Manual de gráficos en R con la librería ggplot2

Código y ejemplos en R

Danny Murillo González

ORCID: 0000-0003-0297-7213

email: danny.murillo@utp.ac.pa

CIDITIC

Universidad Tecnológica de Panamá

Yostin Añino

ORCID: 0000-0002-8870-8155

email: yostin.anino@up.ac.pa

Museo de Invertebrados

Universidad de Panamá

2024-04-08

Tabla de contenido

1Acerca del contenido

El material presentado se ha desarrollado como una guía en el uso de la librería ggplot2 en R para la creación de gráficos. El objetivo es hacer uso efectivo de la reglas, principios y recomendaciones en la visualización de datos al generar gráficos utilizando lenguaje R y el paquete ggplot2 para crear visualización de datos.

1.1 Descripción

La visualización de datos es la técnica utilizada para brindar información sobre los datos utilizando señales visuales como gráficos, tablas, mapas y muchos otros. Esto es útil ya que ayuda a comprender fácilmente y de forma intuitiva grandes cantidades de datos y, por lo tanto, a tomar mejores decisiones al respecto.

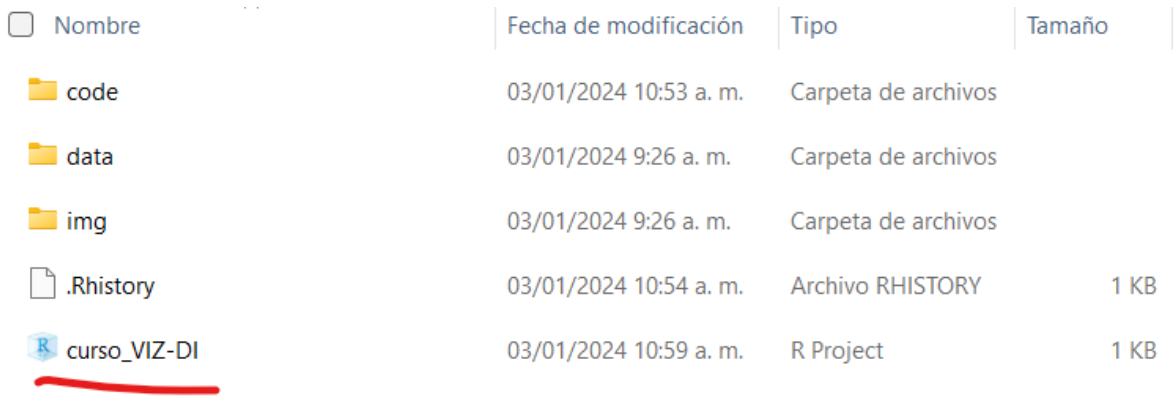
La visualización de datos tiene dos propósitos principales:

- Análisis de datos exploratorios e identificación de problemas de datos.
- Comunicar conocimientos y resultados

Entre las herramientas más populares de visualización de datos disponibles están Tableau, Plotly, R, Phyton, Google Charts, DataWraper, Flourish , entre otros. Las distintas plataformas de visualización de datos tienen diferentes capacidades, funcionalidades, sin embargo, una de las mayores diferencias de R, es que es lenguaje diseñado para análisis de datos, estadística, visualización e investigación científica. Si bien como lenguaje de programación, su curva de aprendizaje suele resultar compleja, el componente de reproducibilidad y flexibilidad en las funciones que contienen son una de las mayores ventajas de quienes lo utilizan.

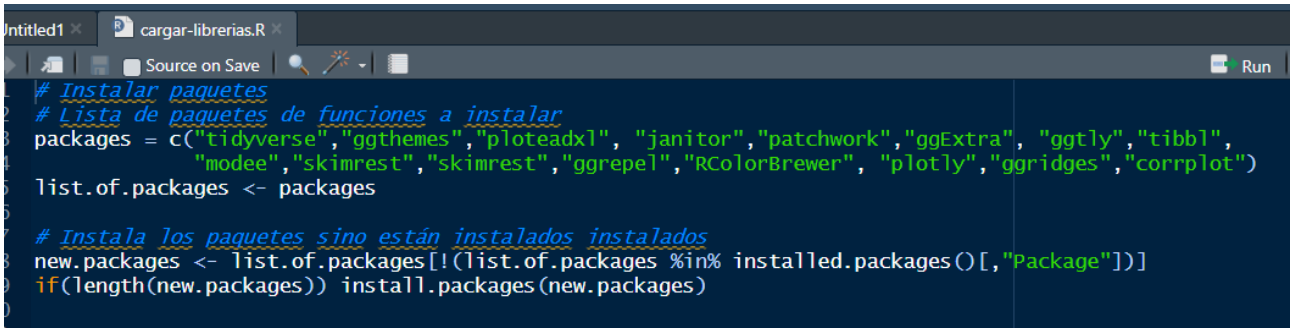
1.2 Crear proyecto

Para seguir estea guía debes descargar el documento **curso_VIZ-DI.zip** que se encuentra en el Repositorio Ridda2. Al descargar el documento, coloque el archivo en cualquier ubicación de su computador y descomprimalo. Dentro del documento aparecerá un archivo llamado **curso_VIZ-DI.Rproj** con un icono parecido a un cubo de color celeste y una R (ver imagen), haga doble click sobre el. Se abrirá un **proyecto** el Rstudio con la configuración del proyecto a utilizar. esta es la carpeta con los códigos y datos iniciales que utilizará en esta guía.



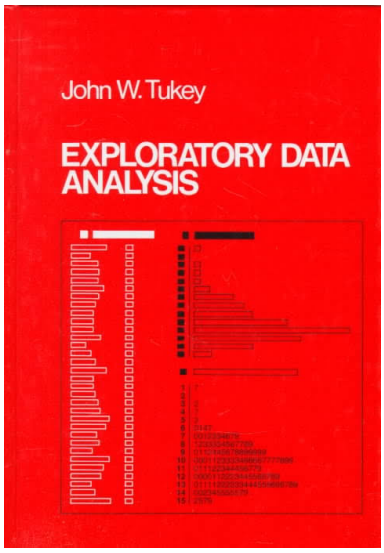
En la carpeta **code** del proyecto creado, hay un archivo con el nombre **cargar-librerias.R**, puede ejecutar el código que se muestra en la figura para instalar los paquetes que nececita para el curso.

En caso de que no se carguen todas las librerias puede instalar de forma independiente cada librería utilizando el comando **install.packages("nombre libreria")**.



2Conceptos

2.1 EDA (Análisis Exploratorio de Datos)



En el año de 1971 el matemático **Jhon W. Turkey** creo una filosofía o enfoque para el análisis de datos que empleaba técnicas (principalmente visuales) para **maximizar el conocimiento de un conjunto de datos**, extraer variables importantes, **detectar valores atípicos**, desarrollar modelos, **detectar patrones**, entre otros, a este conjunto de técnicas, a la cual le llamo **E.D.A.(Exploratory data analysis)**. De manera simplificada el EDA ayuda a determinar las mejores formas de manipular la fuente de datos dada para obtener la respuesta que se necesita garantizando que los resultados sean válidos y aplicables, para ello se apolla de diversos tipos de análisis, donde cada análisis utiliza diveras representaciones gráficas para mostrar los datos de manera visual.

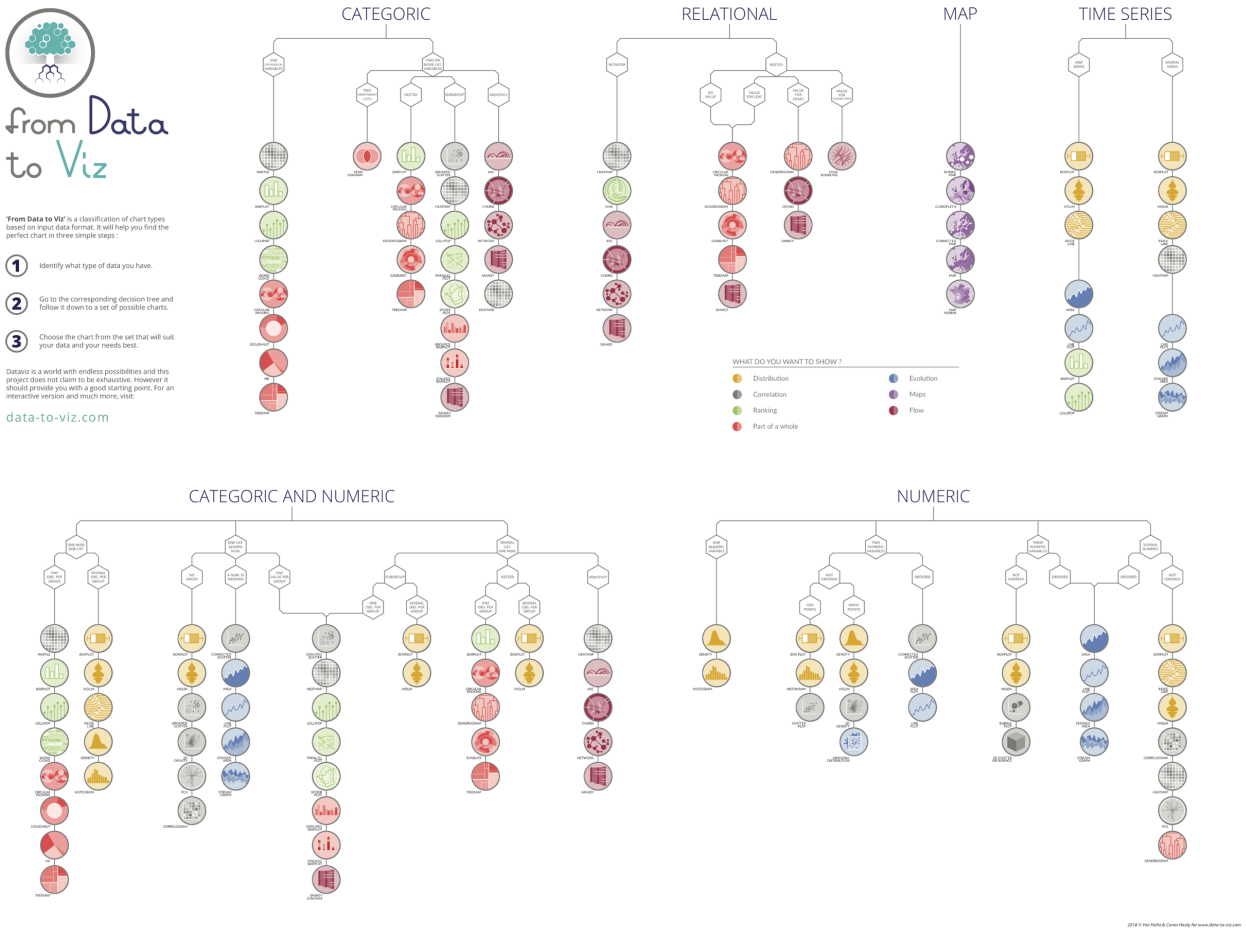
Según Turkey, **Sólo examinando los datos podemos encontrar lo que no esperamos**, por lo que el objetivo del EDA se centra en:

- Explorar los datos para descubrir en ellos pautas subyacentes de estructura y relación que de otro modo no se detectarían.
- Explorar para observar si se cumplen los supuestos paramétricos y en función de ello elegir los estadísticos más adecuados en cada caso o realizar las modificaciones oportunas.

