FACULTAD DE MATEMÁTICAS
PONTIFICIA UNIVERSIDAD
CATÓLICA DE CHILE

EYP1113 - Probabilidad y Estadística

Laboratorio 08

Pilar Tello Hernández
pitello@uc.cl

Facultad de Matemáticas
Departamento de Estadística
Pontificia Universidad Católica de Chile

Segundo Semestre 2021

tidyr

El paquete `tidyr` sirve para poder manipular y reordenar bases de datos en `R`. Esta librería cumple otras funciones distintas a las vistas con `dplyr`. Vamos a conocer funciones útiles tales como: `gather`, `spread`, `separate` y `unite`.

tidyr

gather

La función gather nos permite cambiar el formato de una tabla o base de datos "ancha" a "larga". Permite agrupar varias columnas en una. Primero debemos entregarle la base de datos como argumento, luego el nombre de la nueva columna que contendrá los nombres de las columnas que estamos agrupando, el nombre de la nueva columna que contendrá los datos y finalmente las columnas a agrupar.

Para ejemplificar armaremos una nueva base de datos para aplicar esta función:

```
df <- data.frame(ID=1:10,
Tipo1=rep(c("A","B"),by=5),
Tipo2=rep(c("C","D"),each=5),
var1=rnorm(10), var2=rnorm(10), var3=rnorm(10))
df
df2 <- gather(data=df, key="Col",value="Valor",
var1:var3)
df2
```

Es importante contar con la variable ID, que es el identificador de la fila.

tidyr

spread

La función `spread` es lo contrario a `gather`, nos permite cambiar el formato de una tabla o base de datos "larga" a "ancha". Primero debemos entregarle la base de datos como argumento, luego el nombre de la columna que contiene los identificadores y luego la de los valores.

Siguiendo con la base `df2` que se construyó en el ejemplo anterior:

```
df3 <- spread(data=df2, key=Col, value=Valor)
df3
```

Con la variable `ID`, la función entiende cuáles observaciones corresponden a la misma fila.

tidyr

separate

La función `separate` sirve para separar una columna en varias que tengan el mismo delimitador. Primero debemos entregarle la base de datos como argumento, luego el nombre de la columna a separar y luego los nombres de las nuevas columnas.

Para ejemplificar armaremos una nueva base de datos para aplicar esta función:

```
df4 <- data.frame(Col1=c("1-a","2-b","3-c","4-d"))
df5 <- separate(data=df4, col=Col1, into=c("numero","letra"))
df5
```

Alternativamente, se puede entregar el separador con el argumento, por ejemplo para el guión:

```
sep="-"
```

tidyr

unite

La función `unite` es la función contraria a `separate`. Primero debemos entregarle la base de datos como argumento, luego el nombre de la nueva columna que contendrá los datos unidos y luego los nombres de las columnas a unir. Por defecto el separador que usará es el guión bajo, pero éste se puede modificar con el argumento `sep`.

Para ejemplificar armaremos una nueva base de datos para aplicar esta función:

```
df6 <- unite(data=df5, col="Col1", numero:letra, sep="-")
df6
```

ggplot2

El paquete ggplot2 es uno de los paquetes de R más populares para la visualización de datos. Permite una mayor edición de los gráficos con respecto a los que vienen por defecto en R.

ggplot2

Para poder ejemplificar se usará la base de datos `mpg` que viene integrada en `R`. Este conjunto de datos contiene un subconjunto de los datos de economía de combustible que la EPA pone a disposición en `http://fueleconomy.gov.sep`.

Contiene solo modelos que se han lanzado cada año entre 1999 y 2008; este es un proxy de la popularidad del automóvil. Sus variables son las siguientes:

- manufacturer = marca / fabricante
- model = modelo
- displ = desplazamiento del motor, en litros
- year = año de fabricación
- cyl = número de cilindros
- trans = tipo de transmisión
- drv = tracción
- cty = millas por galón en la ciudad
- hwy = millas por galón en la carretera (highway)
- fl = tipo de combustible
- class= tipo de vehículo

ggplot2

Base de gráficos

Para poder crear un gráfico con ggplot2 se debe ocupar como base la función ggplot, a ésta se le entrega la base de **datos** en el argumento data. Luego dentro de la misma función se le entrega la **estética** del gráfico en el argumento aes. Dentro del argumento aes, puede ir definido qué variables van en el eje x, en el eje y y distintos subargumentos que se le pueden entregar a éste. Finalmente se agrega la **geometría** del gráfico, es decir, el tipo de gráfico que se quiere construir. sep.

