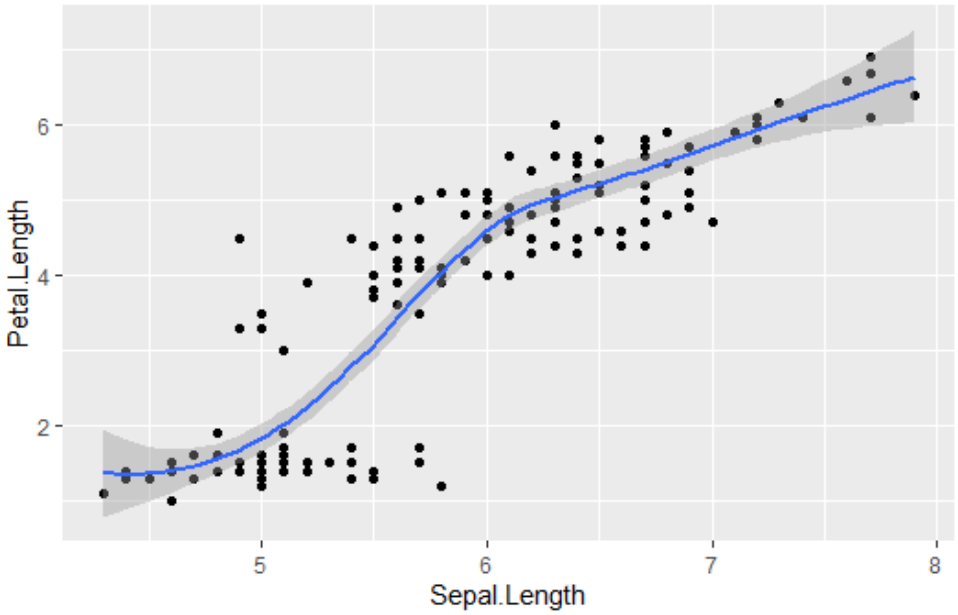
Provarem els tres mètodes amb una regió de confiança del 90%, com ens demana el problema:

```
>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 diferencies. glm funciona millor que lm sol quan hi ha una variable resultat que funciona com a binaria.

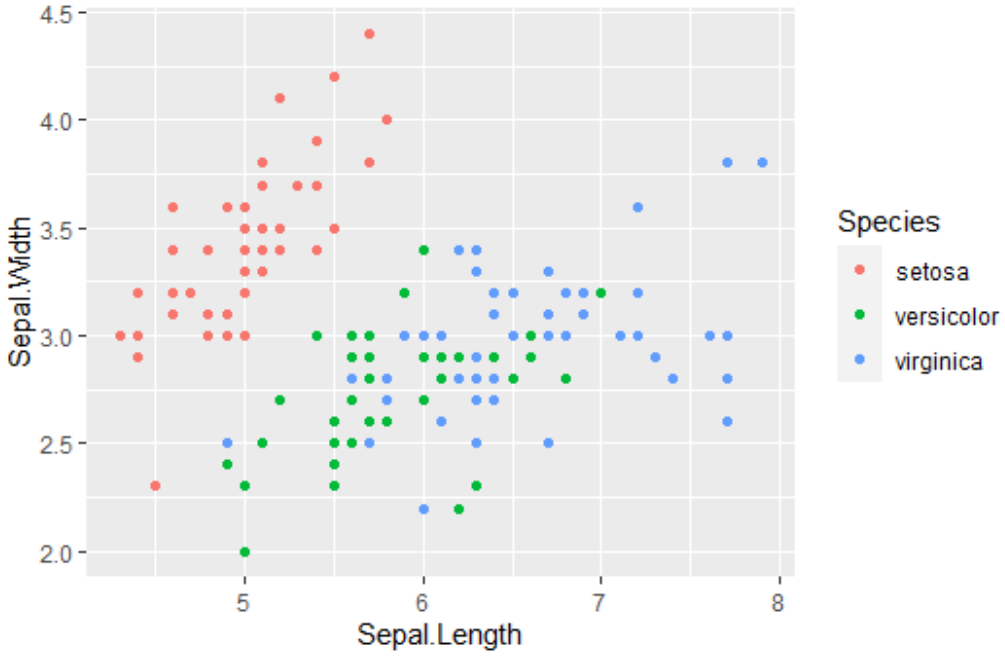
Nota: El rlm no funciona per algunes versions de R.

