ESTADISTICA

COMO GRAFICAR EN R

Gráficos de Barras (Barplot)

Uno de los tipos de gráficos más comunes en el anállisis de datos es el gráfico de barras, en el que simplemente se representa con barras de distintas “alturas” las dimensiones de una cantidad numérica comparada con otra. Por ejemplo, tomemos como dataset inicial el `mtcars`, cuyos primeros valores son:

Hide

```
head(mtcars)
```

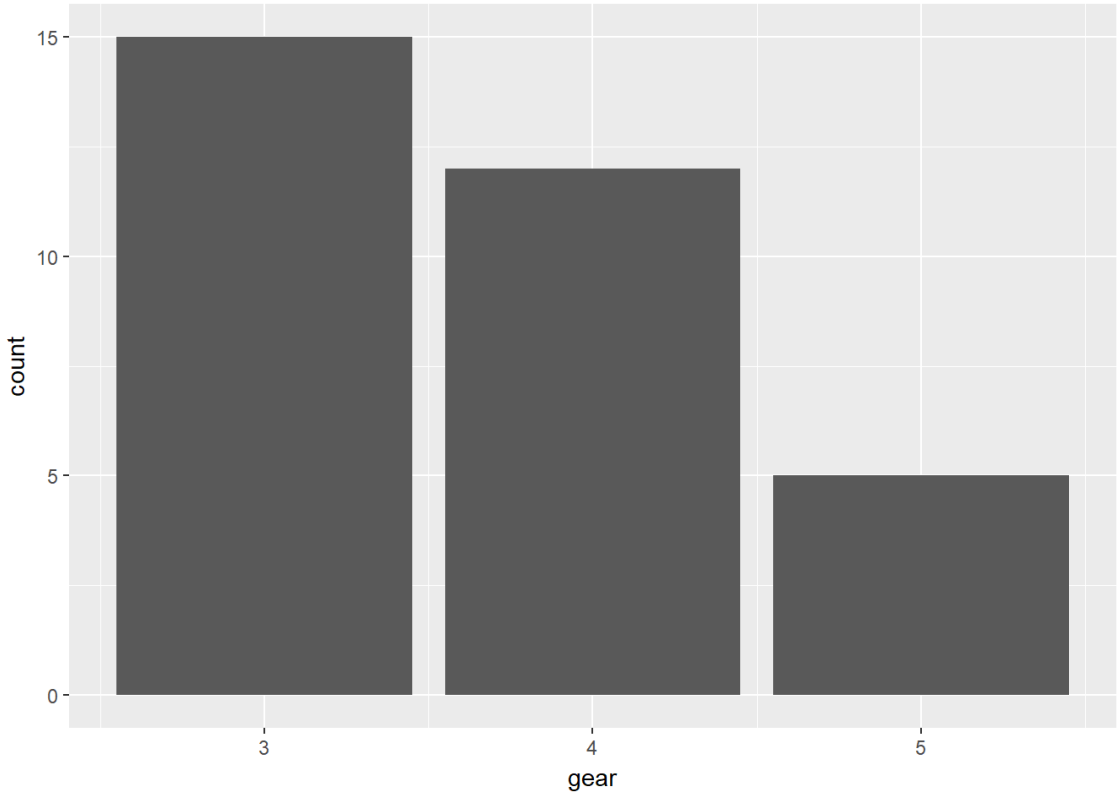
```
##                    mpg cyl disp  hp drat    wt  qsec vs am gear car
b
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4
4
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4
4
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4
1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3
1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3
2
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3
1
```

Y que contiene información técnica de distintos tipos de marca y modelos de vehículos.

Grafiquemos entonces el total de autos en el dataset agrupados por la cantidad de “cambios o velocidades” (columna ‘gear’):

Hide

```
ggplot(data = mtcars, aes(x = gear)) + geom_bar()
```

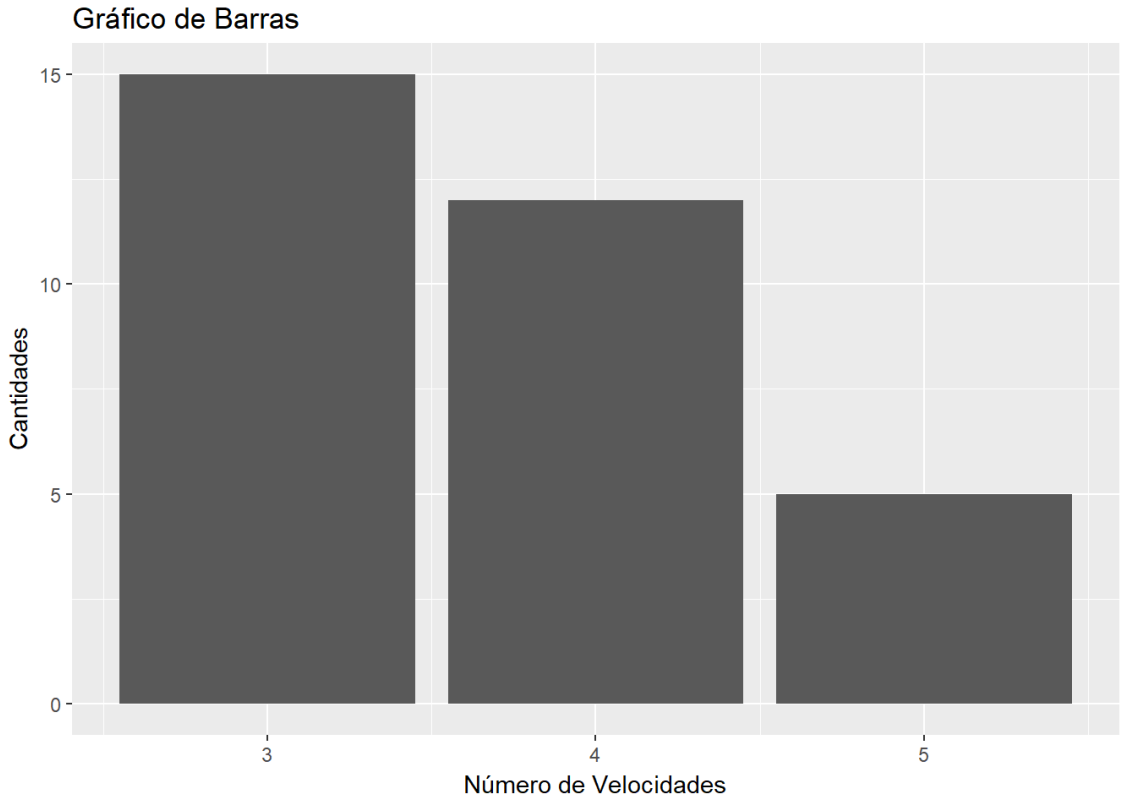


Como se observa en el ejemplo, todo gráfico hecho con ggplot debe tener un primer argumento `ggplot()` en donde debe especificarse cuál es el dataset que contiene la información que se desea graficar. Esta representa la primera capa que guarda la información del conjunto de datos de partida. Con el argumento data = mtcars se establece que el dataset es el indicado, y en términos de R este conjunto de datos puede ser cualquier dataframe construido o cargado previamente. Luego, se observa en el ejemplo el comando `aes()`, que se refiere a la "estética" del gráfico, es decir, en este caso se especifica o "mapea" cuál variable del conjunto de datos es el que se va a representar en el eje "x". Una vez completada esta función, se agrega una segunda capa usando el operador + , y luego se establece que el gráfico a construir es de tipo barra con la función `geom_bar()`. Con solo esos argumentos es suficiente para producir el gráfico mostrado en el ejemplo. Ahora bien, aunque el gráfico enumera las cantidades de autos

con 3, 4 y 5 velocidades que forman parte del conjunto de datos, vamos a modificar ciertas características visuales iniciales a fin de obtener un resultado final más agradable. Por ejemplo, cambiemos las etiquetas de los ejes y agregemos un título al gráfico:

Hide

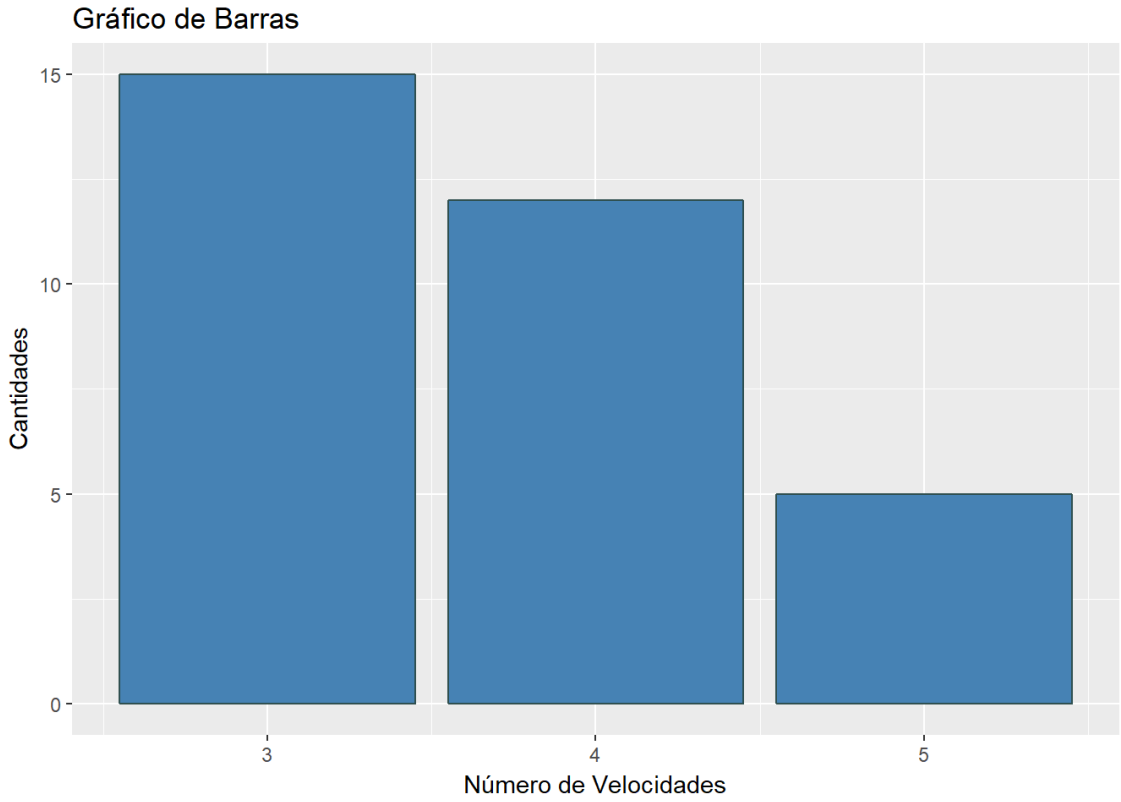
```
ggplot(data = mtcars, aes(x = gear)) +
  geom_bar() +
  xlab("Número de Velocidades") +
  ylab("Cantidades") +
  ggtitle("Gráfico de Barras")
```



Obsérvese que esto se logra agregrando capas con el operador + , e incorporando los elementos deseados. El gráfico luce mucho mejor, pero tal vez el color gris por defecto no resulta muy atractivo. Si lo deseamos, podemos cambiar el color con el que se representan tanto el área interior de las barras, como su borde, colocando explícitamente los valores dentro de la función `geom_bar()`:

Hide

```
ggplot(data = mtcars, aes(x = gear)) +
  geom_bar(color = 'darkslategray', fill = 'steelblue') +
  xlab("Número de Velocidades") +
  ylab("Cantidades") +
  ggtitle("Gráfico de Barras")
```

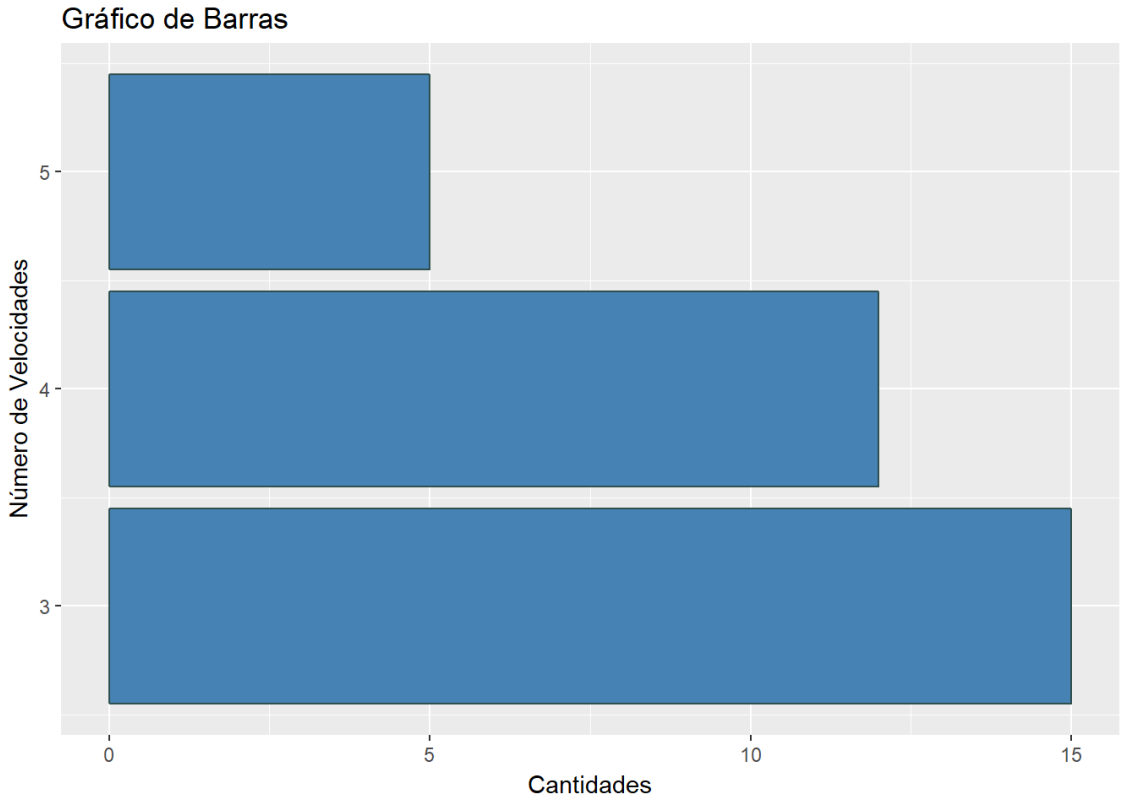


Como referencia para los nombres de los colores usados, recomiendo revisar el siguiente link.