Luego, un gráfico se puede separar por **facetas**, es decir, distintos niveles según un factor. También se pueden manipular las **coordenadas** de los gráficos, tales como los límites del eje x o y. Además, se pueden añadir **estadísticos**, tales como rectas de regresión, suavizamiento de datos, etc. Finalmente se edita el tema de nuestro gráfico, tales como tamaño de fuente, cuadrículas, etc.

Pueden descargar un cheat sheet de ggplot2 en el siguiente link: https://github.com/rstudio/cheatsheets/raw/master/data-visualization-2.1.pdf.

ggplot2

geom_histogram()

Comencemos con los histogramas. Para los histogramas la variable base de **estética** que necesitamos es `x` en donde estarán los datos para los que queremos hacer un histograma. La función `geom_histogram()` es la que crea el histograma.

```
p1 <- ggplot(data=data, #datos
aes(x=hwy)) # estética del gráfico
p1 + geom_histogram() # geometría
```

Podemos editar el ancho de banda con el comando `binwidth`. Añadir colores de borde y relleno con los argumentos `col` y `fill`. Indicar la transparencia del histograma con el argumento `alpha`:

```
p2 <- p1 + geom_histogram(binwidth = 2.5, # ancho de banda
col="white", fill="steelblue", # borde y relleno
alpha=0.5) # transparencia
p2
```

Los nombres de los ejes, los podemos editar con la función `labs`. Con la función `ggtitle` podemos agregar título y subtítulo a nuestro gráfico:

```
p3 <- p2 + labs(x="Millas por galón en la carretera",
y="Frecuencia")
p3
p4 <- p3 + ggtitle("Histograma de millas por galón en la carretera",
subtitle = "Base de datos: mpgcars")
p4
```

ggplot2

geom_histogram()

Para indicar que se quiere un histograma de densidad hay que indicarlo dentro de la estética del histograma con `aes(y=..density..)`:

