

QPLOT

Aitor Vázquez Veloso

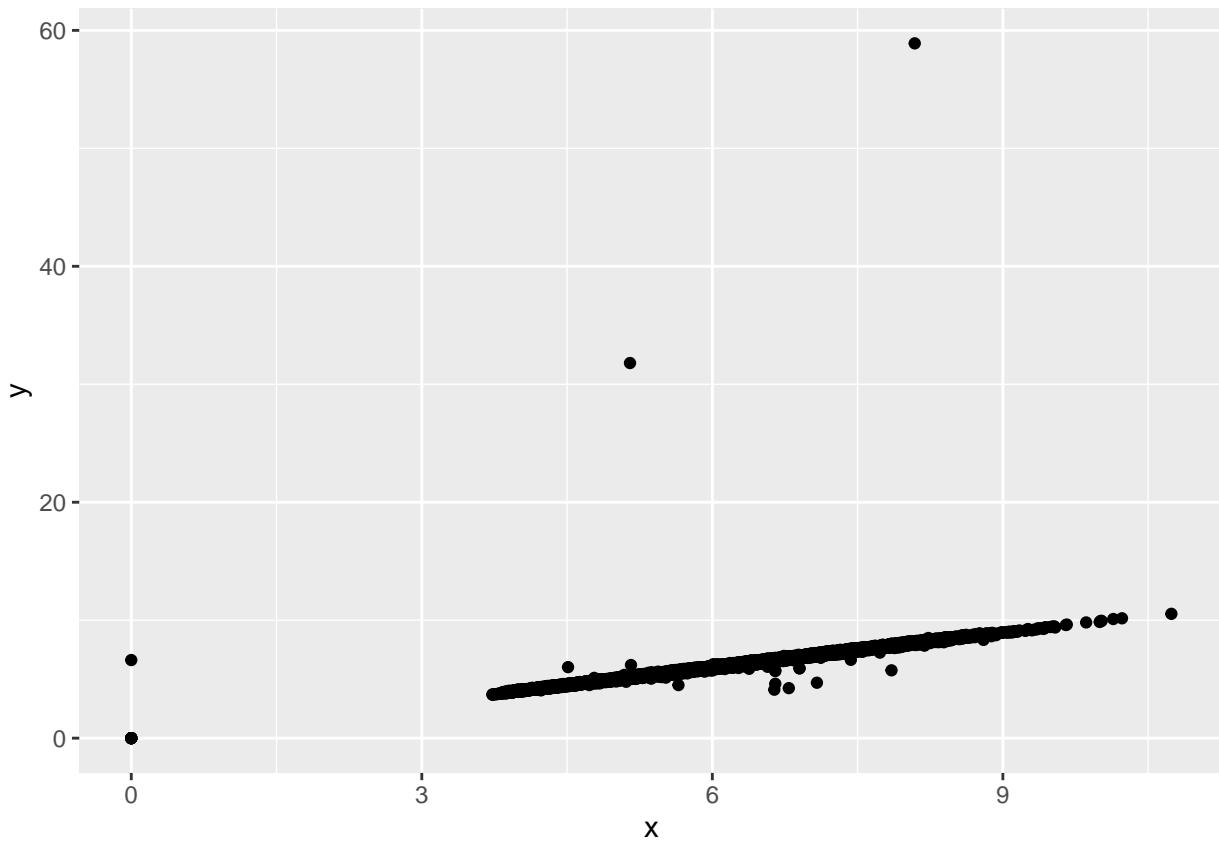
8/9/2021

En este documento se recoge una pequeña explicación de la función **qplot**, elaborada para practicar yo mismo con ella y, además, probar el uso de **RMarkdown**. **qplot** (quick plot), que es una función auxiliar del paquete ggplot2, reduce la complejidad de la graficación. Para utilizarla, cargamos la librería y los datos que nos proporciona de ejemplo.

```
library(ggplot2)
data("diamonds") # es un df que viene por defecto con el paquete ggplot2
```

Estructura principal de la función qplot

```
qplot( #función para dibujar las gráficas
      x, # columna x del df
      y, # columna y del df
      data = diamonds # df a utilizar, al cual pertenecen las columnas x e y
)
```

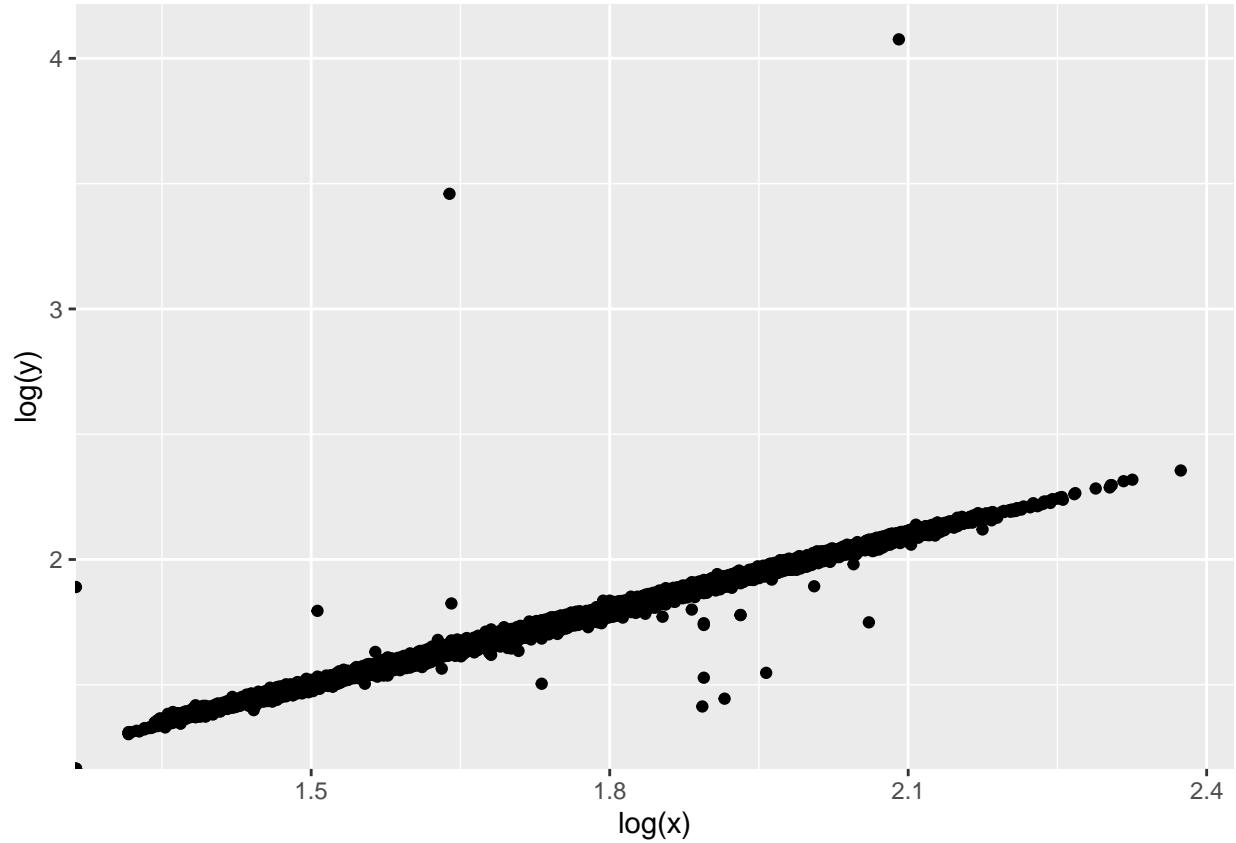


Cálculos directamente en la graficación

qplot permite realizar cálculos sobre las columnas ya existentes en un df en el momento de graficar; de este modo, se pueden combinar columnas entre sí o modificar la escala de los datos.

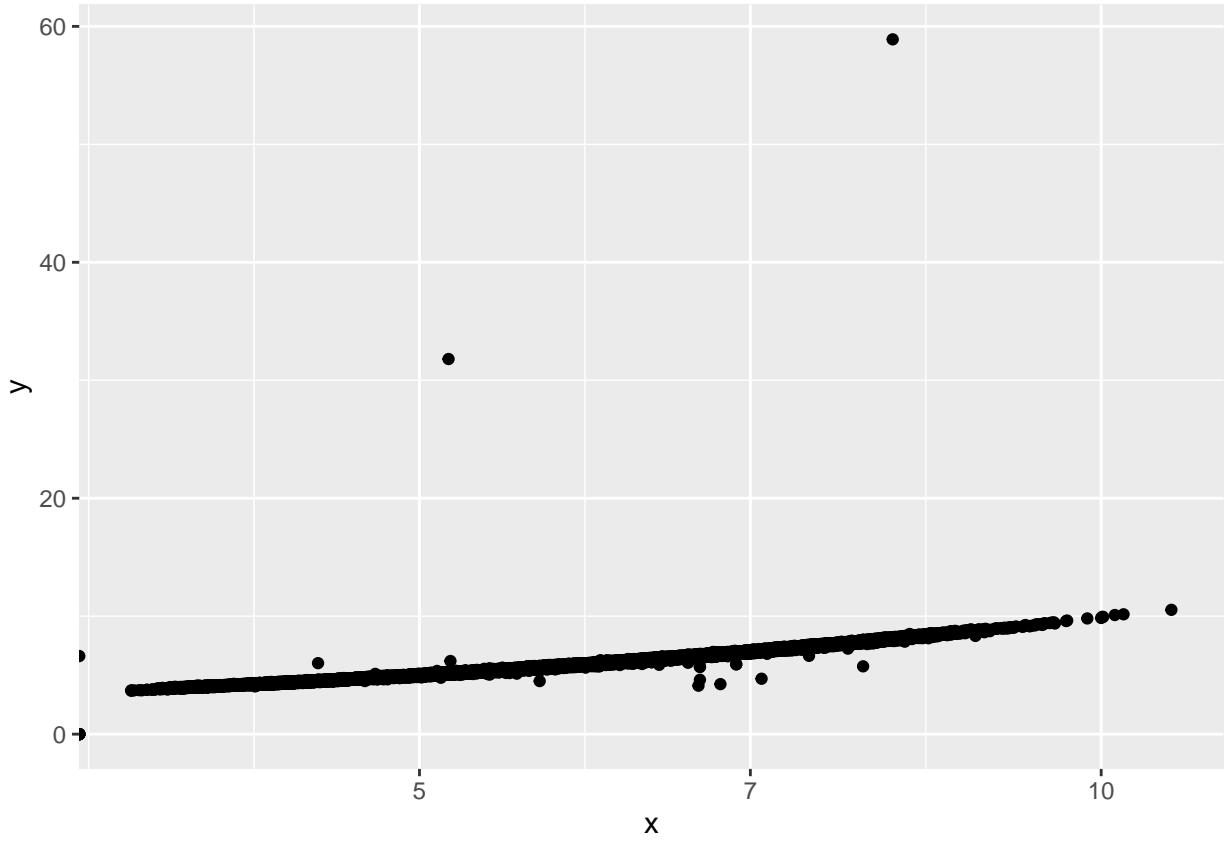
Ejemplo para hacer una *transformación logarítmica*:

```
qplot(  
    log(x),  
    log(y),  
    data = diamonds  
)
```



