

Gráficas

Josman

21 de octubre de 2014

Introducción

En este documento explicaremos nociones básicas para el diseño de gráficas en R a través del paquete ggplot2, explicaremos lo elemental en cada tipo de gráfica y desarrollaremos la intuición detrás del código.

```
library(dplyr)
library(tidyr)
library(ggplot2)
library(RColorBrewer) # Paletas de colores
library(gcookbook) # De aquí sacaremos varios datos para graficar
```

El paquete ggplot2 se llama a sí mismo como “una implementación de la gramática de las gráficas”. Este paquete nos permite crear gráficas mapeando capas paso por paso de múltiples bases de datos. Para el diseño de las gráficas usaremos “aesthetics” lo que describe como las variables en los datos son mapeadas a través de características visuales y geométricas.

Gráficas de barras

Digamos que tenemos una base de datos como la siguiente:

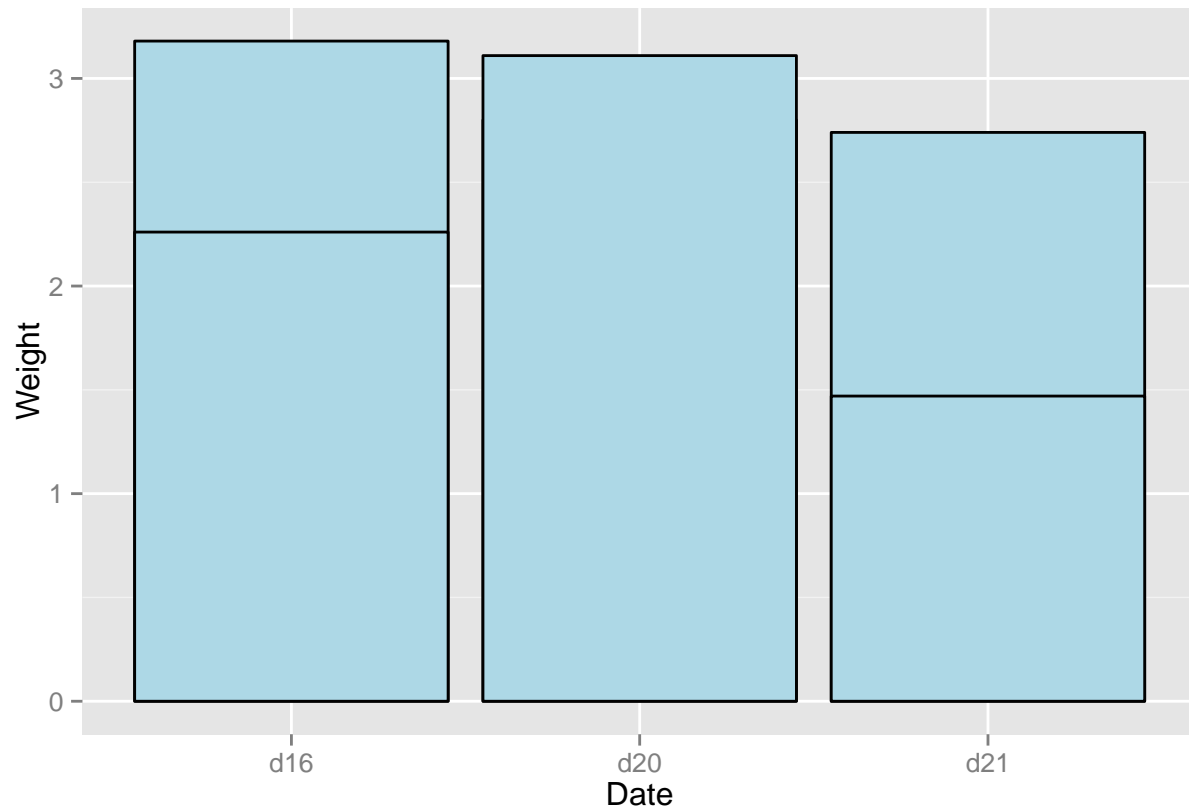
```
cabbage_exp
```

##	Cultivar	Date	Weight	sd	n	se
## 1	c39	d16	3.18	0.9566144	10	0.30250803
## 2	c39	d20	2.80	0.2788867	10	0.08819171
## 3	c39	d21	2.74	0.9834181	10	0.31098410
## 4	c52	d16	2.26	0.4452215	10	0.14079141
## 5	c52	d20	3.11	0.7908505	10	0.25008887
## 6	c52	d21	1.47	0.2110819	10	0.06674995

Queremos graficar el peso obtenido para cada fecha. Para ello usaremos una gráfica de barras. Detallaremos cada capa que vayamos agregando en la gráfica. Mientras vayamos avanzando en el tema, detallaremos menos la explicación del código, pues se sigue la misma lógica que en los primeros ejemplos.

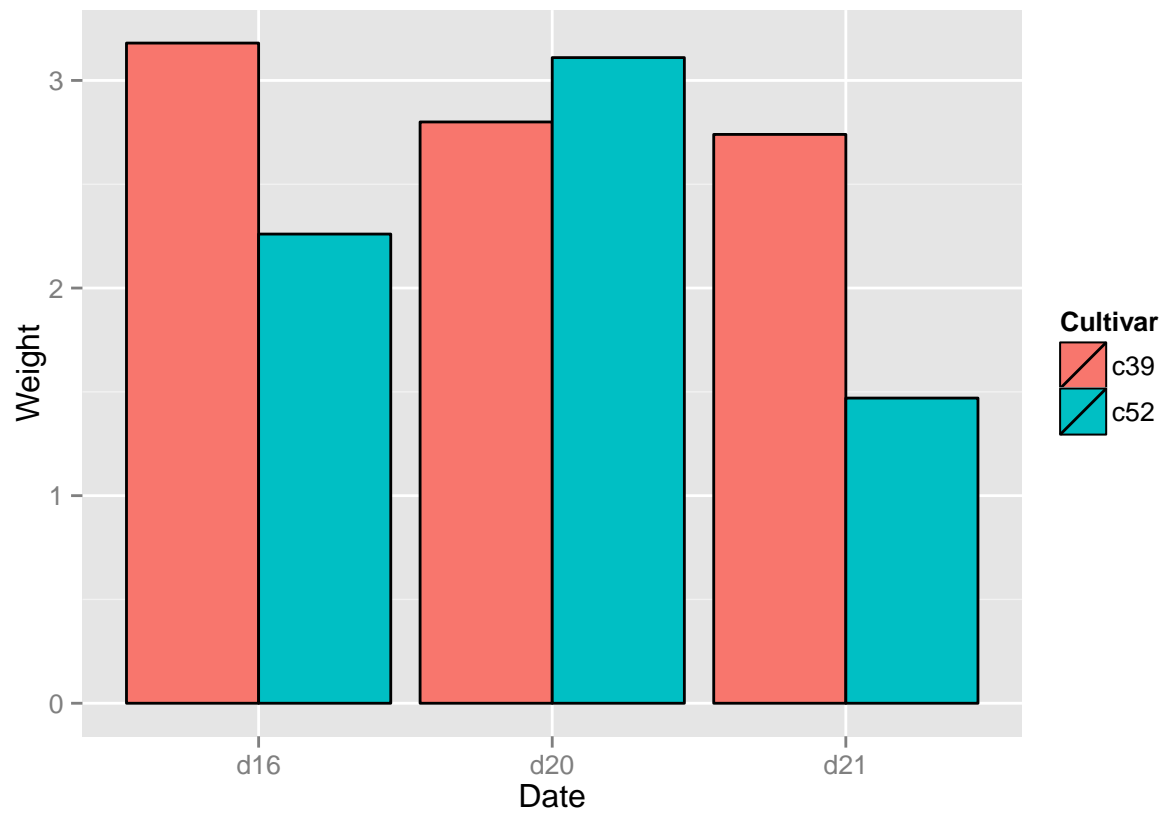
```
# La primera capa crea un objeto 'ggplot', indica la base de datos principal de la gráfica y que variables
plot <- ggplot(data = cabbage_exp, aes(x = Date, y = Weight))

# Este objeto solo conoce los datos, ahora indicaremos cómo graficarlos.
# Vamos agregando capas con '+'.
# geom_bar() indica que agregamos una capa con una gráfica de barras
# position = "dodge" sirve para que las barras queden separadas
# stat = "identity" indica que esta no es una gráfica de frecuencias, sino de dos variables
# fill indica el color de relleno
# col indica el color del contorno
plot <- plot + geom_bar(position = "dodge", stat = "identity", fill = "lightblue", col = "black")
plot
```

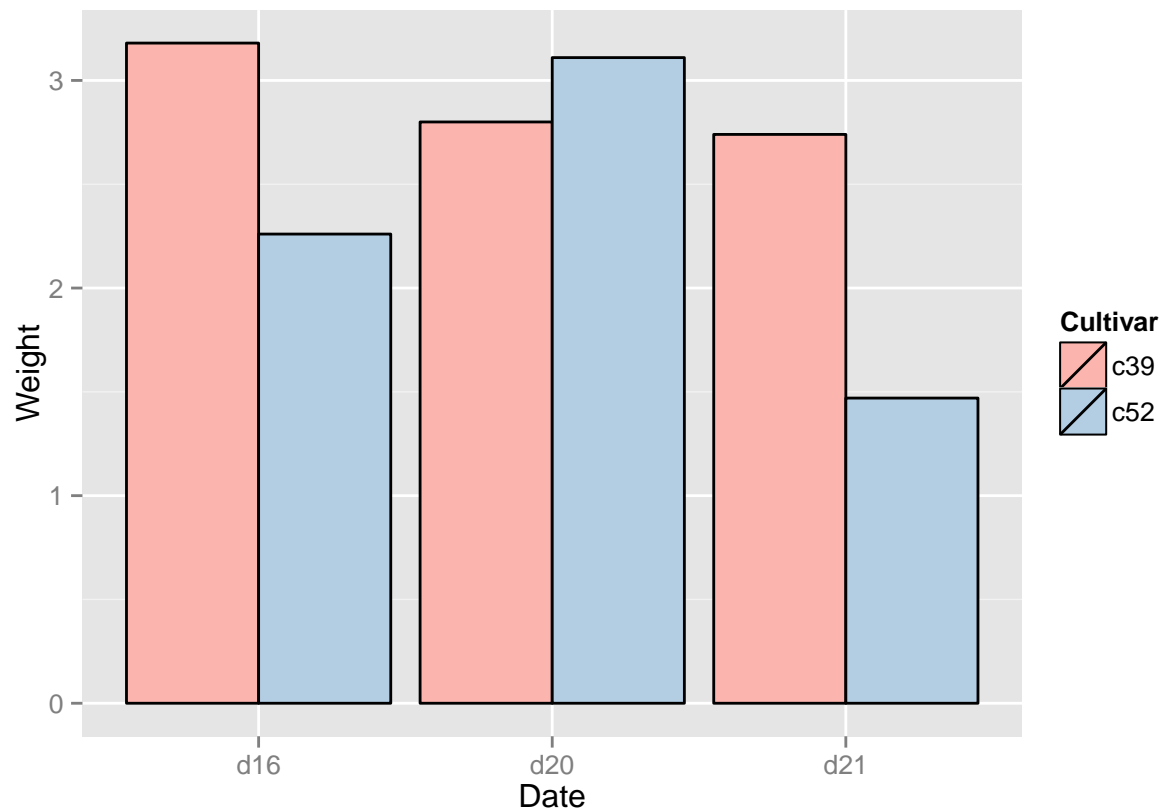


Digamos que ahora queremos ver como afecta el factor 'Cultivar' a la variable 'Weight' en cada fecha. Para ello podemos hacer gráficas de barras agrupadas.

```
# Ahora a la primera capa le estamos indicando que el color de relleno va a ser determinado por el factor  
plot <- ggplot(data = cabbage_exp, aes(x = Date, y = Weight, fill = Cultivar))  
  
# A la segunda capa le quitamos el indicador del color de relleno pues ya está determinado por la variable  
plot <- plot + geom_bar(position = "dodge", stat = "identity", col = "black")  
plot
```

