Práctica 2. Visualización de Datos

Análisis Exploratorio de Datos, Máster en Ciencia de Datos - UV

Contents


```
library(pacman)
packages =  c("tidyverse", "knitr", "ggplot2", "datasets", "RColorBrewer",'latex2exp')
pacman::p_load(char=packages)
```

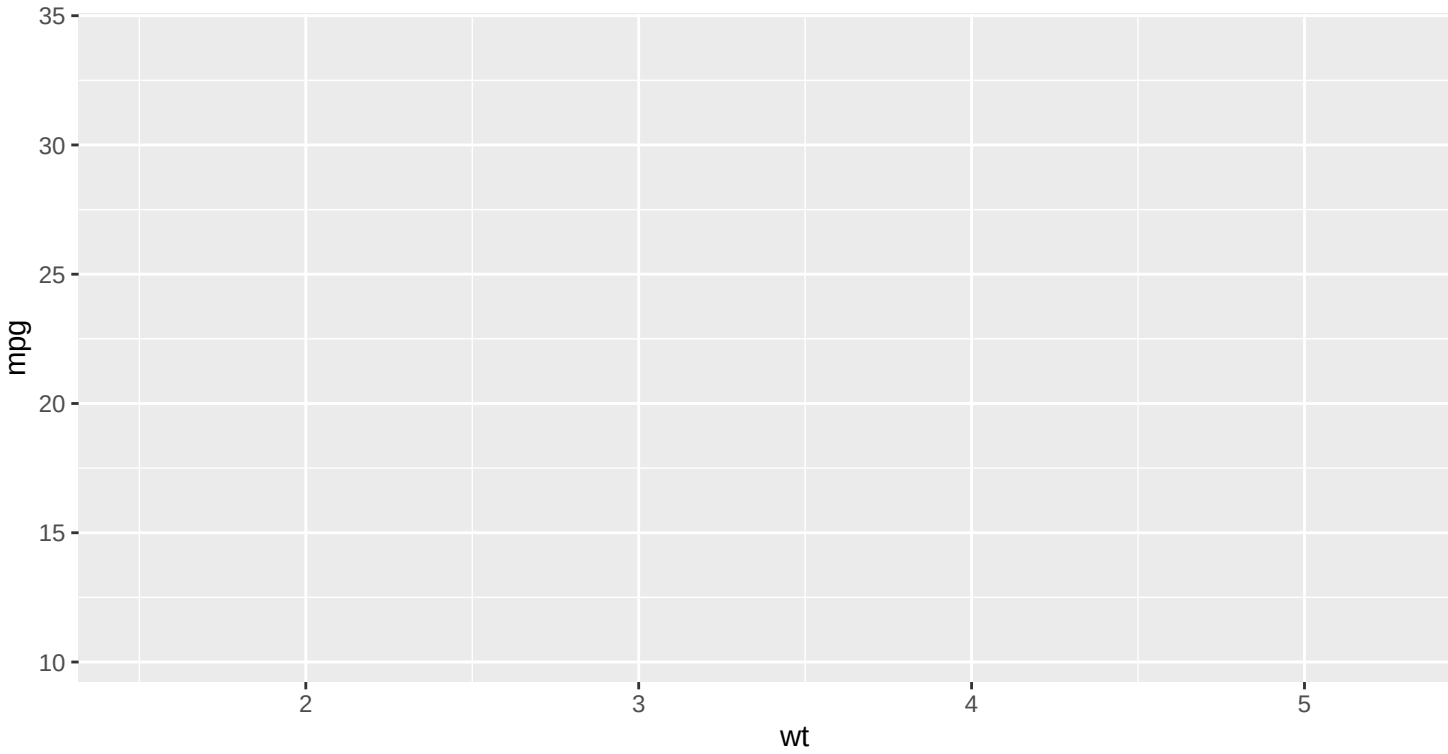
ggplot2. Una gramática para escribir gráficos.

R ofrece una serie de funciones para hacer gráficos en su distribución de base. Sin embargo, estas funciones no son las más utilizadas por los científicos de datos desde que Wickham y Chang, en 2016, implementaron la librería **ggplot2**. La librería **ggplot2** implementa un conjunto de funciones que podíamos denominar una "gramática" para escribir código y generar gráficos en R. Para construir un gráfico con **ggplot2** necesitamos un dataset, un conjunto de *aesthetics* (eje x, eje y, etc.), un elemento geométrico (puntos, líneas, barras, etc.), operaciones estadísticas y otros componentes.

Creando la primera capa de ggplot.

Como hemos dicho anteriormente **ggplot2** define una "gramática" para definir gráficos. En primer lugar tenemos que definir qué dataset vamos a utilizar mediante la función **ggplot()**, y un conjunto de *aesthetics* mediante la función **aes()**. En este ejemplo se va a definir el conjunto de datos que vamos a utilizar (**mtcars**) y vamos a definir un conjunto de aesthetics básico, donde indicaremos qué variable se va a representar en el eje x (wt) y qué variable en el eje y (mpg).

```
p<-ggplot(mtcars, aes(x = wt, y = mpg))
p
```

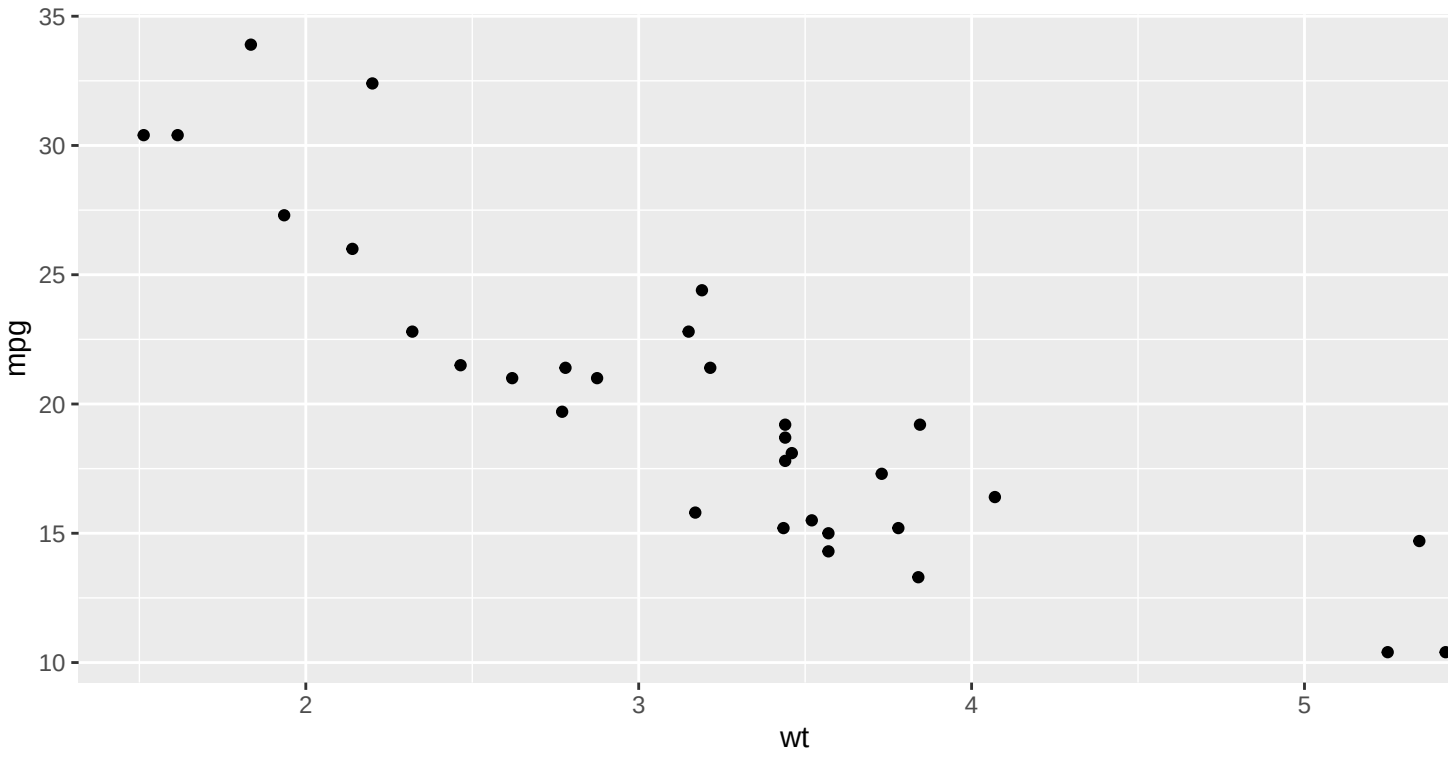


Como puedes observar, a priori no se ve ningún punto ni ninguna línea. Es normal, no hemos especificado el tipo de objeto geométrico que queremos representar.

Gráficos de dispersión con ggplot2

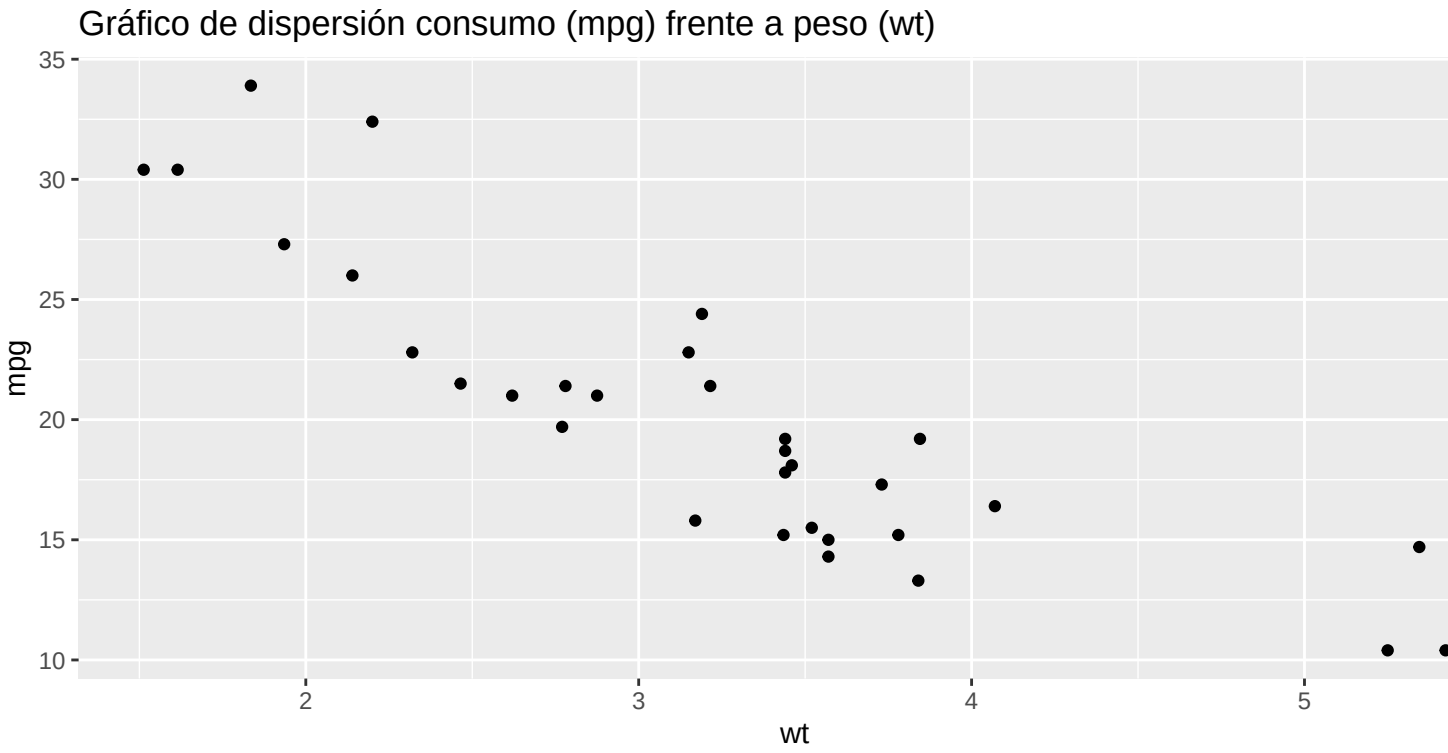
Si queremos representar un gráfico de dispersión tendremos que añadir una capa, a la capa que crea ggplot, indicando que queremos representar puntos. Esto lo haremos mediante la función **geom_point()**.

```
p1<-p+geom_point()
p1
```



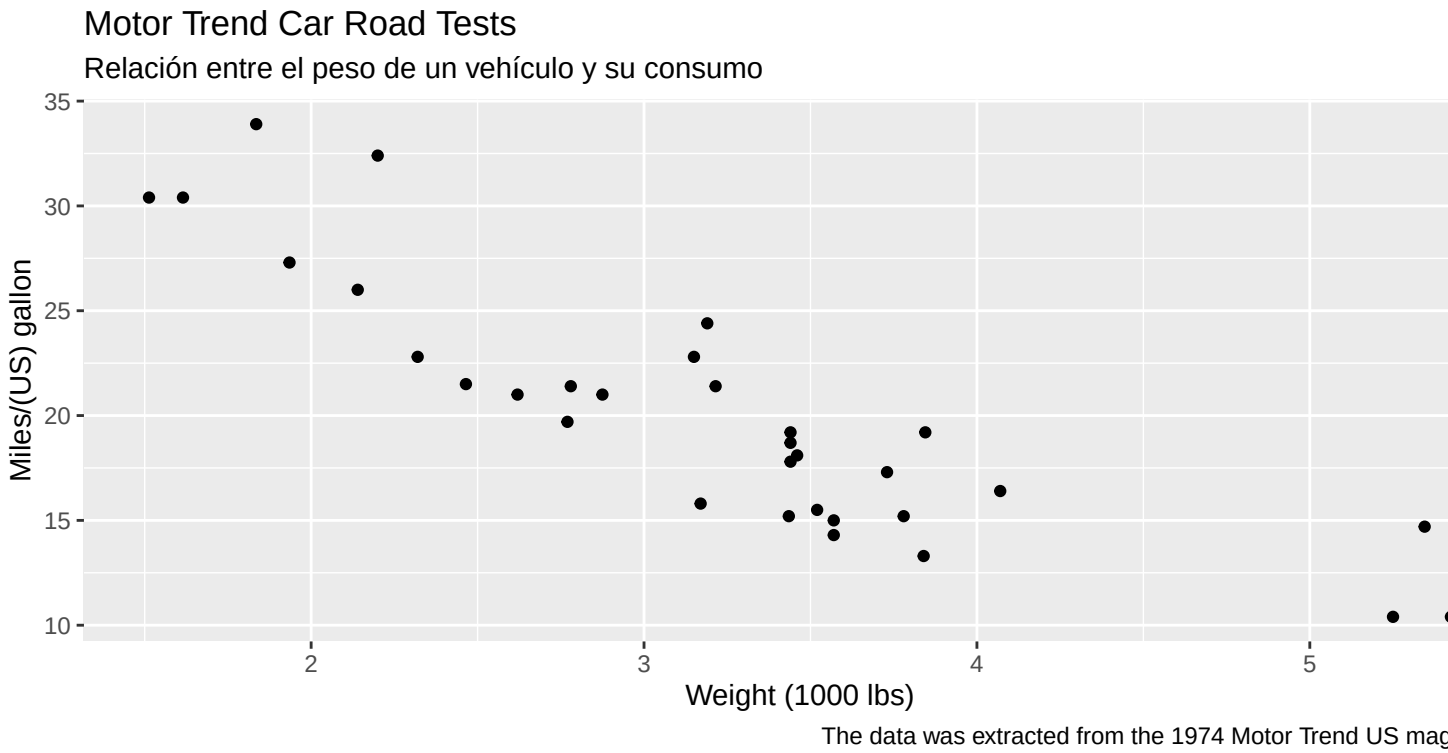
En un momento determinado puede ser interesante incluir un título en nuestro gráfico, para definir una capa de título lo haremos con la función **ggtitle()**.

```
p2<-p1+ggtitle("Gráfico de dispersión consumo (mpg) frente a peso (wt)")
p2
```



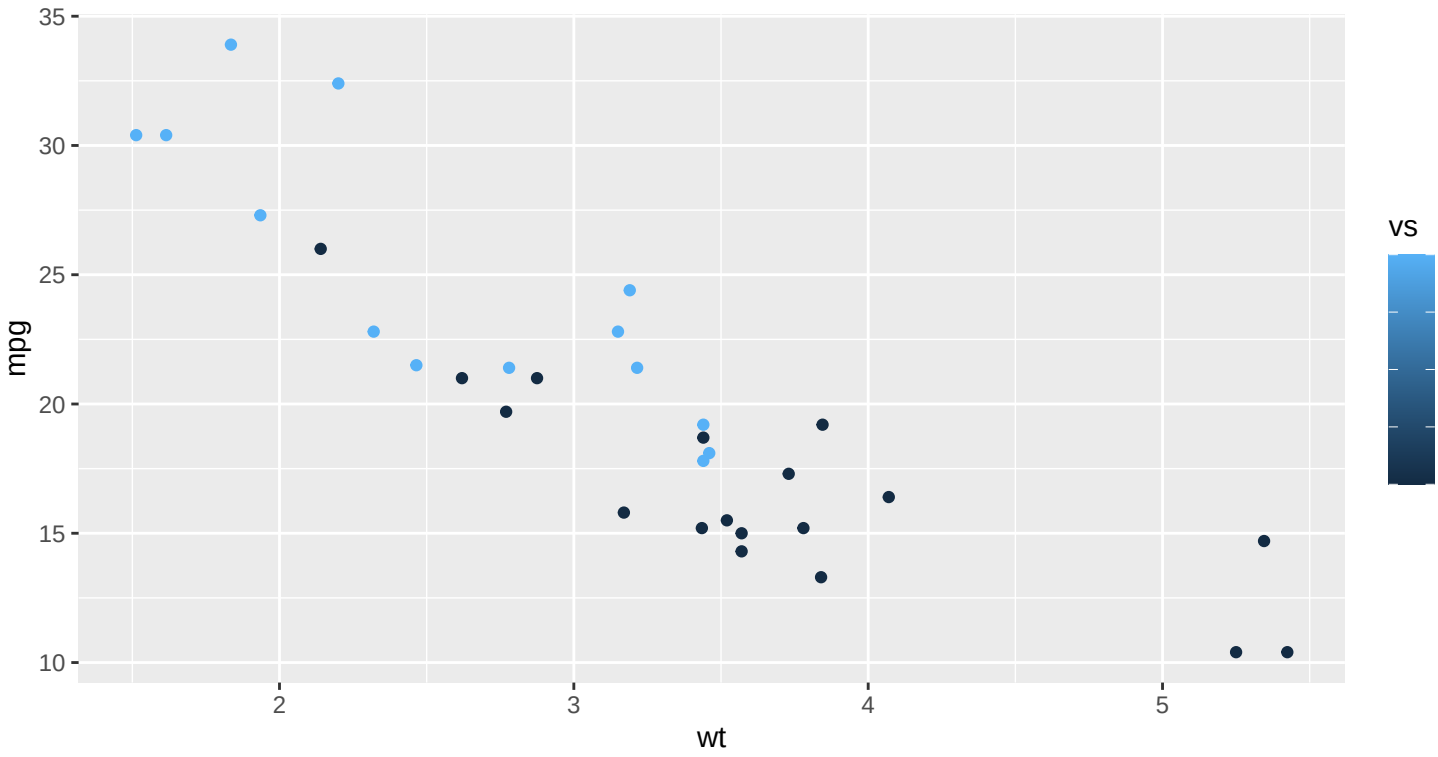
Podemos utilizar la función `labs()` para añadir al gráfico, título, subtítulo, leyendas en los ejes y pie de figura (`caption`), etc. Por defecto los ejes se etiquetan con los nombres de las variables representadas.

```
p<-ggplot(data=mtcars,
          mapping=aes(x = wt,
                      y = mpg))+
  geom_point()+
  labs(title = "Motor Trend Car Road Tests",
       subtitle = "Relación entre el peso de un vehículo y su consumo",
       x = "Weight (1000 lbs)",
       y = "Miles/(US) gallon",
       caption = "The data was extracted from the 1974 Motor Trend US magazine.")
p
```



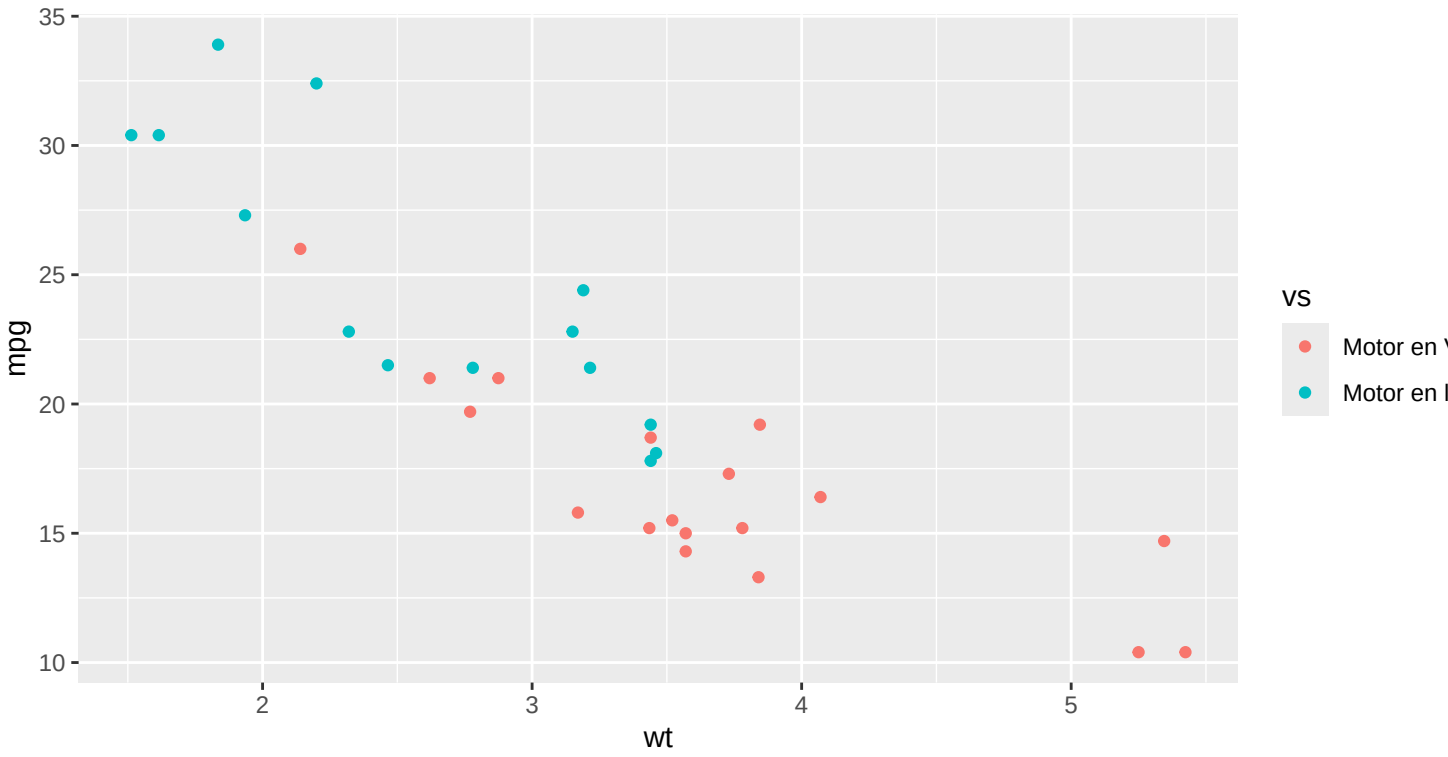
Podemos incluir información de una variable categórica de muchas maneras en un gráfico de dispersión, color, tamaño o forma del punto. Por ejemplo imaginemos que queremos hacer la representación anterior pero haciendo que los puntos tengan diferentes colores dependiendo de la variable **vs**.

```
p<-ggplot(mtcars, aes(x = wt, y = mpg, colour=vs))+geom_point()
p
```



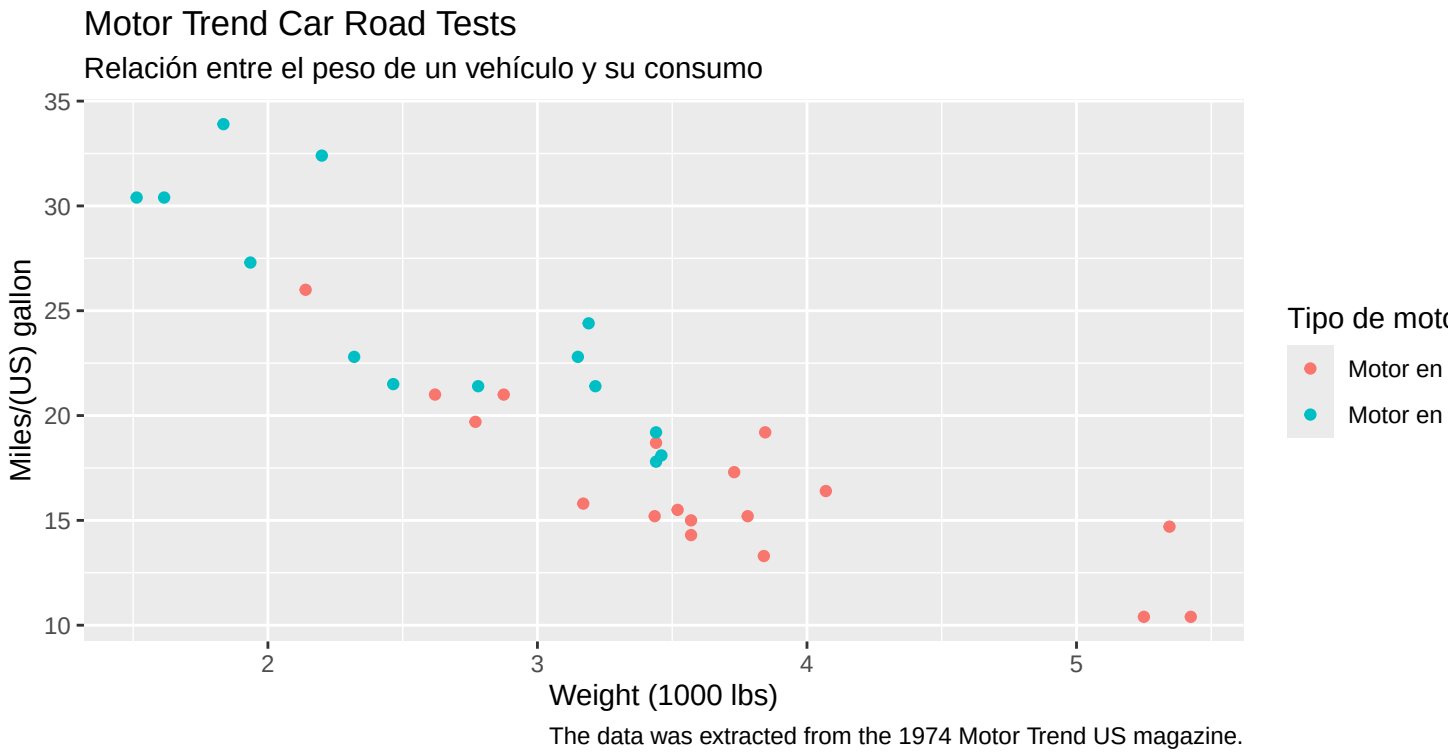
Como puedes observar ggplot ha tratado a la variable **vs** como una variable continua ya que es de tipo numérico (a pesar de que sólo toma valores 0 y 1), si la convertimos a factor, la cosa será diferente.

```
mtcars1<-mtcars
mtcars1$vs<-factor(mtcars1$vs,levels=c(0,1),labels=c("Motor en V", "Motor en línea"))
p<-ggplot(mtcars1, aes(x = wt, y = mpg, colour=vs))+geom_point()
p
```



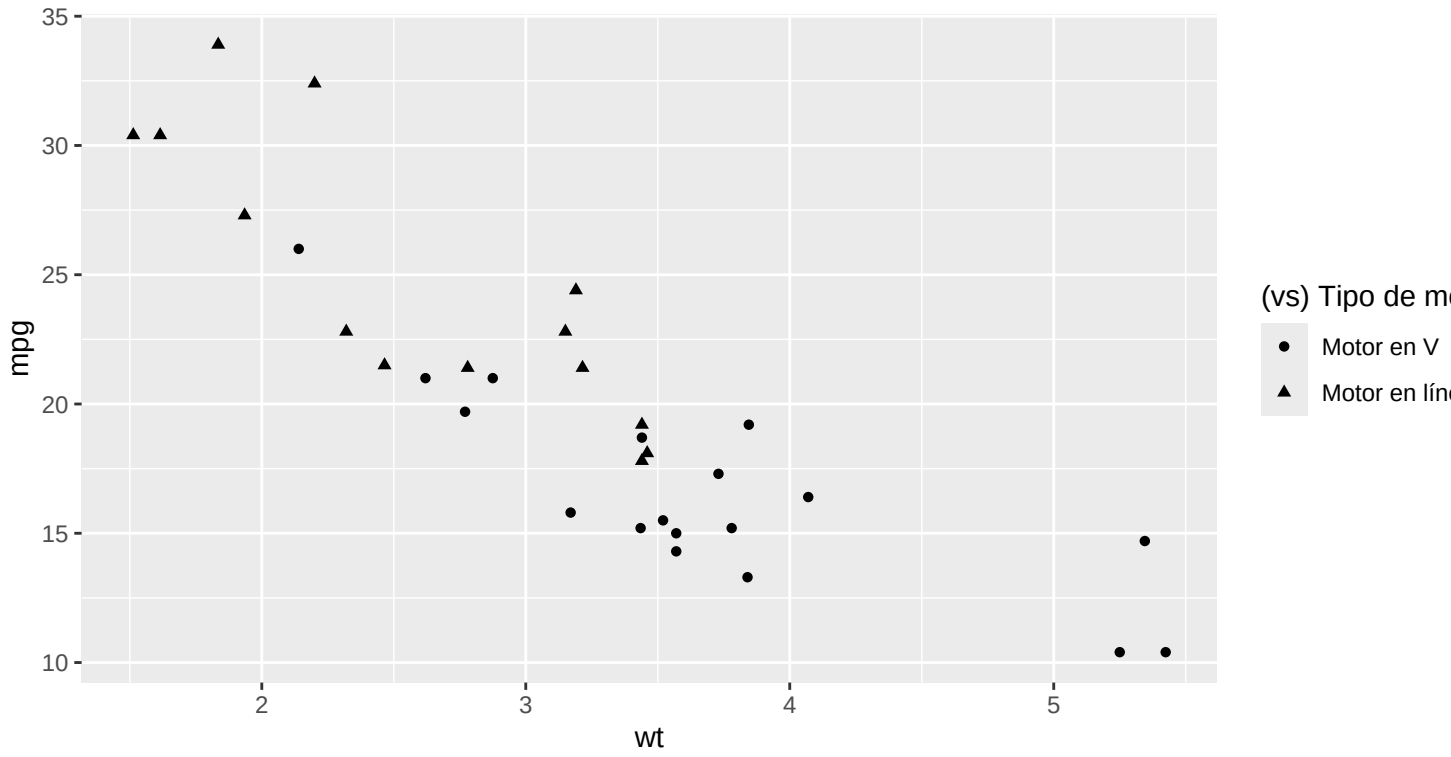
Además de personalizar ejes y título también podemos cambiar la leyenda de la variable que modifica, en este caso el tipo de motor, asociado al estético `colour`

```
p+
  labs(title = "Motor Trend Car Road Tests",
       subtitle = "Relación entre el peso de un vehículo y su consumo",
       x = "Weight (1000 lbs)",
       y = "Miles/(US) gallon",
       caption = "The data was extracted from the 1974 Motor Trend US magazine.",
       colour='Tipo de motor')
```