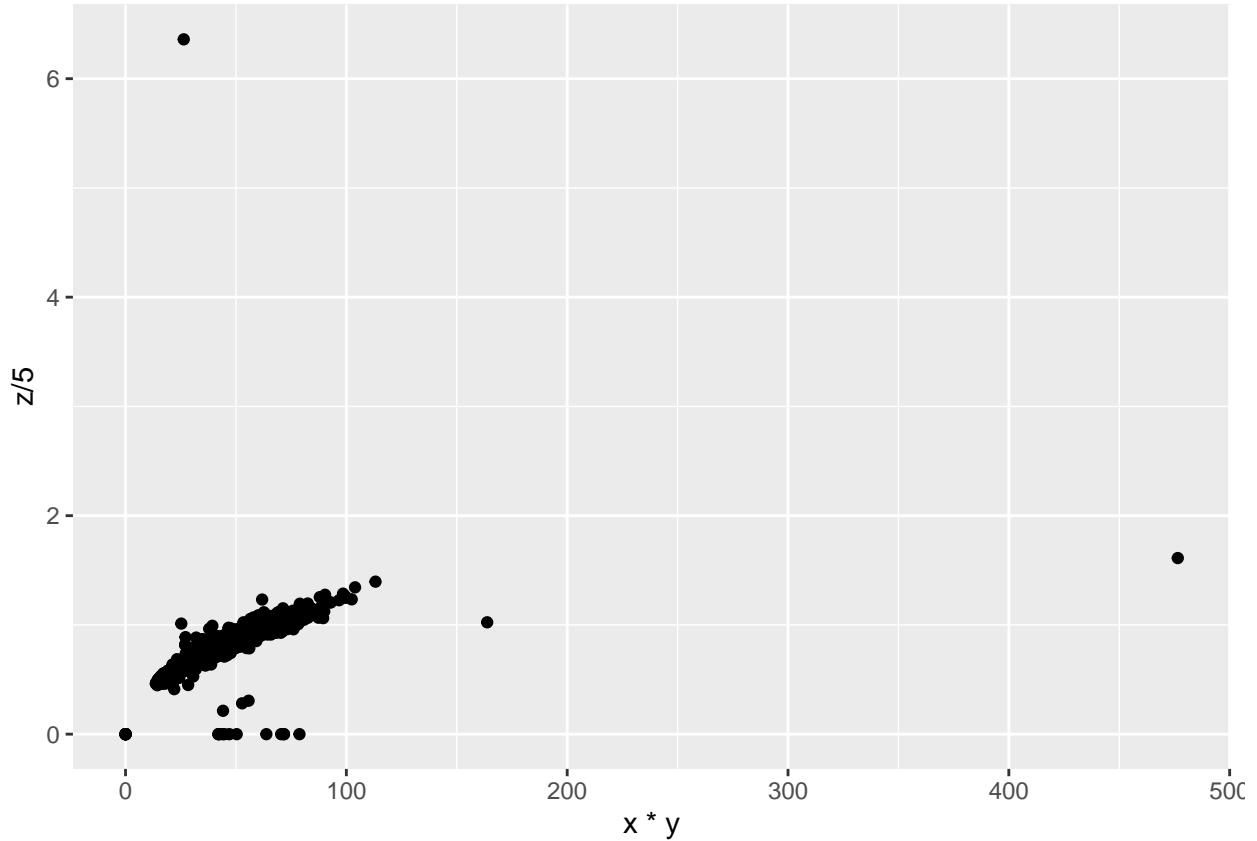
ó

```
qplot(
  x,
  y,
  log = c("x", "y"),
  data = diamonds
)
## Warning: Transformation introduced infinite values in continuous x-axis
```



Otra posibilidad es la de *combinar columnas* entre sí o hacer *cambios de unidades* directamente en el gráfico

```
qplot(  
  x*y,  
  z/5,  
  data = diamonds  
)
```



Atributos estéticos básicos

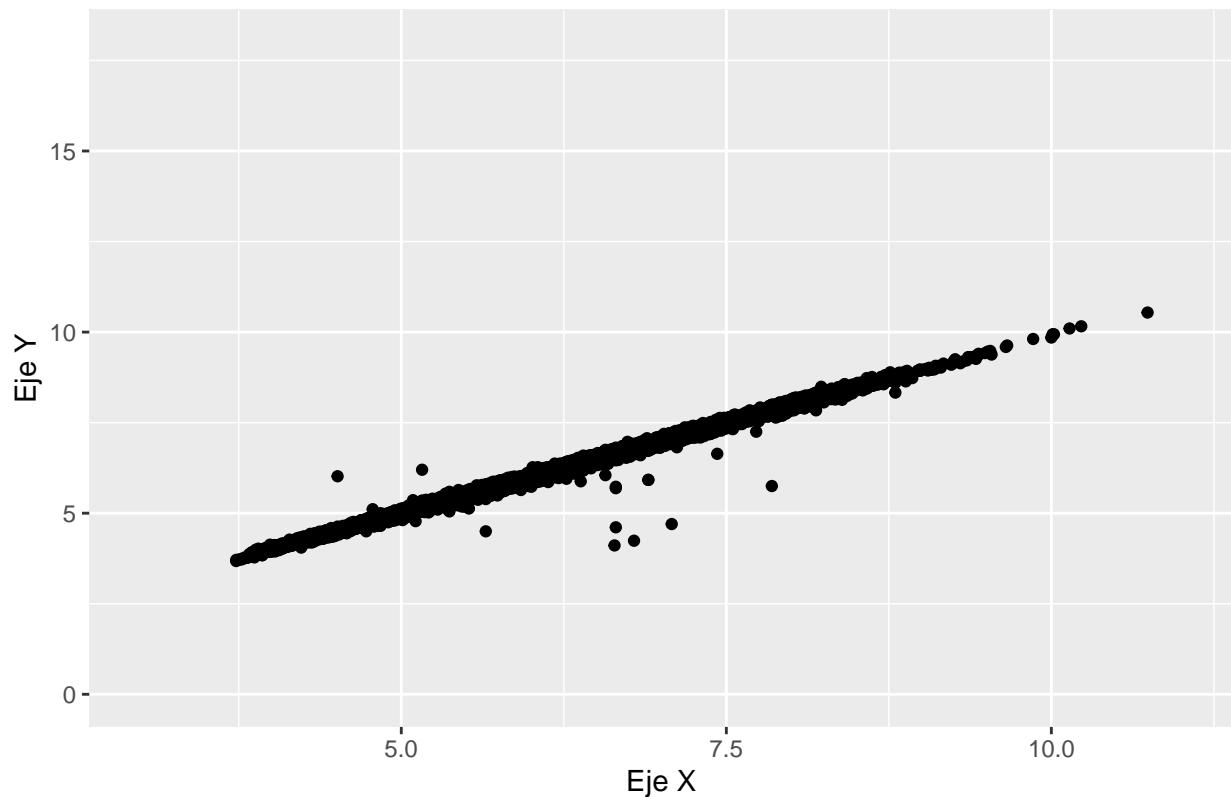
El título se escribe con *main*.

Los títulos de los ejes se escriben con *xlab* e *ylab*.

Los límites del gráfico se escriben con *xlim* e *ylim*.

```
qplot(
  x,
  y,
  data = diamonds,
  main = "Título del gráfico", # título
  xlab = "Eje X", # título del eje x
  ylab = "Eje Y", # título del eje y
  xlim = c(3, 11), # límite inferior y superior, eliminando del gráfico los puntos que sobrepasan
  ylim = c(0, 18) # límite inferior y superior, eliminando del gráfico los puntos que sobrepasan
)
## Warning: Removed 10 rows containing missing values (geom_point).
```

Título del gráfico



Atributos estéticos avanzados

color permite modificar el color de los datos a graficar.

shape permite modificar la forma en la que se representan los datos (círculos, cuadrados, triángulos...).

size permite graficar distintos tamaños de puntos para una misma variable dependiendo de una categoría a la que pertenezcan los datos.

alpha permite reducir la densidad de puntos en la gráfica localizados muy próximos.

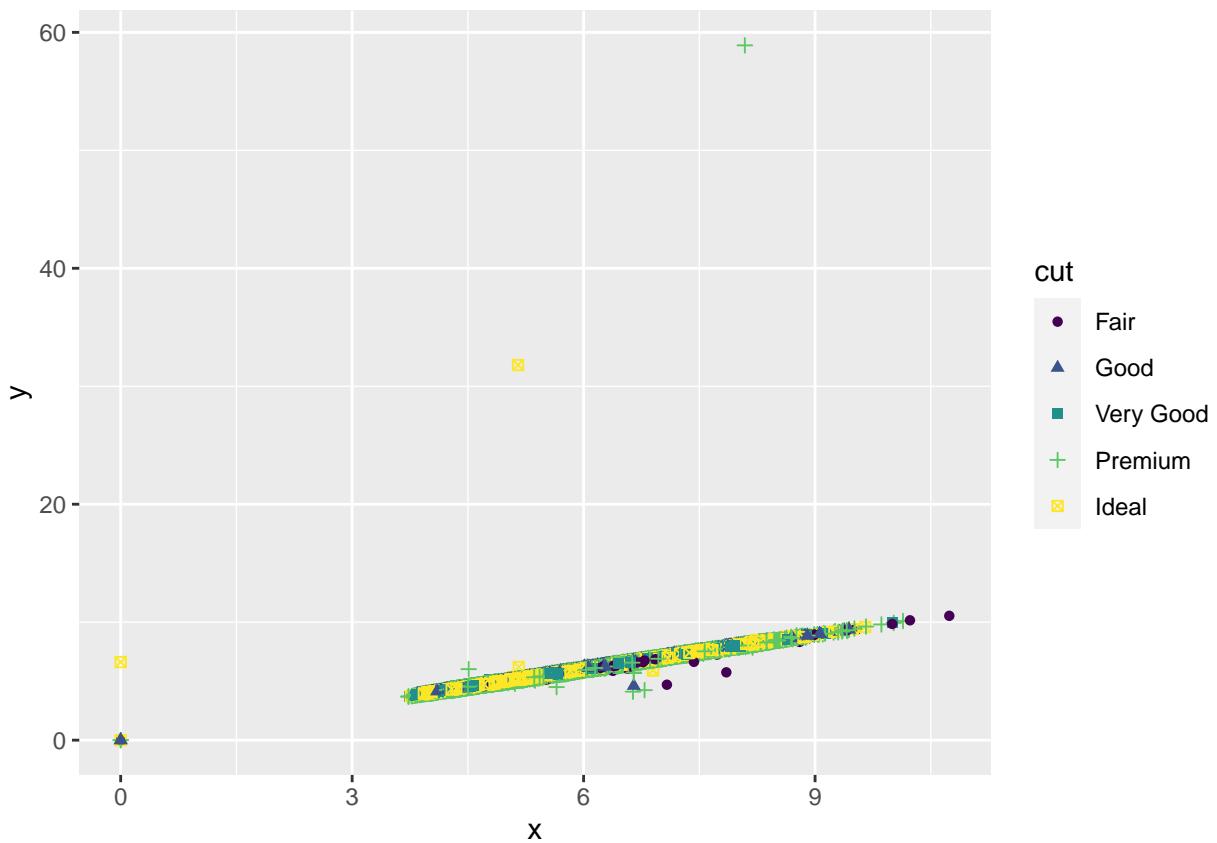
color y *shape* se usan para variables categóricas.

size se usa para variables numéricas.