Y si el gráfico lo amerita, podemos también cambiar la orientación de los ejes haciendo uso de la función `coord_flip()`:

Hide

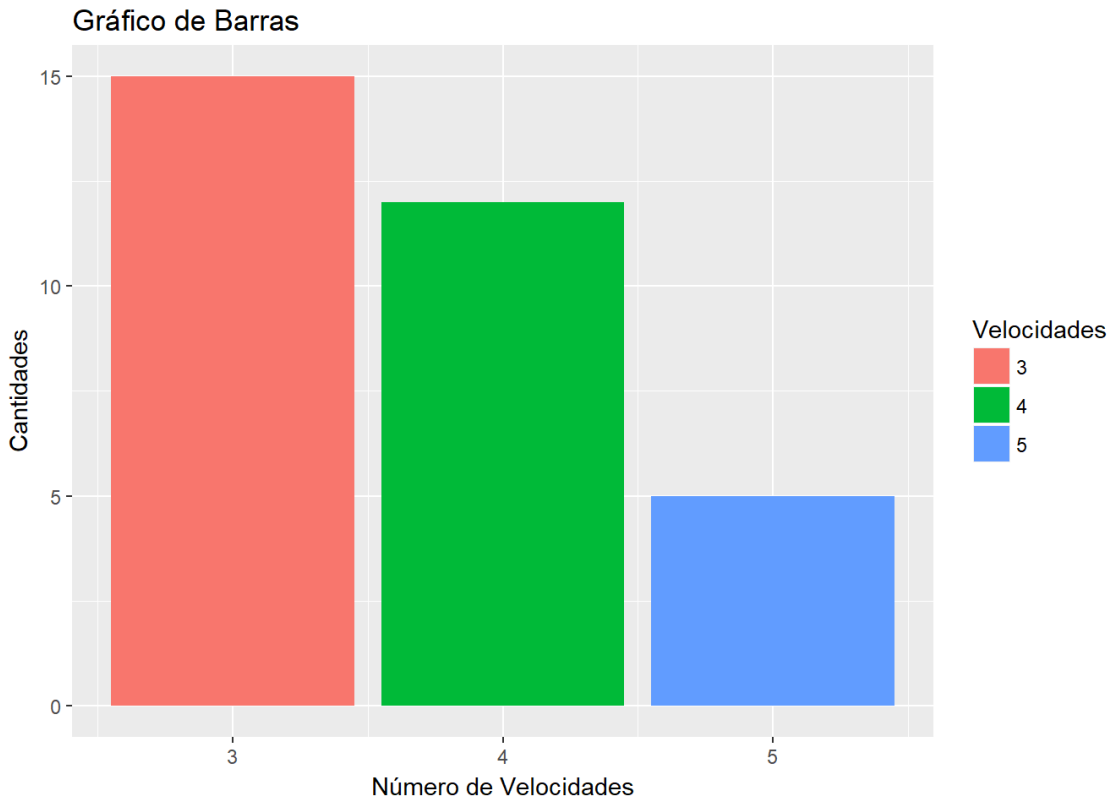
```
ggplot(data = mtcars, aes(x = gear)) +
  geom_bar(color = 'darkslategray', fill = 'steelblue') +
  xlab("Número de Velocidades") +
  ylab("Cantidades") +
  ggtitle("Gráfico de Barras") +
  coord_flip()
```



Ahora bien, si deseamos que cada grupo de datos (velocidades, en este caso) tenga colores distintos, basta con asignar, dentro de la “estética”, el color como parámetro, y mapearlo a alguna variable. En este caso, al propio valor de “gear”, pero dicho valor debe ser convertido antes a una variable tipo `factor`, es decir, una variable categórica:

Hide

```
ggplot(data = mtcars, aes(x = gear, fill = as.factor(gear))) +
  geom_bar() +
  xlab("Número de Velocidades") +
  ylab("Cantidades") +
  ggtitle("Gráfico de Barras") +
  labs(fill = "Velocidades")
```

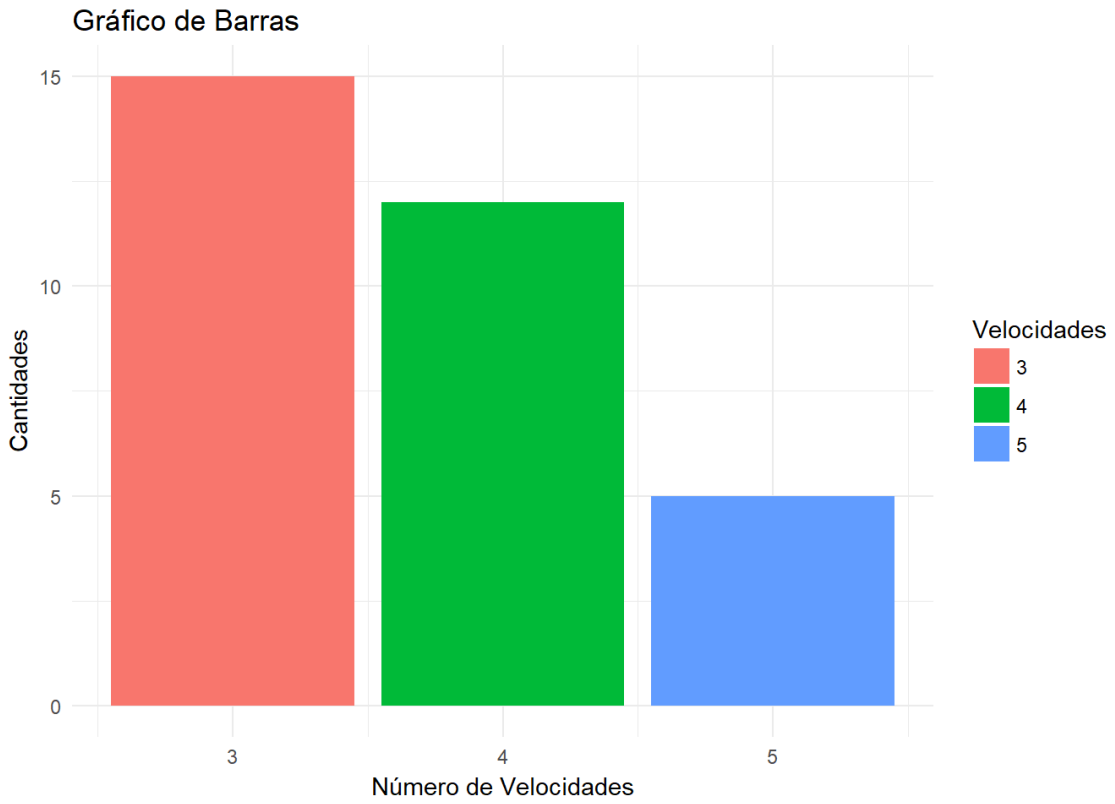


En este caso, se introdujo la función `labs(fill = ... )` para cambiar el título de la leyenda. De lo contrario, en dicha leyenda aparecería `as.factor(gear)`.

Si queremos, adicionalmente, cambiar el estilo del fondo, podemos hacerlo seleccionando el “tema” de visualización de la siguiente manera:

Hide

```
ggplot(data = mtcars, aes(x = gear, fill = as.factor(gear))) +
  geom_bar() +
  xlab("Número de Velocidades") +
  ylab("Cantidades") +
  ggtitle("Gráfico de Barras") +
  labs(fill = "Velocidades") +
  theme_minimal()
```



La lista completa con todos los temas disponibles por defecto se encuentra en el siguiente link.

Por último (y esto es aplicable a cualquier otro gráfico que mostremos a lo largo de todo el ejercicio práctico), es importante mencionar que cualquier indicación de parámetros (por ejemplo, la "estética") que coloquemos en la función `ggplot()`, será sobreescrita si modificamos los argumentos de la función `geom_...` posterior. Es decir, veamos el siguiente ejemplo:

Hide

```
ggplot(data = mtcars, aes(x = gear, fill = as.factor(gear))) +
  geom_bar(color = 'slateblue', fill = 'skyblue2') +
  xlab("Número de Velocidades") +
  ylab("Cantidades") +
  ggtitle("Gráfico de Barras") +
  labs(fill = "Velocidades") +
  theme_minimal()
```

Como puede observarse, aún cuando en la primera función se estableció el relleno de las barras en función de los valores de la variable 'gear', al fijar el color y el relleno de nuevo en la función `geom_bar()` se tomará esta última como aquella a aplicar al gráfico.

Histograma (Histogram) y Gráfico de Densidad (Density Plot)

Un histograma es un tipo de gráfico de barras en donde la altura de éstas hacen referencia a la frecuencia con la que aparecen los valores que se representan. Los histogramas son utilizados en la mayoría de las ocasiones para observar la distribución de alguna variable continua, lo que nos permite de una manera sencilla y rápida obtener información como el comportamiento de los datos, tendencias, variabilidad u homogeneidades, entre otros. A efectos de conocer cómo se implementan los histogramas con `ggplot2`, vamos a hacer uso esta vez del dataset `diamonds`, cuyos primeros valores son:

Hide

```
head(diamonds)
```

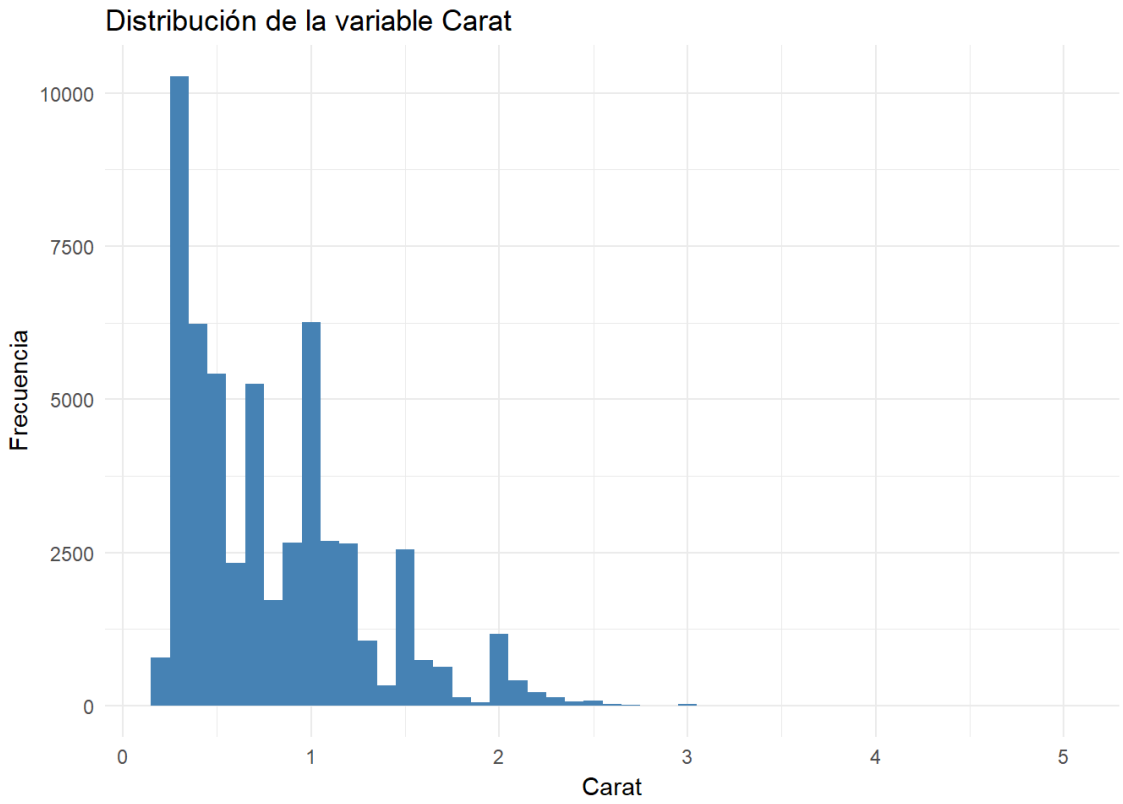
```
## # A tibble: 6 x 10
##   carat cut       color clarity depth table price     x     y     z
##   <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
## 1 0.23  Ideal     E     SI2      61.5    55   326  3.95  3.98  2.43
## 2 0.21  Premium   E     SI1      59.8    61   326  3.89  3.84  2.31
## 3 0.23  Good      E     VS1      56.9    65   327  4.05  4.07  2.31
## 4 0.290 Premium   I     VS2      62.4    58   334  4.2   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
```

Y que contiene información sobre los distintos tipos de cortes de los diamantes, su color y precio, entre otros.

Un primer histograma que podemos hacer a partir de este conjunto es ver el cómo están distribuidos los datos de la variable `carat`:

Hide

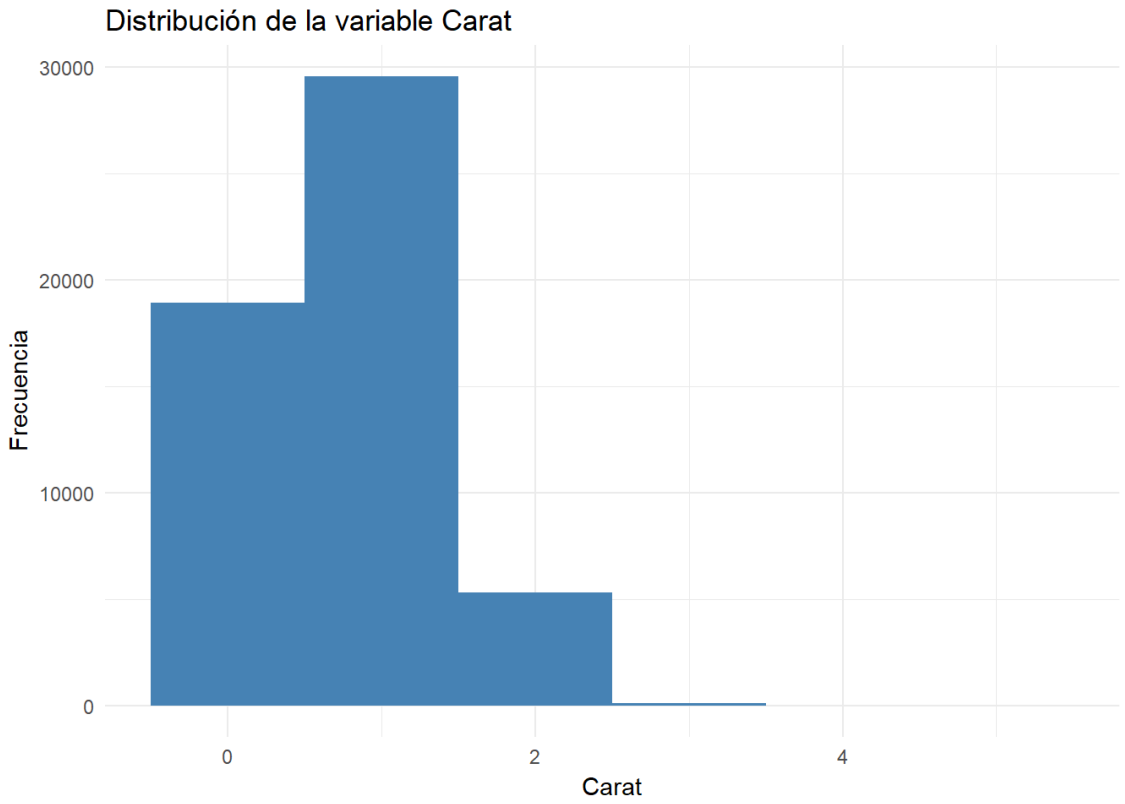
```
ggplot(diamonds) +
  geom_histogram(binwidth = 0.1, aes(x = carat), fill = 'steelblue') +
  xlab("Carat") +
  ylab("Frecuencia") +
  ggtitle("Distribución de la variable Carat") +
  theme_minimal()
```



Como puede observarse, el histograma nos permite ver una distribución sesgada a la derecha, con la mayoría de los valores agrupados en el rango 0 a 1 de la variable carat. Sin embargo, como se observa en la función `geom_histogram()`, uno de los argumentos es el 'bidwidth', es decir, el ancho de las barras que recogen los rangos de representación de la variable. Este argumento es importante ya que de no establecer un valor adecuado, se puede perder la forma o los detalles de la distribución. Por ejemplo, si el ancho es demasiado grande:

Hide

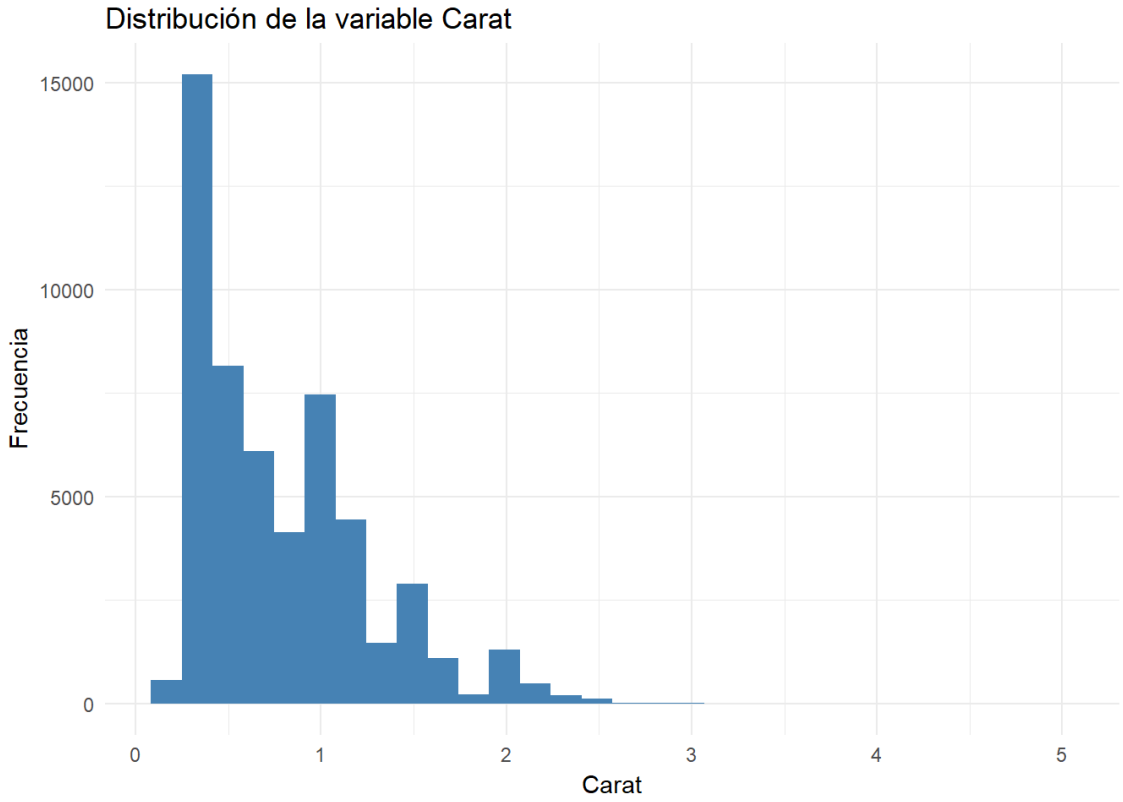
```
ggplot(diamonds) +
  geom_histogram(binwidth = 1, aes(x = carat), fill = 'steelblue') +
  xlab("Carat") +
  ylab("Frecuencia") +
  ggtitle("Distribución de la variable Carat") +
  theme_minimal()
```