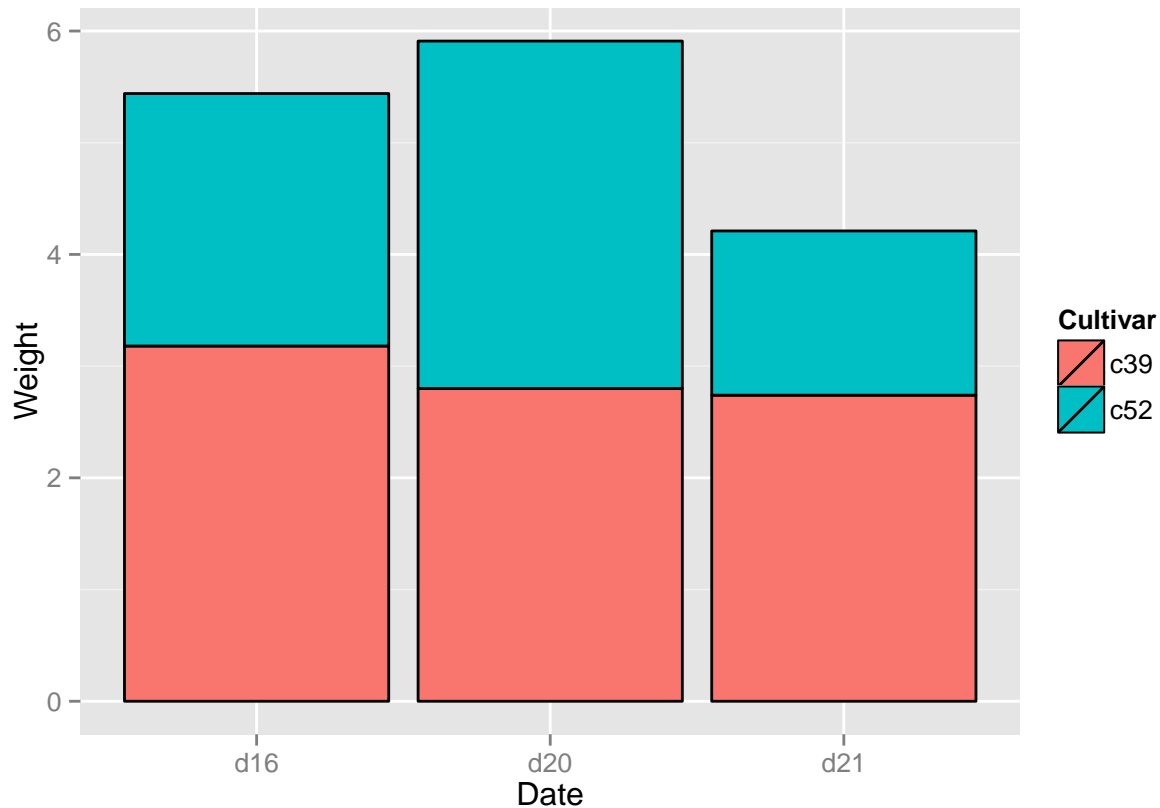


```
# Indicamos otra paleta de colores del paquete RColorBrewer
plot <- plot + scale_fill_brewer(palette = "Pastel1")
plot
```



Si quisieramos graficar las barras unas encima de otras, podemos quitar el indicador 'position = "dodge"':

```
plot <- ggplot(data = cabbage_exp, aes(x = Date, y = Weight, fill = Cultivar)) +  
  geom_bar(stat = "identity", col = "black")  
plot
```



Gráficas de frecuencias

A veces queremos representar la frecuencia con la que se repiten los datos en una muestra aleatoria. Para ello podemos usar gráficas de barras similares a las anteriores, pero ahora cambiaremos el tipo de 'stat'. Usaremos la base de datos 'diamonds' para ejemplificar el caso. Haremos una gráfica de frecuencias de la variable 'cut'. Primero observemos un poco los datos.

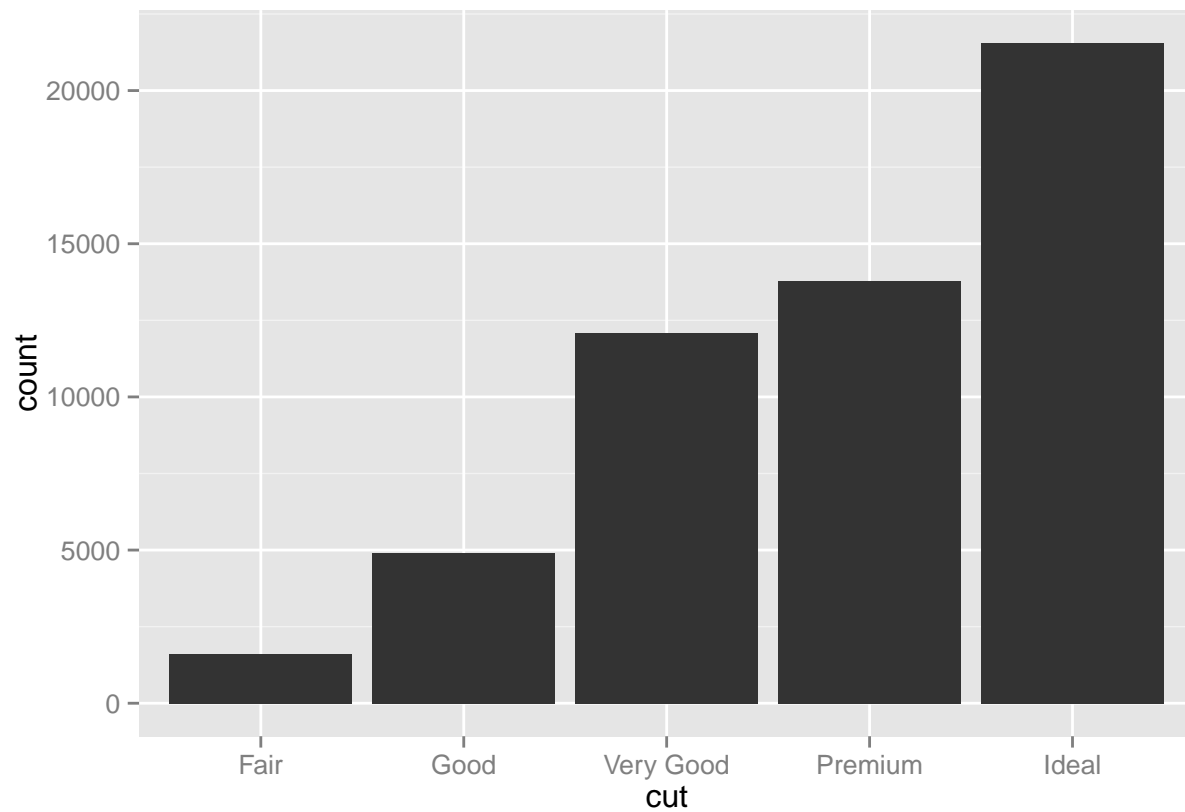
```
head(diamonds)
```

```
##   carat      cut color clarity depth table price     x     y     z
## 1  0.23    Ideal     E    SI2   61.5    55   326  3.95  3.98  2.43
## 2  0.21  Premium     E    SI1   59.8    61   326  3.89  3.84  2.31
## 3  0.23     Good     E    VS1   56.9    65   327  4.05  4.07  2.31
## 4  0.29  Premium     I    VS2   62.4    58   334  4.20  4.23  2.63
## 5  0.31     Good     J    SI2   63.3    58   335  4.34  4.35  2.75
## 6  0.24 Very Good     J   VVS2   62.8    57   336  3.94  3.96  2.48
```

Ahora grafiquemos:

```
# Ahora nuestra gráfica solo necesita los datos de una variable (cut)
plot <- ggplot(data = diamonds, aes(x = cut))
```

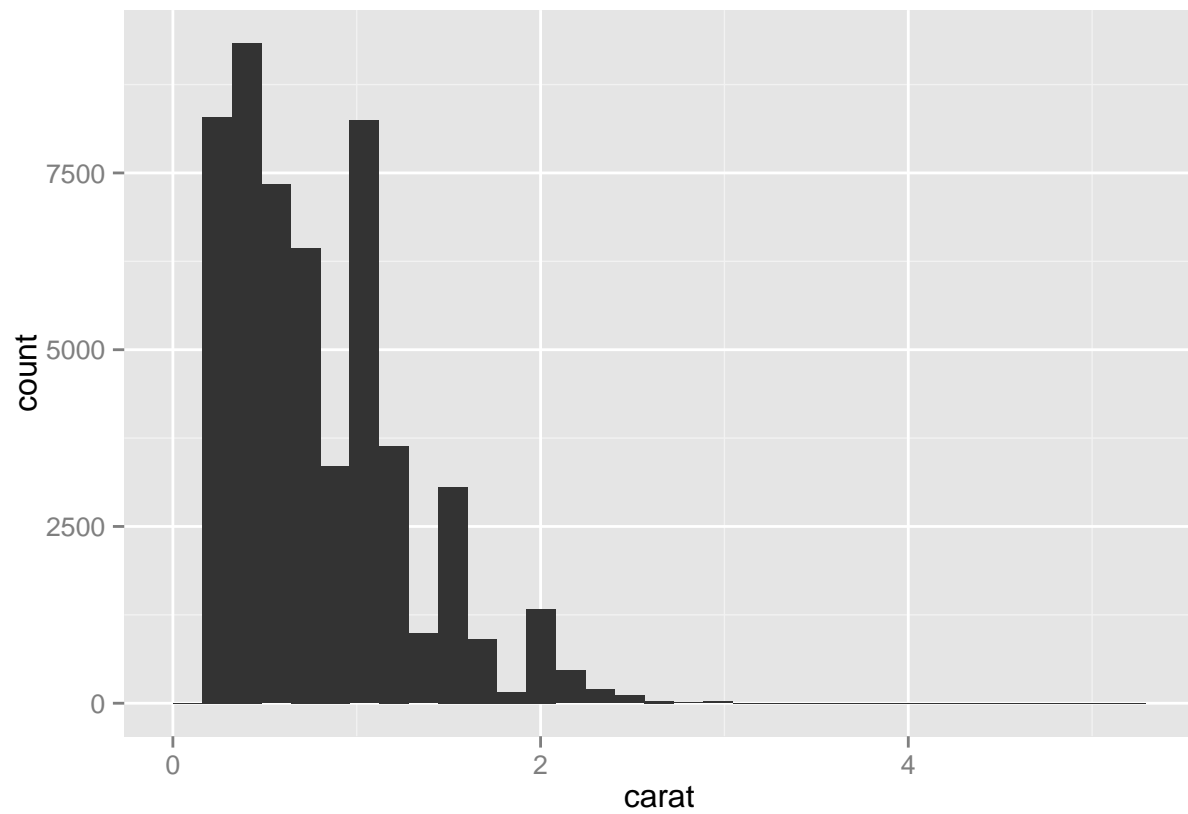
```
# Cuando graficamos frecuencias indicamos stat = "bin". Éste es el stat por default, así que si dejamos
plot + geom_bar()
```



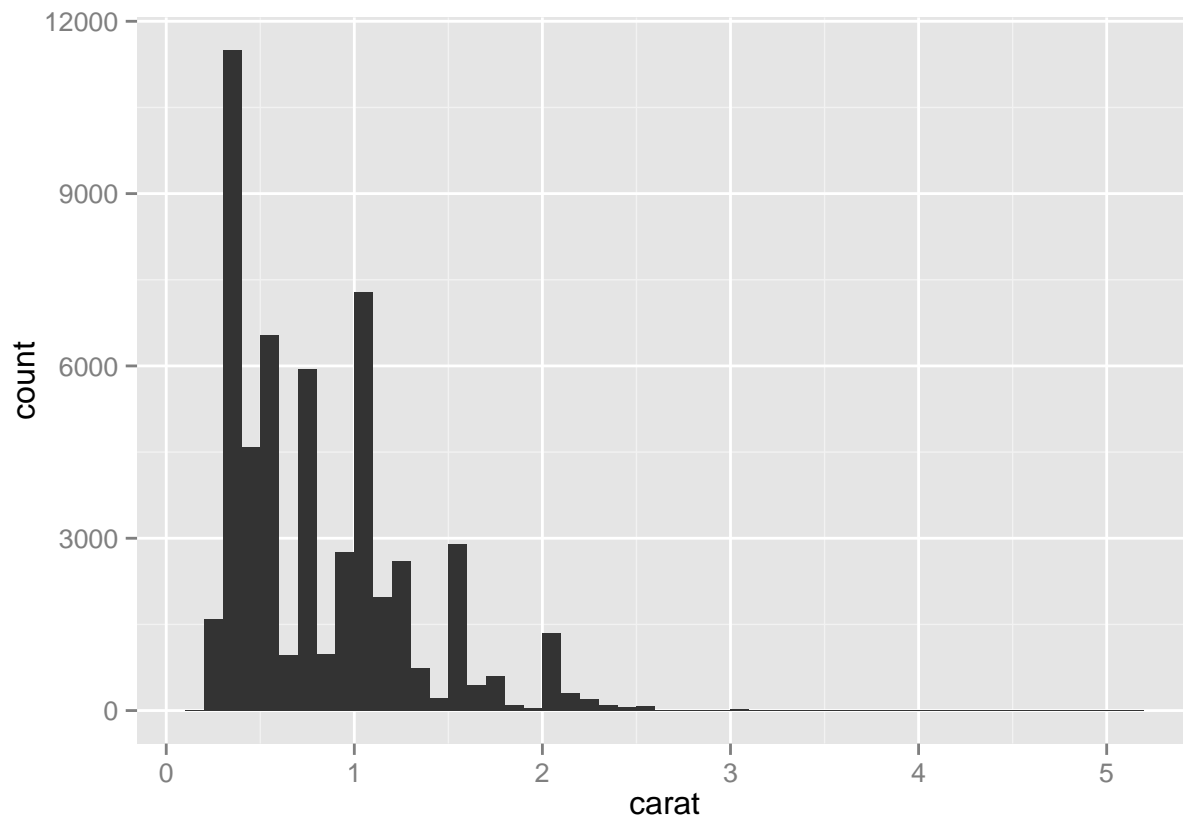
El caso anterior es para una variable categórica. También podemos graficar para una variable continua, como es el caso de 'carat'.

```
ggplot(data = diamonds, aes(x = carat)) + geom_bar()
```

```
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
```



```
# Podemos ajustar el ancho de las barras indicando 'width' en geom_bar()
ggplot(data = diamonds, aes(x = carat)) + geom_bar(binwidth = 0.1)
```



Si estuviéramos en el caso de `stat = "identity"`, usamos `'width'` en lugar de `'binwidth'`

Gráficas de líneas

Existen muchos datos que su mejor representación se da a través de gráficas de líneas, por ejemplo, las series de tiempo. Las gráficas de líneas siguen la misma lógica que las gráficas de barras, solo que cambiaremos el tipo de `'geom'`.