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.

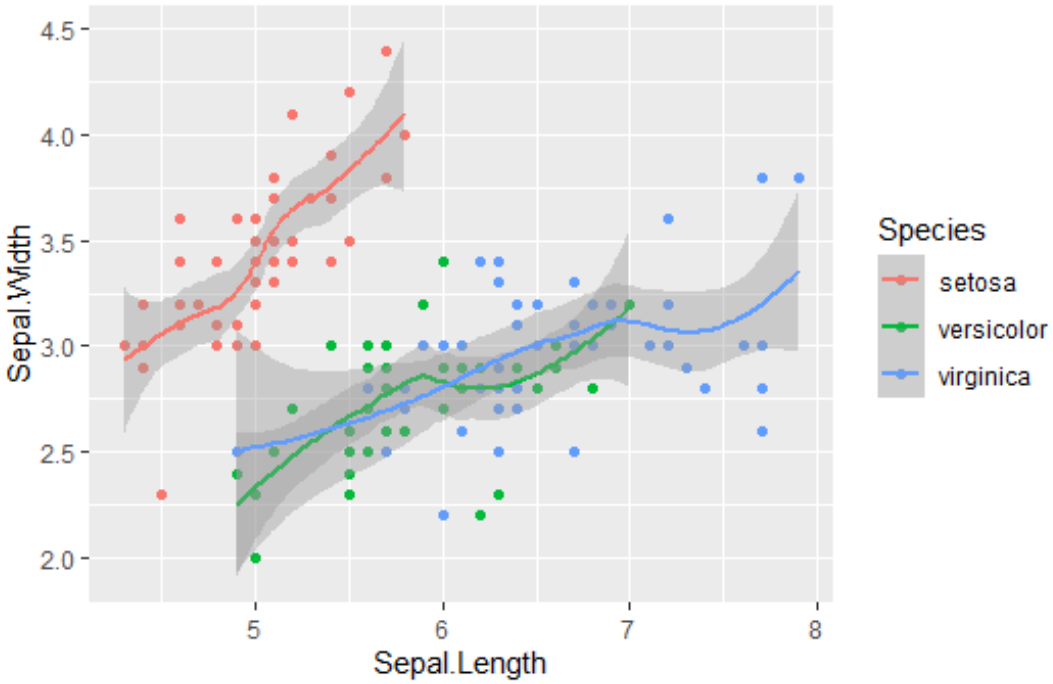
Dins d'aes fem un mapeig de color segons les 'species'

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



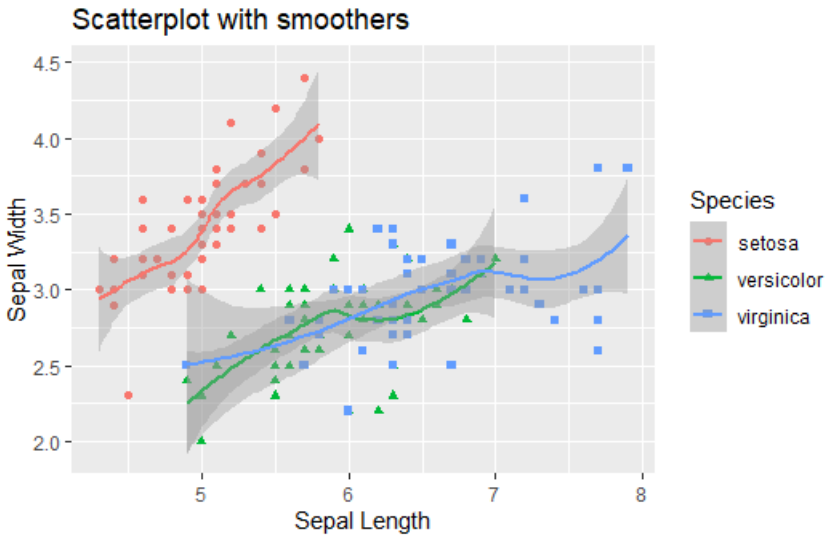
b) Com en l'exercici anterior, agregueu un canal que afegeixi al gràfic de punts de l'apartat (a), un ombrejat basat en càlculs estadístics de funcions de suavitzat (canal '`stat/geom_smooth()`').