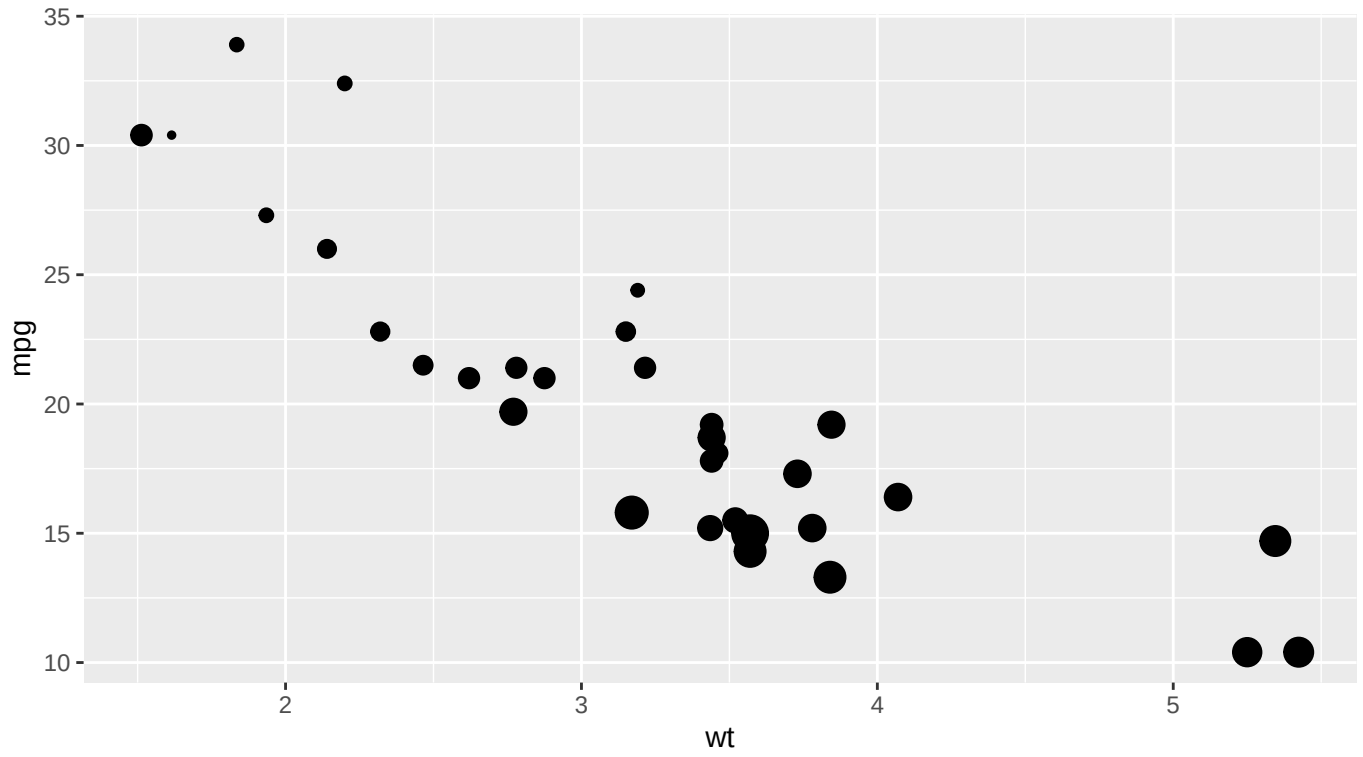
Al igual que hemos incluido información de una variable categórica con el color del punto en el gráfico, podemos hacer algo similar con la forma.

```
p<-ggplot(mtcars1, aes(x = wt, y = mpg, shape=vs))+geom_point()+
  labs(shape='(vs) Tipo de motor')
p
```

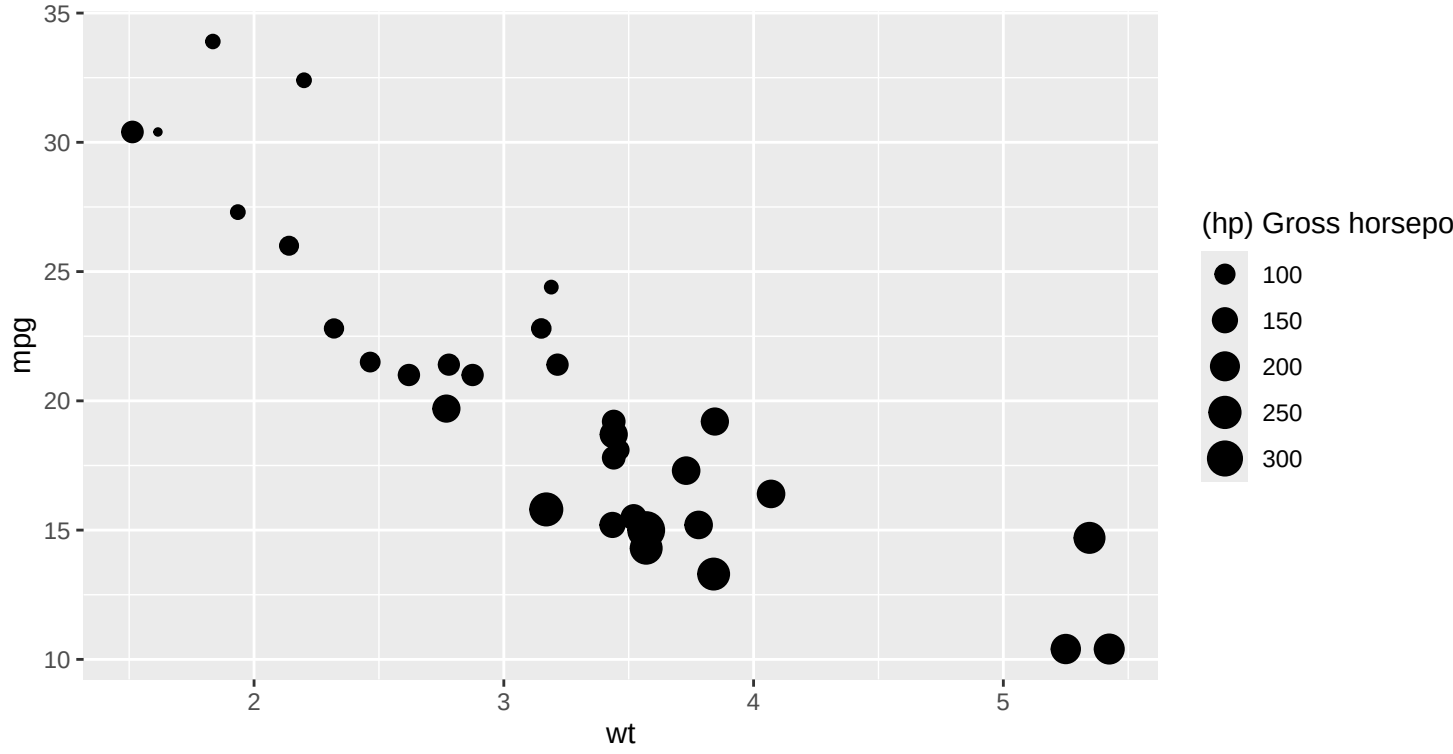


También podemos incluir información de una tercera variable, pero en este caso numérica, a con el tamaño, o la transparencia de los puntos.

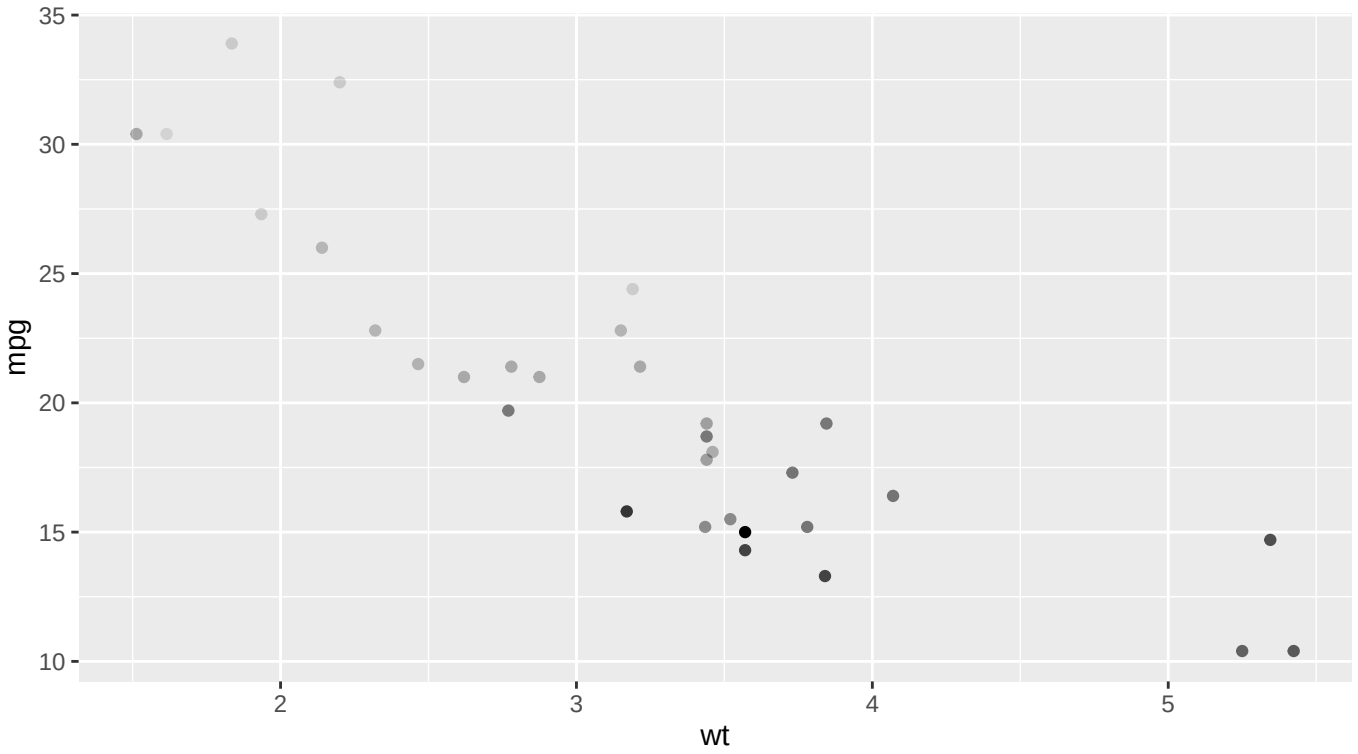
```
p<-ggplot(mtcars, aes(x = wt, y = mpg, size=hp))+geom_point()
p
```



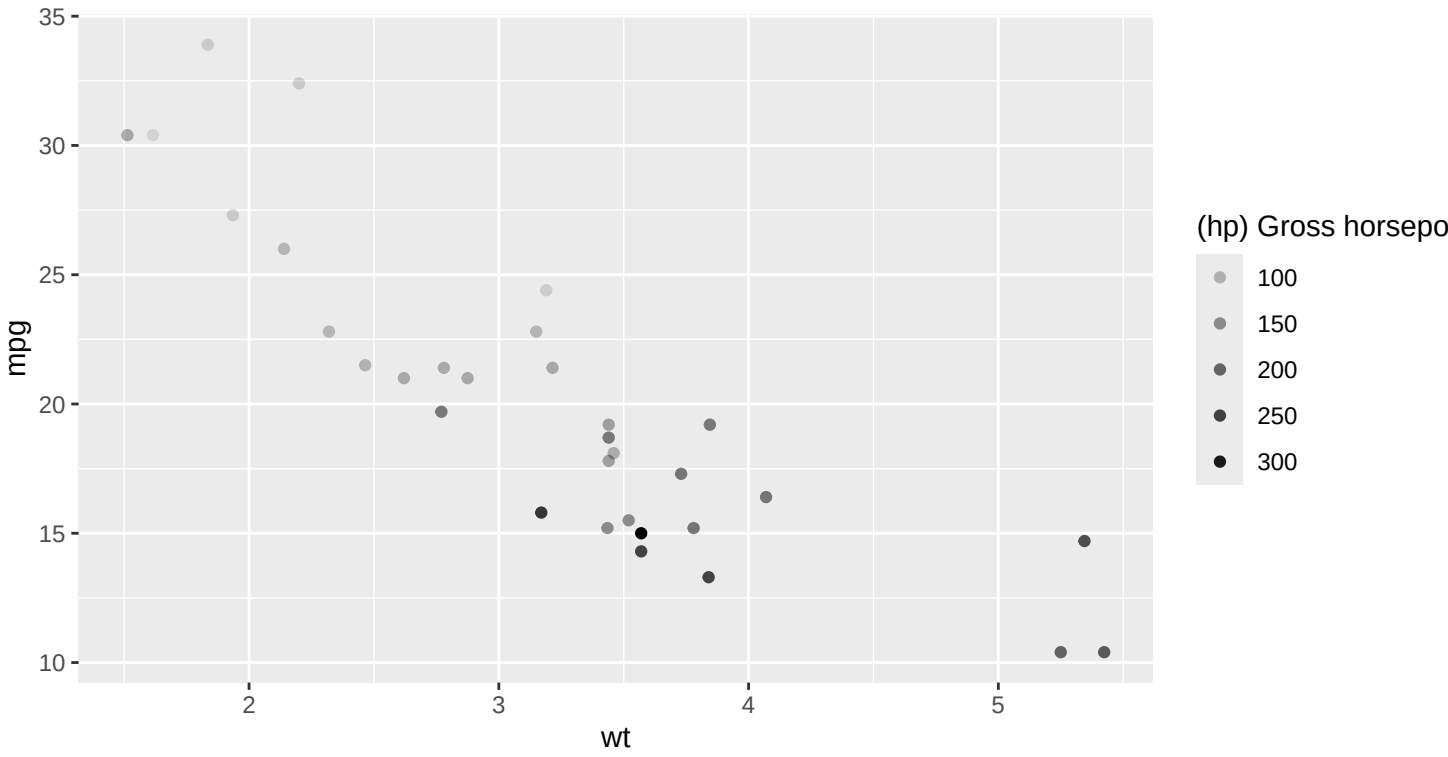

```
p+labs(size='(hp) Gross horsepower')
```



```
p<-ggplot(mtcars, aes(x = wt, y = mpg, alpha=hp))+geom_point()
p
```

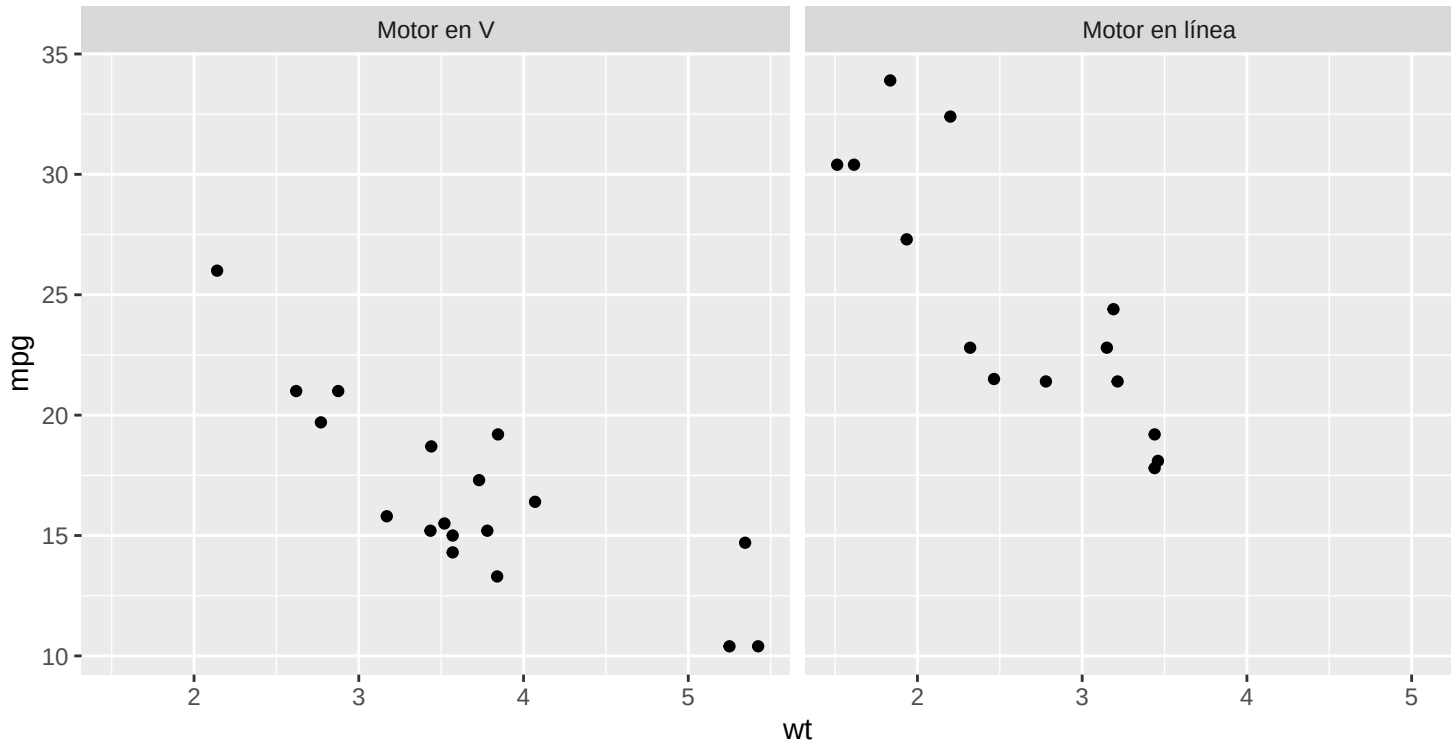



```
p+labs(alpha='(hp) Gross horsepower')
```

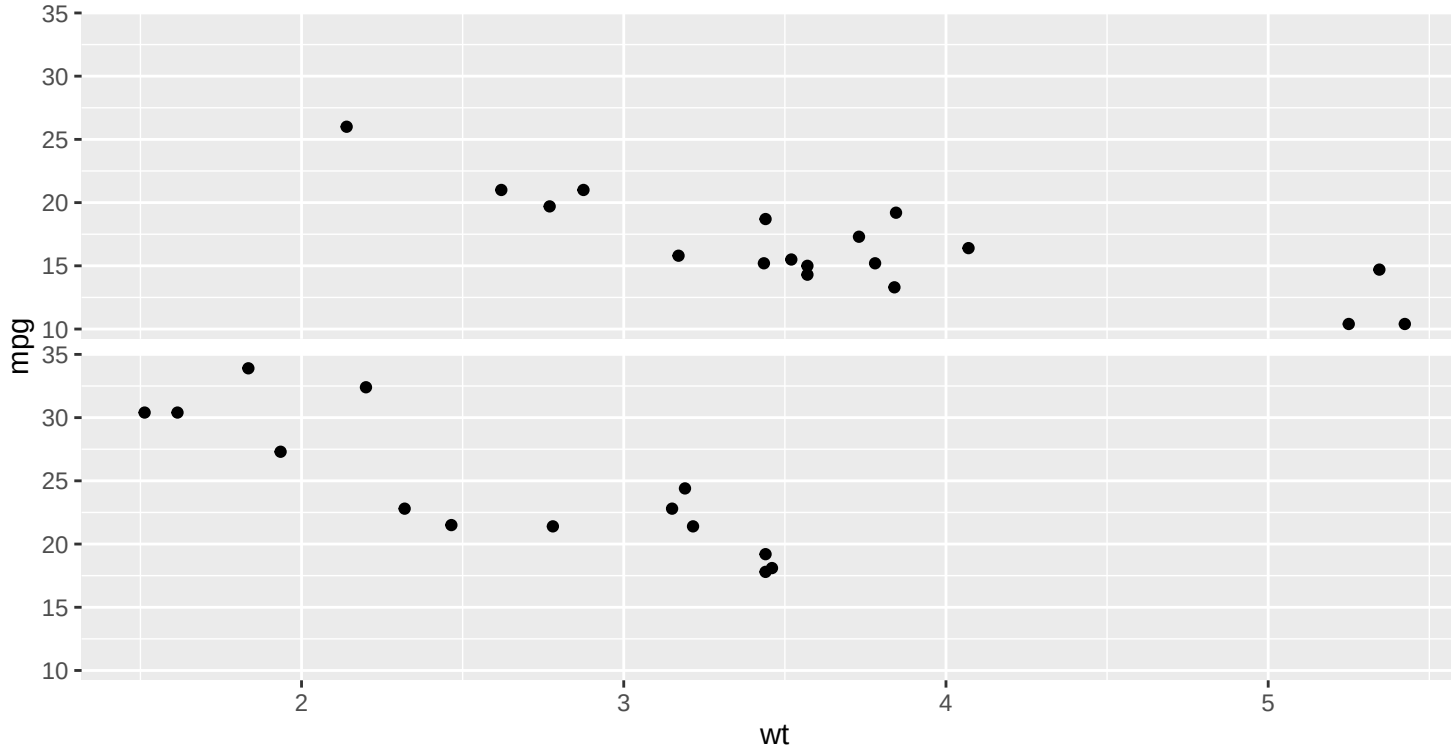


En ocasiones es interesante obtener diferentes gráficos de dispersión separados por el valor de una variable. Veamos un ejemplo en el que haremos dos gráficos de dispersión de las variables del ejemplo anterior separados por la variable `vs`. Para ello utilizaremos la función **facet_grid()**.

```
p1<-ggplot(mtcars1, aes(x = wt, y = mpg))+geom_point() + facet_grid(.~vs )

p1
```

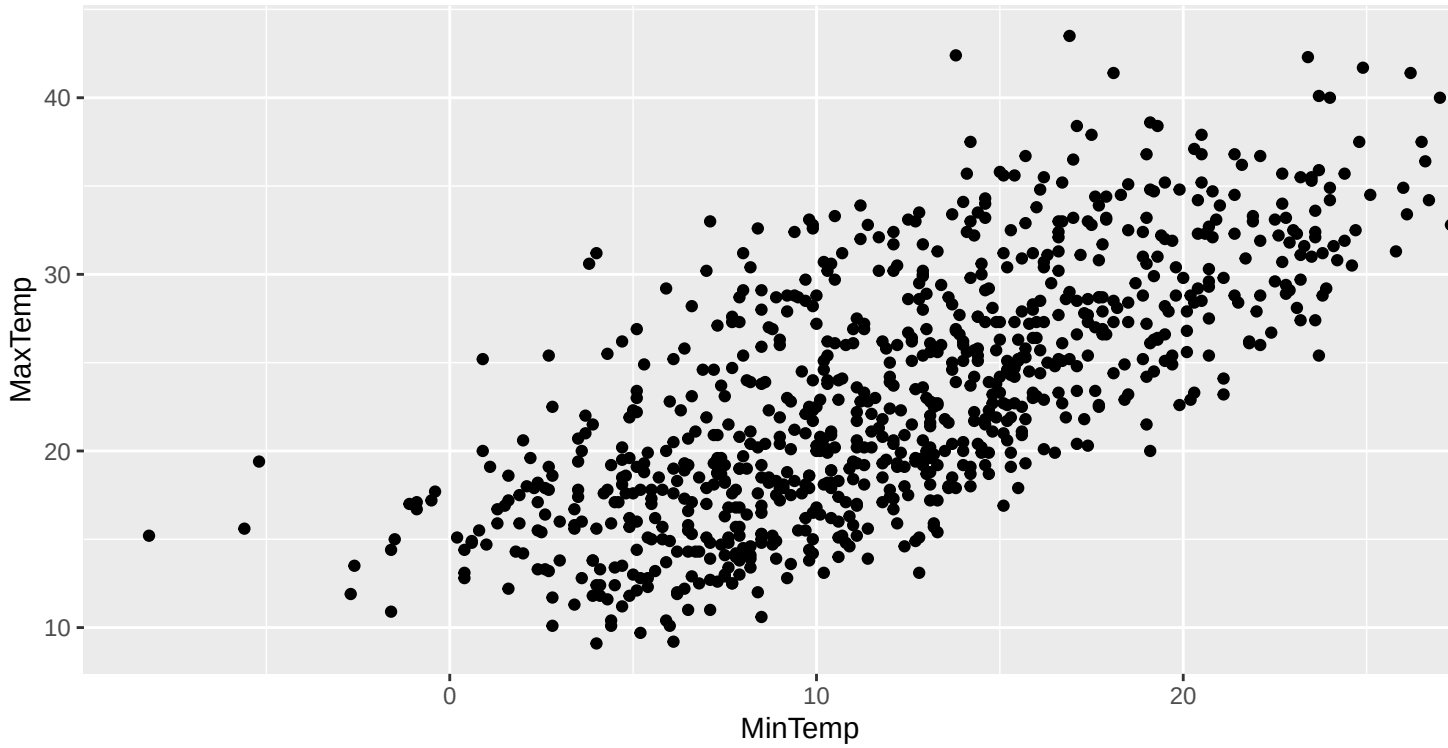


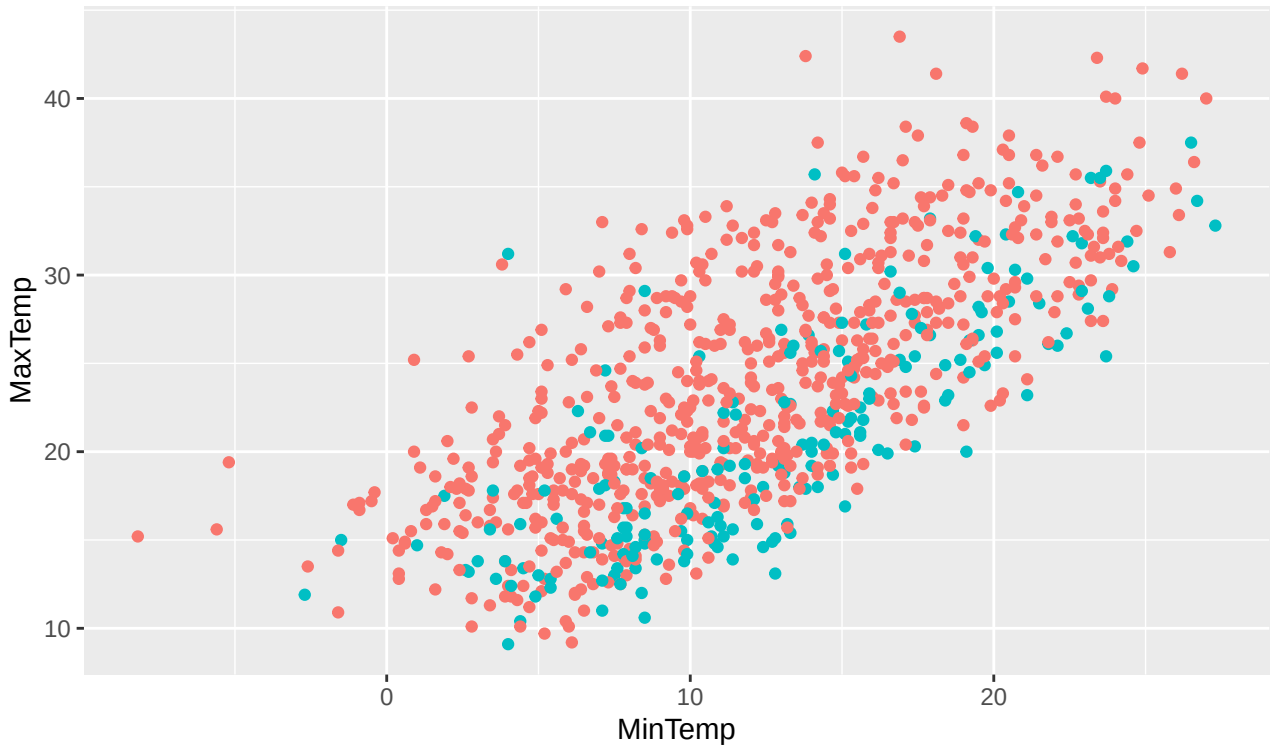

```
p2<-ggplot(mtcars1, aes(x = wt, y = mpg))+geom_point() + facet_grid(vs~. )
p2
```