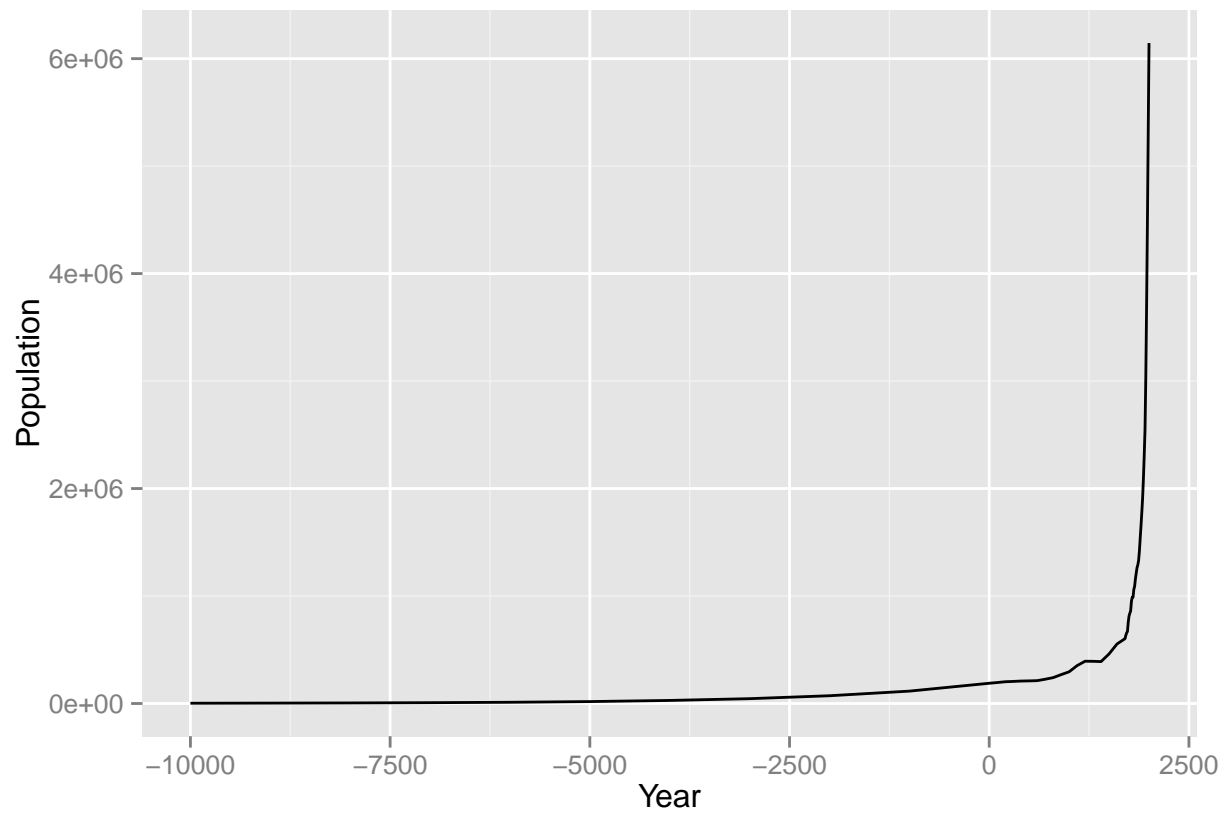
Veamos los siguientes datos:

```
head(worldpop)
```

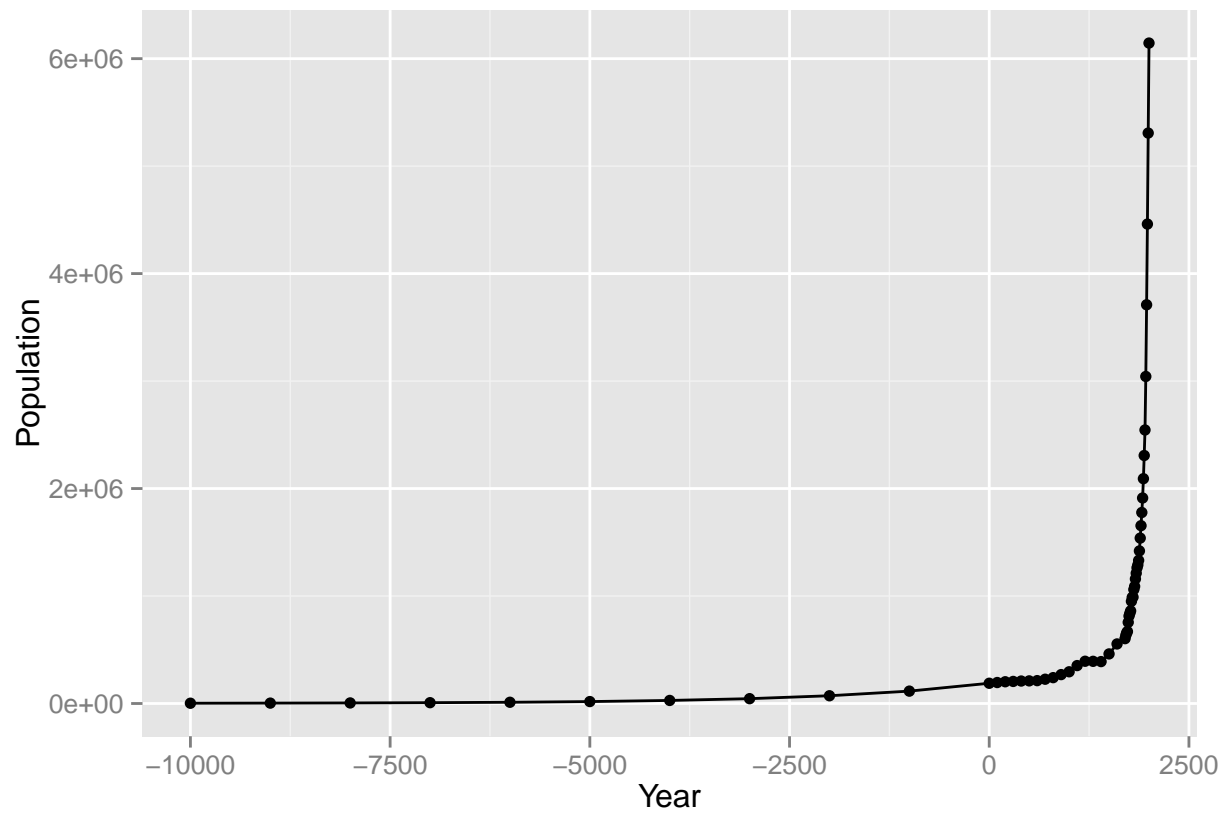
```
##      Year Population
## 1 -10000      2431
## 2  -9000      3564
## 3  -8000      5136
## 4  -7000      7562
## 5  -6000     11461
## 6  -5000     17920
```

Vemos que es una base de datos de series de tiempo que muestran el tamaño de la población de un determinado lugar a través de los años. Usaremos una gráfica de barras para representar visualmente los datos:

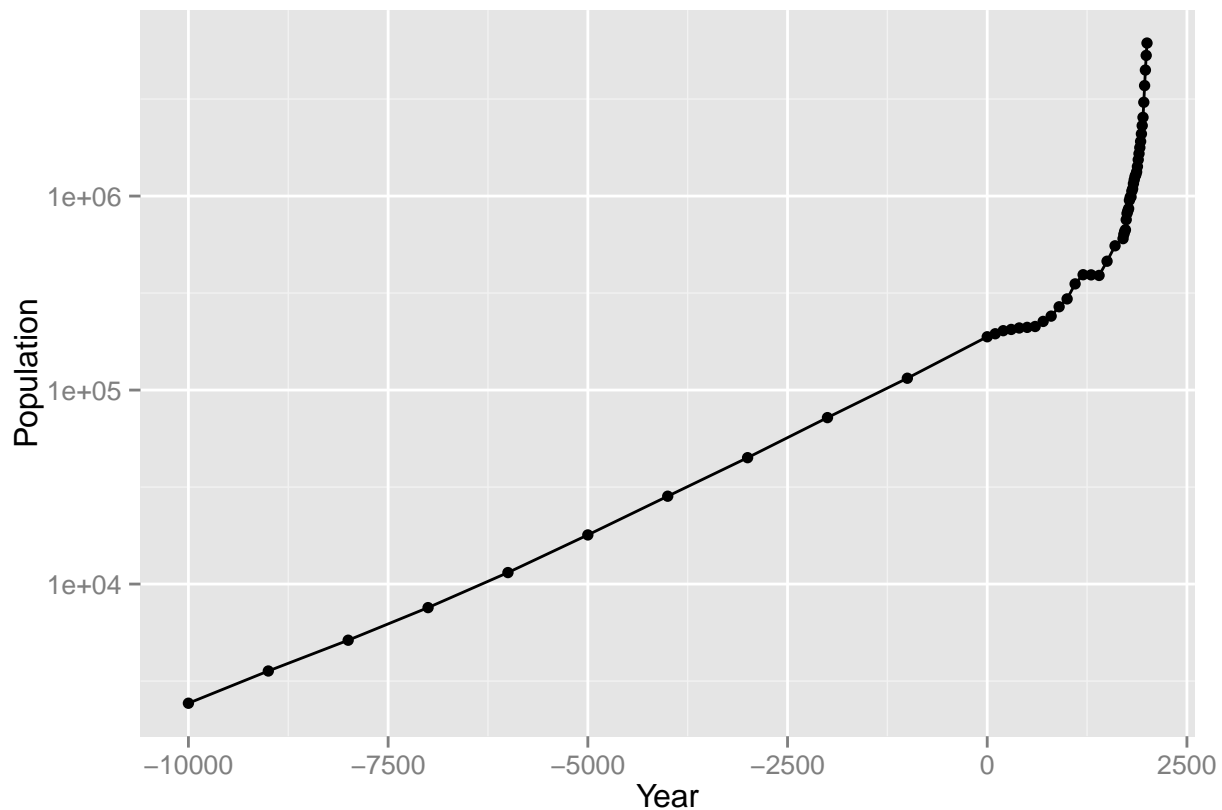
```
plot <- ggplot(data = worldpop, aes(x = Year, y = Population)) + geom_line()
plot
```

```
# Para indicar la muestra de datos agregaremos puntos en cada pareja de datos de nuestra muestra  
plot <- plot + geom_point()  
plot
```



```
# Muchas veces en este tipo de datos es mejor usar escala logarítmica  
plot <- plot + scale_y_log10()  
plot
```



Ahora pensemos que tenemos unos datos como los siguientes:

```
head(ToothGrowth)
```

```
##      len supp dose
## 1  4.2   VC  0.5
## 2 11.5   VC  0.5
## 3  7.3   VC  0.5
## 4  5.8   VC  0.5
## 5  6.4   VC  0.5
## 6 10.0   VC  0.5
```