El EDA se vale de diferentes tipos de análisis exploratorios y estas utilizan las representaciones gráficas para mostra de manera visual los datos. Las representaciones graficas pueden darse según el tipo de análisis exploratorio (univariado, bivariado, multivariado), según el tipo de variable o según objetivo del estudio.

Las técnicas gráficas típicas utilizadas en EDA son:

- Histograma
- Diagrama de caja
- Gráficos de barras
- Diagrama de puntos
- Diagrama de tallo y hojas
- Gráficos multivariables
- Mapas de calor
- otros



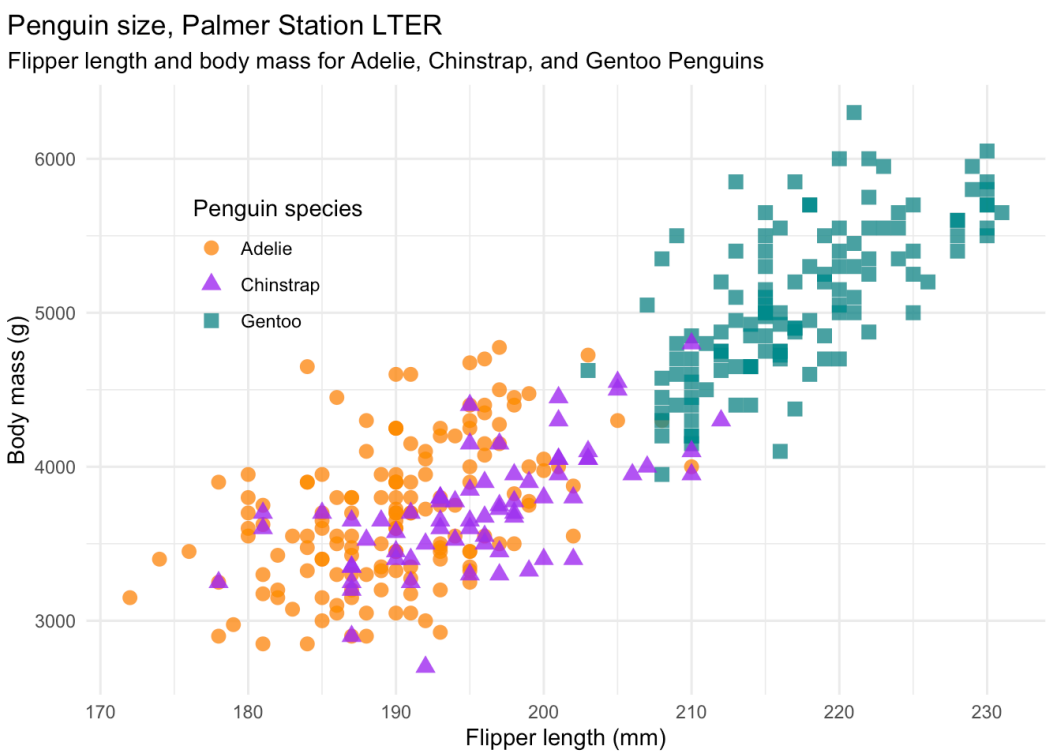
2.2 Visualización de datos

La visualización de datos es una técnica utilizada para brindar información sobre los datos utilizando **señales visuales** como gráficos, tablas, mapas y muchos otros. Esto es útil ya que ayuda a comprender fácilmente y de forma intuitiva grandes cantidades de datos y, por lo tanto, a tomar mejores decisiones al respecto.

La visualización de datos tiene dos propósitos principales:

- Análisis de datos exploratorios e identificación de problemas de datos.
- Comunicar conocimientos y resultados

Las herramientas populares de visualización de datos disponibles son Tableau, Plotly, R, Phyton, Google Charts, DataWraper, Flourish , entre otros. Las distintas plataformas de visualización de datos tienen diferentes capacidades, funcionalidades , sin embargo una de las mayores diferencias de R, es que es lenguaje diseñado para análisis de datos, estadística, visualización e investigación científica. Si bien como lenguaje de programación, su curva de aprendizaje suele resultar compleja, el componente de reprodicibilidad y flexibilidad en las funciones que contienen son una de las mayores ventajas de quienes lo utilizan.


Penguin size, Palmer Station LTER
Flipper length and body mass for Adelie, Chinstrap, and Gentoo Penguins
Penguin species
Adelie
Chinstrap
Gentoo
Body mass (g)
6000
5000
4000
3000
170
180
190
200
210
220
230
Flipper length (mm)

3Librería ggplot

3.1 Cargar librerias

Librerias necesarias que se deben haber instalado previamente con **install.packages(nombre del paquete)** o en el panel **Package**.

```
library(tidyverse) # incluye ggplot2
library(readxl) # funciones para importar xlsx
library(janitor) # funciones de limpieza
library(patchwork) #combinar gráficos de ggplot
library(ggExtra)
library(ggthemes) # nuevas temas para los plots
library(plotly) #gráficos interactivos # remotes::install_github("plotly/plotly")
library(tibble)
library(skimr) # reseumen numerico
library(modeest)
library(ggrepel) # añadir etiquetas a los gráficos
library(RColorBrewer) #paletas de colores
library(plotly) #graficos interactivos
library(corrplot)
library(ggridges)
```

3.2 Carga de datos

Para cargar los datos debe crear un nuevo archivo script de R y empezar a utilizar lo ssiguientes códigos.

Se utilizará una versión de los datos **palmerpinguins**, recopilados y puestos a disposición por la **Dra. Kristen Gorman** y la Estación Palmer, Antártida LTER , miembro de **Long Term Ecological Research Network**.

```
datos_pinguinos <- read.csv("data/data_penguins.csv")
head(datos_pinguinos,5)
```

```
  X SPECIES   ISLAND. bill.lenght bill.depth flipper.lenght MALE FEMALE
1 1  Adelie Torgersen        39.1       18.7            181 3750     NA
2 2  Adelie Torgersen        39.5       17.4            186   NA   3800
3 3  Adelie Torgersen        40.3       18.0            195   NA   3250
4 4  Adelie Torgersen        36.7       19.3            193   NA   3450
5 5  Adelie Torgersen        39.3       20.6            190 3650     NA
```

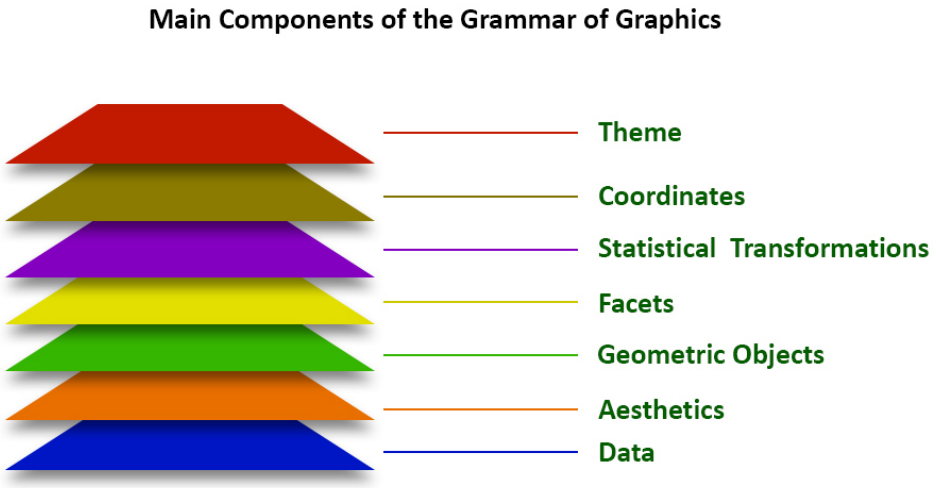
3.3 Librería ggplot2

El paquete **ggplot2** es una paquete incluido en la librería **tidyverse (incluye: ggplot2, dplyr, tidyr, readr, purr,tibble, forcats, lubridate)**, cuya finalidad es entender los gráficos como parte integrar del procesamiento y modelado de los datos, basándose en la idea de *Leland Wilkinson*, **lgramática de los gráficos**. Este concepto permite combinar diferentes elementos en el gráficos

como si fueran capas superpuestas para vincularlas a los datos, además de mapear atributos estéticos como , color, forma, tamaño y la visualización de objetos geométricos (puntos, barras, lineas). **gplot** es la abreviación de *Grammar of Graphics plot*.

Un gráfico en **ggplot** se compone de las siguientes capas:

- **Datos**: conjunto de datos.
- **Mapeado de elementos (aesthetics):** ejes, color, forma, tamaño, etc (en función de los datos)
- **Elementos geométricos (geom):** puntos, líneas, barras, polígonos, etc.
- **Componer gráficas (facet):** visualizar varias gráficas a la vez.
- **Transformaciones estadísticas (stat):** ordenar, resumir, agrupar, etc.
- **Sistema de coordenadas (coord):** coordenadas, grids, etc.
- **Temas (theme):** fuente, tamaño de letra, subtítulos, captions, leyenda, ejes, etc.