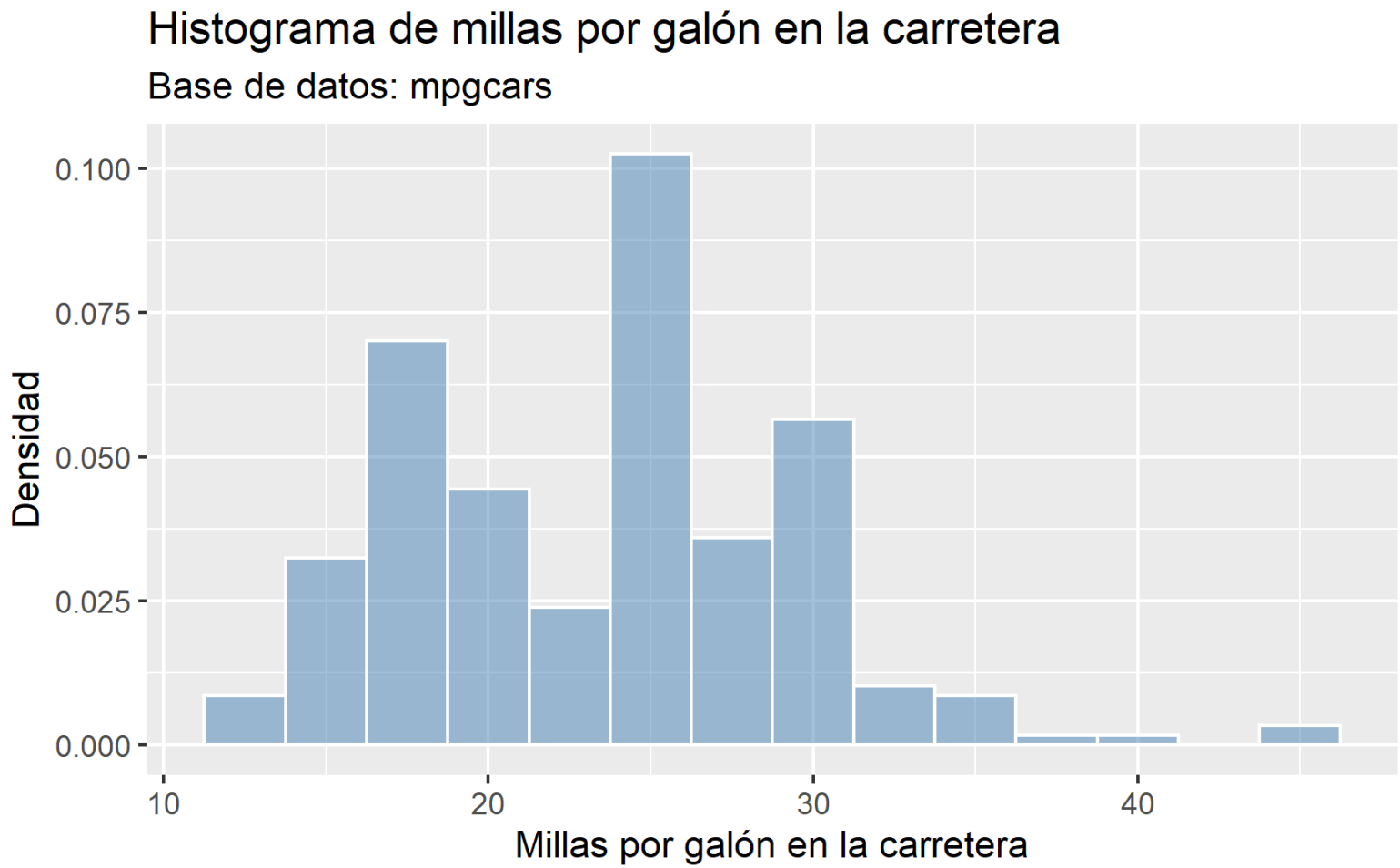
```
p5 <- p1 + geom_histogram(aes(y=..density..), # histograma de densidad
binwidth = 2.5,
col="white",
fill="steelblue",
alpha=0.5)+
labs(x="Millas por galón en la carretera",
y="Densidad")+
ggtitle("Histograma de millas por galón en la carretera",
subtitle = "Base de datos: mpgcars")
p5
```

Para guardar el último gráfico ejecutado se puede utilizar la función `ggsave`, en ella se puede editar el largo y ancho de gráfico con los argumentos `height` y `width` respectivamente:

```
ggsave("geomhistogram.png")
```