Tenemos una muestra de varios tamaños de dientes clasificados por los factores 'supp' y 'dose'. A nosotros nos interesa representar la dosis que se debe aplicar según el tamaño de los dientes. Para eso no nos sirve graficar toda la dispersión de los datos, sino un resumen de ellos. Por lo cual antes de graficar tenemos que manipular los datos para que nos sean útiles:

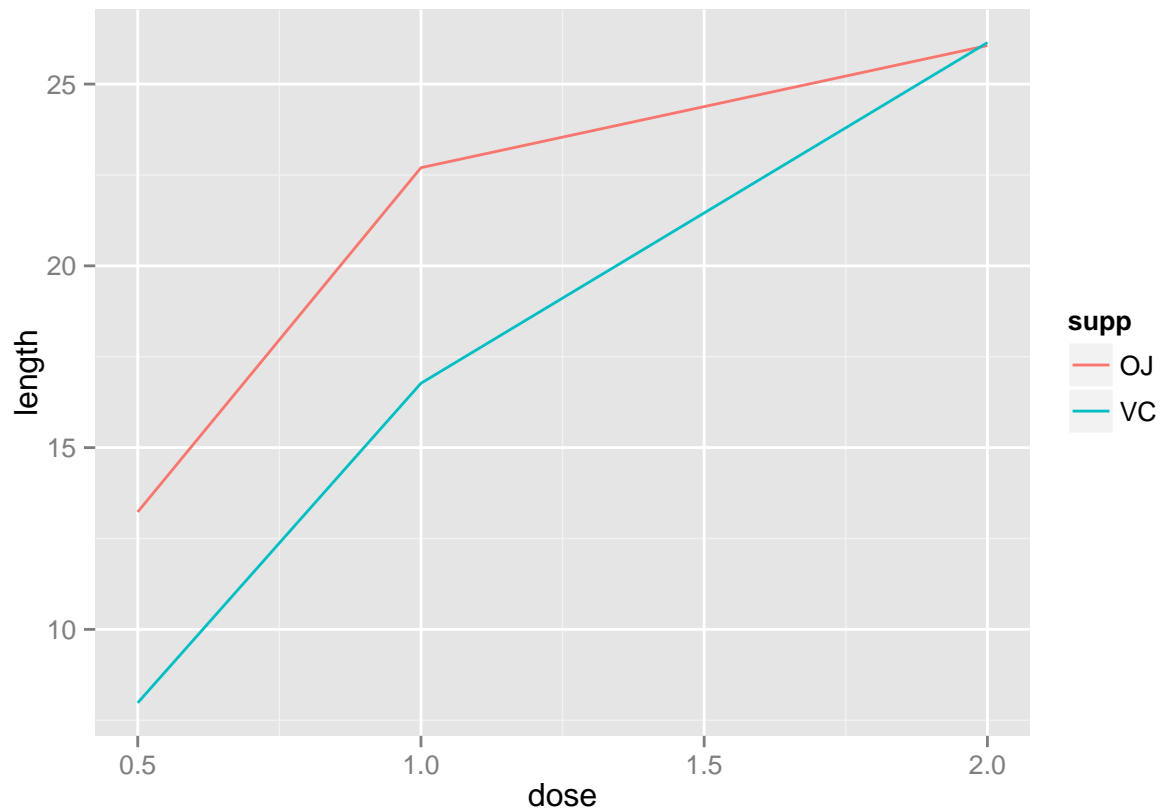
```
tg <- ToothGrowth %>%
  group_by(supp, dose) %>%
  summarise(length = mean(len))
tg
```

```
## Source: local data frame [6 x 3]
## Groups: supp
##
##   supp dose length
## 1    OJ  0.5  13.23
```

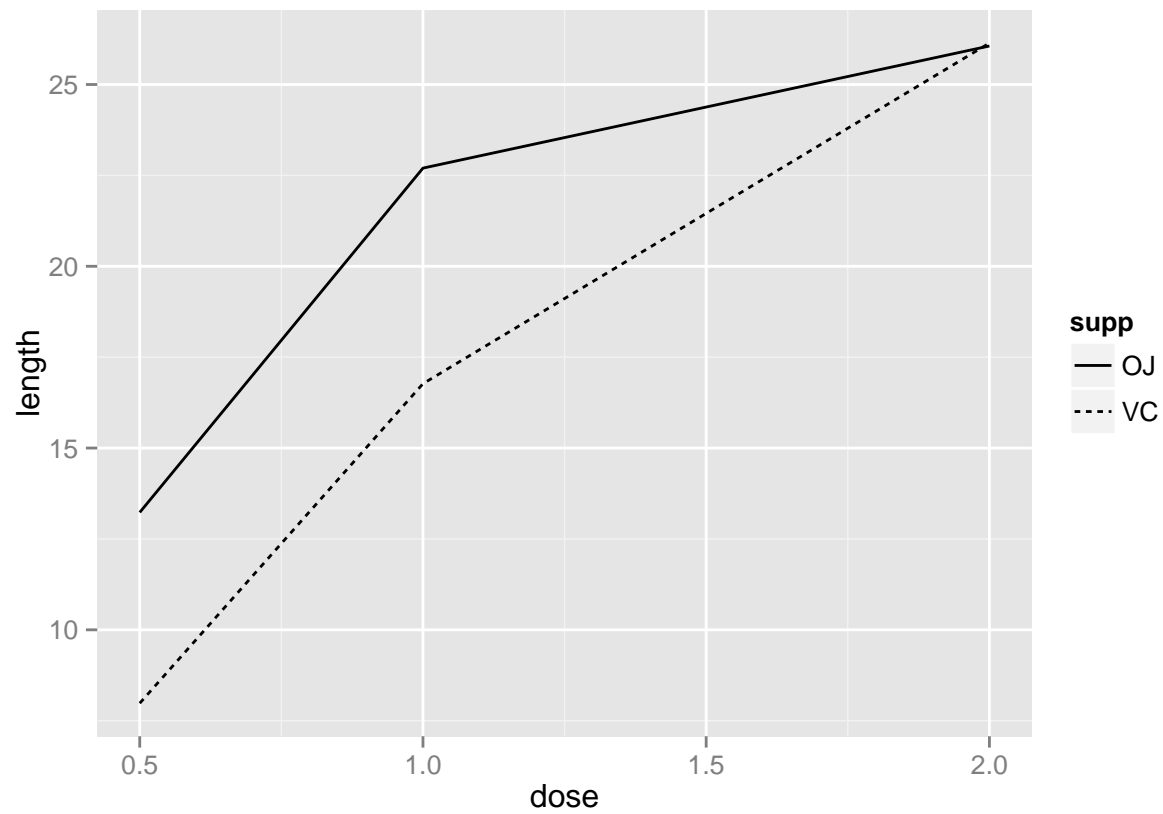
```
## 2    OJ  1.0  22.70
## 3    OJ  2.0  26.06
## 4    VC  0.5   7.98
## 5    VC  1.0  16.77
## 6    VC  2.0  26.14
```

Entonces podemos graficar la dosis con el tamaño y además ver la diferencia cuando cambia 'supp'.

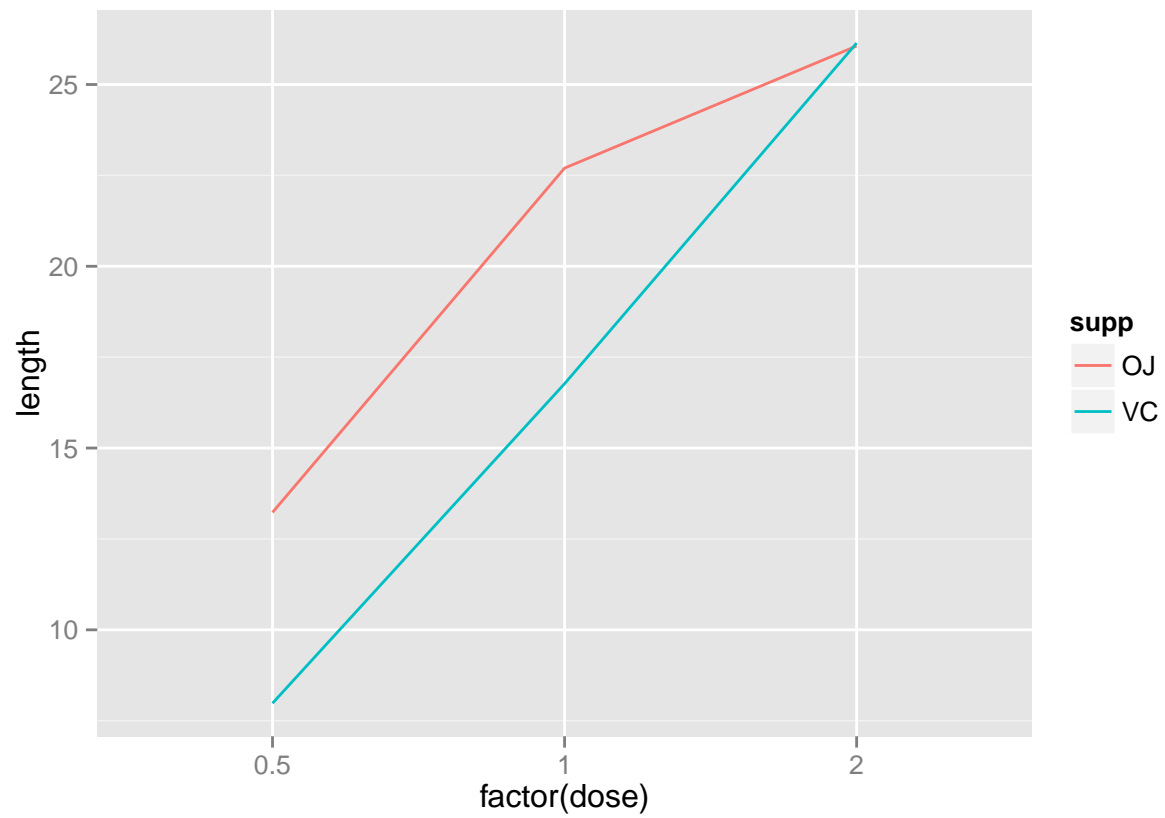
```
# Mapeamos 'supp' al color
plot <- ggplot(tg, aes(x = dose, y = length, col = supp)) + geom_line()
plot
```



```
# Mapeamos 'supp' al tipo de línea
plot <- ggplot(tg, aes(x = dose, y = length, linetype = supp)) + geom_line()
plot
```

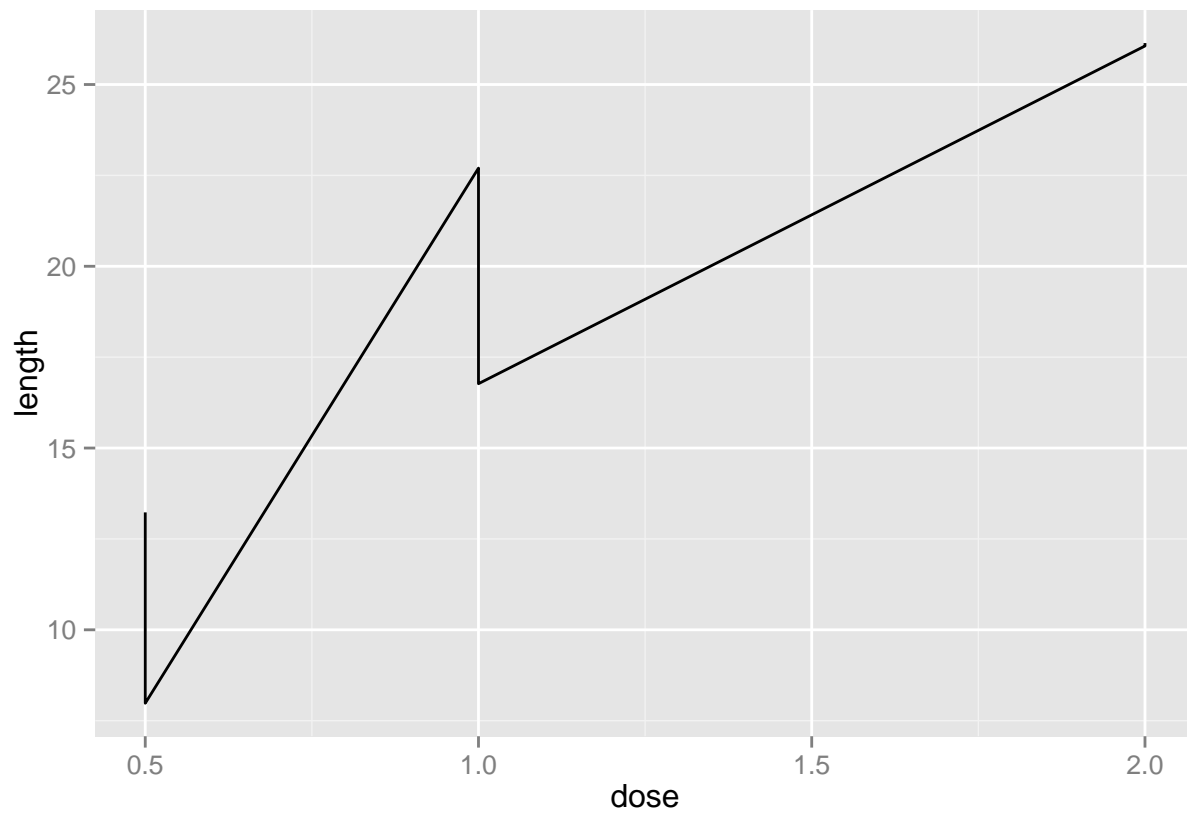


```
# Podemos pensar en la dosis como una variable categórica  
# Notar que cuando usamos variables categóricas debemos indicar como agrupar las líneas  
plot <- ggplot(tg, aes(x = factor(dose), y = length, col = supp, group = supp)) + geom_line()  
plot
```



¿Qué pasa si no diferenciamos según la variable 'supp'?