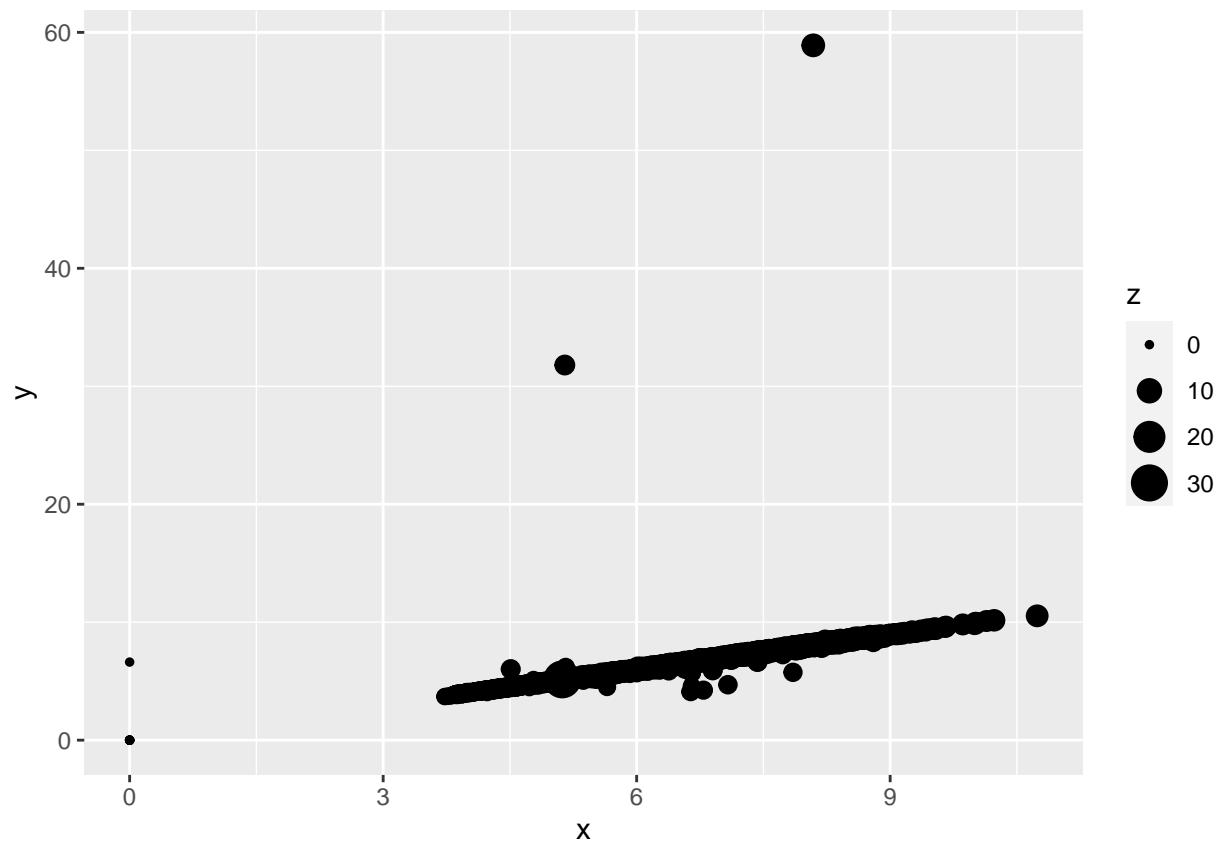
Ejemplos:

```
qplot(
  x,
  y,
  data = diamonds,
  color = cut, # color asignado auto. a cada categoría de la columna "cut"
  shape = cut # forma asignada auto. a cada categoría de la columna "cut"
)

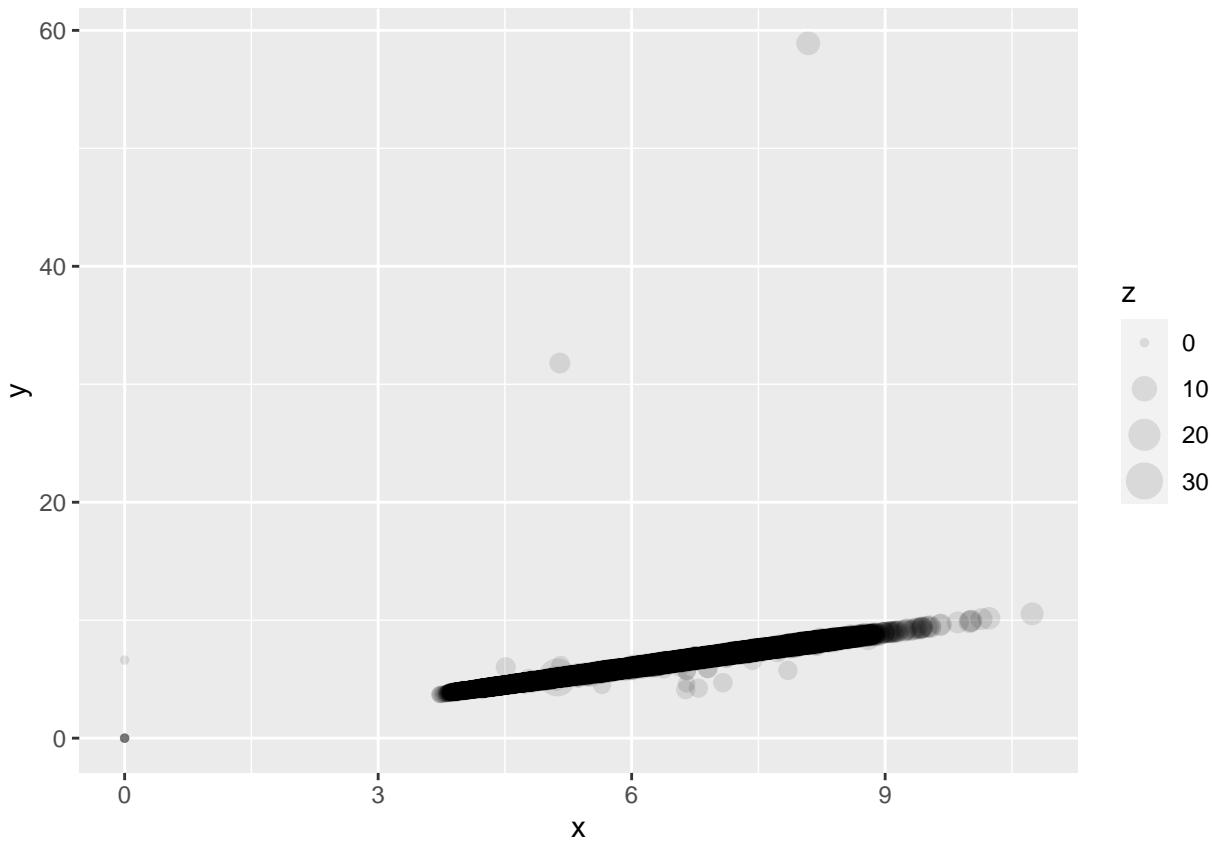
## Warning: Using shapes for an ordinal variable is not advised
```



```
qplot(  
  x,  
  y,  
  data = diamonds,  
  size = z  # tamaño de los puntos determinado por las categorías creadas auto. a partir de la variable cut  
)
```



```
qplot(  
    x,  
    y,  
    data = diamonds,  
    alpha = I(1/10), # para este ejemplo, estamos indicando que en la gráfica se muestre un punto de  
    size = z # tamaño de los puntos determinado por las categorías creadas auto. a partir de la variable z)
```



Sobre los colores

Para hacer gráficos vistosos importan, en cierto modo, los colores utilizados.

`colors()` nos devuelve la lista de colores disponibles.

`palette()` nos devuelve la lista de colores que actualmente están en la paleta que utilizan los gráficos.

Además, los siguientes comandos nos permiten cambiar las paletas de colores de nuestras gráficas a otras prediseñadas

`rainbow()`, escala con los colores del arco iris

`heat.colors()`, escala de calor que utiliza rojo > naranja > amarillo

`topo.colors()`, escala azul > verde > amarillo

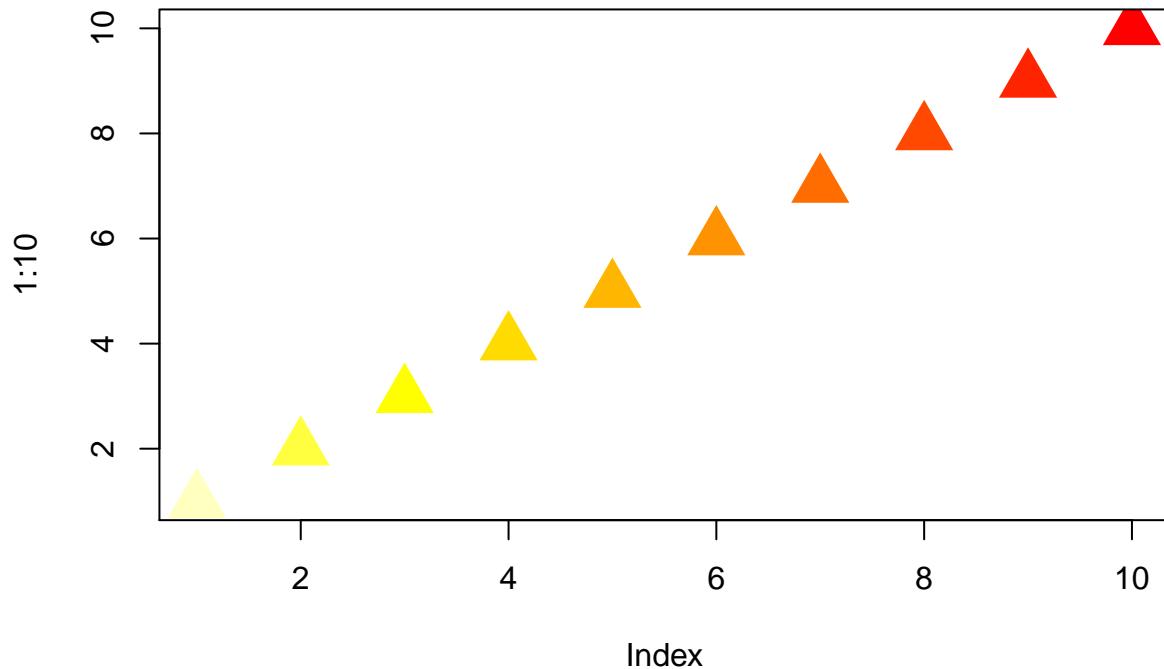
`terrain.colors()`, escala verde > amarillo > marrón

Estas las podemos asignar utilizando los siguientes comandos:

```
paleta_1 <- heat.colors(10, # asignamos una escala de 10 colores en este caso
                         rev = 1, # invertimos el orden de los colores
                         alpha = NULL) # NO asignamos transparencia
palette(paleta_1) # comando para cambiar la paleta de colores por defecto

# ejemplo de visualización
plot(1:10, col=1:10, main="Paleta Cálida", pch=17, cex=3)
```

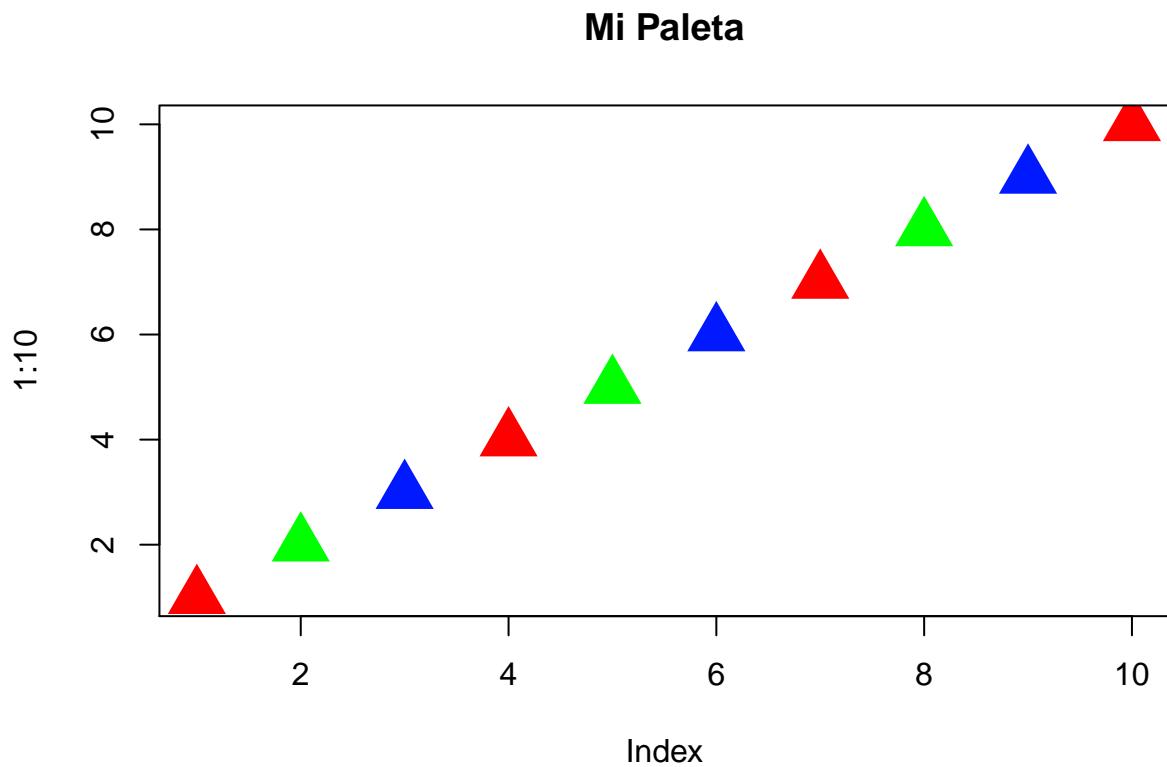
Paleta Cálida



Existen dos formas básicas de asignar una paleta de colores:

- forma 1: asignamos los colores que deseamos a una variable, y esta la ponemos como argumento del gráfico

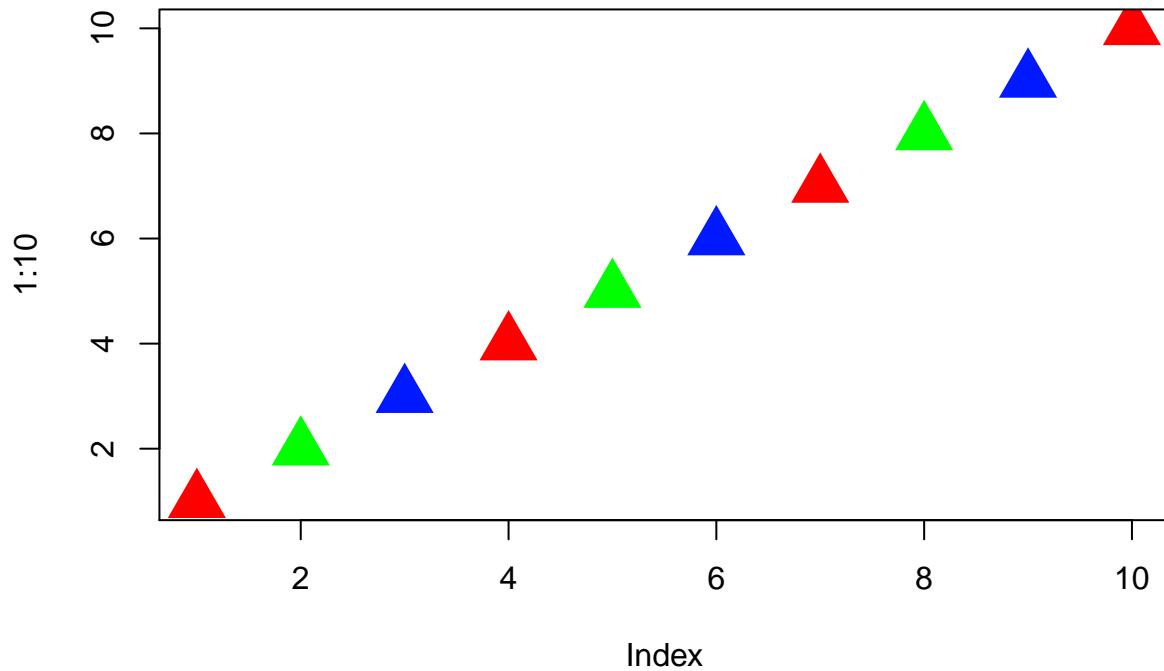
```
mi_paleta <- c("red", "green", "#0019FFFF")
plot(1:10, col=mi_paleta, main="Mi Paleta", pch=17, cex=3)
```



- forma 2: asignamos una variable que contiene los colores deseados a la paleta de colores de R, sin necesidad de ponerla como argumento en el gráfico

```
palette(mi_paleta)
plot(1:10, col=1:10, main="Mi Paleta", pch=17, cex=3)
```

Mi Paleta



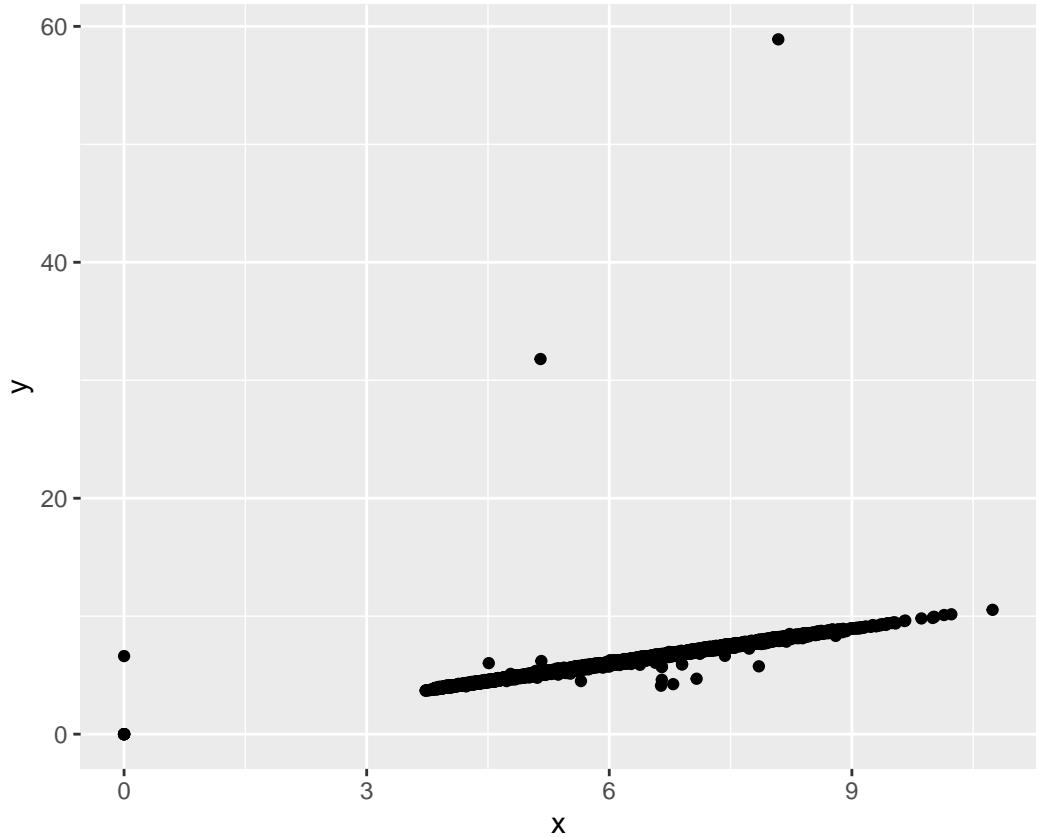
Forma por defecto de la paleta de colores (se restablece al cerrar R)

```
palette("default")
```

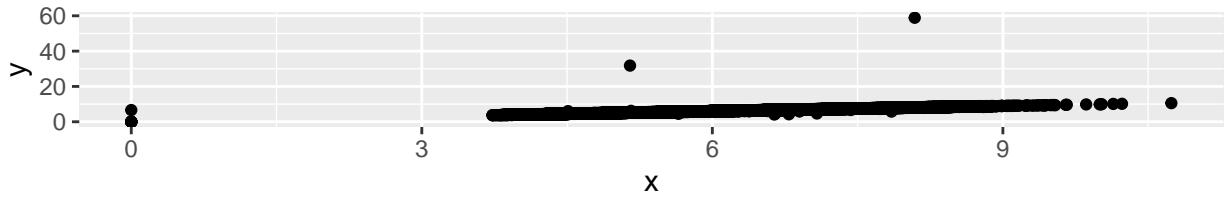
Forma y agrupación de gráficos

`asp` permite alterar el ratio y/x del gráfico, es decir, cambiar las dimensiones x e y del gráfico, estirándolo horizontal o verticalmente.

```
qplot(
  x,
  y,
  data = diamonds,
  asp = 0.8 # modifica el ratio y/x del gráfico, disminuyendo la dimensión horizontal y haciendo que sea más alta que ancha)
```

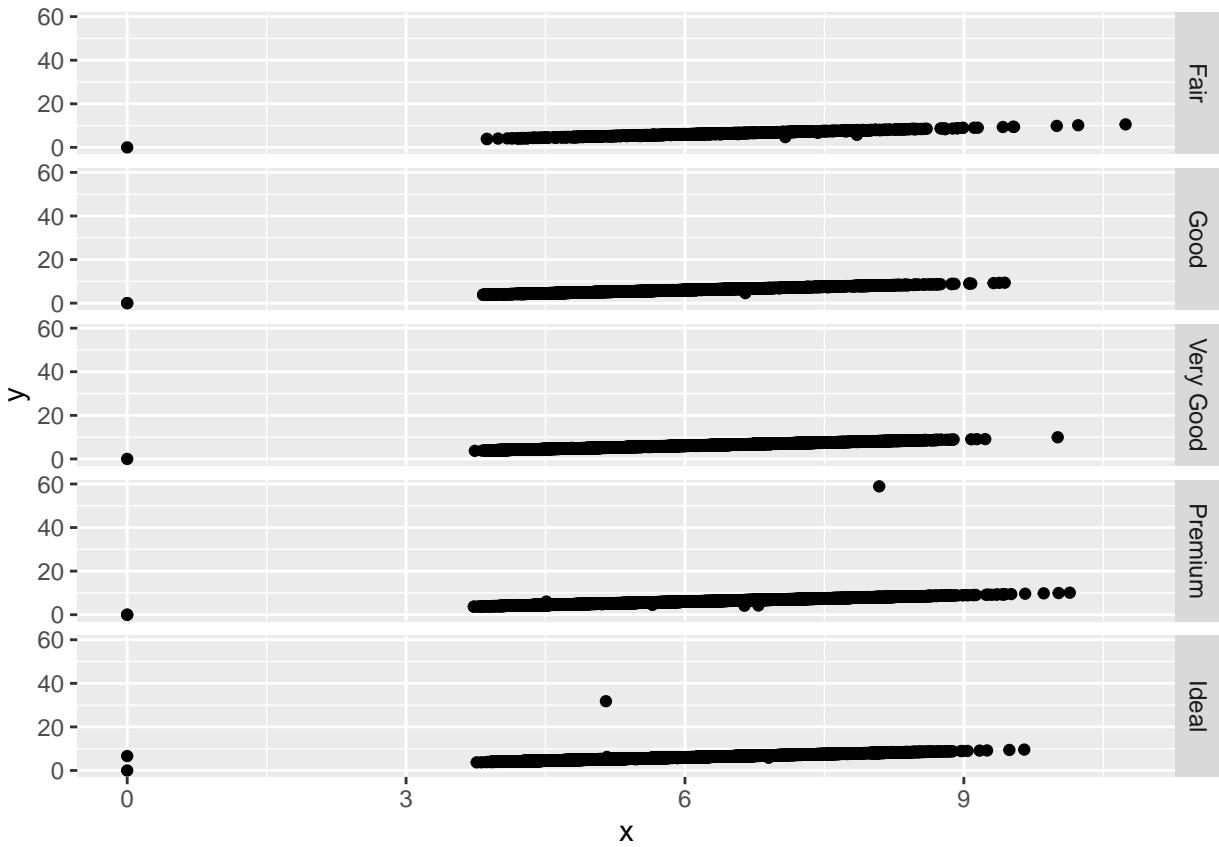


```
qplot(  
  x,  
  y,  
  data = diamonds,  
  asp = 0.1  # modifica el ratio y/x del gráfico, aumentando la dimensión horizontal y haciendo e  
)
```



facets nos permite subdividir el gráfico en tantos gráficos diferentes como campos cualitativos existan en la variable que le indiquemos.

```
qplot(  
    x,  
    y,  
    data = diamonds,  
    facets = cut ~ . # de este modo, indicamos que cree tantos gráficos como categorías diferentes  
)
```



Tipos de gráficos

Para modificar el tipo de gráfico que queremos dibujar utilizaremos el argumento `geom()` de la función `qplot`.