3.4 Operador pipe

El operador **pipe** que simbólicamente se puede utilizar como %>% o |> **(versiones recientes)** se utiliza para tomar el resultado de unaexpresión y pasarlo a la siguiente expresión, como si estuvieras *encajando o uniendo* las operaciones una después de la otra. Para utilizar el simbolo del pipe, podemos utilizar las siguientes teclas **CTRL+SHIFT+M**. Este componente es parte de los elementos utilizados en la transformación de datos o Data Wrangling y se utilizará en algunos códigos en este contenido.

```
# Ejemplo 1
# Listar los primeros 10 obs de la tabla
head(datos_pinguinos, 10)
```

```
   X SPECIES   ISLAND. bill.lenght bill.depth flipper.lenght MALE FEMALE
1  1  Adelie Torgersen        39.1       18.7            181 3750     NA
2  2  Adelie Torgersen        39.5       17.4            186   NA   3800
3  3  Adelie Torgersen        40.3       18.0            195   NA   3250
4  4  Adelie Torgersen        36.7       19.3            193   NA   3450
5  5  Adelie Torgersen        39.3       20.6            190 3650     NA
6  6  Adelie Torgersen        38.9       17.8            181   NA   3625
7  7  Adelie Torgersen        39.2       19.6            195 4675     NA
8  8  Adelie Torgersen        41.1       17.6            182   NA   3200
```

```
9   9  Adelie Torgersen        38.6       21.2            191 3800     NA
10 10  Adelie Torgersen        34.6       21.1            198 4400     NA
```

```r
# Utilizando pipe
# llamamos el tibble de datos
datos_pinguinos |>
  head(10)
```

```
    X SPECIES   ISLAND. bill.lenght bill.depth flipper.lenght MALE FEMALE
1   1  Adelie Torgersen        39.1       18.7            181 3750     NA
2   2  Adelie Torgersen        39.5       17.4            186   NA   3800
3   3  Adelie Torgersen        40.3       18.0            195   NA   3250
4   4  Adelie Torgersen        36.7       19.3            193   NA   3450
5   5  Adelie Torgersen        39.3       20.6            190 3650     NA
6   6  Adelie Torgersen        38.9       17.8            181   NA   3625
7   7  Adelie Torgersen        39.2       19.6            195 4675     NA
8   8  Adelie Torgersen        41.1       17.6            182   NA   3200
9   9  Adelie Torgersen        38.6       21.2            191 3800     NA
10 10  Adelie Torgersen        34.6       21.1            198 4400     NA
```

```r
# Ejemplo 2
# Resumen de lso datos
summary(datos_pinguinos)
```

```
       X         SPECIES            ISLAND.           bill.lenght   
 Min.   :  1   Length:333         Length:333         Min.   :32.10  
 1st Qu.: 84   Class :character   Class :character   1st Qu.:39.50  
 Median :167   Mode  :character   Mode  :character   Median :44.50  
 Mean   :167                                         Mean   :43.99  
 3rd Qu.:250                                         3rd Qu.:48.60  
 Max.   :333                                         Max.   :59.60  

  bill.depth    flipper.lenght      MALE          FEMALE     
 Min.   :13.10   Min.   :172    Min.   :3250   Min.   :2700  
 1st Qu.:15.60   1st Qu.:190    1st Qu.:3900   1st Qu.:3350  
 Median :17.30   Median :197    Median :4300   Median :3650  
 Mean   :17.16   Mean   :201    Mean   :4546   Mean   :3862  
 3rd Qu.:18.70   3rd Qu.:213    3rd Qu.:5312   3rd Qu.:4550  
 Max.   :21.50   Max.   :231    Max.   :6300   Max.   :5200  
                                NA's   :165    NA's   :168   
```

```r
# Usando pipe
datos_pinguinos |>
  summary()
```

```
       X         SPECIES            ISLAND.           bill.lenght   
 Min.   :  1   Length:333         Length:333         Min.   :32.10  
 1st Qu.: 84   Class :character   Class :character   1st Qu.:39.50  
 Median :167   Mode  :character   Mode  :character   Median :44.50  
```

```
Mean   :167                                          Mean   :43.99
3rd Qu.:250                                          3rd Qu.:48.60
Max.   :333                                          Max.   :59.60

  bill.depth    flipper.lenght      MALE          FEMALE
Min.   :13.10   Min.   :172    Min.   :3250   Min.   :2700
1st Qu.:15.60   1st Qu.:190    1st Qu.:3900   1st Qu.:3350
Median :17.30   Median :197    Median :4300   Median :3650
Mean   :17.16   Mean   :201    Mean   :4546   Mean   :3862
3rd Qu.:18.70   3rd Qu.:213    3rd Qu.:5312   3rd Qu.:4550
Max.   :21.50   Max.   :231    Max.   :6300   Max.   :5200
                               NA's   :165    NA's   :168
```

```r
# Ejemplo 3
# Seleccionar las columnas 3 y 5 del tibble datospinguinos
# donde island es igual a Torgersen
# mostrar priemaras 5 filas
head(datos_pinguinos[datos_pinguinos$ISLAND.=="Torgersen",c(4,5)],5)
```

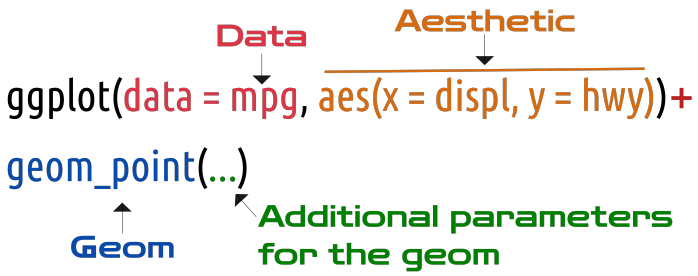
```
  bill.lenght bill.depth
1        39.1       18.7
2        39.5       17.4
3        40.3       18.0
4        36.7       19.3
5        39.3       20.6
```

```r
# Utilizando pipe
datos_pinguinos |>
  filter(ISLAND.=="Torgersen") |>
  select(bill.lenght,bill.depth) |>
  head(5)
```

```
  bill.lenght bill.depth
1        39.1       18.7
2        39.5       17.4
3        40.3       18.0
4        36.7       19.3
5        39.3       20.6
```

3.5 Estructura de ggplot

La estrutura base de **ggplot** es la siguiente:



3.6 Función ggplot()

Utilizando solo la función **ggplot()**, se crea un lienzo o espacio de trabajo para la creación de un gráfico.

```
ggplot()
```


3.7 Cargar datos en ggplot

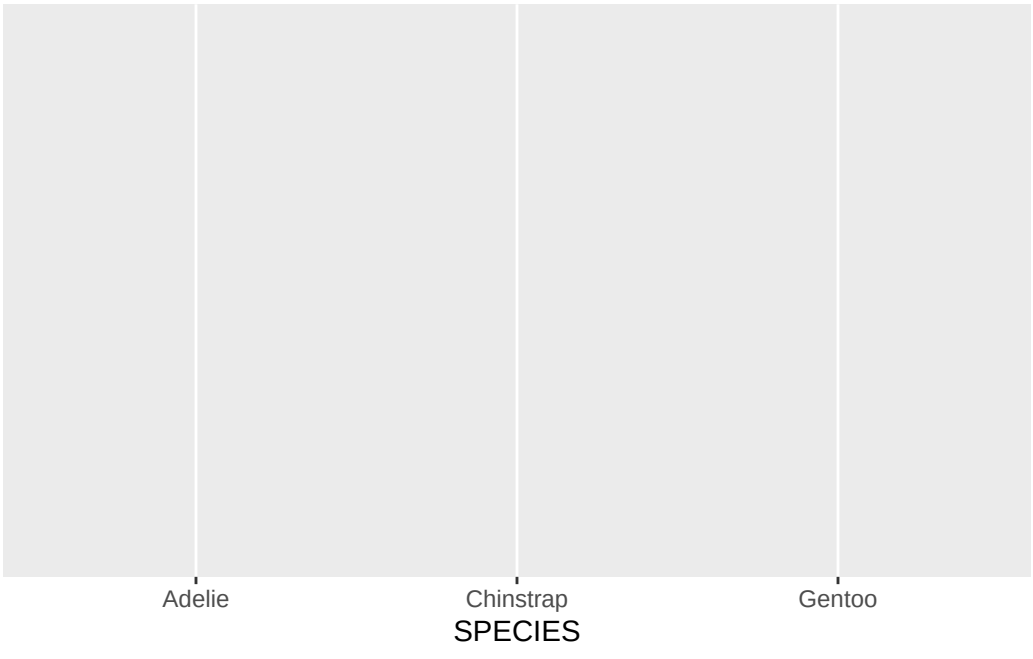
data: es el nombre del tabla en formato tibble o dataframe y se utiliza para cargar los datos, en este caso la tabla **datos_pinguinos**.

```
ggplot(data=datos_pinguinos)
```

3.8 Mapeado y estética

mapping: define el mapeo estético de señales proporcionadas por el set de datos, las variables que modifican estas señales deben estar integradas en *aes()* que integran las escalas, x, y, que representarán cada observación según variables, tambien en este apartado se identifican otras señales como color, size. En este ejemplo utilizaremos solo la varable **SPECIES**. Se mostrará en el eje de las X los datos **CATEGORICOS** de la variable **SPECIES**.