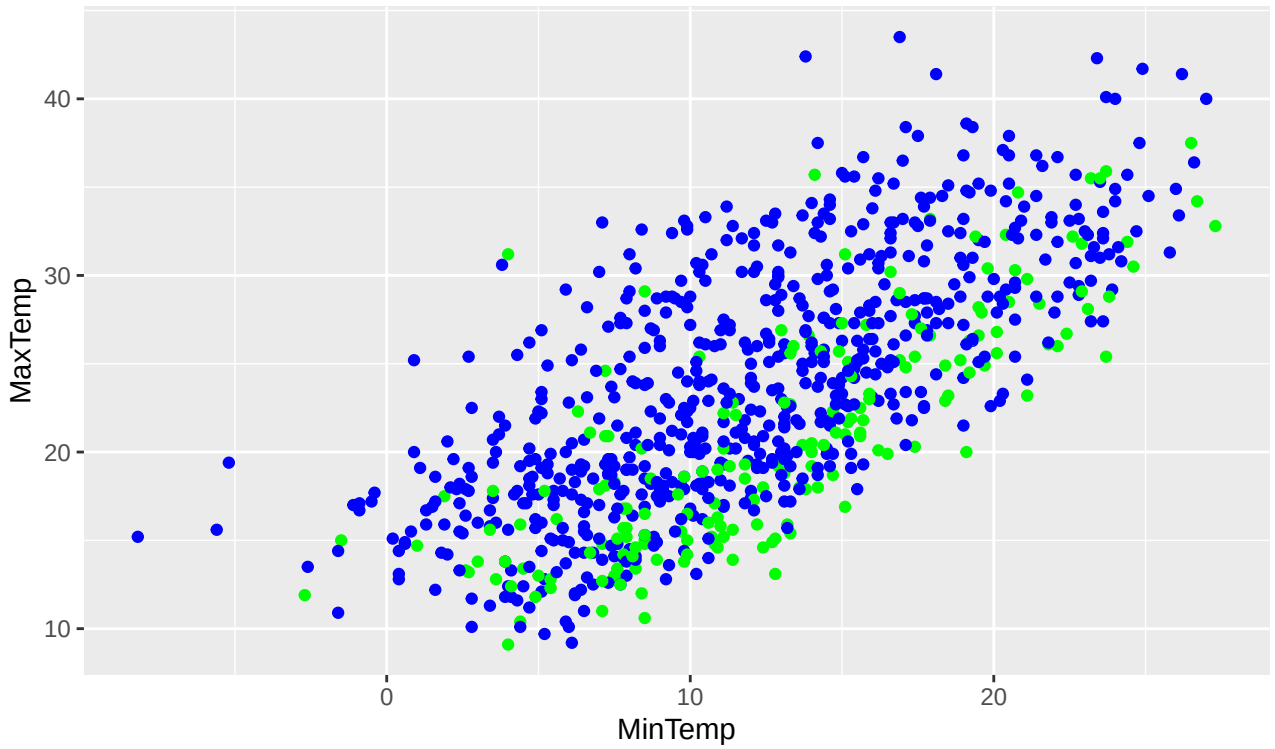


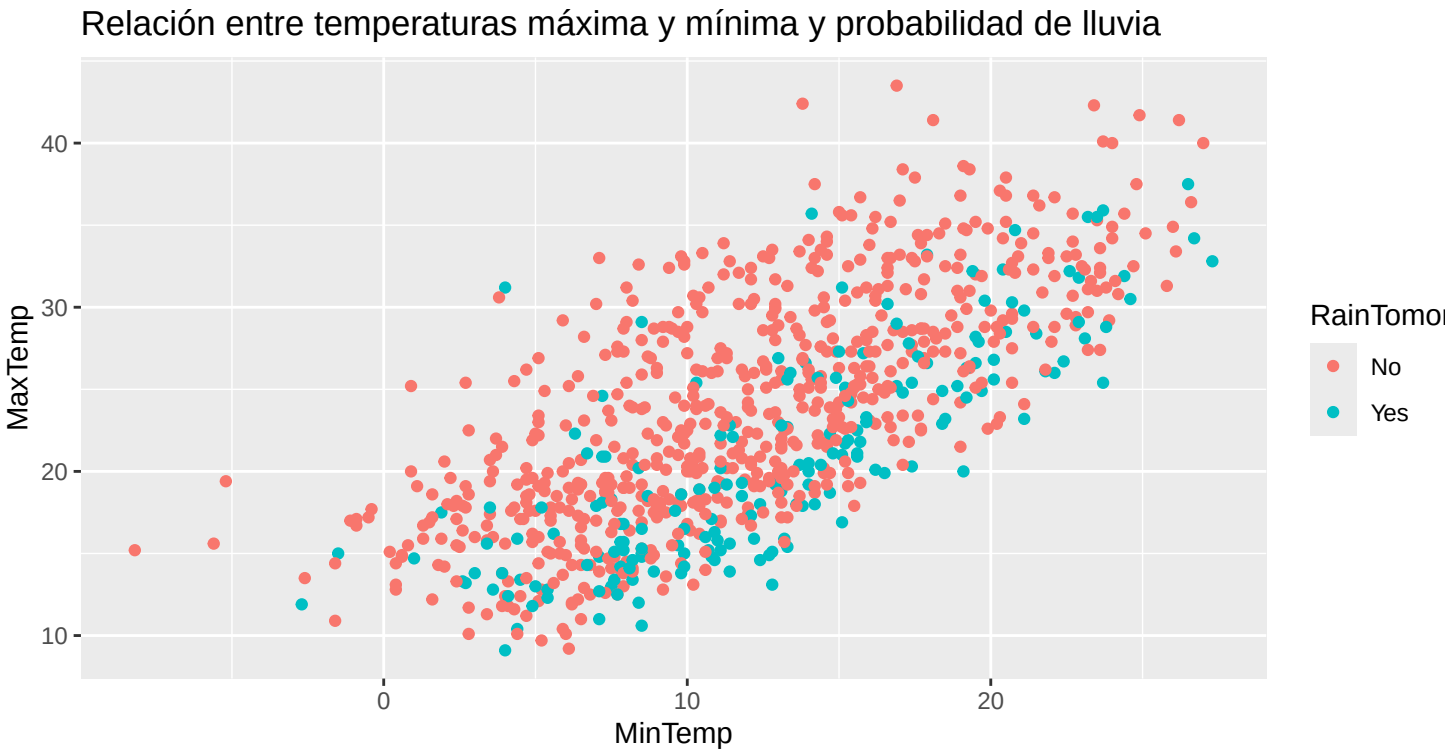
Ejercicio 1

- Carga el fichero `weatherAUS.RData` utilizando lo aprendido en la práctica 1.
- Crea un objeto ggplot denominado `grafico1` en el que esté definido el conjunto de datos `ds` alojado en el fichero `weatherAUS.RData` y un *aesthetics* en el que en el eje x esté definida la variable `MinTemp` y en el eje y esté definida la variable `MaxTemp`.
- Añade una capa que defina el objeto geométrico necesario para construir un gráfico de dispersión de las variables `MinTemp` y `MaxTemp`.
- Genera ahora el mismo gráfico y almacénalo en `grafico2` pero define el *aesthetics* para visualizar las variables `MinTemp` y `MaxTemp` en los ejes de la gráfica y la variable categórica `RainTomorrow` como el color. A partir del objeto `grafico2`, representa un gráfico de dispersión en el que se muestren los puntos con el color del punto correspondiente a la variable `RainTomorrow`.
- Añade a `grafico2` una capa para incluir un título con **ggtitle()** y almacénalo en `grafico3`.
- Almacena en formato **pdf** el gráfico resultante en un fichero denominado `Ejercicio1.pdf`.



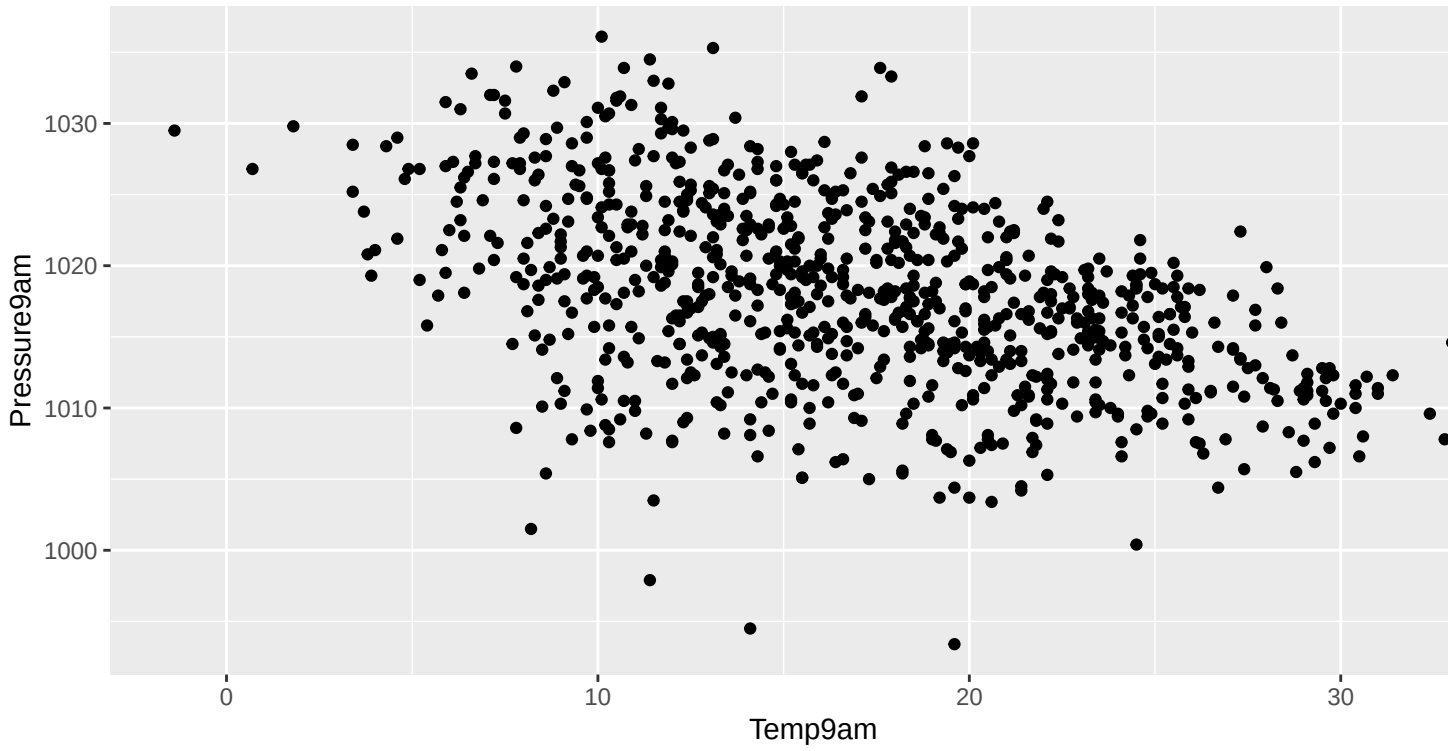

MaxTemp
40
30
20
10
0
10
20
MinTemp
RainTomor
No
Yes

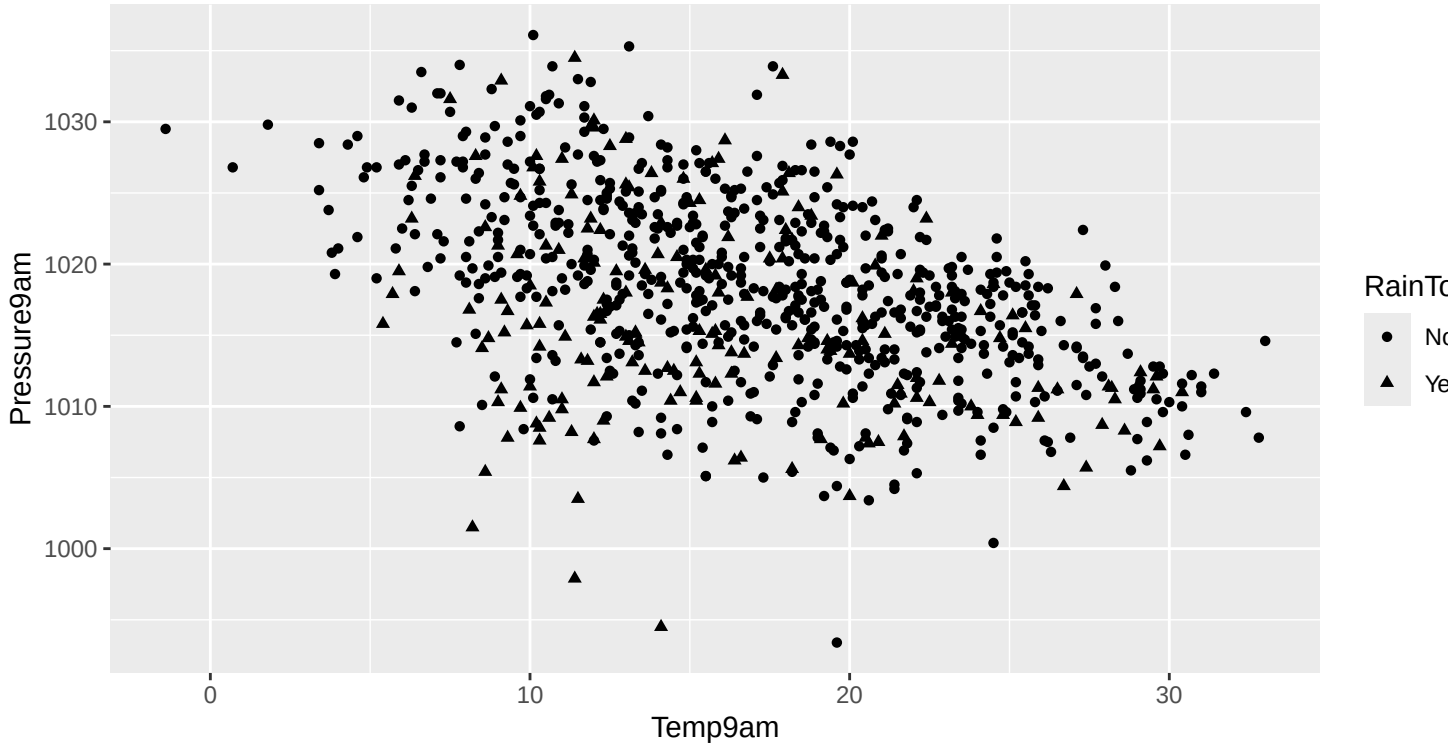

MaxTemp
40
30
20
10
0
10
20
MinTemp
RainTomor
No
Yes

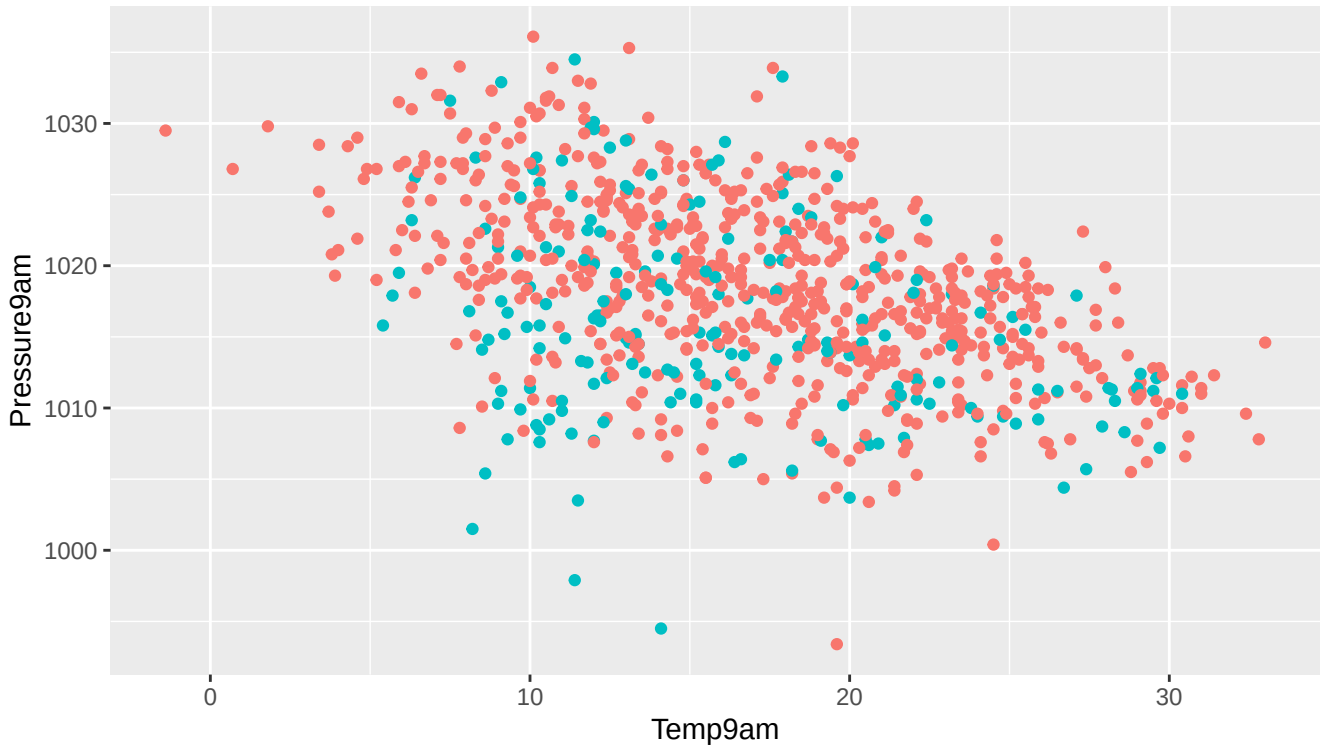


Ejercicio 2

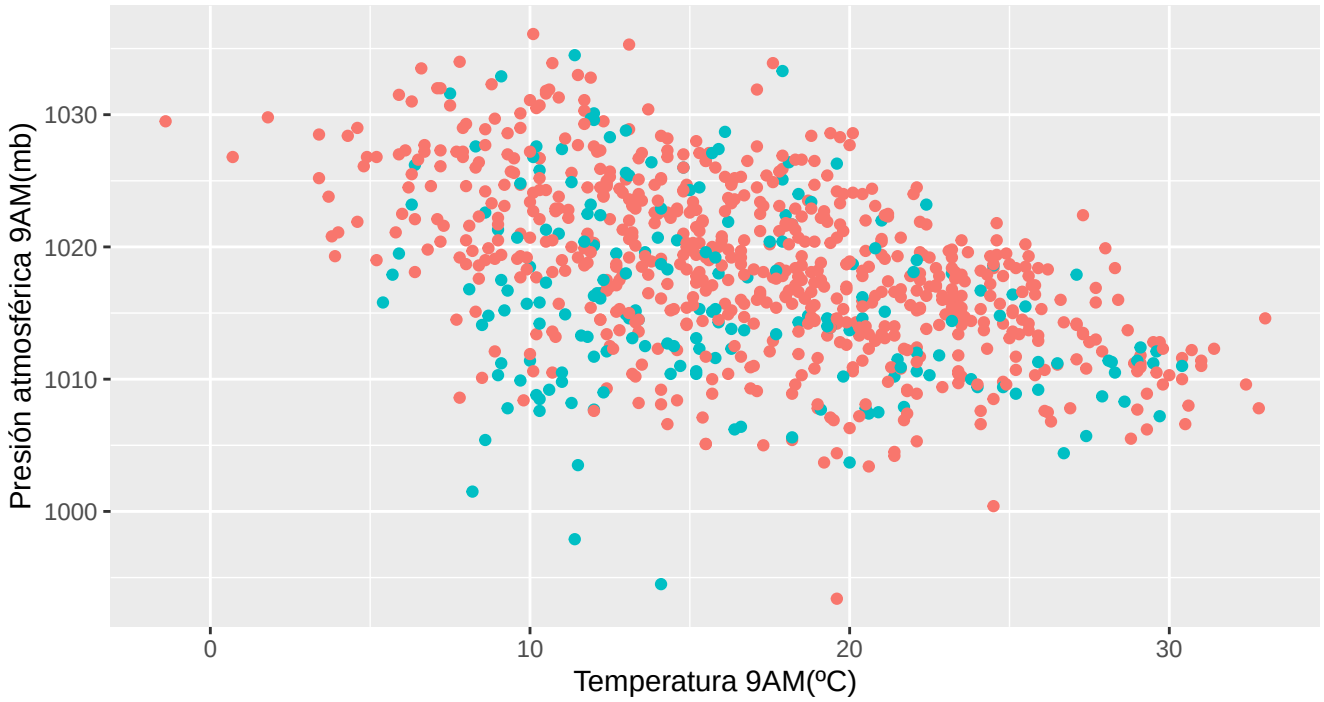
- Utilizando ggplot haz un grafico de dispersión en el que representes las variables `Temp9am` y `Pressure9am` en los ejes x e y respectivamente.
- Modifica el gráfico anterior para incluir información categórica proporcionada por la variable `RainToday` con la forma del punto.
- Modifica el gráfico anterior para incluir información categórica proporcionada por la variable `RainToday` con el color del punto. ¿Con qué representación (color o forma) se observa mejor la información que introduce la variable `RainToday`?
- Añade títulos explicativos al gráfico anterior (debe aparece un título, subtítulo, leyendas e información de las variables representadas en los ejes).
- Almacena en formato **pdf** el gráfico resultante en un fichero denominado `Ejercicio2.pdf`.


Pressure9am
1030
1020
1010
1000
0
10
20
30
Temp9am


Pressure9am
1030
1020
1010
1000
0
10
20
30
Temp9am
RainTo
No
Ye



Probabilidad de lluvia según condiciones meteorológicas

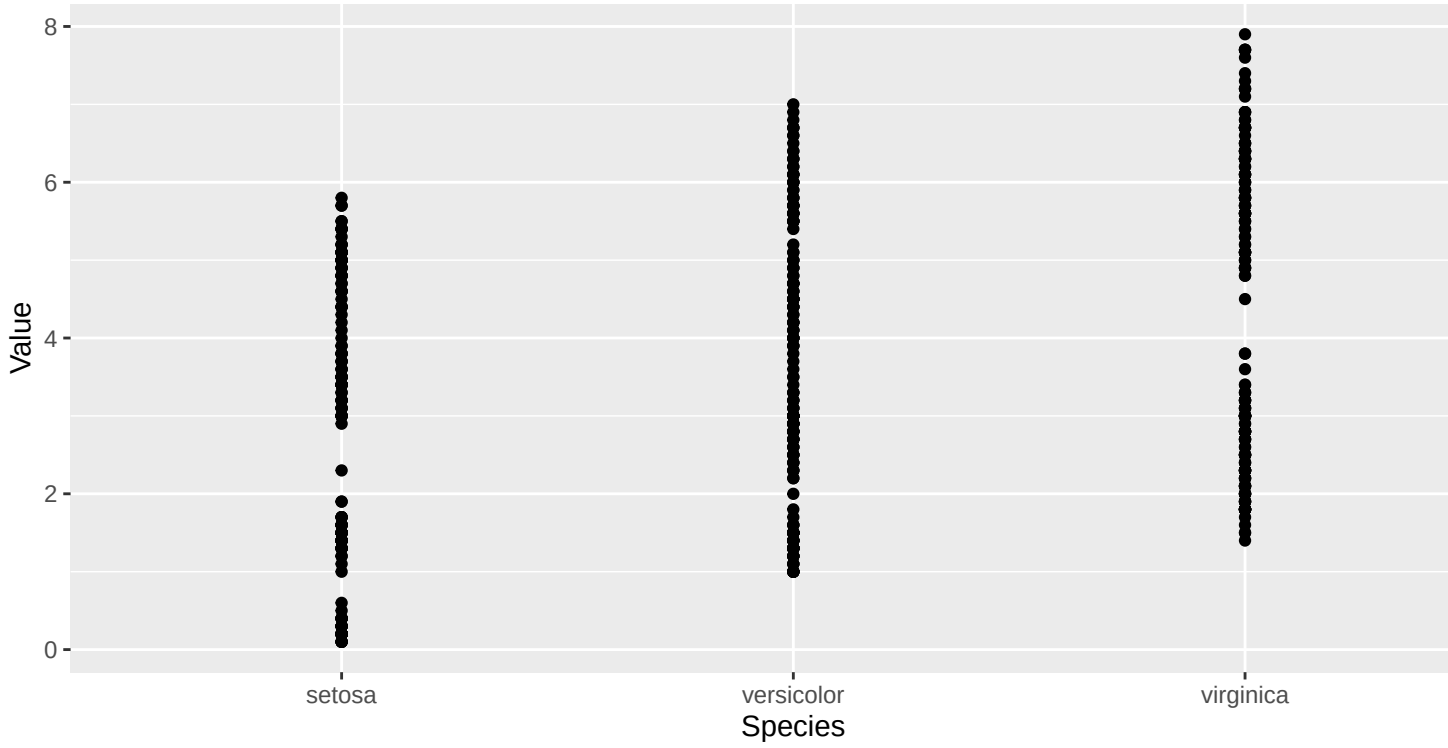


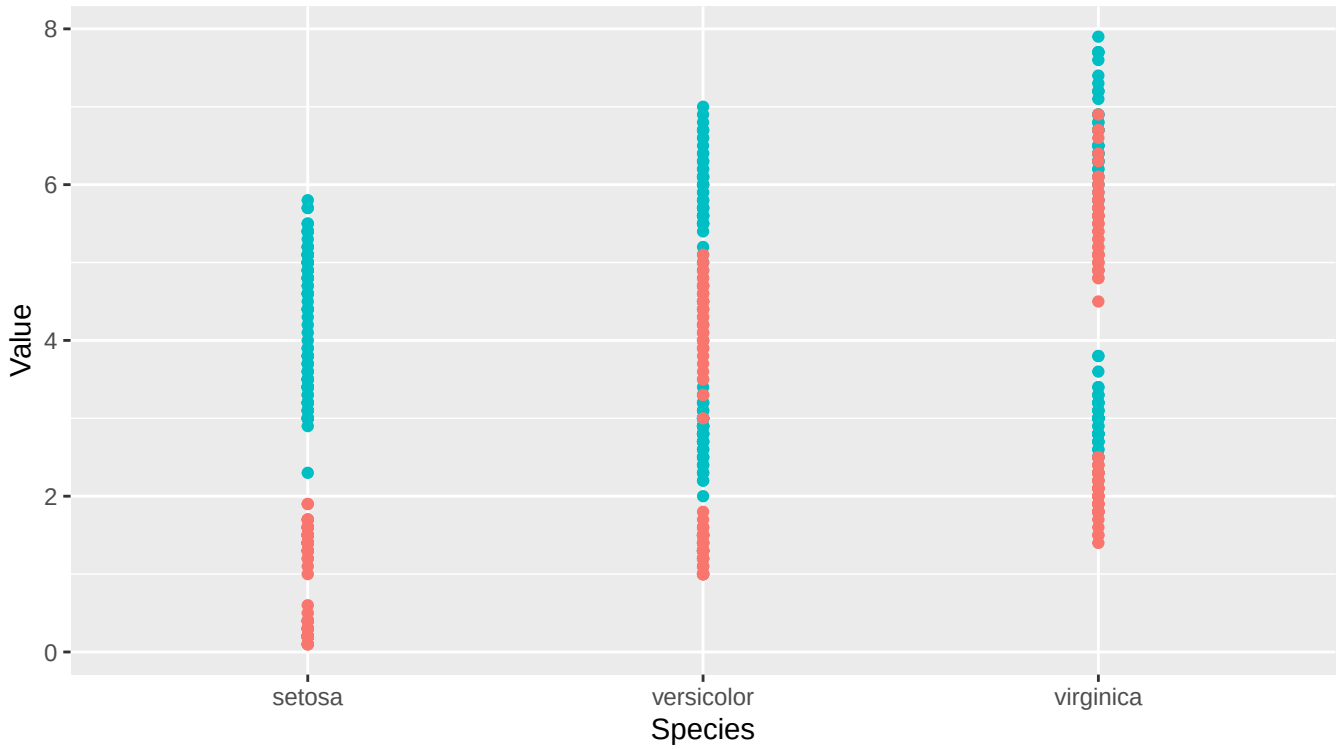
Ejercicio 3

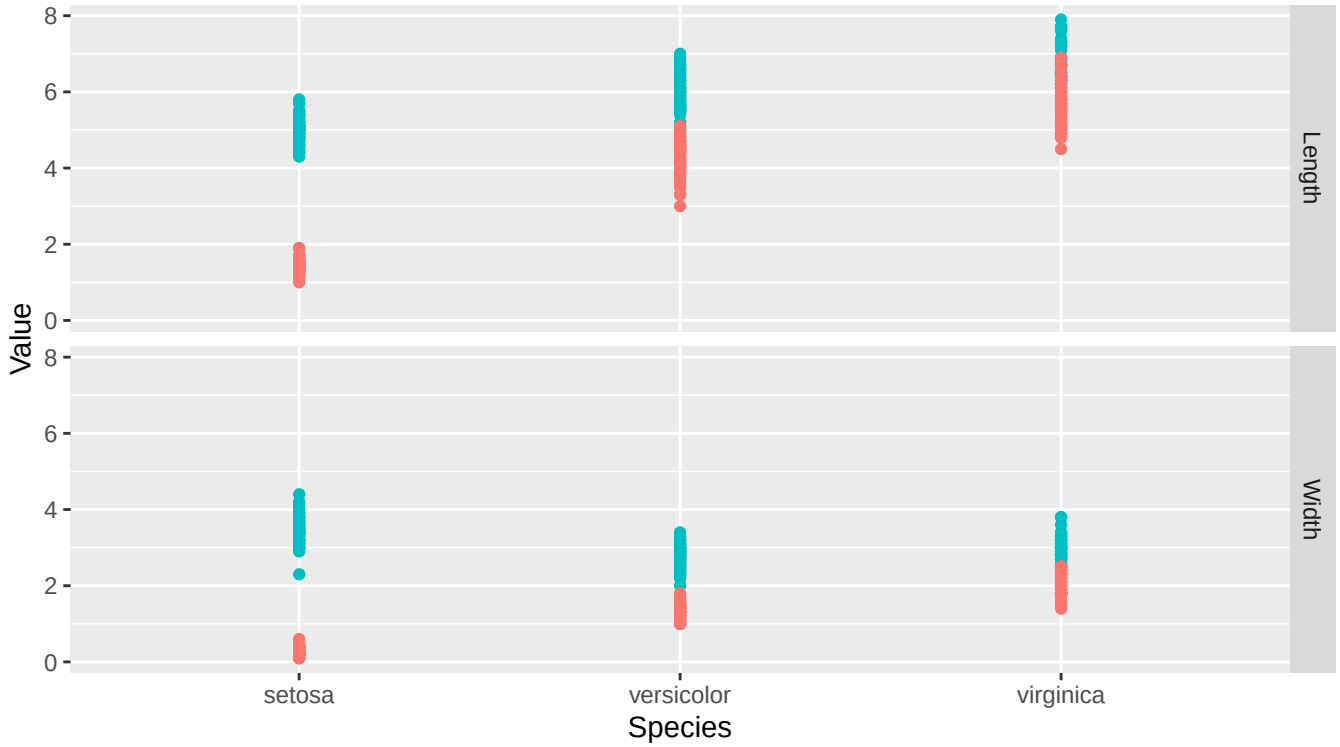
- Carga el fichero `iris.tidy.Rdata` utilizando lo aprendido en la práctica 1.