El gráfico resultante da una idea de la distribución, pero la información interna de picos o fluctuaciones se pierde. Si no se especifica el valor del ancho, la función tomará un valor por defecto para, en esta oportunidad, la cantidad de `bins`, y arrojará un mensaje:

Hide

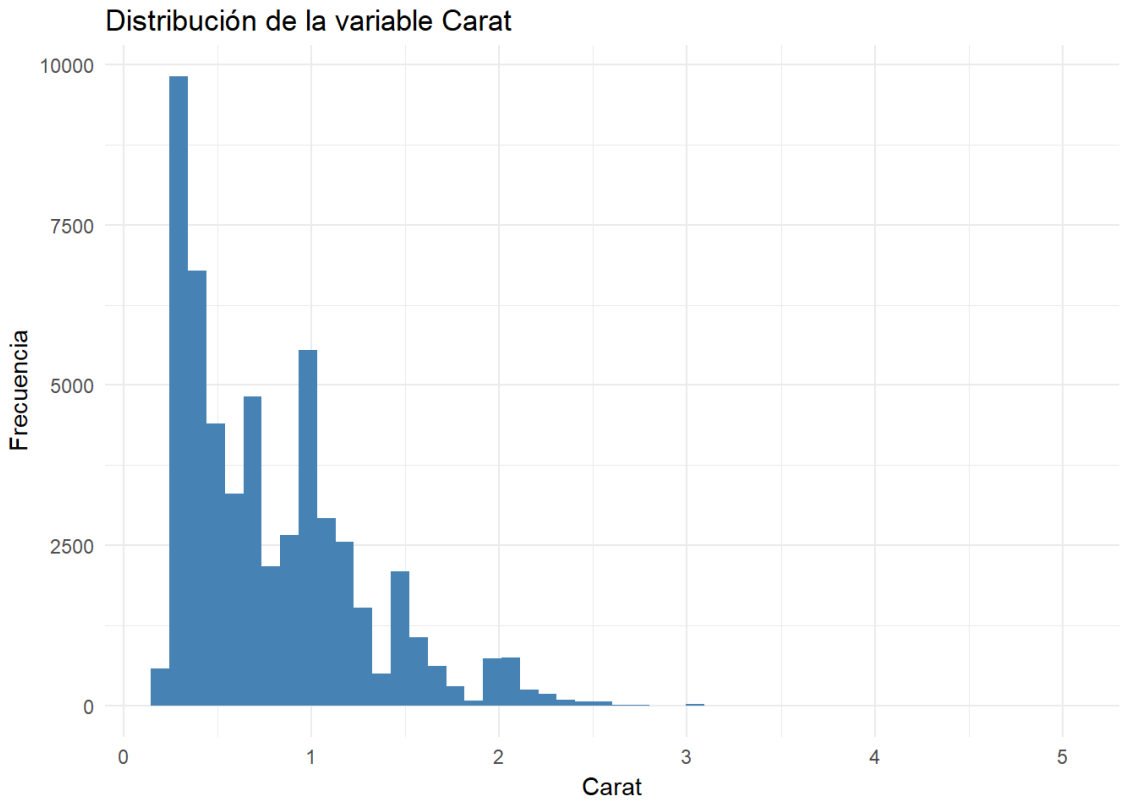
```
ggplot(diamonds) +
  geom_histogram(aes(x = carat), fill = 'steelblue') +
  xlab("Carat") +
  ylab("Frecuencia") +
  ggtitle("Distribución de la variable Carat") +
  theme_minimal()
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



De modo que, para el histograma, podemos establecer tanto la cantidad de barras tanto por el argumento `bins` como por el `binwith`:

Hide

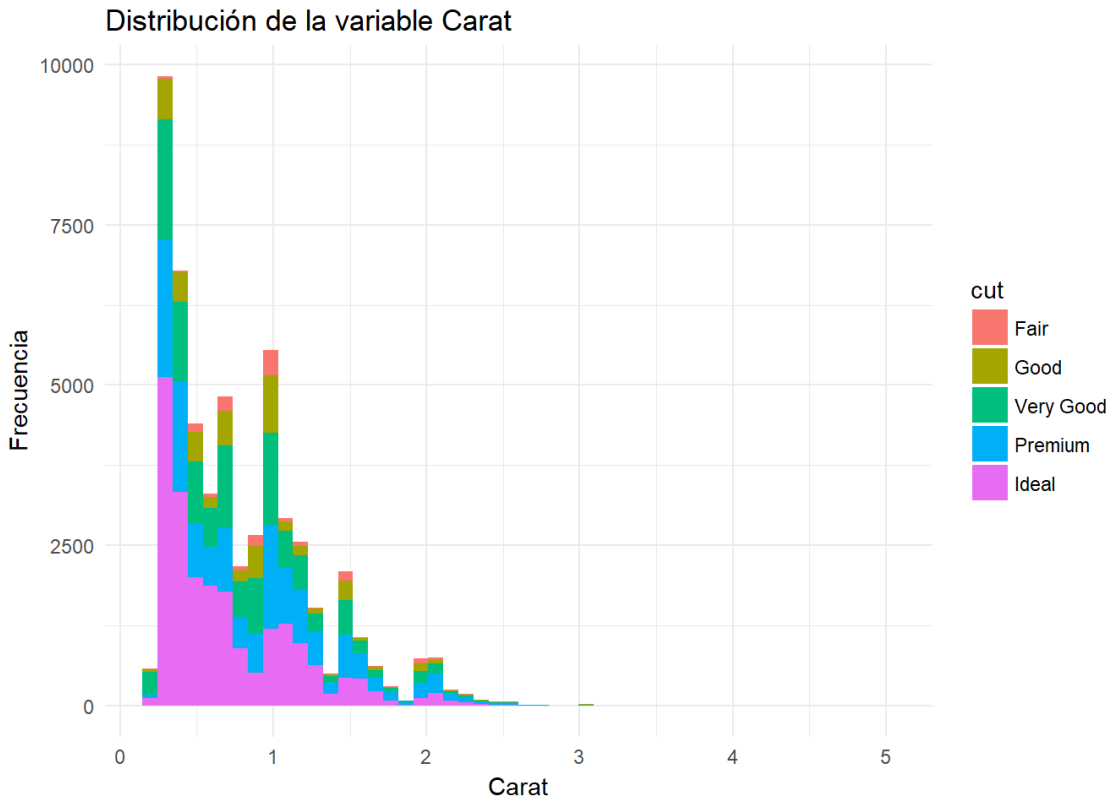
```
ggplot(diamonds) +
  geom_histogram(bins = 50, aes(x = carat), fill = 'steelblue') +
  xlab("Carat") +
  ylab("Frecuencia") +
  ggtitle("Distribución de la variable Carat") +
  theme_minimal()
```



Ahora bien, dado que el dataset original contiene, por ejemplo, información categórica del tipo de corte de los diamantes, podemos aprovechar esta variable para representar, en distintos colores, cada una de estas categorías. Para ello, “mapeamos” la variable `cut` al argumento `fill` de la función:

Hide

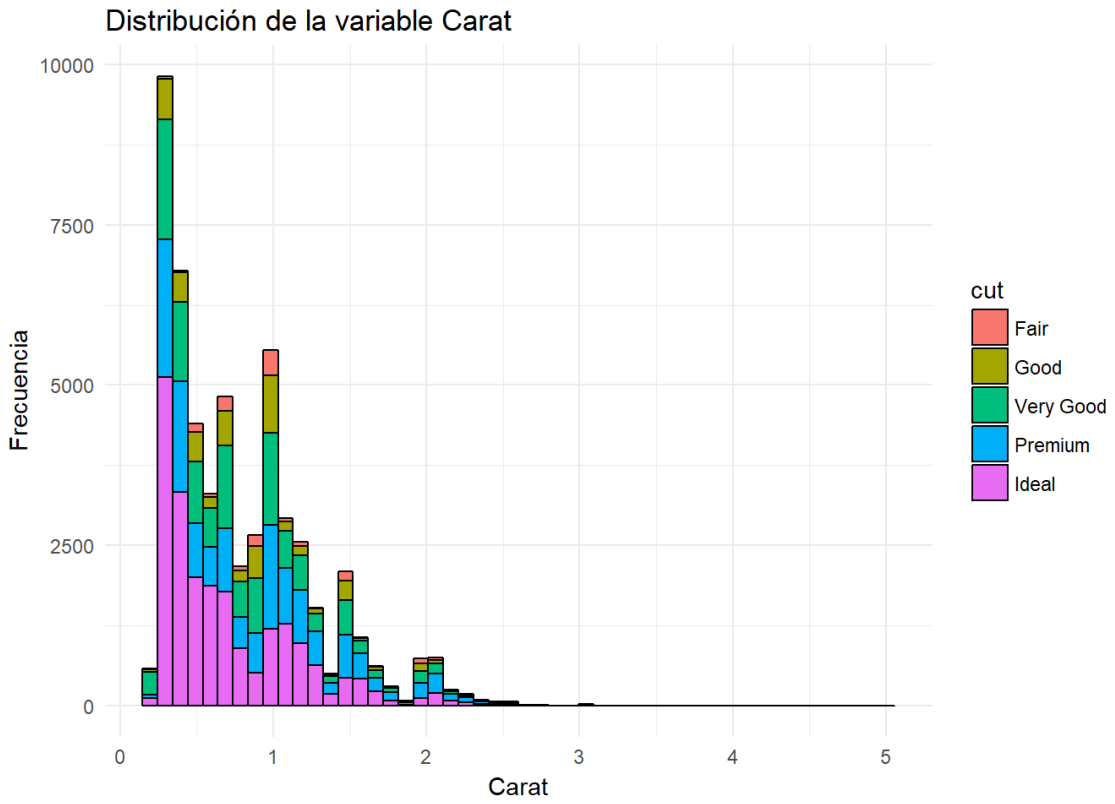
```
ggplot(diamonds) +
  geom_histogram(bins = 50, aes(x = carat, fill = cut)) +
  xlab("Carat") +
  ylab("Frecuencia") +
  ggtitle("Distribución de la variable Carat") +
  theme_minimal()
```



De esta manera podemos observar las distribuciones de cada tipo de corte según cada grupo de barras del histograma. A fin de lograr una mejor distinción de las regiones de cada categoría, podemos incorporar un borde negro a las barras:

Hide

```
ggplot(diamonds) +
  geom_histogram(bins = 50, aes(x = carat, fill = cut), color = 'black
') +
  xlab("Carat") +
  ylab("Frecuencia") +
  ggtitle("Distribución de la variable Carat") +
  theme_minimal()
```

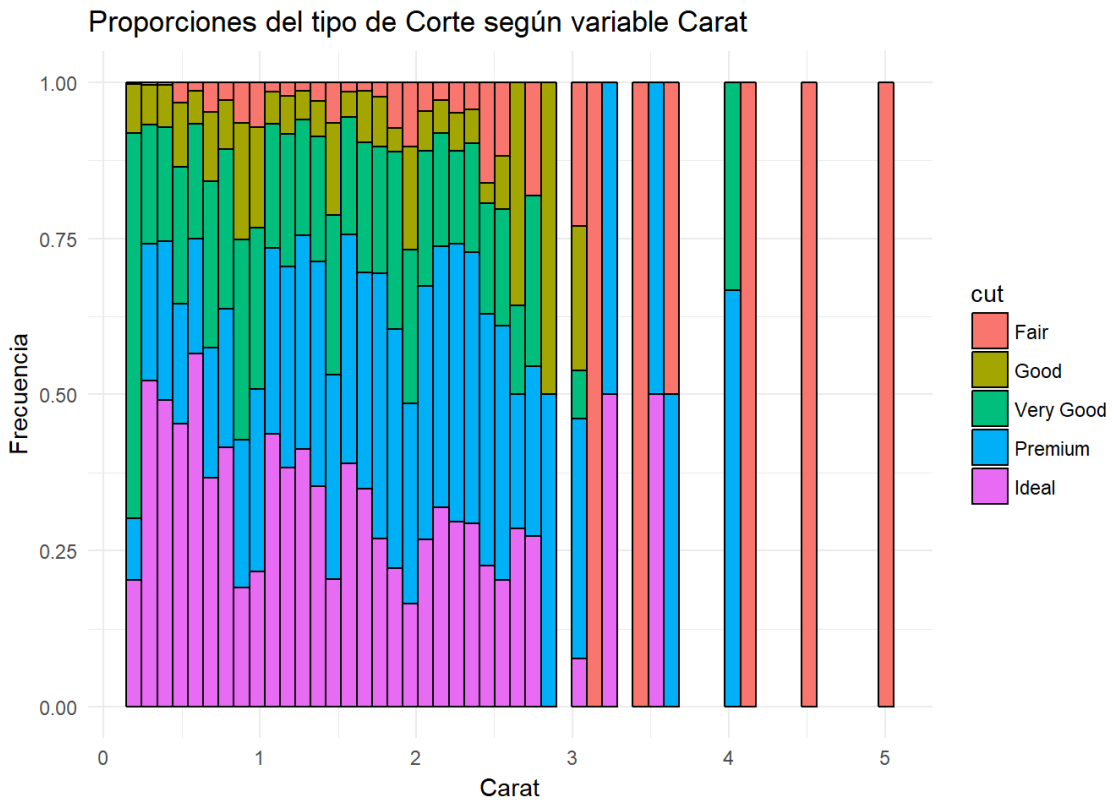


Con lo que obtenemos una muy buena visualización de la distribución de la variable `carat`, así como de las proporciones del tipo de corte, o variable `cut`.

Para todos estos casos se observa que el eje de la 'Frecuencia' cuenta la cantidad total de ocurrencias de la variable en cada caso. Sin embargo, otra opción posible sería visualizar las proporciones de las distintas categorías de la variable `cut` normalizando la escala del eje 'y'. Para ello, incorporamos el argumento `position` en la función:

Hide