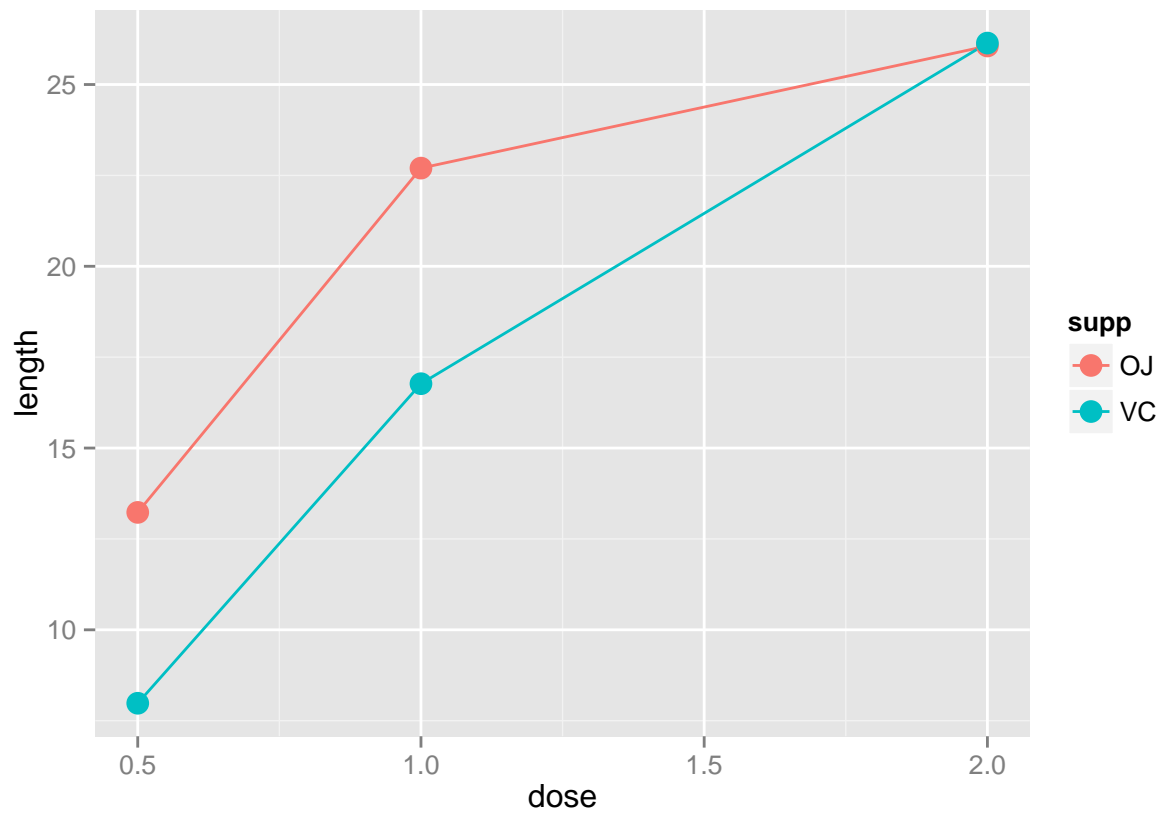
```
plot <- ggplot(tg, aes(x = dose, y = length)) + geom_line()
plot
```



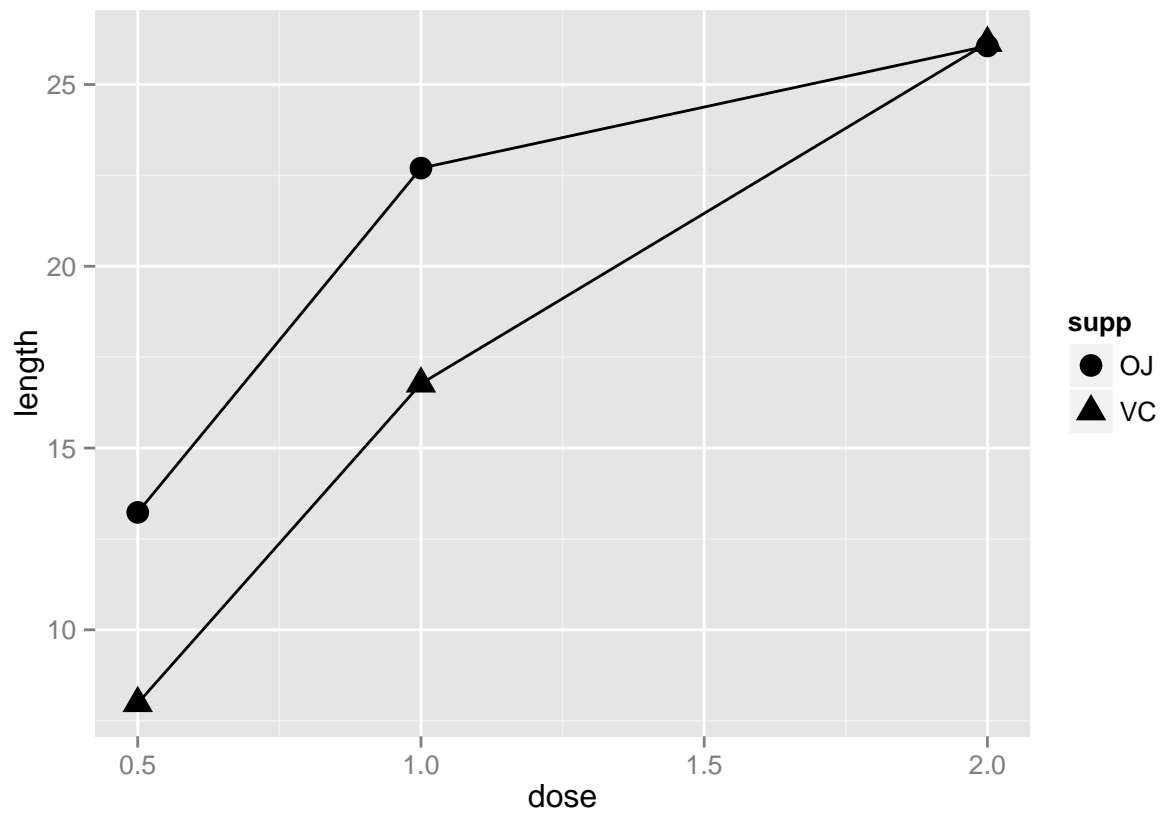
A través de la gráfica nos damos cuenta que hay algo mal con como estamos agrupando los datos.

Podemos agregar puntos en los nodos dados por los datos:

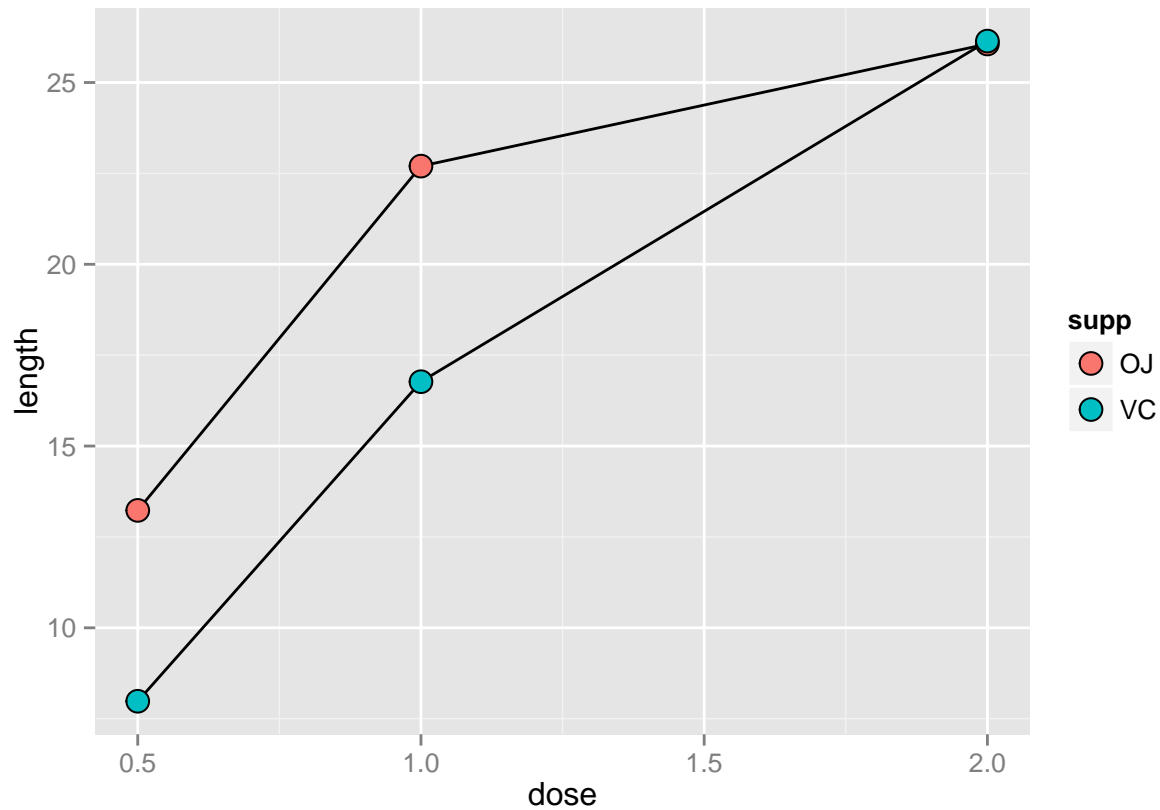
```
# Indicamos el tamaño de los puntos  
plot <- ggplot(tg, aes(x = dose, y = length, col = supp)) + geom_line() + geom_point(size = 4)  
plot
```



```
# Podemos separar los datos por el tipo de punto en lugar del color  
# 'shape' afecta únicamente a los puntos, las líneas no se ven afectadas por el atributo shape  
plot <- ggplot(tg, aes(x = dose, y = length, shape = supp)) + geom_line() + geom_point(size = 4)  
plot
```

```
# Indicamos los colores solo en los puntos  
# fill afecta solo a los puntos (las líneas no tienen relleno)  
plot <- ggplot(tg, aes(x = dose, y = length, fill = supp)) +  
  geom_line() + geom_point(size = 4, shape = 21)  
plot
```



Intervalos de confianza

Veamos los siguientes datos:

```
head(climate)
```

```
##      Source Year Anomaly1y Anomaly5y Anomaly10y Unc10y
## 1 Berkeley 1800      NA      NA      -0.435  0.505
## 2 Berkeley 1801      NA      NA      -0.453  0.493
## 3 Berkeley 1802      NA      NA      -0.460  0.486
## 4 Berkeley 1803      NA      NA      -0.493  0.489
## 5 Berkeley 1804      NA      NA      -0.536  0.483
## 6 Berkeley 1805      NA      NA      -0.541  0.475
```

