

TEMA 2. Gráficos ggplot2.

Máxima Formación

Máster en Estadística Aplicada con R software

Contenidos

El paquete ggplot2	2
Comparación con otros gráficos	2
¿Cuál es la gramática de los gráficos?	5
La configuración (aes)	6
Las capas (geom)	8
Las etiquetas (labs)	13
El Tema (theme)	14
El aspecto (facet)	15
La leyenda: eliminación y cambio de posición.	19
Unir varios gráficos en una misma figura	22
Tipos de gráficos	23
Diagrama de dispersión	23
Curvas de ajuste	24
Histograma	28
Diagramas de cajas	30
Gráficos de barras	34
Gráficas de medias e intervalos de confianza	38
Opciones avanzadas	40
Gráficos por subgrupos	40
Transformación de las escalas de los ejes	44
Cambiar temas	46
Guardar el gráfico	47
Referencias	47

El paquete **ggplot2**

El paquete **ggplot2** permite definir gráficos de forma eficiente, elegante y sencilla.

Comparación con otros gráficos

En comparación con los otros gráficos, ggplot2:

- Es más detallado para los gráficos simples
- Es menos detallado para gráficos complejos o personalizados
- Los datos deben estar siempre en un archivo data.frame
- Utiliza un sistema diferente para agregar elementos de trama

Como primer ejemplo vamos a comparar gráficos del paquete básico de R y gráficos del paquete **ggplot2**. Utilizaremos los datos **iris** del paquete **dataset** en el que aparecen las medidas de ancho y largo de sépalo y pétalo de 3 especies diferentes de lirios.

```
data(iris)
head(iris)

##   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

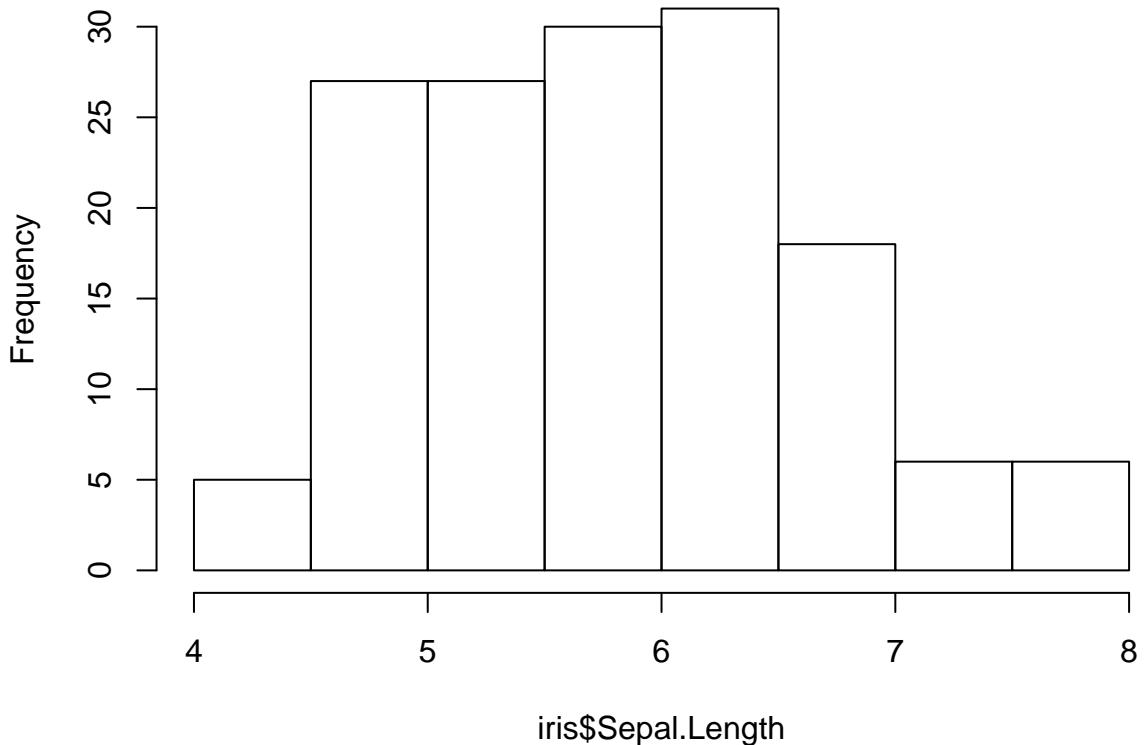
str(iris)

## 'data.frame': 150 obs. of 5 variables:
## $ Sepal.Length: num 5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
## $ Sepal.Width : num 3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
## $ Petal.Length: num 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
## $ Petal.Width : num 0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
## $ Species     : Factor w/ 3 levels "setosa","versicolor",...: 1 1 1 1 1 1 1 1 1 1 ...
```

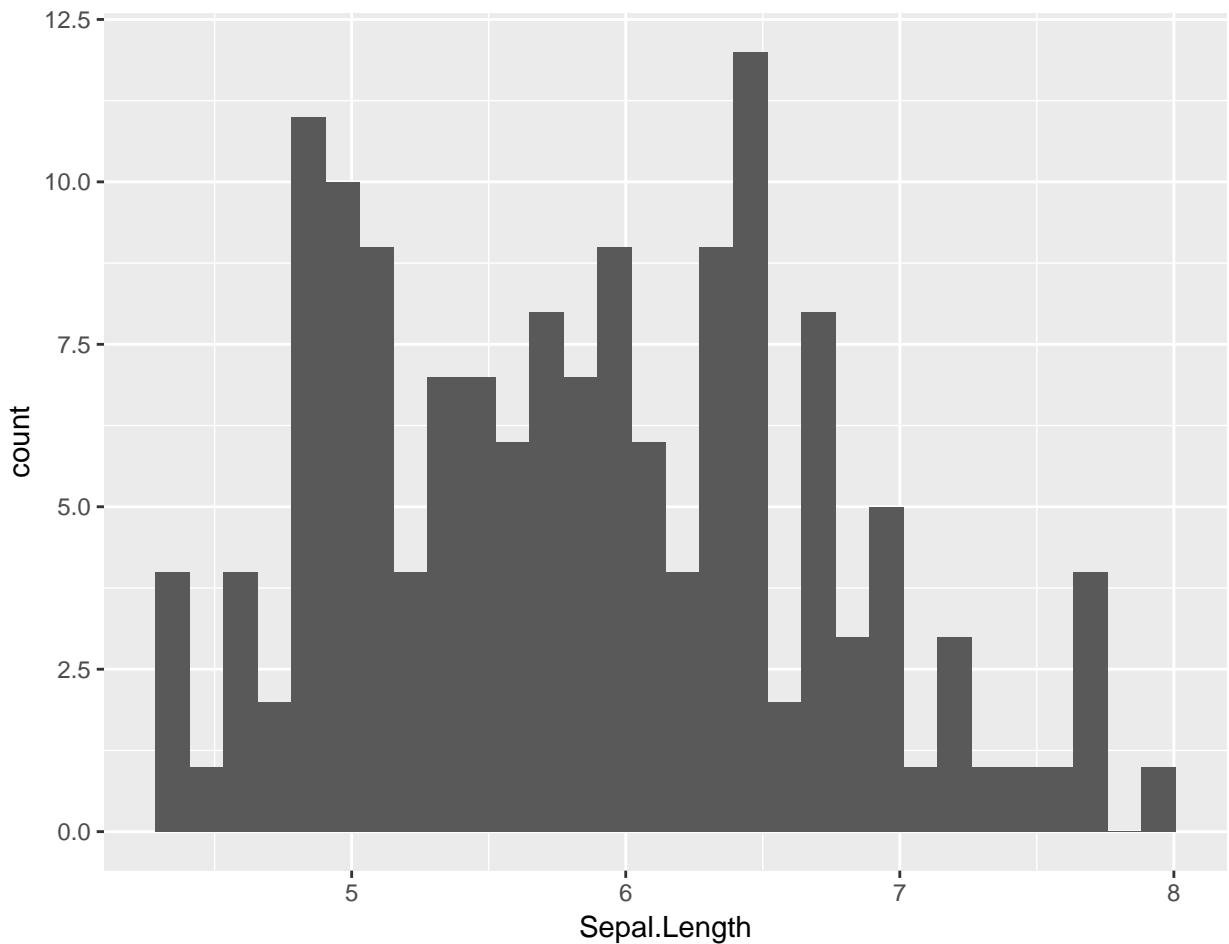
Realizamos un histograma con las fórmulas básicas y otro en ggplot2:

```
#histograma con las fórmulas básicas
hist(iris$Sepal.Length)
```

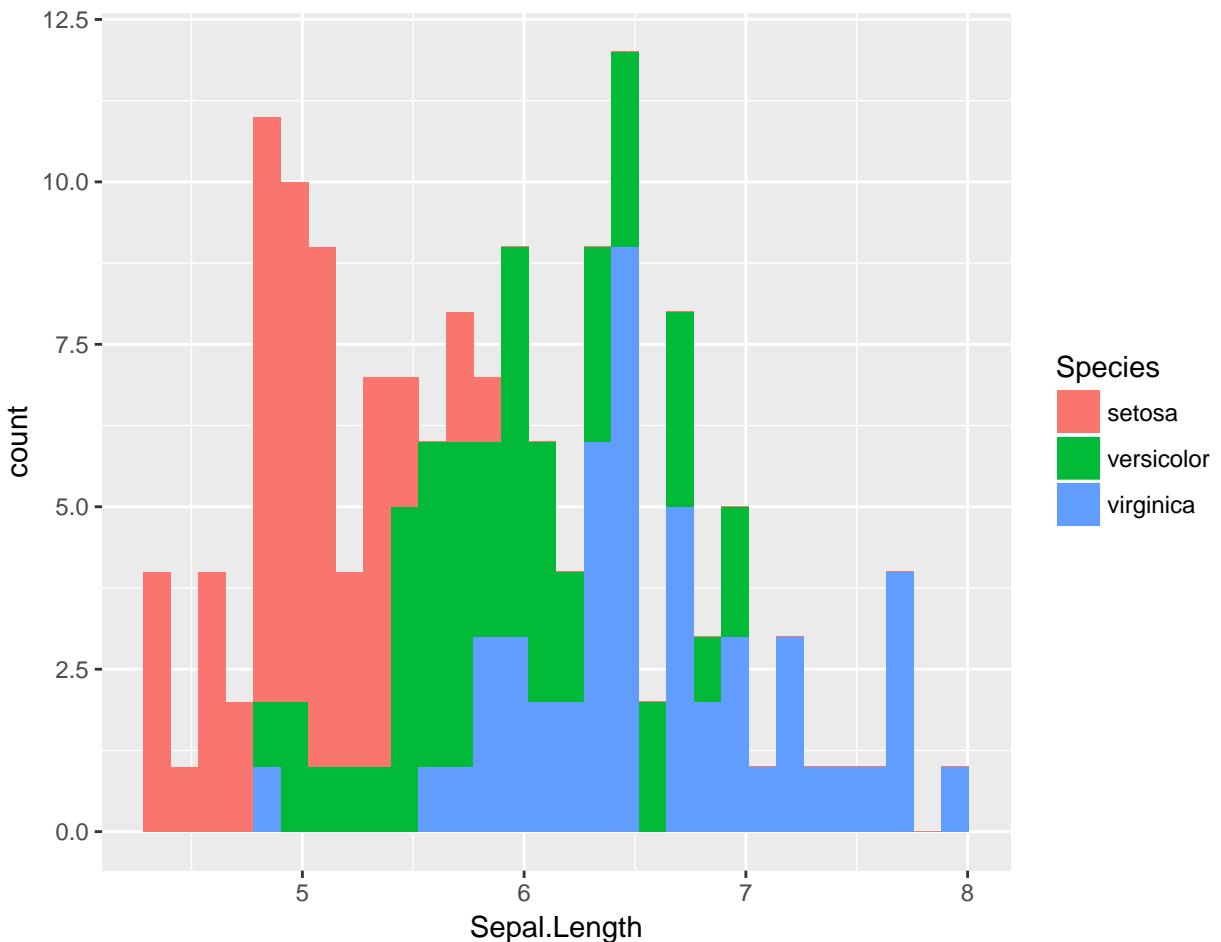
Histogram of iris\$Sepal.Length



```
#histograma con ggplot2
library(ggplot2) #primero activamos la librería
ggplot(iris, aes(x = Sepal.Length)) +
  geom_histogram()
```



```
# diferenciando las especies por color  
ggplot(iris, aes(x = Sepal.Length, fill=Species)) +  
  geom_histogram()
```



Como podemos observar, el código de los gráficos con **ggplot2** es algo más complicado pero la estética es mucho mejor.

¿Cuál es la gramática de los gráficos?

El paquete **ggplot2** se basa en la gramática de los gráficos (Wilkinson, 2005) y eso reporta varias ventajas respecto a otros gráficos como una estética muy buena o la simplificación en los comandos de gráficos complejos. Aunque también presenta algunas limitaciones ya que no se pueden realizar gráficos tridimensionales o interactivos.

La idea básica: especificar de forma independiente los bloques de construcción y combinarlos para crear prácticamente cualquier tipo de visualización gráfica que se desee.

Los bloques de construcción de un gráfico incluyen:

- Datos
- Mapeo estético
- Transformaciones estadísticas
- Escalas
- Sistema de coordenadas
- Ajustes de posición
- Aspecto
- Objeto geométrico

La configuración (`aes`)

La principal característica de la estructura `ggplot2` es la forma en que se hacen tramas de gráficos mediante la **adición de “capas”**. Veamos el proceso paso a paso.

Lo primero que debes hacer es decirle a la función `ggplot` qué **conjunto de datos (o dataset) debe utilizar**. Esto se hace escribiendo `ggplot(df)`, donde `df` es un **dataframe** que contiene todas las características necesarias para hacer la trama. A diferencia de los gráficos de base, `ggplot` no toma los vectores como argumentos. Este es el primer paso.

El argumento `aes()` es sinónimo de estética, `ggplot2` considera que el **eje X e Y** de la gráfica es estético, junto con el *color*, el *tamaño*, la *forma*, el *relleno*, etc. Se puede agregar cualquier **estética** que se desee dentro del argumento `aes()`, como por ejemplo indicar los **ejes X e Y**, especificando las variables respectivas del conjunto de datos. La variable en función de la cual el **color, tamaño, forma y trazo** debe cambiar también se puede especificar aquí mismo. Debes tener en cuenta que la estética especificada aquí será heredada por todas las capas `geom` que se agregarán posteriormente.

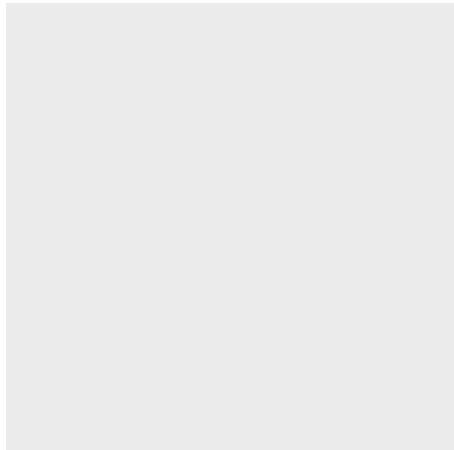
Para ir descubriendo paso a paso cómo debemos indicar las características del gráfico vamos a trabajar con los datos `diamonds` del paquete `ggplot2`. Tiene estructura de **data.frame** y variables numéricas y categóricas. Para conocer mejor estos datos accede a la ayuda de R `?diamonds`.

Ejemplo

Comenzamos con unos ejemplos en los que no se imprimirá ningún gráfico, ya que no vamos a añadir la capa de las geometrías.

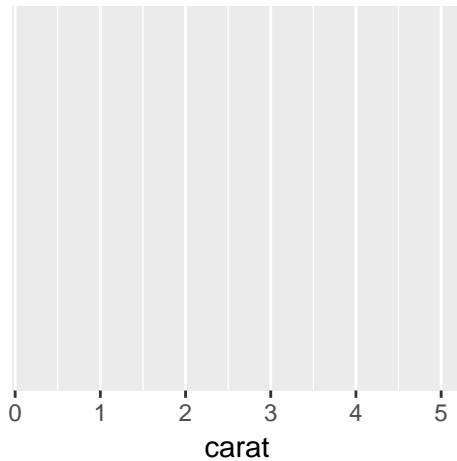
Si sólo introducimos la referencia de los datos.

```
ggplot(diamonds)
```



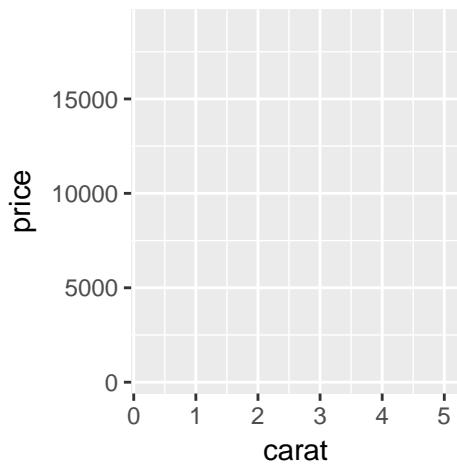
Sólo indicamos el eje X. El eje Y debe ser especificado en la geometría.

```
ggplot(diamonds, aes(x=carat))
```



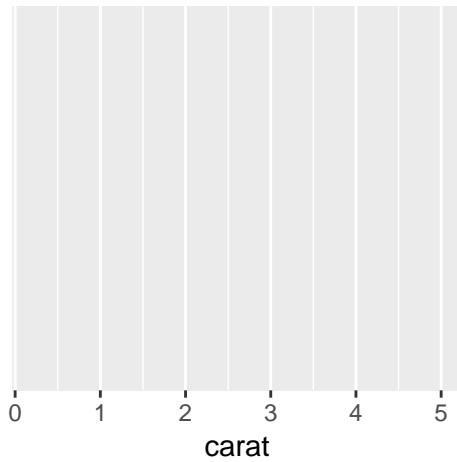
Especificamos ambos ejes X e Y, que ahora serán fijos en todas las capas.

```
ggplot(diamonds, aes(x=carat, y=price))
```



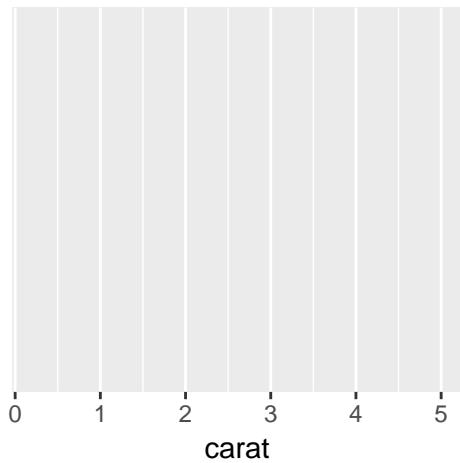
Cada categoría de la variable “cut” tendrá un color diferente, una vez le indiquemos la geometría.

```
ggplot(diamonds, aes(x=carat, color=cut))
```



Si desea tener el color, el tamaño, etc, fijo (es decir, no varía basándose en una variable del marco de datos), es necesario especificarlo fuera de los `aes()`, como en este caso.

```
ggplot(diamonds, aes(x=carat), color="steelblue")
```

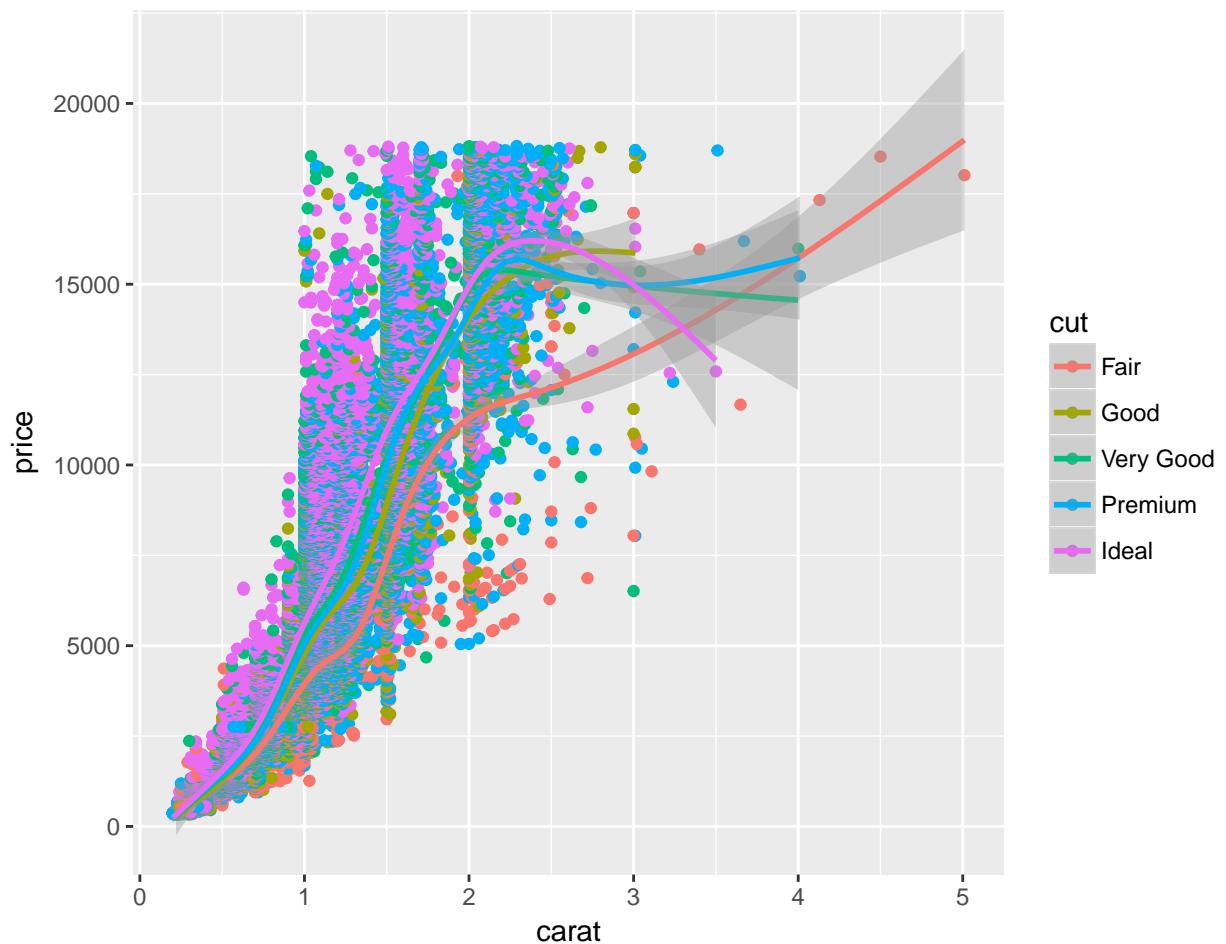


Las capas (geom)

Las capas de ggplot2 también se denominan `geom`. Una vez que la configuración de base se hace, se pueden agregar los geoms uno encima del otro.