```
ggplot(diamonds) +
  geom_histogram(bins = 50, aes(x = carat, fill = cut), color = 'black
', position = 'fill') +
  xlab("Carat") +
  ylab("Frecuencia") +
  ggtitle("Proporciones del tipo de Corte según variable Carat") +
  theme_minimal()
```

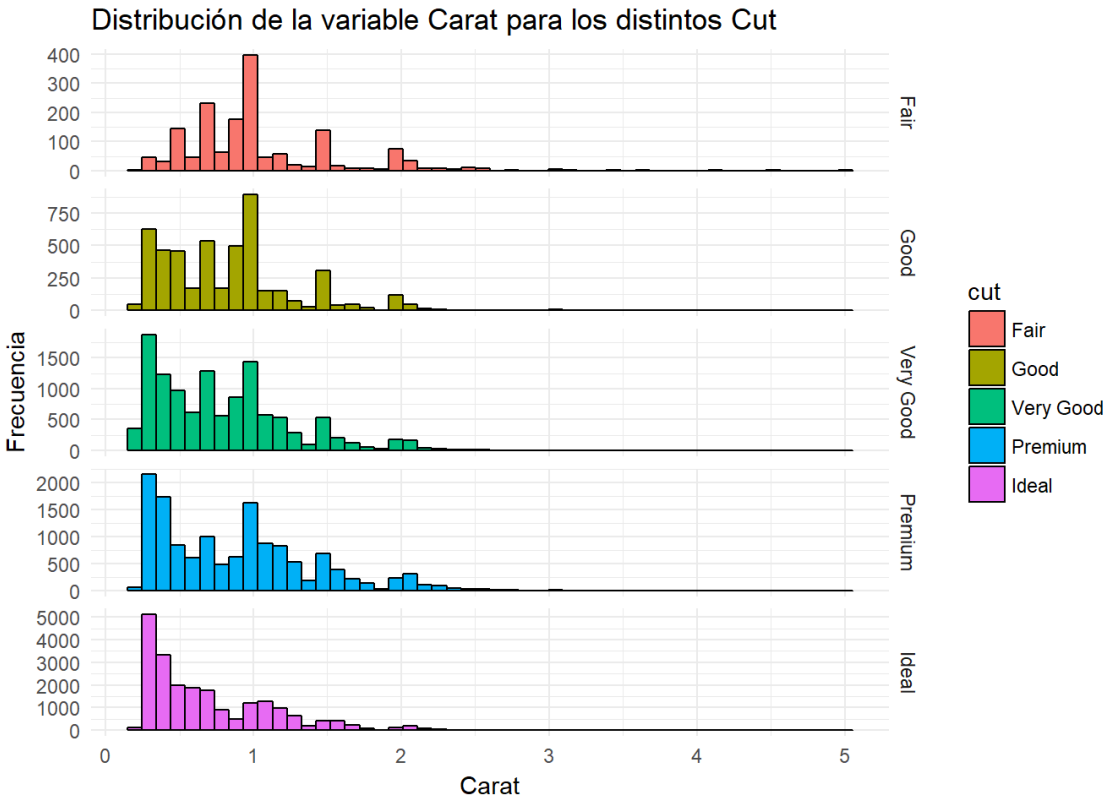


En este caso, si bien la gráfica no nos permite tener una visualización clara de la distribución de los valores, sí podemos conocer las proporciones entre los distintos tipos de 'corte' presentes en los datos según el tipo de 'carat'.

Si, en su lugar, tenemos la necesidad de observar las distribuciones por separado para cada tipo de corte, entonces podemos hacer uso de la función `facet_grid()` para separar los histogramas en filas (o columnas). Por ejemplo:

Hide

```
ggplot(diamonds) +
  geom_histogram(bins = 50, aes(x = carat, fill = cut), color = 'black
') +
  facet_grid(cut~., scales = 'free') +
  xlab("Carat") +
  ylab("Frecuencia") +
  ggtitle("Distribución de la variable Carat para los distintos Cut")
+
  theme_minimal()
```

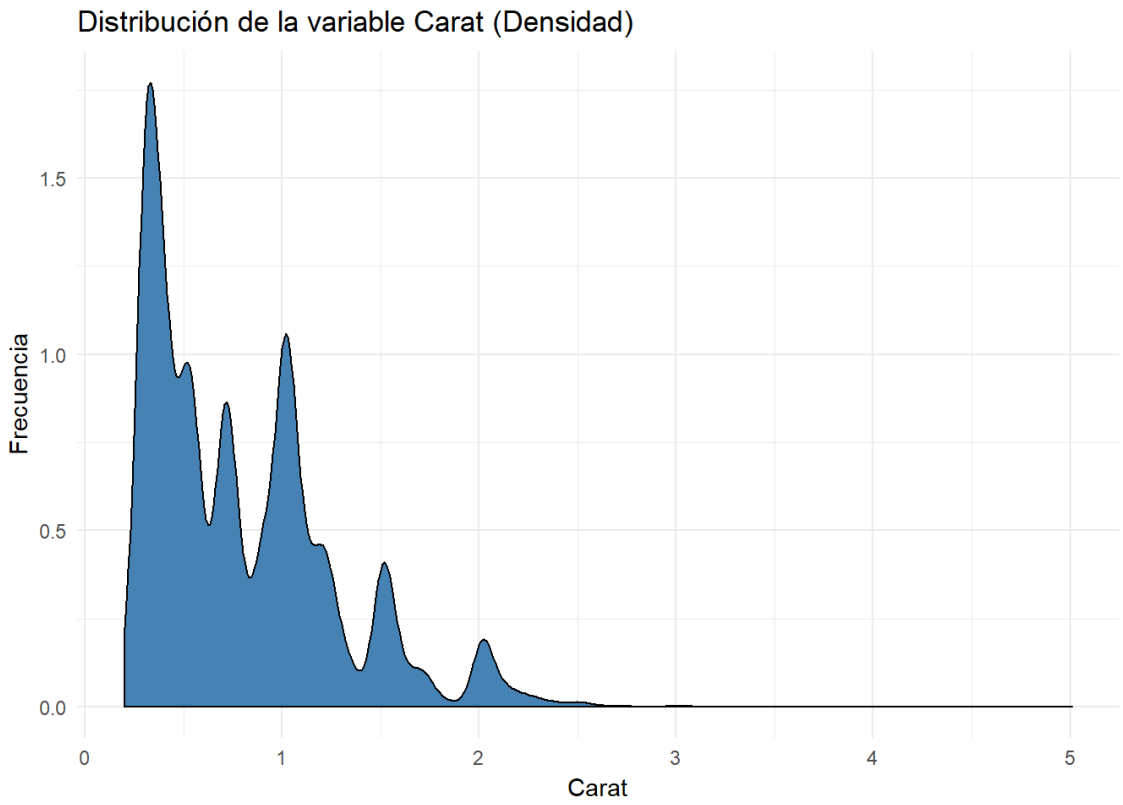


En este caso, el primer argumento de la función `facet_grid()` hace referencia a qué se va a representar en las filas y columnas. Así, la expresión `cut~.` puede leerse como: la variable 'cut' en cada fila, el resto en las columnas. Más adelante veremos otras formas de usar y entender esto con otros ejemplos. Por otro lado, el argumento `scales = 'free'` lo que produce es que cada histograma por separado se reescala de manera automática para lograr una mejor representación de los datos.

Del mismo modo, así como podemos usar la función `geom_histogram()` para representar el histograma, podemos emplear la función `geom_density()` para trazar un gráfico de densidad que asemeja una distribución continua:

Hide

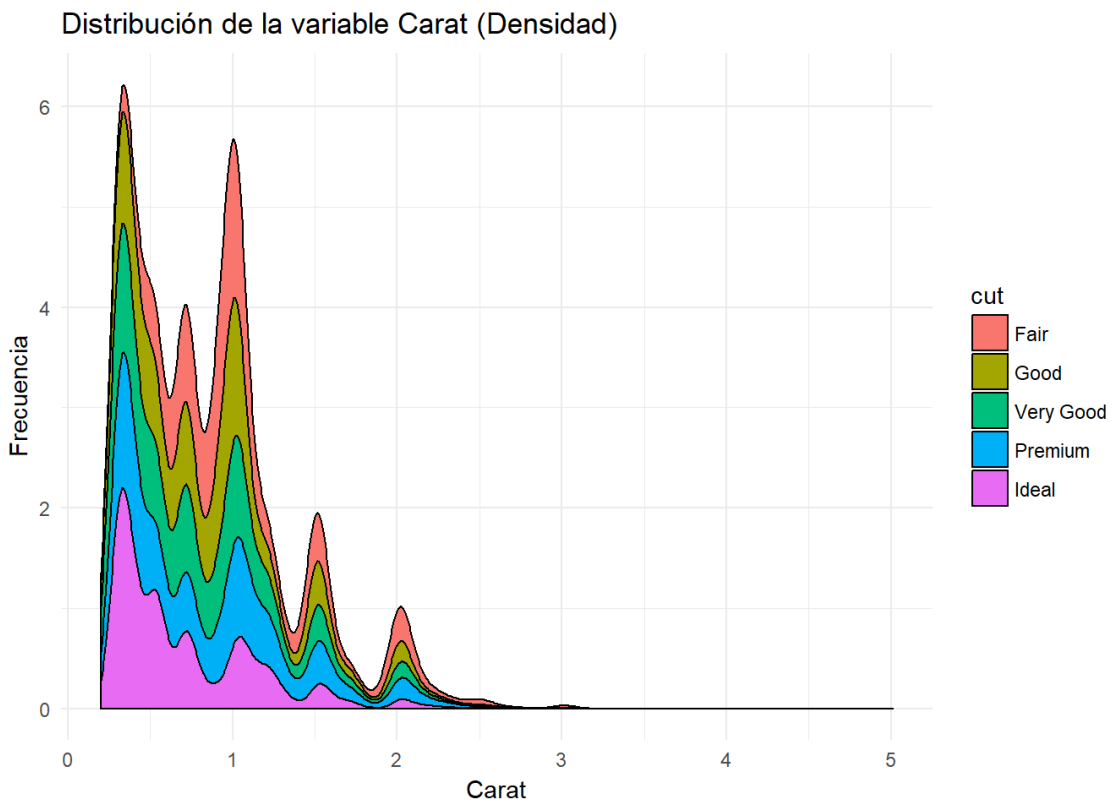
```
ggplot(diamonds) +
  geom_density(aes(x = carat), fill = 'steelblue') +
  xlab("Carat") +
  ylab("Frecuencia") +
  ggtitle("Distribución de la variable Carat (Densidad)") +
  theme_minimal()
```



Y también podemos incorporar a la estética el mapeo a otra variable categórica:

Hide

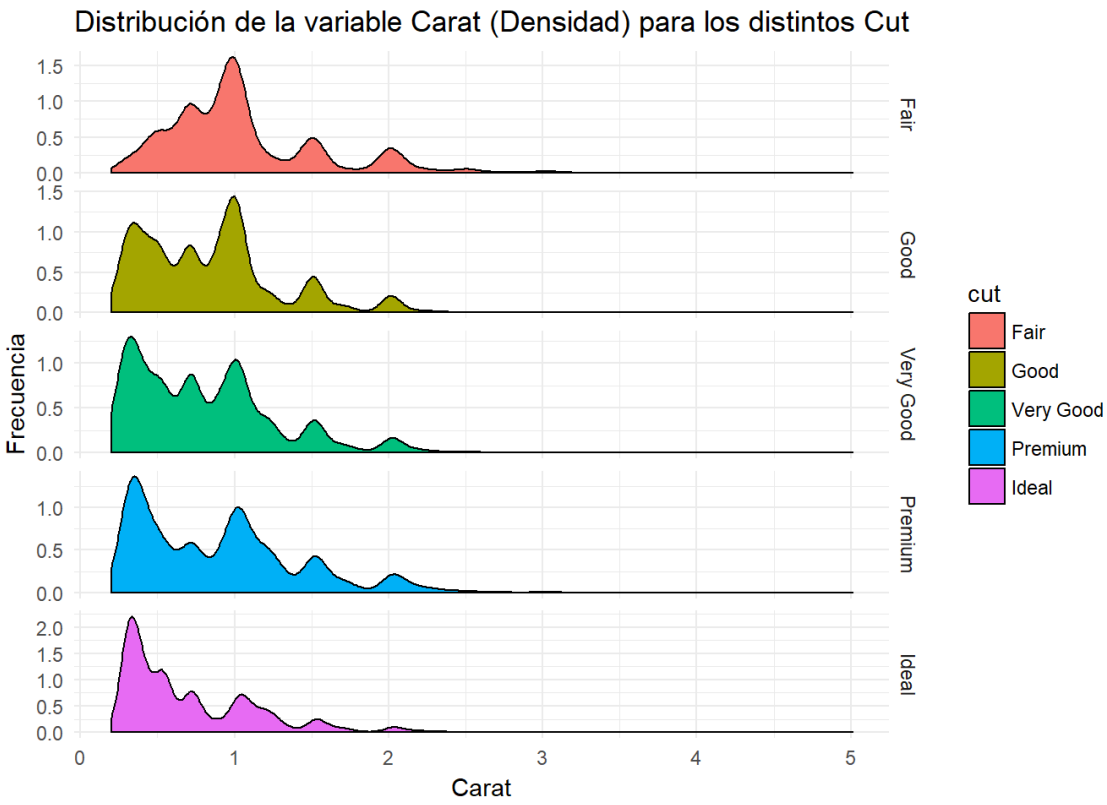
```
ggplot(diamonds) +
  geom_density(aes(x = carat, fill = cut), position = 'stack') +
  xlab("Carat") +
  ylab("Frecuencia") +
  ggtitle("Distribución de la variable Carat (Densidad)") +
  theme_minimal()
```



O bien graficarlo por separado para cada caso de la variable `cut`:

Hide

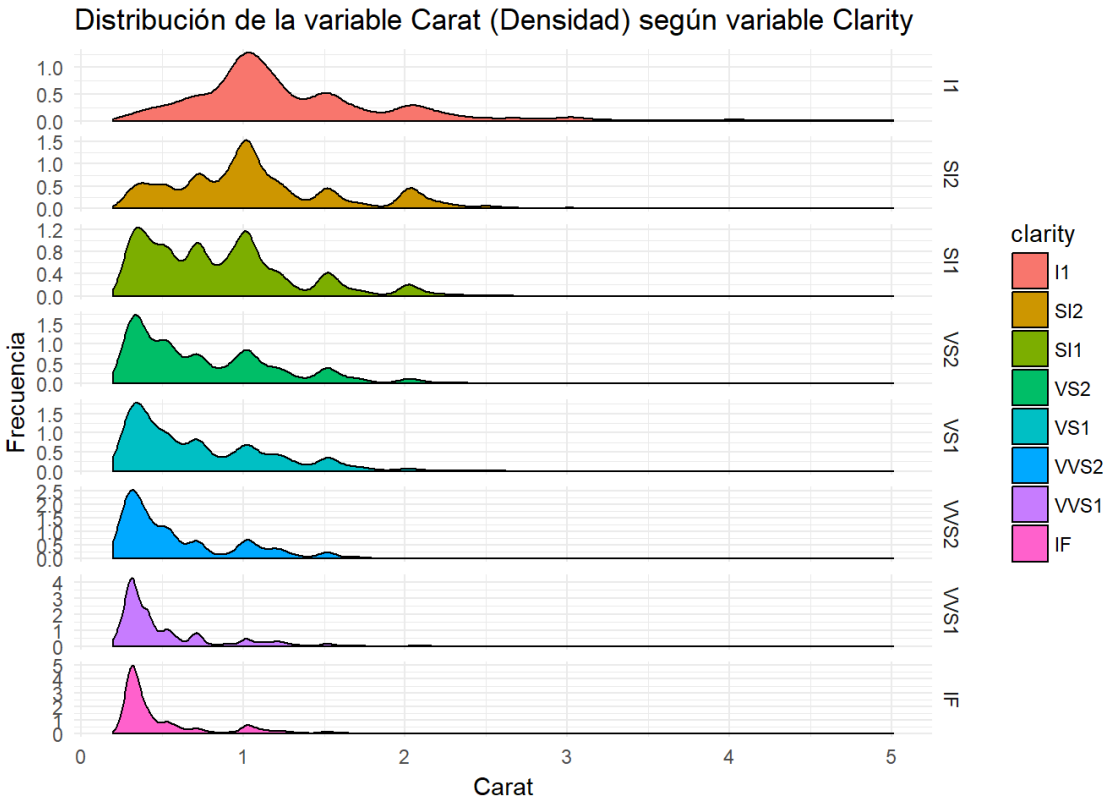
```
ggplot(diamonds) +
  geom_density(aes(x = carat, fill = cut), position = 'stack') +
  facet_grid(cut~., scales = 'free') +
  xlab("Carat") +
  ylab("Frecuencia") +
  ggtitle("Distribución de la variable Carat (Densidad) para los distintos Cut") +
  theme_minimal()
```



Como podemos ver, tanto con el histograma como con el gráfico de densidad podemos tener información adecuada de cómo se distribuyen los datos asociados a una variable determinada y podemos separar la visualización a partir de alguna otra variable de tipo categórica que nos permita obtener información adicional de dataset que se estudie. Por ejemplo, aplicando la misma visualización pero asociando el color a las variables `clarity` o `color` del dataset `diamonds` tenemos:

Hide

```
ggplot(diamonds) +
  geom_density(aes(x = carat, fill = clarity), position = 'stack') +
  facet_grid(clarity~., scales = 'free') +
  xlab("Carat") +
  ylab("Frecuencia") +
  ggtitle("Distribución de la variable Carat (Densidad) según variable Clarity") +
  theme_minimal()
```



También es posible ocultar la leyenda de la siguiente manera:

Hide