ggplot2

`geom_histogram()`

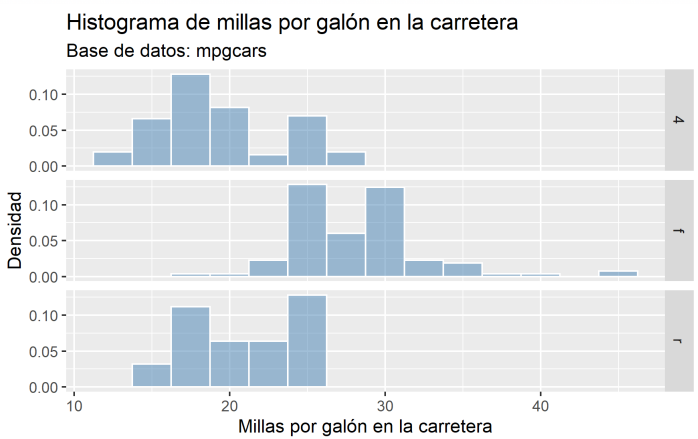


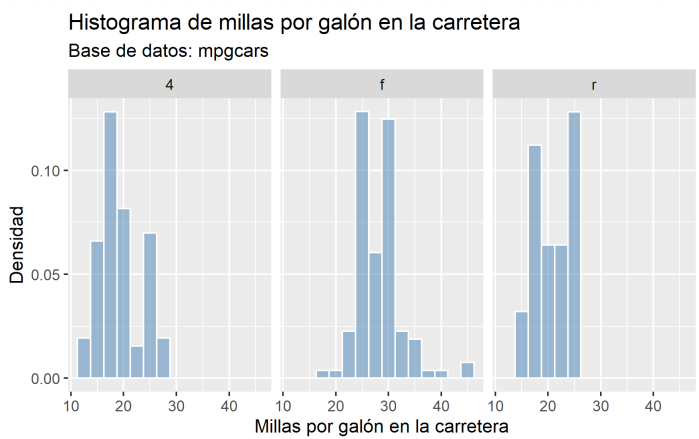
ggplot2

geom_histogram()

Para agregar **facetas**, se pueden usar las funciones `facet_grid` y `facet_wrap` para separar el gráfico por factores.

```
p5 + facet_grid(rows="drv")
ggsave("geomhistogramgrid.png")
p5 + facet_wrap(facets ="drv")
ggsave("geomhistogramwrap.png")
```



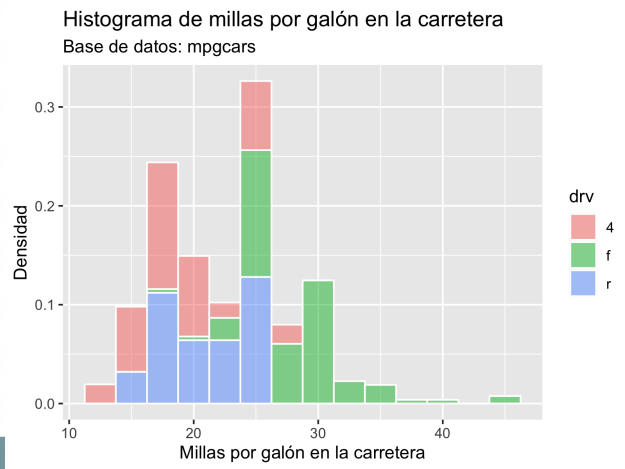


ggplot2

geom_histogram()

Si queremos en un histograma tener estos histogramas integrados, diferenciados por color o por relleno podemos incluirlo dentro de la estética base del gráfico.

```
p1 <- ggplot(data=data,
aes(x=hwy,fill=drv)) # el relleno dependerá de la tracción
p2 <- p1 + geom_histogram(aes(y=..density..),
binwidth = 2.5,
col="white",
alpha=0.5)+
labs(x="Millas por galón en la carretera",
y="Densidad")+
ggtitle("Histograma de millas por galón en la carretera",
subtitle = "Base de datos: mpgcars")
p2
```



ggplot2

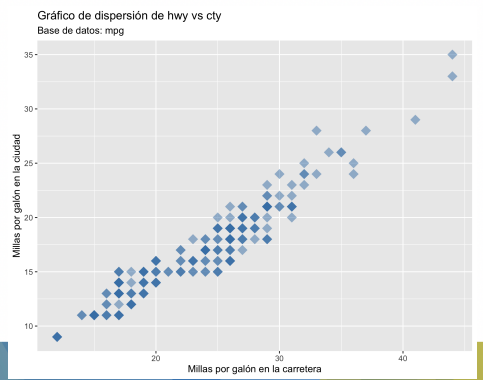
geom_point()

Ahora veremos los gráficos de dispersión o scatterplot con la función `geom_point()`. La base **estética** del gráfico, es indicar los datos de los ejes `x` e `y`.

```
p1 <- ggplot(data=data, #datos
aes(x=hwy, y=cty)) # estética del gráfico
p1 + geom_point() # geometría
```