>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-lenght), utilitzar la forma ens ajuda.

d) Fent ús dels gràfics multipanels (*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") + ggtitle("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

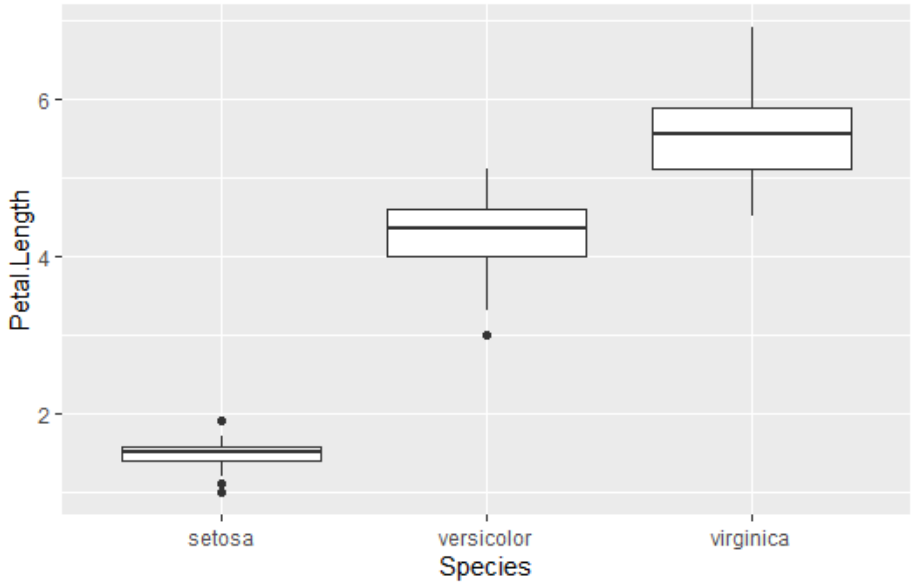
a) Us interessa majorment conèixer les medianes i els *outliers* que teniu en cada espècie segons la longitud

b) Us interessa majorment conèixer la distribució de la longitud del sèpal per cada espècie.

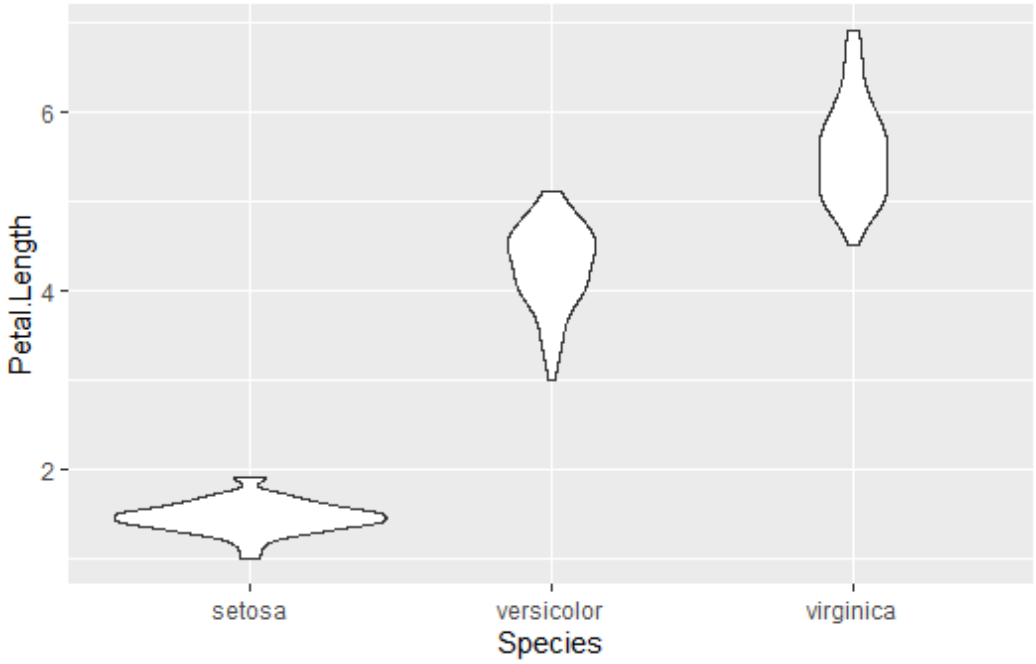
c) En les visualitzacions de l'exercici 3.b, podeu afegir algunes mesures estadístiques fent us del canal `stat_summary`. Exemple: `stat_summary(fun=median, geom="point", color="red")` #fun és la funció estadística que volem afegir, podeu testejar la mitjana amb *mean*

Fem primer ambdós:

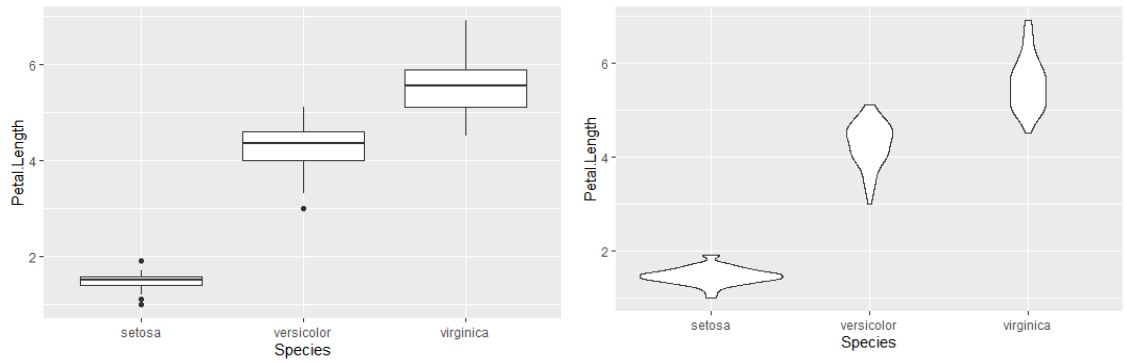
```
>ggplot(iris, aes(Species, Petal.Length)) +geom_boxplot()
```



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



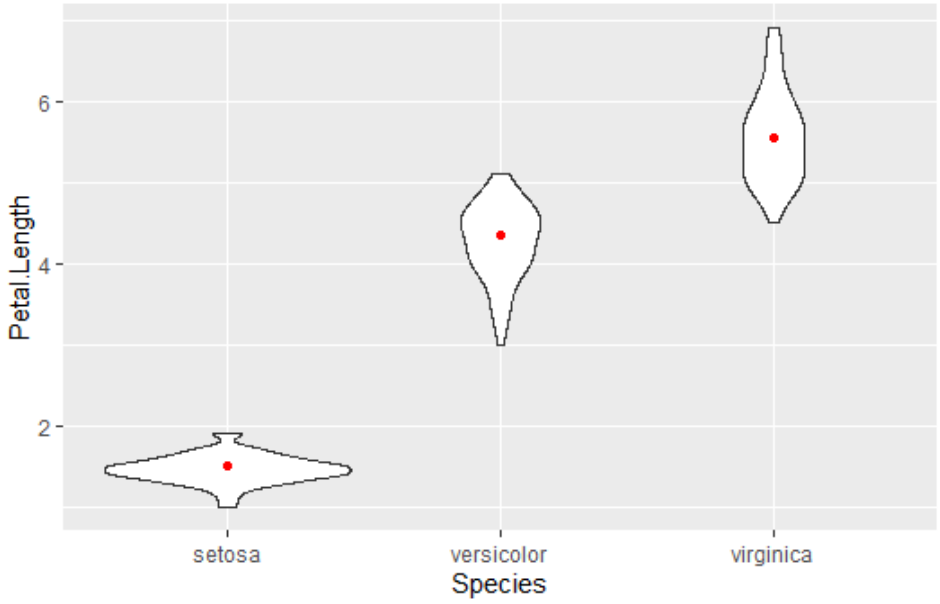
Si veiem ambdós un al costat de l'altre:



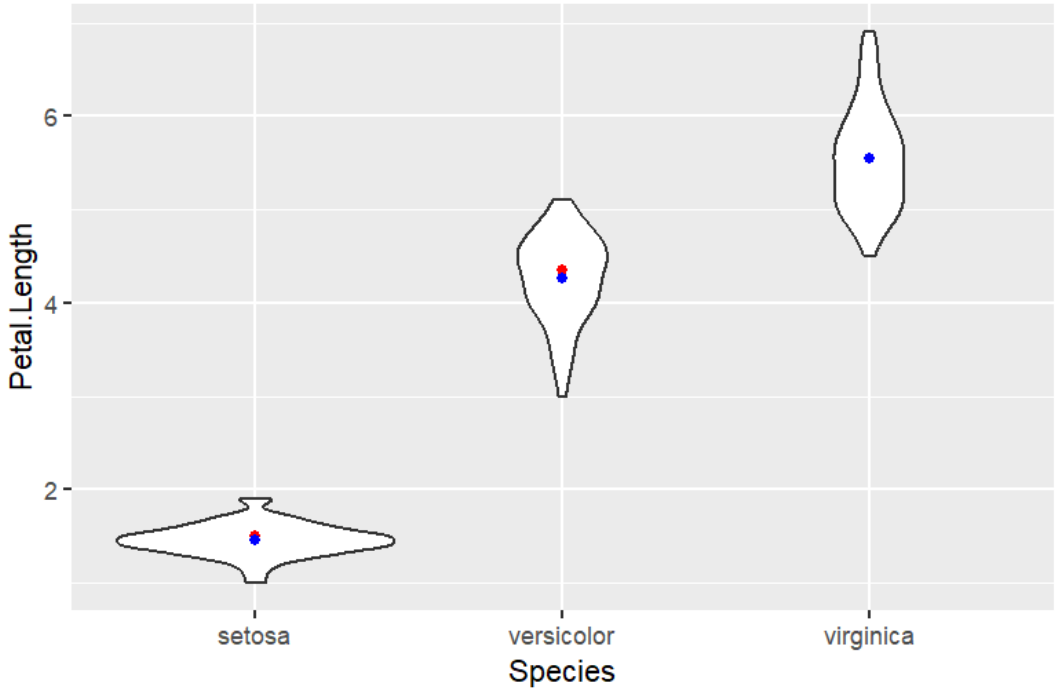
Els boxplots ens mostra molt clarament les medianes i outliers de la longitud dels pètals per cada espècie, i els violins la distribució d'aquests.

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")**