Por ejemplo, vamos a agregar la geometría de puntos y otra de suavizado.

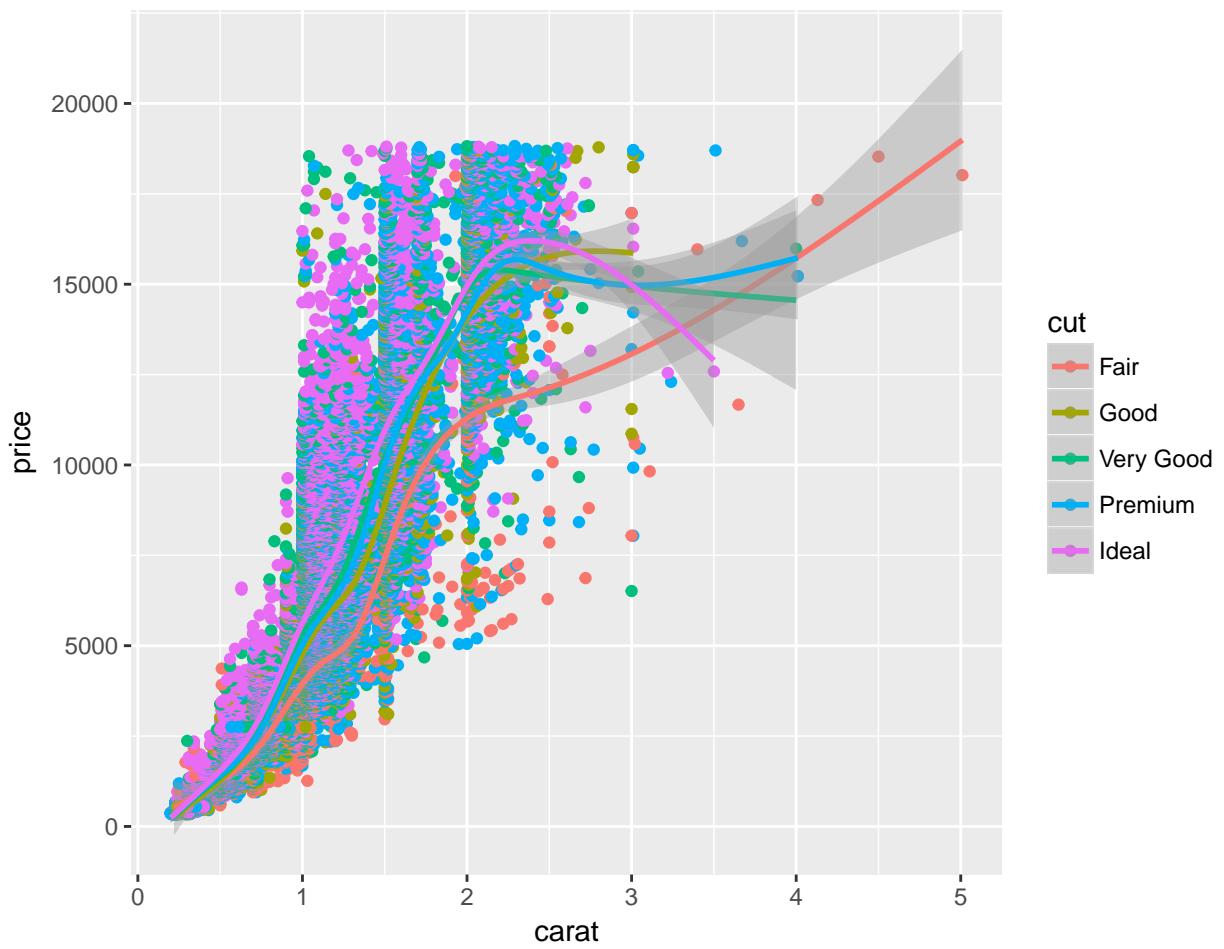
```
ggplot(diamonds, aes(x=carat, y=price, color=cut)) +  
  geom_point() +  
  geom_smooth()
```



Hemos añadido dos capas (geoms) a esta gráfica: `geom_point()` y `geom_smooth()`. Dado que el eje Y del eje X y el color se definieron en la propia configuración del ggplot, en `aes()`, estas dos capas heredaron esa estética.

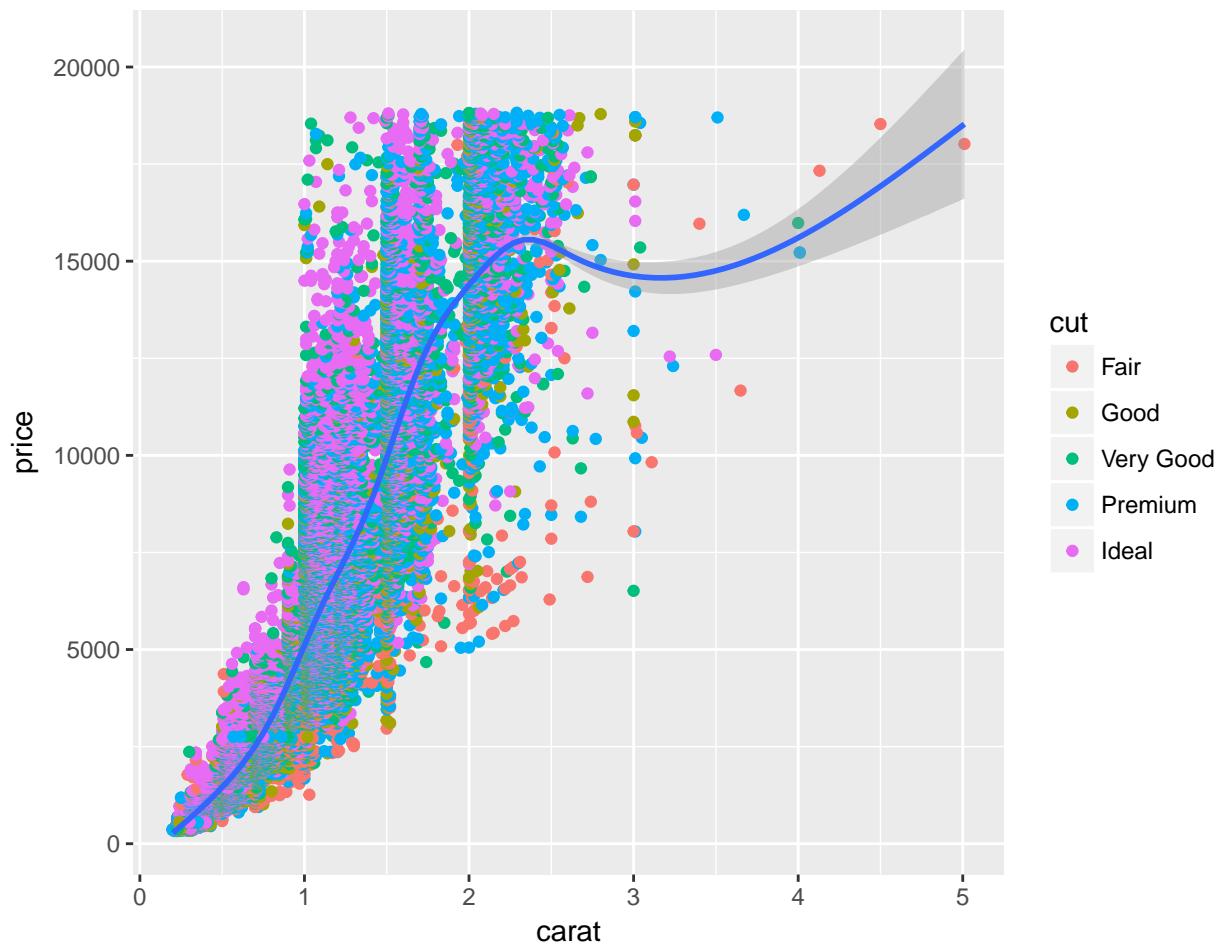
Como alternativa, se puede especificar las estéticas dentro de la capa geom, como se muestra a continuación.

```
ggplot(diamonds) +
  geom_point(aes(x=carat, y=price, color=cut)) +
  geom_smooth(aes(x=carat, y=price, color=cut))
```

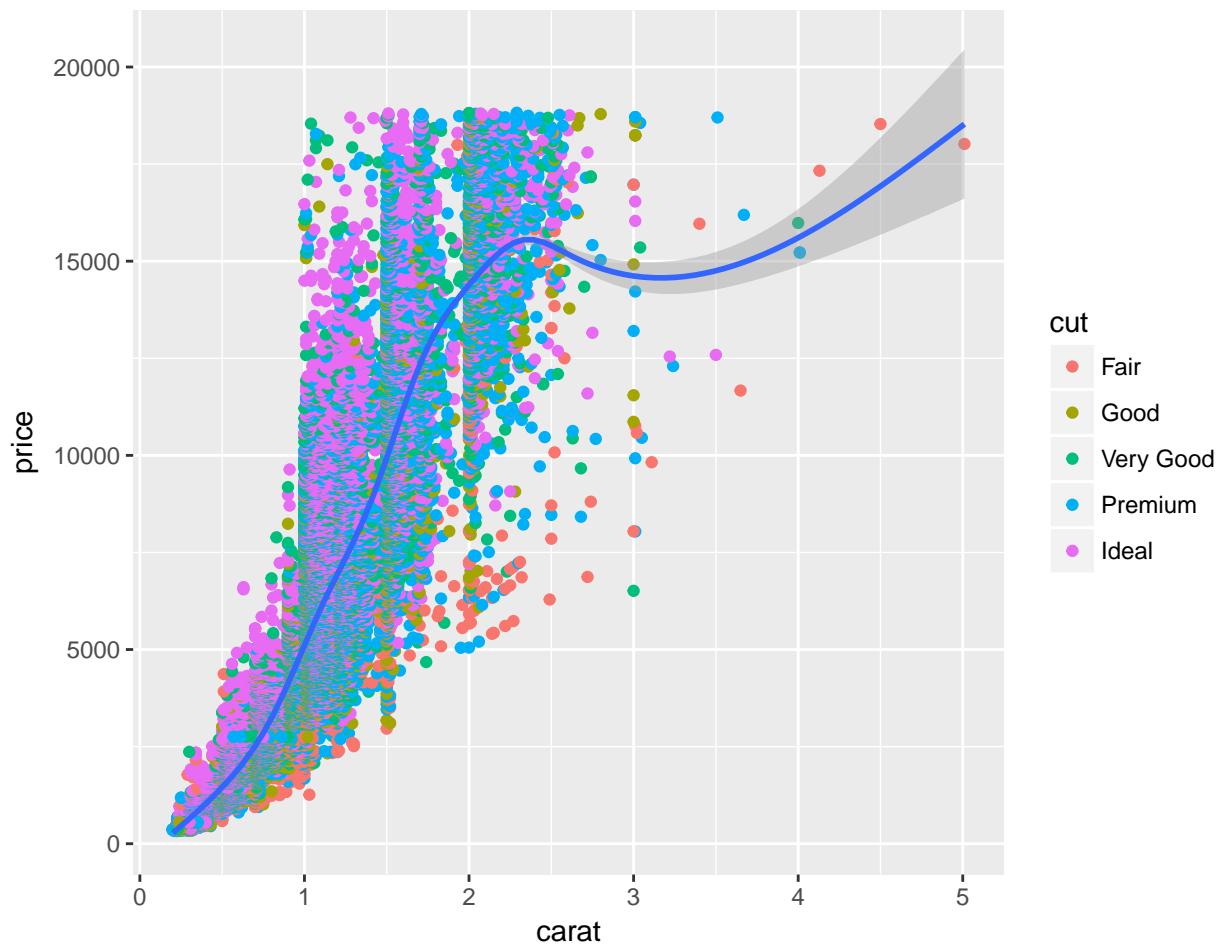


Hemos realizado el mismo gráfico escribiendo las órdenes de dos formas diferentes. Observa el eje X e Y y cómo el color de los puntos varía en función del valor de la variable de corte. La leyenda se agregó automáticamente. Sin embargo, si en lugar de tener múltiples líneas de suavizado para cada nivel de corte, queremos integrarlos todos bajo una sola línea, debemos quitar la estética del color de la capa de `geom_smooth()`, así:

```
ggplot(diamonds) +
  geom_point(aes(x=carat, y=price, color=cut)) +
  geom_smooth(aes(x=carat, y=price))
```

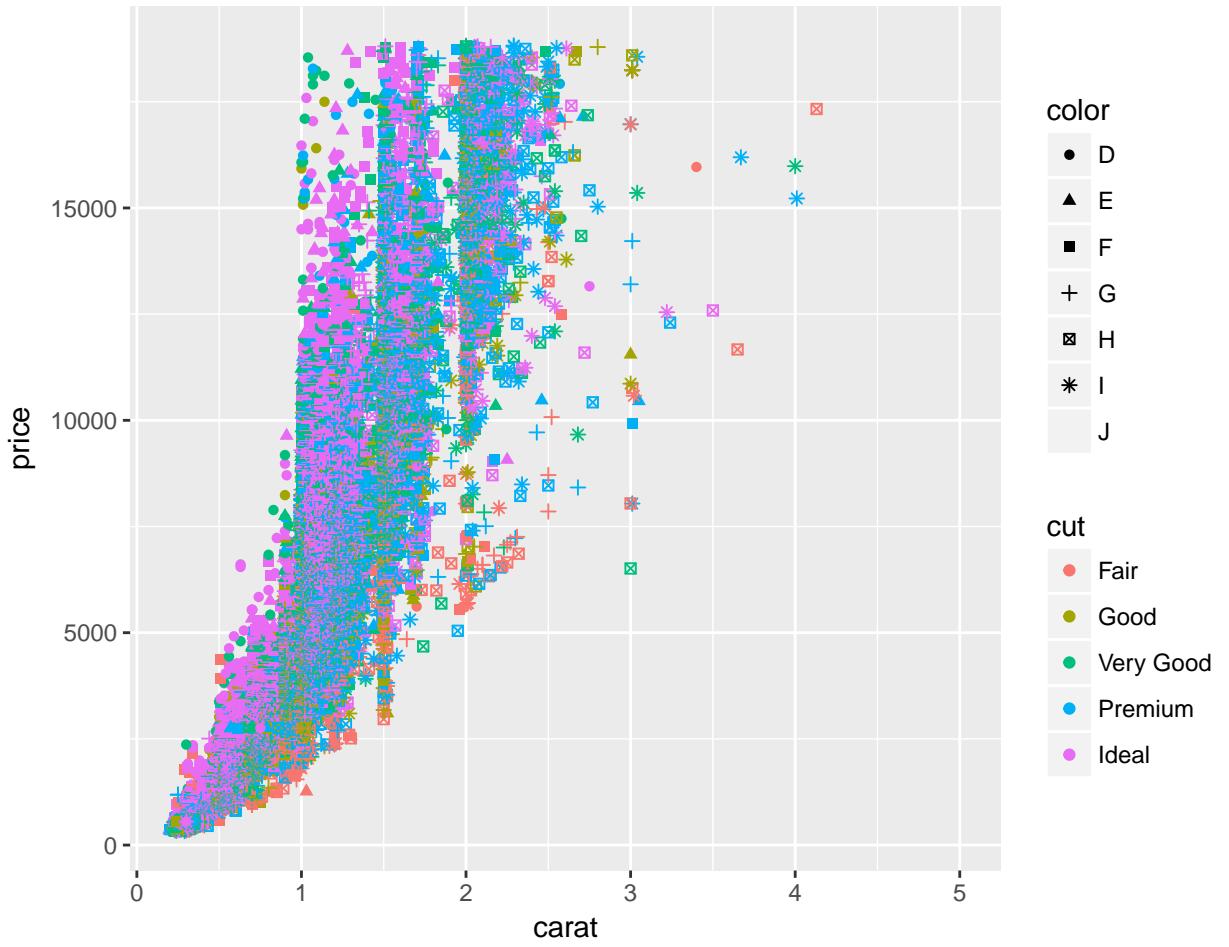


```
# o más sencillo
ggplot(diamonds, aes(x=carat, y=price)) +
  geom_point(aes(color=cut)) +
  geom_smooth()
```



Ahora vamos a hacer que la forma de los puntos cambie con la característica “color” del conjunto de datos.

```
ggplot(diamonds, aes(x=carat, y=price, color=cut, shape=cut)) +  
  geom_point()
```



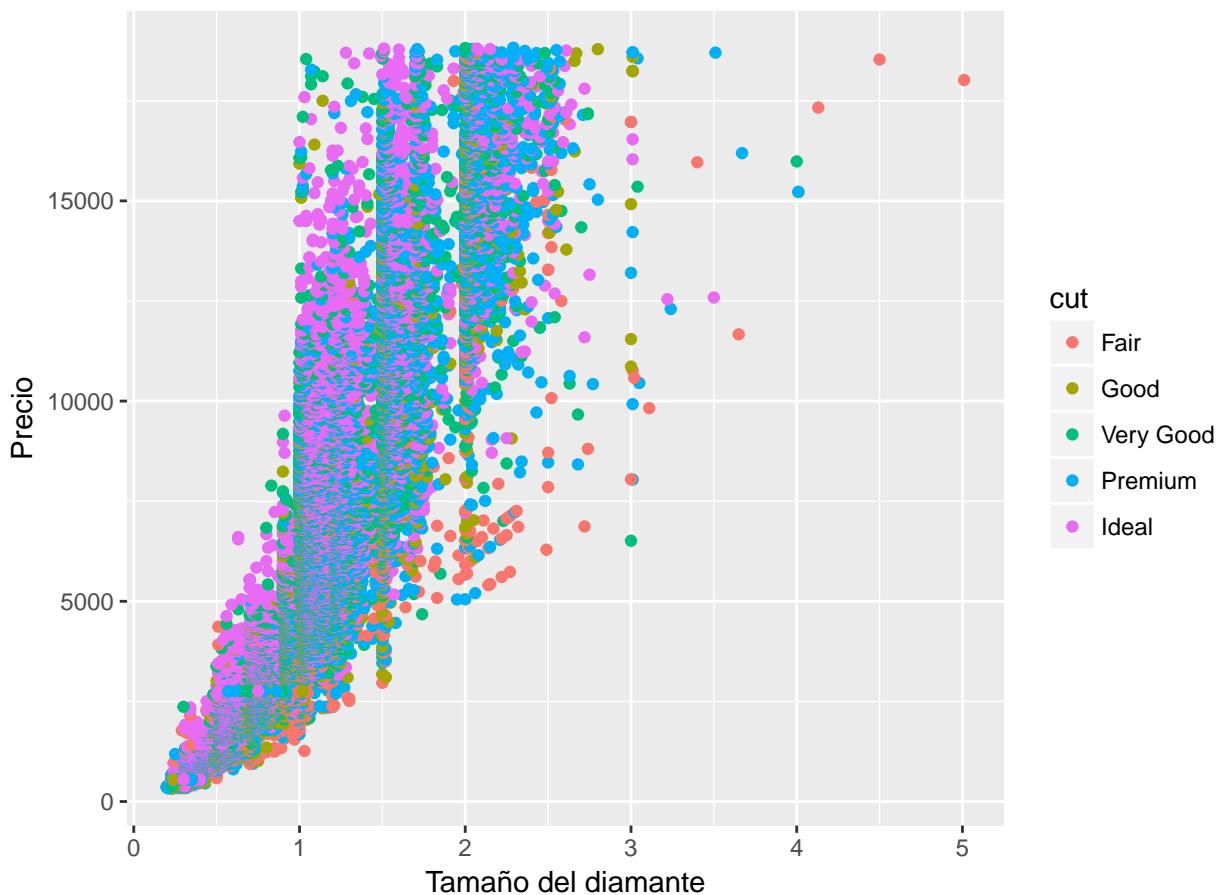
Las etiquetas (labs)

Las **etiquetas** ayudan a visualizar lo que queremos representar en un gráfico. Las más comunes son el **título principal** del gráfico, los **títulos de los ejes** y las **leyendas**. Para ello utilizamos la capa `labels`, destinada a especificar las etiquetas. Sin embargo, la manipulación del tamaño, el color de las etiquetas es trabajo del `theme` que veremos más adelante.

```
gg <- ggplot(diamonds, aes(x=carat, y=price, color=cut)) +
  geom_point() +
  labs(title="Diagrama de puntos", x="Tamaño del diamante", y="Precio")
```

gg

Diagrama de puntos



NOTA: Fíjate que podemos guardar el gráfico en un objeto (aquí llamado `gg`) y luego pedir visualizarlo.

El Tema (`theme`)

Lo único que queda por especificar es el tamaño de las etiquetas y el título de la leyenda.