Estos datos presentan anomalías en el clima a través de los años. Nos interesa graficar los datos para Berkeley. La variable 'Unc10y' indica el intervalo de confianza del 95% para cada valor de la anomalía. Seleccionemos únicamente las variables que nos interesan para nuestra gráfica.

```
clim <- climate %>%
  filter(Source == "Berkeley") %>%
  select(Year, Anomaly10y, Unc10y)

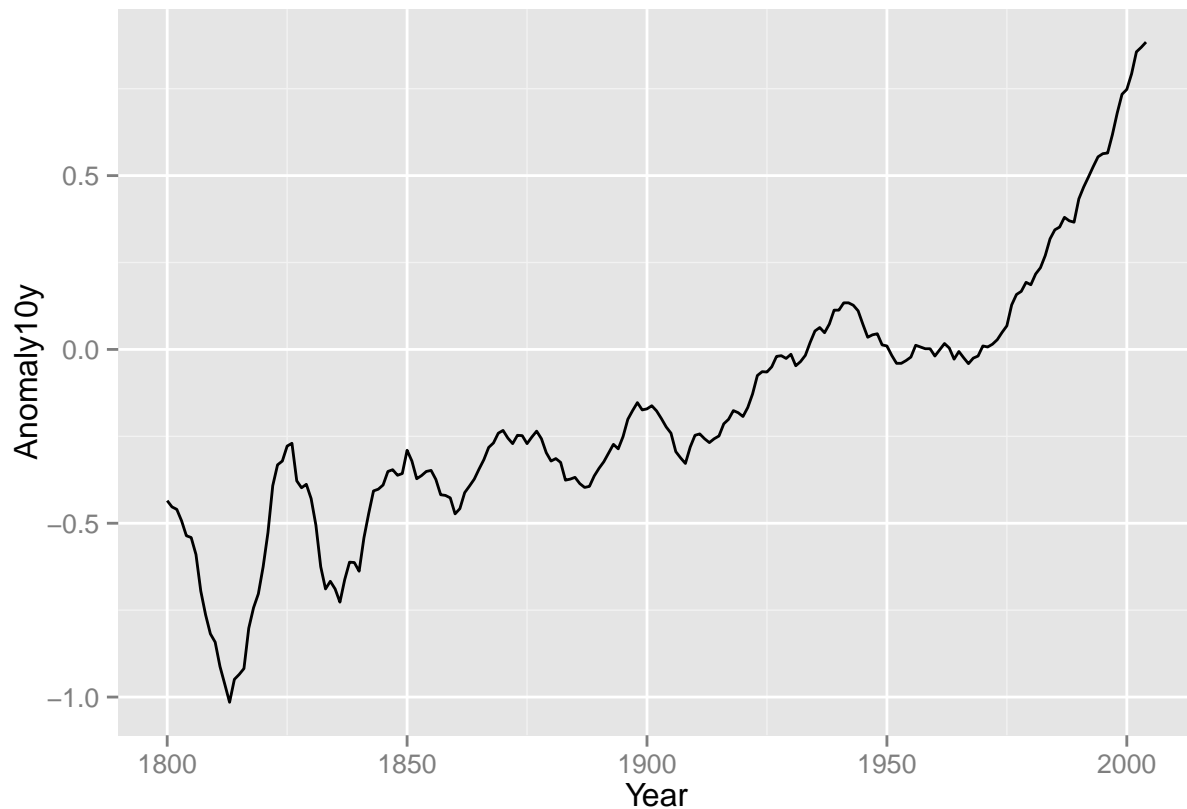
head(clim)
```

```
##      Year Anomaly10y Unc10y
## 1 1800      -0.435  0.505
```

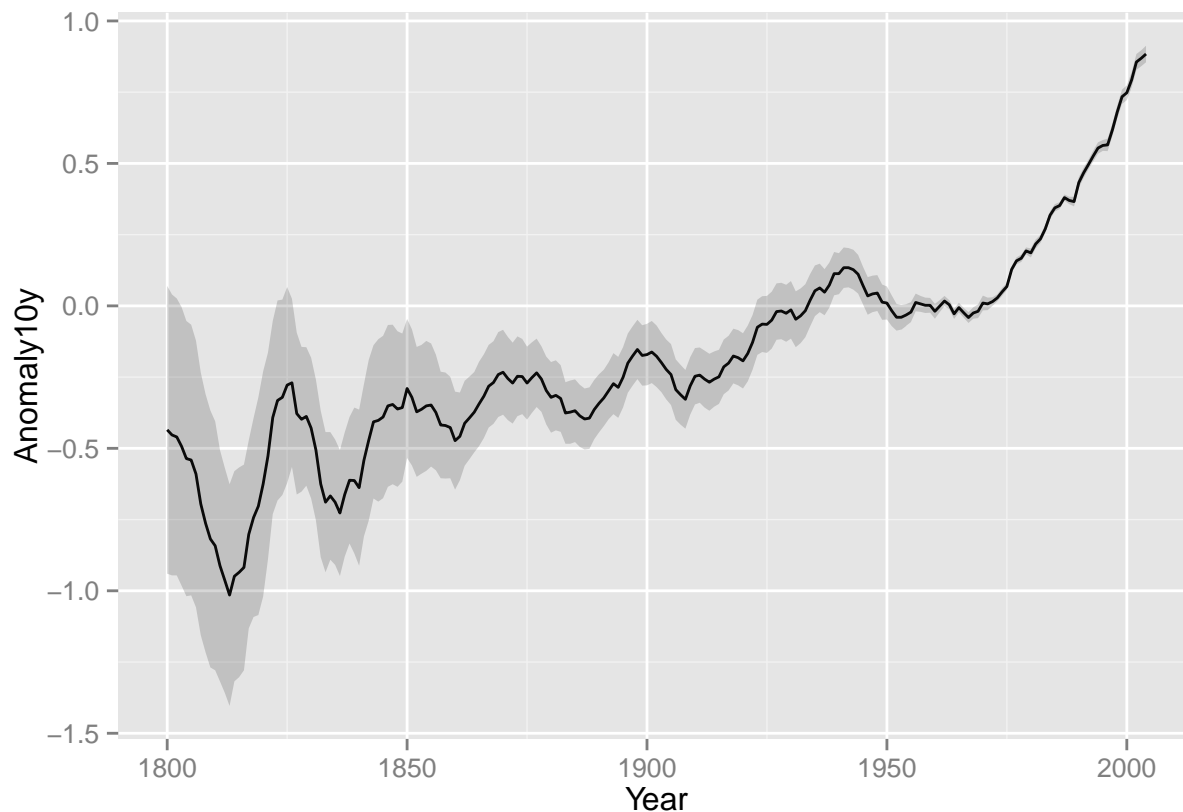
```
## 2 1801      -0.453  0.493
## 3 1802      -0.460  0.486
## 4 1803      -0.493  0.489
## 5 1804      -0.536  0.483
## 6 1805      -0.541  0.475
```

Ahora grafiquemos:

```
# Graficamos con una línea los valores de la anomalía a través de los años
plot <- ggplot(clim, aes(x = Year, y = Anomaly10y)) + geom_line()
plot
```



```
# Ahora añadimos el intervalo de confianza con geom_ribbon
# alpha indica la opacidad de la gráfica
plot <- plot + geom_ribbon(aes(ymin = Anomaly10y - Unc10y, ymax = Anomaly10y + Unc10y), alpha = 0.2)
plot
```



Gráficas de dispersión (puntos)

Veamos la siguiente base de datos. Nos interesa graficar como va cambiando la estatura conforme a la edad. Para ello graficaremos la dispersión de los datos e iremos añadiendo capas que nos ayuden a ver más a fondo los datos.

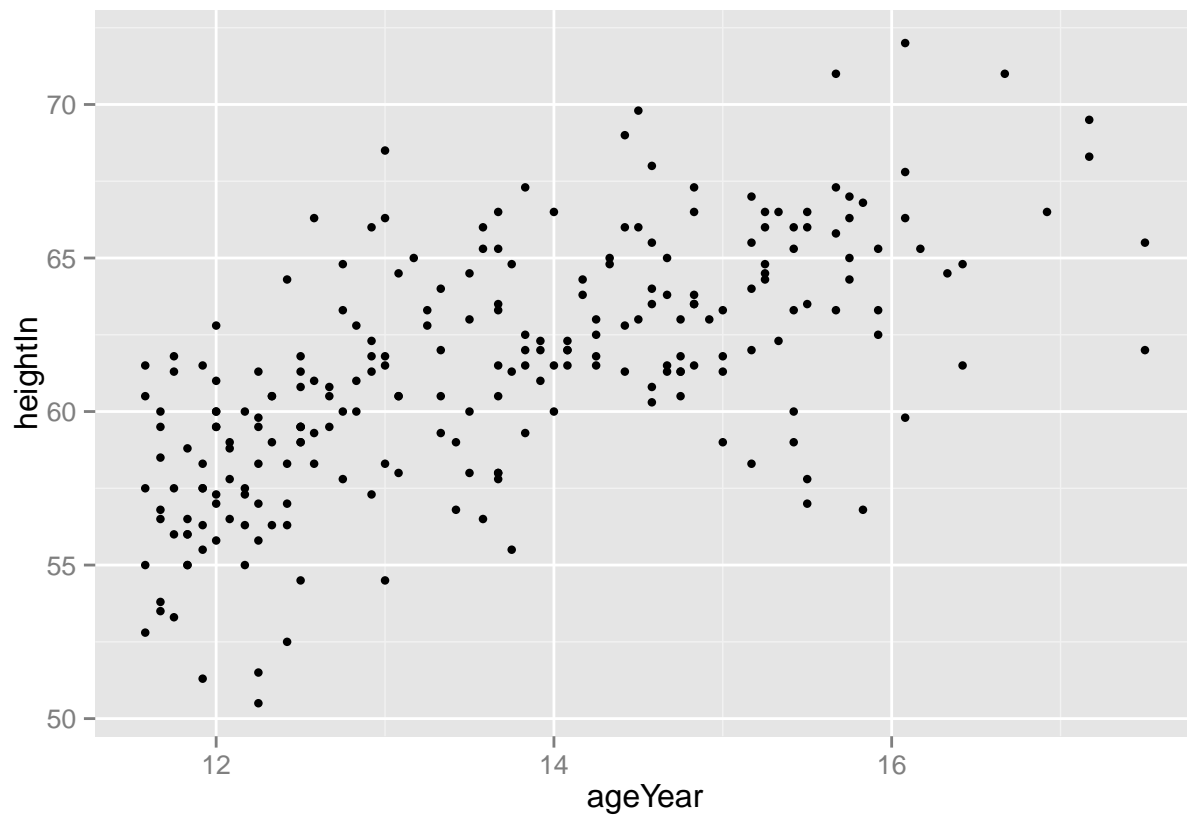
```
# Cambio el nombre de la base de datos para nos escribir todo
hw <- heightweight

head(hw)
```

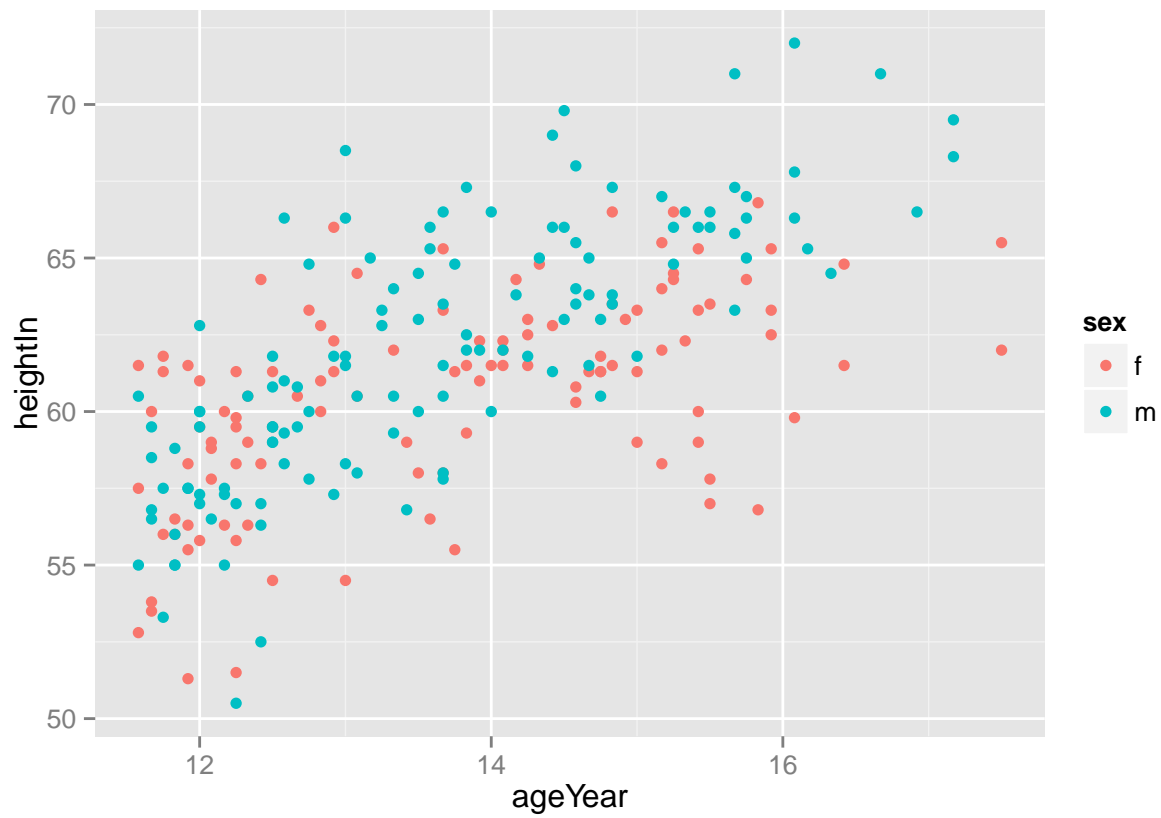
```
##   sex ageYear ageMonth heightIn weightLb
## 1  f   11.92     143     56.3     85.0
## 2  f   12.92     155     62.3    105.0
## 3  f   12.75     153     63.3    108.0
## 4  f   13.42     161     59.0     92.0
## 5  f   15.92     191     62.5    112.5
## 6  f   14.25     171     62.5    112.0
```