```
ggplot(diamonds) +
  geom_density(aes(x = carat, fill = color), position = 'stack') +
  facet_grid(color~., scales = 'free') +
  xlab("Carat") +
  ylab("Frecuencia") +
  ggtitle("Distribución de la variable Carat (Densidad) según variable Color") +
  theme_minimal() +
  theme(legend.position="none")
```

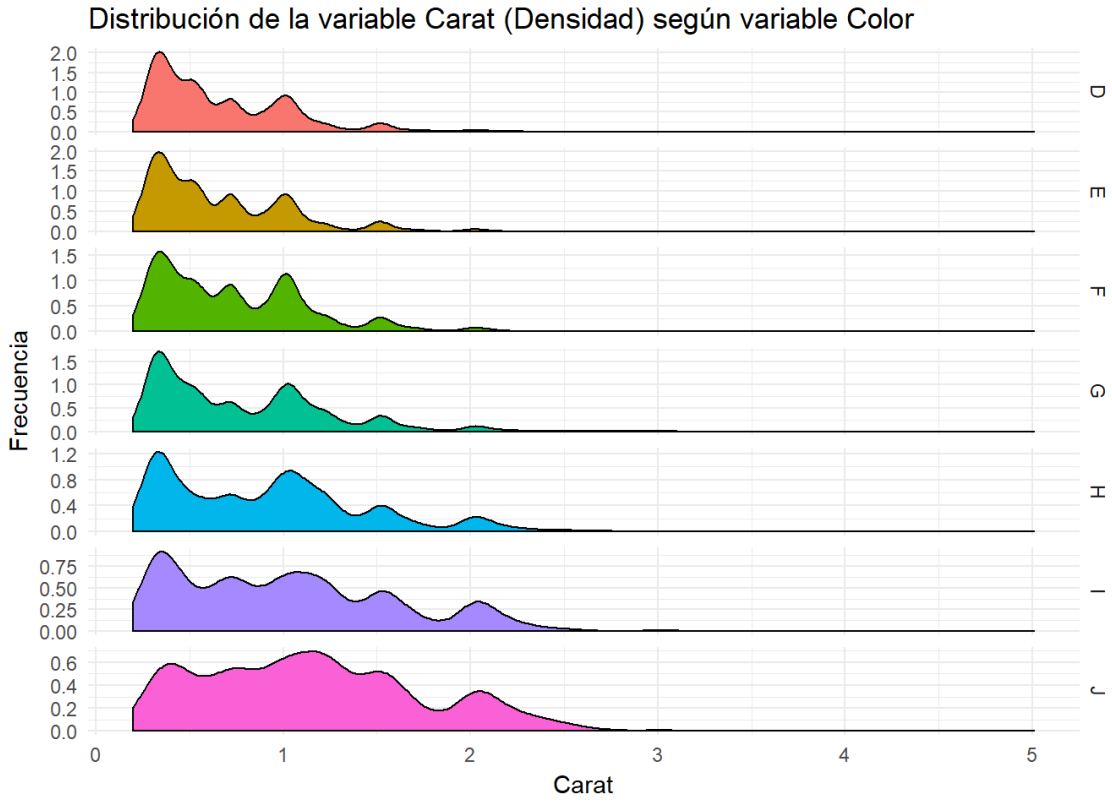

Distribución de la variable Carat (Densidad) según variable Color
Frecuencia
Carat
D
E
F
G
H
I
J

Gráfico de Dispersión (Scatterplot)

El Gráfico de Dispersión o Scatterplot es tal vez uno de los tipos de gráficos matemáticos más empleados. En él, se representan en un eje cartersiano las relaciones entre dos variables generalmente continuas, en la forma de puntos cuyas posiciones en los ejes horizontal y vertical están asociados al valor de cada variable.

Para este ejemplo, usaremos el conjunto de datos Movie Ratings disponible en el siguiente link, el cual contiene información de 562 películas con sus ratings de Rotten Tomatoes, género, presupuesto, entre otras variables.

Hide

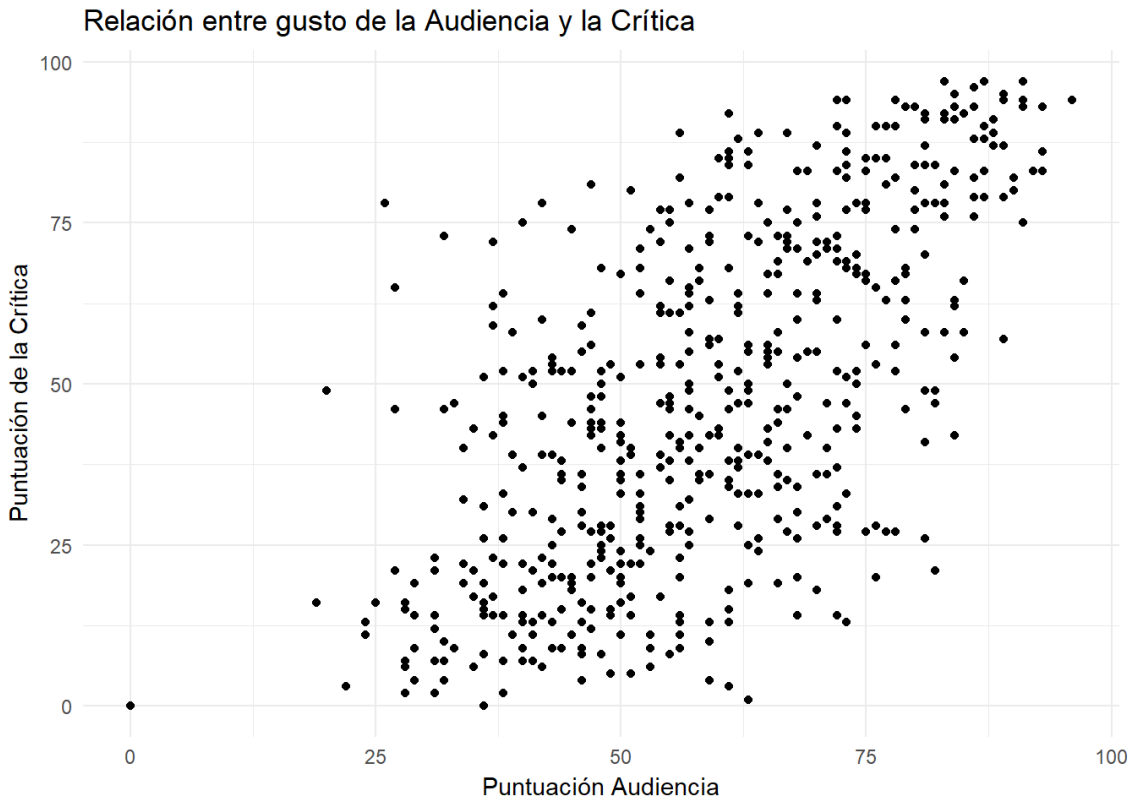
```
head(movies)
```

```
##                     Film     Genre Rotten.Tomatoes.Ratings..
## 1 (500) Days of Summer     Comedy                          87
## 2           10,000 B.C. Adventure                           9
## 3            12 Rounds     Action                          30
## 4             127 Hours Adventure                          93
## 5             17 Again     Comedy                          55
## 6                 2012     Action                          39
##   Audience.Ratings.. Budget..million... Year.of.release
## 1                 81                  8            2009
## 2                 44                105            2008
## 3                 52                 20            2009
## 4                 84                 18            2010
## 5                 70                 20            2009
## 6                 63                200            2009
```

A partir del dataset seleccionado, podriamos estar interesados en conocer la correlación entre la puntuación de la audiencia y la puntuación de la crítica para todas las películas consideradas. Para ello se hace uso de la función `geom_point()` de la siguiente manera:

Hide

```
ggplot(data = movies, aes(x = Audience.Ratings.., y = Rotten.Tomatoes.
Ratings..)) +
  geom_point() +
  xlab('Puntuación Audiencia') +
  ylab('Puntuación de la Crítica') +
  ggtitle('Relación entre gusto de la Audiencia y la Crítica') +
  theme_minimal()
```

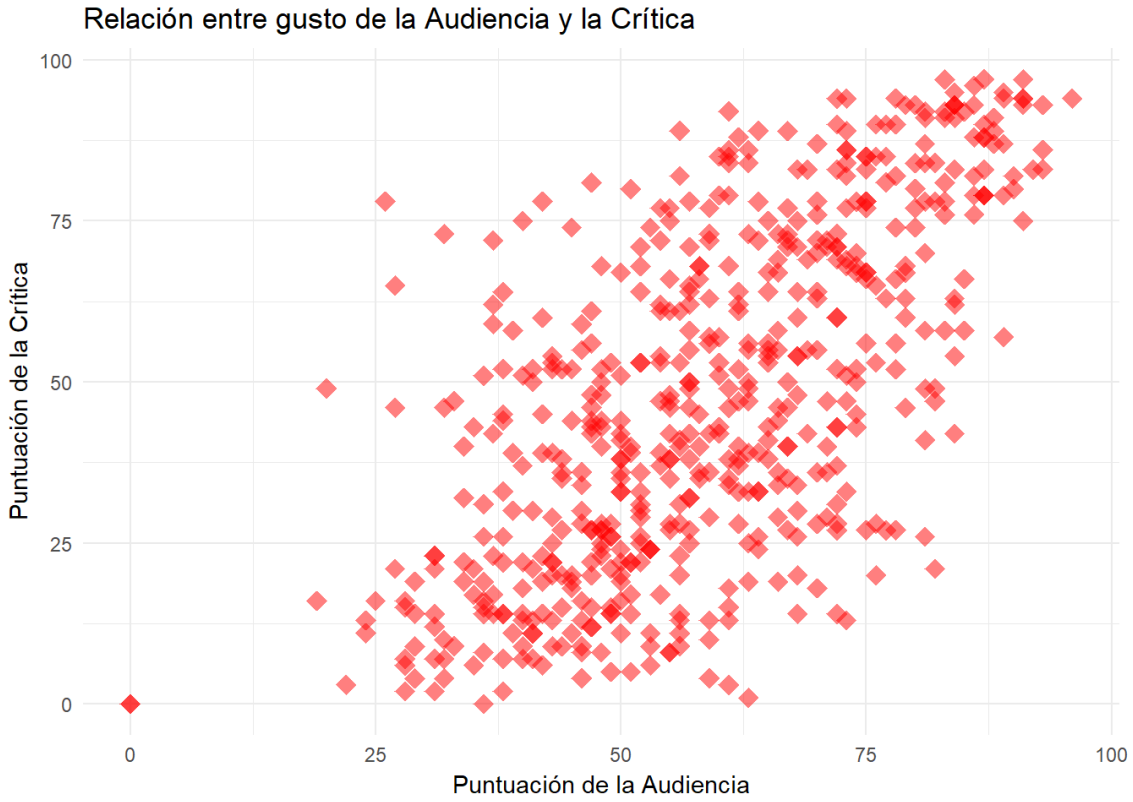


Como se observa en la gráfica obtenida, los puntos describen cierta correlación entre el gusto de la audiencia y la crítica, teniéndose que a mayor puntaje de la audiencia, pareciera que también es mayor la puntuación de la crítica. El gráfico de dispersión nos permite entonces observar de una manera directa este tipo de relaciones entre datos.

Ahora bien, al igual que con los ejemplos de gráficos anteriores, también es posible modificar las características de color, tamaño, símbolo y transparencia con los que se representan los datos. Por ejemplo:

Hide

```
ggplot(data = movies, aes(x = Audience.Ratings.., y = Rotten.Tomatoes.
Ratings..)) +

  geom_point(color = 'red', fill = 'red', size = 4, shape = 18, alpha
= 0.5) +

  xlab('Puntuación de la Audiencia') +

  ylab('Puntuación de la Crítica') +

  ggtitle('Relación entre gusto de la Audiencia y la Crítica') +

  theme_minimal()
```

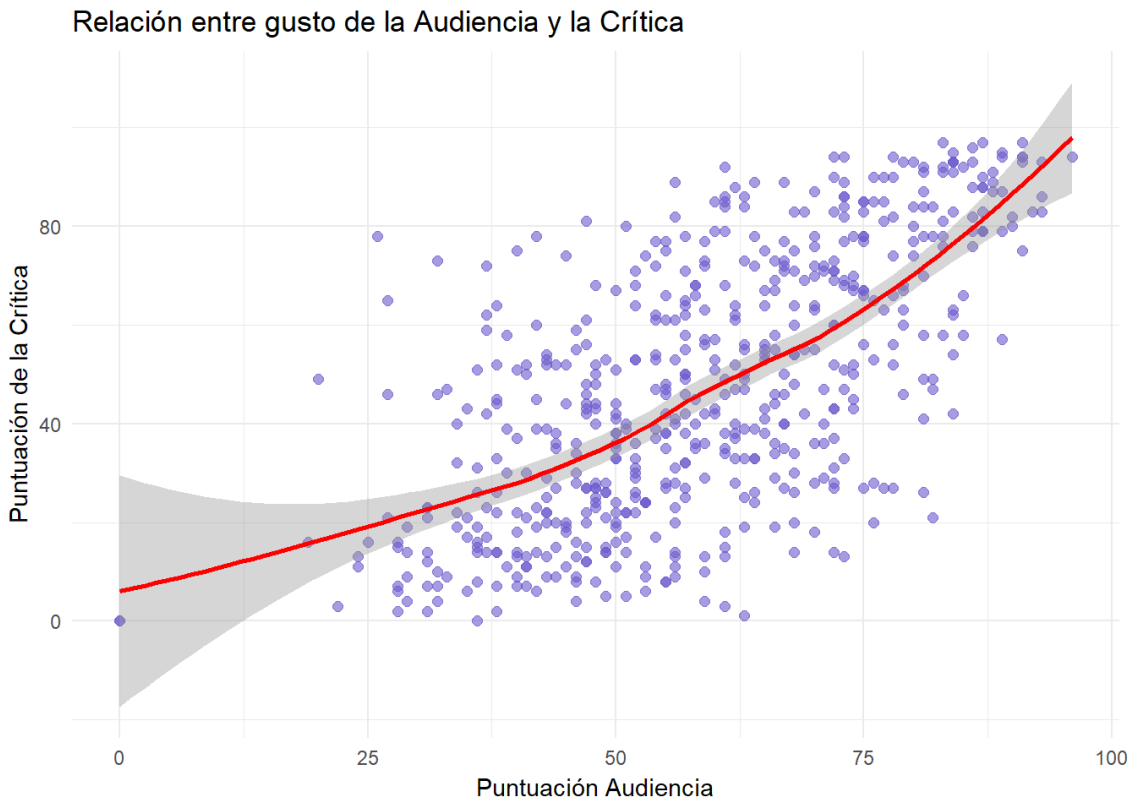


En este caso, el argumento `shape` hace referencia a la forma del símbolo con el que se dibuja el dato. Una lista con los valores que corresponden a cada forma está disponible en el siguiente link.

Cuando se trabaja con gráficos de dispersión, muchas veces es conveniente dibujar una línea que muestre la 'tendencia' inherente a los mismos. Para ello, puede emplearse la función `geom_smooth()`, a fin de incorporar dicha tendencia a la gráfica. Veamos:

Hide