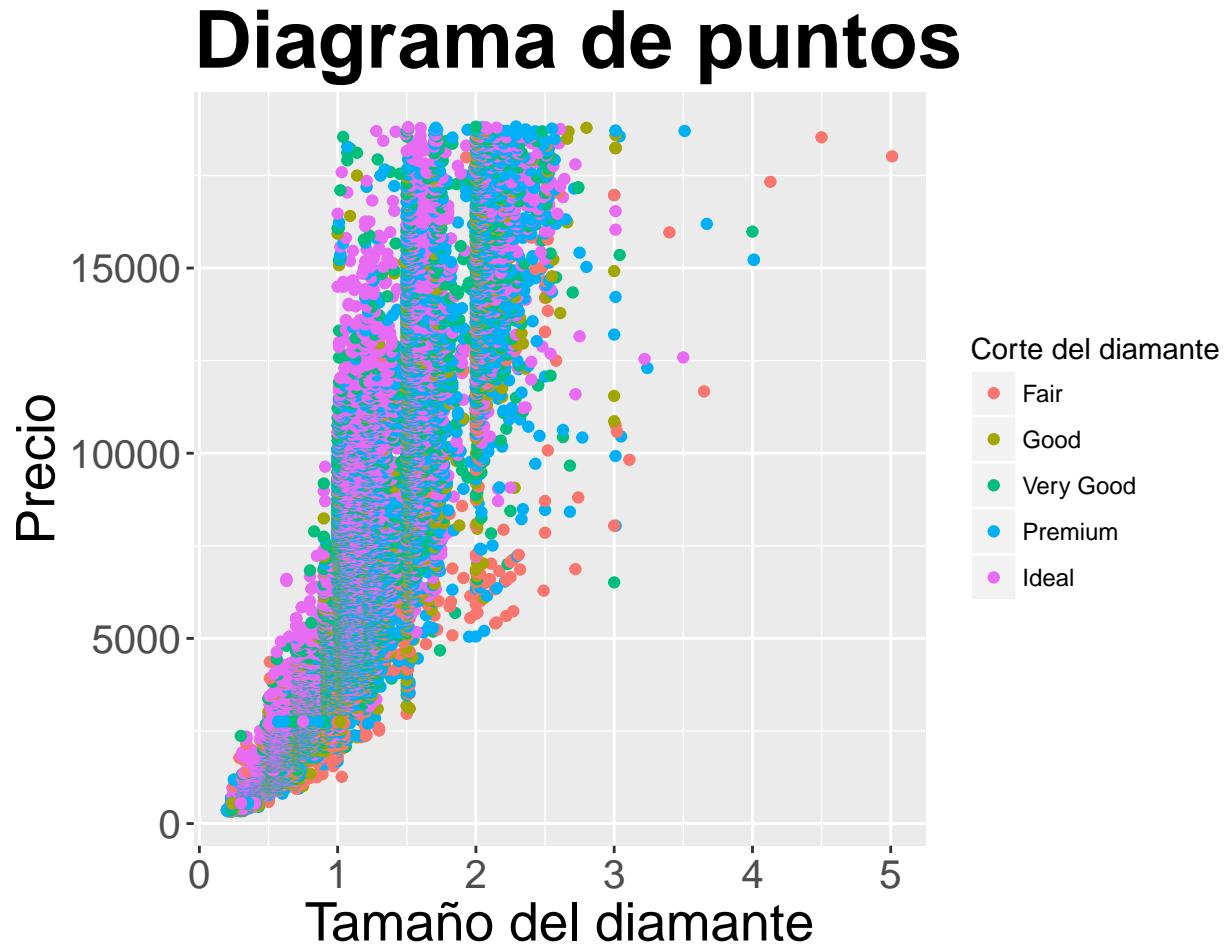
El ajuste del **tamaño de las etiquetas** se puede hacer usando la función `theme()` configurando `plot.title`, `axis.text.x` y `axis.text.y`, que deben ser especificados dentro del `element_text()`.

Ajustar el **título de la leyenda** es un poco complicado. Si la leyenda es la de un atributo de color y varía en función de un factor, debe establecer el nombre utilizando `scale_color_discrete()`, donde la parte de color pertenece al atributo de color y discreta porque la leyenda se basa en una variable de factor.

Vamos a continuar con nuestro ejemplo anterior que lo habíamos guardado en el objeto `gg` y cambiamos el tamaño de las etiquetas y el título de la leyenda.

```
gg1 <- gg +
  theme(plot.title=element_text(size=30, face="bold"),
        axis.text.x=element_text(size=15),
        axis.text.y=element_text(size=15),
        axis.title.x=element_text(size=20),
        axis.title.y=element_text(size=20)) +
  scale_color_discrete(name="Corte del diamante")
```

```
gg1 # imprimir el nuevo plot
```



Otros comandos para cambiar el **título de la leyenda** son:

- **scale_shape_discrete(name = "título leyenda")**: Si la leyenda muestra un atributo de forma basado en una variable de factor.
- **scale_shape_continuous(name = "título leyenda")**: Si es un atributo de forma basado en una variable continua.
- **scale_fill_continuous(name = "título leyenda")**: Si se basa en un atributo de relleno en una variable continua.

Si escribes **scale_** en la linea de la **consola** y al final presionas la tecla de tabulación (“Tab”) podrás desplazarte con las flechas hacia arriba y hacia abajo viendo todas las opciones posibles relacionadas con esta función. Al posicionarte encima de cada función obtendrás una descripción que explica cada una de las opciones.

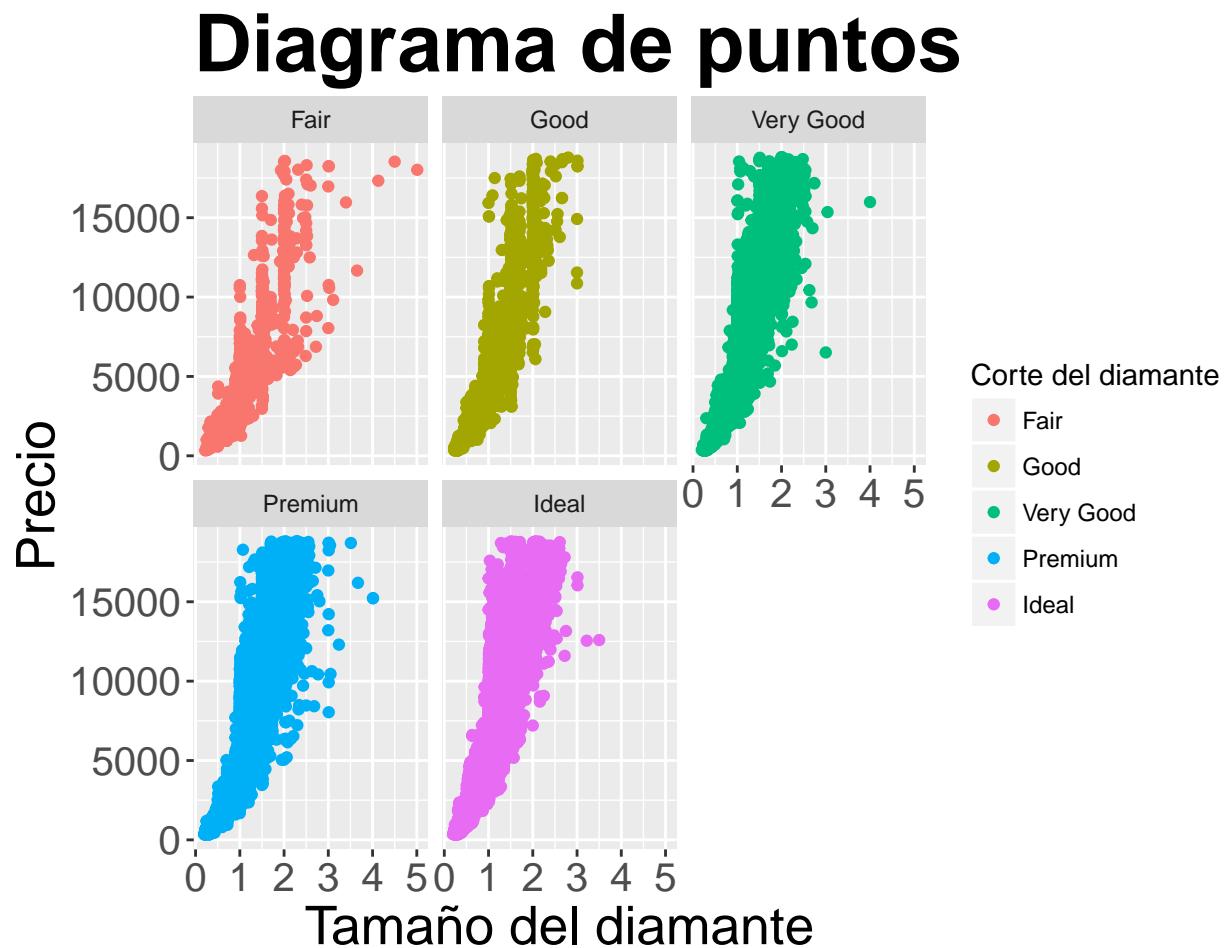
El aspecto (facet)

En el gráfico anterior, veíamos un diagrama de dispersión para todos los diferentes valores de corte trazado en el mismo gráfico. Si lo que preferimos es ver un gráfico separago por grupos (e.g. para cada uno de los tipos de **cut**) debemos utilizar la opción **facet_wrap(fórmula)**.

facet_wrap(fórmula) toma una fórmula como argumento. Los elementos que coloquemos a la derecha corresponden a las columnas, y los que coloquemos a la izquierda definen las filas.

Por ejemplo, si definimos las columnas por ‘cut’.

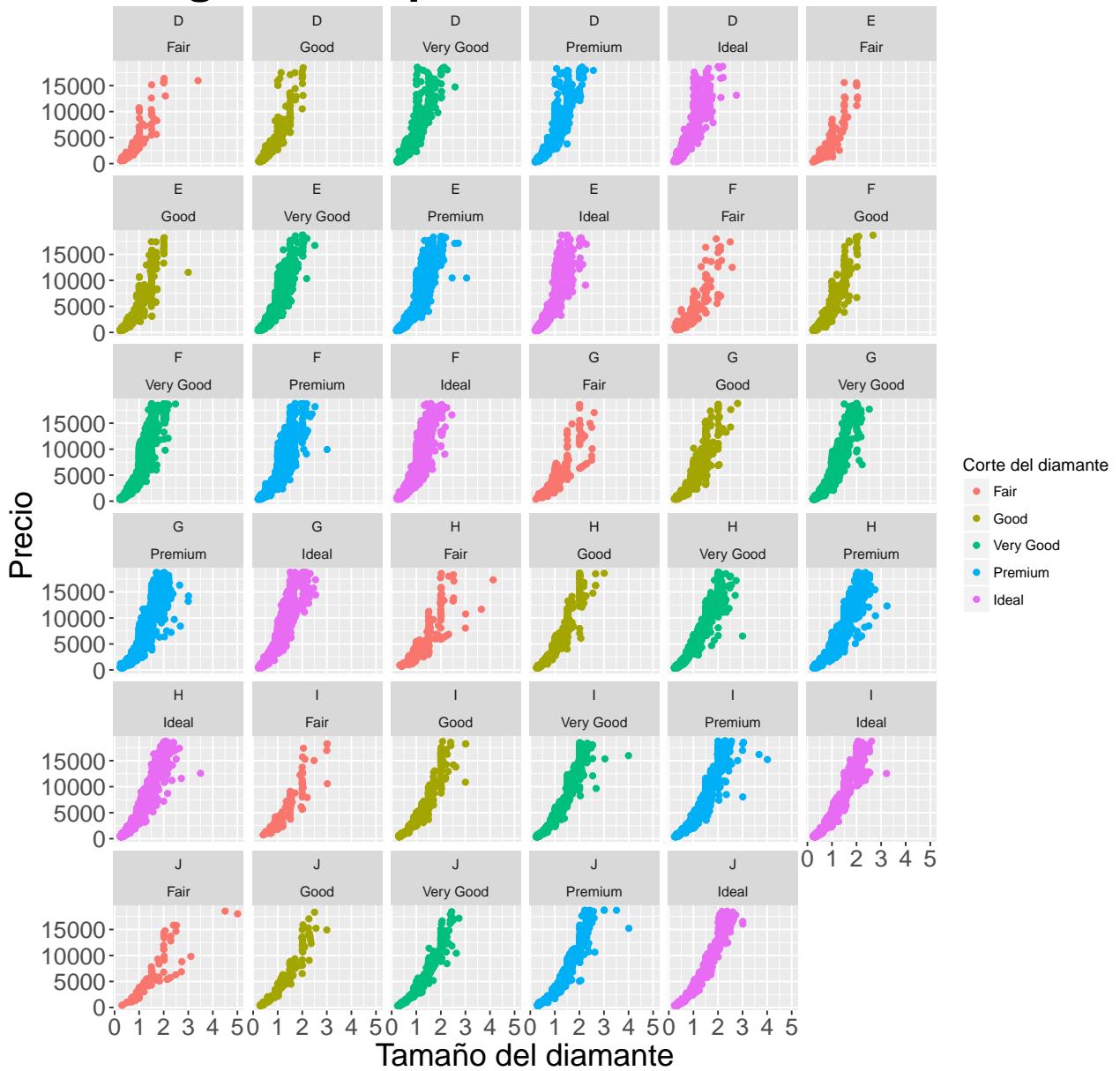
```
gg1 +
  facet_wrap(~ cut, ncol=3)
```



Si definimos las filas por “color” y columnas “cut”.

```
gg1 +
  facet_wrap(color ~ cut)
```

Diagrama de puntos



En `facet_wrap`, las escalas de los ejes X e Y se fijan para acomodar todos los puntos de forma predeterminada. Esto haría que la comparación de los atributos sea sencilla porque estarían en la misma escala. Sin embargo, es posible hacer que las escalas sean libres haciendo que las gráficas parezcan distribuidas más uniformemente estableciendo el argumento `scales = free`.

```
gg1 +
  facet_wrap(color ~ cut, scales="free")
```

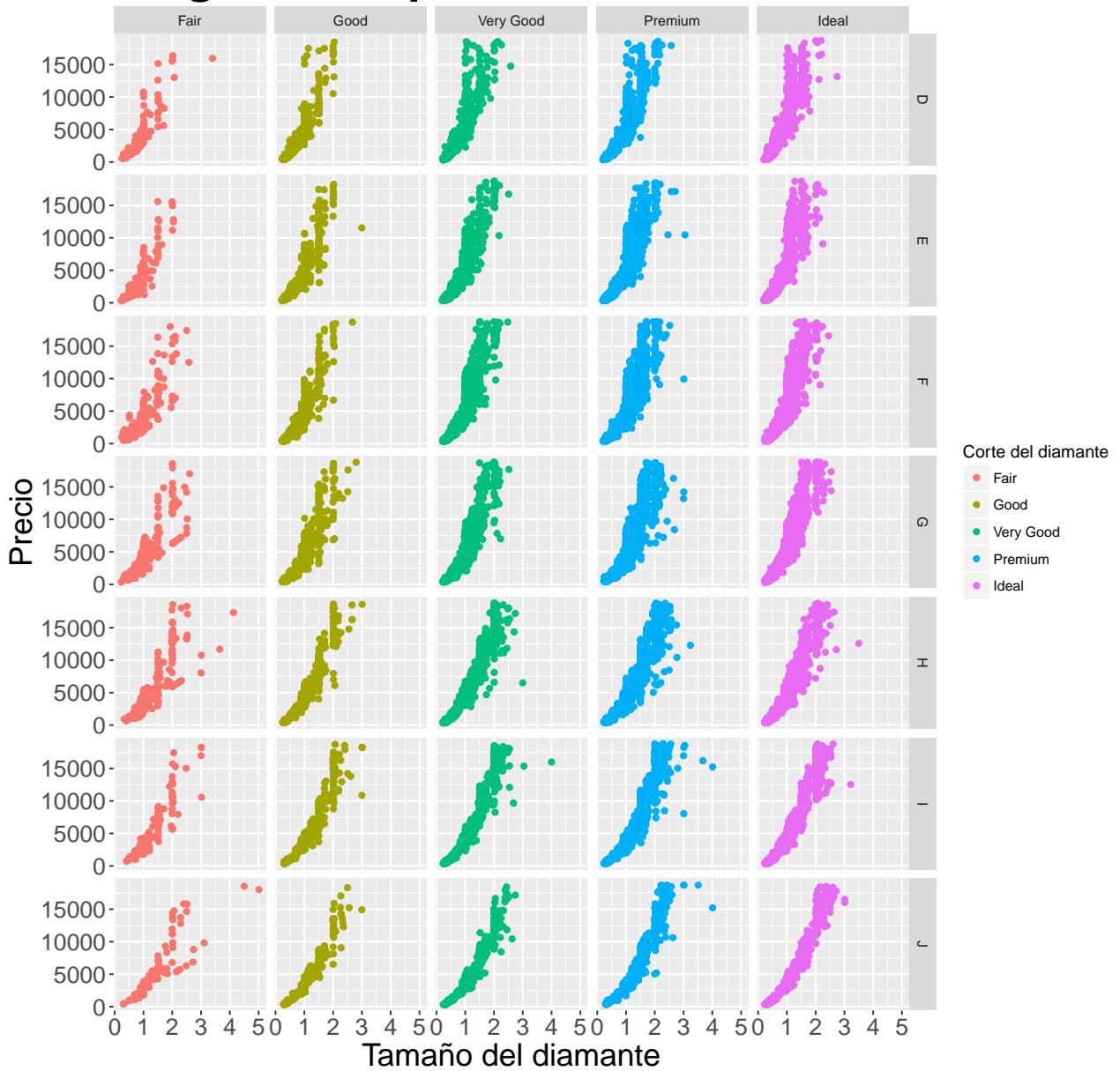
Diagrama de puntos



Para establecer mejor las comparaciones entre los gráficos, se pueden poner todas las parcelas en una cuadrícula usando `facet_grid(fórmula)`.

```
gg1 +
  facet_grid(color ~ cut)
```

Diagrama de puntos



Ya conocemos la estructura fundamental de los gráficos en `ggplot2`, ahora vamos a ver algunas características que complementan lo que ya sabemos.

La leyenda: eliminación y cambio de posición.

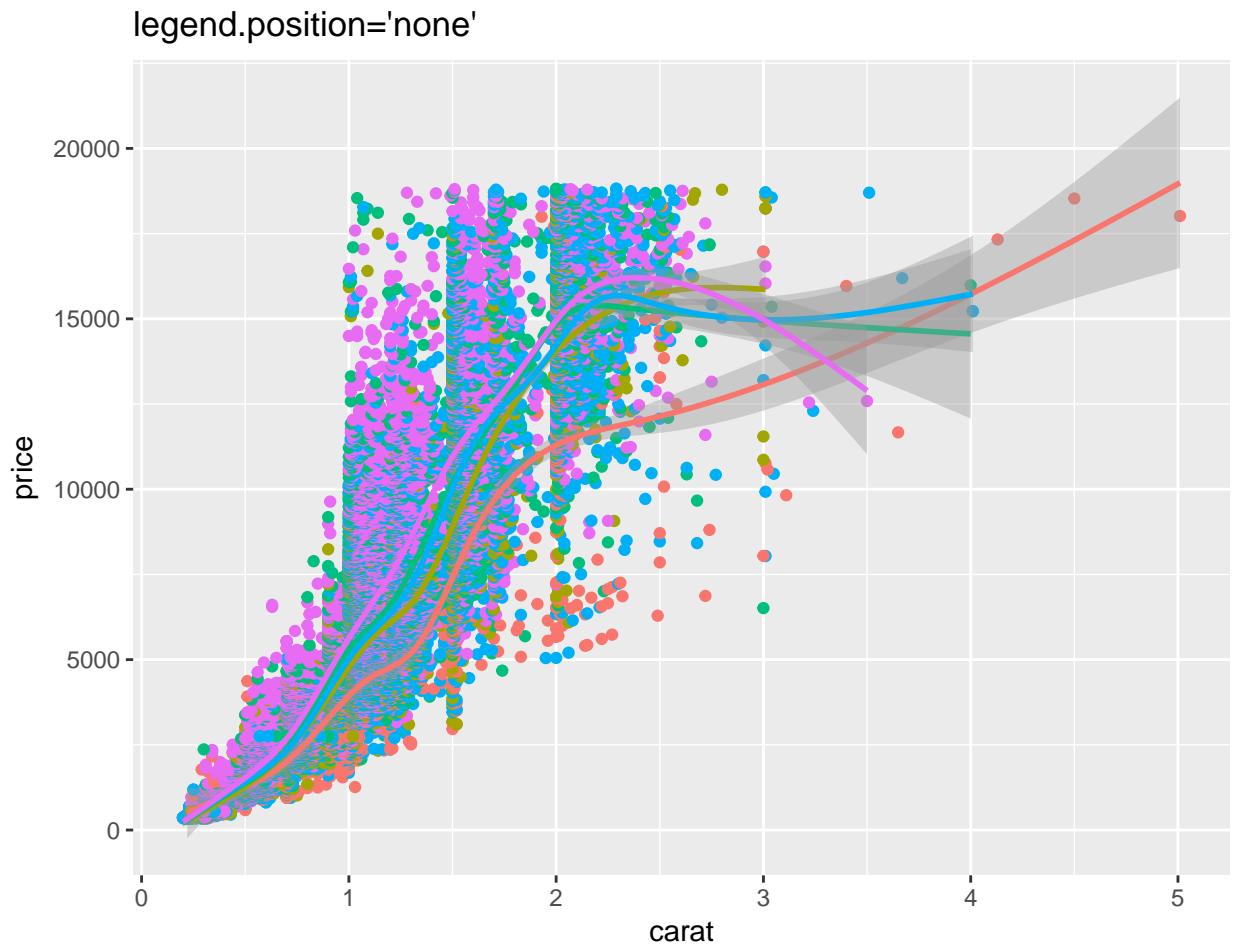
Si introducimos `theme(legend.position = "none")`, se puede quitar la leyenda. Mediante `theme(legend.position = "top")` se puede mover la leyenda alrededor de la trama. Por ejemplo, podemos establecerlo en el margen superior ('top'). Al establecer `legend.position` en una coordenada dentro de la trama, se puede colocar la leyenda dentro del gráfico.

`legend.justification` denota el punto de anclaje de la leyenda, es decir, el punto que se colocará en las coordenadas dadas por `legend.position`.