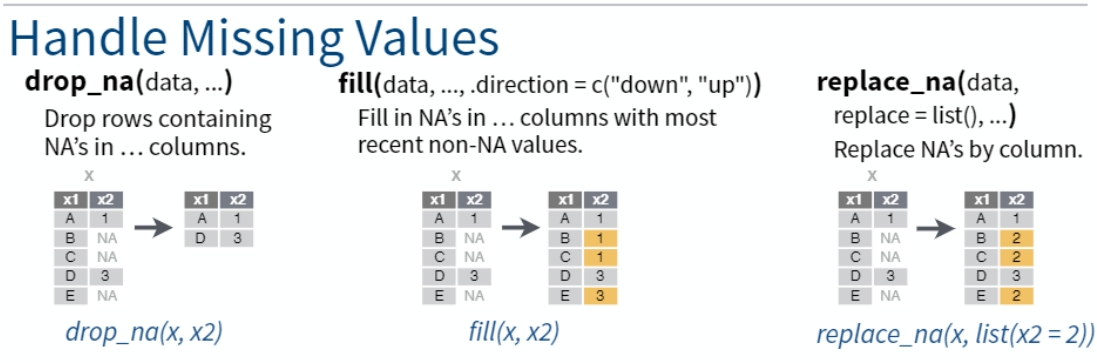


Veiem que la mitjana i mediana difereixen per la setosa i versicolor però no per la virginica.

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:



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

Filter

	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 *datafr ame* de nom df_no_na. Verifiqueu que no hi ha NA's. Quantes observacions i col umnes 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 1 0 1 1 1 0 0 ...
 $ am  : int  1 1 0 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', co mpleteu 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'entr ada anterior.