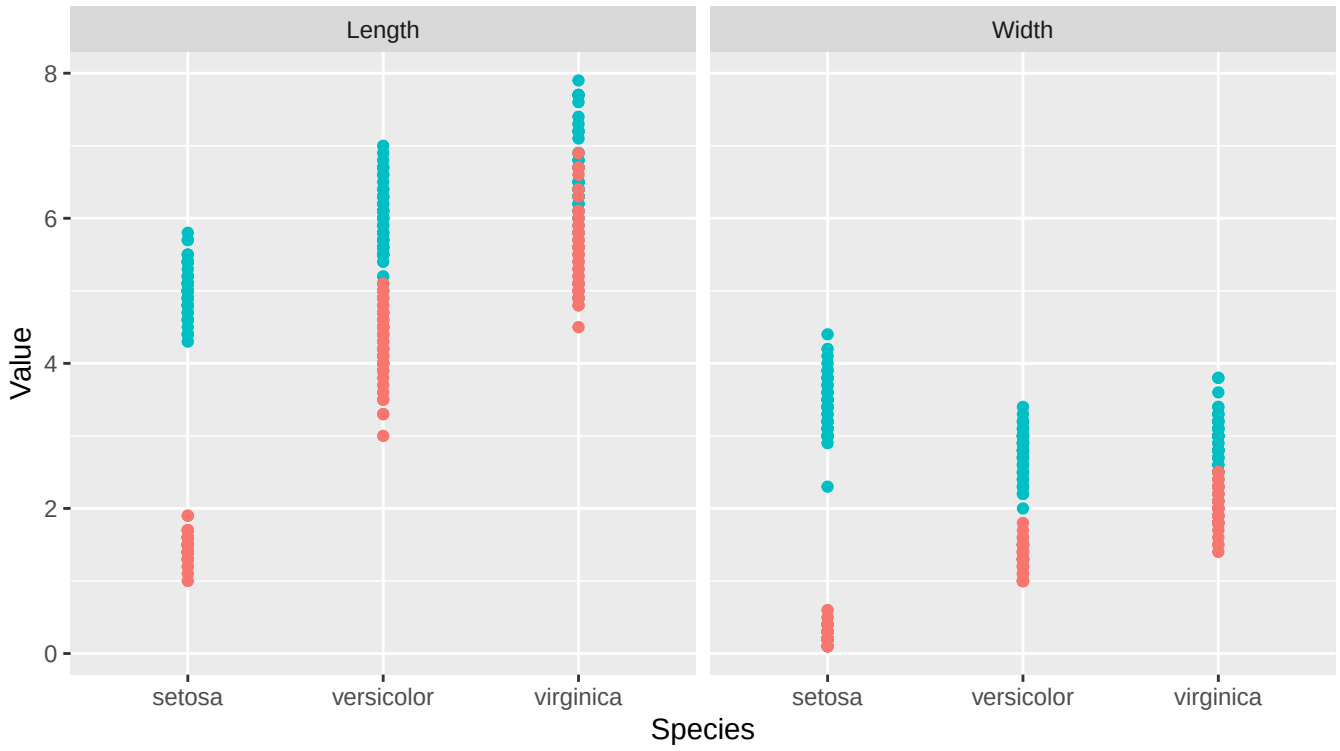
- Utiliza la función **str()** para ver qué variables contiene el dataset.
- Crea un objeto ggplot denominado `iris.grafico1` en el que esté definido el conjunto de datos `iris.tidy` alojado en el fichero `iris.tidy.RData` y un *aesthetics* en el que en el eje x esté definida la variable `Species` y en el eje y esté definida la variable `Value`.
- Añade una capa que defina el objeto geométrico necesario para construir un gráfico de dispersión de las variables `Species` y `Value`.
- Modifica el *aesthetics* de `iris.grafico1` para visualizar la variable categórica `Part` y almacénalo en `iris.grafico2`. A partir del objeto `iris.grafico2`, representa un gráfico de dispersión en el que se muestren los puntos dados por las variables `Species` y `Value`, y el color del punto se corresponda con el valor de la variable `Part`.
- Incluye información de la variable `Measure` haciendo dos gráficos de dispersión (dispuestos en filas) con **facet_grid()**, como en el apartado de introducción, para los dos valores que toma la variable `Measure`. Prueba a cambiar la disposición de los gráficos para que sea en columnas ¿Qué disposición, en filas o en columnas, permite comparar mejor los valores de las dos variables?
- Cambia ahora la capa **geom_point()** del gráfico anterior por la capa **geom_jitter()**. ¿Qué diferencia observas?
- Almacena en formato **pdf** el gráfico resultante en un fichero denominado `Ejercicio3.pdf`.

```
'data.frame':   600 obs. of  4 variables:
 $ Species: Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ Part   : Factor w/ 2 levels "Petal","Sepal": 2 2 2 2 2 2 2 2 2 2 ...
 $ Measure: chr  "Length" "Length" "Length" "Length" ...
 $ Value  : num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
```




Value
8
6
4
2
0
setosa
versicolor
virginica
Species
Part


Value
8
6
4
2
0
Length
Width
setosa
versicolor
virginica
Species
Part




```
Se comparan mejor en filas
```

Gráficos de línea con ggplot2

ggplot2, como no podía ser de otra manera, incorpora funciones para representar curvas. Vamos a ver algunos ejemplos de cómo representar líneas con **ggplot2**. En concreto vamos a presentar la función **geom_line()**.

Vamos a generar varios dataframes para ver las posibilidades de esta función.

```
Dosis.Medicina<-data.frame(Dosis=c("D0.5mg", "D1mg", "D2mg"),
                Efectividad=c(30.2, 80.1, 99.9))
head(Dosis.Medicina)
```

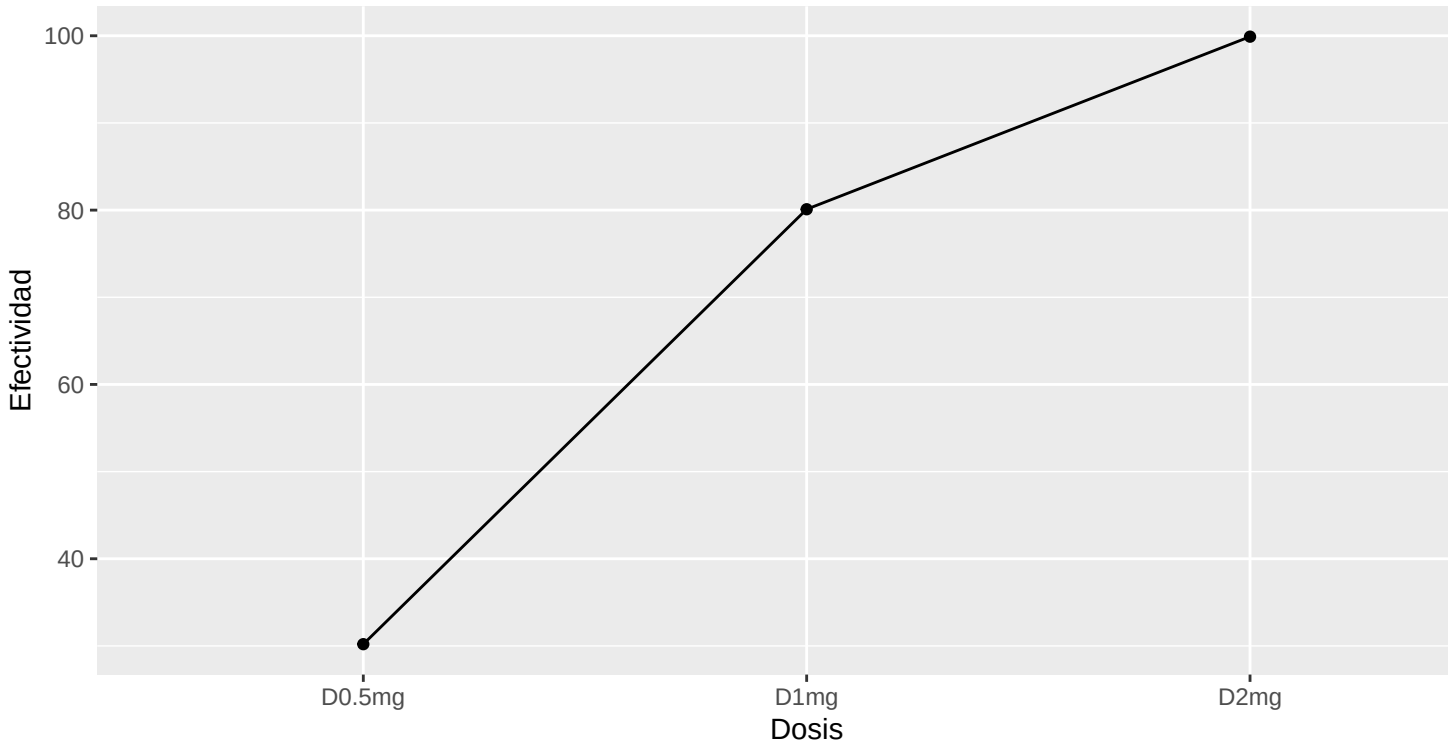
```
   Dosis Efectividad
1 D0.5mg        30.2
2   D1mg        80.1
3   D2mg        99.9
```

```
Dosis.Medicina2 <- data.frame(Complemento=rep(c("Vitamina A", "Vitamina C"), each=3),
                Dosis=rep(c("D0.5mg", "D1mg", "D2mg"),2),
                Efectividad=c(6.8, 15, 33, 4.2, 10, 29.5))
head(Dosis.Medicina2)
```

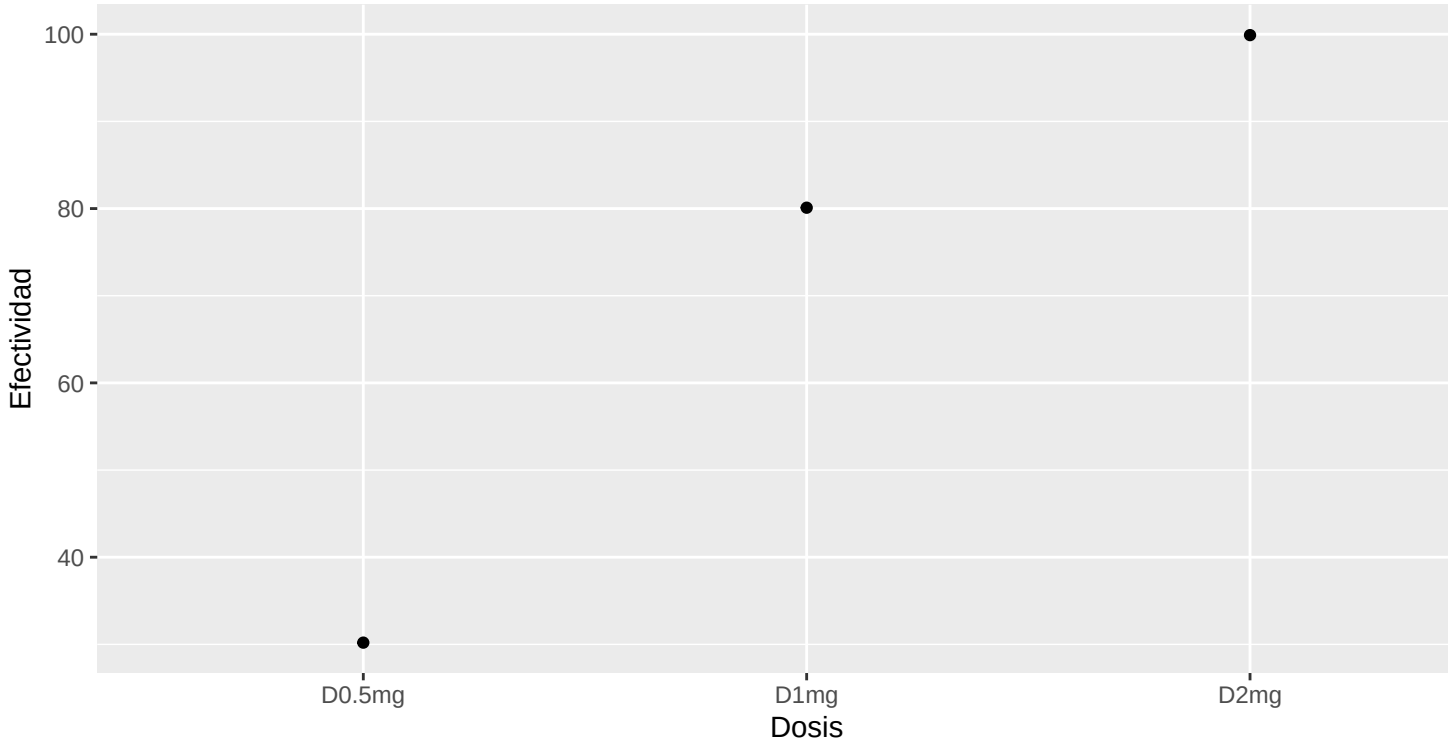
```
  Complemento  Dosis Efectividad
1  Vitamina A D0.5mg         6.8
2  Vitamina A   D1mg        15.0
3  Vitamina A   D2mg        33.0
4  Vitamina C D0.5mg         4.2
5  Vitamina C   D1mg        10.0
6  Vitamina C   D2mg        29.5
```

Vamos a representar en el eje x la variable `Dosis` y en el eje y la `Efectividad` del tratamiento. Puedes observar que estamos juntando dos capas, por una parte los puntos en el gráfico (**geom_point()**) y la línea (**geom_line()**). La opción `group` permite que se unan puntos consecutivos, en el gráfico de líneas, cuando las dos variables representadas no son ambas numéricas.

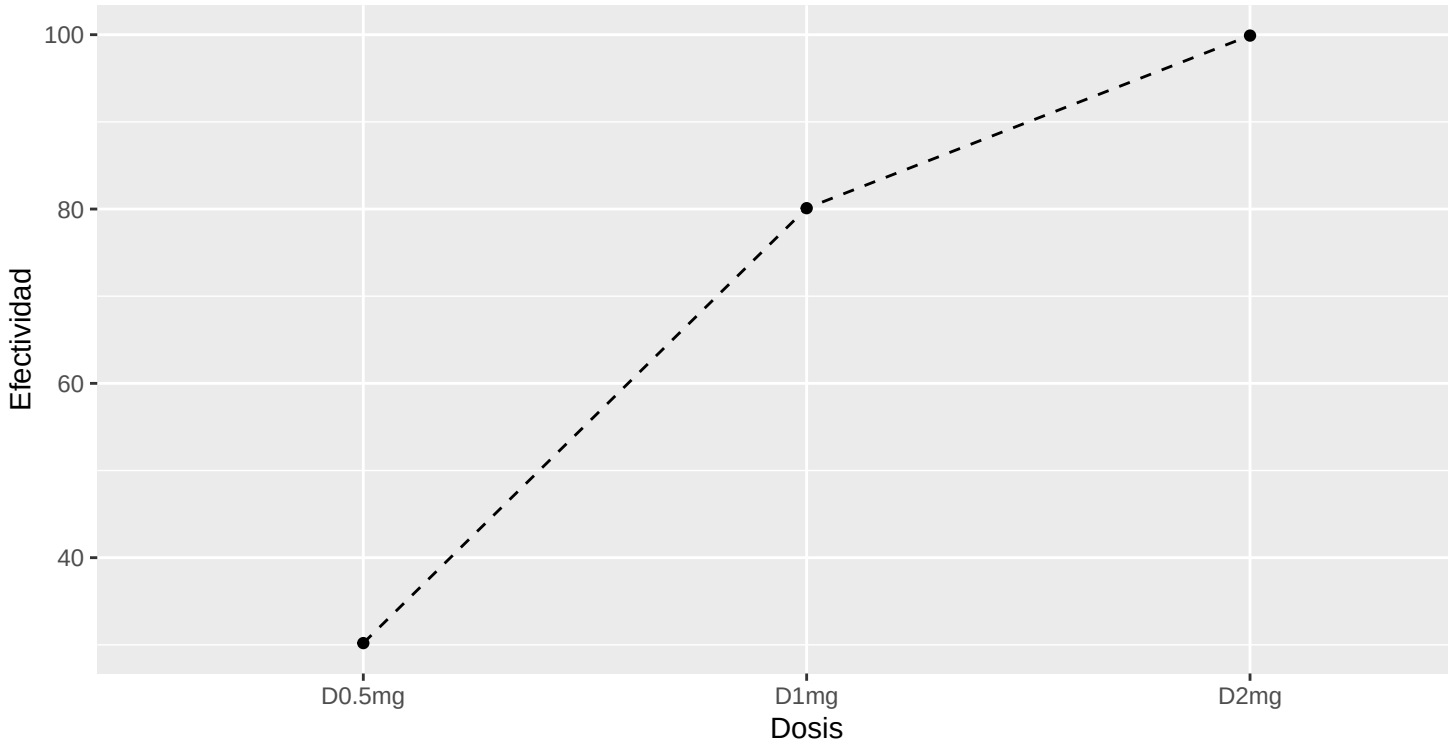
```
library(ggplot2)
# Gráfico de línea básico
# El aesthetic group=1 es un agrupamiento "tonto" (da igual si se pone otro valor) para evitar el agrup
ggplot(data=Dosis.Medicina, aes(x=Dosis, y=Efectividad, group=1)) +
  geom_line()+
  geom_point()
```



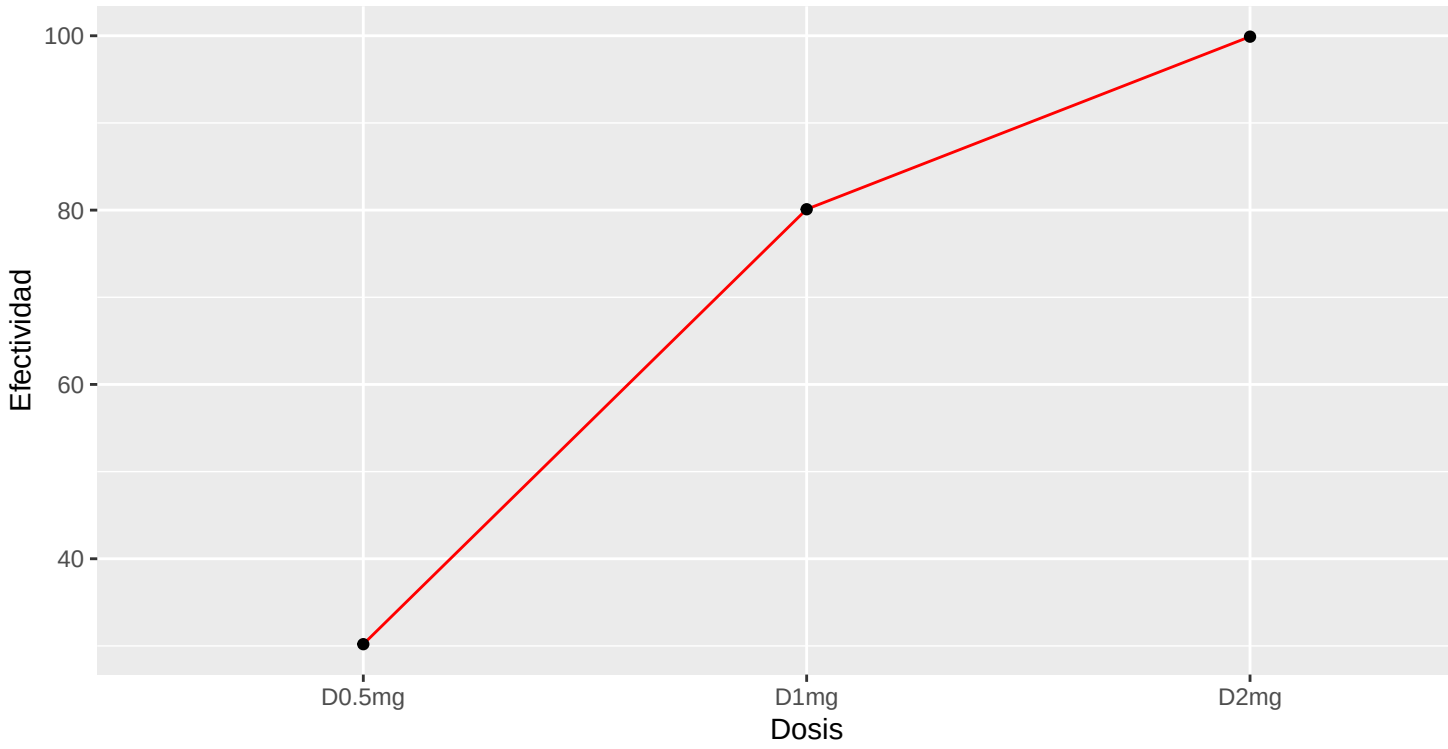

```
ggplot(data=Dosis.Medicina, aes(x=Dosis, y=Efectividad)) +
  geom_line()+
  geom_point()
```



```r
# Podemos definir otro tipo de línea
ggplot(data=Dosis.Medicina, aes(x=Dosis, y=Efectividad, group=1)) +
  geom_line(linetype = "dashed")+
  geom_point()
```

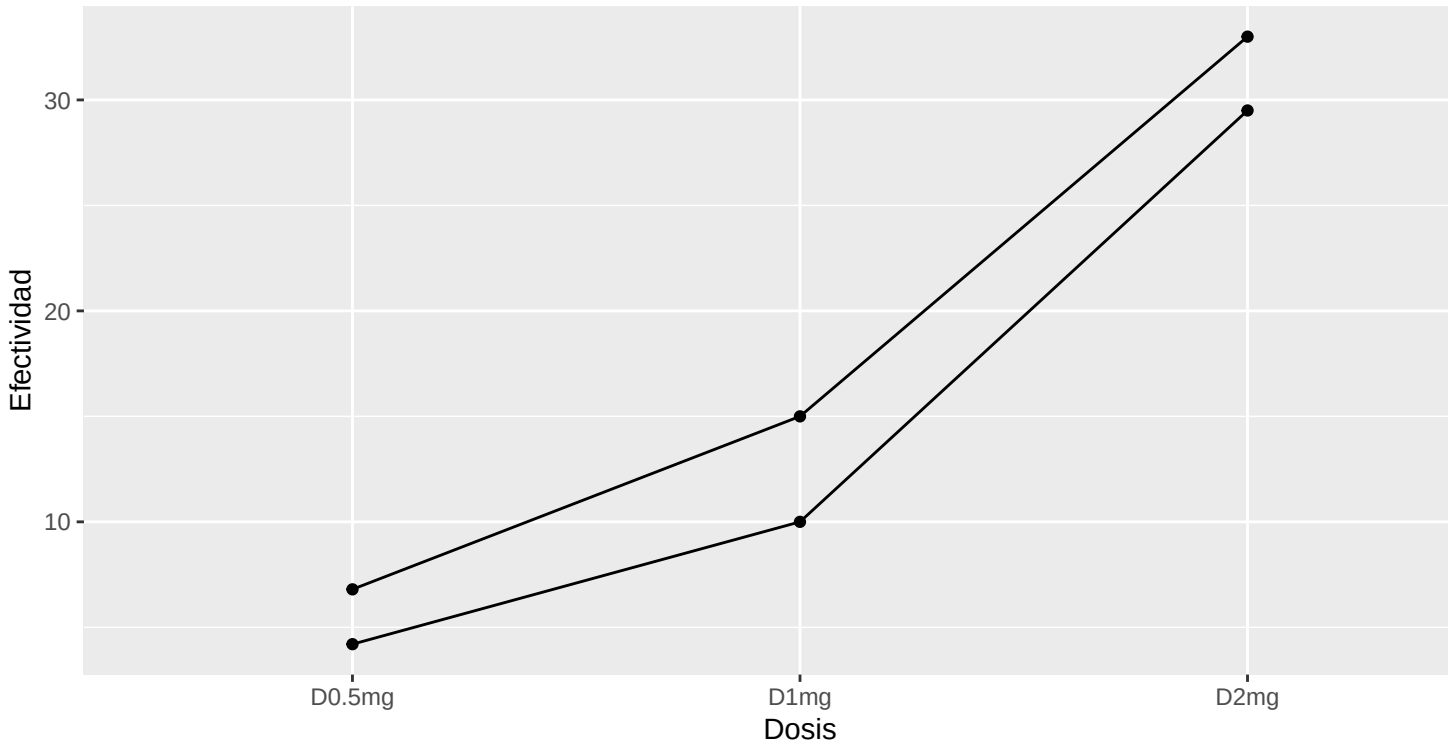



```r
# O cambiar el color de la línea
ggplot(data=Dosis.Medicina, aes(x=Dosis, y=Efectividad, group=1)) +
  geom_line(color="red")+
  geom_point()
```

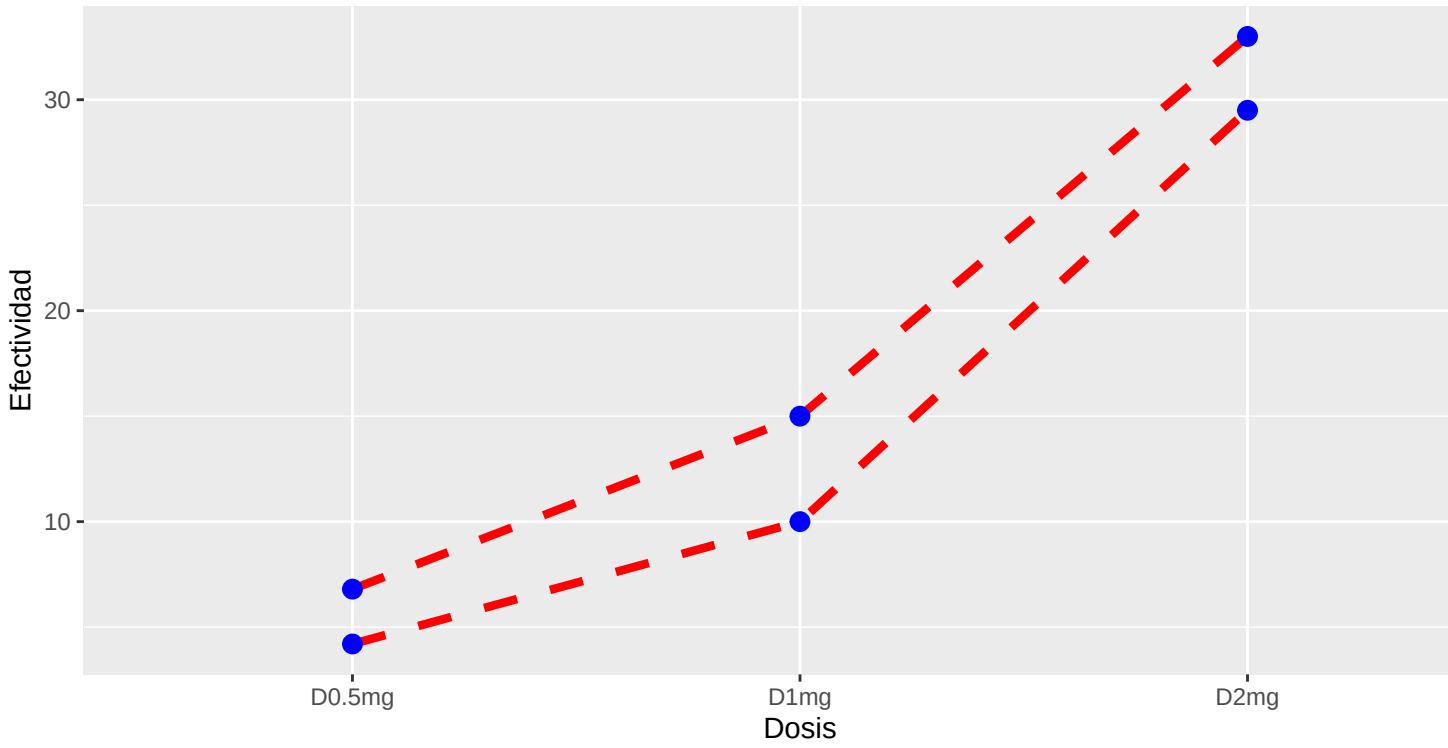


Podemos representar más de un gráfico de línea, con los datos agrupados en base a una tercera variable. Veamos un ejemplo.

```
# Podemos agrupar los datos en base a una tercera variable
ggplot(data=Dosis.Medicina2, aes(x=Dosis, y=Efectividad, group=Complemento)) +
  geom_line()+
  geom_point()
```