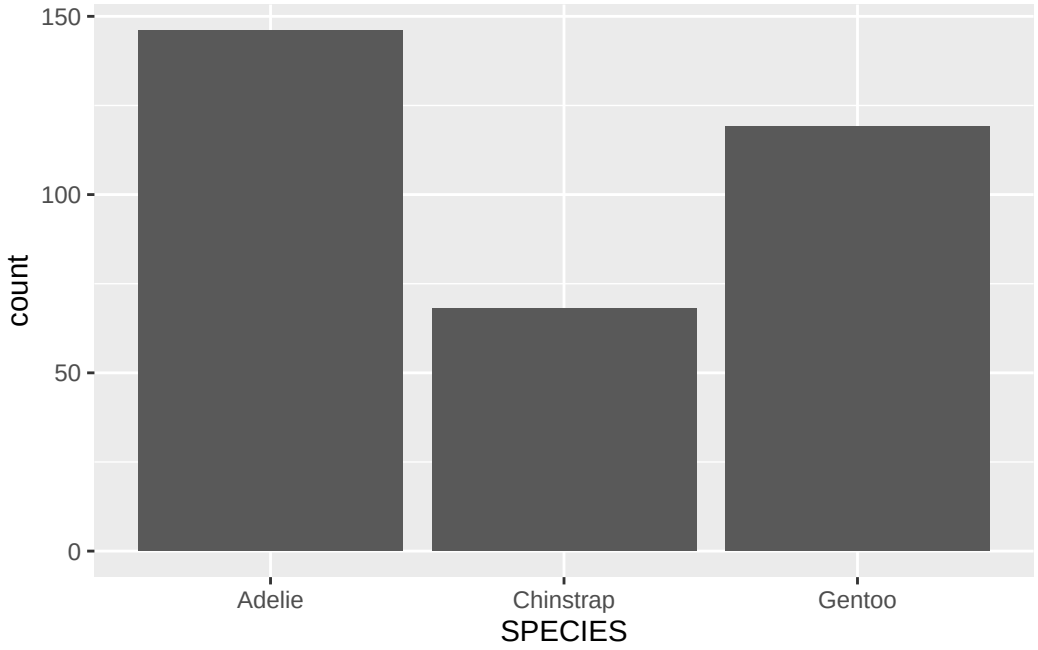
```
ggplot(data = datos_pinguinos,
       mapping = aes(x=SPECIES))
```



3.9 Geometría

Para poder dibujar el gráfico, ggplot necesita que añadamos una nueva capa de la geometría o tipo de gráfico a utilizar. Para añadir la nueva capa debemos de colocar el signo de + al final de la línea de **mapping** y luego el tipo de geometría Geometrías ggplot. En este ejemplo utilizaremos **geom_bar()**: representa el tipo de geometría de barras que muestra la figura de los gráficos que interperta la agrupación de datos de la variable categórica **SPECIES**.

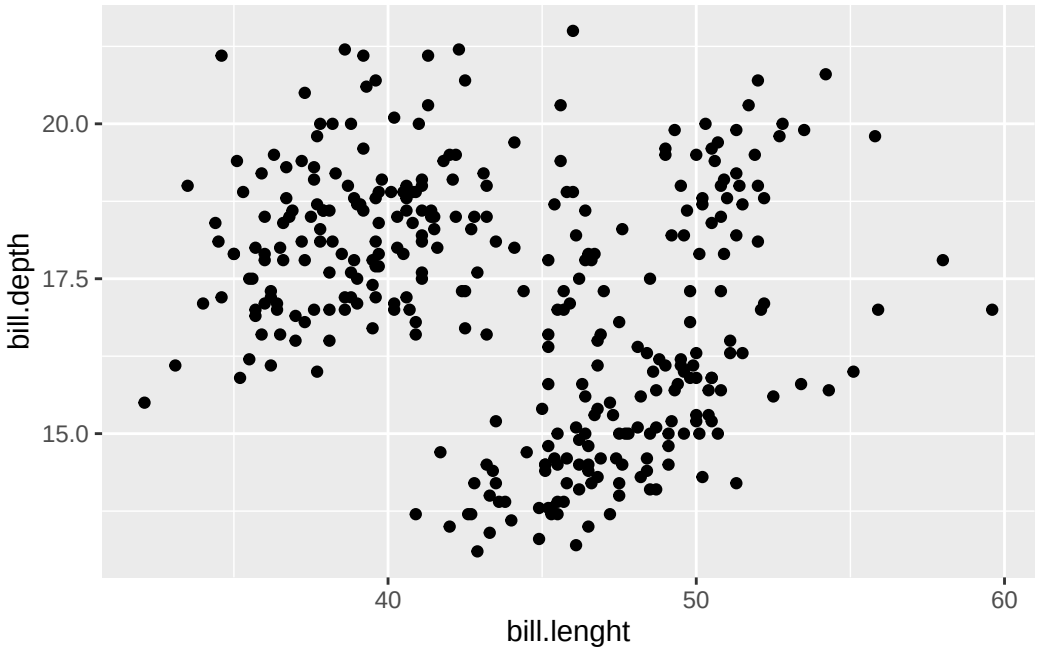
```
ggplot(data = datos_pinguinos,
       mapping = aes(x=SPECIES))+
  geom_bar()
```



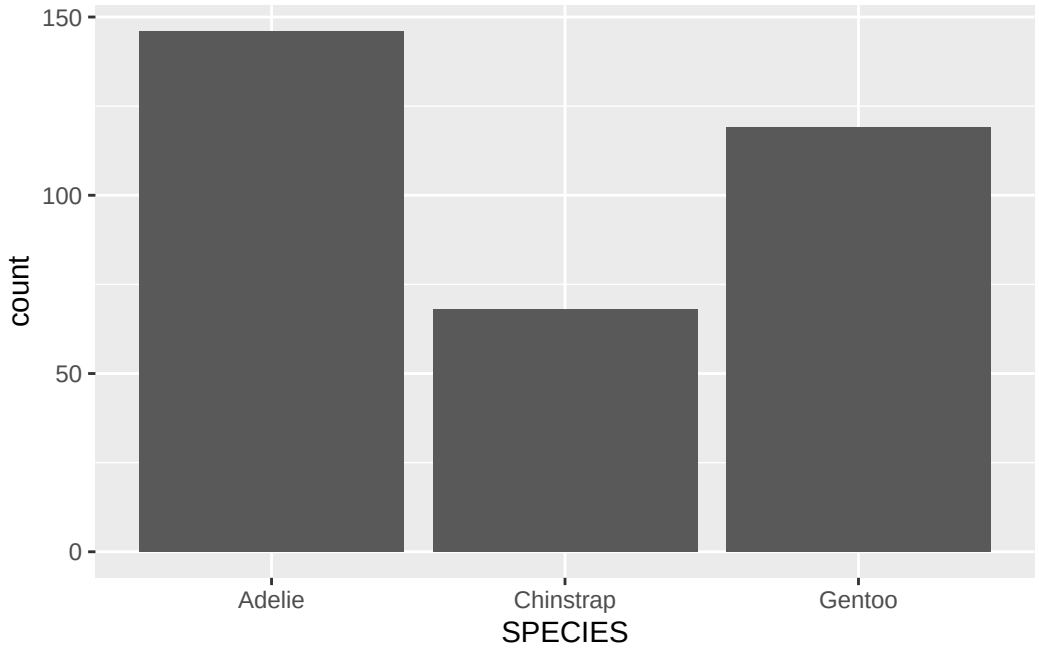
3.10 Ejemplo de gráficos en ggplot

A manera de ejemplo mostraremos tres tipos de gráficos en ggplot con **diferentes geometrías**, solo con el objetivo de ver la estructura básica funcionando donde con los mismos datos solo cambio el tipo de geometría **geom_**. Al avanzar el contenido se mostrará como funciona el paquete **ggplot** y el funcionamiento de estas geometrias de forma más detallada.

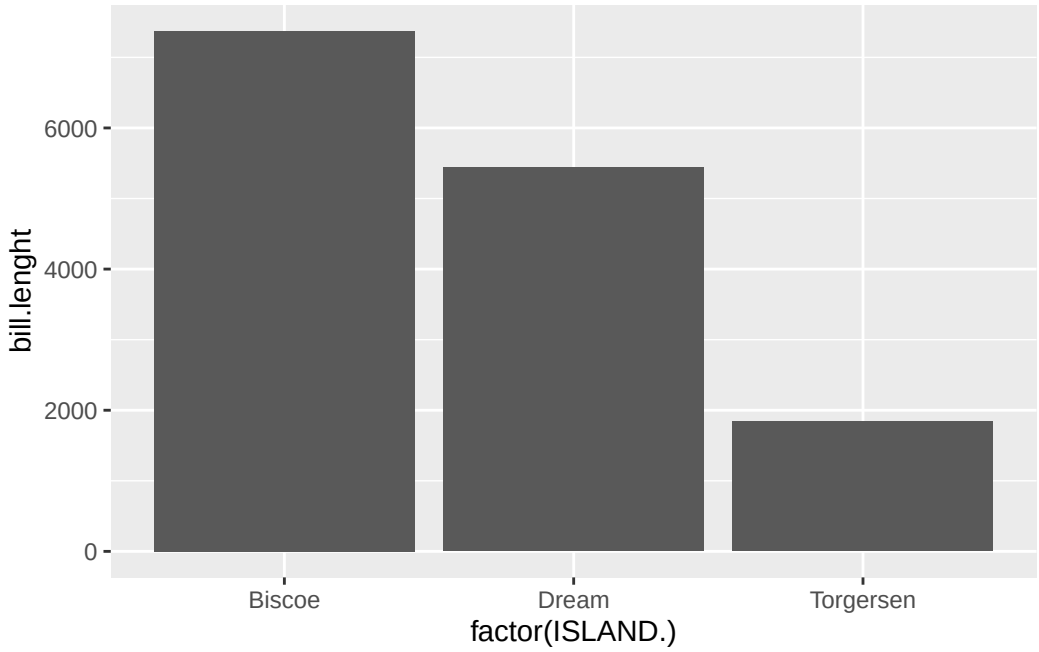
```
#Ejemplo de gráfico de puntos
ggplot(data = datos_pinguinos,
       mapping = aes(x=bill.lenght, y=bill.depth))+
       geom_point()
```




```
# Ejemplo de gráfico de barras una variable
ggplot(data = datos_pinguinos,
       mapping = aes(x=SPECIES))+
       geom_bar()
```




```
# Ejemplo de gráfico de barras dos variables
ggplot(data = datos_pinguinos,
       mapping = aes(x=factor(ISLAND.), y=bill.lenght))+
       geom_col()
```



3.11 Tipos de datos en ggplot

Para hacer gráficos con ggplot es necesario que el conjunto de datos sea un tipo de datos estructurado de dos dimensiones que permiten diferentes tipos de datos, por lo que son heterogéneos, en este caso , este tipo de estructuras de datos en R, son los **dataframes** o los **tibble**.