Ejemplo sin leyenda:

```
p1 <- ggplot(diamonds, aes(x=carat, y=price, color=cut)) +
  geom_point() +
  geom_smooth() +
  theme(legend.position="none") +
  labs(title="legend.position='none'")
```

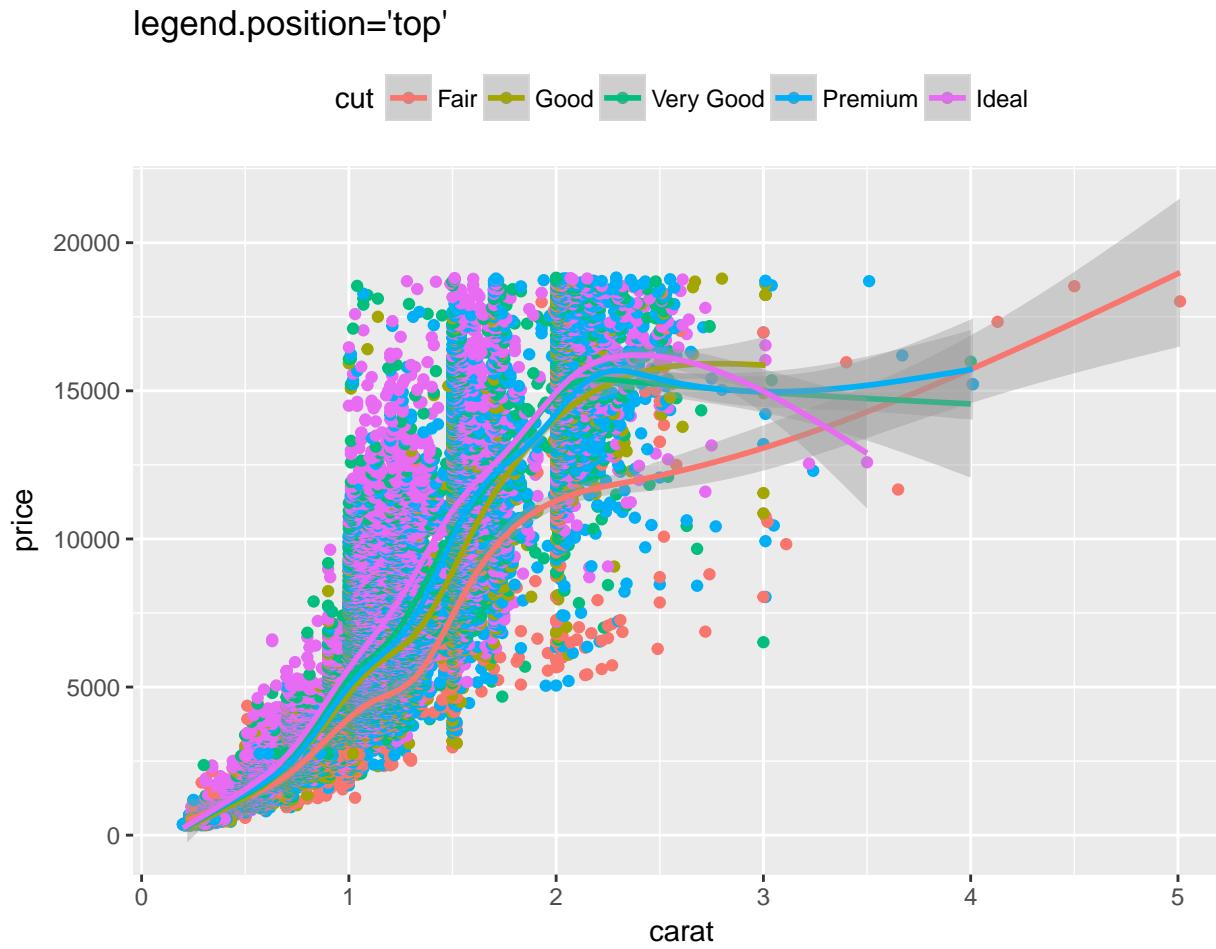
p1



Leyenda arriba:

```
p2 <- ggplot(diamonds, aes(x=carat, y=price, color=cut)) +
  geom_point() +
  geom_smooth() +
  theme(legend.position="top") +
  labs(title="legend.position='top'")
```

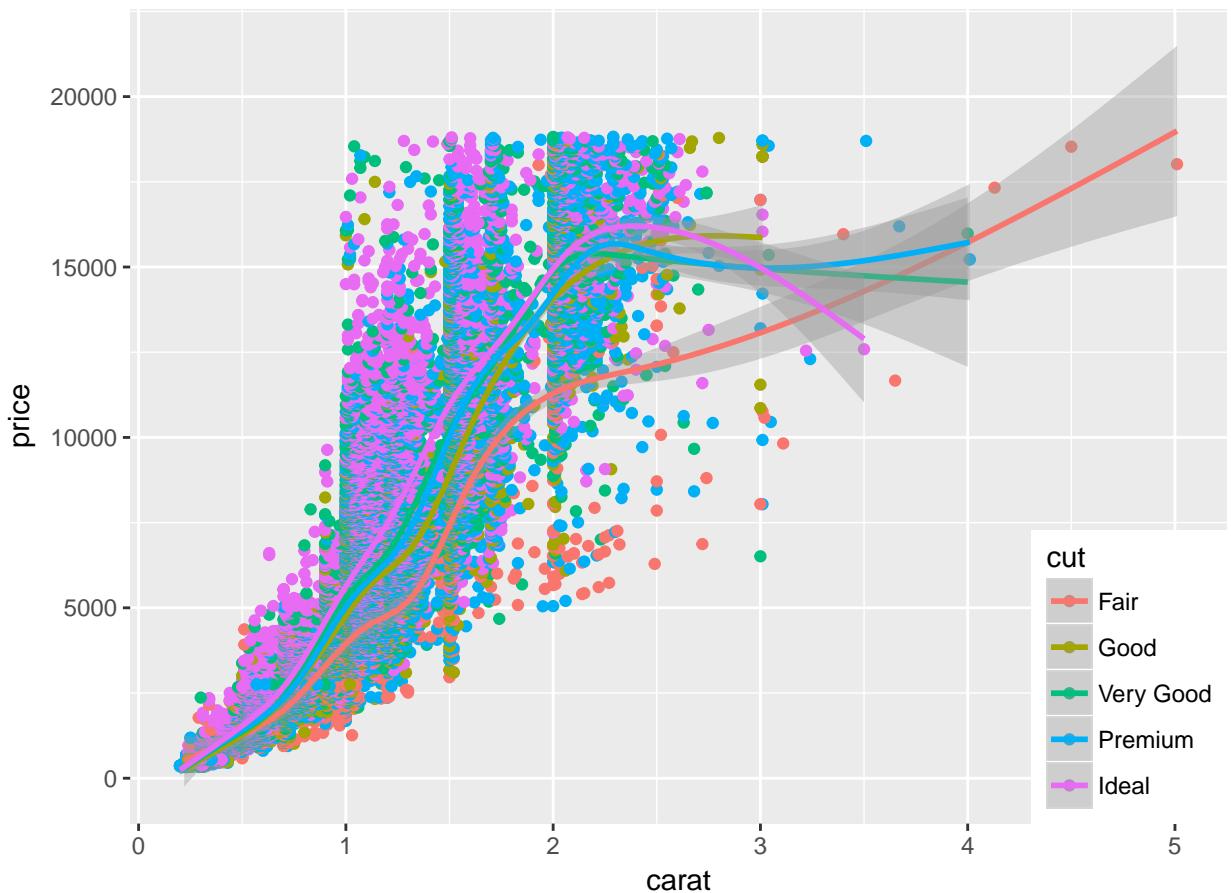
p2



Leyenda dentro del gráfico:

```
ggplot(diamonds, aes(x=carat, y=price, color=cut)) +
  geom_point() +
  geom_smooth() +
  labs(title="legend.position='coords inside plot'") +
  theme(legend.justification=c(1,0), legend.position=c(1,0))
```

legend.position='coords inside plot'

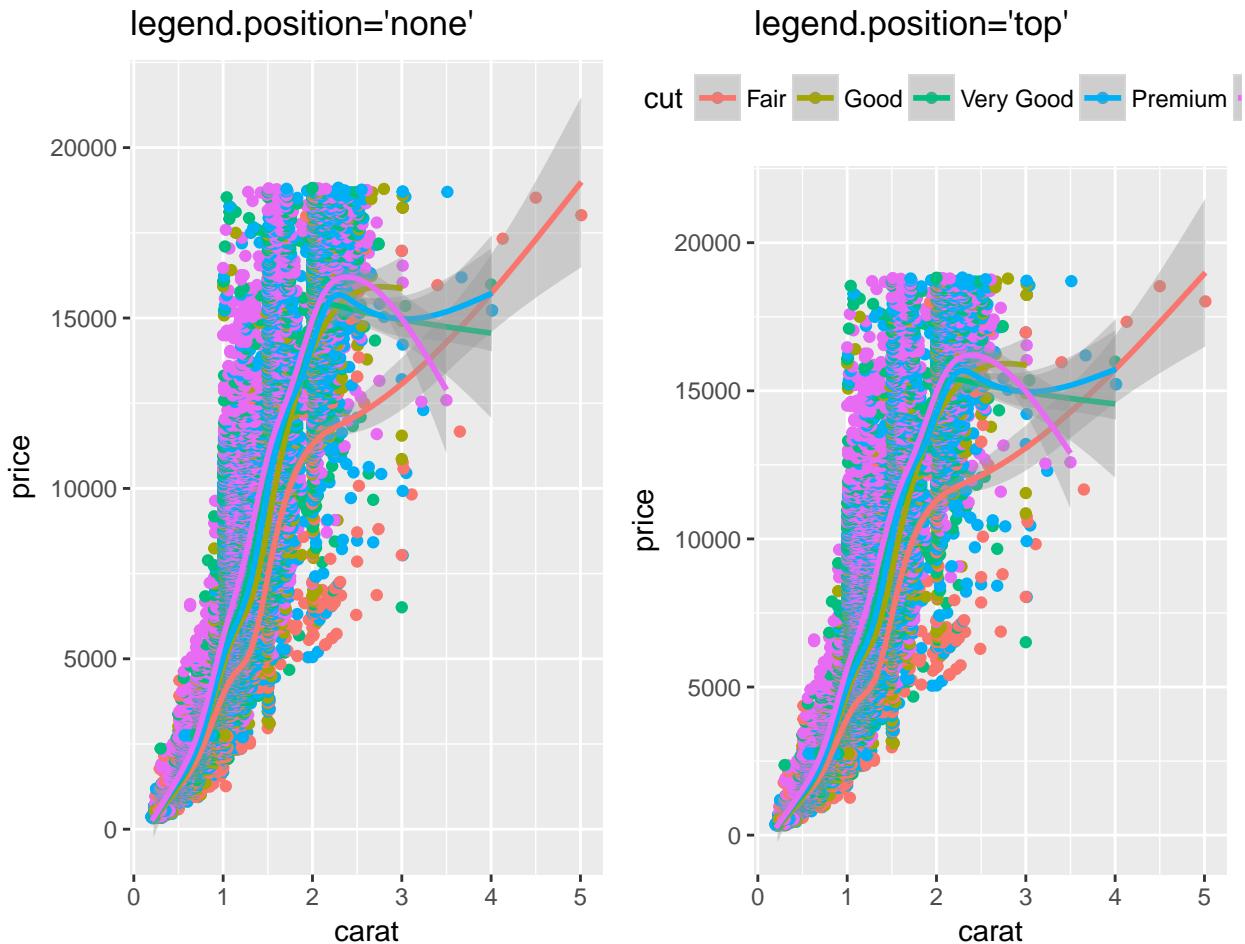


Unir varios gráficos en una misma figura

Antes hemos guardado dos graficos como objetos “p1” y “p2”, para que aparezcan en una misma imagen sólo tenemos que utilizar la función `grid.arrange()` que se encuentra en el paquete `gridExtra`.

```
library(gridExtra)
grid.arrange(p1, p2, ncol=2)
```

```
## `geom_smooth()` using method = 'gam'
## `geom_smooth()` using method = 'gam'
```



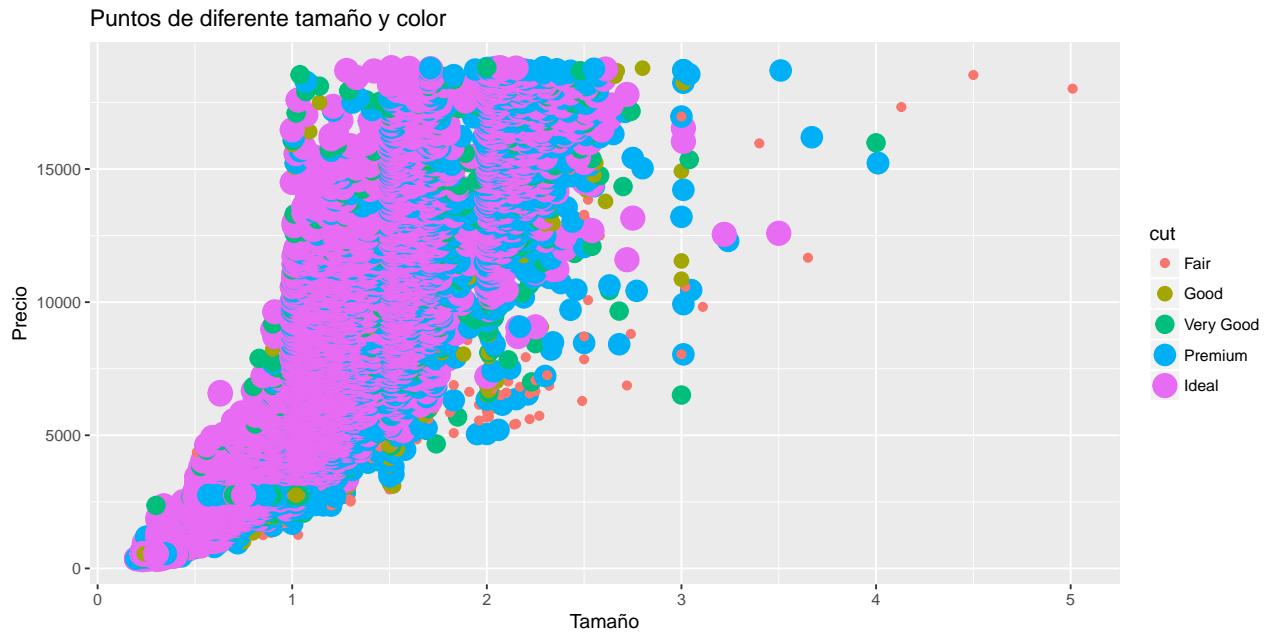
Tipos de gráficos

Diagrama de dispersión

El diagrama de dispersión (o nube de puntos) representa pares de valores como las coordenadas de un punto. Sobre la nube de puntos puede trazarse una recta, o cualquier tipo de función, que se ajuste a ellos lo mejor posible. Es la representación gráfica más útil para describir el comportamiento conjunto de dos variables numéricas.

Continuamos con los datos `diamonds`. Pero ahora vamos a cambiar el tamaño y el color de los puntos en función de `cut`.

```
ggplot(diamonds,aes(x=carat,y=price)) +
  geom_point(aes(size=cut, color=cut)) +
  labs(title="Puntos de diferente tamaño y color",
       x="Tamaño",
       y="Precio")
```



Curvas de ajuste

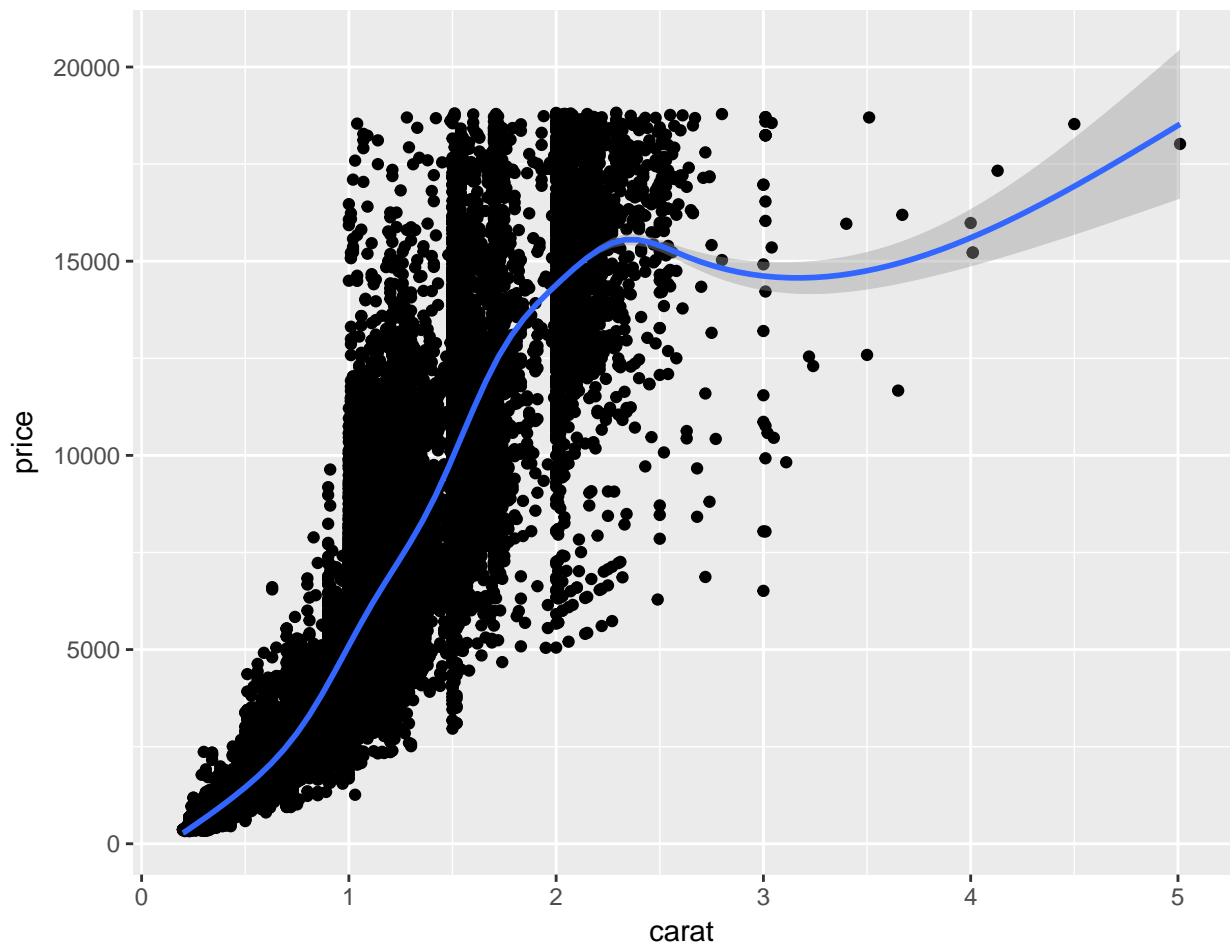
Si queremos incluir **curvas de ajuste** que muestren la tendencia general de los datos podemos utilizar la función `geom_smooth`.

1. Agregar una curva de suavizado con región de confianza al 95% (opción por defecto).

El área gris alrededor de la curva es un intervalo de confianza que indica cuánta incertidumbre (error) hay en esta curva de suavizado.

```
ggplot(diamonds, aes(x=carat, y=price)) +
  geom_point() +
  geom_smooth()

## `geom_smooth()` using method = 'gam'
```

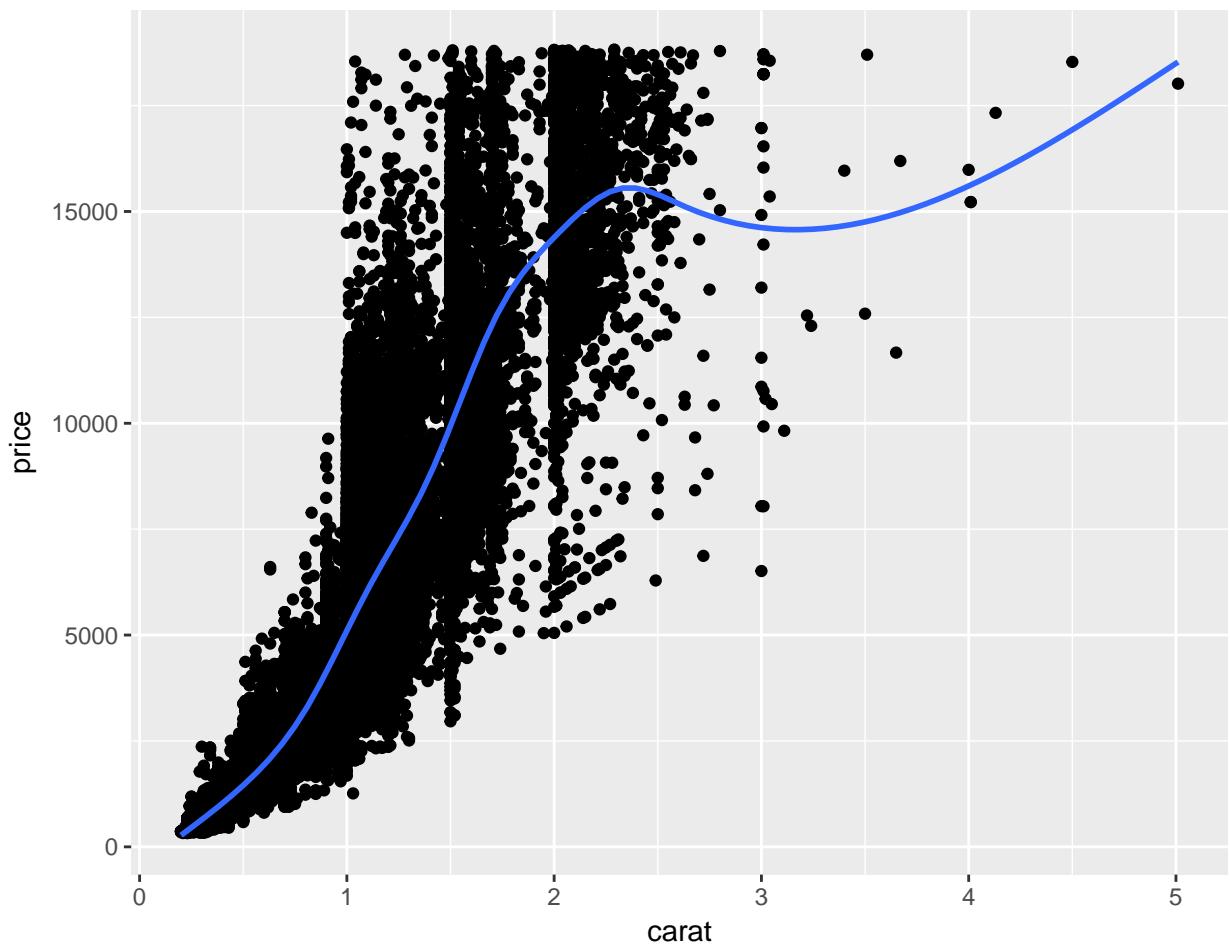


2. Si queremos quitar la región de confianza.

Si queremos desactivar el intervalo de confianza, podemos agregar la opción “se = FALSE” al `geom_smooth`, donde `se` significa error estándar (*standard error*).

```
ggplot(diamonds, aes(x=carat, y=price)) +
  geom_point() +
  geom_smooth(se=FALSE)

## `geom_smooth()` using method = 'gam'
```