Gráficos que combinan puntos con líneas y/o texto

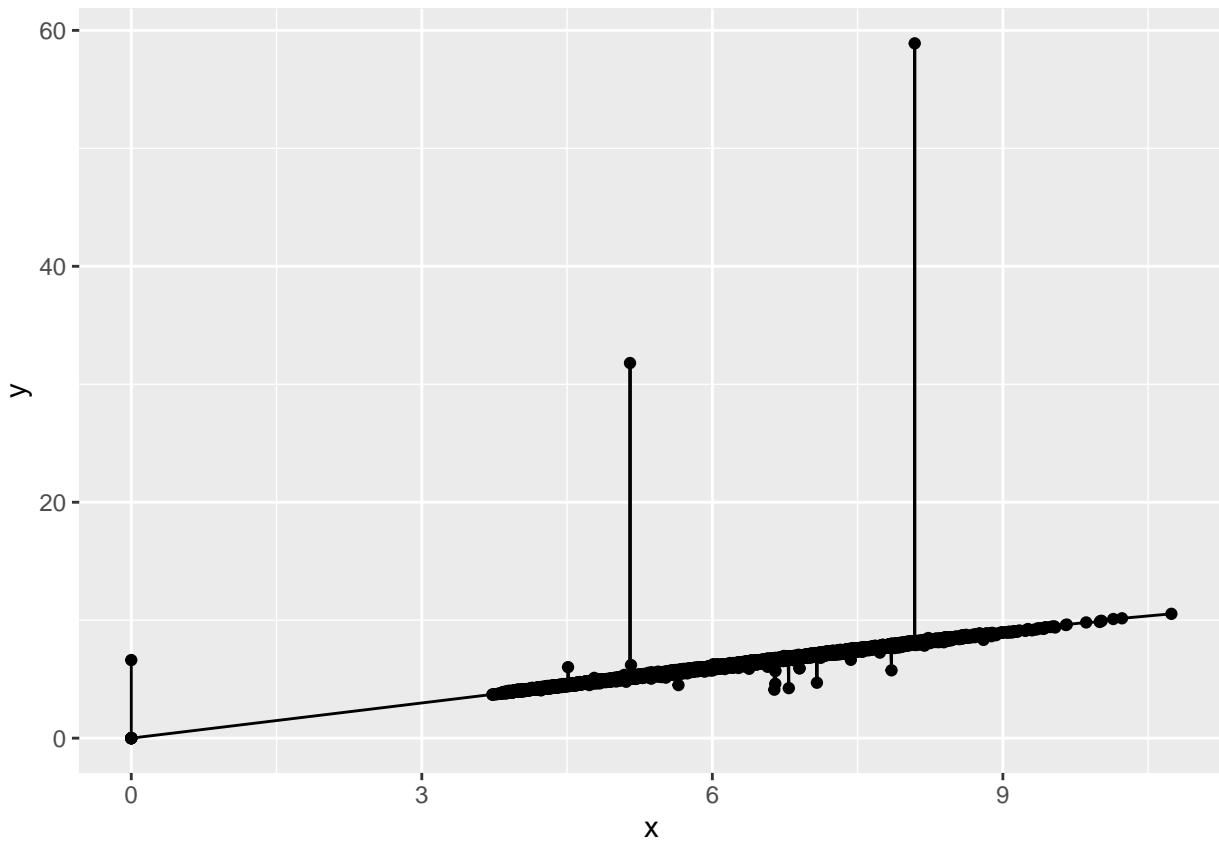
`geom = c("point", "line/smooth/text")` nos dibuja sobre la gráfica los puntos (`point`) y otra opción que le envíemos:

`line` dibuja una línea que une los puntos estamos pintando

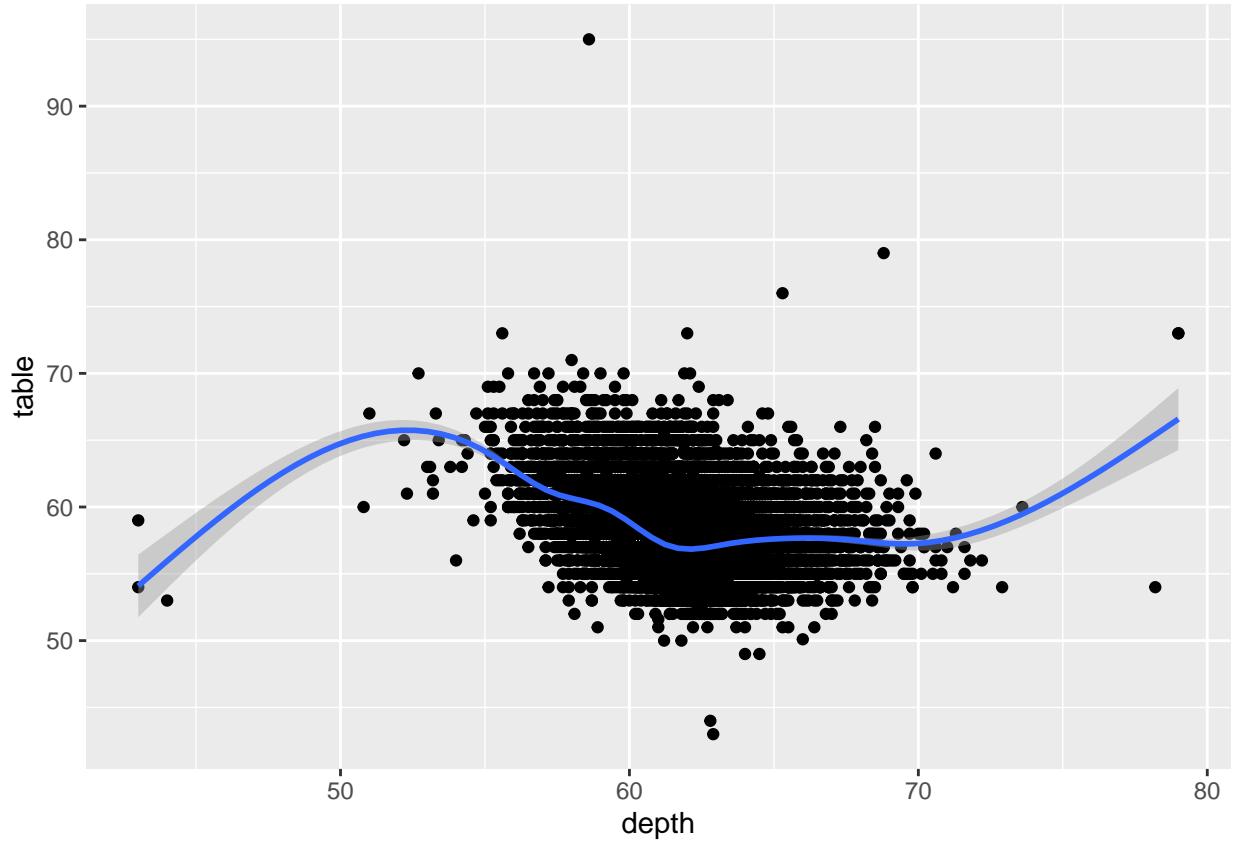
`smooth` dibuja una línea que se ajusta a los datos que estamos pintando y, además, su error estándar