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

Filter

	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', reemplaceu-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	258.0	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:

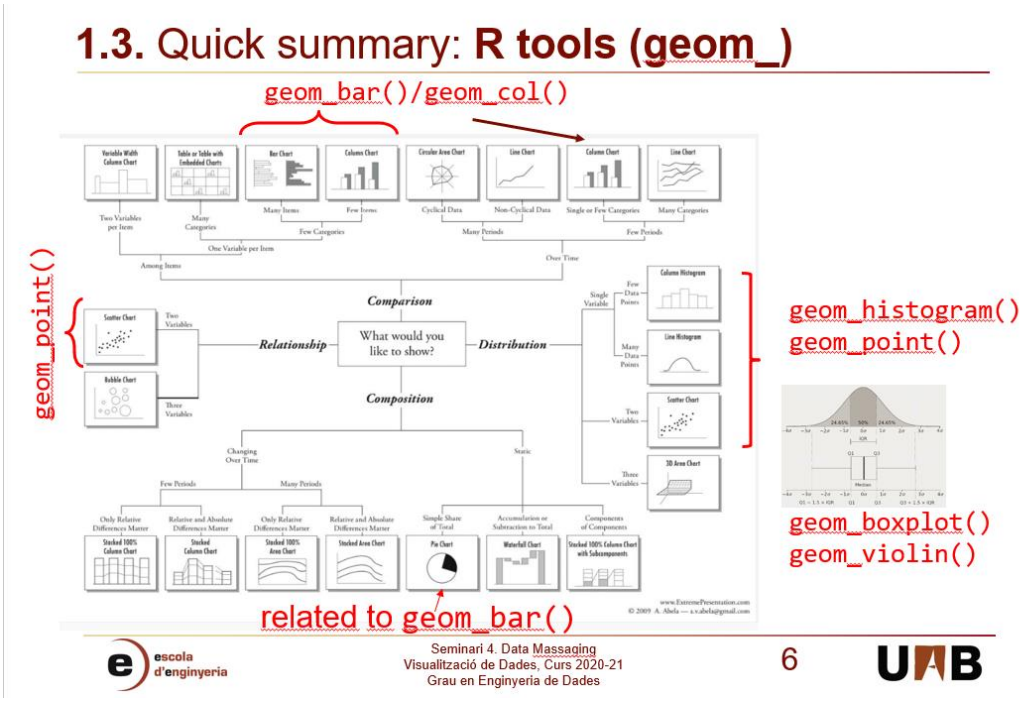
1.- Feu una gràfica que permeti veure la distribució del nombre de visualitzacions ('views') de la primera temporada ('*season*'). Expliqueu l'elecció del gràfic i les conclusions que podeu extreure'n.

Llegim el fitxer

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

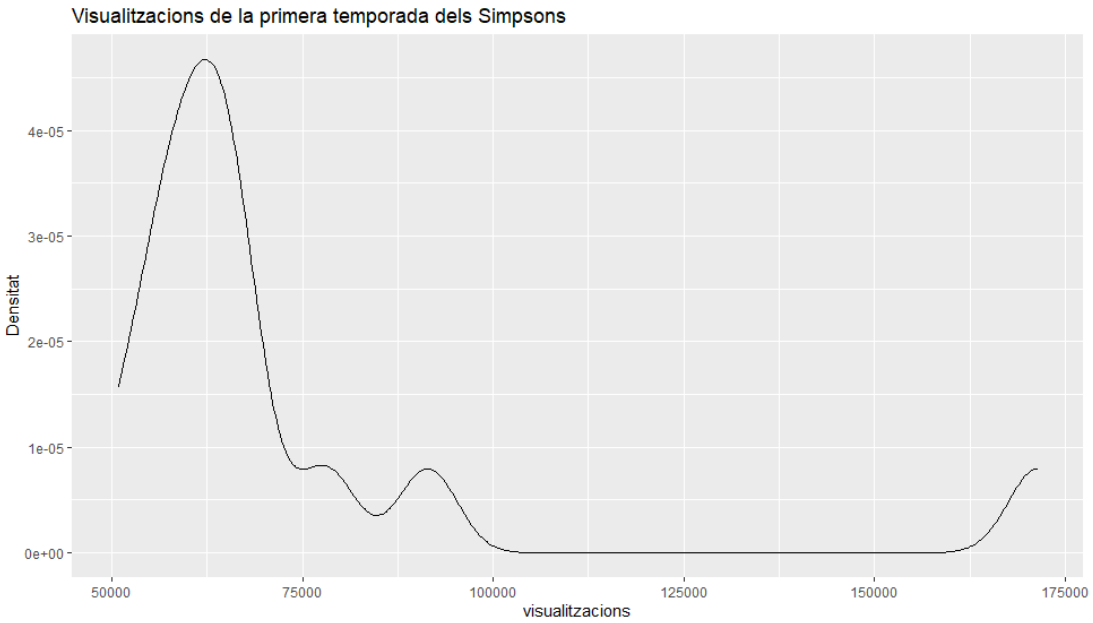
Primer necessitem filtrar la primera temporada. Sembla que no hi ha nans ni valors 'nulls' en aquesta (pel que fa a les visualitzacions), per tant podem prosseguir.