```
ggplot(data = movies, aes(x = Audience.Ratings.., y = Rotten.Tomatoes.
Ratings..)) +
  geom_point(color = 'slateblue', size = 2, alpha = 0.6) +
  geom_smooth(color = 'red') +
  xlab('Puntuación Audiencia') +
  ylab('Puntuación de la Crítica') +
  ggtitle('Relación entre gusto de la Audiencia y la Crítica') +
  theme_minimal()
```

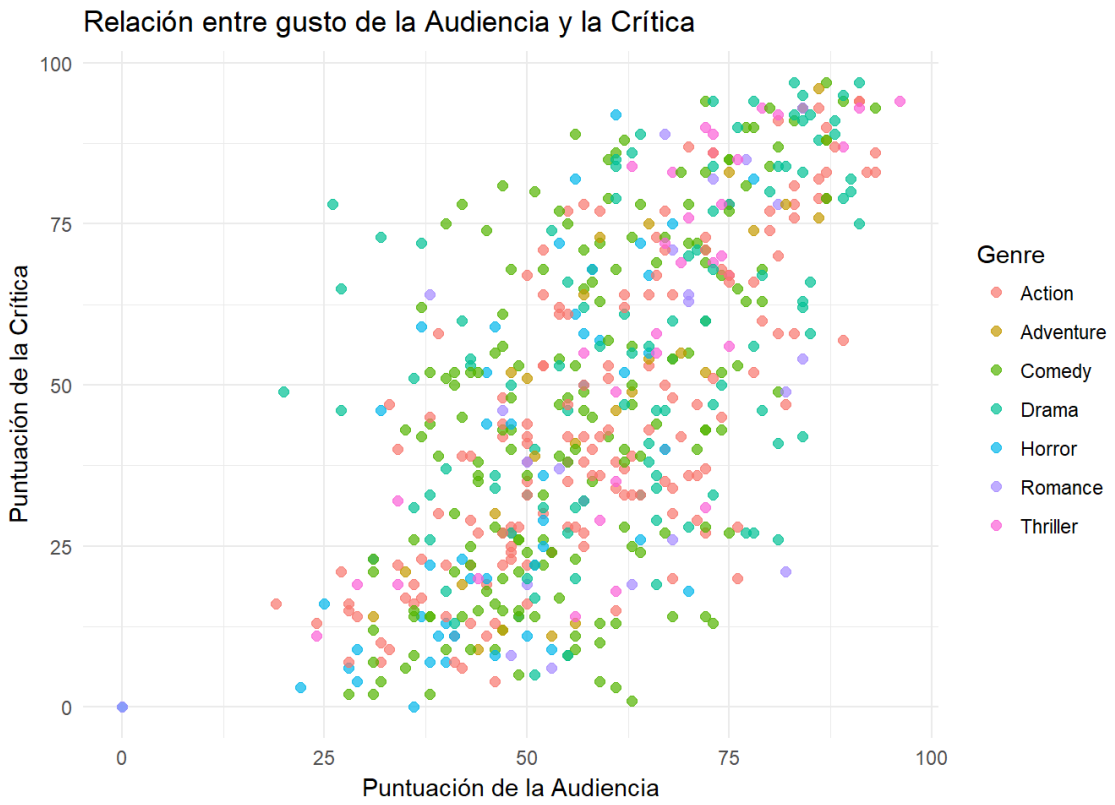


La tendencia descrita se muestra en la línea de color rojo, mientras que las bandas grises reflejan los intervalos de confianza del ajuste.

Ya que el conjunto de datos usado incorpora información, por ejemplo, del género `Genre` de las películas consideradas, podemos incorporar dicha variabla a la estética, a fin de obtener aún más información del gráfico:

Hide

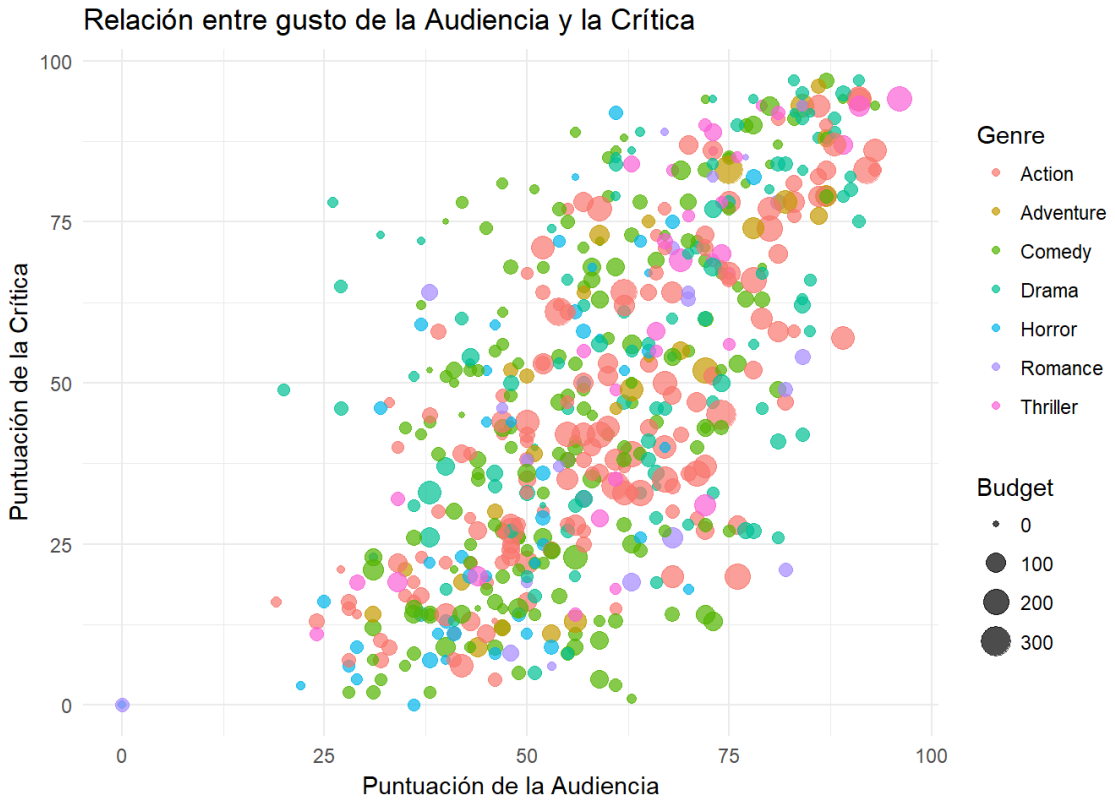
```
ggplot(data = movies, aes(x = Audience.Ratings.., y = Rotten.Tomatoes.
Ratings..)) +
  geom_point(aes(color = Genre), size = 2, alpha = 0.7) +
  xlab('Puntuación de la Audiencia') +
  ylab('Puntuación de la Crítica') +
  ggtitle('Relación entre gusto de la Audiencia y la Crítica') +
  theme_minimal()
```



Como contamos con información del 'Presupuesto' de cada película, podríamos incorporar tal variable como estética y asociarla, por ejemplo, al tamaño del símbolo empleado para representar cada punto, de modo que las películas com mayor presupuesto se representen con un círculo más grande, y viceversa (en este caso, guardamos la gráfica en la variable plot a fin de modificar el nombre de los títulos de la leyenda por separado):

Hide

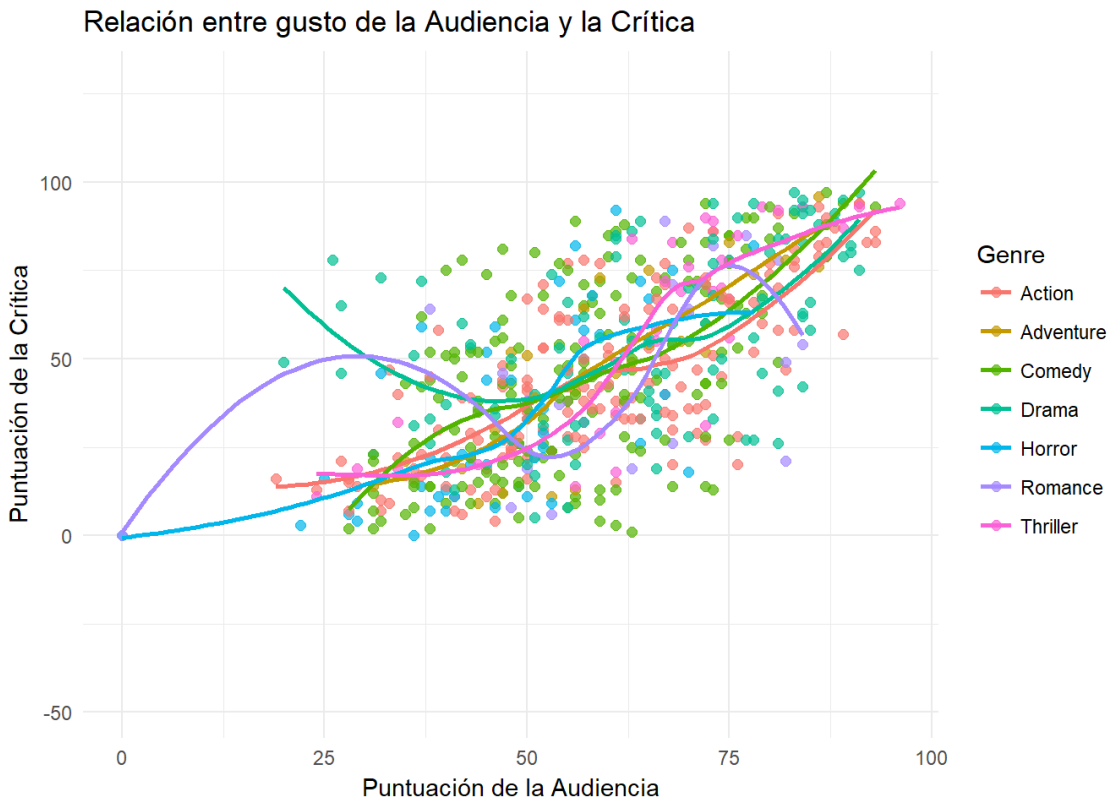
```
plot <- ggplot(data = movies, aes(x = Audience.Ratings.., y = Rotten.T
omatoes.Ratings..)) +

          geom_point(aes(color = Genre, size = Budget..million...), al
pha = 0.7) +

          xlab('Puntuación de la Audiencia') +

          ylab('Puntuación de la Crítica') +

          ggtitle('Relación entre gusto de la Audiencia y la Crítica')
+

          theme_minimal()

plot$labels$colour = "Genre"

plot$labels$size = "Budget"

plot
```



O también es posible incluir la tendencia según el género de película:

Hide

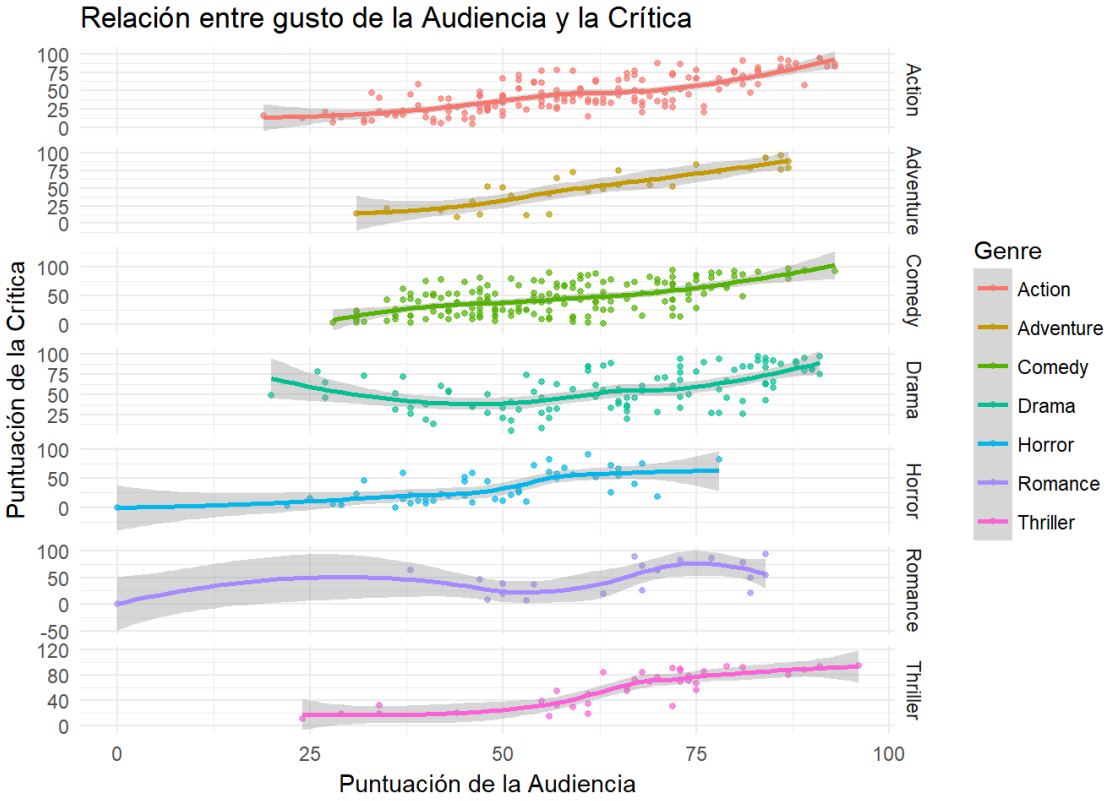
```
ggplot(data = movies, aes(x = Audience.Ratings.., y = Rotten.Tomatoes.
Ratings..)) +
  geom_point(aes(color = Genre), size = 2, alpha = 0.7) +
  geom_smooth(aes(color = Genre), fill = NA) +
  xlab('Puntuación de la Audiencia') +
  ylab('Puntuación de la Crítica') +
  ggtitle('Relación entre gusto de la Audiencia y la Crítica') +
  theme_minimal()
```



Ya que el gráfico cuenta ahora con muchos elementos, podemos hacer uso de la función `facet_grid()` para separar graficos según variables de interés. Por ejemplo, sería interesante observar las puntuaciones según género, cada uno por separado:

Hide

```
ggplot(data = movies, aes(x = Audience.Ratings.., y = Rotten.Tomatoes.
Ratings..)) +
  geom_point(aes(color = Genre), size = 1, alpha = 0.7) +
  geom_smooth(aes(color = Genre)) +
  facet_grid(Genre~., scales = 'free') +
  xlab('Puntuación de la Audiencia') +
  ylab('Puntuación de la Crítica') +
  ggtitle('Relación entre gusto de la Audiencia y la Crítica') +
  theme_minimal()
```



E incluso, ya que tenemos una variable que hace referencia al año en el que fue estrenada cada película, podemos incorporar esto a la gráfica:

Hide