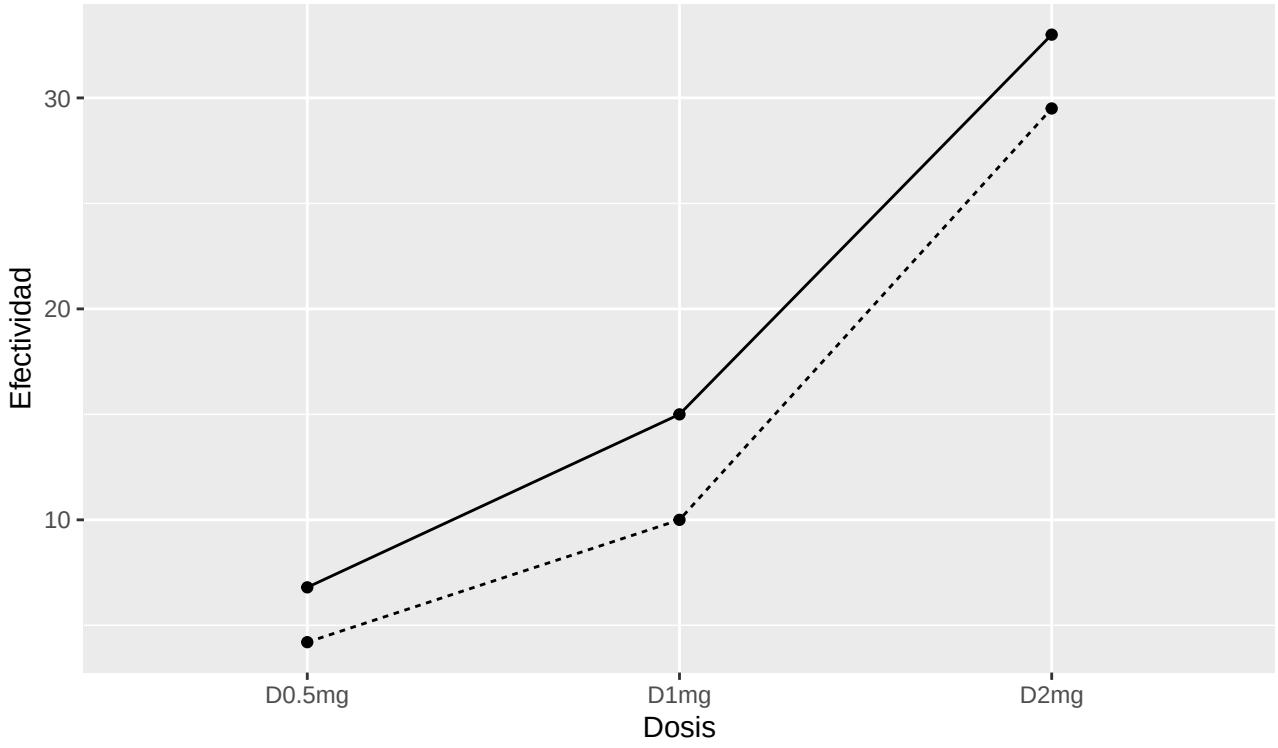



```
# O cambiar el tipo de línea y su grosor, y el tamaño de los puntos
ggplot(data=Dosis.Medicina2, aes(x=Dosis, y=Efectividad, group=Complemento)) +
  geom_line(linetype="dashed", color="red", size=1.5)+
  geom_point(color="blue", size=3)
```

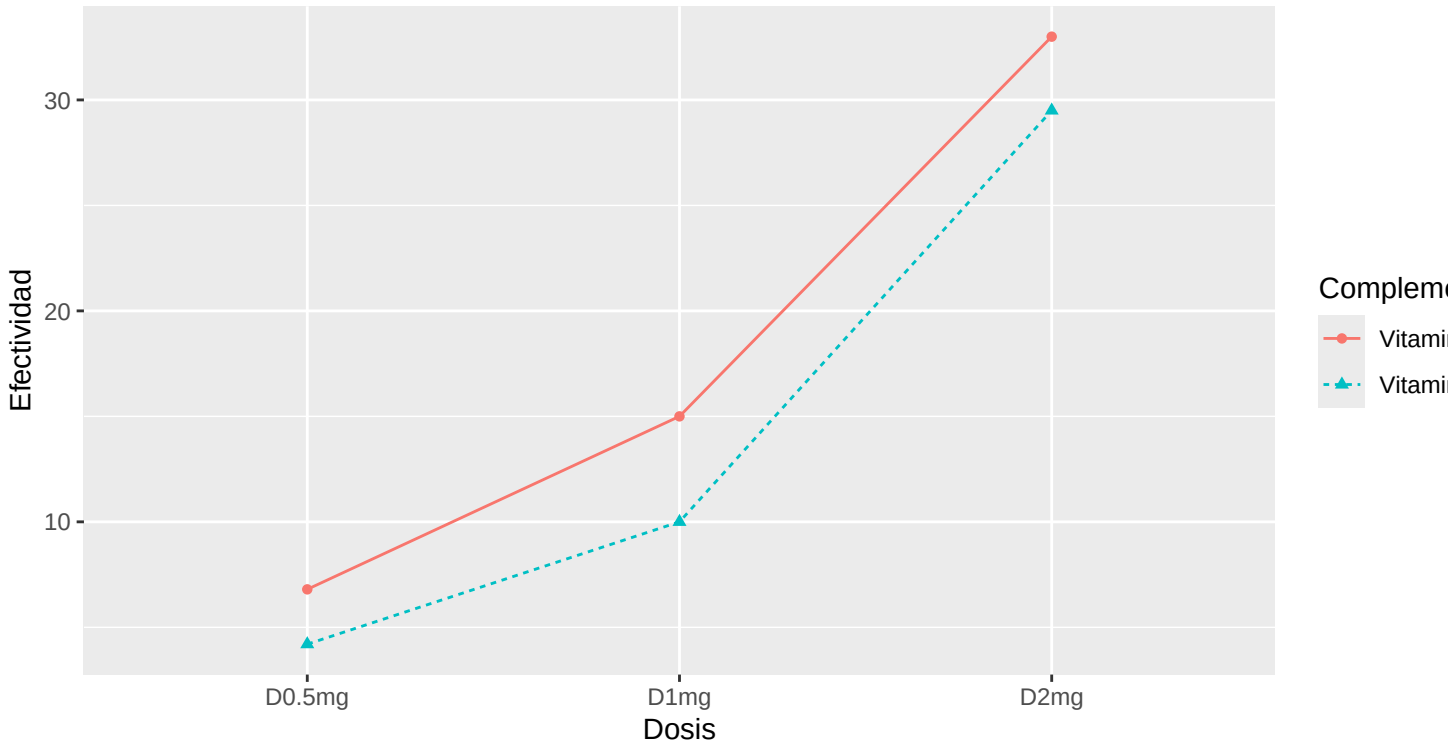


Se puede modificar el tipo de línea, la forma del punto o color dependiendo del grupo que queramos representar. En este caso el tipo de línea o la forma del punto no será un parámetro de **geom_line()** ya que estas propiedades van asociadas a una variable. Para poder conseguir esto tendremos que definir un nuevo **aes()** dentro de **geom_line()** para el tipo de línea y otro en **geom_point()** para la forma del punto. El parámetro `group` permite especificar respecto de qué variable se realiza el agrupamiento, sin necesidad de cambiar, por ejemplo, el color o la forma.

```
# Change line types by groups (supp)
ggplot(data=Dosis.Medicina2, aes(x=Dosis, y=Efectividad, group=Complemento)) +
  geom_line(aes(linetype=Complemento))+
  geom_point()
```



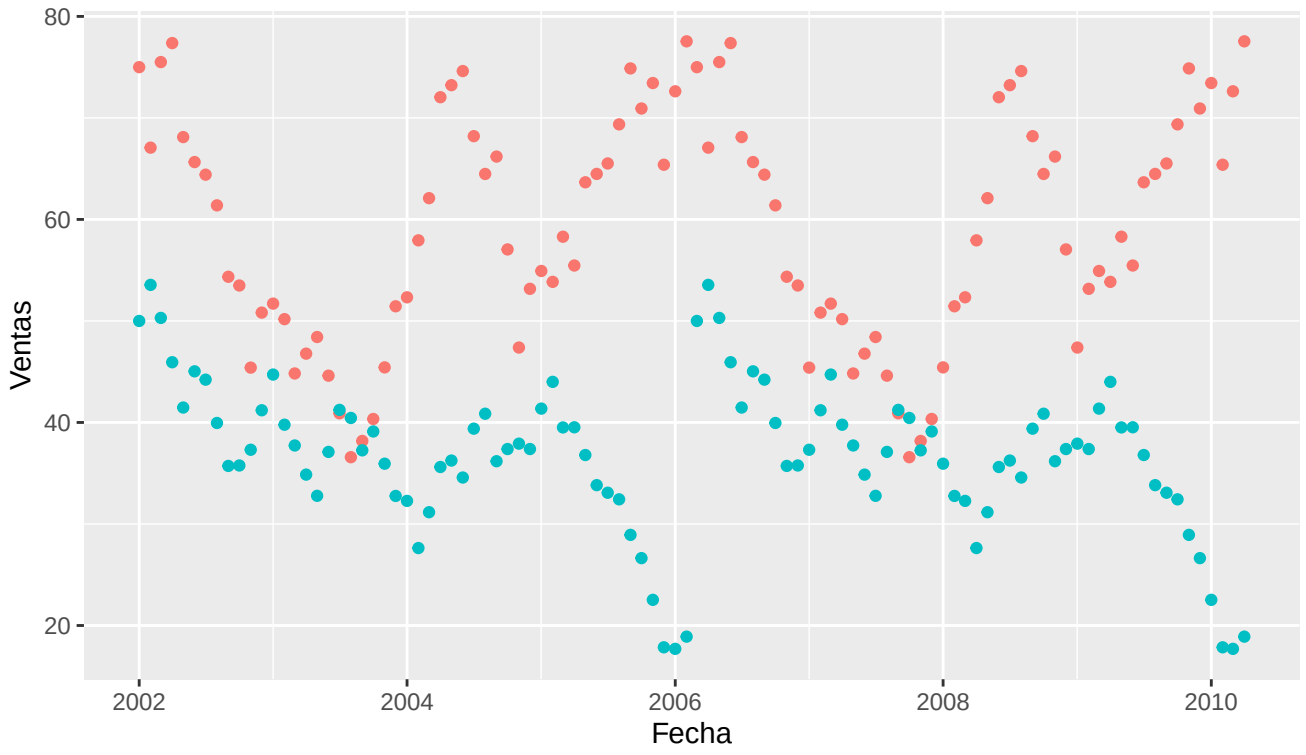

```
# Change line types and point shapes
ggplot(data=Dosis.Medicina2, aes(x=Dosis, y=Efectividad, group=Complemento)) +
  geom_line(aes(linetype=Complemento,color=Complemento))+
  geom_point(aes(shape=Complemento,color=Complemento))
```

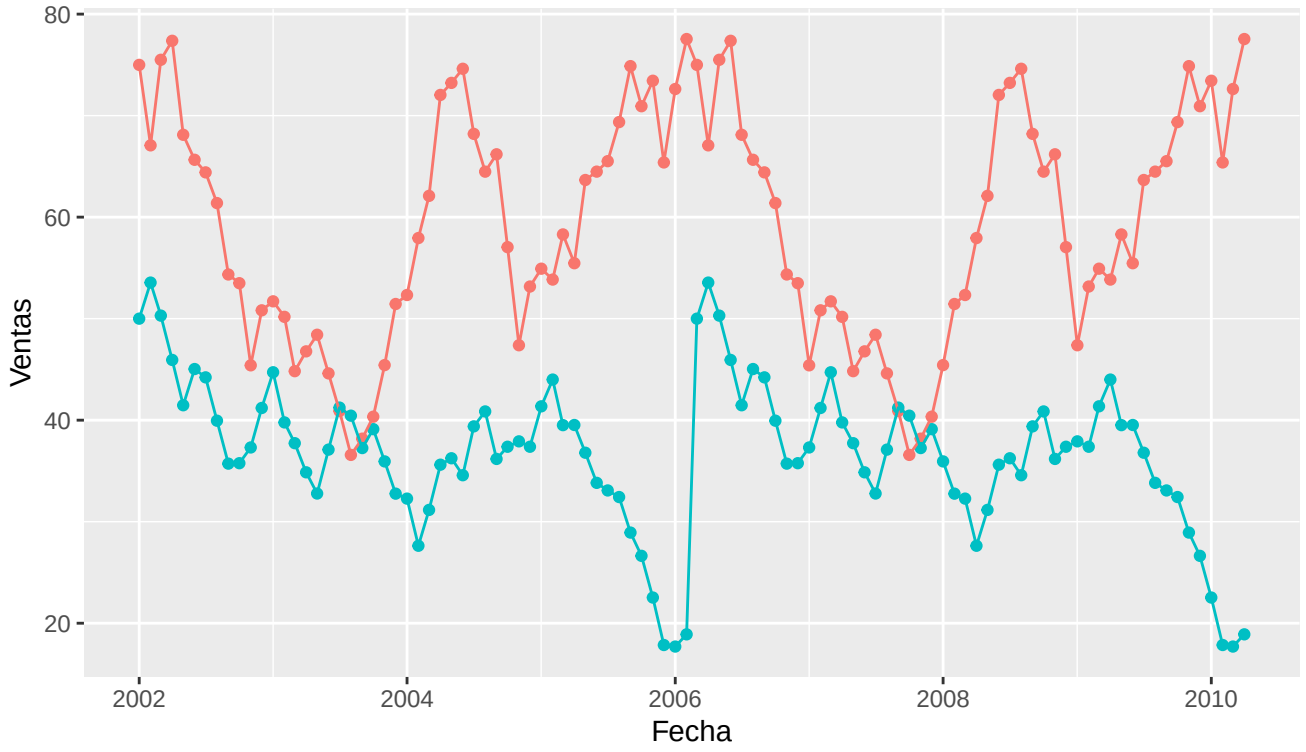


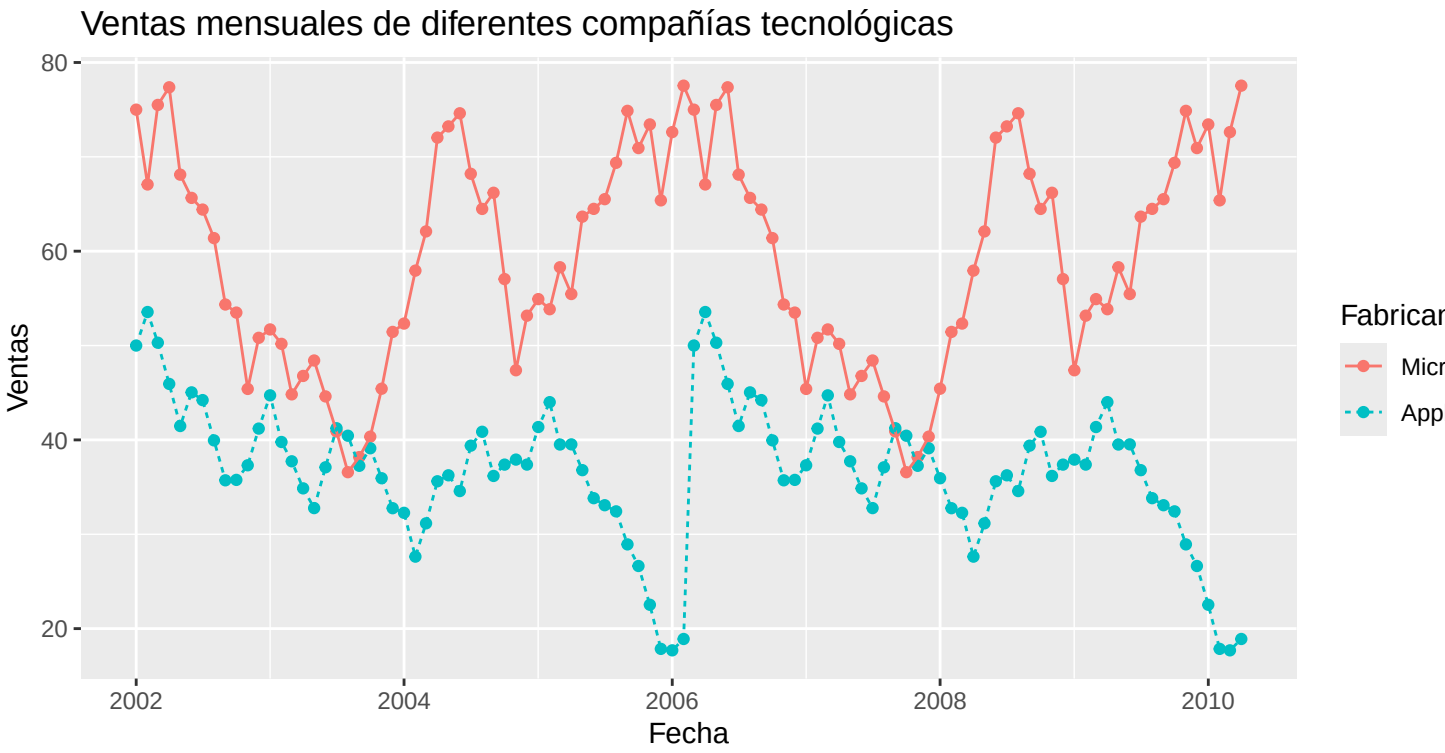
Ejercicio 4

- Carga el fichero `Ejercicio4_Data.Rdata`. Examina las variables que tiene dicho dataset.
- Representa un gráfico de puntos, en el que se muestre la fecha en el eje x y las ventas en el eje y. Queremos que los datos de cada fabricante sean de un color distinto. ¿Es este gráfico válido para evaluar la evolución de las ventas de cada fabricante?
- Representa un gráfico como el anterior en el que se muestre la línea que une los puntos que antes has representado.
- Modifica el gráfico anterior para que la tendencia de cada fabricante este trazada con un tipo de línea diferente.
- Añade el siguiente título al gráfico *"Ventas mensuales de diferentes compañías tecnológicas"*.
- Almacena en formato **pdf** el gráfico resultante en un fichero denominado `Ejercicio4.pdf`.

```
'data.frame':   200 obs. of  3 variables:
 $ Fecha     : Date, format: "2002-01-01" "2002-02-01" "2002-03-01" ...
 $ Fabricante: Factor w/ 2 levels "Microsoft","Apple": 1 1 1 1 1 1 1 1 1 1 ...
 $ Ventas    : num  75 67.1 75.5 77.4 68.1 ...
```


80
60
40
20
Ventas
2002
2004
2006
2008
2010
Fecha
Fabricar
Micr
App

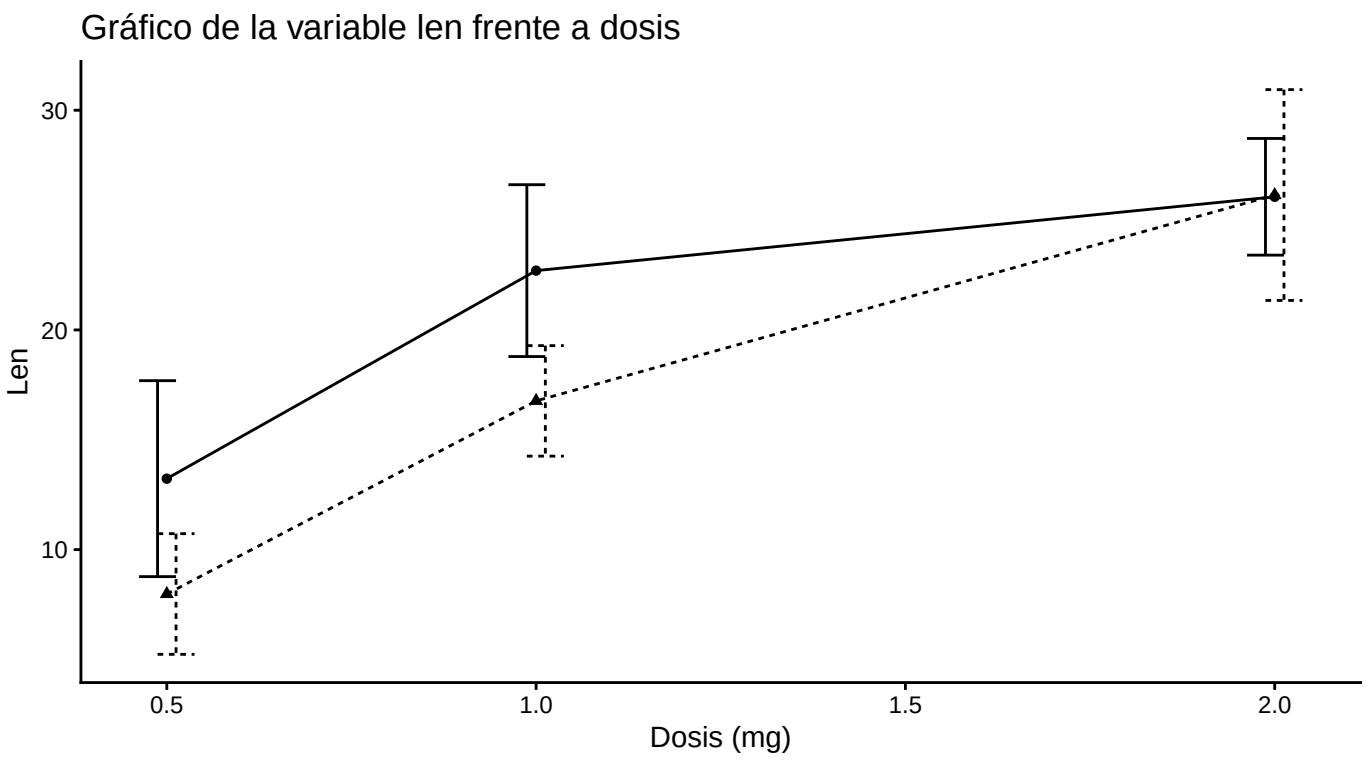

80
60
40
20
Ventas
2002
2004
2006
2008
2010
Fecha
Fabricar
Micr
App



Barras de error con ggplot2

En un gráfico de línea podemos añadir también barras de error. Para ver de lo que estamos hablando cargamos el fichero `ExperimentoEjemplo.Rdata`. Fíjate en cómo se calculan y añaden las barras de error. Del mismo modo, este ejemplo es interesante para ver otra forma de añadir títulos y etiquetas en los ejes mediante la función **labs()**. ¿Podemos inferir qué hace el argumento **position_dodge(0.05)** de la función **geom_errorbar()**?

```
load("./data/ExperimentoEjemplo.Rdata")
# Un gráfico simple
# Podemos cambiar los tipos de línea y la forma de los puntos por grupos.
ggplot(df3, aes(x=dose, y=len, shape=supp, linetype=supp))+
  # position_dodge: desplaza la barras de error para facilitar la visualización
    geom_errorbar(aes(ymin=len-sd, ymax=len+sd), width=.1, position=position_dodge(0.05)) +
    geom_line() +
    geom_point()+
    labs(title="Gráfico de la variable len frente a dosis",x="Dosis (mg)", y = "Len")+
    theme_classic()
```




```
# Si no nos gustan los colores... los podemos cambiar...
# y añadir barras de error
p <- ggplot(df3, aes(x=dose, y=len, group = supp, color=supp))+
    geom_errorbar(aes(ymin=len-sd, ymax=len+sd), width=0.1,
    position=position_dodge(0.05)) +
    geom_line(aes(linetype=supp)) +
    geom_point(aes(shape=supp))+
    labs(title="Gráfico de la variable len frente a dosis",x="Dosis (mg)", y = "Len") +
    theme_classic()
p + theme_classic() + scale_color_brewer(palette="Blues")
```

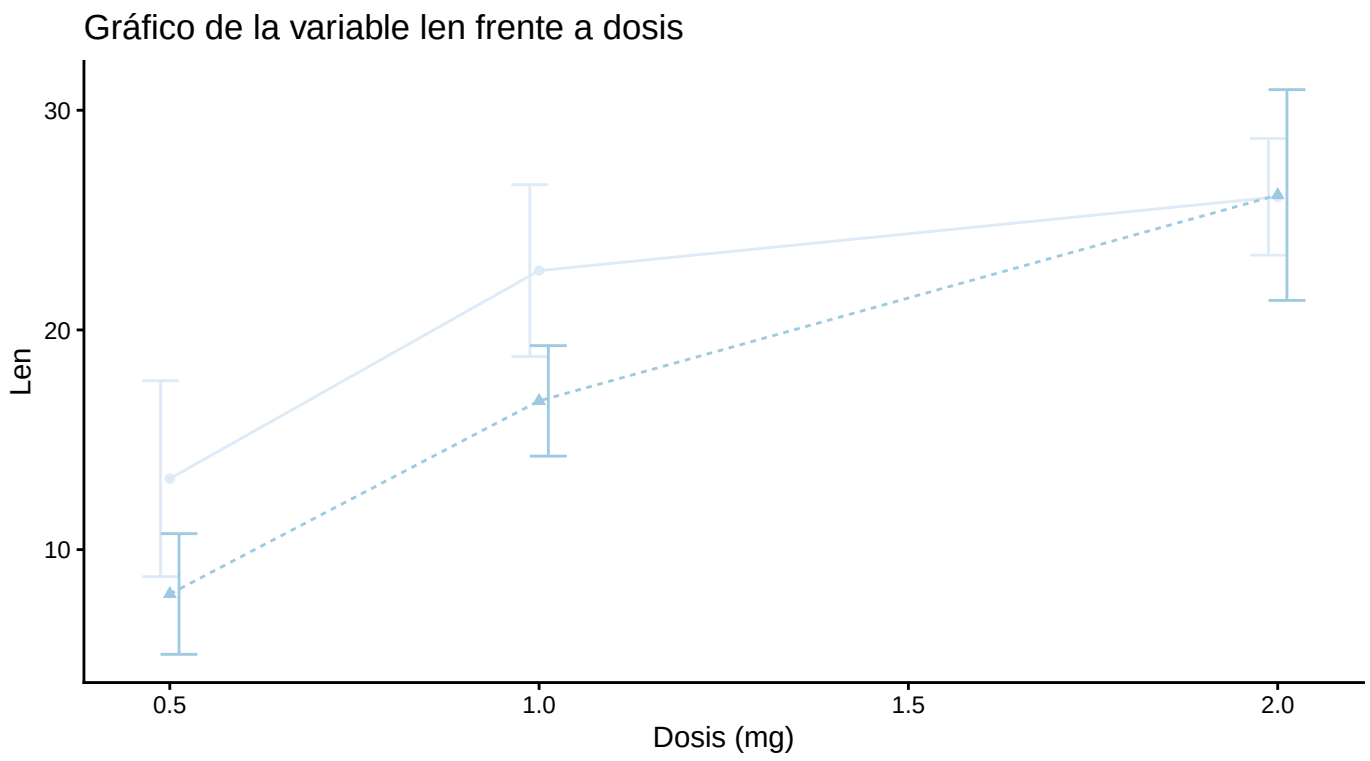


Gráfico de la variable len frente a dosis
30
20
10
Len
0.5
1.0
1.5
2.0
Dosis (mg)
sup

```
p
```

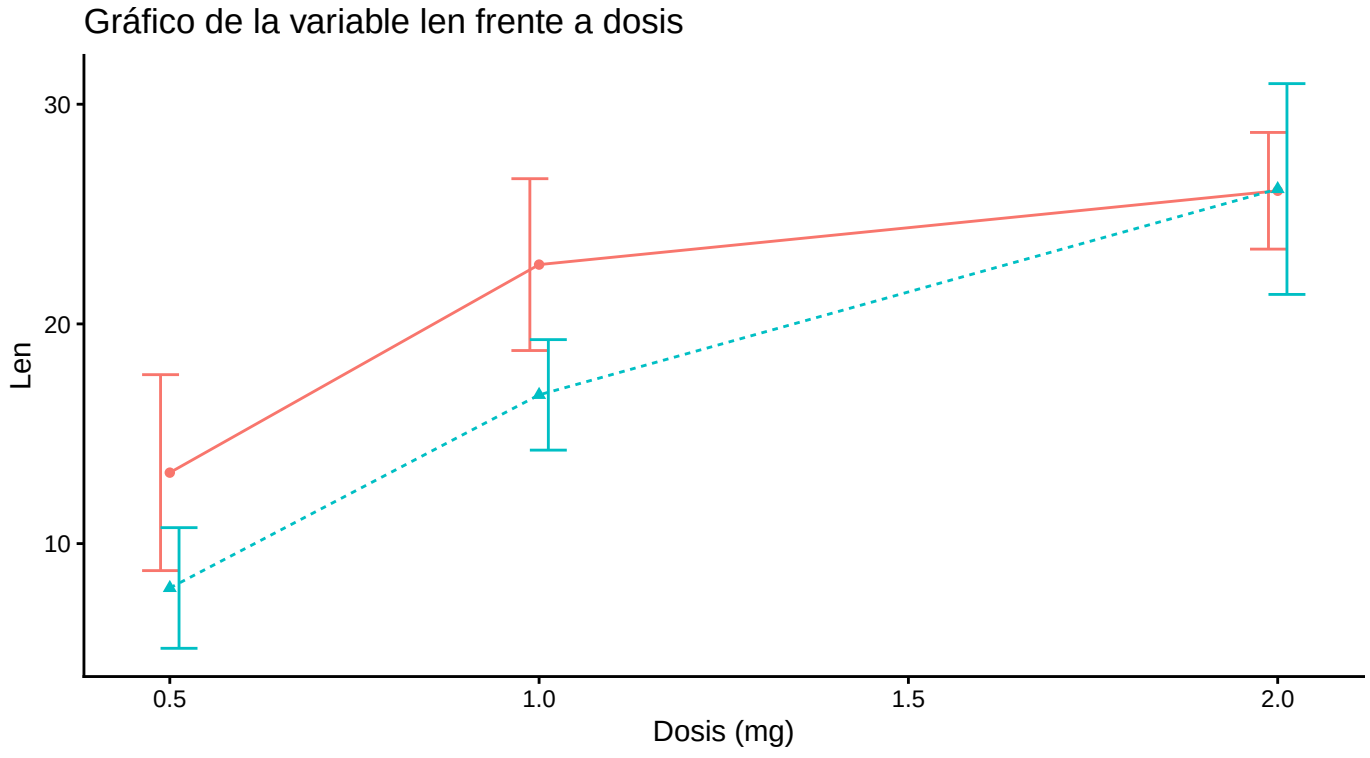
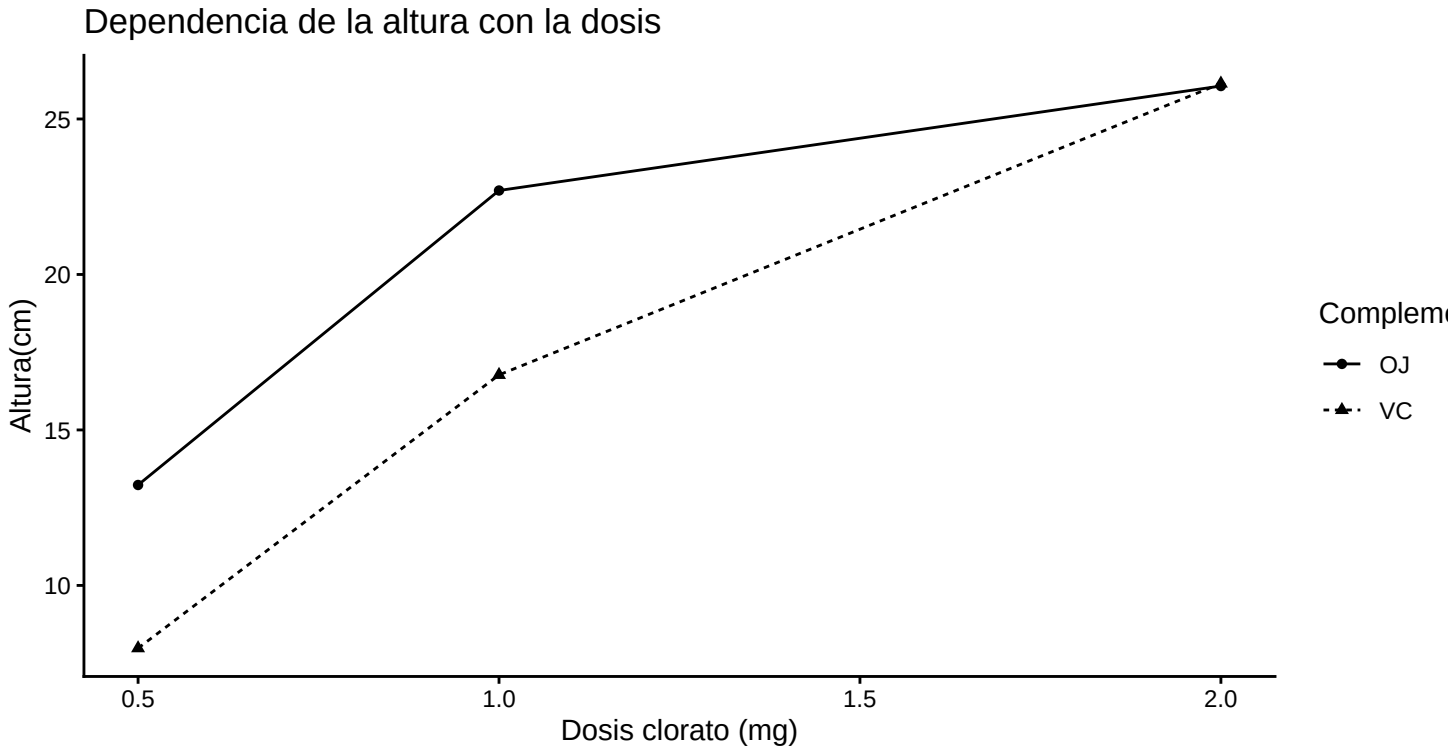