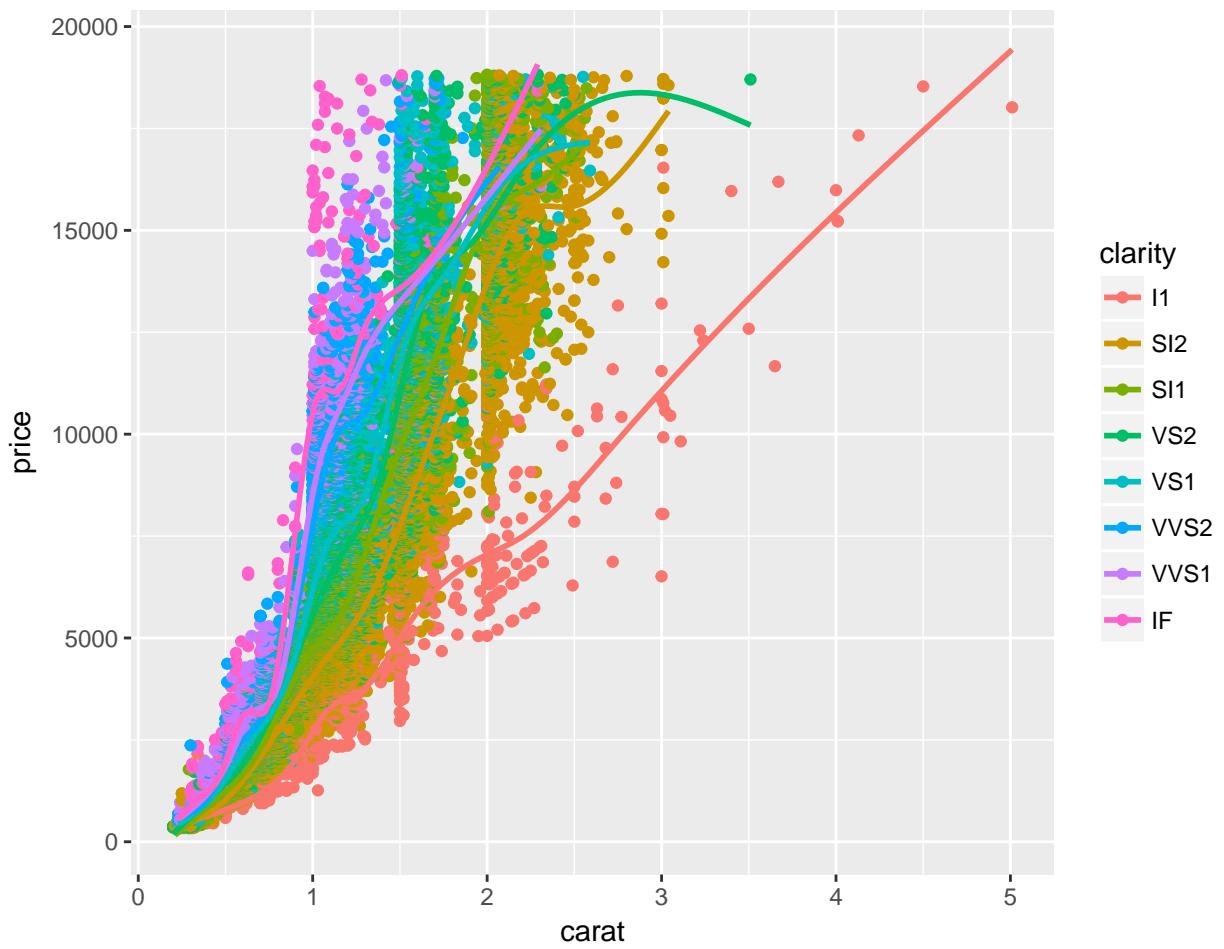


3. Realizar una curva de suavizado con grupo.

Si utilizamos una estética de color, `ggplot` creará una curva de suavizado para cada color. Por ejemplo, si agregamos “color = clarity” obtenemos:

```
ggplot(diamonds, aes(x=carat, y=price, color=clarity)) +
  geom_point() +
  geom_smooth(se=FALSE)

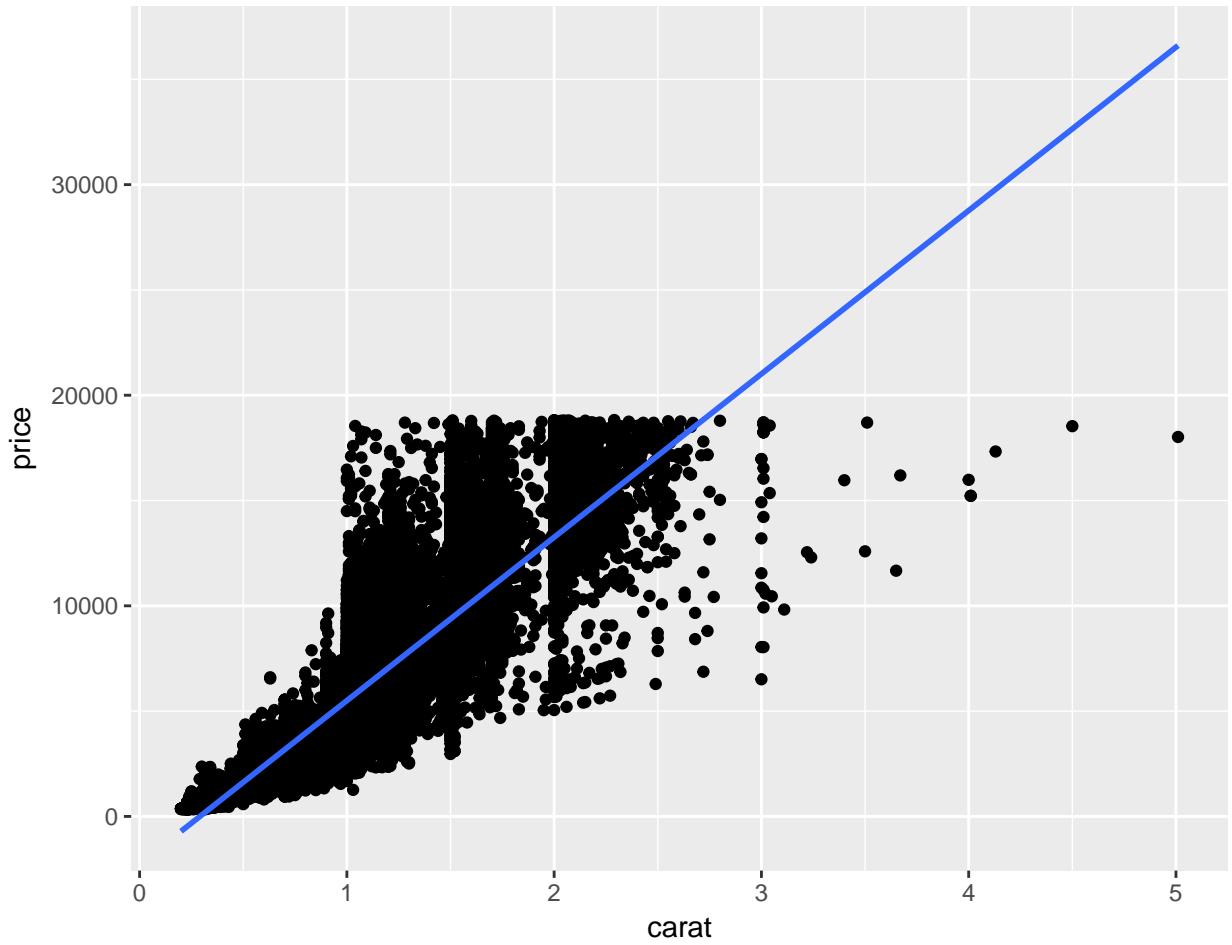
## `geom_smooth()` using method = 'gam'
```



4. Utilizar una recta.

Usamos ‘method = x’ para cambiar el método de suavizado, en este caso “lm” indica el modelo lineal (*linear model*).

```
ggplot(diamonds, aes(x=carat, y=price)) +
  geom_point() +
  geom_smooth(se=FALSE, method="lm")
```



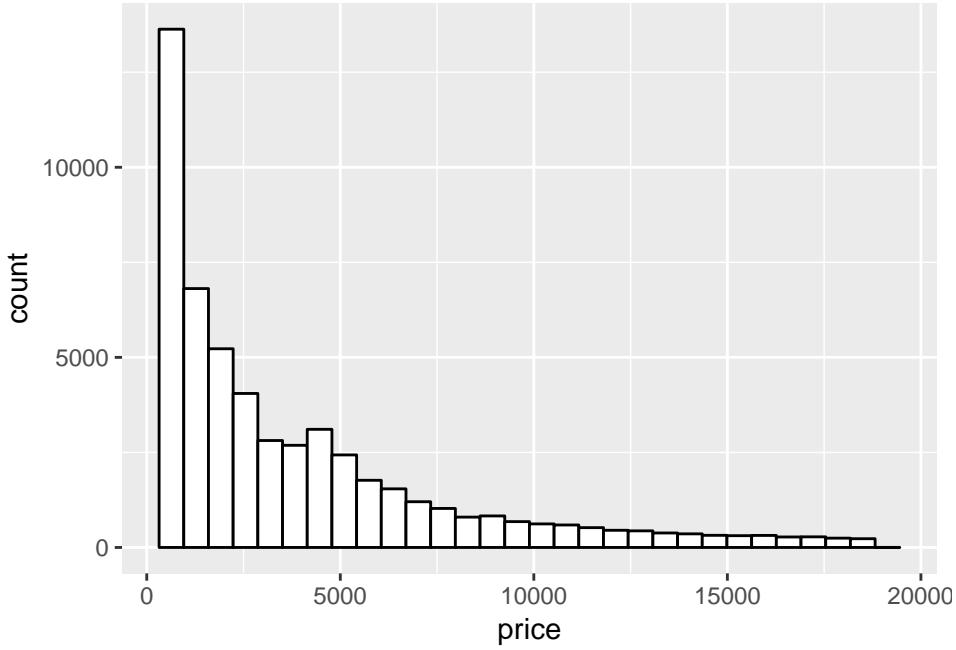
NOTA: más adelante en el curso veremos lo que es una curva de suavizado (en el tema 6).

Histograma

El histograma nos permite representar las distribuciones de frecuencias mediante barras, donde la superficie de cada barra es proporcional a la frecuencia de los valores representados. Así podemos obtener una idea de la distribución de la muestra, respecto a una característica, cuantitativa y continua (como la longitud o el peso).

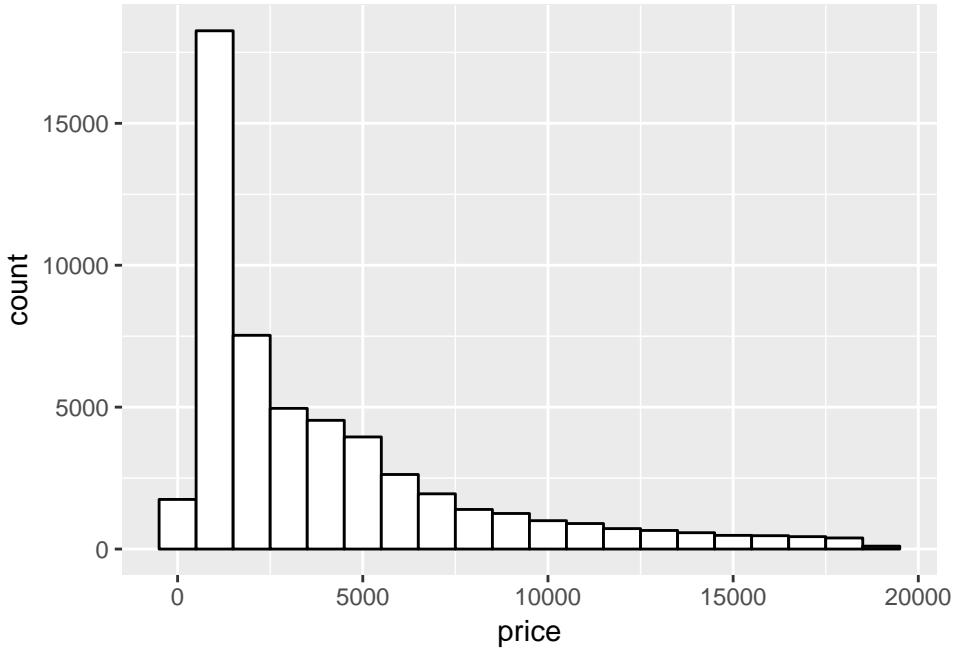
Realizamos un histograma del precio en el que indicamos que el relleno de las barras debe ser de color blanco y los bordes negros.

```
ggplot(diamonds,aes(x=price)) +
  geom_histogram(fill="white",color="black")
```



Cambiamos la anchura de los intervalos dentro de la geometría con el argumento `binwidth`, al aumentar el número cada vez tenemos menos intervalos:

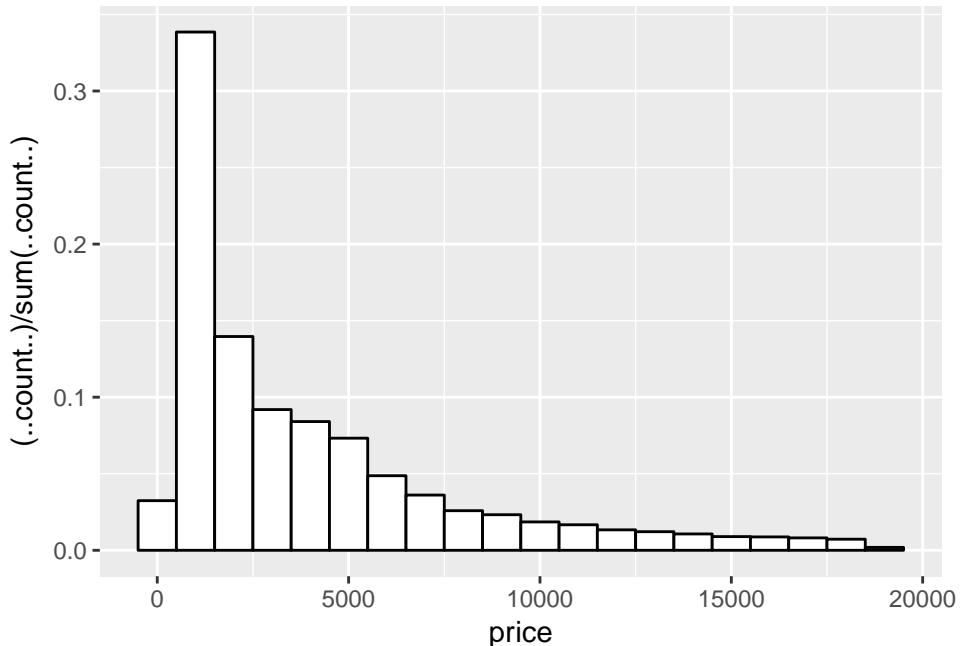
```
ggplot(diamonds,aes(x=price)) +
  geom_histogram(fill="white",color="black",binwidth=1000)
```



Los histogramas representan la **frecuencia absoluta**, si queremos que aparezca la **frecuencia relativa**, debemos especificarlo mediante `y = (.count..)/sum(..count..)`:

```
ggplot(diamonds,aes(x=price)) +
  geom_histogram(aes(y = (.count..)/sum(..count..)),
  fill="white",
  color="black",
```

```
binwidth=1000)
```

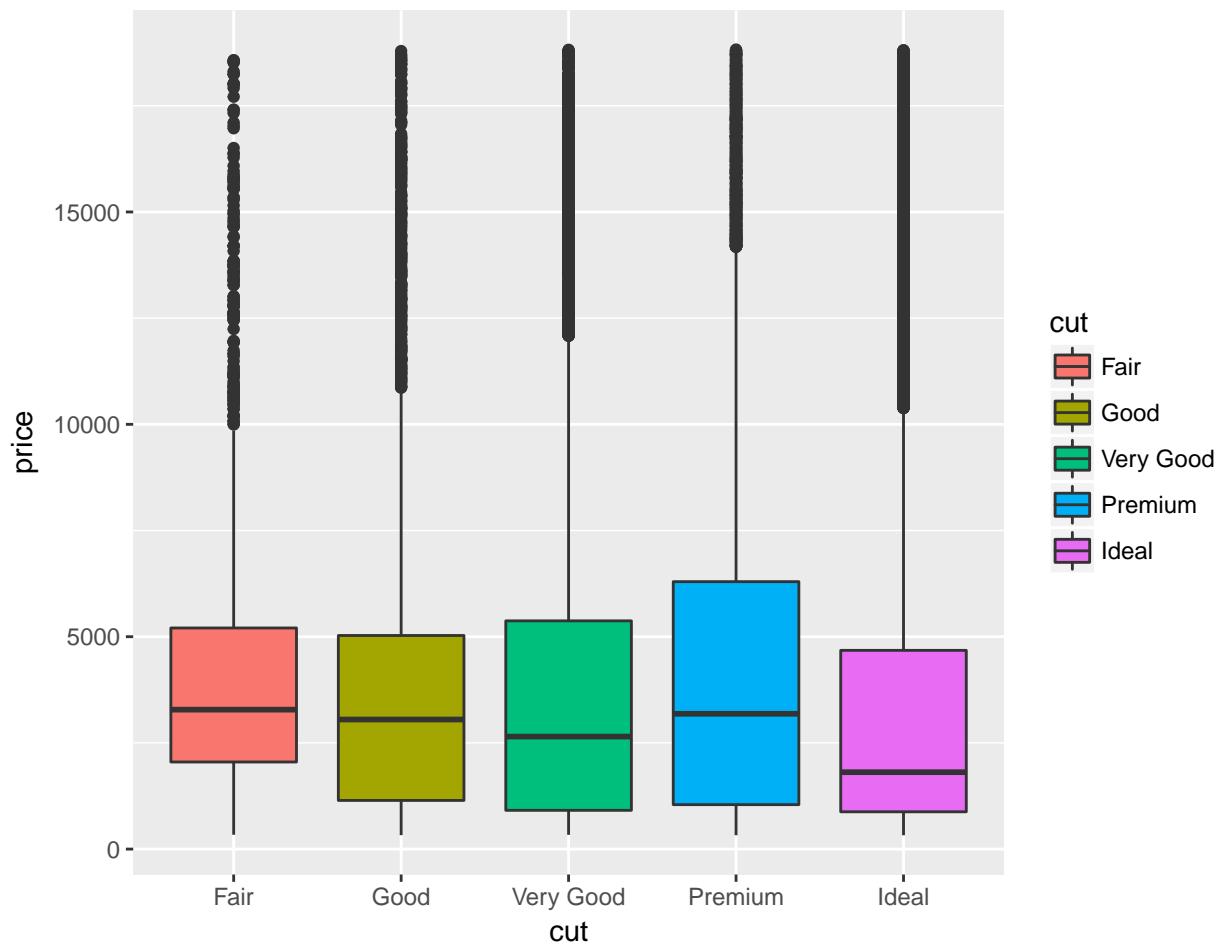


Diagramas de cajas

Un diagrama de caja es un diagrama que muestra la distribución estadística de un conjunto de datos. Los valores correspondientes al límite superior de la caja indican el percentil 75, el límite inferior corresponde al percentil 25 y la linea horizontal a la mediana o percentil 50. Es decir, la caja representa el 50% de los valores centrales. Las líneas verticales indican el valor máximo y mínimo observado. Los valores que quedan más allá de estas líneas son considerados valores atípicos (*outliers*).