```
ggplot(data = movies, aes(x = Audience.Ratings.., y = Rotten.Tomatoes.
Ratings..)) +
  geom_point(aes(color = Genre), size = 1, alpha = 0.7) +
  geom_smooth(aes(color = Genre)) +
  facet_grid(Genre~Year.of.release, scales = 'free') +
  xlab('Puntuación de la Audiencia') +
  ylab('Puntuación de la Crítica') +
  ggtitle('Relación entre gusto de la Audiencia y la Crítica') +
  theme_minimal()
```

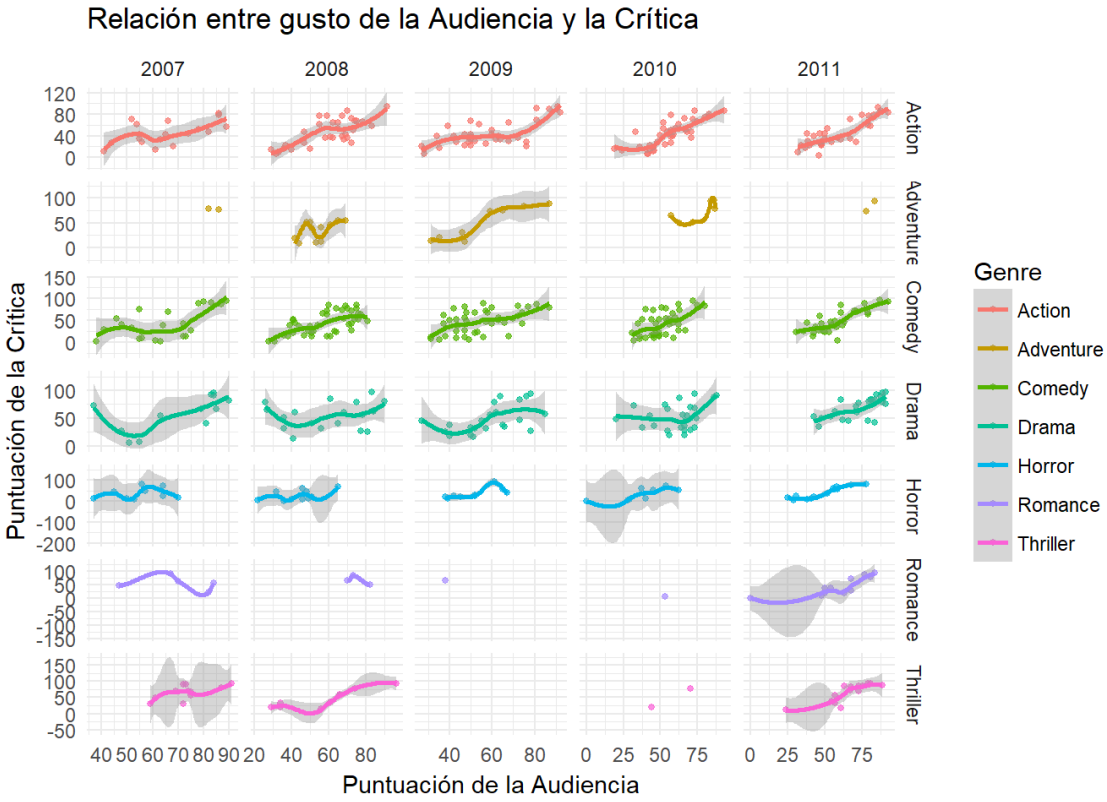

Relación entre gusto de la Audiencia y la Crítica
2007
2008
2009
2010
2011
Action
Adventure
Comedy
Drama
Horror
Romance
Thriller
Puntuación de la Crítica
Puntuación de la Audiencia
Genre
Action
Adventure
Comedy
Drama
Horror
Romance
Thriller

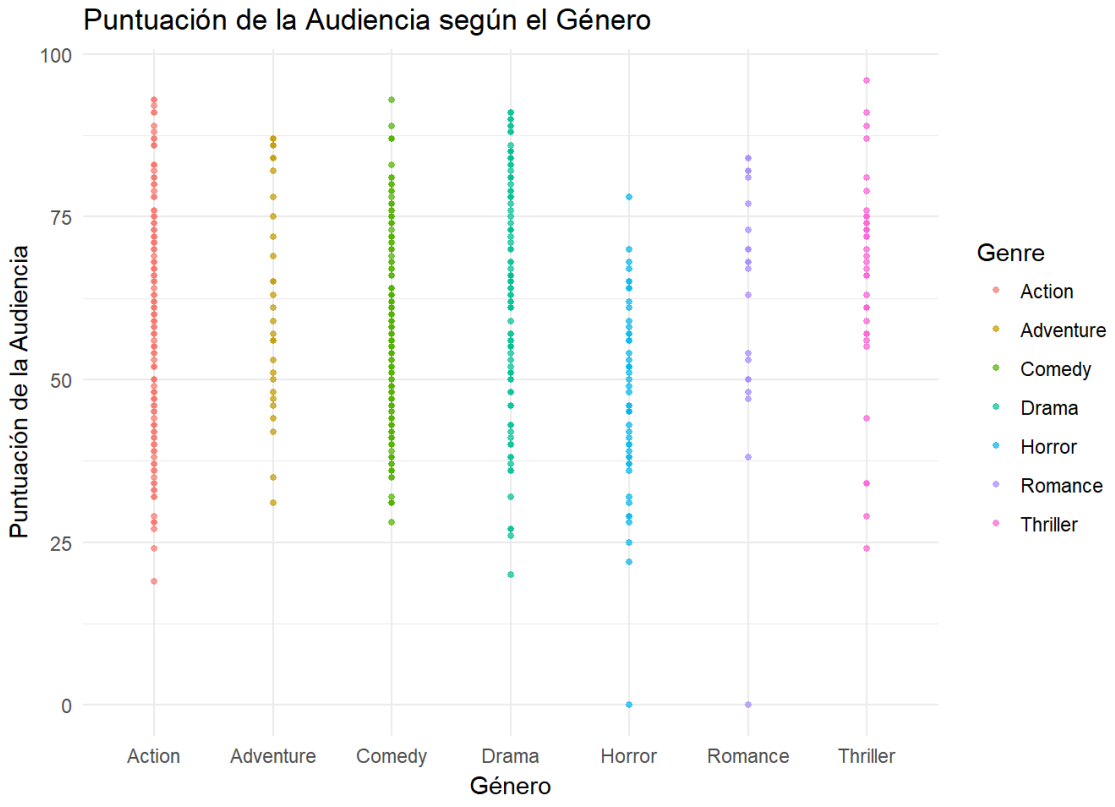
Gráfico de Cajas (Boxplot) y Gráfico de Violín (Violin plot)

El gráfico de cajas es, quizá, uno de los tipos de visualizaciones más complicados de entender, por lo que vamos a elaborar en el ejemplo a fin de presentar la idea de una manera más clara.

Partiendo del mismo conjunto de datos de las películas y sus puntuaciones, supongamos que queremos hacer una gráfica que muestre la distribución de puntuaciones de la audiencia según el género de la película. Para ello, podríamos realizar una gráfica como la siguiente:

Hide

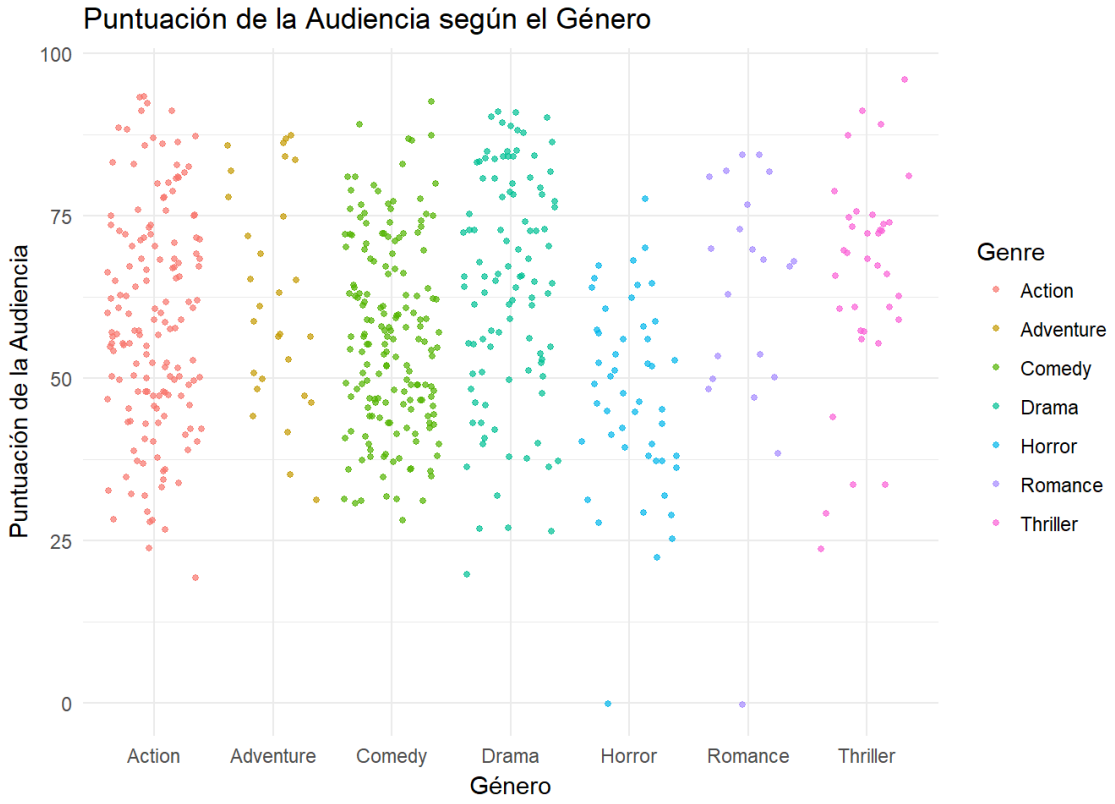
```
ggplot(data = movies, aes(x = Genre, y = Audience.Ratings..)) +
  geom_point(aes(color = Genre), size = 1, alpha = 0.7) +
  xlab('Género') +
  ylab('Puntuación de la Audiencia') +
  ggtitle('Puntuación de la Audiencia según el Género') +
  theme_minimal()
```



En la gráfica mostrada, la distribución de las puntuaciones otorgadas por la audiencia se está graficando en el eje vertical, pero ya que se trata puntos representados en una sola dimensión, las formas de dichas distribuciones no resulta clara. Una opción para intentar visualizar mejor el significado de la gráfica es representar, con la función `geom_jitter()`, una dispersión horizontal y al azar de los datos:

Hide

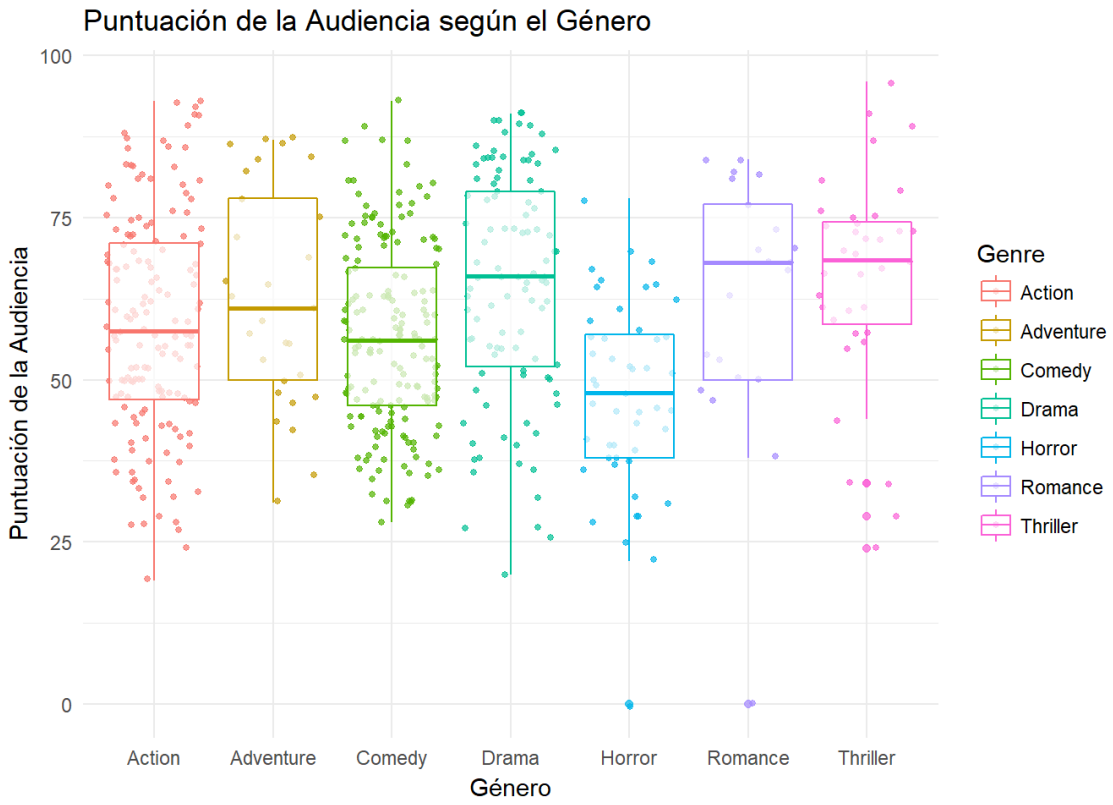
```
ggplot(data = movies, aes(x = Genre, y = Audience.Ratings..)) +
  geom_jitter(aes(color = Genre), size = 1, alpha = 0.7) +
  xlab('Género') +
  ylab('Puntuación de la Audiencia') +
  ggtitle('Puntuación de la Audiencia según el Género') +
  theme_minimal()
```



Sin embargo, esto todavía no nos permite visualizar con claridad la distribución de las puntuaciones para cada género. Incorporemos entonces gráficos de caja:

Hide

```
ggplot(data = movies, aes(x = Genre, y = Audience.Ratings..)) +
  geom_jitter(aes(color = Genre), size = 1, alpha = 0.7) +
  geom_boxplot(aes(color = Genre), alpha = 0.7) +
  xlab('Género') +
  ylab('Puntuación de la Audiencia') +
  ggtitle('Puntuación de la Audiencia según el Género') +
  theme_minimal()
```



Ahora que las cajas han sido dibujadas en el gráfico, podemos entender la forma de las distribuciones de puntuaciones para cada género ya que: para cada caja, la línea gruesa central representa la 'mediana' de los datos, la línea horizontal de la primera caja inferior representa el 1er cuartil, mientras que la línea horizontal de la caja superior representa el 3er cuartil. A las líneas verticales que se extienden de las cajas se les llama 'bigotes', y estas alcanzan los valores mínimos y máximos, mientras que los puntos que se encuentren fuera de estos rangos se les denomina 'outliers'. Una explicación a detalle puede ser encontrada en el siguiente link

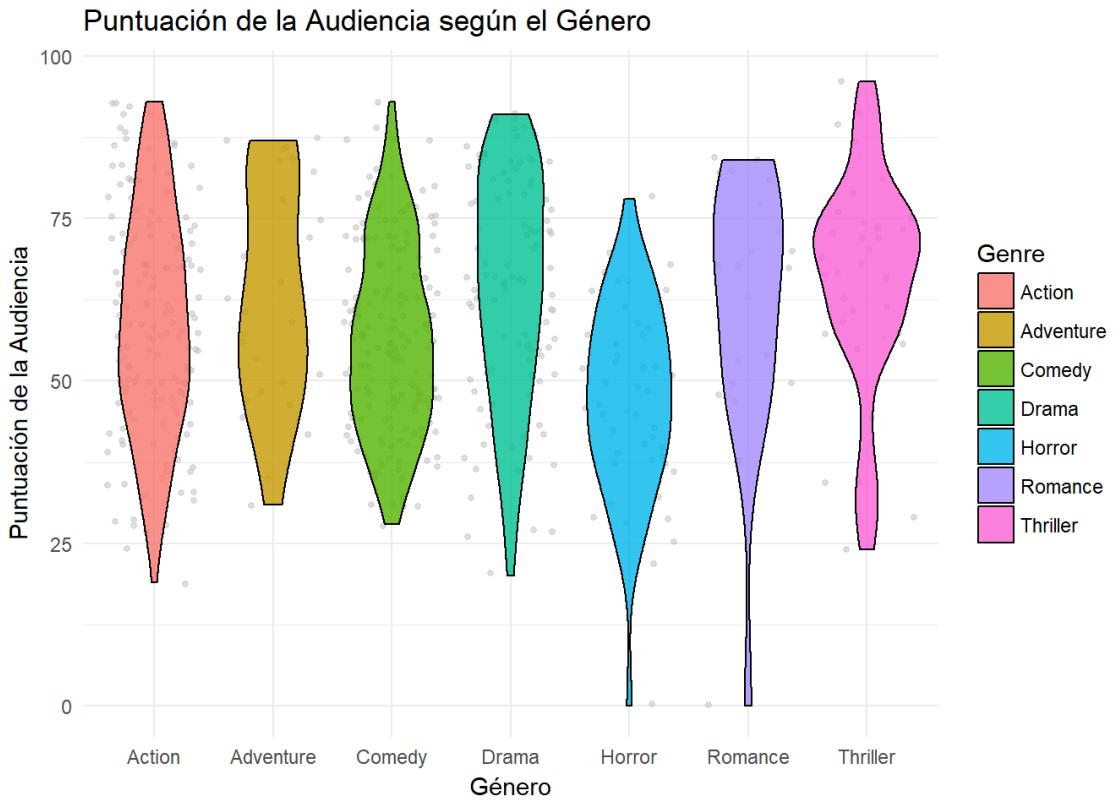
Ya que los diagramas de cajas nos permiten ubicar tanto la mediana como los cuartiles de los datos, podemos entonces entender el cómo están distribuidos los mismos en relación, por ejemplo, a otros datos. En la gráfica anterior puede observarse que, entre todos los géneros de películas puntuados, los géneros de Romance y Thriller tiene las puntuaciones

medias más altas, aunque sus distribuciones no son simétricas, mientras que géneros como Comedy y Action tienen puntuaciones medias más bajas y distribuciones más simétricas y variables a lo largo del conjunto de datos.

Como alternativa a los gráficos de caja, podemos emplear gráficos de violín para observar, del mismo modo, la distribución de los datos subyacentes:

Hide

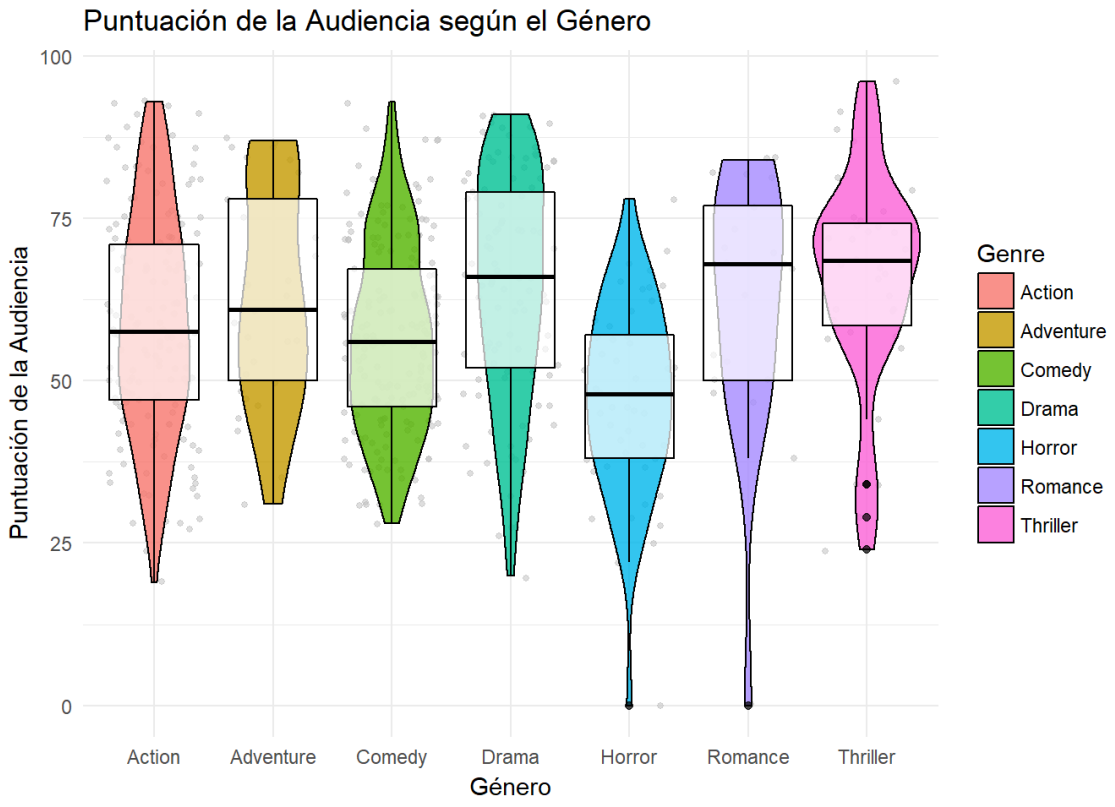
```
ggplot(data = movies, aes(x = Genre, y = Audience.Ratings..)) +
  geom_jitter(size = 1, color = 'gray', alpha = 0.5) +
  geom_violin(aes(fill = Genre), color = 'black', alpha = 0.8) +
  xlab('Género') +
  ylab('Puntuación de la Audiencia') +
  ggtitle('Puntuación de la Audiencia según el Género') +
  theme_minimal()
```



En el gráfico de violín se representa, en el eje vertical, la distribución de los datos como si se tratara de un gráfico de densidad, y se traza la imagen especular en el eje a fin de obtener el resultado en forma de 'violín' como se observa en la gráfica. Si sobre este gráfico superponemos el boxplot:

Hide