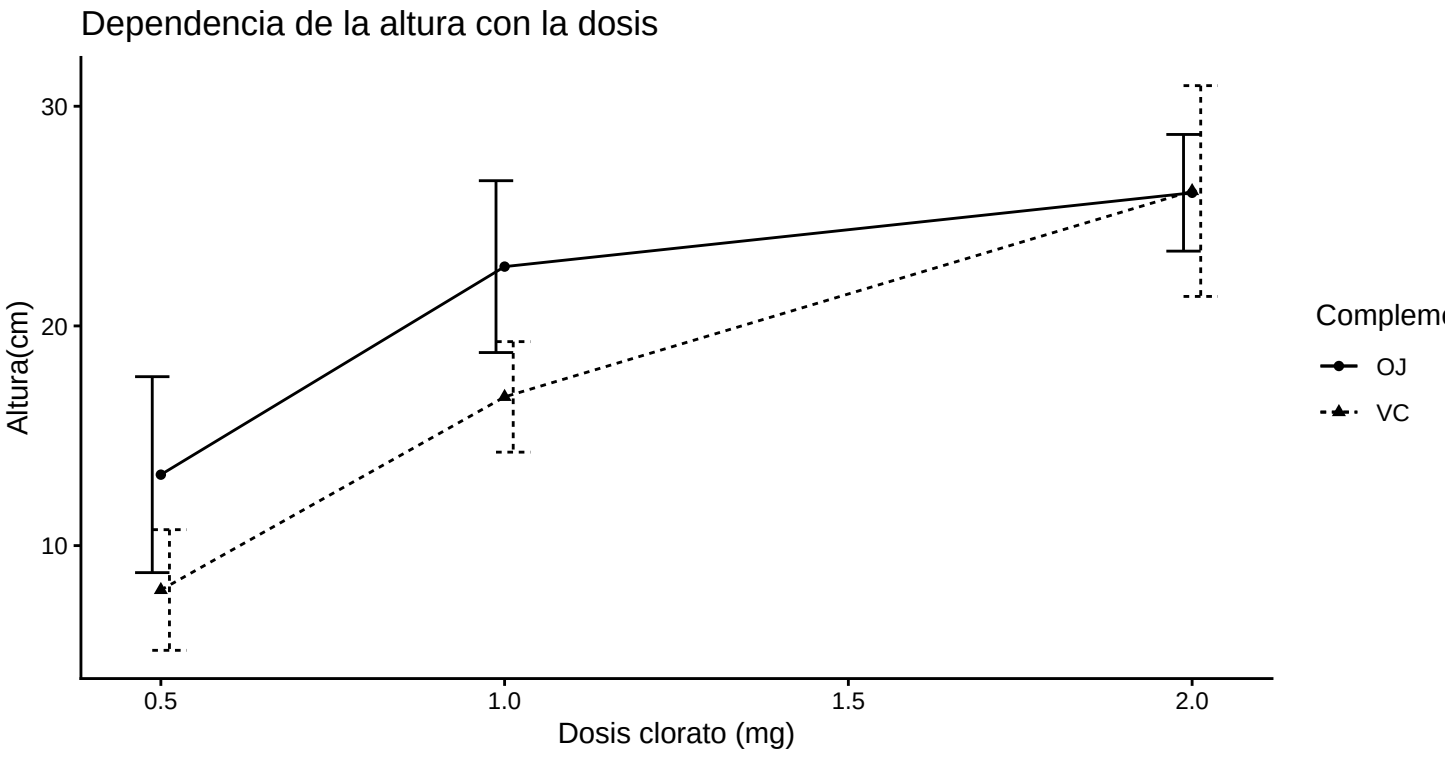
Gráfico de la variable len frente a dosis
30
20
10
Len
0.5
1.0
1.5
2.0
Dosis (mg)
sup

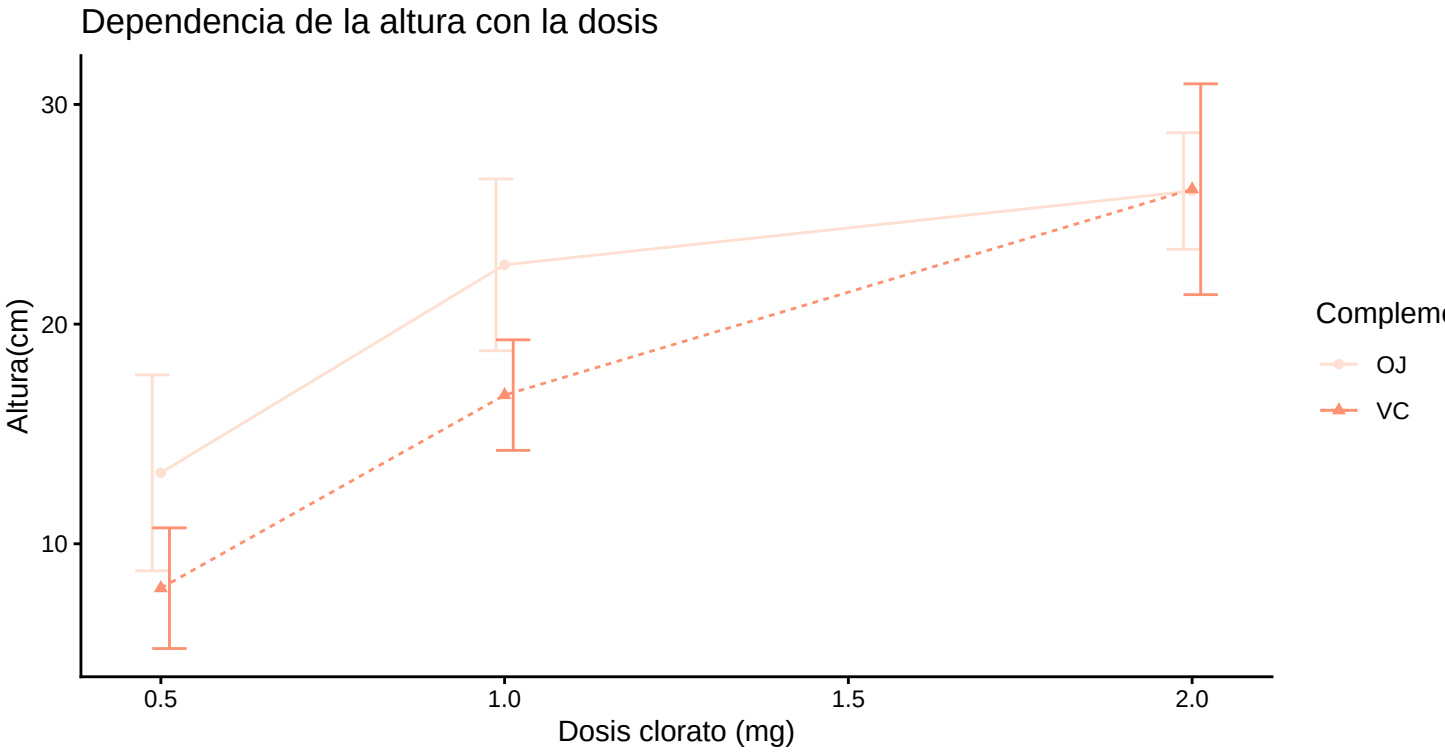
Ejercicio 5

Añadir barras de error a un gráfico de línea.

- Carga el fichero `Experimento.Rdata`. Examina las variables que tiene dicho dataset.
- Representa un gráfico de puntos, en el que se muestre la `Dosis` en el eje x y la `Altura` en el eje y. Queremos que los datos de cada `Complemento` se representen en una línea diferente con diferentes trazos. Hazlo incluyendo las opciones de tipo de línea y forma del punto dentro del **aes()** de la capa **ggplot**.
- Crea otro gráfico añadiendo al gráfico anterior barras de error en cada punto. El error viene dado por la variable `Error_Altura`.
- Crea otro gráfico, como el anterior (**idéntico**), pero en este caso no puedes incluir el tipo de línea y el tipo de punto en aes() de la capa ggplot. Además queremos que la paleta de colores sea en tonos de rojos (`scale_color_brewer`).
- Almacena en formato **pdf** el gráfico resultante en un fichero denominado `Ejercicio5.pdf`.



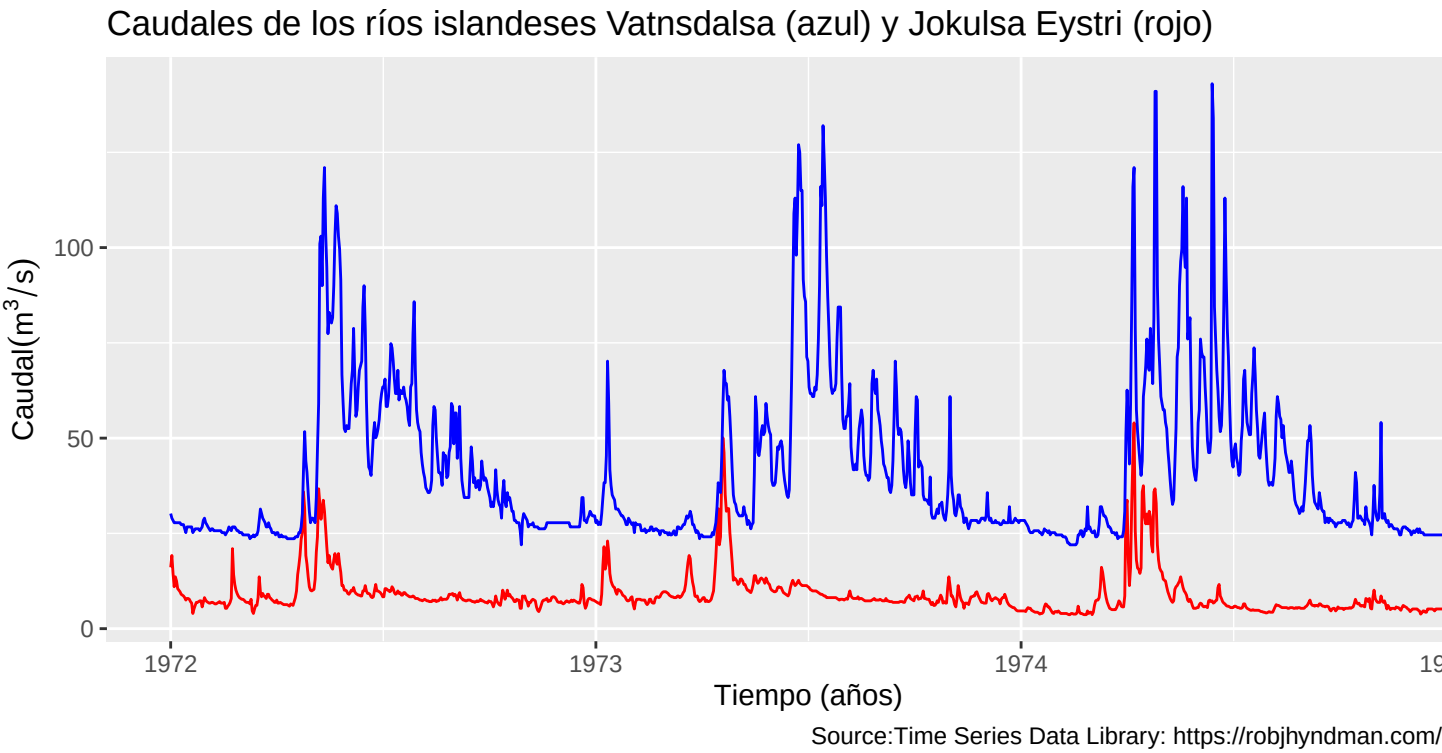


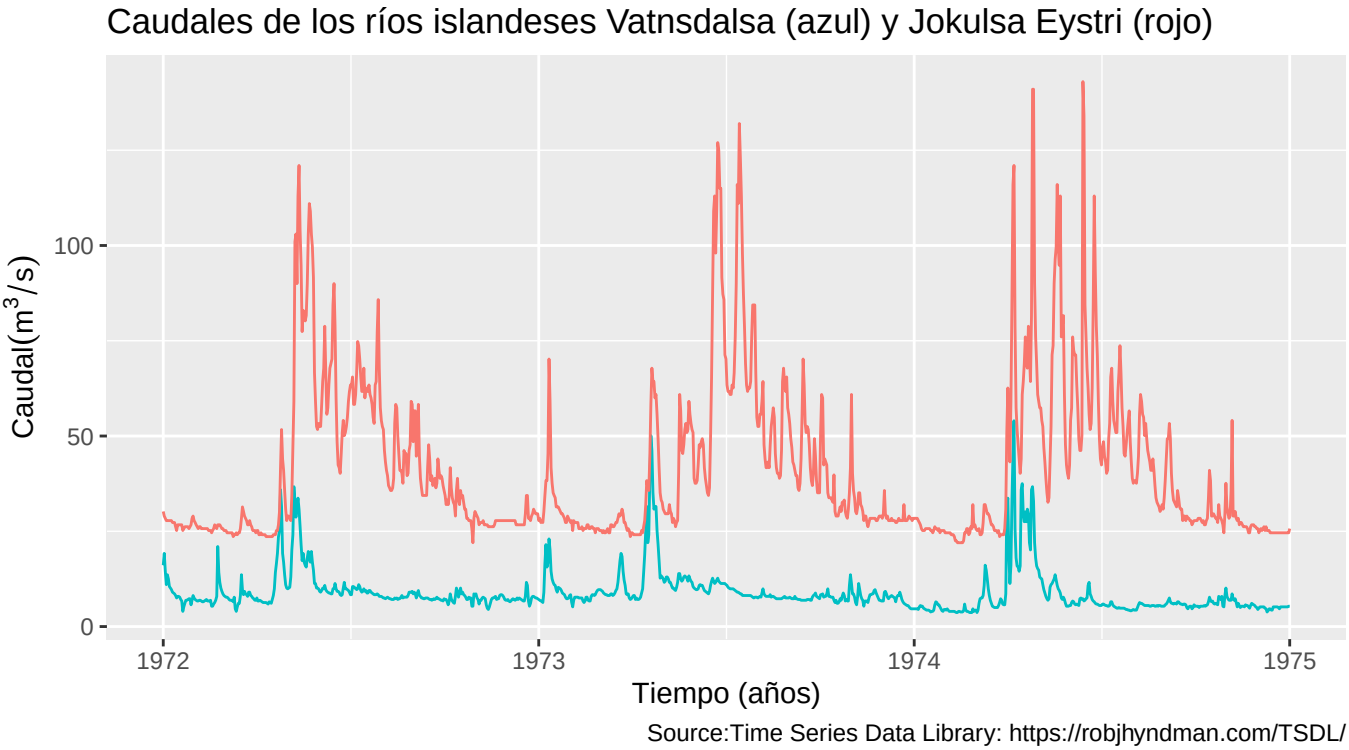


Ejercicio 6

El fichero `iceRiver.RData` contiene información del caudal de dos ríos islandeses medidos a diarios durante 3 años. Puedes obtener información detallada de las variables usando `> ?tseries::ice.river`

- Representa el caudal de ambos ríos a lo largo del período considerado coloreando una serie (`flow.vat`) en color rojo y la otra en color azul.
- Utiliza la función `labs` para enriquecer el gráfico.
- Almacena en formato **pdf** el gráfico resultante en un fichero denominado `Ejercicio5.pdf`.



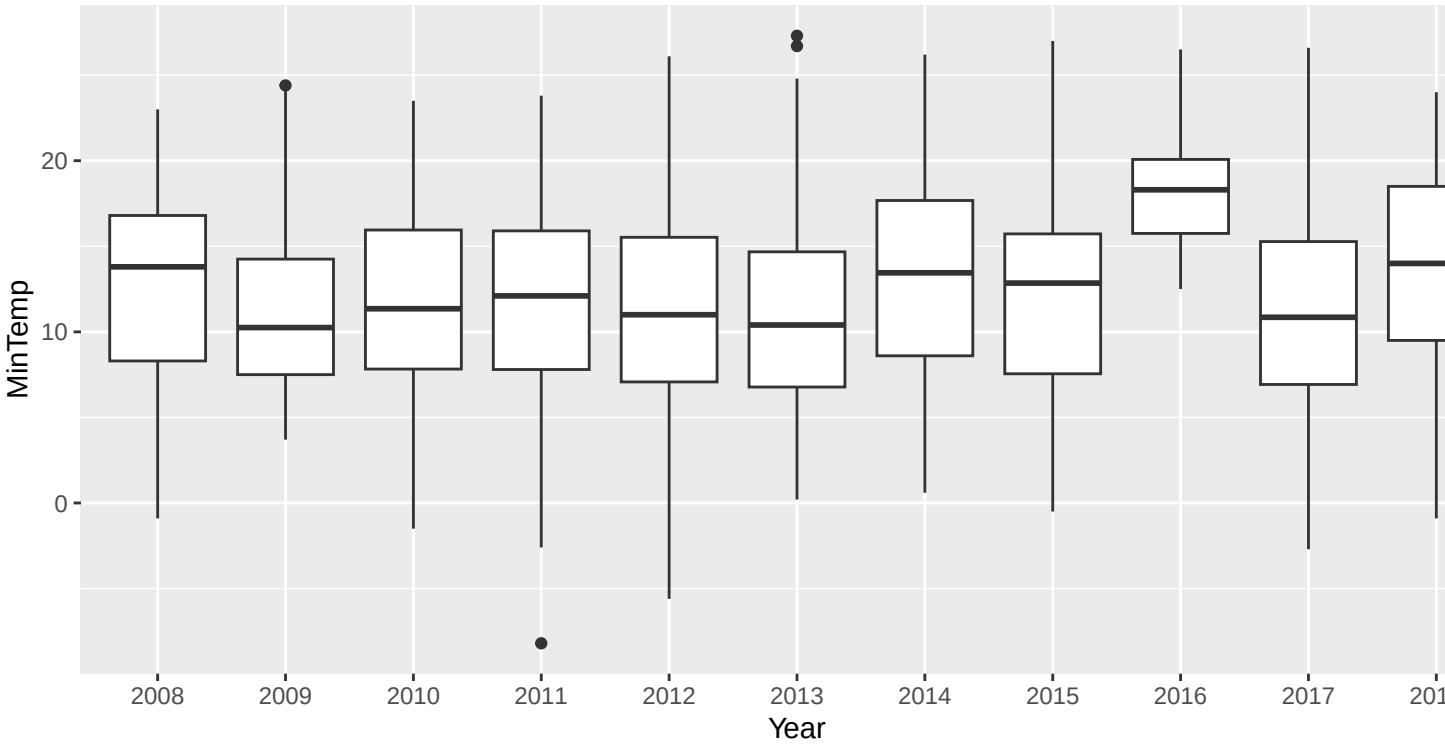


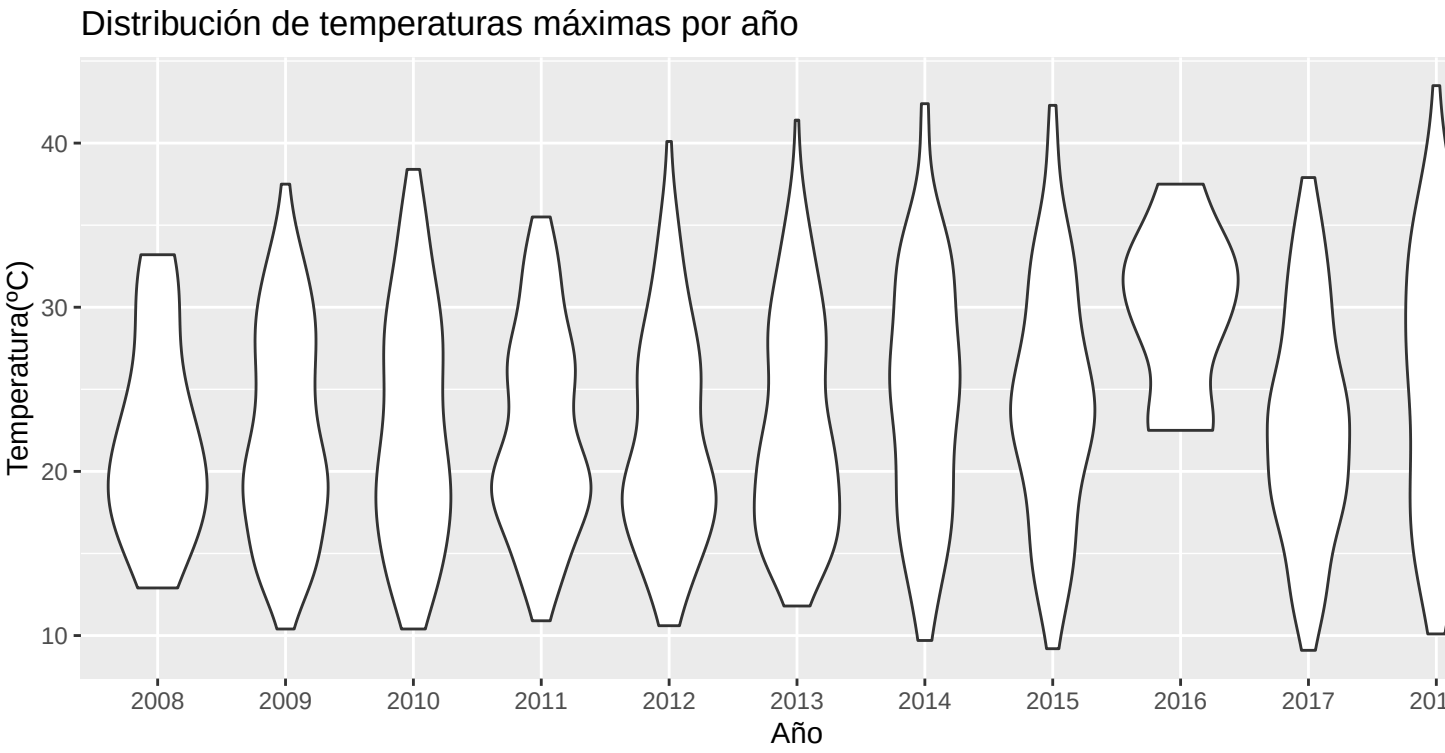
Gráficos boxplot con ggplot2

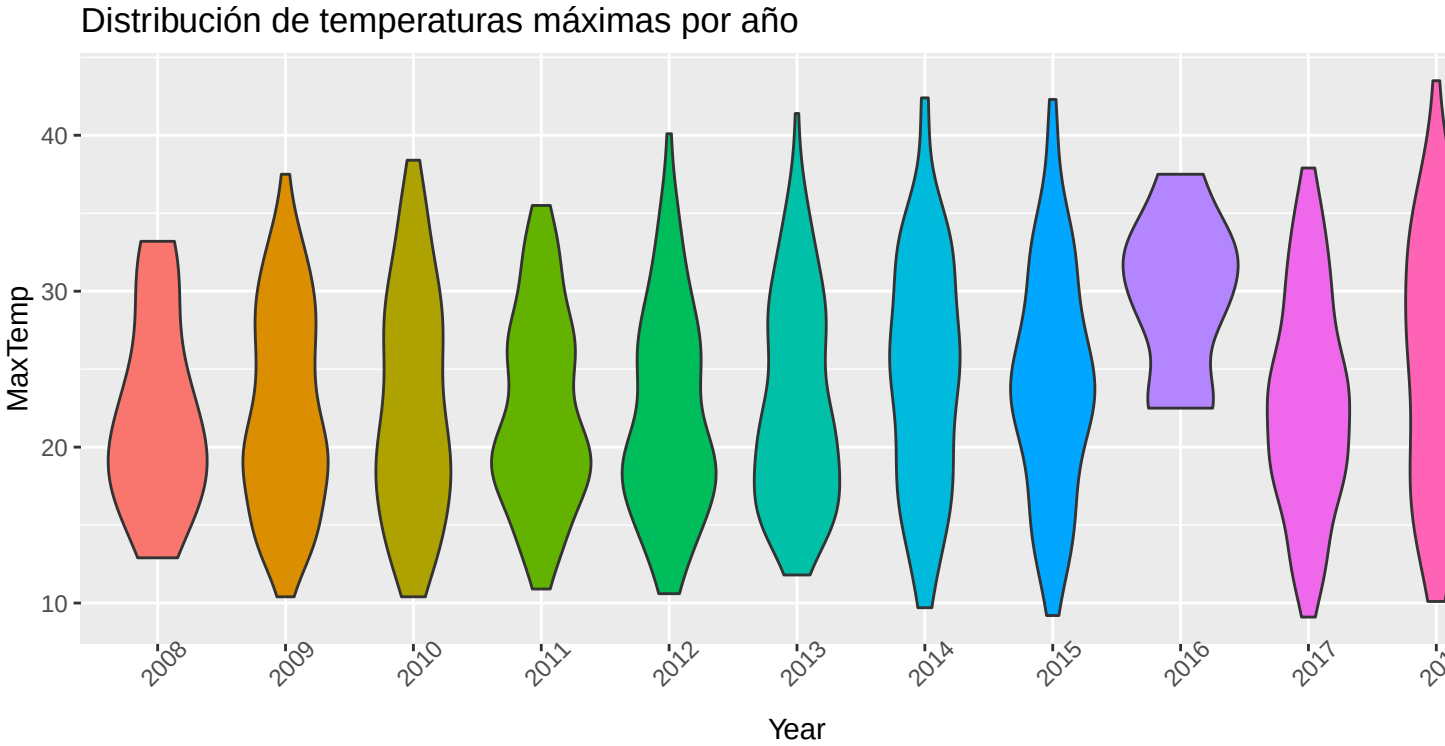
En esta sección vamos a utilizar **ggplot2** para representar distribuciones de datos. Para ello vamos a ver cómo representar gráficos de tipo boxplot. La función **geom_boxplot()** y **geom_violin()** permiten obtener gráficos de tipo de boxplot. La forma de definir los **aes()** es muy similar a como lo hacíamos con otras funciones "geom" que ya hemos utilizado.

Ejercicio 7

- Carga el fichero `weatherAUS_distributions.RData`.
- Crea un gráfico en el que estén los boxplots de la variable `MinTemp` para cada uno de los años presentes en la variable `Year` del dataset.
- Crea un gráfico en el que estén los gráficos de violín de la variable `MaxTemp` para cada uno de los años presentes en la variable `Year` del dataset. Añade en el mismo gráfico una capa de etiquetas de título (Distribución de temperaturas máximas por año), eje x (Año) y eje y (Temperatura (ºC)). Colorea el interior de los violines según la la variable `Year`.
- En el gráfico anterior elimina la leyenda y coloca los años en el eje x a 45 grados de inclinación. Busca información sobre cómo hacer esto.
- Almacena en formato **pdf** el gráfico resultante en un fichero denominado `Ejercicio7.pdf`.