Entonces grafiquemos:

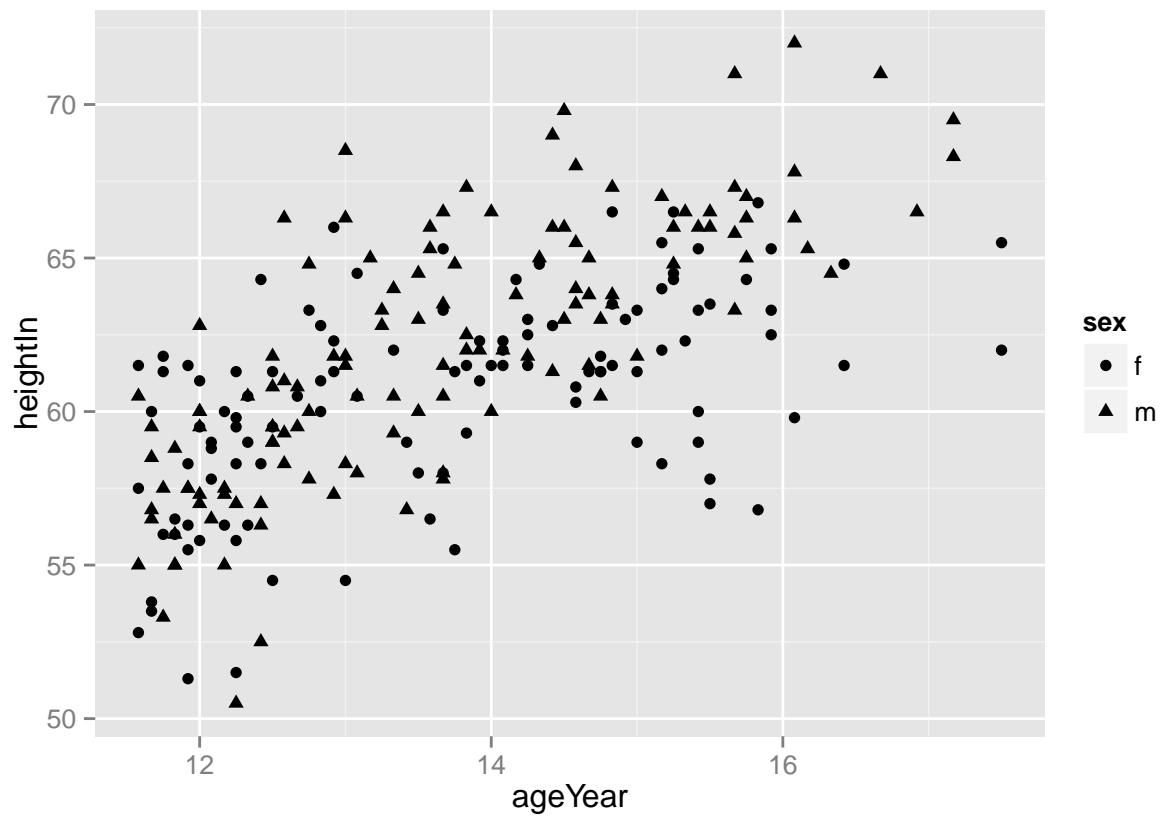
```
# Primero hacemos una gráfica de dispersión sencilla
plot <- ggplot(hw, aes(x = ageYear, y = heightIn)) + geom_point(size = 1.5)
plot
```



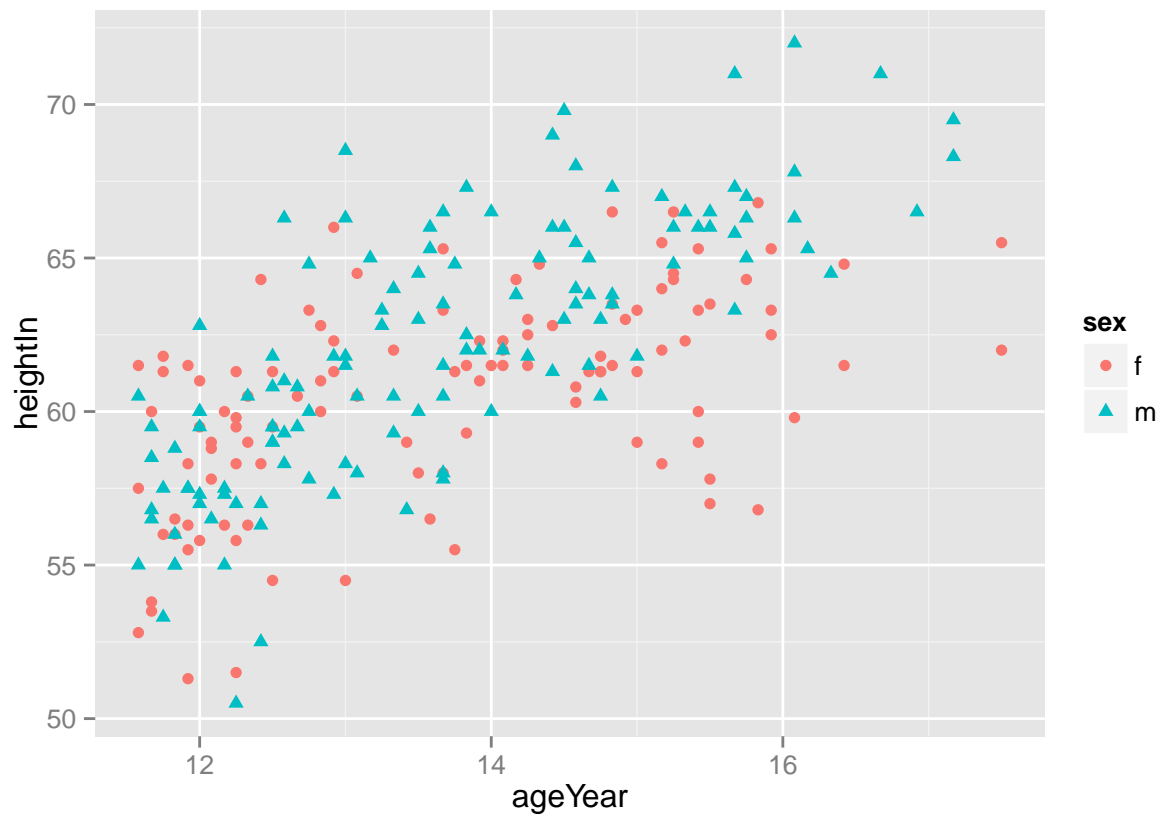
```
# Agrupemos los datos según el sexo, indicando con el color de los puntos  
plot <- ggplot(hw, aes(x = ageYear, y = heightIn, col = sex)) + geom_point()  
plot
```



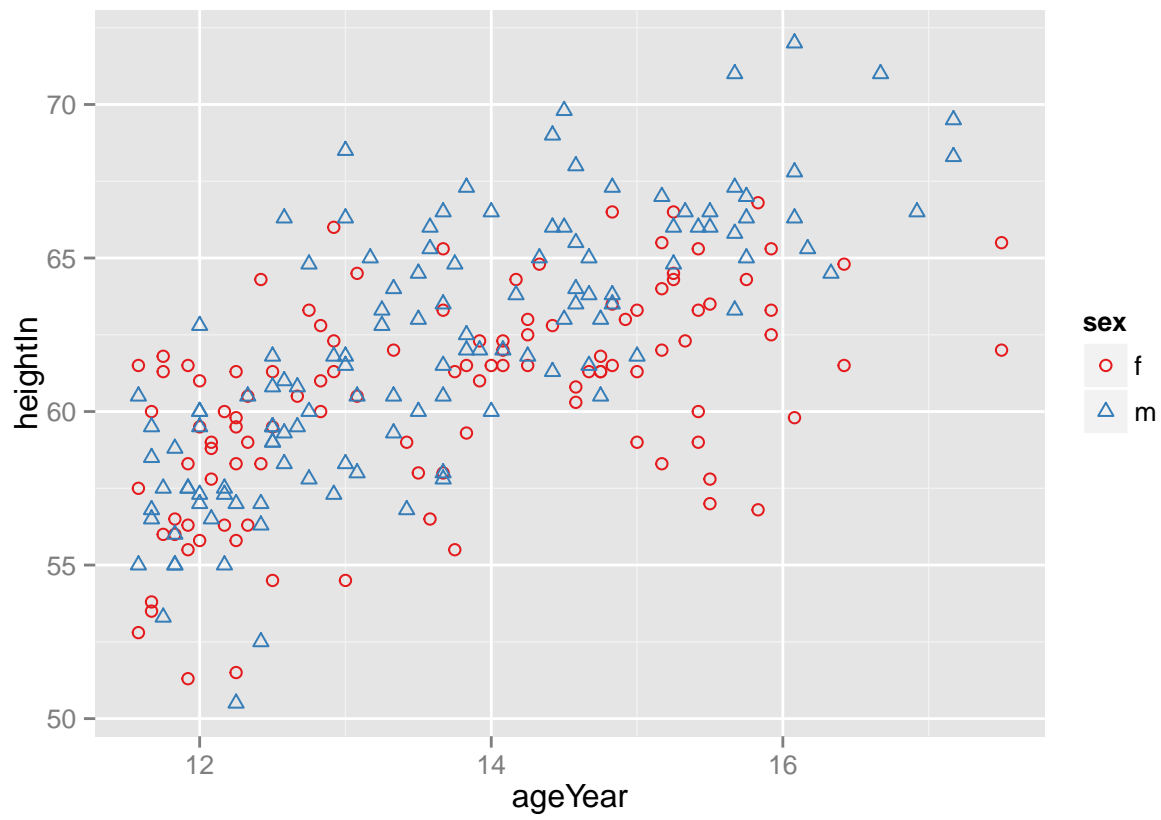
```
# Agrupemos los datos según el sexo, indicando con la forma de los puntos  
plot <- ggplot(hw, aes(x = ageYear, y = heightIn, shape = sex)) + geom_point()  
plot
```