3.11 Objeto tipo dataframe

Los **data frame** son datos estructurados, lo que comunmente conocemos como *tablas*. Su mayor características es que los datos que lo conforman pueden ser de diferentes tipos, numéricos (discretos, continuos), caracteres, lógicos, etc. Una de sus limitantes es que al cargar los datos R puede modificarr el nombre de las variables, si contienen espacios en blancos o algunos carateres especiales, colocan puntos a la variable.

```
# data.frame mtcars muestra todas las filas, se indica con head, mostrar solo 25
head(datos_pinguinos,25)
```

```
    X SPECIES   ISLAND. bill.lenght bill.depth flipper.lenght MALE FEMALE
1   1  Adelie Torgersen        39.1       18.7            181 3750     NA
2   2  Adelie Torgersen        39.5       17.4            186   NA   3800
3   3  Adelie Torgersen        40.3       18.0            195   NA   3250
4   4  Adelie Torgersen        36.7       19.3            193   NA   3450
5   5  Adelie Torgersen        39.3       20.6            190 3650     NA
6   6  Adelie Torgersen        38.9       17.8            181   NA   3625
7   7  Adelie Torgersen        39.2       19.6            195 4675     NA
8   8  Adelie Torgersen        41.1       17.6            182   NA   3200
9   9  Adelie Torgersen        38.6       21.2            191 3800     NA
10 10  Adelie Torgersen        34.6       21.1            198 4400     NA
11 11  Adelie Torgersen        36.6       17.8            185   NA   3700
12 12  Adelie Torgersen        38.7       19.0            195   NA   3450
13 13  Adelie Torgersen        42.5       20.7            197 4500     NA
```

```
14 14  Adelie Torgersen        34.4       18.4            184   NA   3325
15 15  Adelie Torgersen        46.0       21.5            194 4200     NA
16 16  Adelie    Biscoe        37.8       18.3            174   NA   3400
17 17  Adelie    Biscoe        37.7       18.7            180 3600     NA
18 18  Adelie    Biscoe        35.9       19.2            189   NA   3800
19 19  Adelie    Biscoe        38.2       18.1            185 3950     NA
20 20  Adelie    Biscoe        38.8       17.2            180 3800     NA
21 21  Adelie    Biscoe        35.3       18.9            187   NA   3800
22 22  Adelie    Biscoe        40.6       18.6            183 3550     NA
23 23  Adelie    Biscoe        40.5       17.9            187   NA   3200
24 24  Adelie    Biscoe        37.9       18.6            172   NA   3150
25 25  Adelie    Biscoe        40.5       18.9            180 3950     NA
```

```
# Verificar Objeto tipo data.frame
class(datos_pinguinos)
```

```
[1] "data.frame"
```

3.11 Objeto tipo tibble

Los **tibble** son un tipo de **data frame** para una gestión más ágil, eficiente y coherente. Las tablas en formato tibble tienen 4 ventajas principales frente a los data frame:

- Permiten imprimir por consola la tabla con **mayor información de las variables**, solo imprime por defecto las primeras filas (todas si son 20 o menos, 10 si son más de 20 filas). Puedes imprimir las filas y columnas que quieras con **print(nombre de tibble)**, pero por defecto te aseguras de no saturar la consola.
- Mantiene la **integridad de los datos** (no cambia los tipos de las variables y hace una carga de datos inteligente, **interpretando los tipos de datos** y colocandolos encima de cada variable.
- Si accedes a una columna que no existe **avisa con un warning**.
- No solo no cambiará el tipo de datos sino que **no cambiará el nombre de las variables** (los data frame transforma los caracteres que no sean letras).

```
# transformat dataframe a tible
datos_pinguinos <- as_tibble(datos_pinguinos)

# Verificar Objeto tipo tibble
class(datos_pinguinos)
```

```
[1] "tbl_df"     "tbl"        "data.frame"
```

```
# tibble imprime solo las primeras 10
# coloca el tipo de dato sencima del nombre de la variable
head(datos_pinguinos,10)
```

```
# A tibble: 10 x 8
       X SPECIES ISLAND.   bill.lenght bill.depth flipper.lenght  MALE FEMALE
   <int> <chr>   <chr>           <dbl>      <dbl>          <int> <int>  <int>
 1     1 Adelie  Torgersen        39.1       18.7            181  3750     NA
```

```
 2     2 Adelie  Torgersen        39.5       17.4            186    NA   3800
 3     3 Adelie  Torgersen        40.3       18              195    NA   3250
 4     4 Adelie  Torgersen        36.7       19.3            193    NA   3450
 5     5 Adelie  Torgersen        39.3       20.6            190  3650     NA
 6     6 Adelie  Torgersen        38.9       17.8            181    NA   3625
 7     7 Adelie  Torgersen        39.2       19.6            195  4675     NA
 8     8 Adelie  Torgersen        41.1       17.6            182    NA   3200
 9     9 Adelie  Torgersen        38.6       21.2            191  3800     NA
10    10 Adelie  Torgersen        34.6       21.1            198  4400     NA
```

3.11 Resumen de datos

Existen diversas formas de mostrar la estructura o resumen de datos en R, en este caso los datos **datos_pinguinos** (dataframe o tibble) con sus tipos de datos según variables es utilizando la función **summary()** y **glipmse()**, esto permite identificar anomalías en los nombre de las variables o en el tipo de variables y es un elemento indispensable para conocer los datos y sus tipos.

```
# Muestra la estructura de los datos y el tipo de datos
datos_pinguinos |>
  summary()
```

```
       X          SPECIES            ISLAND.           bill.lenght   
 Min.   :  1   Length:333         Length:333         Min.   :32.10  
 1st Qu.: 84   Class :character   Class :character   1st Qu.:39.50  
 Median :167   Mode  :character   Mode  :character   Median :44.50  
 Mean   :167                                         Mean   :43.99  
 3rd Qu.:250                                         3rd Qu.:48.60  
 Max.   :333                                         Max.   :59.60  

  bill.depth    flipper.lenght      MALE          FEMALE     
 Min.   :13.10   Min.   :172    Min.   :3250   Min.   :2700  
 1st Qu.:15.60   1st Qu.:190    1st Qu.:3900   1st Qu.:3350  
 Median :17.30   Median :197    Median :4300   Median :3650  
 Mean   :17.16   Mean   :201    Mean   :4546   Mean   :3862  
 3rd Qu.:18.70   3rd Qu.:213    3rd Qu.:5312   3rd Qu.:4550  
 Max.   :21.50   Max.   :231    Max.   :6300   Max.   :5200  
                                NA's   :165    NA's   :168   
```

```
# La función glimpse es otra forma de mostrar los datos
# sobre todo cuando son muchas variables
# Muestra tambien el tipo de dato por variable
# Permite mostrar la estructura de datos de forma más ordenada
datos_pinguinos |>
  glimpse()
```

```
Rows: 333
Columns: 8
$ X             <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, ~
$ SPECIES       <chr> "Adelie", "Adelie", "Adelie", "Adelie", "Adelie", "Adel~
$ ISLAND.       <chr> "Torgersen", "Torgersen", "Torgersen", "Torgersen", "To~
$ bill.lenght   <dbl> 39.1, 39.5, 40.3, 36.7, 39.3, 38.9, 39.2, 41.1, 38.6, 3~
```

```
$ bill.depth     <dbl> 18.7, 17.4, 18.0, 19.3, 20.6, 17.8, 19.6, 17.6, 21.2, 2~
$ flipper.lenght <int> 181, 186, 195, 193, 190, 181, 195, 182, 191, 198, 185, ~
$ MALE           <int> 3750, NA, NA, NA, 3650, NA, 4675, NA, 3800, 4400, NA, N~
$ FEMALE         <int> NA, 3800, 3250, 3450, NA, 3625, NA, 3200, NA, NA, 3700,~
```

3.11 Limpieza de nombre de variables

Si bien la limpieza de los datos es un elemento importante al visualizar datos, no es un tema que veremos en este curso, sin embargo podemos evaluar si el nombre de las variables no están normalizados, que tengan espacios en blanco, ya que si la estructura de datos es un dataframe, R los carga con um punto (.), además de que algunas están en mayúscula y otra en minúscula. Por otro lado podemos eliminar las columnas irrelevantes como la columna **X** en el tibble, que no representa ningún dato de interés en la tabla.

```
# Nombre de las variables
datos_pinguinos |>
  names()
```

```
[1] "X"              "SPECIES"        "ISLAND."        "bill.lenght"
[5] "bill.depth"     "flipper.lenght" "MALE"           "FEMALE"
```

```
# limpieza y normalización de nombres de las variables con janitor
datos_pinguinos <- janitor::clean_names(datos_pinguinos)

# eliminar primera columna
datos_pinguinos <- datos_pinguinos[,-1]
datos_pinguinos |>
  names()
```

```
[1] "species"        "island"         "bill_lenght"    "bill_depth"
[5] "flipper_lenght" "male"           "female"
```

```
# Resumen de datos
datos_pinguinos |>
  glimpse()
```

```
Rows: 333
Columns: 7
$ species        <chr> "Adelie", "Adelie", "Adelie", "Adelie", "Adelie", "Adel~
$ island         <chr> "Torgersen", "Torgersen", "Torgersen", "Torgersen", "To~
$ bill_lenght    <dbl> 39.1, 39.5, 40.3, 36.7, 39.3, 38.9, 39.2, 41.1, 38.6, 3~
$ bill_depth     <dbl> 18.7, 17.4, 18.0, 19.3, 20.6, 17.8, 19.6, 17.6, 21.2, 2~
$ flipper_lenght <int> 181, 186, 195, 193, 190, 181, 195, 182, 191, 198, 185, ~
$ male           <int> 3750, NA, NA, NA, 3650, NA, 4675, NA, 3800, 4400, NA, N~
$ female         <int> NA, 3800, 3250, 3450, NA, 3625, NA, 3200, NA, NA, 3700,~
```