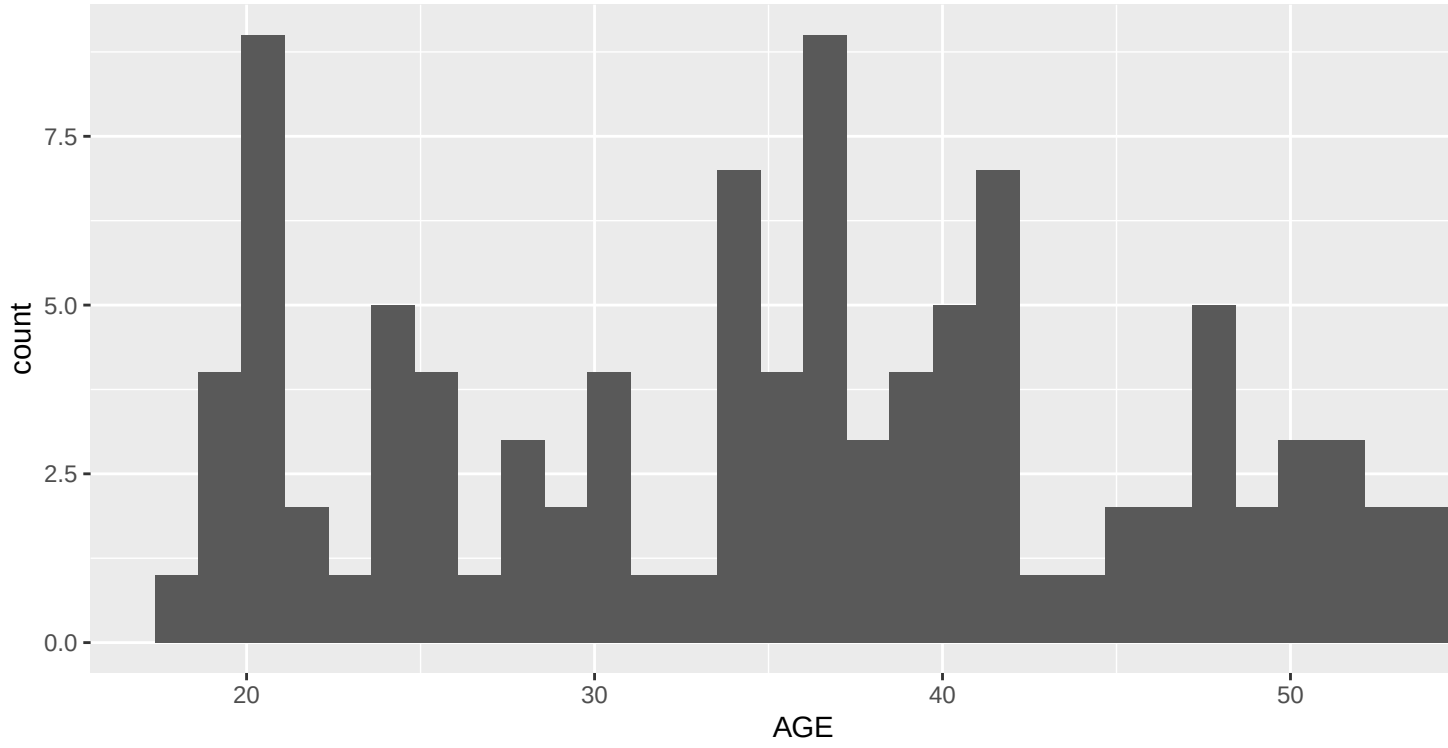
Histogramas con ggplot2

Como no podía ser de otra manera, **ggplot2()** permite obtener histogramas para representar una distribución de datos, para ello utilizaremos la función **geom_histogram()**. Veamos algunos ejemplos con un dataset

contenido en el fichero `dataset_hist1.Rdata` para que comprendas cómo utilizar esta función.

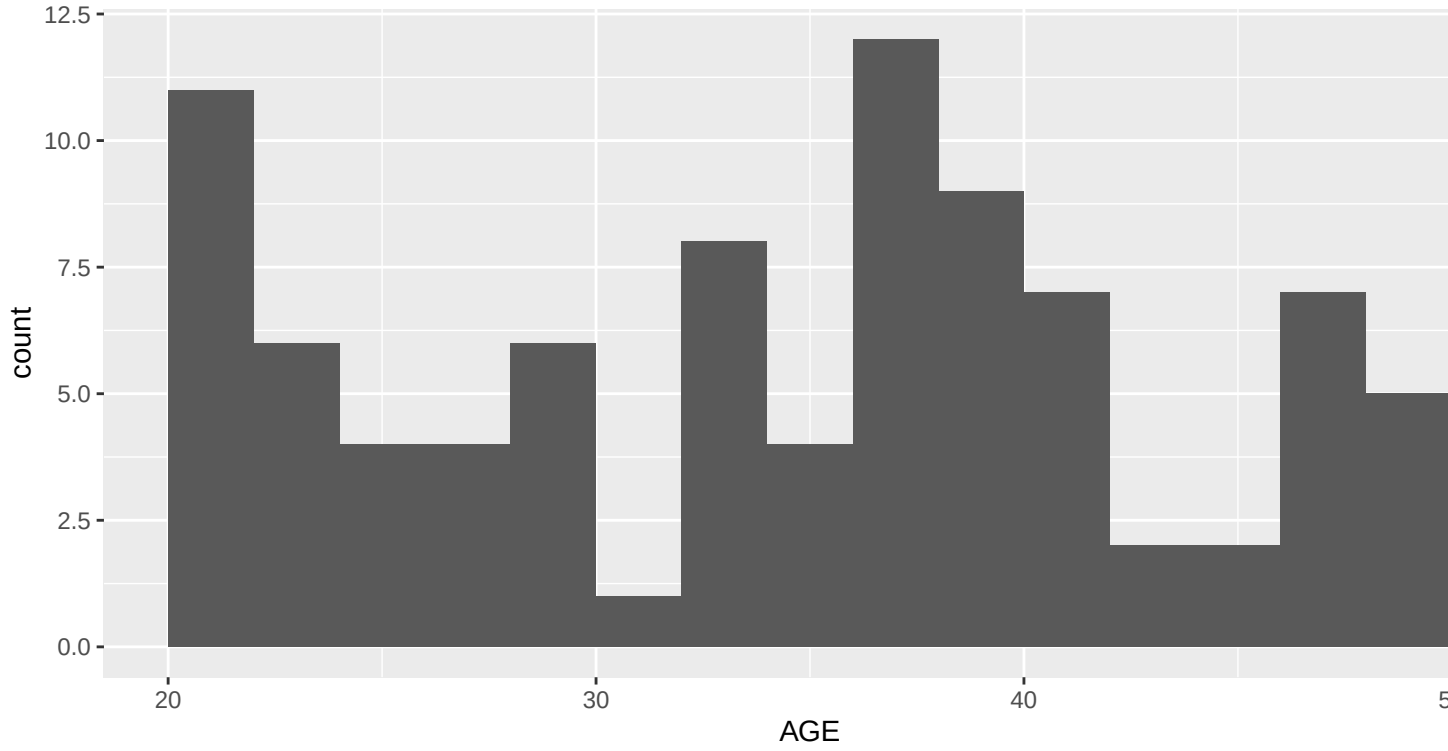
Veamos en primer lugar cómo hacer un histograma básico de la variable `AGE`. Podrás observar que la nomenclatura es muy intuitiva con todo lo que has ido viendo hasta el momento.

```
load("./data/dataset_hist1.Rdata")
#Histograma básico.
ggplot(data=dataset_hist1, aes(x=AGE)) + geom_histogram()
```



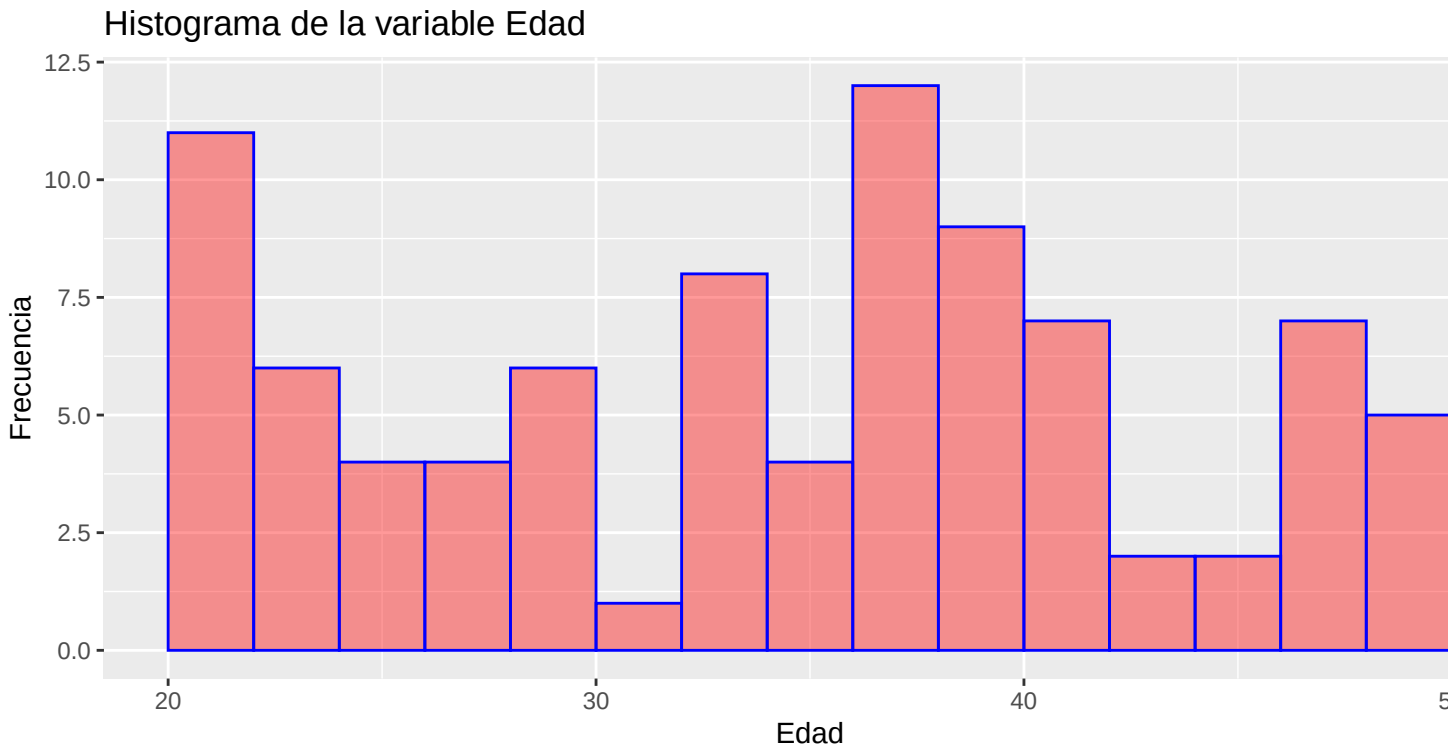
Podemos ajustar el rango de representación del histograma y el ancho de los *bins*.

```
# Podemos ajustar el rango de representación del histograma y el tamaño de los bins
ggplot(data=dataset_hist1, aes(x=AGE)) + geom_histogram(breaks=seq(20, 50, by=2))
```



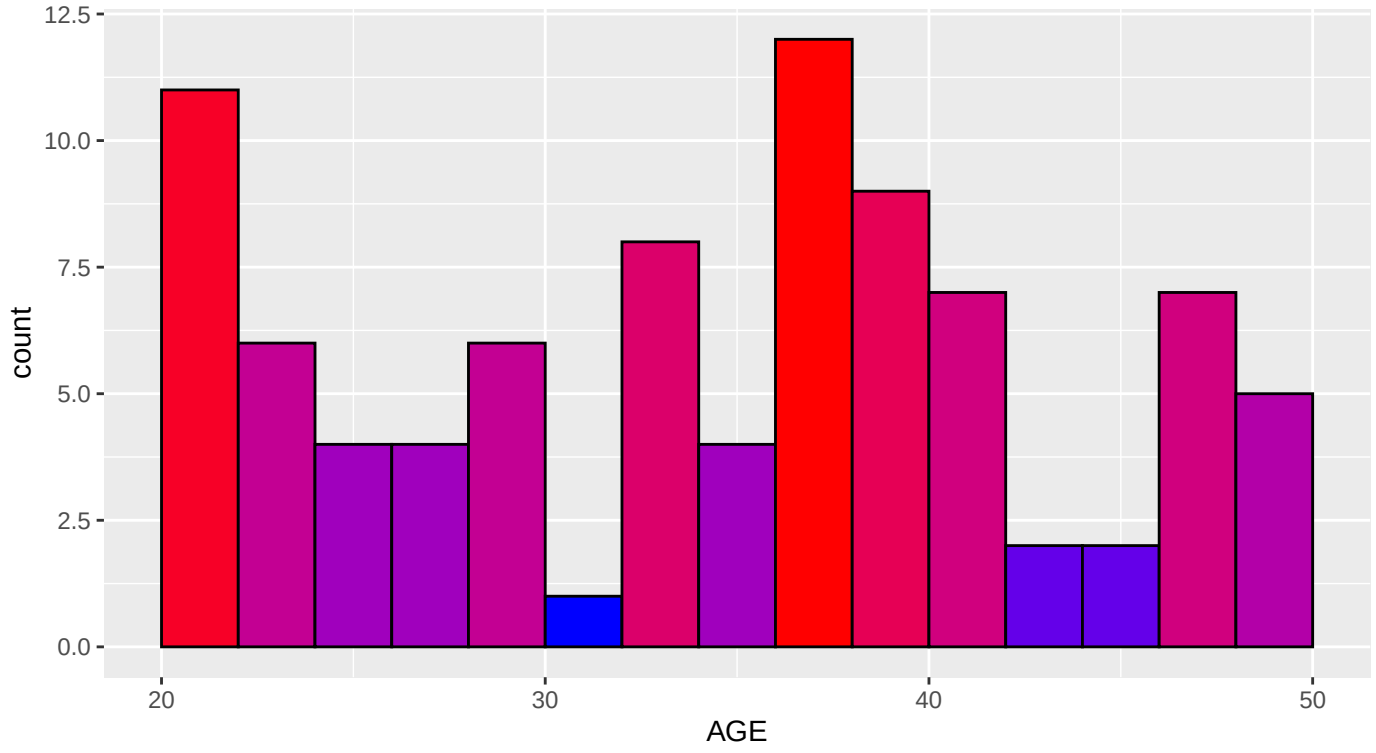
Si queremos ajustar el color del contorno, el relleno y el grado de transparencia, no es un problema para **ggplot2**.

```
ggplot(data=dataset_hist1, aes(x=AGE)) +
  geom_histogram(breaks=seq(20, 50, by=2),
                 col="blue",
                 fill="red",
                 alpha = .4) +
  labs(title="Histograma de la variable Edad", x="Edad", y="Frecuencia")
```



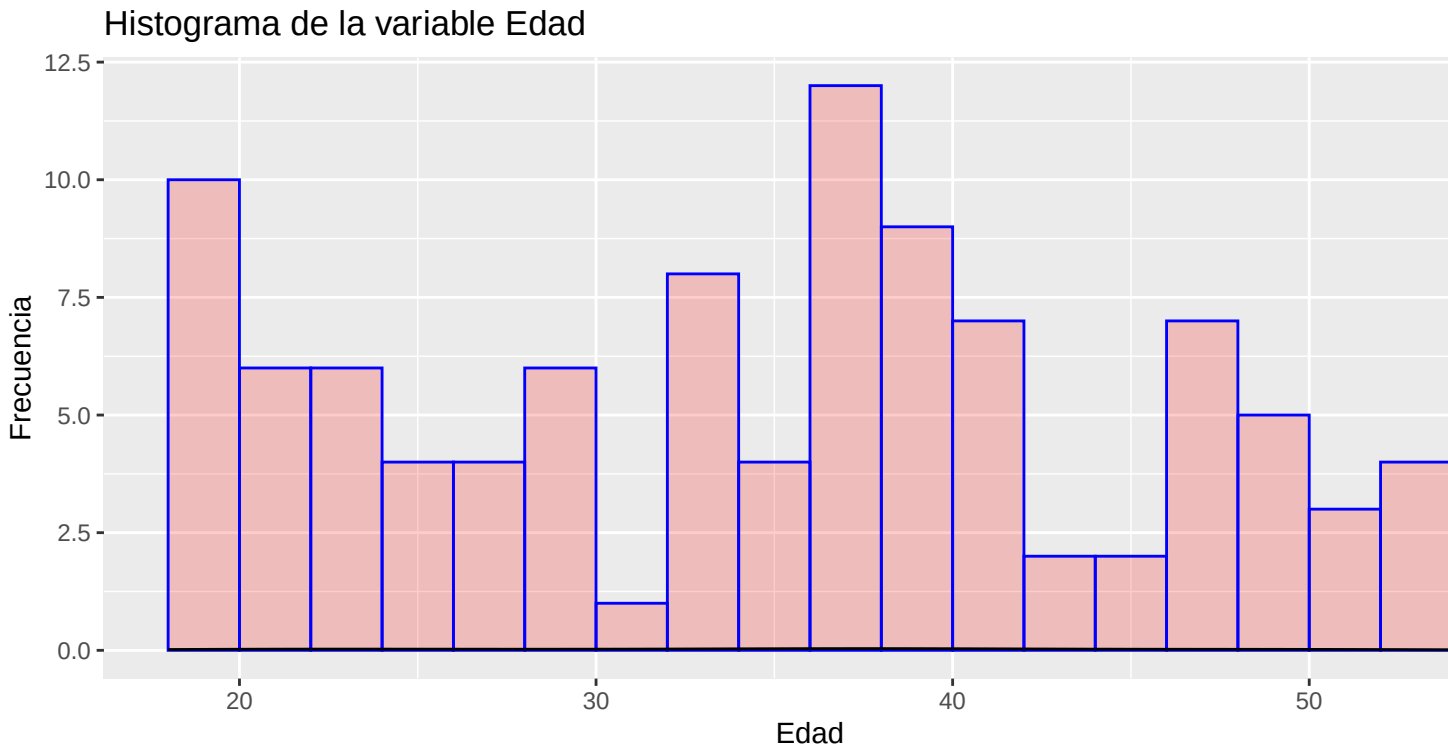
Si queremos que el color de las barras informe sobre la altura de las mismas para enfatizar la información proporcionada por el histograma podemos hacerlo con la ayuda de la función **scale_fill_gradient()**.

```
ggplot(data=dataset_hist1, aes(x=AGE)) +
  geom_histogram(breaks=seq(20, 50, by=2),
                 col="black",
                 aes(fill=after_stat(count))) +
  scale_fill_gradient(low="blue", high="red")
```



Podemos añadir la curva con la estimación de la distribución de datos con `geom_density` Pero, ¿Se observa la función de densidad ? Observa la escala `Y`

```
ggplot(data=dataset_hist1, aes(x=AGE)) +
  geom_histogram(
                 breaks=seq(18, 54, by = 2),
                 col="blue",
                 fill="red",
                 alpha=.2) +
  geom_density(col="black") +
  labs(title="Histograma de la variable Edad", x="Edad", y="Frecuencia")
```

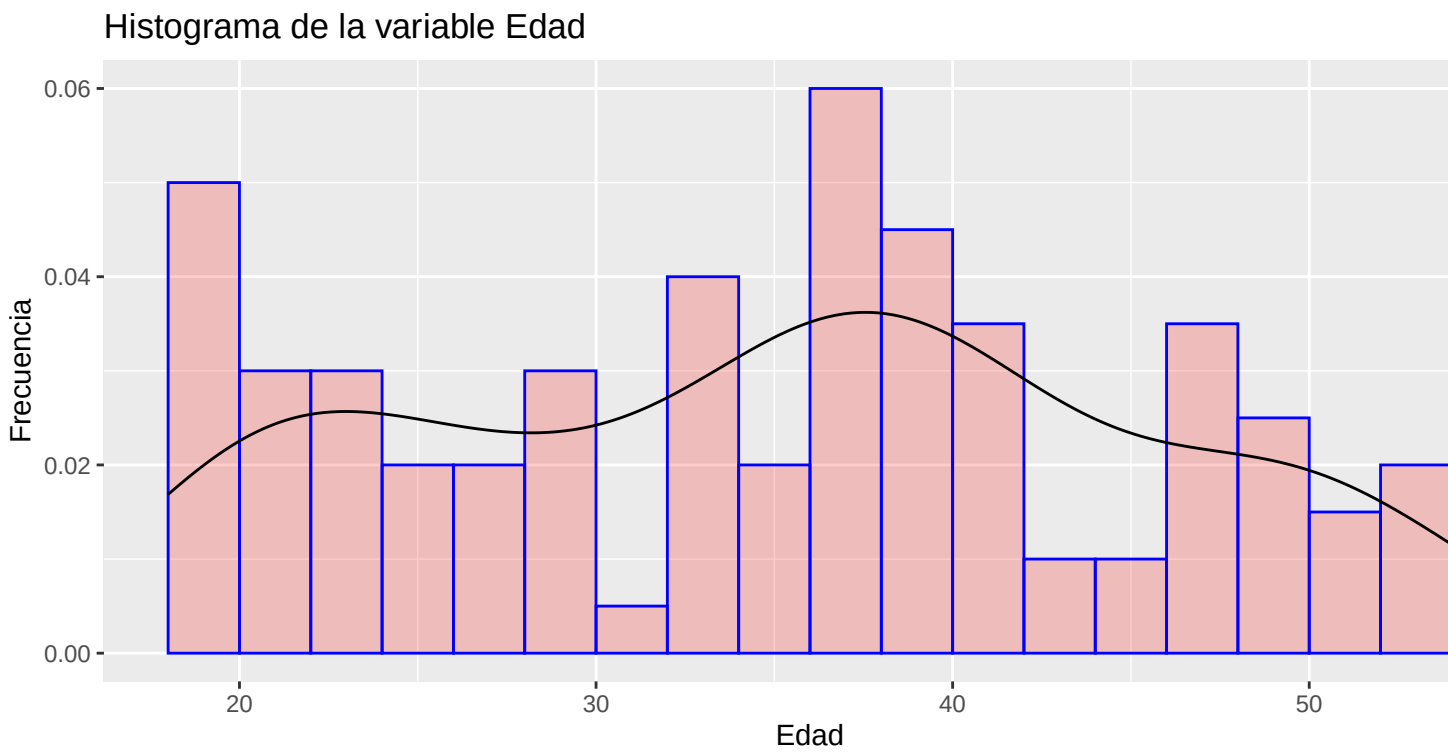


Para que ver el histograma y la función de densidad hemos usado la versión normalizada y =after_stat(density). Variable calculada a realizar el histograma.

```
geom_histogram() uses the same aesthetics as geom_bar(); geom_freqpoly() uses the same aesthetics as ge

Computed variables
These are calculated by the 'stat' part of layers and can be accessed with delayed evaluation.

after_stat(count)
number of points in bin.

after_stat(density)
density of points in bin, scaled to integrate to 1.

after_stat(ncount)
count, scaled to a maximum of 1.

after_stat(ndensity)
density, scaled to a maximum of 1.

after_stat(width)
widths of bins.
```

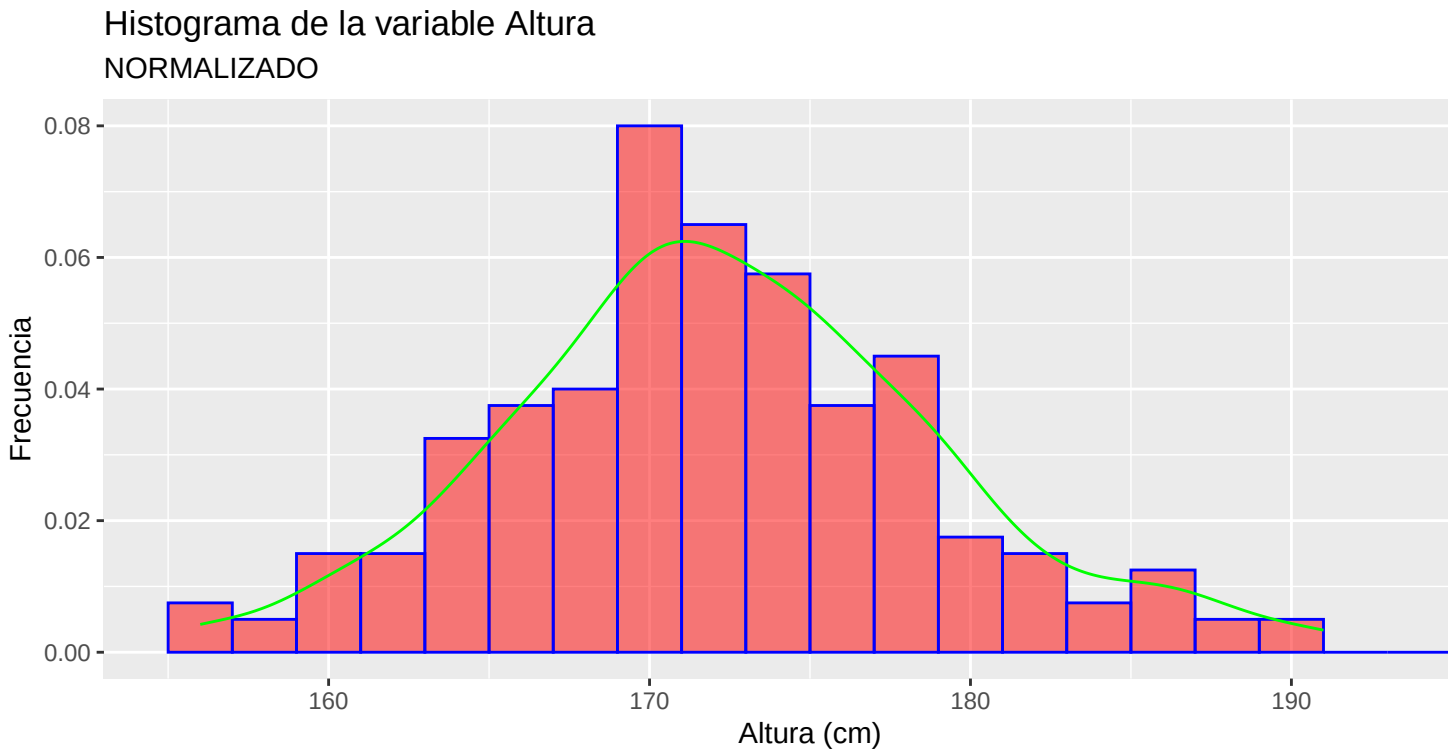
```r
ggplot(data=dataset_hist1, aes(x=AGE)) +
  geom_histogram(aes(y =after_stat(density)),  #OJO USA UNA VARIABLE CALCULADA
                 breaks=seq(18, 54, by = 2),
                 col="blue",
                 fill="red",
                 alpha=.2) +
```

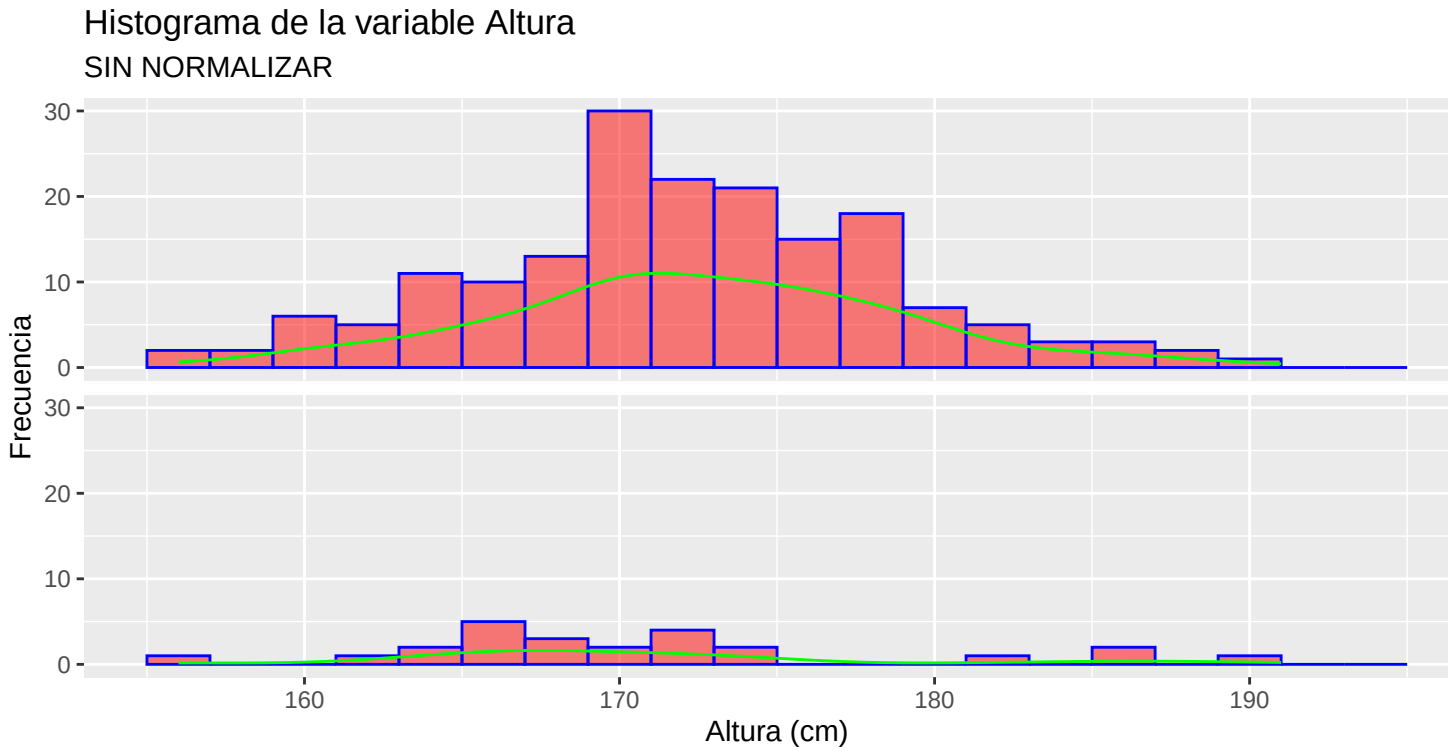
```
  geom_density(col="black") +
  labs(title="Histograma de la variable Edad", x="Edad", y="Frecuencia")
```



Ejercicio 8

- Descarga de la url "http://assets.datacamp.com/blog_assets/chol.txt" el dataset de Datacamp que utilizarás para este ejercicio y cárgalo en un datafrae .
- Representa el histograma de la variable `HEIGHT` del dataframe cargado del fichero anterior.
- Añade título ("Histograma de la variable Altura"), una etiqueta en el eje x ("Altura (cm)") y una etiqueta en el eje y ("Frecuencia").
- Ajusta el rango de representación entre 155 y 195, además fuerza que el histograma tenga 20 *bins*.(usa parámetro `breaks`)
- Haz que las barras del histograma sean de color azul con una grado de transparencia del 50% y el contorno de color rojo.
- Añade la curva de distribución estimada de color verde.
- Representa el mismo histograma del apartado anterior para cada valor de la variable `MORT`, recuerda utilizar la función **facet_grid()**.
- Almacena en formato **pdf** el gráfico resultante en un fichero denominado `Ejercicio8.pdf`.