Podemos editar el tamaño de los puntos con `size`, la transparencia con `alpha`, el color con `col` y la forma con `shape`:

```
p2 <- p1 + geom_point(size=5, alpha=0.5, col="steelblue", shape=18)+
labs(x="Millas por galón en la carretera",
y="Millas por galón en la ciudad")+
ggtitle("Gráfico de dipsersión de hwy vs cty",
subtitle="Base de datos: mpg")
p2
```

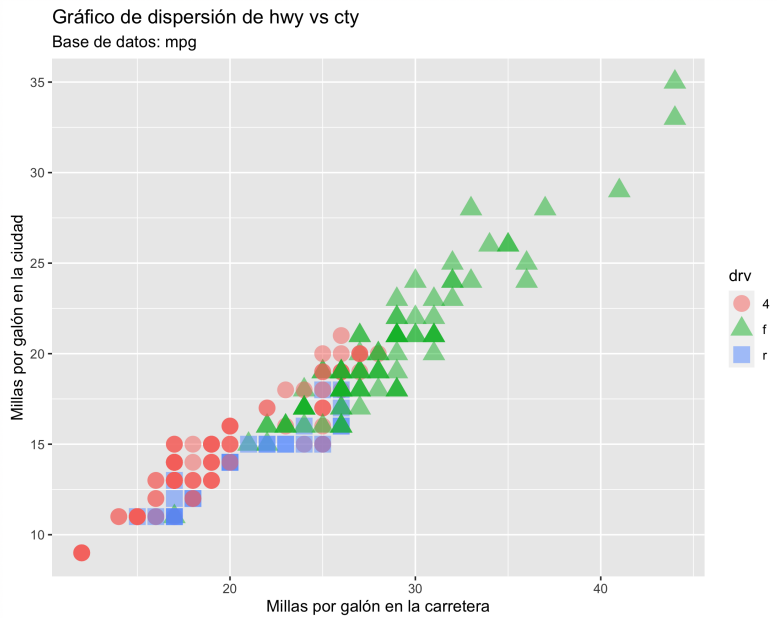


ggplot2

geom_point()

El color y la forma de los puntos pueden depender de algún factor de la base de datos, agregándolo dentro de la estética de geom_point():

```
p3 <- p1 + geom_point(aes(col=drv, shape=drv), size=5, alpha=0.5)+
labs(x="Millas por galón en la carretera",
y="Millas por galón en la ciudad")+
ggtitle("Gráfico de dipsersión de hwy vs cty",
subtitle="Base de datos: mpg")
p3
```

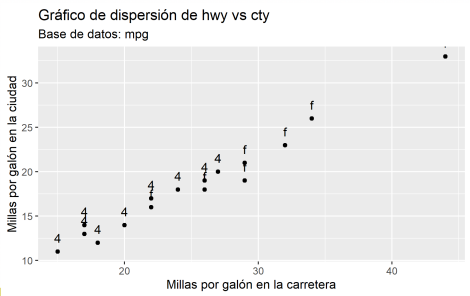


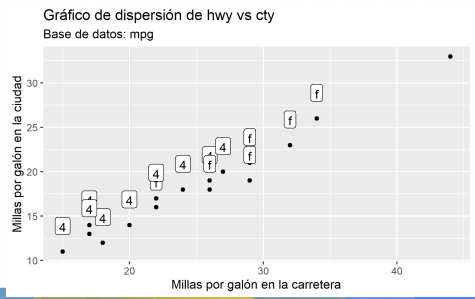
ggplot2

geom_point()

Si tuviéramos un gráfico con una cantidad razonable de puntos para agregarle texto o etiquetas a éstos se pueden usar las funciones geom_text y geom_label()

```
library(dplyr)
data2 <- data %>%  sample_n(size=20)
p1 <- ggplot(data=data2, #datos
aes(x=hwy, y=cty, label=drv))+ # etiqueta de tracción
labs(x="Millas por galón en la carretera",
y="Millas por galón en la ciudad")+
ggtitle("Gráfico de dispersión de hwy vs cty",
subtitle="Base de datos: mpg")
p1 + geom_point()+
geom_text(vjust=-1)
ggsave("geompointtext.png")
p1 + geom_point()+
geom_label(vjust=-1)
ggsave("geompointlabel.png")
```