`text` nos dibuja el contenido de la variable que le asociemos con `label`

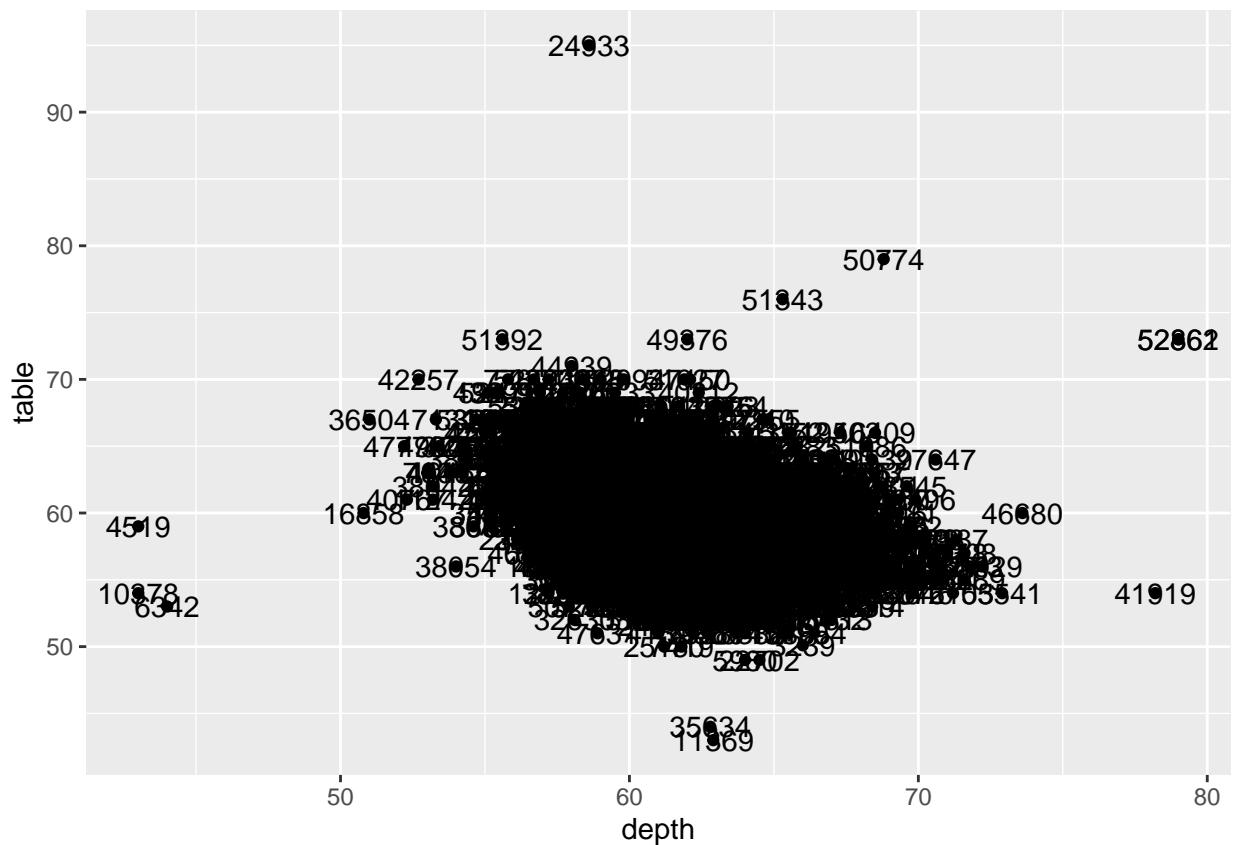
```
qplot(
  x,
  y,
  data = diamonds,
  geom = c("point", "line")  # puntos unidos por una linea
)
```



```
qplot(
  depth,
  table,
  data = diamonds,
  geom = c("point", "smooth") # línea ajustada a los datos con error estándar
)
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



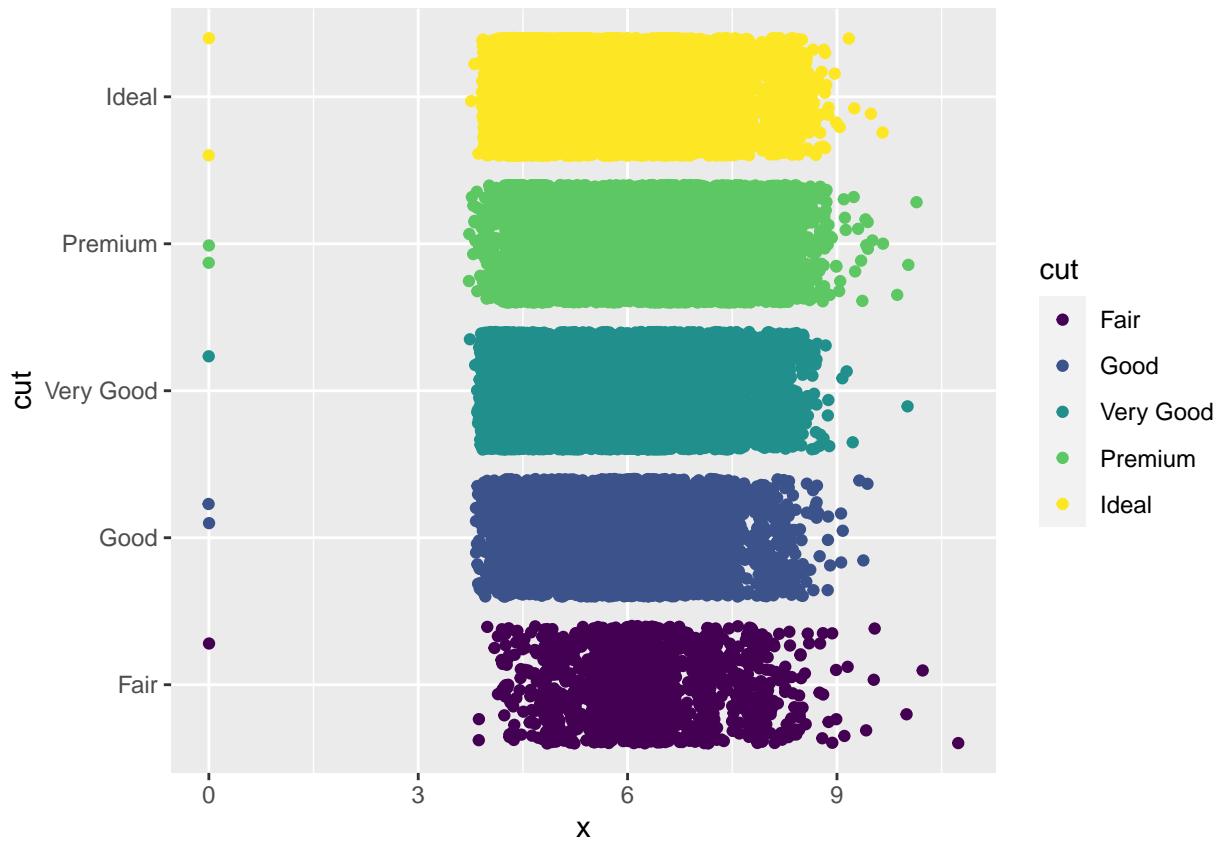
```
qplot(
  depth,
  table,
  data = diamonds,
  label = rownames(diamonds),
  geom = c("point", "text") # puntos con texto
)
```



Gráficos que agrupan puntos

geom = "jitter" nos permite observar los datos por grupos, es decir, generar varios "subgráficos" utilizando una variable categórica para diferenciarlos

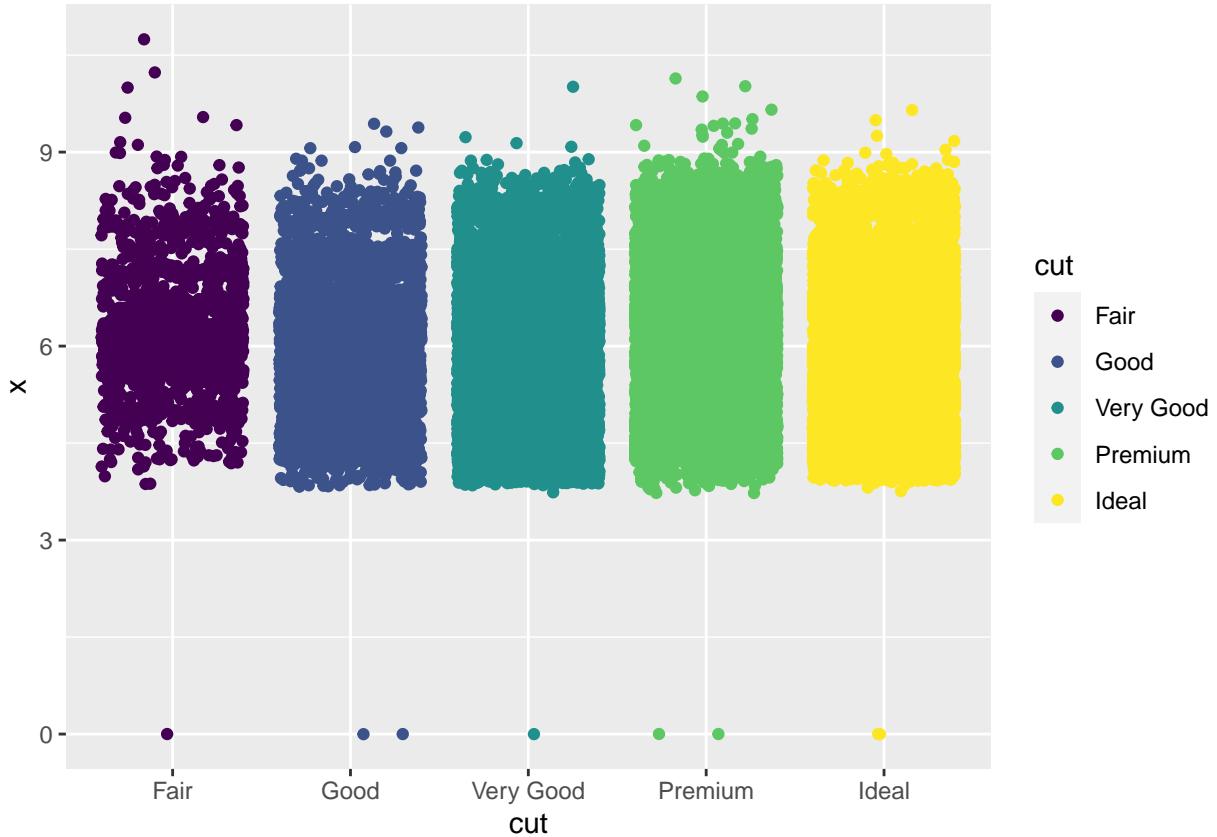
```
qplot(  
    x,  
    cut, # variable por la que categorizamos  
    data = diamonds,  
    geom = "jitter", # nos permite observar por grupos los datos  
    color = cut # variable por la queremos hacer una diferenciación de color  
)
```