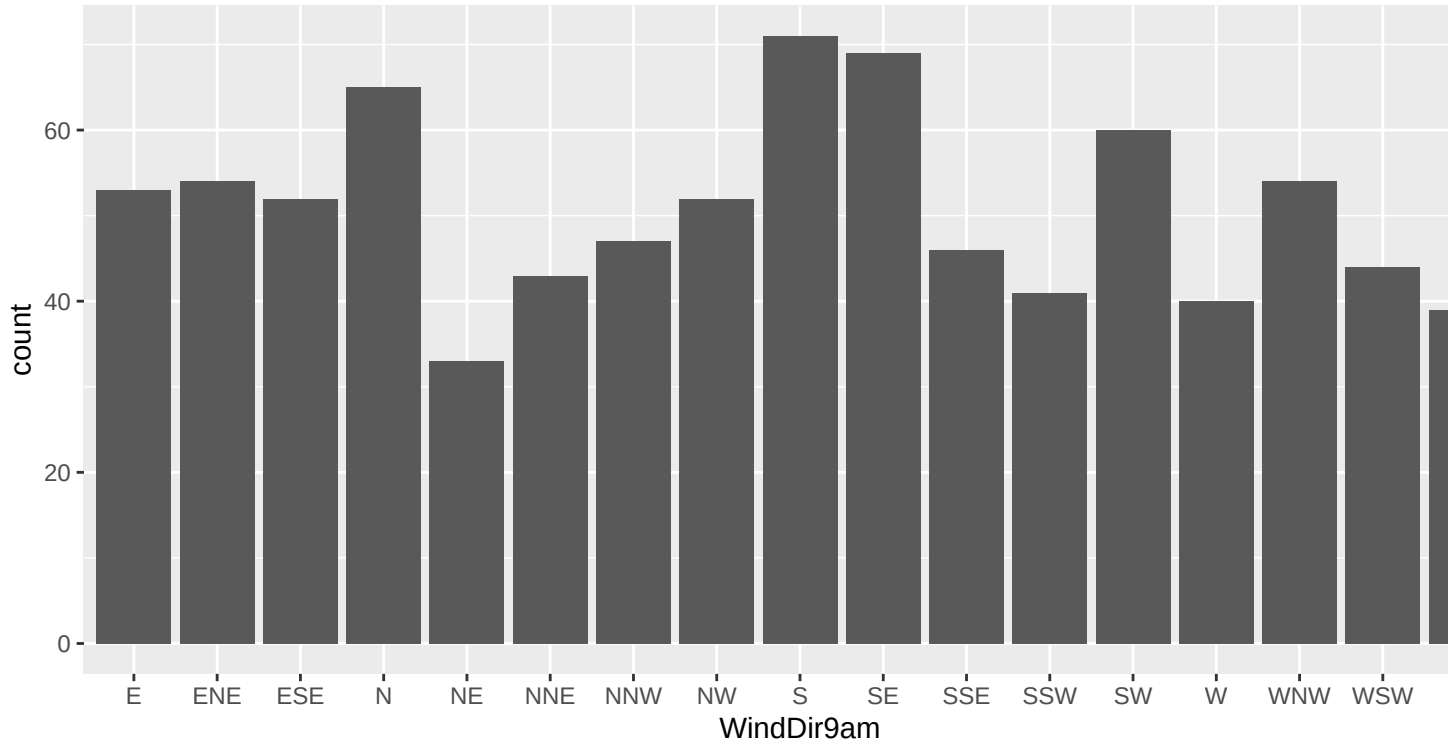
Gráficos de barras con ggplot2

En esta sección vamos a utilizar **ggplot2** para representar gráficos de barras. Para ello vamos a utilizar la función **geom_bar()**. La forma de definir los **aes()** es muy similar a como lo hacíamos con otras funciones

"geom" que ya hemos utilizado, no obstante existen pequeñas diferencias que vamos a tratar de plasmar con varios ejemplos. Empezaremos cargando el fichero `weatherAUS_distributions.RData`.

En primer lugar vamos a presentar un uso de **geom_bar()** para obtener la frecuencia de los diferentes valores que presenta una variable categórica. En este caso vamos a visualizar la frecuencia de los diferentes valores que toma la variable `WinDir9pm`.

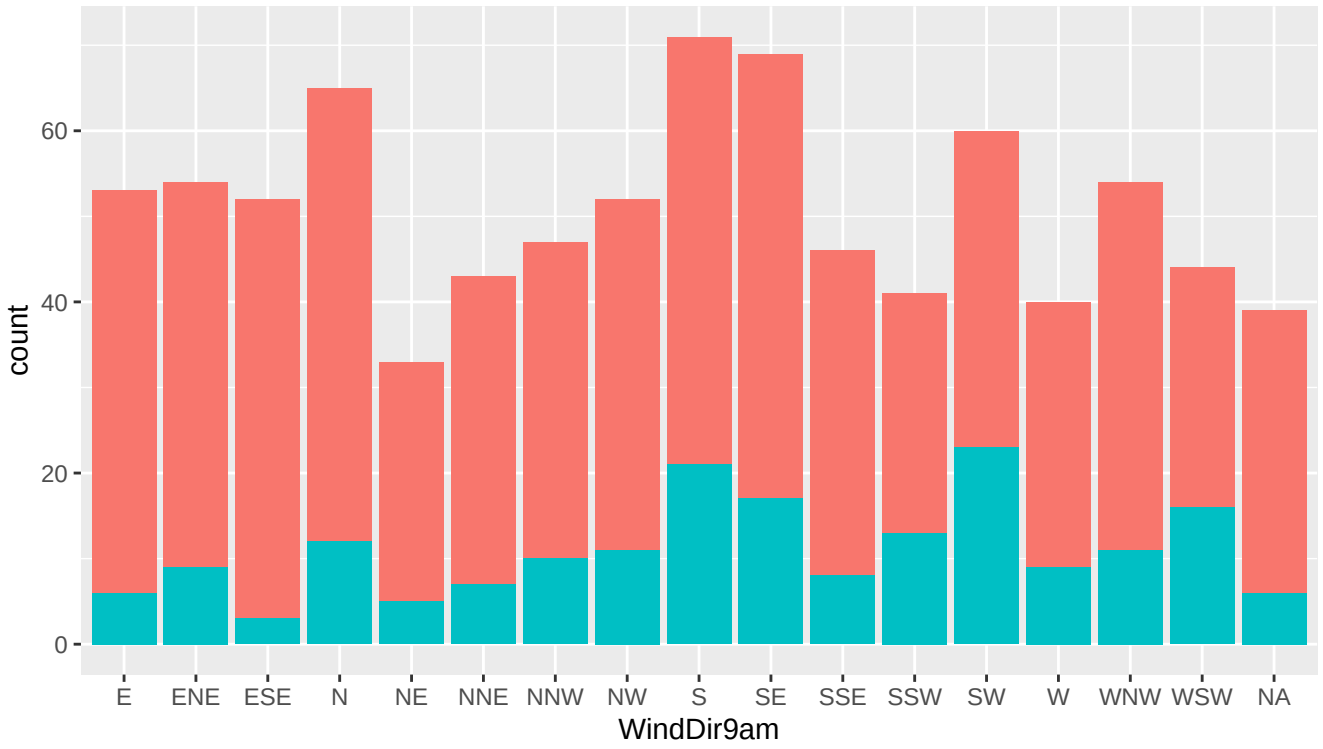
```
load("./data/weatherAUS_distributions.RData")
ggplot(data=ds,aes(x=WindDir9am)) + geom_bar()
```



Podemos incluir más información en este tipo de gráficos. Imaginemos que queremos obtener el mismo gráfico de barras pero para dos grupos de datos proporcionados por la variable `RainToday`.

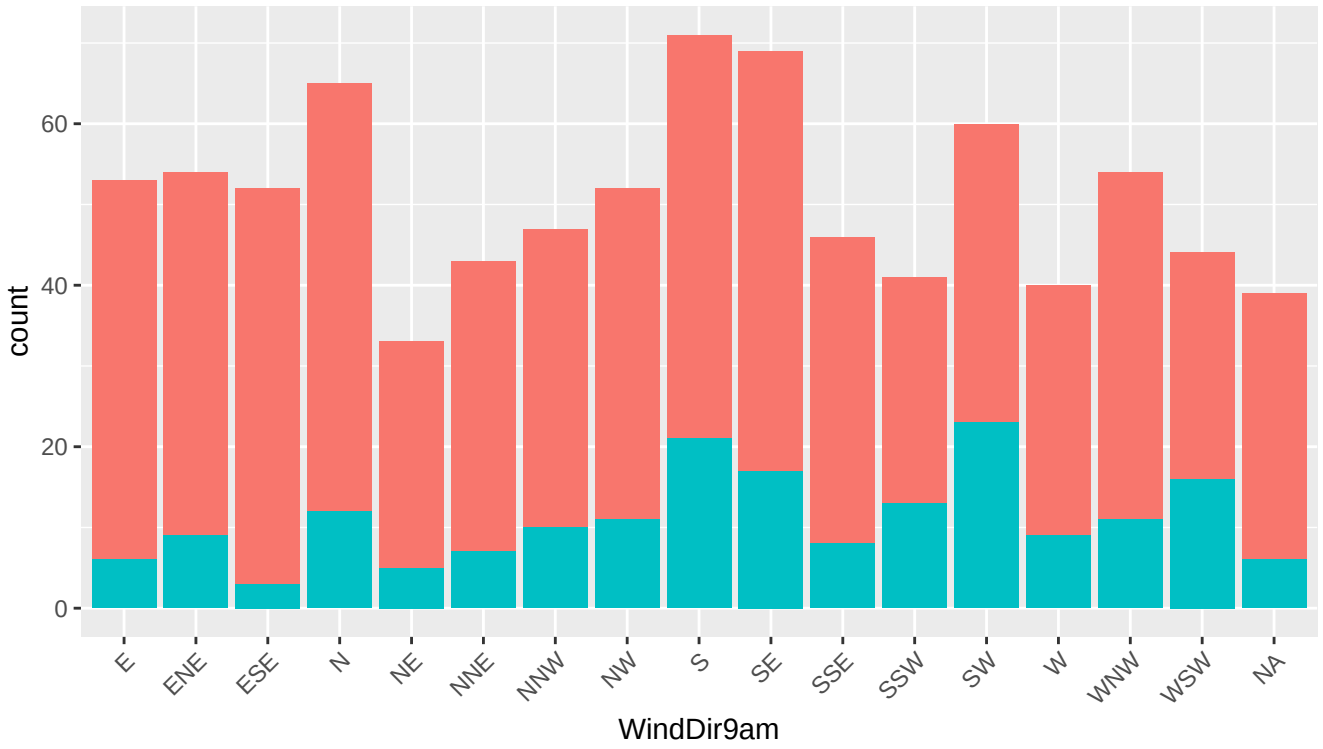
Cuando en una misma posición (`position`) debemos mostrar información diferente, se pueden realizar varios tipos de representación, por ejemplo apilada (`stacked`) como se indica en la siguiente figura. Esta es la opción por defecto.

```
ggplot(data=ds,aes(x=WindDir9am, fill=RainToday)) + geom_bar()
```



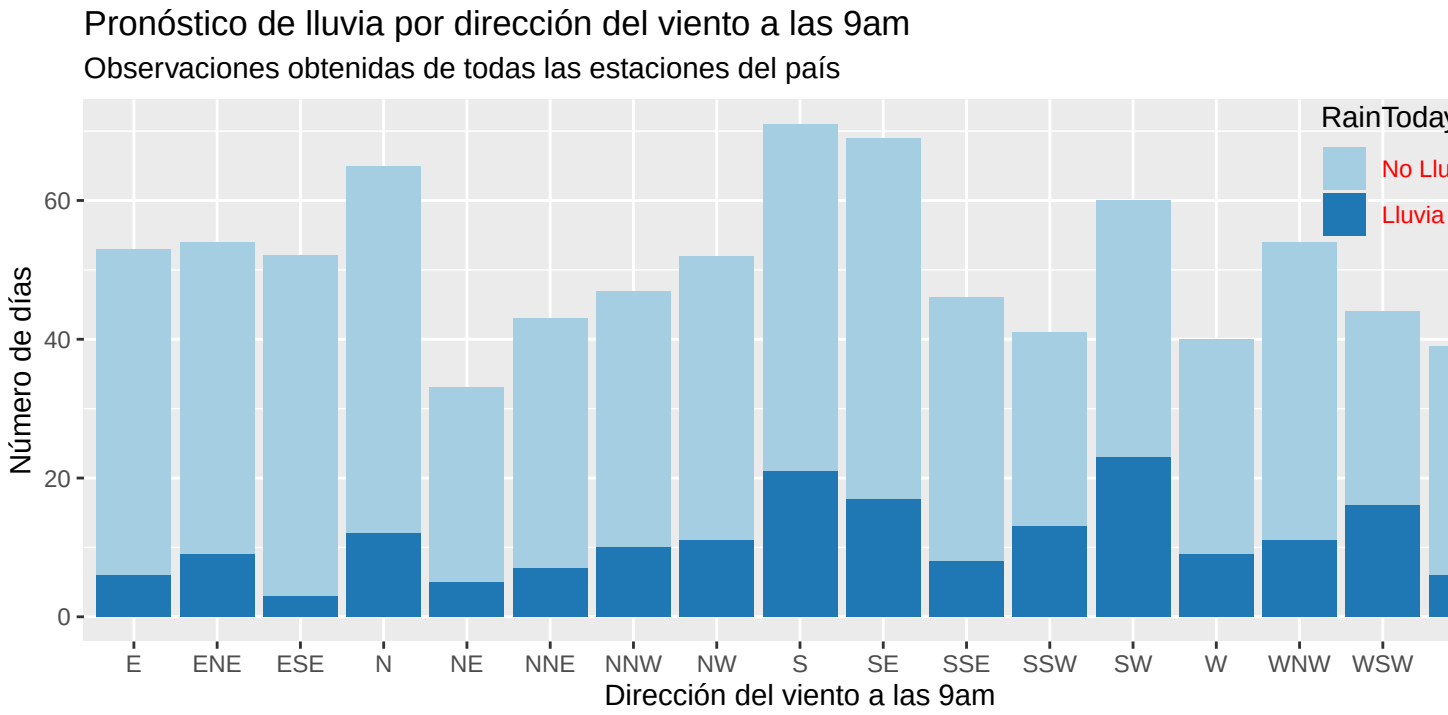
La filosofía de capas permite que estas puedan ser utilizadas en diferentes tipos de gráficos. ¿Recuerdas el ejercicio donde pediamos girar a 45 grados las etiquetas de un gráfico de boxplot? Como puedes imaginar esa capa será igual para un gráfico de barras ...

```
ggplot(data=ds,aes(x=WindDir9am, fill=RainToday)) + geom_bar()+
  theme(axis.text.x=element_text(angle=45, hjust=1))
```



Las opciones para enriquecer los gráficos con **ggplot2** son muy amplias, poco a poco, usando esta librería para hacer gráficos te irás familiarizando con ellas.

```
library('RColorBrewer')
blues2 <- brewer.pal(4, "Paired")[1:2]

ggplot(data=ds, aes(x=WindDir9am, fill=RainToday)) + geom_bar() +
  scale_fill_manual(values=blues2,labels = c("No Lluvia", "Lluvia")) +
  theme(legend.position = c(.92, .87),
        legend.title=element_text(colour="black"),
        legend.text=element_text(colour="red"),
        legend.background = element_rect(fill="transparent")) +
  labs(title = "Pronóstico de lluvia por dirección del viento a las 9am",
     subtitle = "Observaciones obtenidas de todas las estaciones del país",
     caption  = "Fuente: Departamento Australiano de meteorología",
     x        = "Dirección del viento a las 9am",
     y        = "Número de días",
     fill     = "RainToday")
```

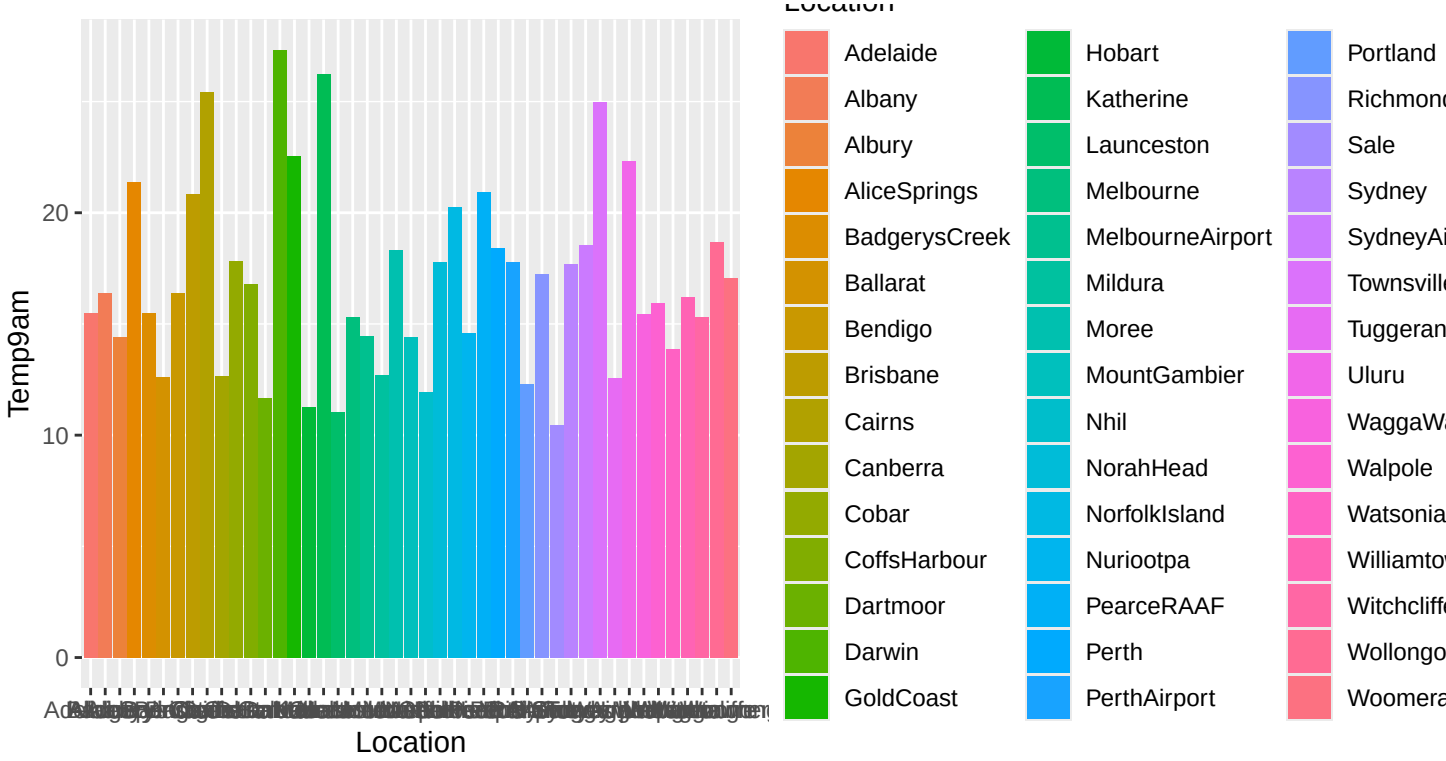


Fuente: Departamento Australiano de meteor

Igual no lo has percibido, pero la función **geom_bar()** de manera implícita, hace cálculos a partir de los datos para construir la representación. Por ejemplo, en los gráficos anteriores se ha calculado las frecuencias (cuentas) de cada valor de la dirección del viento. Estos cálculos se explicitan a través del argumento `stats` de las funciones `geom`. En los casos anteriores la opción por defecto es `stat="count"` que calcula frecuencias. Si el recuento ya está hecho y disponemos del número de repeticiones de cada categoría usaremos **geom_col**, o bien, **geom_bar(stat='identity')**.

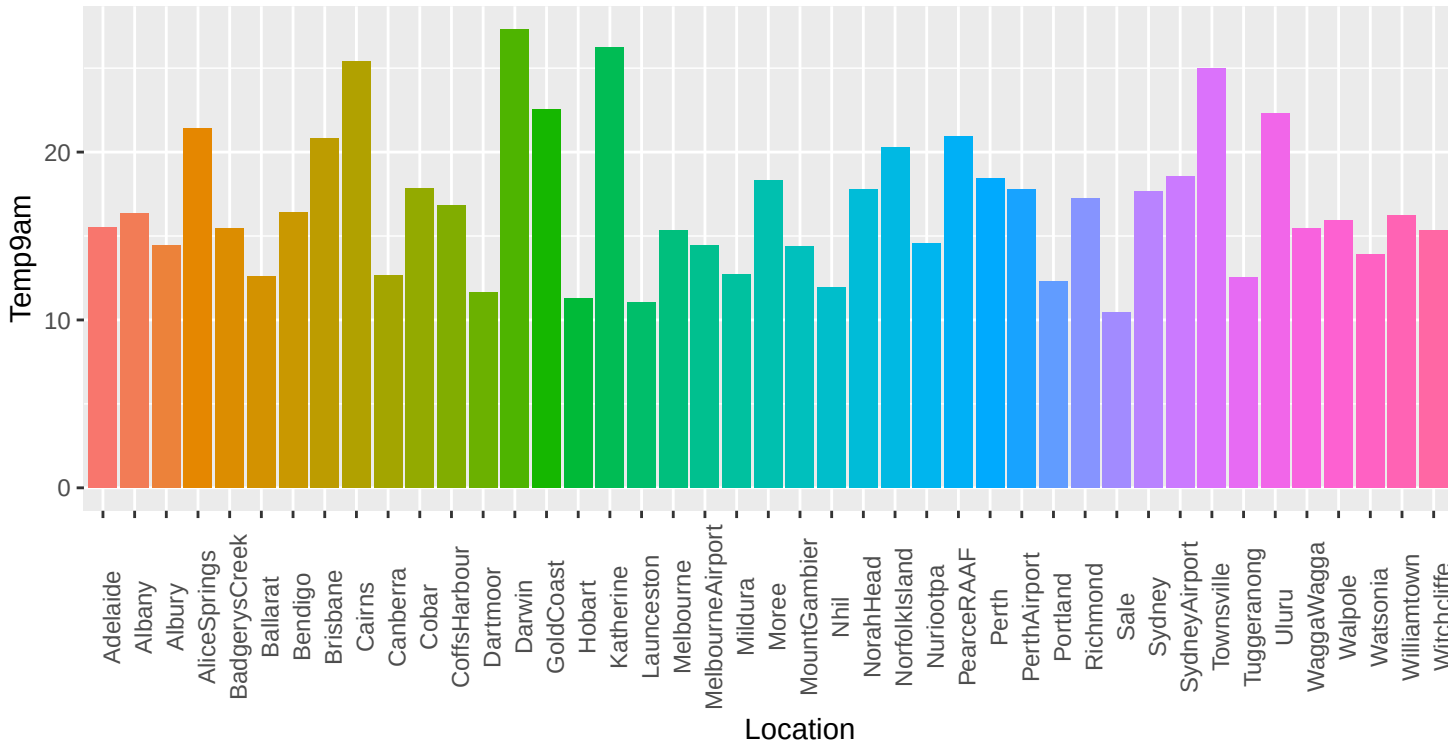
Vamos a ver un ejemplo de uso de otro valor del argumento `stat` . Queremos representar el valor medio de la variable `Temp9am` en función de la variable `Location`. La variable `Temp9am` es numérica (no categórica) y no tendría demasiado sentido utilizar el `stat="count"`.

```
ggplot(data=ds,aes(Location, Temp9am, fill=Location)) +
  geom_bar(stat="summary", fun="mean")
```



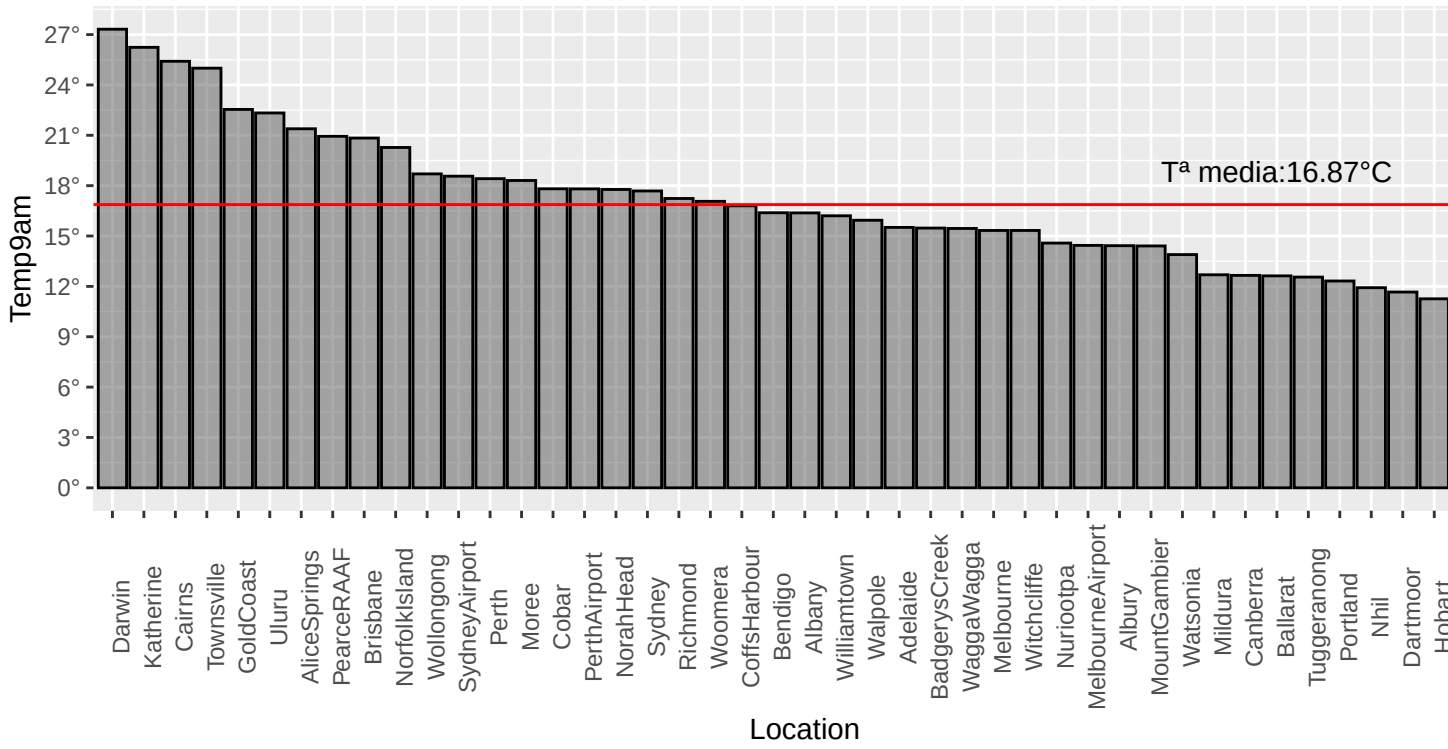
No se ven bien los valores de x en el gráfico... Quitamos la leyenda y giramos las etiquetas de x a 90 grados.

```
ggplot(data=ds,aes(Location, Temp9am, fill=Location)) + #Muy colorido pero fill=Location es poco útil
  geom_bar(stat="summary", fun="mean") +
  theme(legend.position="none") +
  theme(axis.text.x=element_text(angle=90)) # probar la opción element_text(angle=90,hjust=1)
```



Podemos mostrar la gráfica ordenando las ciudades (`Location`) según el valor medio (`mean`) de la temperatura (`Temp9am`) en orden decreciente, usando la función **reorder**.
Observa los otros elementos del gráfico y su efecto en el resultado.

```
#reorder: reordena el factor(Location) segun el resultado de aplicar FUN a Temp9a
ggplot(data=ds,aes(x=reorder(Location,Temp9am,FUN=mean,decreasing=TRUE), y=Temp9am)) +
  geom_bar(stat="summary", fun="mean",color='black',alpha=0.5) +
  geom_hline(yintercept = mean(ds$Temp9am),color='red')+
  #annotate permite añadir etiquetas fijas y puntuales en la coordenadas especificadas de un gráfico. E
  annotate("text", x='Ballarat', y=2+mean(ds$Temp9am), label=paste0('Tª media:',paste0(as.character(rou
  theme(legend.position="none") +
  theme(axis.text.x=element_text(angle=90))+
  labs(x='Location')+
  scale_y_continuous(breaks = seq(0,30,by=3),labels = paste0(seq(0,30,by=3),'°'))
```


Temp9am
27°
24°
21°
18°
15°
12°
9°
6°
3°
0°
Tª media:16.87°C
Darwin
Katherine
Cairns
Townsville
GoldCoast
Uluru
AliceSprings
PearceRAAF
Brisbane
NorfolkIsland
Wollongong
SydneyAirport
Perth
Moree
Cobar
PerthAirport
NorahHead
Sydney
Richmond
Woomera
CoffsHarbour
Bendigo
Albany
Williamtown
Walpole
Adelaide
BadgerysCreek
WaggaWagga
Melbourne
Witchcliffe
Nuriootpa
MelbourneAirport
Albury
MountGambier
Watsonia
Mildura
Canberra
Ballarat
Tuggeranong
Portland
Nhil
Dartmoor
Location