ggplot2

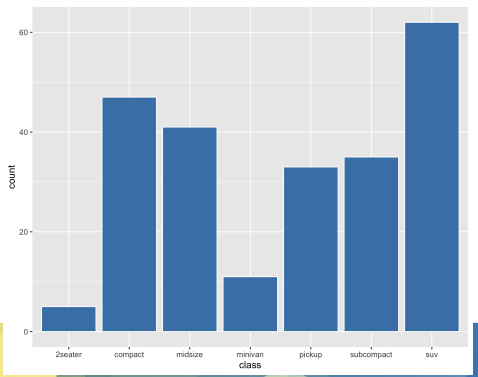
geom_bar()

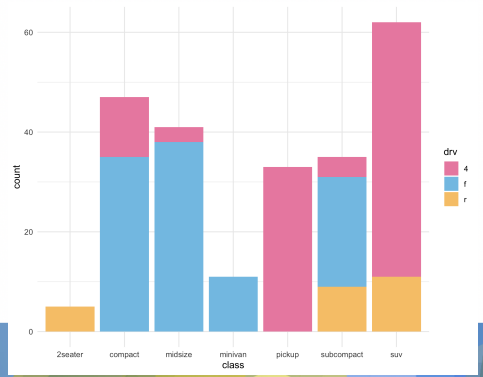
Los gráficos de barra se construyen con la función `geom_bar()`. La base **estética** del gráfico, es indicar los datos de los ejes `x`, porque a partir de ellos se realizará el conteo. Podemos modificar los colores de las barras en `geom_bar()`.

```
p1 <- ggplot(data=data, #datos
aes(x=class)) # estética del gráfico
p1 + geom_bar() # geometría
p1 + geom_bar(col="white", fill="steelblue")
```

Para crear un gráfico de barras apilado por grupos, se debe incluir en la estética de `ggplot`, hay que indicar la variable por la que dependerá el `fill`.

```
p1 <- ggplot(data=data, #datos
aes(x=class, fill=drv)) # relleno de colores
p1 + geom_bar()+
scale_fill_manual(values=paleta)+
theme_minimal()
```





ggplot2

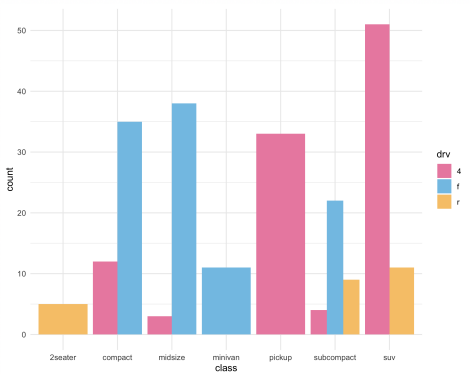
geom_bar()

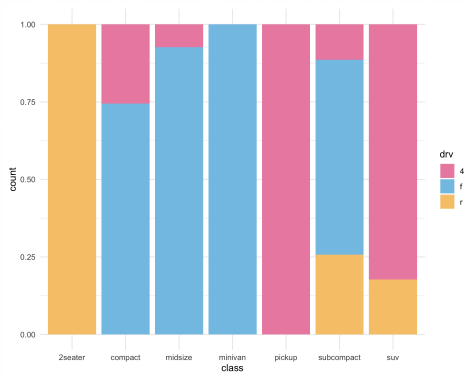
Para crear un gráfico de barras agrupado, dentro del `geom_bar()`, se debe entregar el argumento `position="dodge"`:

```
p1 + geom_bar(position="dodge")+
scale_fill_manual(values=paleta)+
theme_minimal()
```