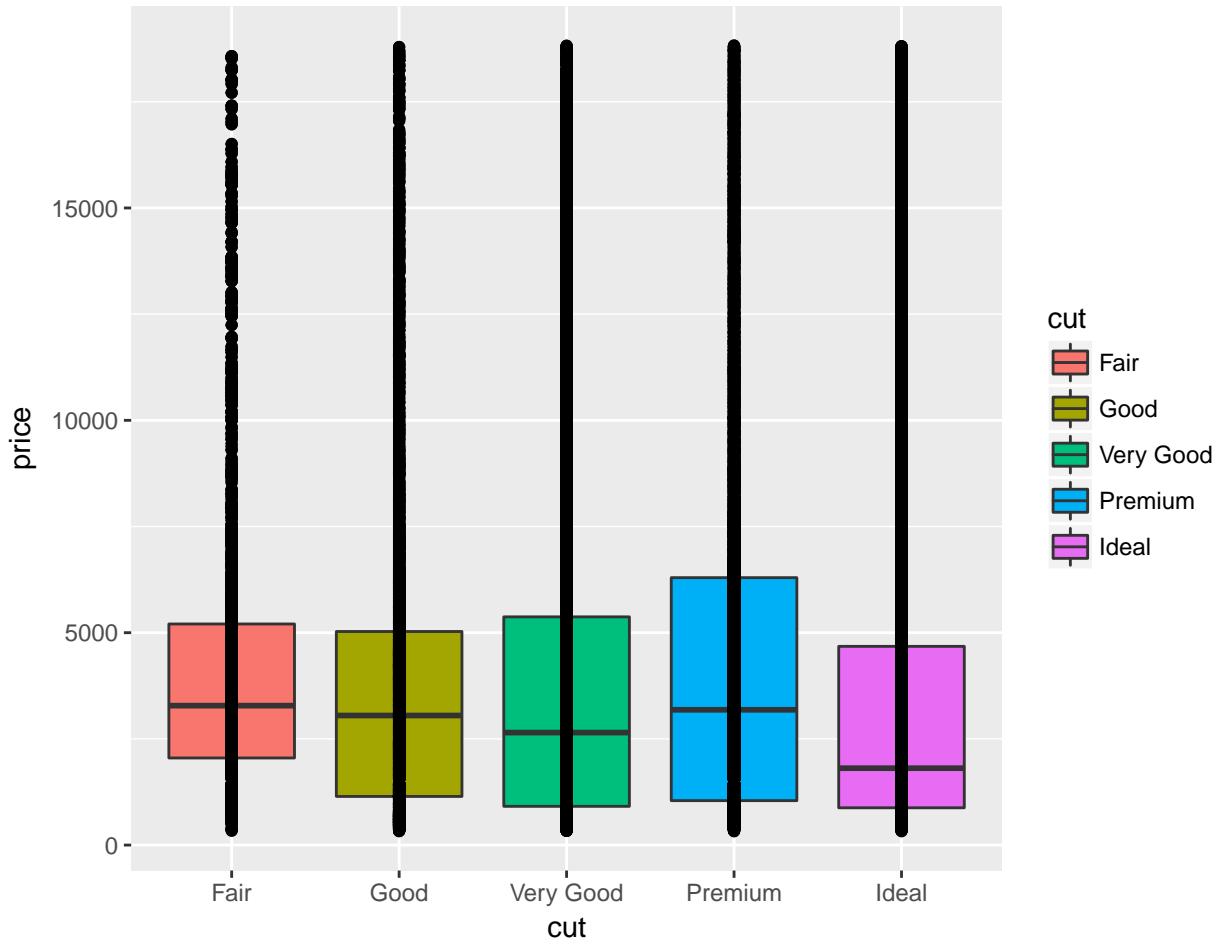
Veamos qué ocurre con nuestro ejemplo:

```
p <- ggplot(diamonds,aes(cut,price))
p +
  geom_boxplot(aes(fill=cut))
```



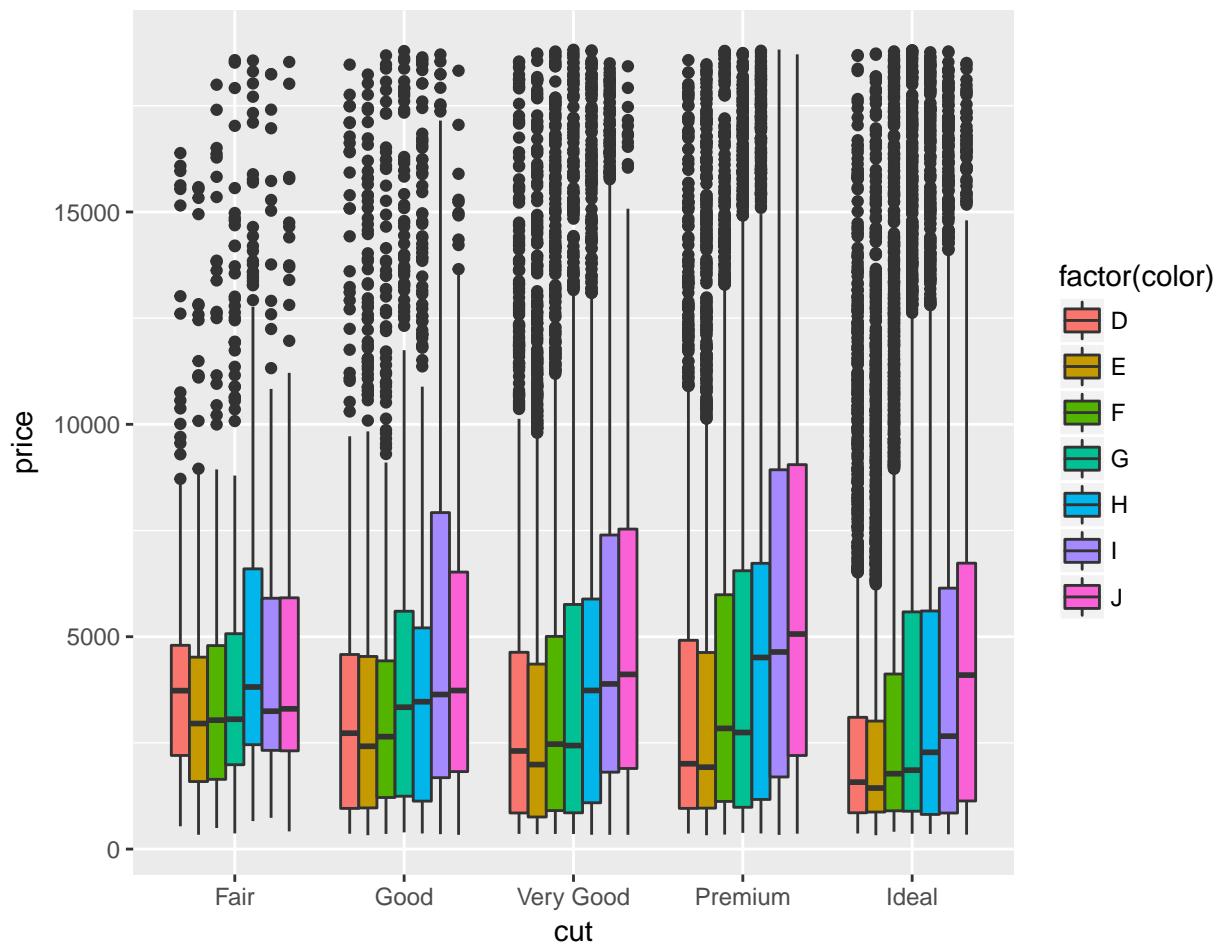
Podemos superponer los puntos correspondientes a las observaciones añadiendo `geom_point()`

```
p +  
  geom_boxplot(aes(fill=cut)) +  
  geom_point()
```



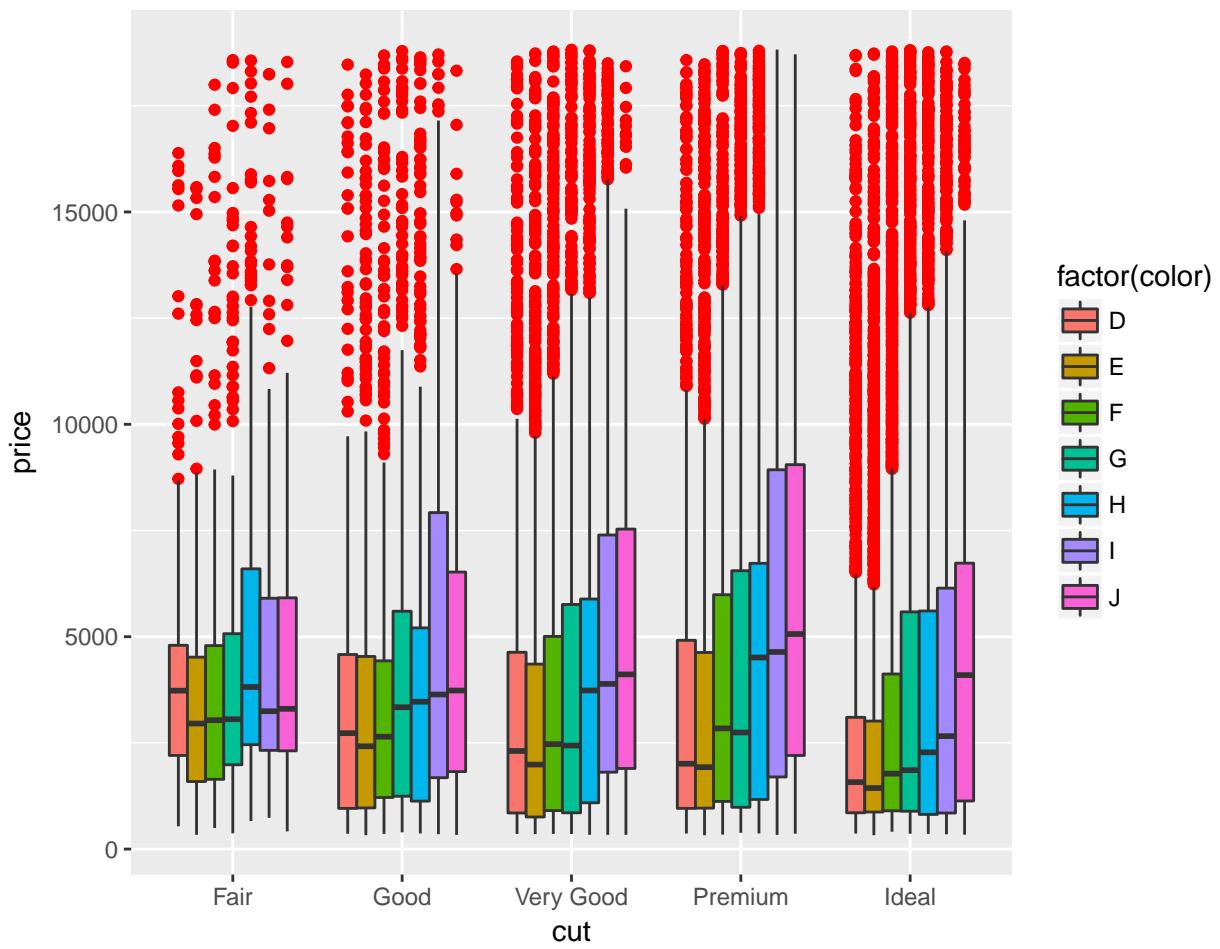
En muchos casos es interesante considerar distintos factores. En los datos del precio de los diamantes podemos considerar la calidad del corte `cut` y el color del diamante `color`. Para ello, basta con añadir una instrucción a la estética del boxplot.

```
p +
  geom_boxplot(aes(fill=factor(color)))
```



También podemos resaltar los outliers:

```
p +  
  geom_boxplot(aes(fill=factor(color)),  
               outlier.colour = "red",  
               outlier.size = 1.5)
```



Gráficos de barras

Los **gráficos de barras** son apropiados para representar las frecuencias absolutas de los valores de factores.

Por ejemplo, la base de datos `birthwt` contiene información acerca del peso de recién nacidos en función de distintas características de la madre.

```
library(MASS)
data(birthwt)
head(birthwt)

##   low age lwt race smoke ptl ht ui ftv bwt
## 85   0 19 182    2     0   0   0   1   0 2523
## 86   0 33 155    3     0   0   0   0   3 2551
## 87   0 20 105    1     1   0   0   0   1 2557
## 88   0 21 108    1     1   0   0   1   2 2594
## 89   0 18 107    1     1   0   0   1   0 2600
## 91   0 21 124    3     0   0   0   0   0 2622

str(birthwt)

## 'data.frame': 189 obs. of 10 variables:
## $ low : int  0 0 0 0 0 0 0 0 0 ...
## $ age : int  19 33 20 21 18 21 22 17 29 26 ...
## $ lwt : int  182 155 105 108 107 124 118 103 123 113 ...
## $ race: int  2 3 1 1 2 2 3 1 2 1 ...
## $ smoke: int  0 1 0 1 0 0 0 0 0 1 ...
## $ ptl : int  0 1 0 0 0 0 0 0 0 0 ...
## $ ht  : int  0 1 0 0 0 0 0 0 0 0 ...
## $ ui  : int  1 0 0 0 1 0 0 0 0 0 ...
## $ ftv : int  0 3 1 2 0 0 0 0 0 0 ...
## $ bwt : num  2523 2551 2557 2594 2600 2622 2580 2621 2597 2547 ...
```

```

## $ race : int  2 3 1 1 1 3 1 3 1 1 ...
## $ smoke: int  0 0 1 1 1 0 0 0 1 1 ...
## $ ptl  : int  0 0 0 0 0 0 0 0 0 0 ...
## $ ht   : int  0 0 0 0 0 0 0 0 0 0 ...
## $ ui   : int  1 0 0 1 1 0 0 0 0 0 ...
## $ ftv  : int  0 3 1 2 0 0 1 1 1 0 ...
## $ bwt  : int  2523 2551 2557 2594 2600 2622 2637 2637 2663 2665 ...

```

Podemos realizar una tabla de frecuencias que indique cuantos niños nacen con poco peso en relación a la raza de la madre (contar el número de medidas que hay en cada combinación).

```

with(birthwt,
  table(race,low))

```

```

##      low
## race 0 1
##   1 73 23
##   2 15 11
##   3 42 25

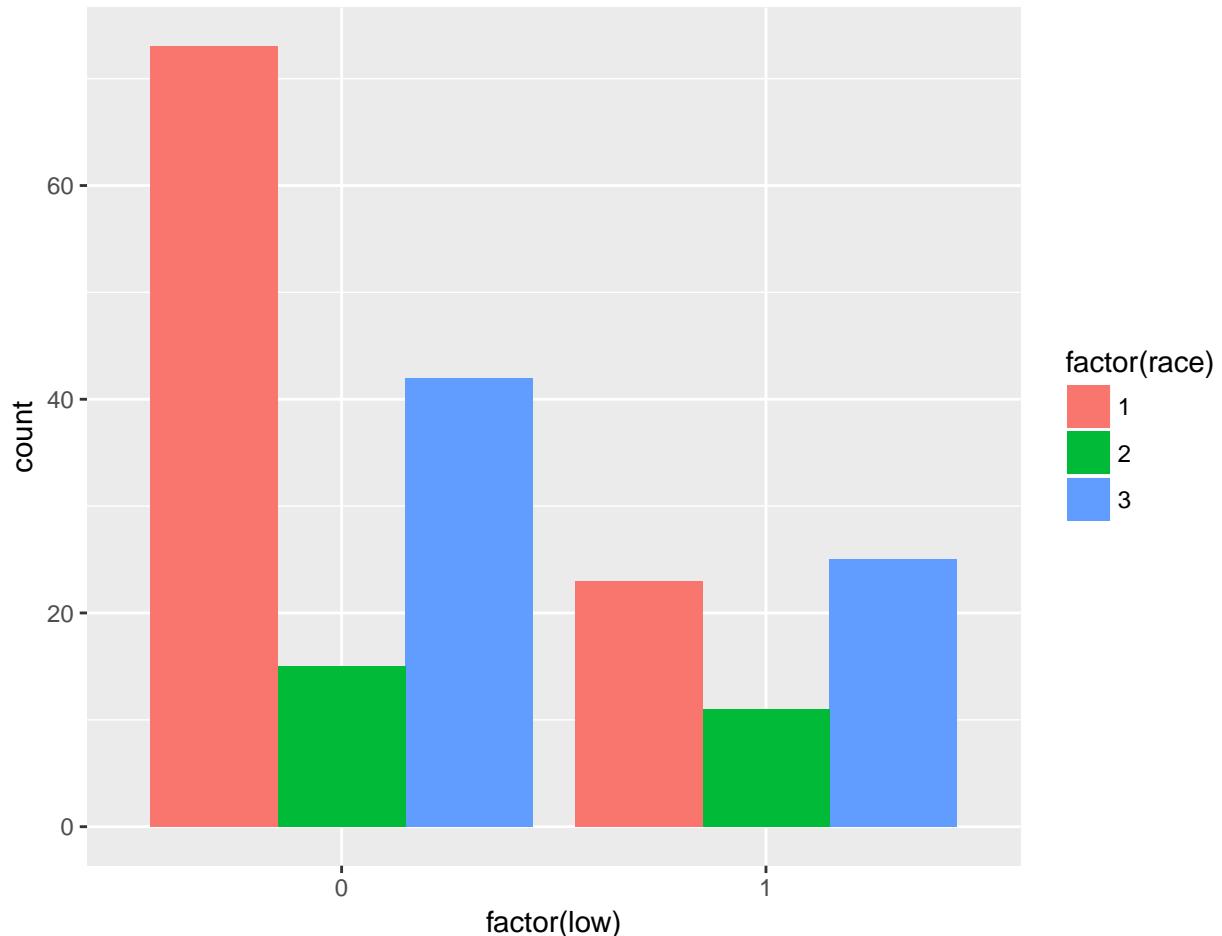
```

Podemos obtener la gráfica mediante el siguiente procedimiento:

```

ggplot(birthwt,aes(x=factor(low),fill=factor(race)))+
  geom_bar(position=position_dodge())

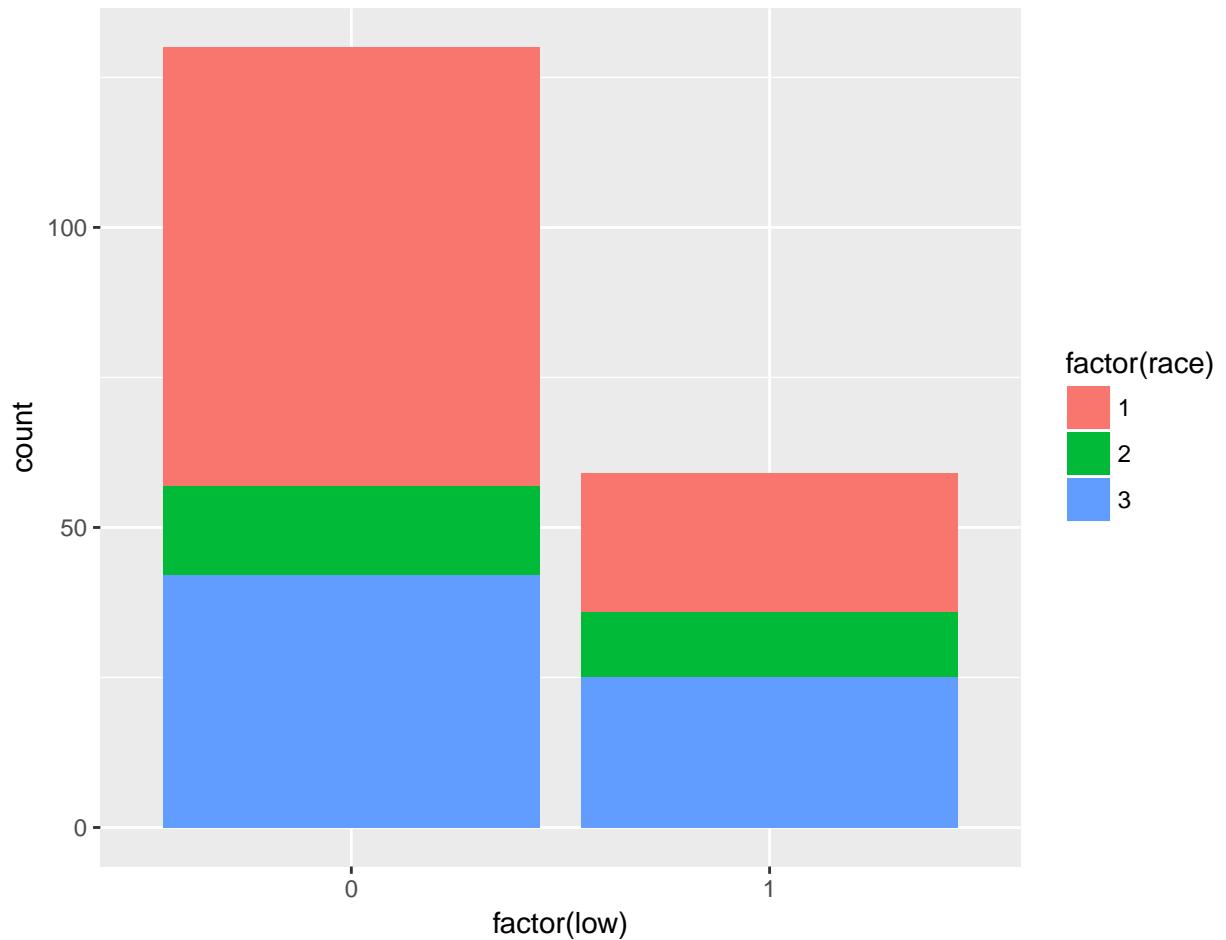
```



NOTA: Consulta la ayuda ??position_dodge.

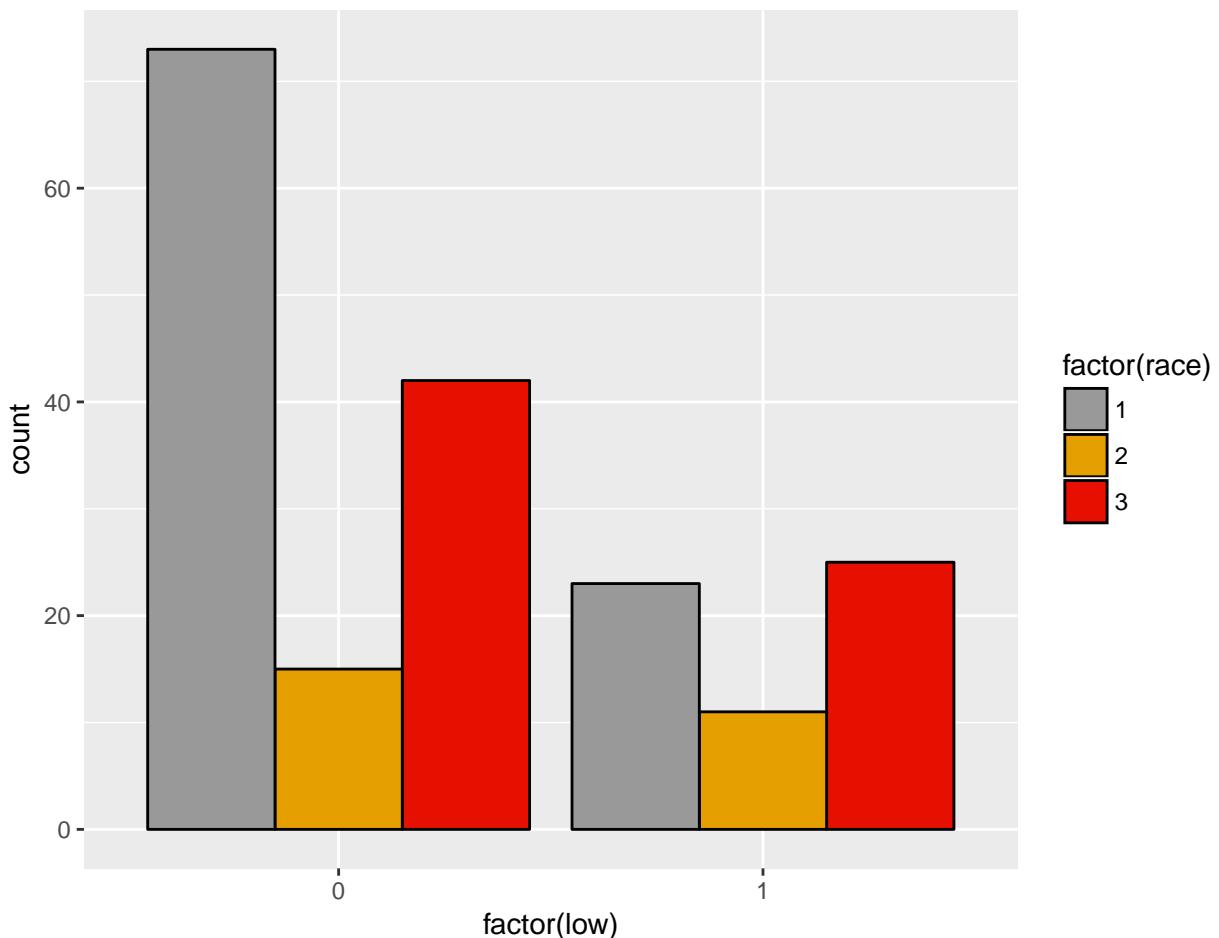
Ahora realizamos el mismo gráfico sin incluir `position_dodge()`:

```
ggplot(birthwt,aes(x=factor(low),fill=factor(race)))+  
  geom_bar()
```