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:



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')
```

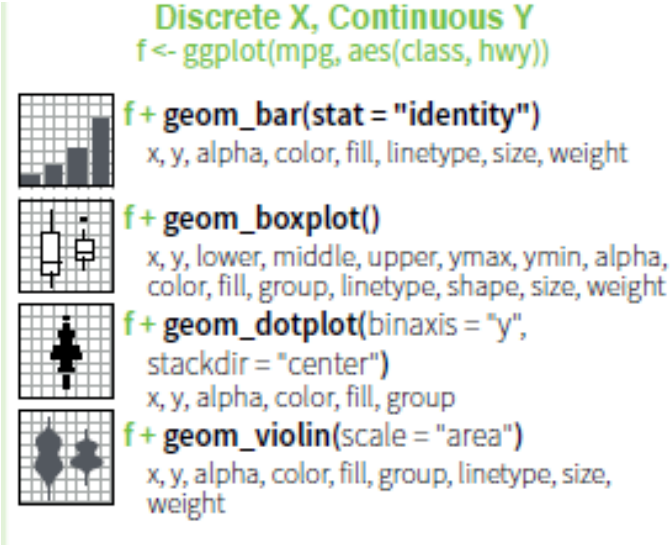


2.- Feu una gràfica que permeti veure la distribució de les 10 primeres temporades (‘*season*’) respecte al nombre de visualitzacions (‘*views*’). Expliqueu l'elecció del gràfic i les conclusions que podeu extreure'n.

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.

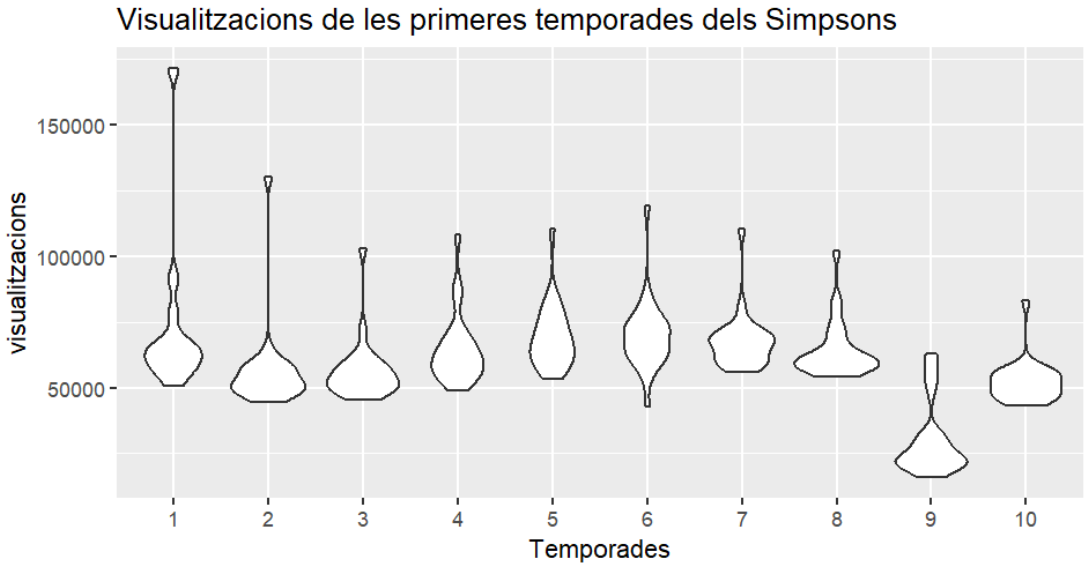
En quant al gràfic, tenim una variable numèrica continua “views” i una variable ‘season’ que podem fer discreta categòrica amb factor i se'ns demana una distribució. Tornem a mirar la diapositiva citada en l’exercici anterior

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:



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