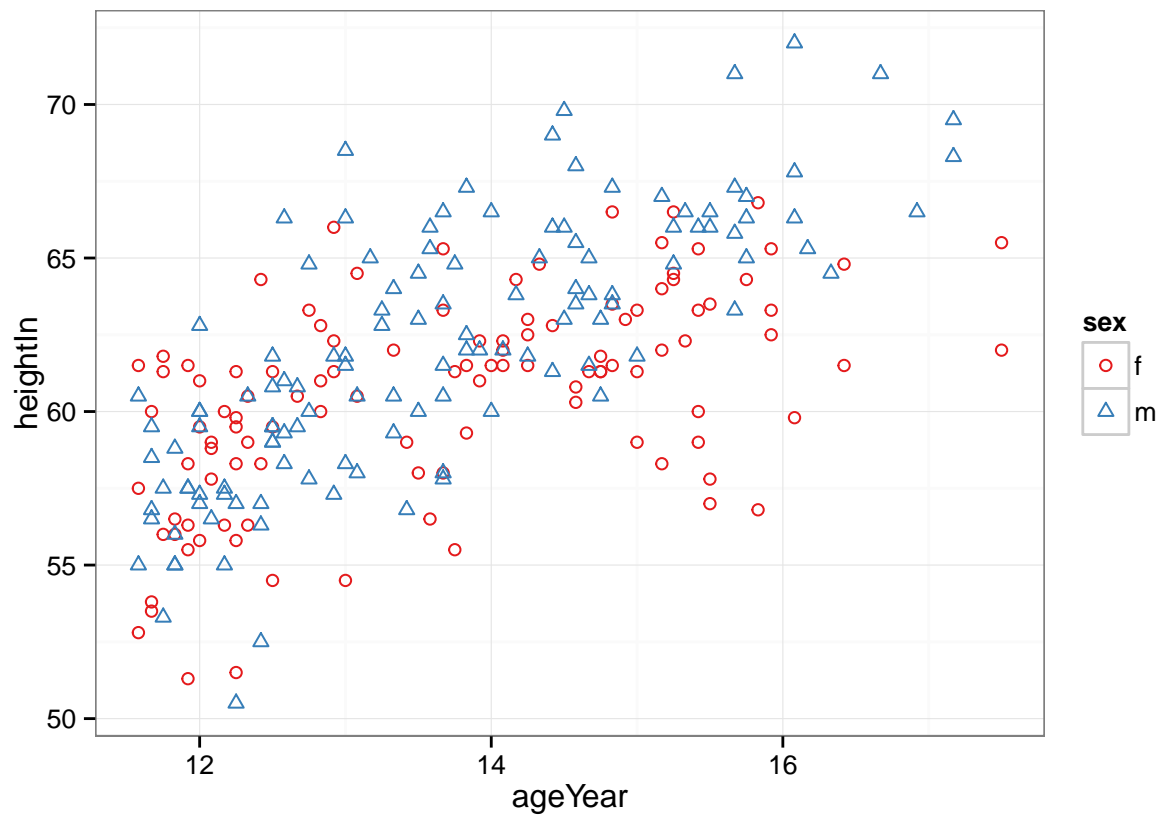
```
# Usemos ambos indicadores  
# Agrupemos los datos según el sexo, indicando con el color de los puntos  
plot <- ggplot(hw, aes(x = ageYear, y = heightIn, col = sex, shape = sex)) + geom_point()  
plot
```



```
# Personalicemos un poco más
plot <- ggplot(hw, aes(x = ageYear, y = heightIn, col = sex, shape = sex)) + geom_point()
plot <- plot + scale_shape_manual(values = c(1,2)) +
  scale_colour_brewer(palette = "Set1")
plot
```

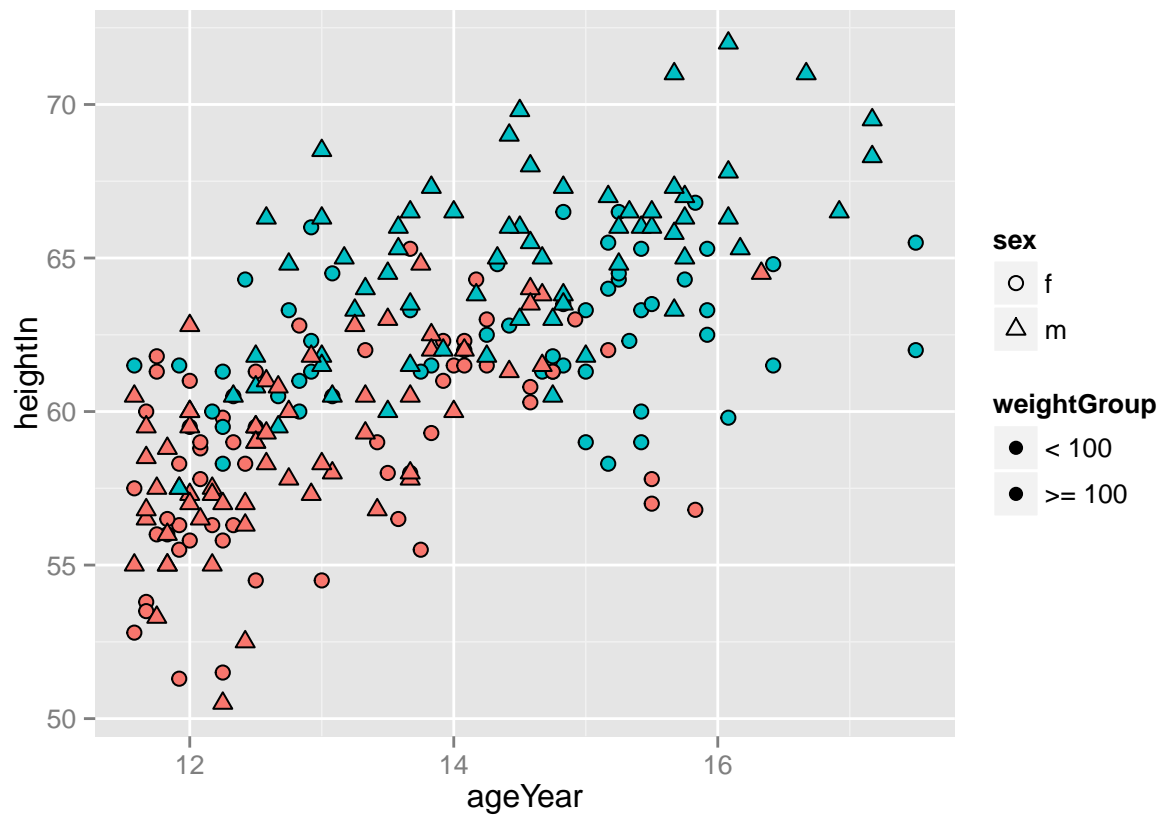



```
# Fondo blanco  
plot + theme_bw()
```

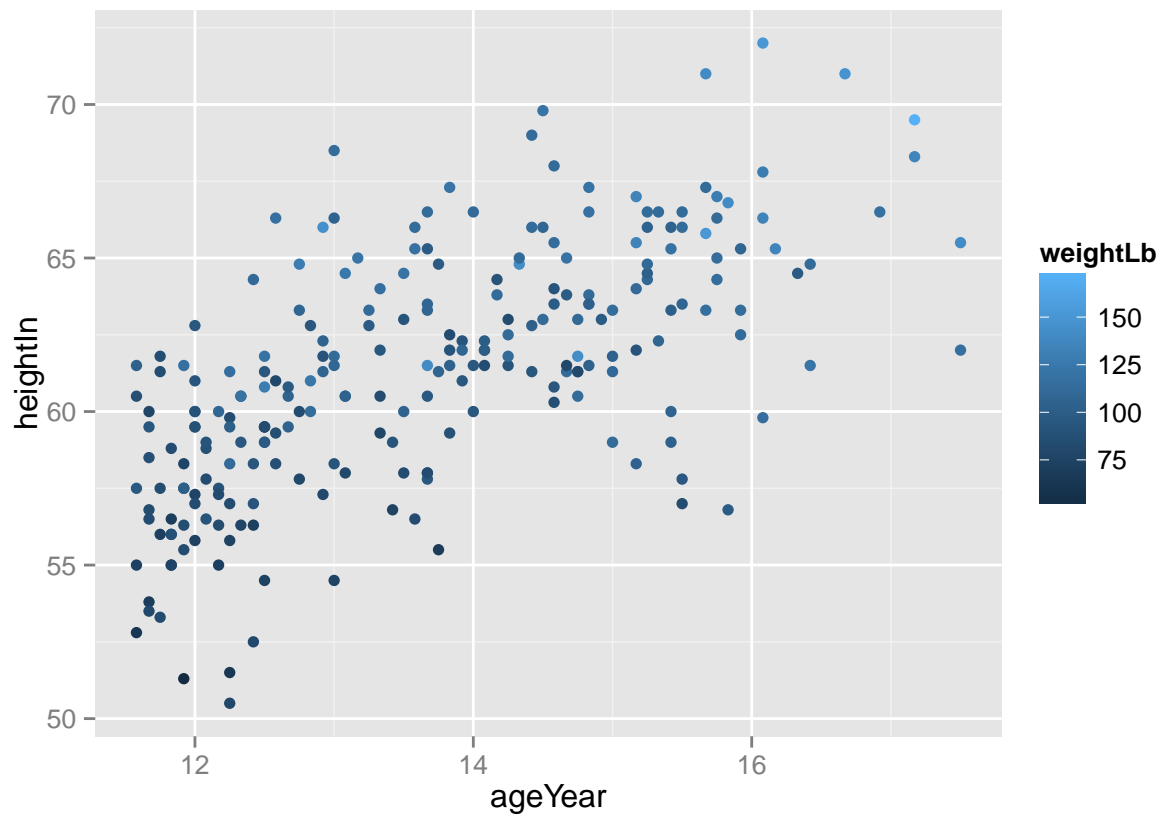


```
# Agrupemos para más de una variable.
# Crearemos una variable categórica para dividir el peso en dos partes
hw$weightGroup <- cut(hw$weightLb, breaks = c(-Inf,100,Inf),
                      labels = c("< 100", ">= 100"))

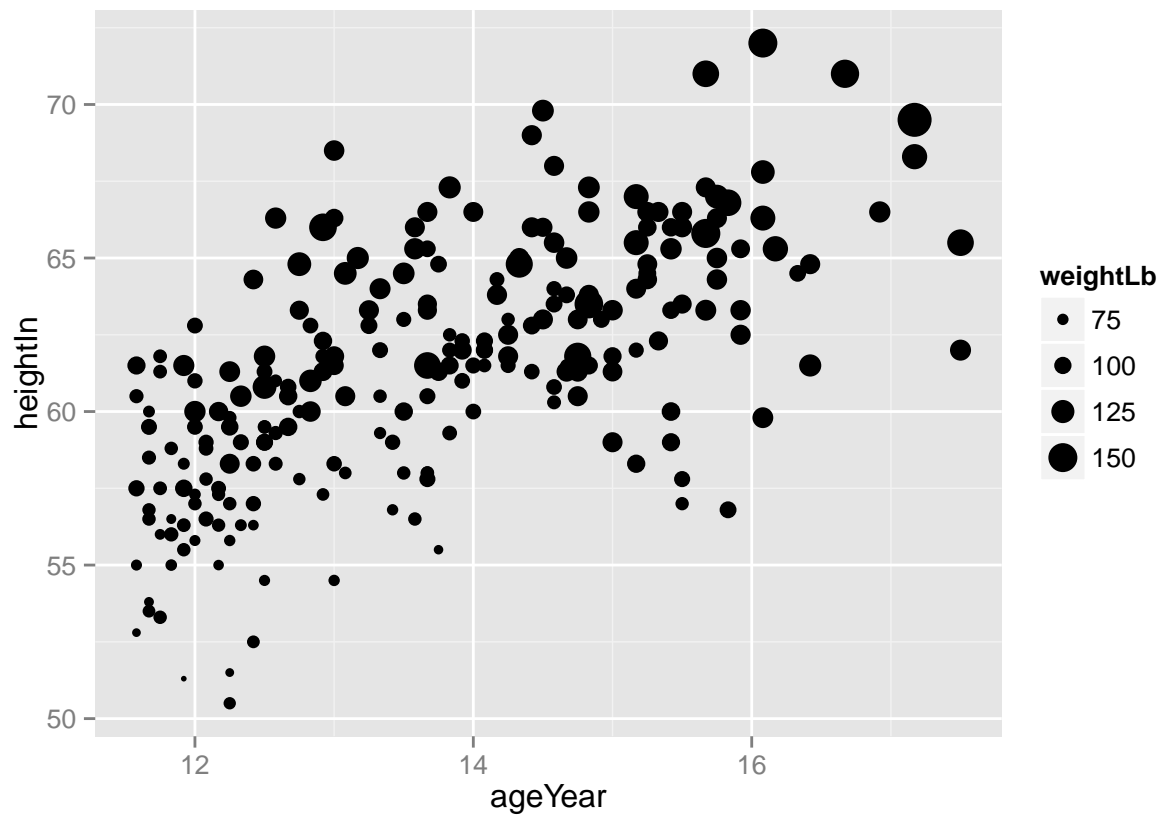
# Graficamos
plot <- ggplot(hw, aes(x = ageYear, y = heightIn, shape = sex, fill = weightGroup)) +
  geom_point(size = 2.5) +
  scale_shape_manual(values = c(21, 24))
plot
```



```
# También podemos usar el peso como una variable continua
# Indicamos el valor con el color
plot <- ggplot(hw, aes(x = ageYear, y = heightIn, col = weightLb)) + geom_point()
plot
```



```
# Indicamos valor con el tamaño de los puntos  
plot <- ggplot(hw, aes(x = ageYear, y = heightIn, size = weightLb)) + geom_point()  
plot
```



```
# Añadimos el indicador del sexo  
plot <- ggplot(hw, aes(x = ageYear, y = heightIn, size = weightLb, col = sex)) + geom_point(alpha = 0.5)  
plot + scale_colour_brewer(palette = "Set1")
```