Podemos **cambiar el color de las barras** haciendo:

```
ggplot(birthwt,aes(x=factor(low),fill=factor(race)))+  
  geom_bar(position=position_dodge(),color="black") +  
  scale_fill_manual(values=c("#999999", "#E69F00", "#E70F00"))
```



En general, es más interesante representar los porcentajes. En las tablas podemos hacer, por ejemplo, el procentaje de bajo peso por raza:

```
t <- with(birthwt,
  table(race,low))
tp <- round(prop.table(t,1),1)
tp
```

```
##      low
## race   0   1
##   1 0.8 0.2
##   2 0.6 0.4
##   3 0.6 0.4
```

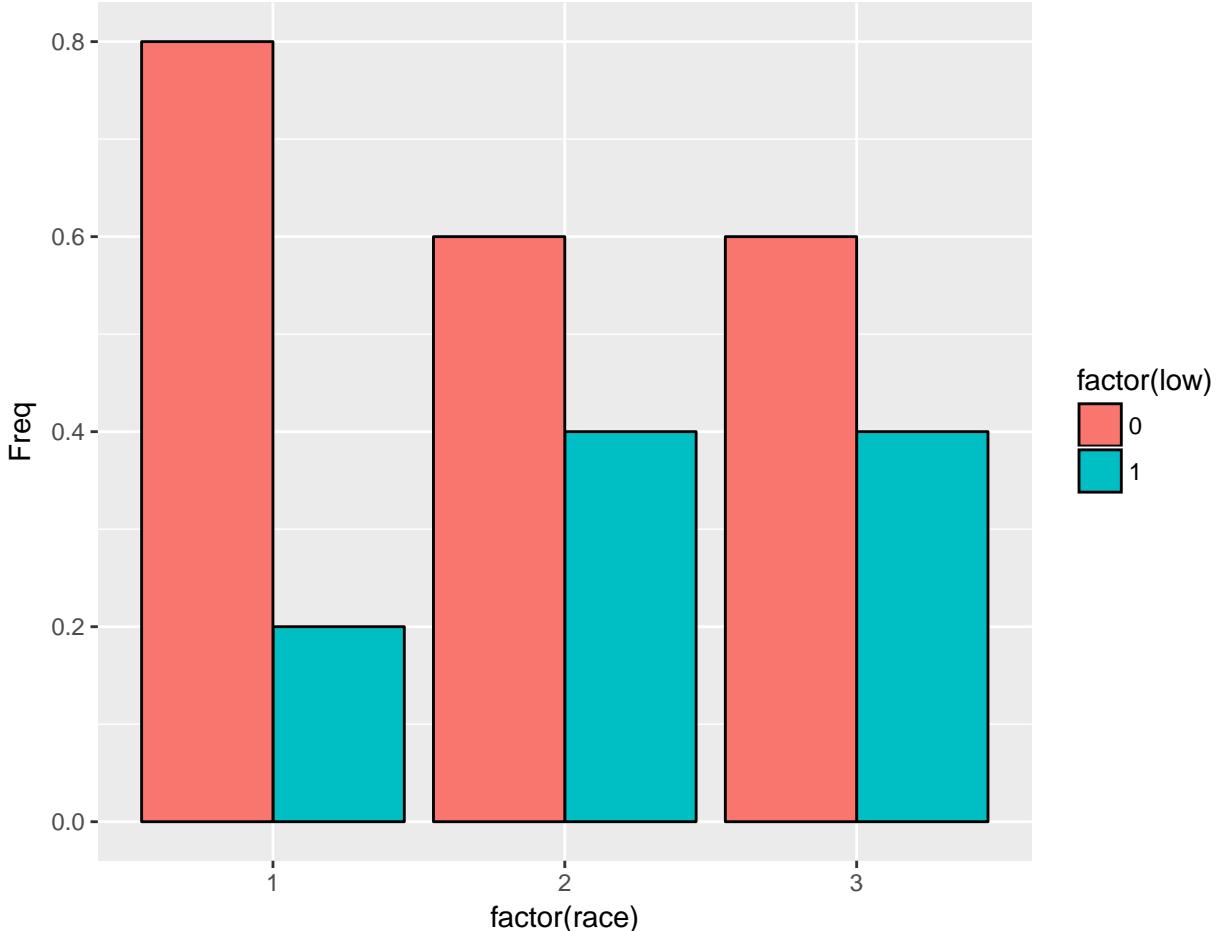
Para poder utilizar los resultados en ggplot2, debemos transformar la tabla a un data.frame:

```
tp <- as.data.frame(round(prop.table(t,1),1))
tp
```

```
##   race low Freq
## 1   1   0  0.8
## 2   2   0  0.6
## 3   3   0  0.6
## 4   1   1  0.2
## 5   2   1  0.4
## 6   3   1  0.4
```

Ahora podemos representar estos porcentajes

```
ggplot(tp,aes(x=factor(race),fill=factor(low), Freq)) +  
  geom_bar(stat="identity",  
           position=position_dodge(),  
           color="black")
```



Gráficas de medias e intervalos de confianza

Necesitamos instalar y activar `plyr` para cargar los siguientes datos que vamos a utilizar (`ToothGrowth`) y `Rmisc` para utilizar la función `resumen`.

El conjunto de datos `ToothGrowth` corresponde a 60 observaciones del largo de los dientes (`len`), realizadas sobre 10 conejillos de Indias a los que se le suplementa uno de los tres niveles de dosis de vitamina C (0.5, 1, y 2 mg; `dose`) con dos tipos de métodos de administración (jugo de naranja o ácido ascórbico; `supp`).

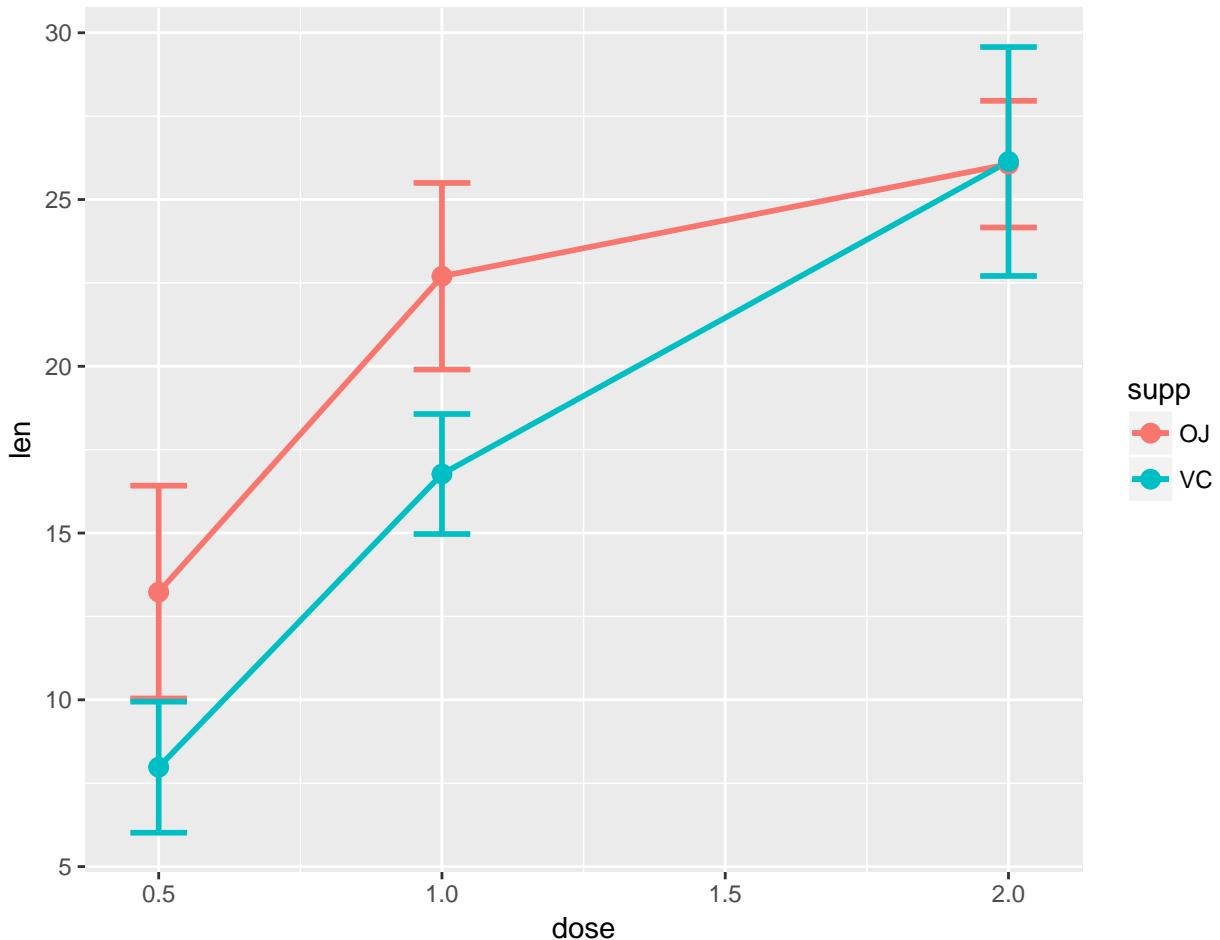
```
library(plyr)  
library(Rmisc)  
  
## Loading required package: lattice  
  
Realizamos un resumen de los datos:  
  
df <- ToothGrowth  
dfc <- summarySE(df, measurevar="len", groupvars=c("supp","dose"))
```

```
dfc
```

```
##   supp dose N  len      sd      se      ci
## 1  OJ  0.5 10 13.23 4.459709 1.410283 3.190283
## 2  OJ  1.0 10 22.70 3.910953 1.2367520 2.797727
## 3  OJ  2.0 10 26.06 2.655058 0.8396031 1.899314
## 4  VC  0.5 10  7.98 2.746634 0.8685620 1.964824
## 5  VC  1.0 10 16.77 2.515309 0.7954104 1.799343
## 6  VC  2.0 10 26.14 4.797731 1.5171757 3.432090
```

Se ha creado un data.frame con la media de longitud por dosis y suplemento. También se ha calculado la correspondiente desviación estandard y el valor de la amplitud del intervalo de confianza. Una vez calculados, podemos representarlos.

```
ggplot(dfc, aes(x=dose, y=len, colour=supp)) +
  geom_errorbar(aes(ymin=len-ci, ymax=len+ci), width=.1, size=1) +
  geom_line(size=1) +
  geom_point(size=3)
```



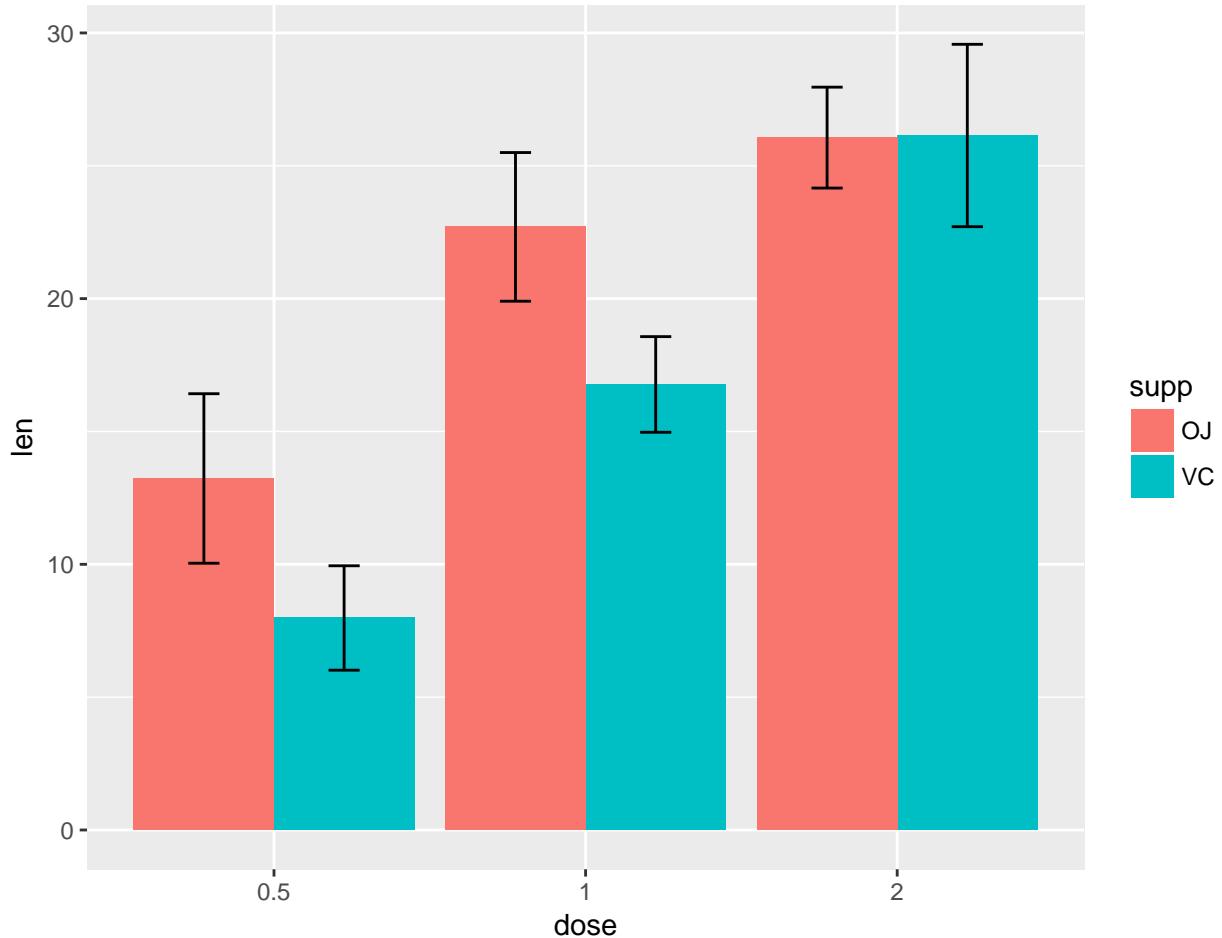
Hemos creado un gráfico en el que se representa la media con un punto y su intervalo de confianza. También se ha añadido un `geom` con líneas.

En algunos casos, se prefiere **representar las medias con barras**. Podemos hacer lo siguiente: primero, definimos la dosis como un factor.

```
dfc2 <- dfc
dfc2$dose <- factor(dfc2$dose)
```

Ahora podemos obtener la gráfica haciendo:

```
ggplot(dfc2, aes(x=dose, y=len, fill=supp)) +
  geom_bar(stat="identity", position=position_dodge()) +
  geom_errorbar(aes(ymin=len-ci, ymax=len+ci),
  width=.2, # Ancho de las barras de error
  position=position_dodge(.9))
```



Personalmente no me gustan demasiado las barras para graficar medias porque asumen un **valor acumulado** y la media en realidad es un **valor puntual**.

Opciones avanzadas

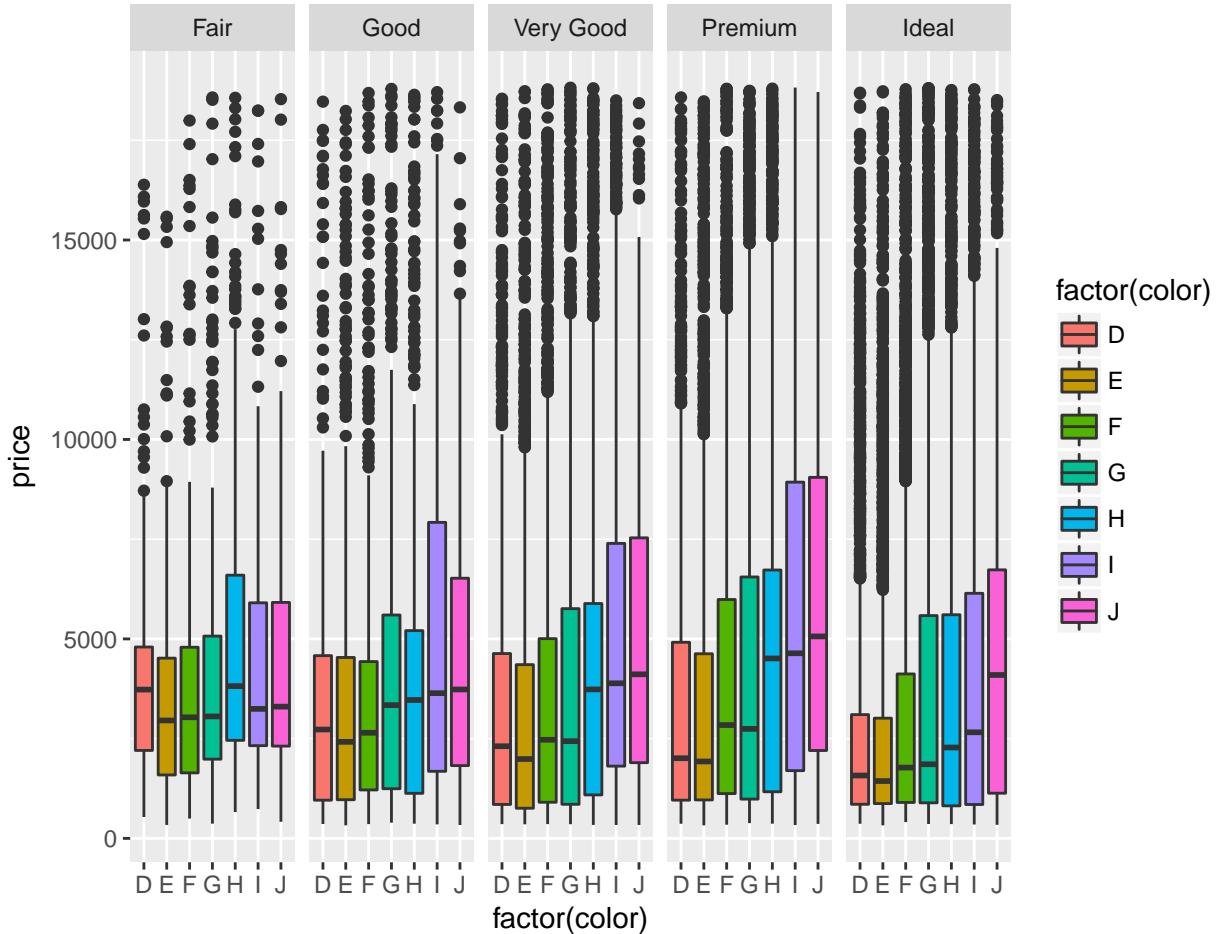
Gráficos por subgrupos

Para resumir los gráficos de cajas y otros gráficos con varios factores resulta muy interesante la opción `facet_grid` que vimos anteriormente.

Para empezar, representaremos un boxplot separando las graficas por `cut`. Para ello, `facet_grid` especifica que variable aparecería en los subgrupos de estas(en este caso ninguna) y qué variable definirá las

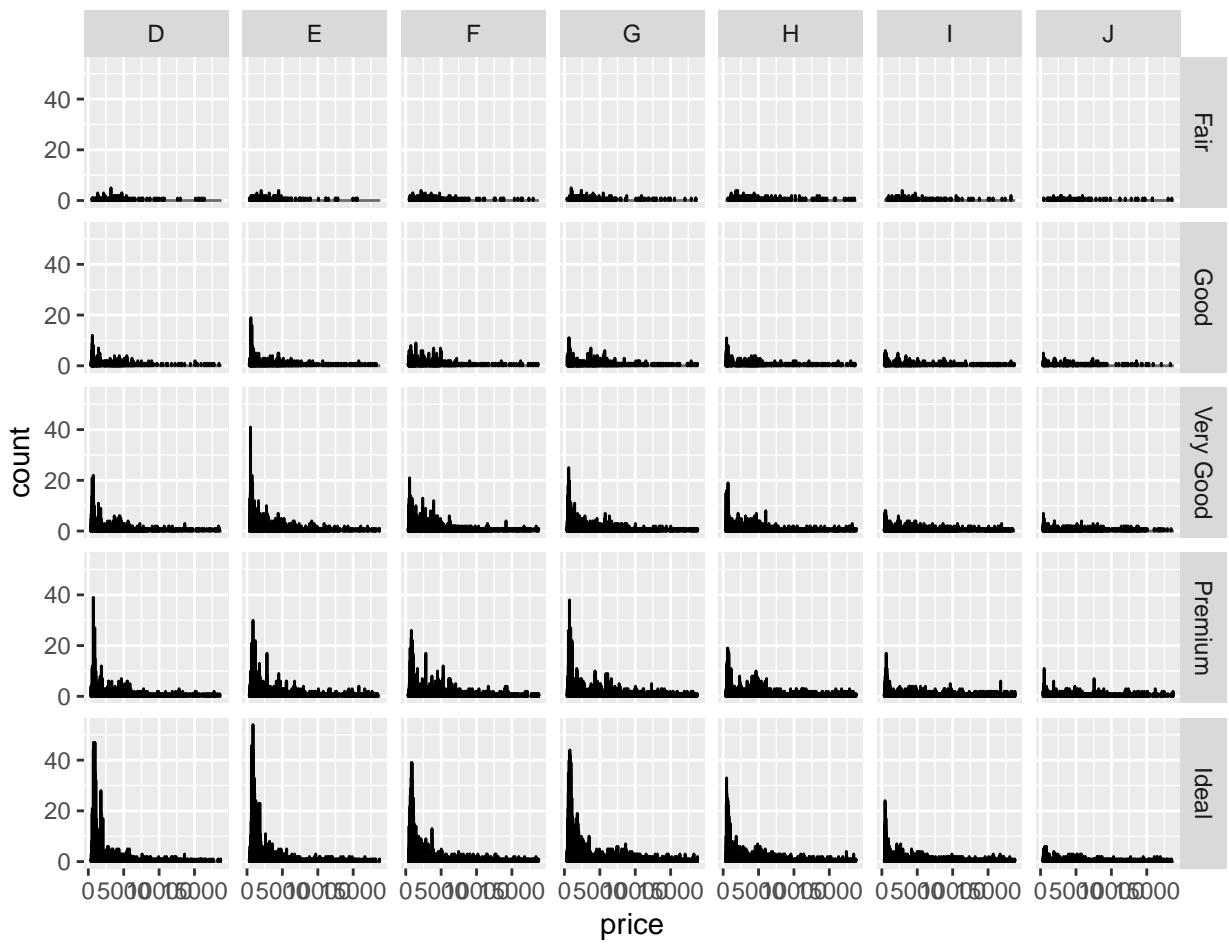
columnas (en este caso el `cut`) en una matriz de gráficos donde se mostraría el grafico requerido para cada combinacion de fila y columna.

```
ggplot(diamonds,aes(factor(color),price)) +
  geom_boxplot(aes(fill=factor(color))) +
  facet_grid(.~cut)
```