```

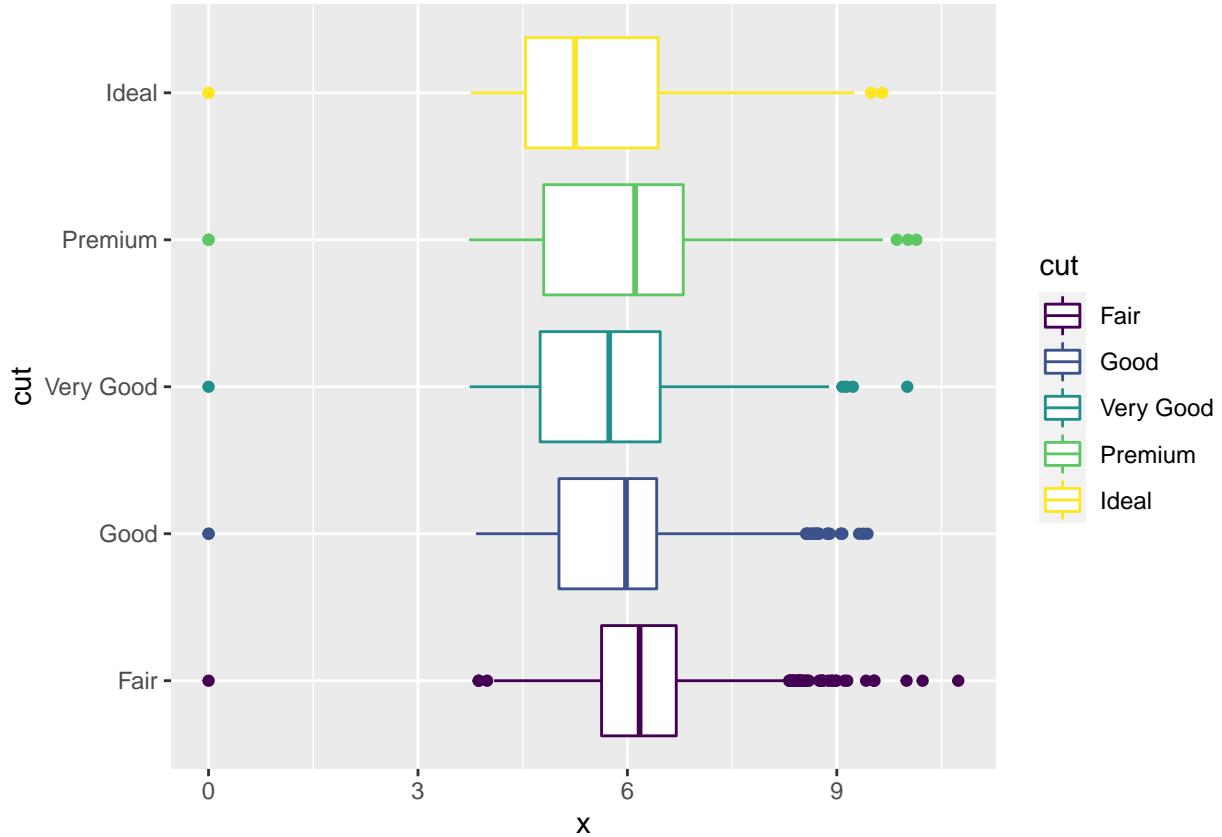
qplot(
  cut, # variable por la que categorizamos
  x,
  data = diamonds,
  geom = "jitter", # nos permite observar por grupos los datos
  color = cut # variable por la queremos hacer una diferenciación de color
)

```



Por otro lado, `geom = "boxplot"` permite realizar gráficos de cajas, configurando los demás parámetros igual que en el caso anterior.

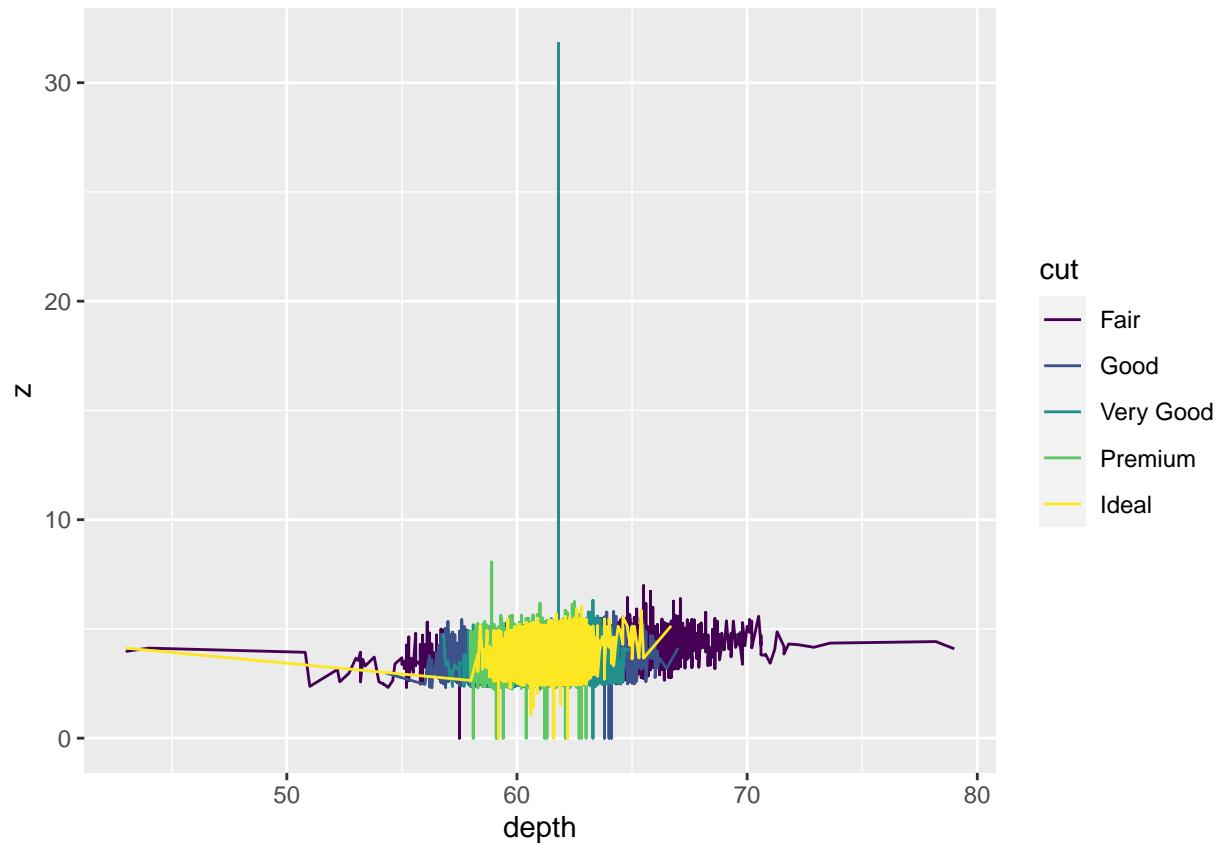
```
qplot(
  x,
  cut, # variable por la que categorizamos
  data = diamonds,
  geom = "boxplot", # nos permite observar los datos agrupados en cajas
  color = cut # variable por la queremos hacer una diferenciación de color
)
```



Gráficos con líneas

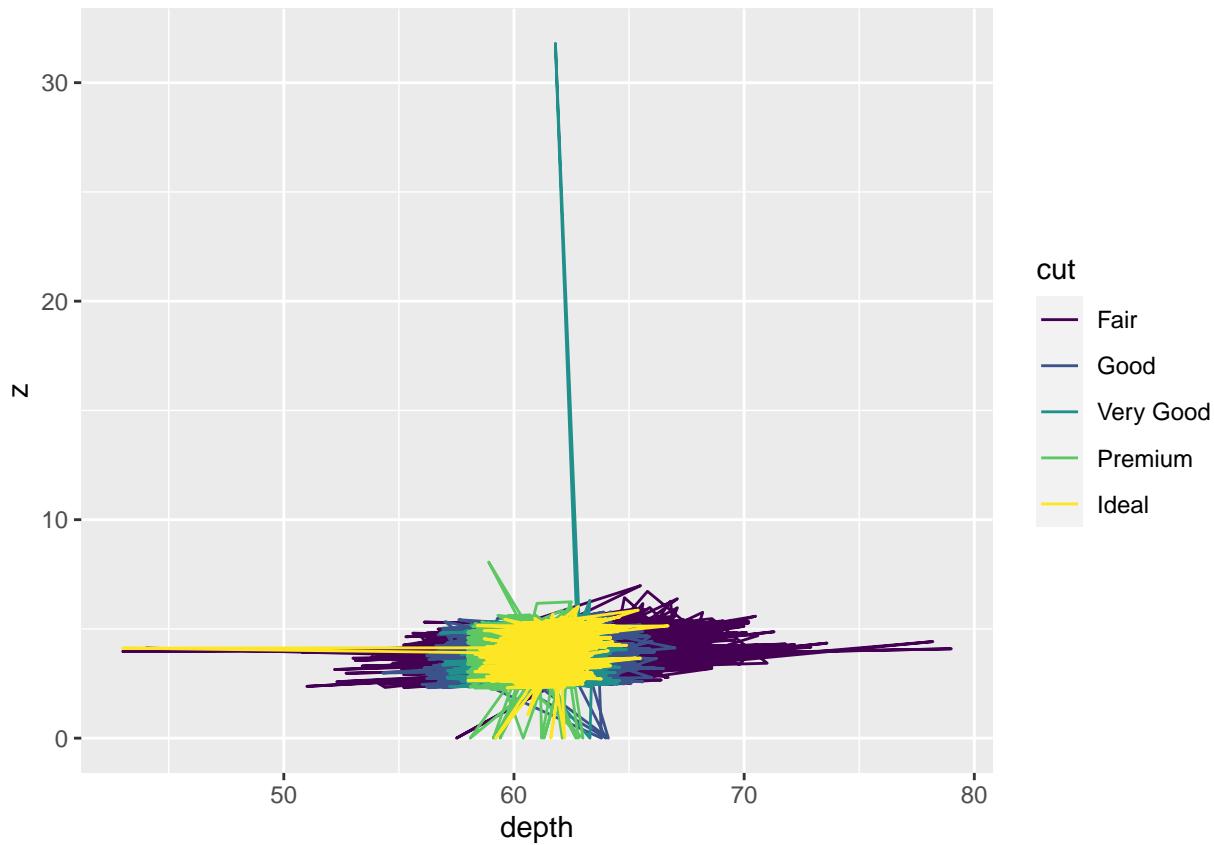
`geom = "line"` nos dibuja líneas entre dos puntos, de manera que nos muestra los puntos unidos por una línea continua, la cual se genera de izquierda a derecha. Se usa para dibujar líneas temporales, principalmente.

```
qplot(
  depth,
  z,
  data = diamonds,
  geom = "line", # nos permite dibujar líneas entre puntos
  color = cut # variable por la queremos hacer una diferenciación de color
)
```



geom = "path" hace lo mismo que *line*, pero sin ir precisamente de izquierda a derecha (¿vecino más cercano?).

```
qplot(
  depth,
  z,
  data = diamonds,
  geom = "path", # nos permite dibujar líneas entre puntos
  color = cut # variable por la queremos hacer una diferenciación de color
)
```

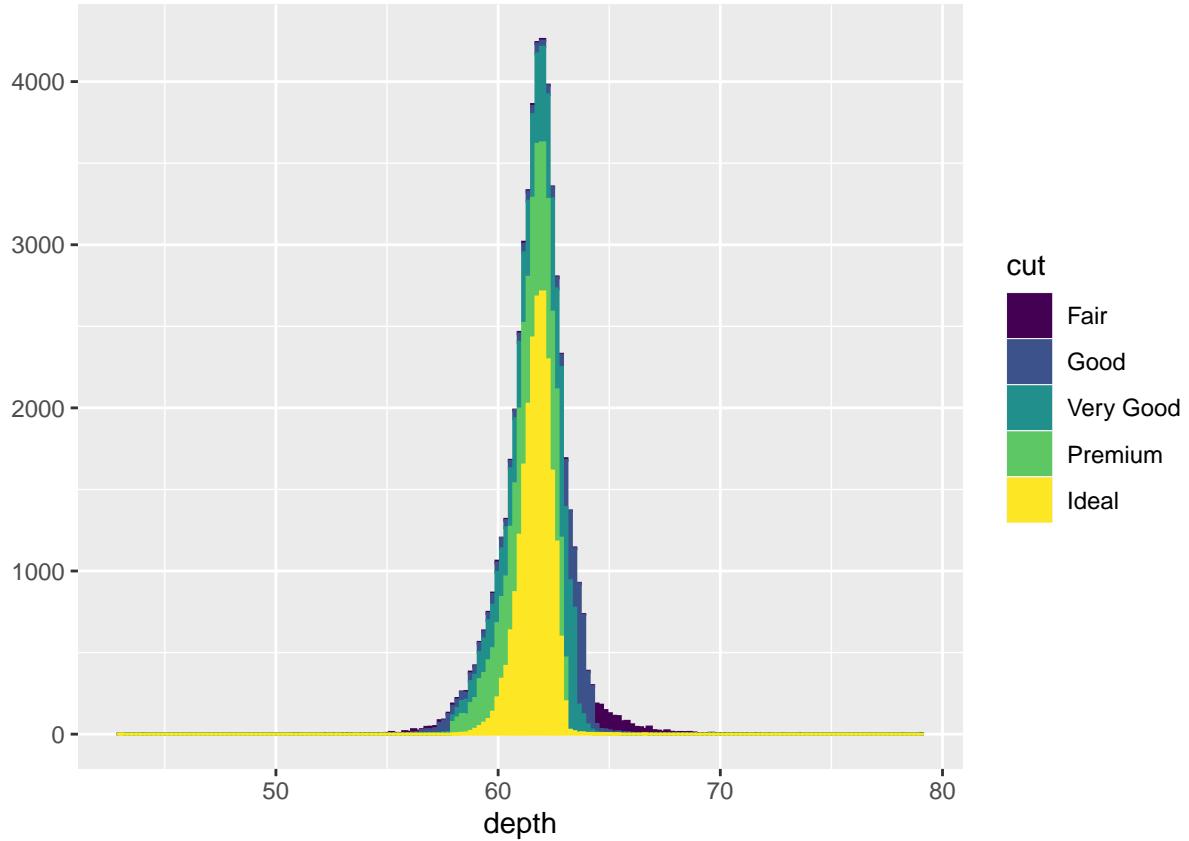


Gráficas para una única variable

`geom = "histogram"` permite realizar un histograma, de manera que solo se grafica una única variable.

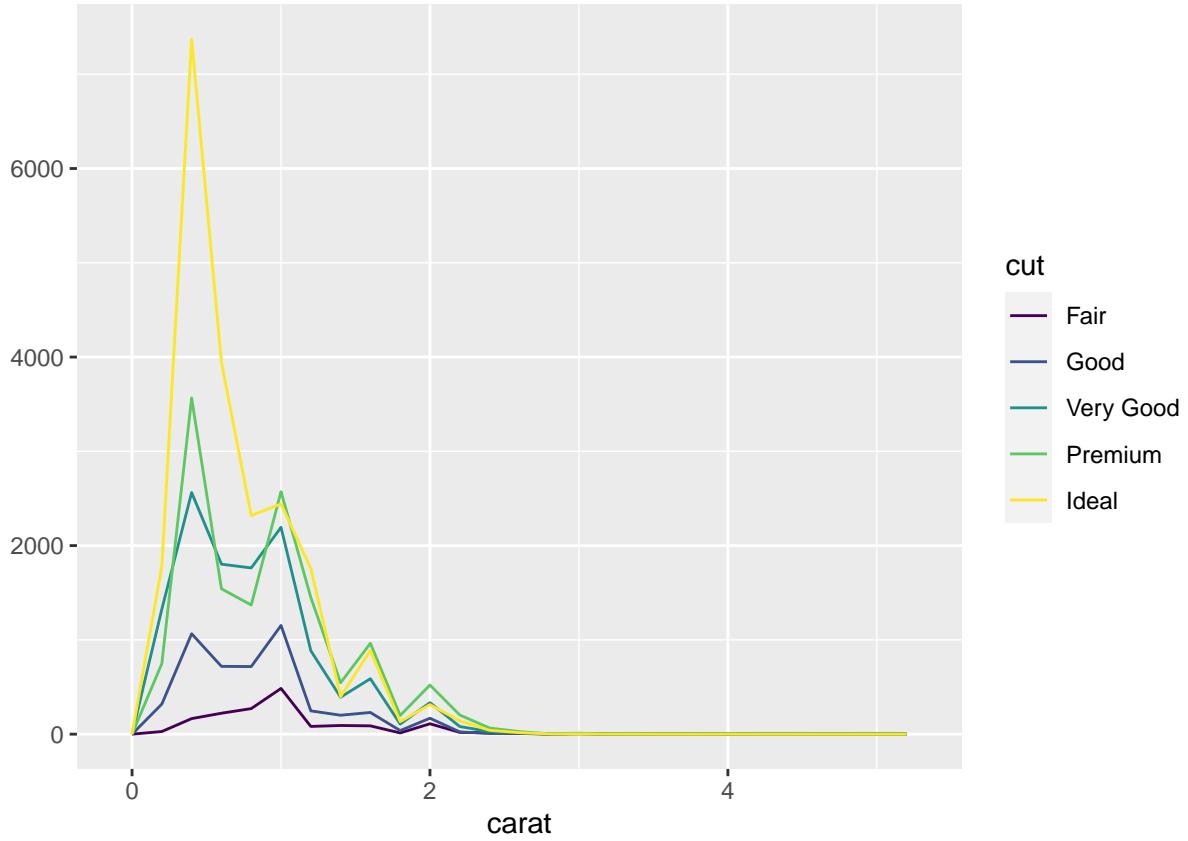
NÓTESE que NO se indica una variable “y”, dado que no es necesaria