Para crear un gráfico de barras apilado por grupos, pero con porcentajes, dentro del `geom_bar()`, se debe entregar el argumento `position="fill"`:

```
p1 + geom_bar(position="fill")+
scale_fill_manual(values=paleta)+
theme_minimal()
```





ggplot2

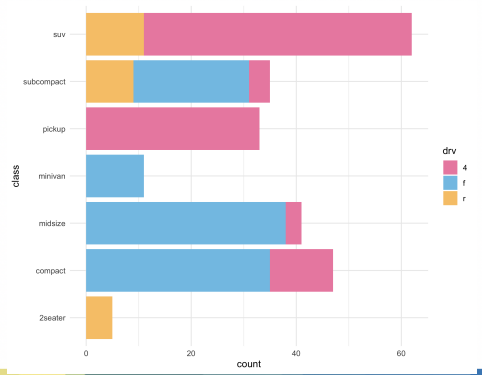
geom_bar()

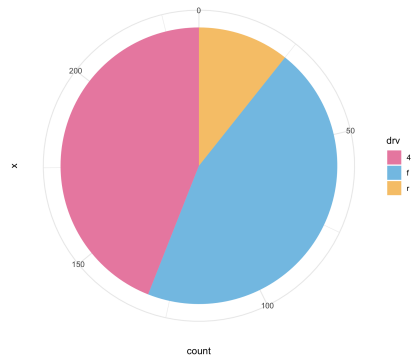
Si se quiere construir un gráfico de barras horizontal, se debe agregar la función `coord_flip()`:

```
p1 + geom_bar()+
scale_fill_manual(values=paleta)+
theme_minimal()+
coord_flip()
```

Dentro de los gráficos de barra, se pueden crear gráficos de torta. Las indicaciones que hay que seguir son las siguientes:

```
ggplot(data=data,aes(x="",fill=drv))+
geom_bar(width=1)+
scale_fill_manual(values=paleta)+
coord_polar("y",start=0)+
theme_minimal()
```





ggplot2

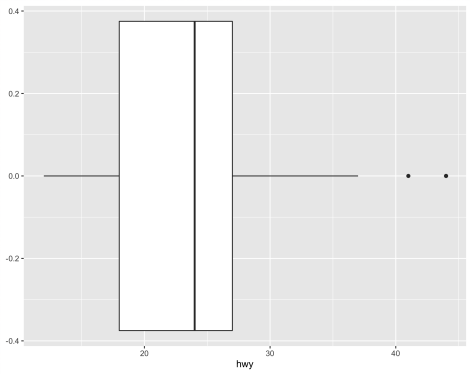
geom_boxplot()

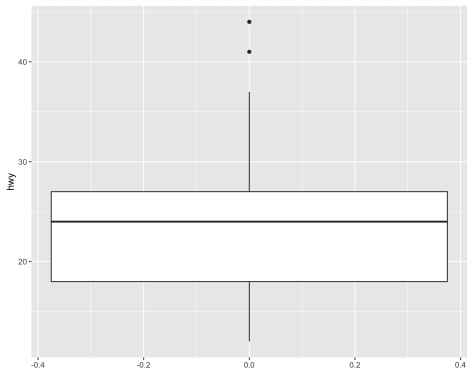
Los gráficos de boxplot se construyen con la función `geom_boxplot()`. La base **estética** del gráfico, es indicar los datos de los ejes `x`, porque a partir de ellos se realizará el boxplot. Podemos modificar los colores de las barras en `geom_boxplot()`.

```
p1 <- p1 <- ggplot(data=data, #datos
aes(x=hwy)) # estética del gráfico
p2 <- p1 + geom_boxplot() # geometría
p2
```

Si se quiere en dirección vertical se agrega la función `coord_flip()`:

```
p2 + coord_flip()
```





ggplot2

geom_boxplot()

Para separar el boxplot por grupos o factores, se debe indicar en la estética base del gráfico que el eje y corresponde a la variable que entregaremos.

```
# Por factor
p3 <- p1 <- ggplot(data=data, #datos
aes(x=hwy,y=drv)) # en y entregamos el factor
p3 + geom_boxplot()+
coord_flip()
```

Para que el color del relleno o `fill` sean según el factor hay que indicarlo en la base estética del gráfico..

```
p4 <- p3 + geom_boxplot(aes(fill=drv))+
coord_flip()+
scale_fill_manual(values=paleta)+
theme_minimal()
p4
```