3.11 Transformación de variables

El resumen de datos tambien permite identificar las variables por tipo, en este caso, dos variables han sido identificadas de tipo caracter **species, island**, pero realmente son variables categoricas. Una representa las especies de pinguinos y la otra la isla donde se realizo la observación, por lo que debe realizarse su transformación a tipo categórico o en este caso de tipo **factor,** para ello utilizamos **as.factor(),** para transformar la variable.

```
#Verifcar si una variable tiene datos cualitaticos repetidos
# lo que indica que puede ser una var categorica
table(datos_pinguinos$species)
```

```
   Adelie Chinstrap    Gentoo
      146        68       119
```

```
#resultado
# Adelie Chinstrap    Gentoo
#  146        68       119


# Transformar variables de tipo caracter a factor
datos_pinguinos$species <- as.factor(datos_pinguinos$species)
datos_pinguinos$island<- as.factor(datos_pinguinos$island)

#verificar variables
datos_pinguinos |>
  glimpse()
```

```
Rows: 333
Columns: 7
$ species        <fct> Adelie, Adelie, Adelie, Adelie, Adelie, Adelie, Adelie,~
$ island         <fct> Torgersen, Torgersen, Torgersen, Torgersen, Torgersen, ~
$ bill_lenght    <dbl> 39.1, 39.5, 40.3, 36.7, 39.3, 38.9, 39.2, 41.1, 38.6, 3~
$ bill_depth     <dbl> 18.7, 17.4, 18.0, 19.3, 20.6, 17.8, 19.6, 17.6, 21.2, 2~
$ flipper_lenght <int> 181, 186, 195, 193, 190, 181, 195, 182, 191, 198, 185, ~
$ male           <int> 3750, NA, NA, NA, 3650, NA, 4675, NA, 3800, 4400, NA, N~
$ female         <int> NA, 3800, 3250, 3450, NA, 3625, NA, 3200, NA, NA, 3700,~
```

3.11 Tidy data

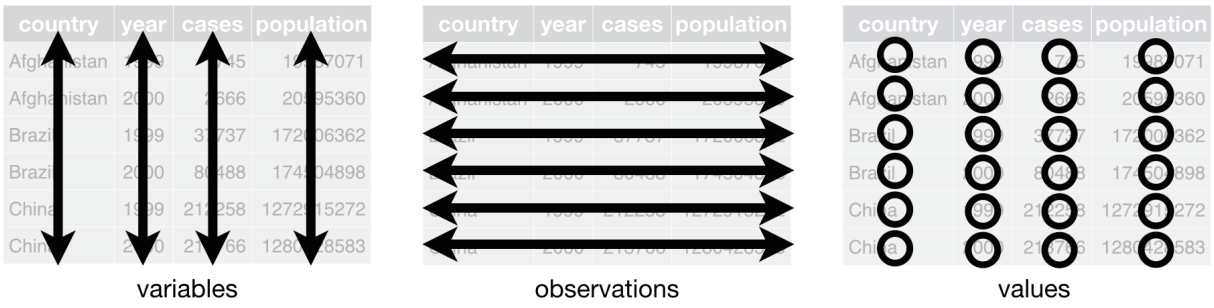
Un elemento importante en los datos que utiliza ggplot, no es solo la estructura de la tabla (tibble o dataframe) ya que es posible generar visualizaciones utilizando cualquiera de las dos estructura de datos, lo que si es relevante es que ggplot solo trabaja con datos tipo **tidy** o datos ordenados.

La librería **dplyr, ggplot2** y todos los demás paquetes de **tidyverse** están diseñados para funcionar con datos ordenados.

Los datos ordenados es una forma de mostrar los datos basados en tres reglas interrelacionadas:

- Cada variable debe tener su propia columna.
- Cada observación debe tener su propia fila.

- Cada valor debe tener su propia celda.




```
datos_pinguinos |>
  head(10)
```

```
# A tibble: 10 x 7
   species island    bill_lenght bill_depth flipper_lenght  male female
   <fct>   <fct>           <dbl>      <dbl>          <int> <int>  <int>
 1 Adelie  Torgersen        39.1       18.7            181  3750     NA
 2 Adelie  Torgersen        39.5       17.4            186    NA   3800
 3 Adelie  Torgersen        40.3       18              195    NA   3250
 4 Adelie  Torgersen        36.7       19.3            193    NA   3450
 5 Adelie  Torgersen        39.3       20.6            190  3650     NA
 6 Adelie  Torgersen        38.9       17.8            181    NA   3625
 7 Adelie  Torgersen        39.2       19.6            195  4675     NA
 8 Adelie  Torgersen        41.1       17.6            182    NA   3200
 9 Adelie  Torgersen        38.6       21.2            191  3800     NA
10 Adelie  Torgersen        34.6       21.1            198  4400     NA
```

Adicional a estas tres reglas podemos identificar que un conjunto de datos no está ordenado porque:

- Los encabezados de columna son valores, no nombres de variables
- Las variables se almacenan tanto en filas como en columnas.
- Los datos de una supuesta variable están en diferentes columnas.

En el caso del tibble **datos_pinguinos** se observa que existen dos variables **male** y **female** que contienen los valores de la masa corporal de los pinguinos **body_mass_g**.

Dentro de las reglas de los datos tidy, **cada variable debe tener su propia columna**, La masa corporal de los pinguinos debiera estar en una sola columan y no separada en las variables **male** y **female**, por otro lado, **los encabezados de columna son valores, no nombres de variables**, las variables **male y female** debieran estra en una variable del tipo sexo, que corresponde a una categoria.

Esto indica que esta tibble **datos_pinguinos** no esta ordenado, este tipo de tabla se conoce como data **wider o tabla ancha,** ya que su crecimiento es a lo ancho de la tabla, por lo que debemos transformar estos datos en formato **data larga o data longer**, datos que crecen a lo largo de la tabla.Para ellos haremos usos de las funciones pivotes de R, especificamente **pivot_longer()**.

```
# Transformar  a datos tidy
# Para transformar los datos a tidy, se identifican las columnas o variables
# que se desean unir y convertir en una sola variable
```

```r
# se coloca el nombre de la nueva variable con names_to
# los datos que contenian las variable male y female pasan a otra variable
# a través de values_to.

datos_pinguinos_tidy <- datos_pinguinos |>
                          pivot_longer(c("male":"female"),
                          names_to = "sex",
                          values_to = "body_mass_g")

# numero de observaciones generadas 666
nrow(datos_pinguinos_tidy)
```

```
[1] 666
```

```r
datos_pinguinos_tidy |>
  head(10)
```

```
# A tibble: 10 x 7
   species island    bill_lenght bill_depth flipper_lenght sex    body_mass_g
   <fct>   <fct>           <dbl>      <dbl>          <int> <chr>        <int>
 1 Adelie  Torgersen        39.1       18.7            181 male          3750
 2 Adelie  Torgersen        39.1       18.7            181 female          NA
 3 Adelie  Torgersen        39.5       17.4            186 male            NA
 4 Adelie  Torgersen        39.5       17.4            186 female        3800
 5 Adelie  Torgersen        40.3       18              195 male            NA
 6 Adelie  Torgersen        40.3       18              195 female        3250
 7 Adelie  Torgersen        36.7       19.3            193 male            NA
 8 Adelie  Torgersen        36.7       19.3            193 female        3450
 9 Adelie  Torgersen        39.3       20.6            190 male          3650
10 Adelie  Torgersen        39.3       20.6            190 female          NA
```

Al mostrar los datos en formato tidy, podemos observar que se realizó el pivote de la tabla y se crearon las nuevas variables **sex** y **body_mass_g**, sin embargo, dentro de los datos había valores vacios **(NA)** en la tabla original que tambien se cargaron en esta tabla ordenada. Para eliminar estos datos debemos realizar nuevamente el proceso de pivoteo e indicarle a **pivot_longer()** con **values_drop_na = TRUE** para que elimine los datos vacios.

```r
# transformar a datos tidy eliminados filas vacias

datos_pinguinos <- datos_pinguinos |>
                          pivot_longer(c("male":"female"),
                          names_to = "sex",
                          values_to = "body_mass_g",
                          values_drop_na = TRUE)

# numero de observaciones generadas 333
nrow(datos_pinguinos)
```

```
[1] 333
```

```r
# Resumend de datos , donde ya no se muestran los datos NA
datos_pinguinos |>
  head(10)
```

```
# A tibble: 10 x 7
   species island    bill_lenght bill_depth flipper_lenght sex    body_mass_g
   <fct>   <fct>           <dbl>      <dbl>          <int> <chr>        <int>
 1 Adelie  Torgersen        39.1       18.7            181 male          3750
 2 Adelie  Torgersen        39.5       17.4            186 female        3800
 3 Adelie  Torgersen        40.3       18              195 female        3250
 4 Adelie  Torgersen        36.7       19.3            193 female        3450
 5 Adelie  Torgersen        39.3       20.6            190 male          3650
 6 Adelie  Torgersen        38.9       17.8            181 female        3625
 7 Adelie  Torgersen        39.2       19.6            195 male          4675
 8 Adelie  Torgersen        41.1       17.6            182 female        3200
 9 Adelie  Torgersen        38.6       21.2            191 male          3800
10 Adelie  Torgersen        34.6       21.1            198 male          4400
```

```r
#transformar la variable sex a factor
datos_pinguinos$sex <- as.factor(datos_pinguinos$sex)

# write.csv(datos_pinguinos, "data/data_penguins_clean.csv")
```

3.11 Resumen de datos con función skim()

Otra de las funciones para hacer resumen de datos es la función **skim()**, donde se muestra el número de observaciones, variables y tipos de datos que contiene la tabla con las modificaciones realizadas, además de otras informaciones de interés, como la distribución de los datos a través de gráficos de histograma.

```r
# Resumen Numérico de la tabla
datos_pinguinos |>
   skim()
```