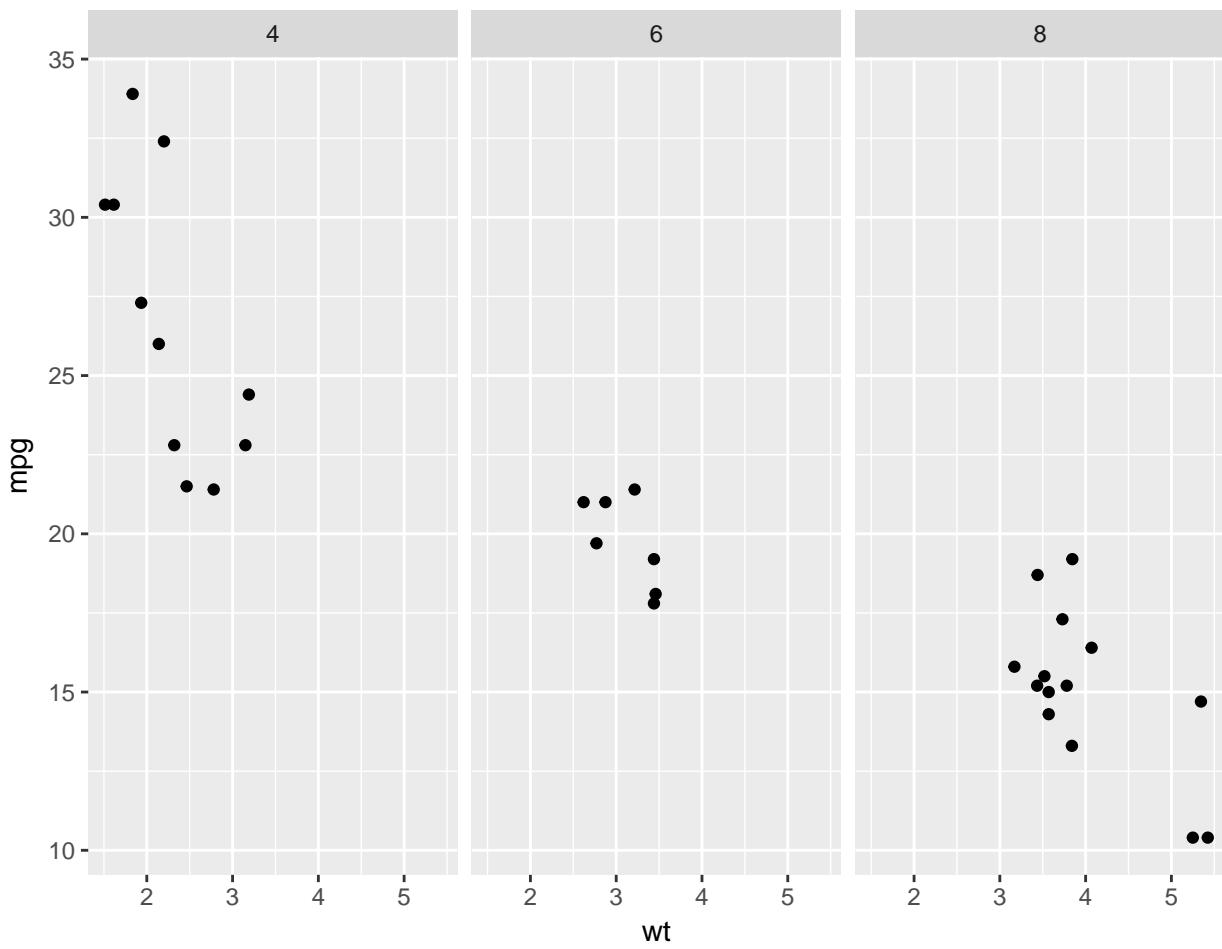
Para apreciar las posibilidades de esta opción, vamos a representar los histogramas de distribución del precio por corte y color.

```
ggplot(diamonds,aes(x=price)) +
  geom_histogram(fill="white",color="black",binwidth=5) +
  facet_grid(cut~color)
```



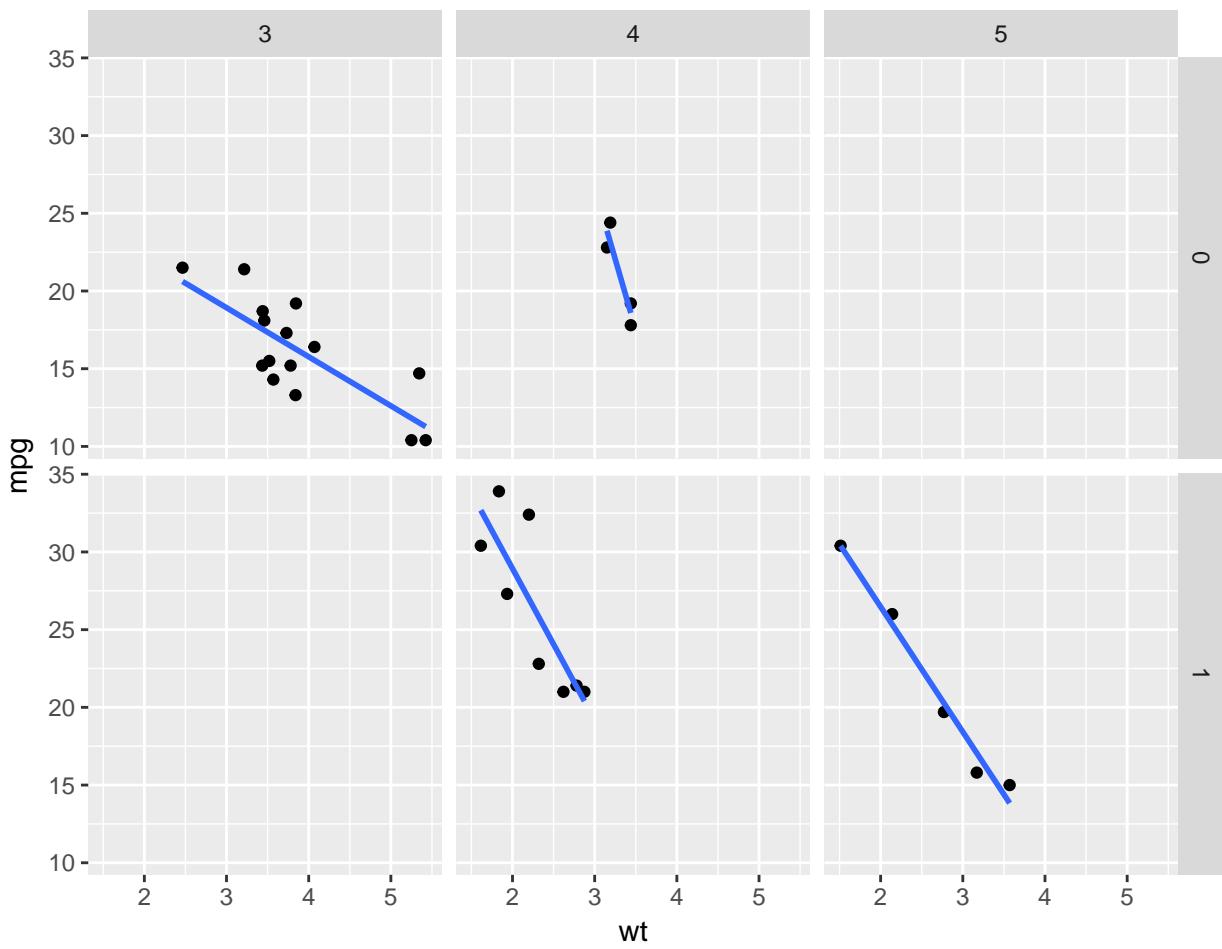
La opción `facet_grid()` puede utilizarse con cualquier tipo de gráficos.

```
data("mtcars")
p <- ggplot(mtcars, aes(x=wt, y=mpg))
p + geom_point() +
  facet_grid(.~cyl)
```



Evidentemente, podemos utilizar más opciones. Por ejemplo, podemos incluir la recta de regresión en las observaciones de cada subgrupo.

```
p + geom_point() +
  facet_grid(am~gear) +
  geom_smooth(se=F,method="lm")
```



Transformación de las escalas de los ejes

Otra opción que nos permite realizar de forma sencilla **ggplot2** es aplicar escalas a los ejes para que las figuras queden mejor representadas ya que en algunos casos es necesario transformar las escalas de los ejes para obtener unos resultados más claros. Veamos como puede hacerse.

Consideremos los datos **mammals** que se encuentran en la librería MASS. Se trata de un conjunto de datos con los pesos cerebrales y corporales promedio para 62 especies de mamíferos terrestres.

```
library(MASS)
data(mammals)
head(mammals)

##           body  brain
## Arctic fox     3.385 44.5
## Owl monkey    0.480 15.5
## Mountain beaver 1.350  8.1
## Cow          465.000 423.0
## Grey wolf     36.330 119.5
## Goat          27.660 115.0

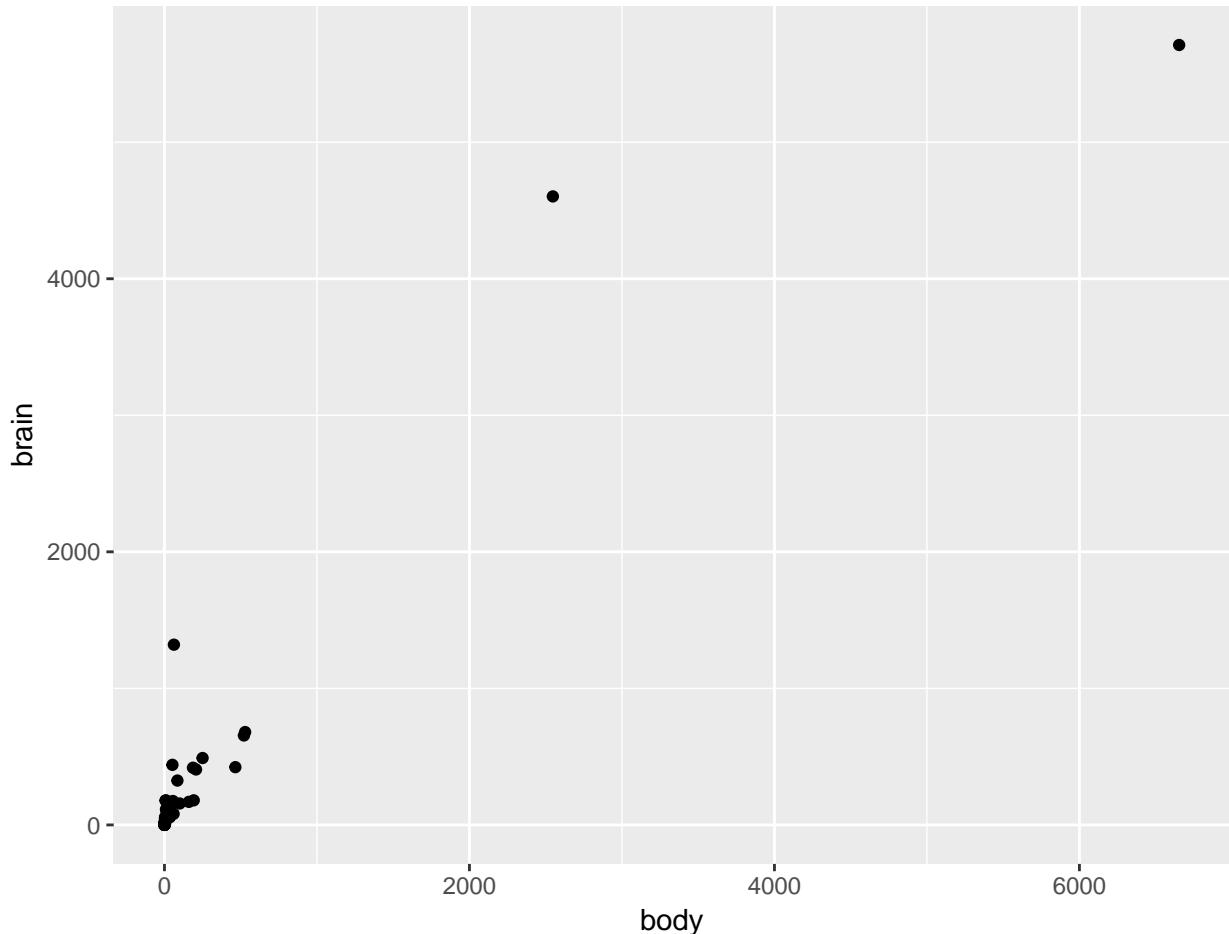
str(mammals)

## 'data.frame':   62 obs. of  2 variables:
##   $ body : num  3.38 0.48 1.35 465 36.33 ...
```

```
## $ brain: num 44.5 15.5 8.1 423 119.5 ...
```

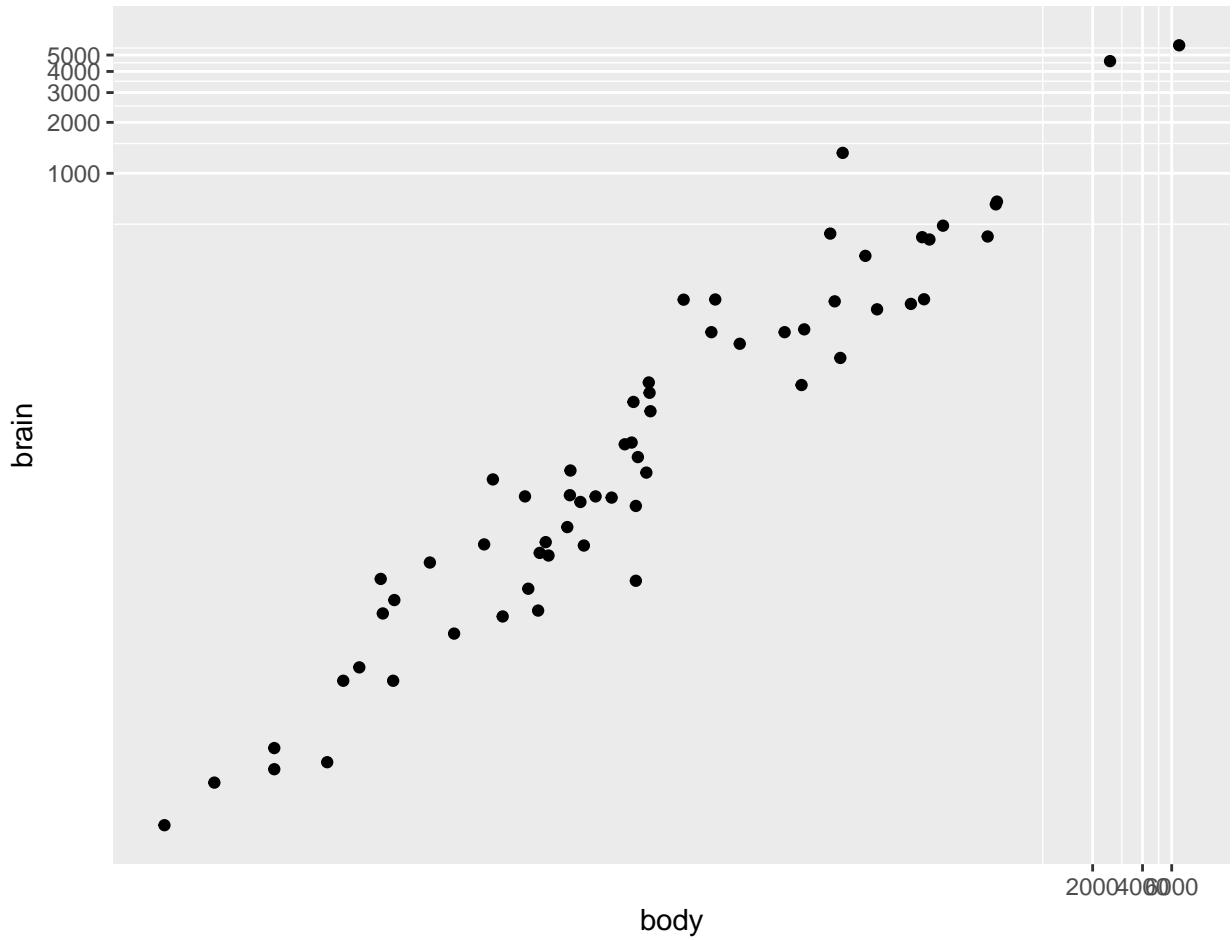
Si representamos el peso del cerebro respecto al peso del cuerpo, obtenemos:

```
mam <- ggplot(mammals,aes(x=body,y=brain))  
mam +  
  geom_point()
```



Dado que existe una diferencia muy grande en el tamaño de los animales, puede ser más interesante representar estos datos en escala logarítmica.

```
mam +  
  geom_point() +  
  coord_trans(x="log",y="log")
```



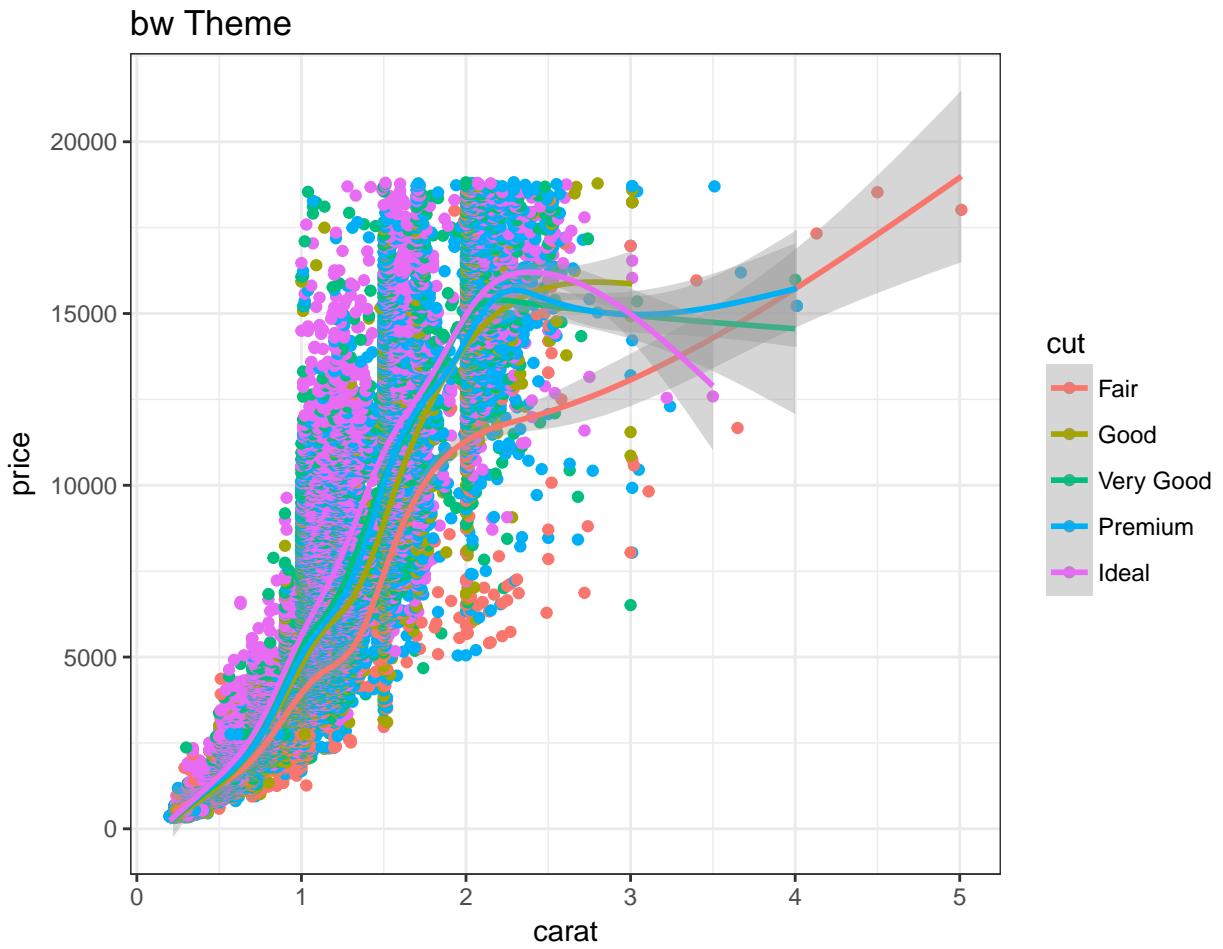
Cambiar temas

A parte del tema básico de `ggplot2`, puedes cambiar la apariencia de tus parcelas usando uno de estos temas incorporados.

- `theme_gray()`
- `theme_bw()`
- `theme_linedraw()`
- `theme_light()`
- `theme_minimal()`
- `theme_classic()`
- `theme_void()`

El paquete `ggthemes` proporciona temas adicionales de `ggplot` que imitan a revistas y software famosos. Vamos a emplear de nuevo los datos `diamonds` para ver un ejemplo de cómo cambiar el tema.

```
ggplot(diamonds, aes(x=carat, y=price, color=cut)) +
  geom_point() +
  geom_smooth() +
  theme_bw() +
  labs(title="bw Theme")
```



Prueba con otros temas en el mismo gráfico para que veas cómo cambia el aspecto.

Guardar el gráfico

Vamos a guardar el gráfico de las medias que hemos realizado antes, para ello sólo tenemos que guardarlo como un objeto de R y después guardar este objeto como una imagen.

```
plot1 <- ggplot(dfc, aes(x=dose, y=len, colour=supp)) +
  geom_errorbar(aes(ymin=len-ci, ymax=len+ci), width=.1, size=1) +
  geom_line(size=1) +
  geom_point(size=3)

# guarda el último plot creado.
ggsave("myggplot.png")

# guarda el plot que especifiquemos
ggsave("myggplot.png", plot=plot1)
```

Referencias

- Chang, Winston. 2012. R Graphics Cookbook. Sebastopol, CA: O'Reilly Media.
- Wickham, Hadley. 2009. Ggplot2: elegant graphics for data analysis. New York: Springer.