```
ggplot(data = movies, aes(x = Genre, y = Audience.Ratings..)) +
  geom_jitter(size = 1, color = 'gray', alpha = 0.5) +
  geom_violin(aes(fill = Genre), color = 'black', alpha = 0.8) +
  geom_boxplot(color = 'black', alpha = 0.7) +
  xlab('Género') +
  ylab('Puntuación de la Audiencia') +
  ggtitle('Puntuación de la Audiencia según el Género') +
  theme_minimal()
```



Podemos entender mejor la relación entre la distribución de los datos y el gráfico de cajas asociados a estos.

Correlogramas (Correlogram)

Cuando se trabaja con conjuntos de datos en donde todas las variables son numéricas y no categóricas, en muchos casos resulta interesante tener una idea de cuán correlacionadas están estas variables entre sí, es decir, cuán dependiente es la variación de una variable respecto a otra. A efectos de análisis e incluso construcción de modelos de predicción, el conocer estas correlaciones nos puede ayudar a seleccionar las variables predictoras adecuadas o bien escoger los modelos correctos según el caso. Es por ello que los 'correlogramas' pueden ser gráficas muy útiles.

Para hacer correlogramas con `ggplot2` es necesario instalar y usar la librería `ggcorrplot`:

Hide

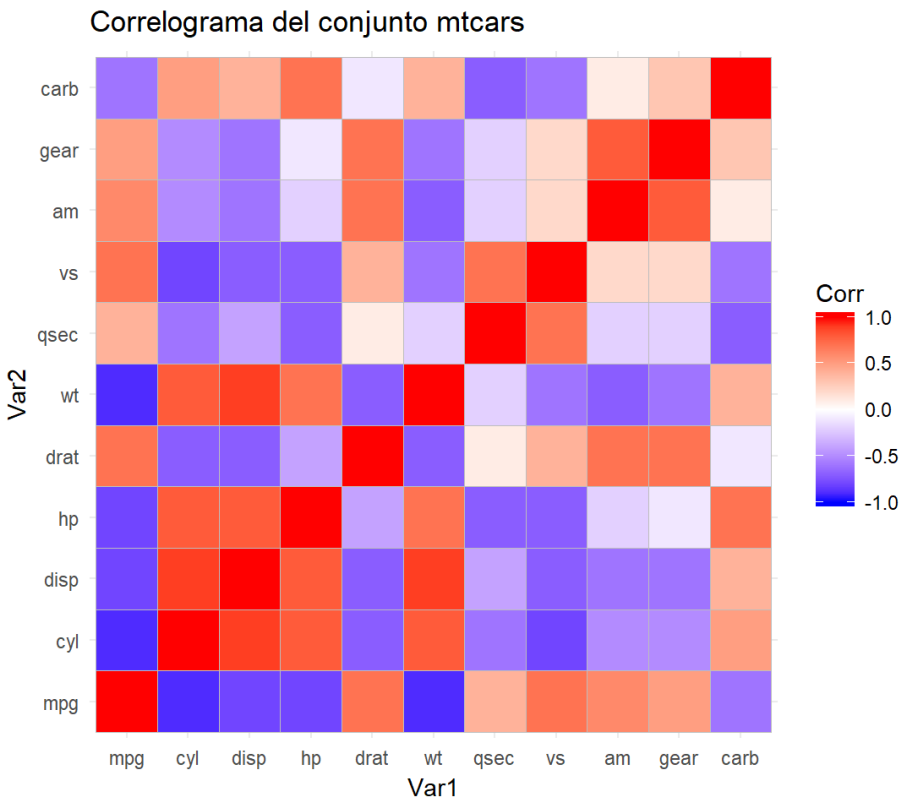
```
library(ggcorrplot)
```

Para el conjunto de datos, que a efectos del ejemplo será el `mtcars` ya trabajado, antes de dibujar el correlograma hay que calcular la matriz de correlación, de modo que:

Hide

```
corr <- round(cor(mtcars), 1)
corr
```

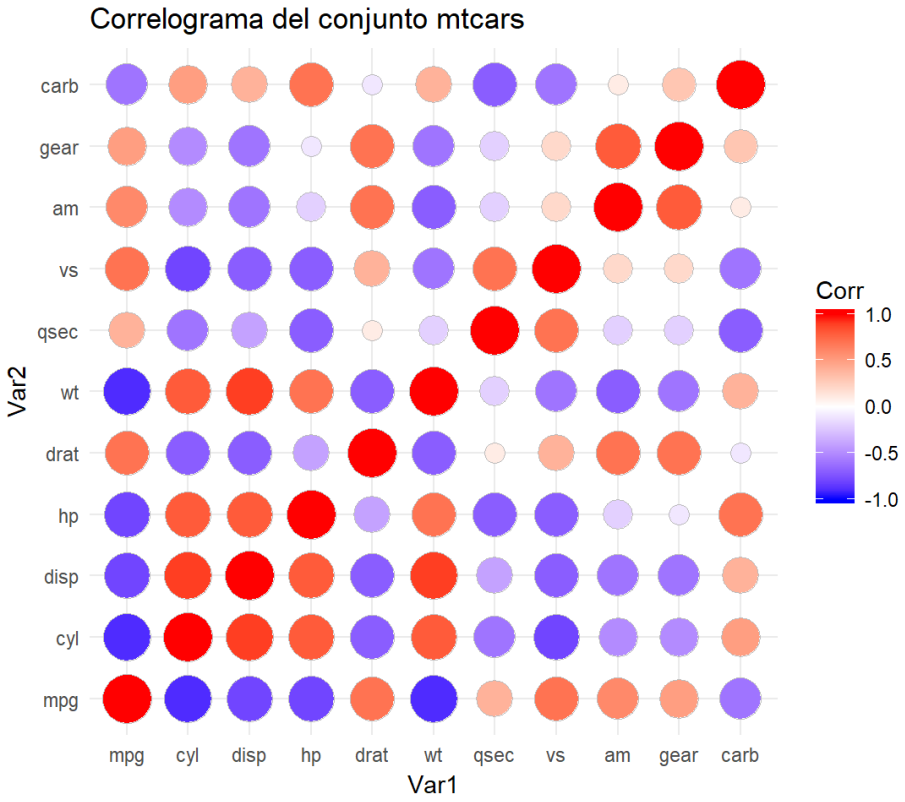
```
##        mpg  cyl disp   hp drat   wt qsec   vs   am gear carb
## mpg   1.0 -0.9 -0.8 -0.8  0.7 -0.9  0.4  0.7  0.6  0.5 -0.6
## cyl  -0.9  1.0  0.9  0.8 -0.7  0.8 -0.6 -0.8 -0.5 -0.5  0.5
## disp -0.8  0.9  1.0  0.8 -0.7  0.9 -0.4 -0.7 -0.6 -0.6  0.4
## hp   -0.8  0.8  0.8  1.0 -0.4  0.7 -0.7 -0.7 -0.2 -0.1  0.7
## drat  0.7 -0.7 -0.7 -0.4  1.0 -0.7  0.1  0.4  0.7  0.7 -0.1
## wt   -0.9  0.8  0.9  0.7 -0.7  1.0 -0.2 -0.6 -0.7 -0.6  0.4
## qsec  0.4 -0.6 -0.4 -0.7  0.1 -0.2  1.0  0.7 -0.2 -0.2 -0.7
## vs    0.7 -0.8 -0.7 -0.7  0.4 -0.6  0.7  1.0  0.2  0.2 -0.6
## am    0.6 -0.5 -0.6 -0.2  0.7 -0.7 -0.2  0.2  1.0  0.8  0.1
## gear  0.5 -0.5 -0.6 -0.1  0.7 -0.6 -0.2  0.2  0.8  1.0  0.3
## carb -0.6  0.5  0.4  0.7 -0.1  0.4 -0.7 -0.6  0.1  0.3  1.0
```



El coeficiente de correlación es un valor numérico definido en el rango [-1, 1]. Un coeficiente de correlación igual a -1 indica total anti-correlación entre las variables, mientras que un coeficiente de correlación igual a 1 indica correlación total entre las variables. Por supuesto, un valor igual a 0 indica que no existe correlación alguna entre las variables. A partir de la matriz de correlación podemos tener idea de dichas correlaciones, pero un correlograma nos permitirá percibir mucho mejor estas relaciones:

Hide

```
ggcorrplot(corr) +
  ggtitle("Correlograma del conjunto mtcars") +
  theme_minimal()
```

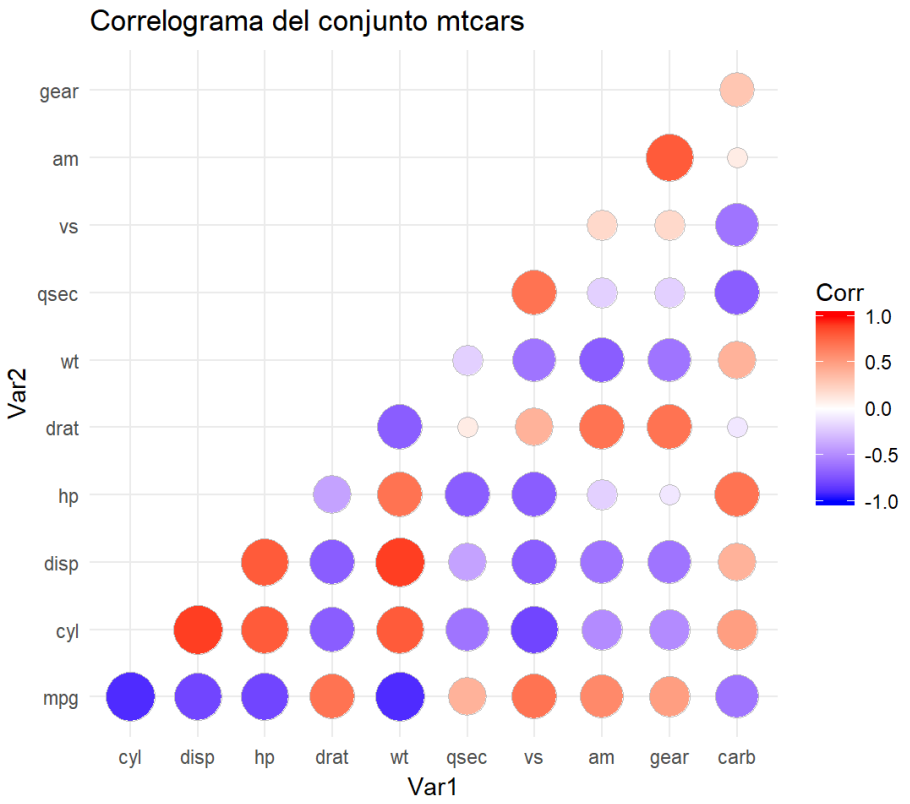


Tal y como lo describe la leyenda, los valores cercanos a 1 tendrán color rojo, mientras que los valores cercanos a -1 tendrán color azul. De este modo, podemos ver cuáles variables están más o menos correlacionadas entre ellas. Por ejemplo, gracias al correlograma se observa que las variables `cyl` y `disp`, así como `cyl` y `hp` están altamente correlacionadas (lo que es coherente pues el número de cilindros de un vehículo afecta de manera proporcional tanto los caballos de fuerza como el desplazamiento (volúmen) del motor). Por otro lado, se observa que variables como `mpg` y `cyl` o `hp` están anti-correlacionadas (es decir, mientras mayor cantidad de cilindros o caballos de fuerza, menos cantidad de millas por galón de eficiencia).

Dentro de las opciones de `corrplot`, podemos hacer el gráfico usando círculos en lugar de cuadrados, de modo que, además, el tamaño de los círculos no da información del valor de la correlación:

Hide

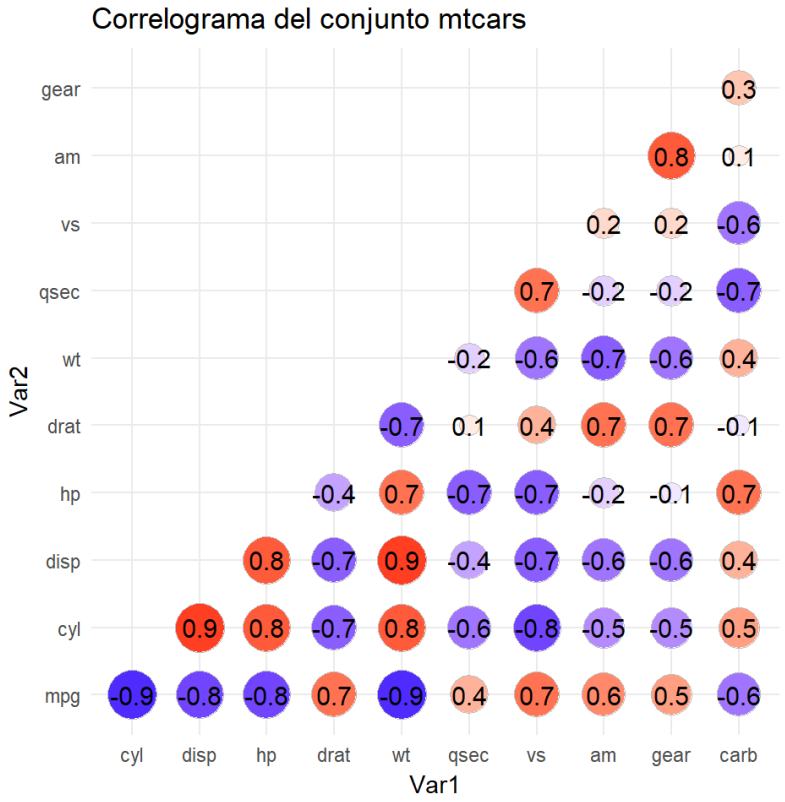
```
ggcorrplot(corr, method = 'circle') +
  ggtitle("Correlograma del conjunto mtcars") +
  theme_minimal()
```



Ya que la matriz de correlación es simétrica, basta con graficar solamente una porción de la misma:

Hide

```
ggcorrplot(corr, method = 'circle', type = 'lower') +
  ggtitle("Correlograma del conjunto mtcars") +
  theme_minimal()
```



Y si deseamos visualizar los valores específicos de correlación para cada caso, podemos incluirlos en el correlograma y quitar la leyenda:

Hide

```
ggcorrplot(corr, method = 'circle', type = 'lower', lab = TRUE) +
  ggtitle("Correlograma del conjunto mtcars") +
  theme_minimal() +
  theme(legend.position="none")
```