Table 3.1: Data summary

Name	datos_pinguinos
Number of rows	333
Number of columns	7
_____	
Column type frequency:	
factor	3
numeric	4
_____	
Group variables	None

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
species	0	1	FALSE	3	Ade: 146, Gen: 119, Chi: 68
island	0	1	FALSE	3	Bis: 163, Dre: 123, Tor: 47
sex	0	1	FALSE	2	mal: 168, fem: 165

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
bill_lenght	0	1	43.99	5.47	32.1	39.5	44.5	48.6	59.6	
bill_depth	0	1	17.16	1.97	13.1	15.6	17.3	18.7	21.5	
flipper_lenght	0	1	200.97	14.02	172.0	190.0	197.0	213.0	231.0	
body_mass_g	0	1	4207.06	805.22	2700.0	3550.0	4050.0	4775.0	6300.0	

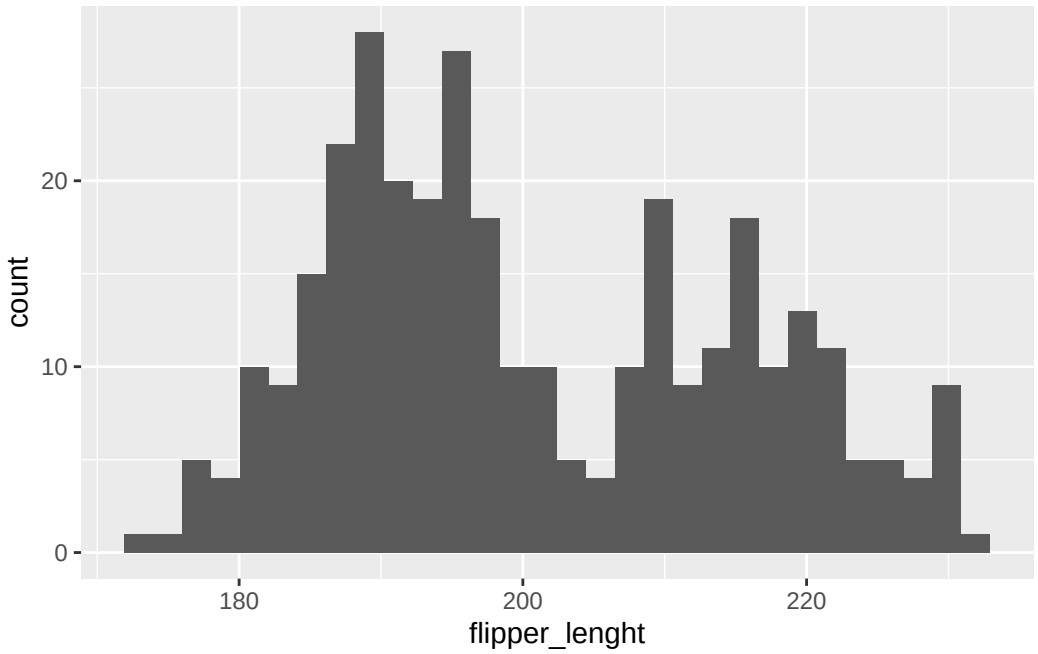
4Gráfico de Histograma en R

Un histograma visualiza la **distribución y frecuencia datos** durante un intervalo continuo o un período de tiempo determinado de **una varible numérica**. Cada barra en un histograma representa la frecuencia tabulada en cada intervalo tambien llamado **bin**. Los histogramas ayudan a dar una estimación de dónde se concentran los valores, cuáles son los extremos y si hay valores inusuales.

Como ejemplo podemos genertar un histograma c utilizando la variable numérica **flipper_lenght** y la geometría **geom_histogram()**. En ggplot el número de bins o intervalos predeterminado es de 30. Este histograma muestra una distribución de los datos entre los valores de 170 a 230 no simétrica y los valores que más se repiten están entre los intervalos 180 a 200.

```
# Gráfico de histograma
datos_pinguinos |>
  ggplot(aes(x= flipper_lenght)) +
  geom_histogram()
```

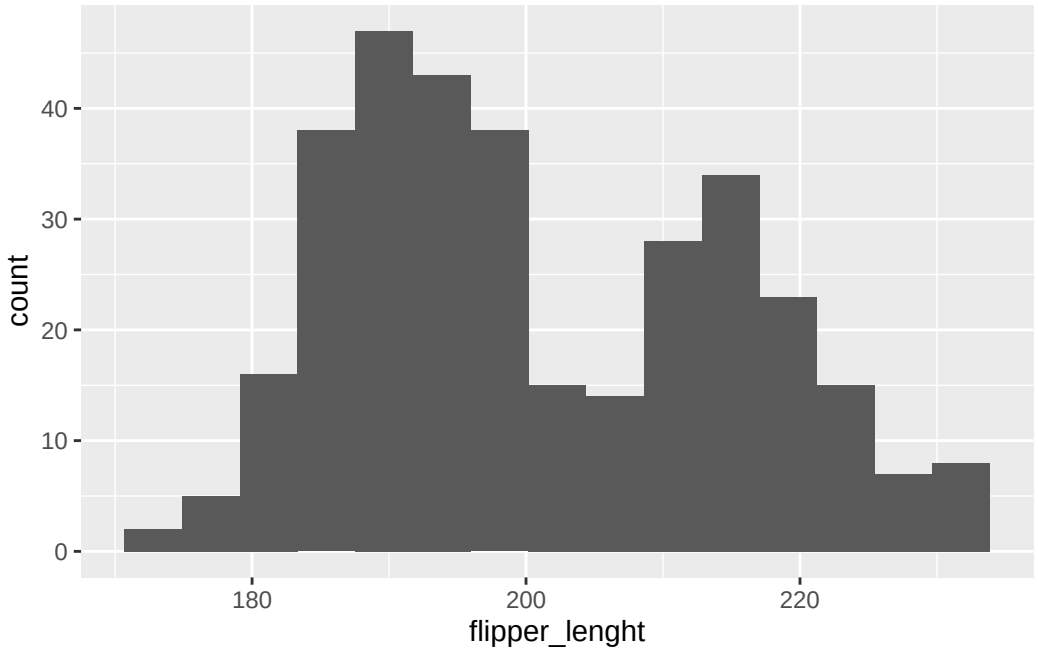
```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



4.1 Modifcar el número de bins

Al modificar el número de intervalos, en este caso menos intervalos, más datos se agrupan en cada intervalo, por lo que el el eje de las **y** tiende a incrementarse.

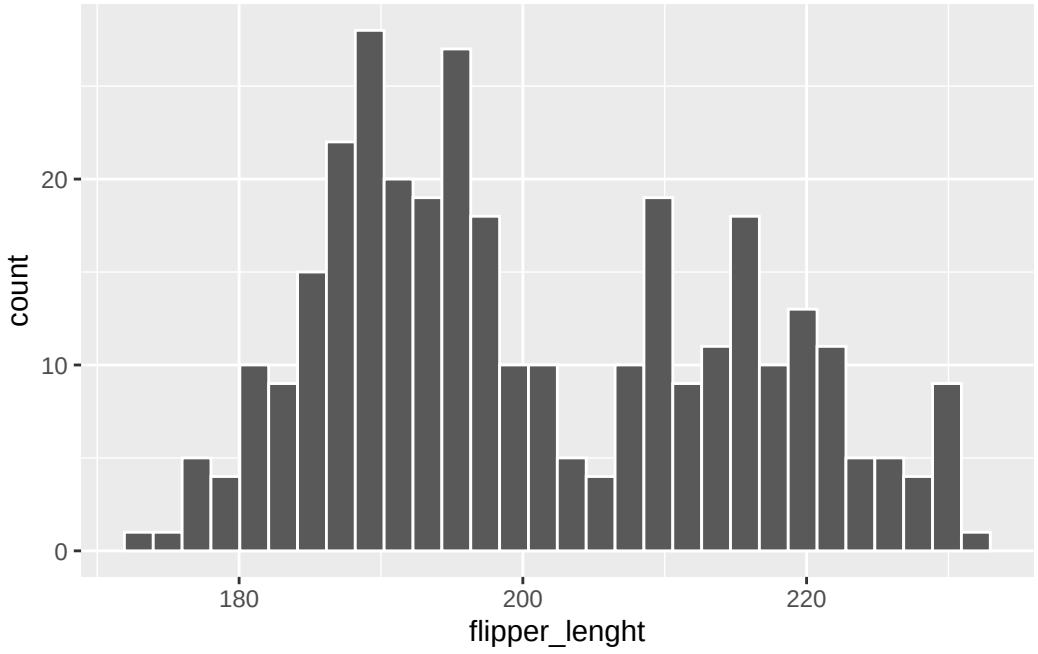
```
# Modificar número de bins
datos_pinguinos |>
  ggplot(aes(x= flipper_lenght)) +
  geom_histogram(bins = 15)
```



4.2 Modificar color de las líneas

Utilizando nuevamente el intervalo de 30, es un poco complicado poder identifcar los intervalos con claridad, es por ello que modificaremos el color de linea de cada intervalo con la propiedad **color =“#ffffff”**. El valor entre comilla representa el color hexadecimal del blanco, pero podemos utilizar los nombres de los colores básicos en inglés (red, blue, yellow, black, white, brown, entre otros).

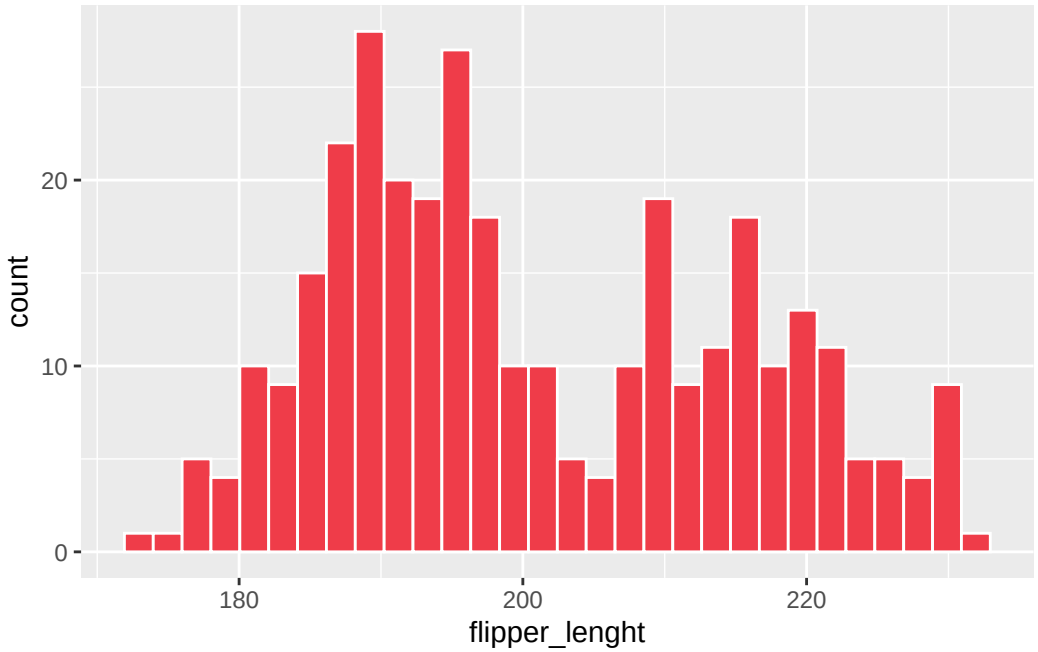
```
# Modificar el color de las lineas
# Bins predeterminados = 30
datos_pinguinos |>
  ggplot(aes(x= flipper_lenght)) +
  geom_histogram(color ="#ffffff", bins = 30)
```



4.3 Modificar color de relleno bins

Para modificar el color del relleno de cada intervalo utilizamos la función **fill**. Los colores se pueden colocar con el *nombre del color* o con el *código hexadecimal*. Colores como nombre Colores hexadecimales