```
qplot(
  depth,
  data = diamonds,
  binwidth = 0.2, # permite modificar la escala del encho del histograma
  geom = "histogram", # nos permite dibujar líneas entre puntos
  color = cut, # cambiar el color de fuera
  fill=cut # cambiar el color del interior
  #fill=I("lightgreen") # cambiar el color del interior
)
```



geom = "freqpoly" nos genera un polinomio de frecuencias, es decir, es una línea que pasa por los “picos” del histograma anterior.

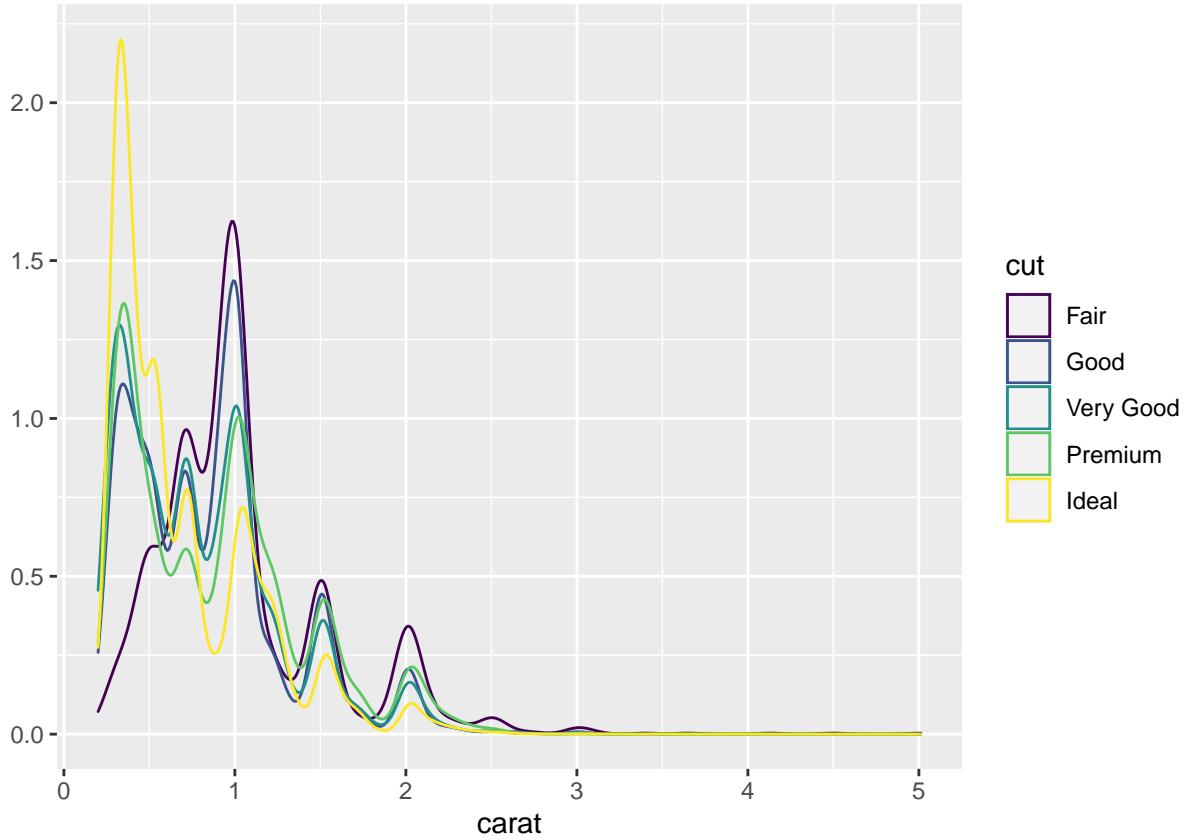
```
qplot(
  carat,
  data = diamonds,
  binwidth = 0.2, # permite modificar la escala del encho del polígono de frecuencias
  geom = "freqpoly", # nos permite dibujar líneas entre puntos
  color = cut, # cambiar el color
)
```



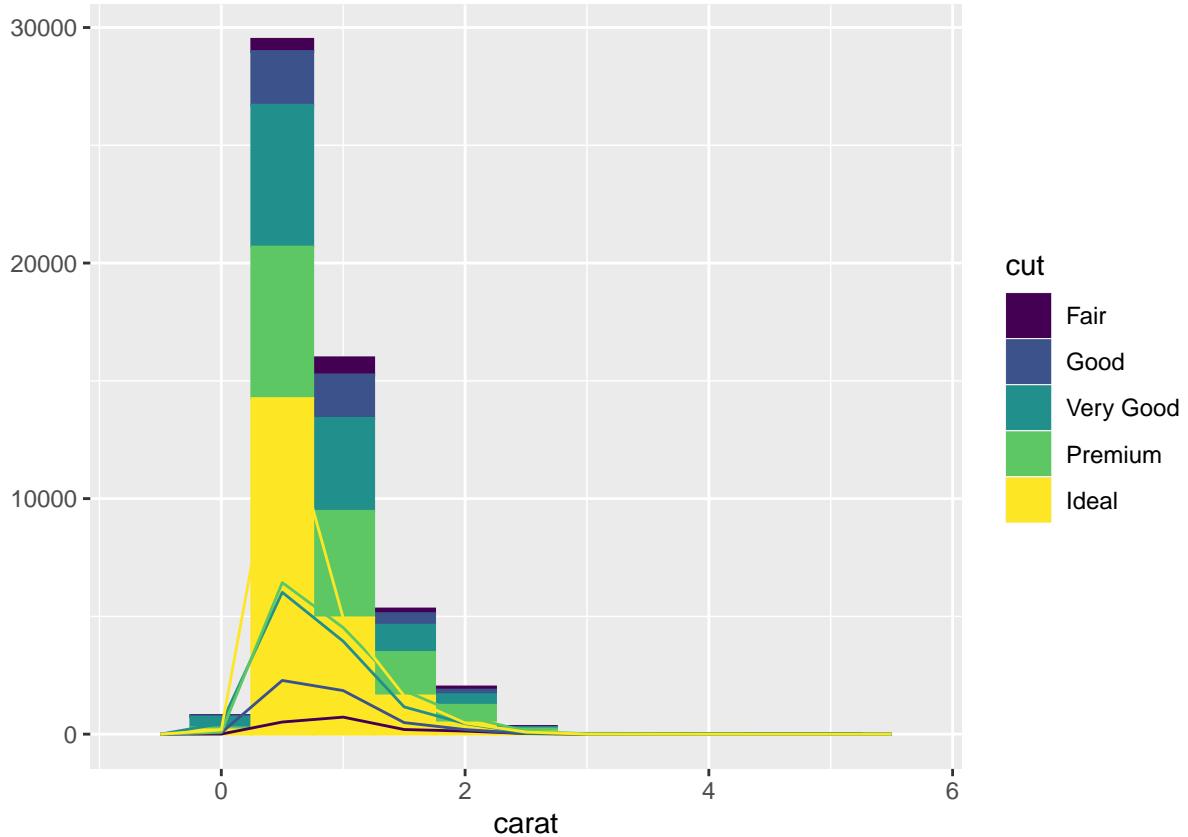
geom = “density” genera una gráfica de densidad de datos.

La diferencia con *frequency* es que en este caso son datos relativos al total (considerando 1 el 100% de datos), mientras que *frequency* nos devuelve los datos de frecuencia, es decir, el nº de veces que se repite un mismo dato.

```
qplot(
  carat,
  data = diamonds,
  geom = "density", # nos permite dibujar la densidad de los datos
  color = cut, # cambiar el color
)
```



```
qplot(
  carat,
  data = diamonds,
  binwidth = 0.5,
  geom = c("histogram", "freqpoly"), # nos permite dibujar líneas entre puntos
  color = cut, # cambiar el color de la línea
  #fill=I("lightgreen") # cambiar el color del interior
  fill=cut # cambiar el color del interior
)
```



Compilación de comandos para los gráficos

Aquí se recopilan todos los comandos utilizados a lo largo del documento, que han de ser probados como se indica anteriormente para que funcionen correctamente.

```
qplot( #función para dibujar las gráficas
      x, # columna x del df
      y, # columna y del df
      data = df, # df a utilizar, al cual pertenecen las columnas x e y
      main = "Título del gráfico", # título
      xlab = "Eje X", # título del eje x
      ylab = "Eje Y", # título del eje y
      xlim = c(3, 11), # límite inferior y superior, eliminando del gráfico los puntos que sobrepasan
      ylim = c(0, 18), # límite inferior y superior, eliminando del gráfico los puntos que sobrepasan
      color = color, # establece una serie de colores en las gráficas
      fill=cut, # cambiar el color del interior (para cierto tipo de gráficos)
      fill=I("green"), # cambiar el color del interior (para cierto tipo de gráficos)
      shape = z, # de esta manera, damos formas diferentes a los distintos parámetros de la columna z
      alpha = I(1/100), # este comando nos permite reducir la densidad de puntos en la gráfica local
      size = z, #pinta distintos tamaños de punto para cada una de las variables cuantitativas de la gráfica
      facets = z ~ ., # nos subdivide el gráfico en tantos gráficos diferentes como campos cualitativos haya
      asp = 0.8, # modifica el ratio y/x del gráfico, disminuyendo la dimensión horizontal y haciendo que la gráfica sea más vertical
      label = rownames(diamonds), # útil si uncluimos puntos con texto sobre el gráfico
      geom = c("point", "smooth", "text", "line"), # nos dibuja sobre la gráfica los puntos (point),
      geom = "jitter", # nos permite observar por cajas los datos que queremos
      geom = "boxplot", # nos permite observar gráficos de cajas por las categorías que nosotros hemos establecido
```

```
geom = "line", # nos dibuja líneas entre dos puntos, de manera que nos muestra los puntos unidos
geom = "path", # hace lo mismo que line, pero sin ir precisamente de izquierda a derecha
geom = "histogram", # nos permite visualizar los valores de una variable por medio de histogramas
geom = "freqpoly", # genera un polinomio de frecuencias
geom = "density" # dibuja una gráfica de densidad relativa
)
```