```
# Gráfico de histograma con colores en los bins para identificar
datos_pinguinos |>
  ggplot(aes(x= flipper_lenght)) +
  geom_histogram(color ="#ffffff" ,
                 fill = "#EF3C49",
                 bins = 30)
```

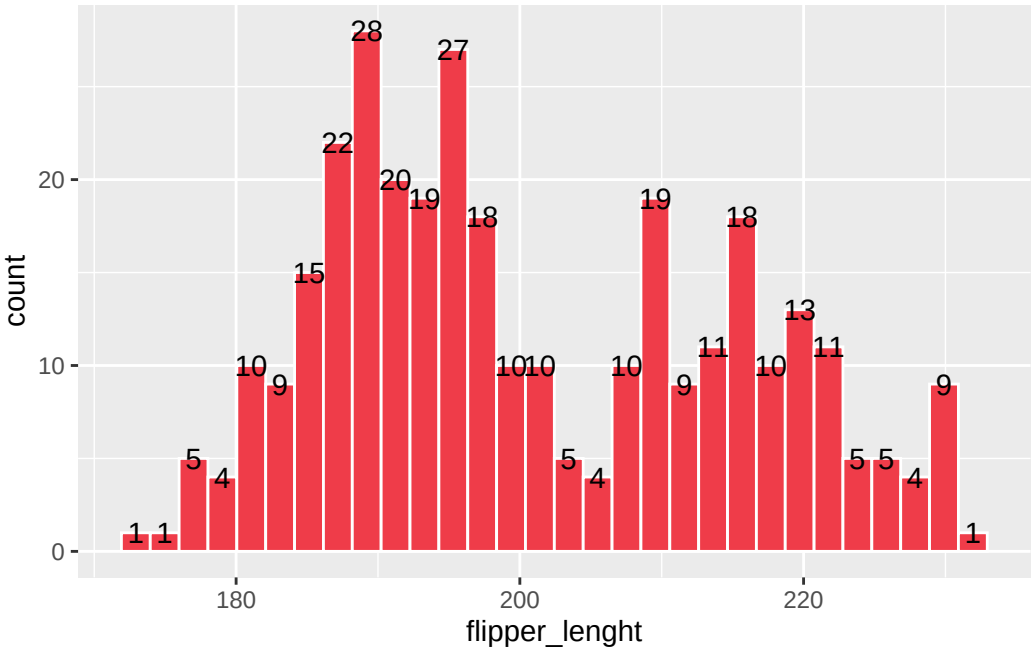


4.4 Añadir datos al gráfico

Para añadir datos del conteo de las observaciones en un histograma, podemos utilizar la función **stat_bin()**, a esta función se le añade las propiedades **label=..count..** , que realiza el conteo por bins, además de la propiedad **geom="text"** que indica que la propiedad es de tipo texto.

```
# Gráfico de histograma con colores en los bins para identificar
datos_pinguinos |>
  ggplot(aes(x= flipper_lenght)) +
  geom_histogram(color ="#ffffff" ,
                 fill = "#EF3C49",
                 bins=30)+
  stat_bin(aes(label=..count..),
           geom="text")
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

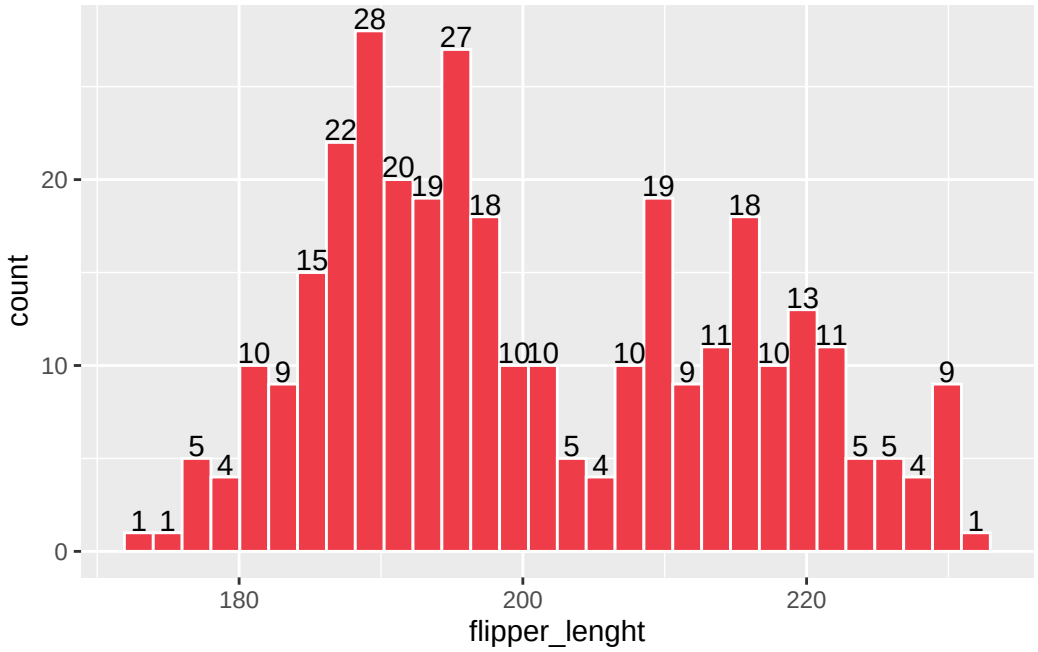


4.5 Ajustar datos en gráfico

Se puede ajustar la posición del texto utilizando las propiedades **vjuts (posición vertical)** y **hjust(posición horizontal)** en la función **stat_bin()** que permiten mover la posición del texto en horizontal y vertical.

```
# Gráfico de histograma con colores en los bins para identificar
datos_pinguinos |>
  ggplot(aes(x= flipper_lenght)) +
  geom_histogram(color ="#ffffff" ,
                 fill = "#EF3C49",
                 bins=30)+
  stat_bin(aes(label=..count..),
           vjust=-0.1,
           hjust= 0.5,
           geom="text")
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

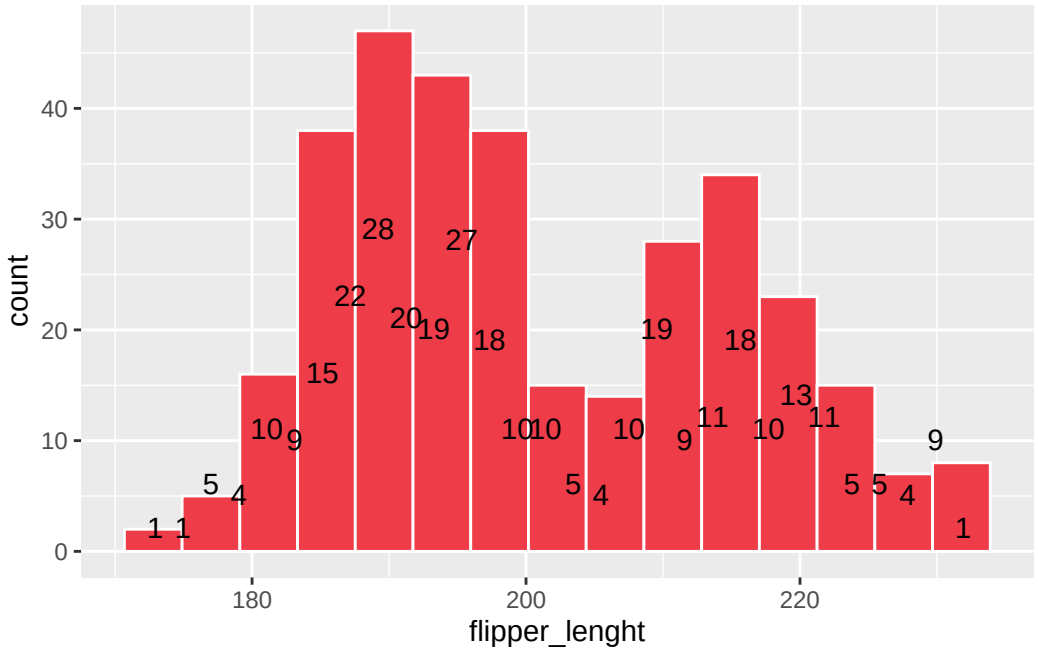


4.6 Datos en gráfico (bins)

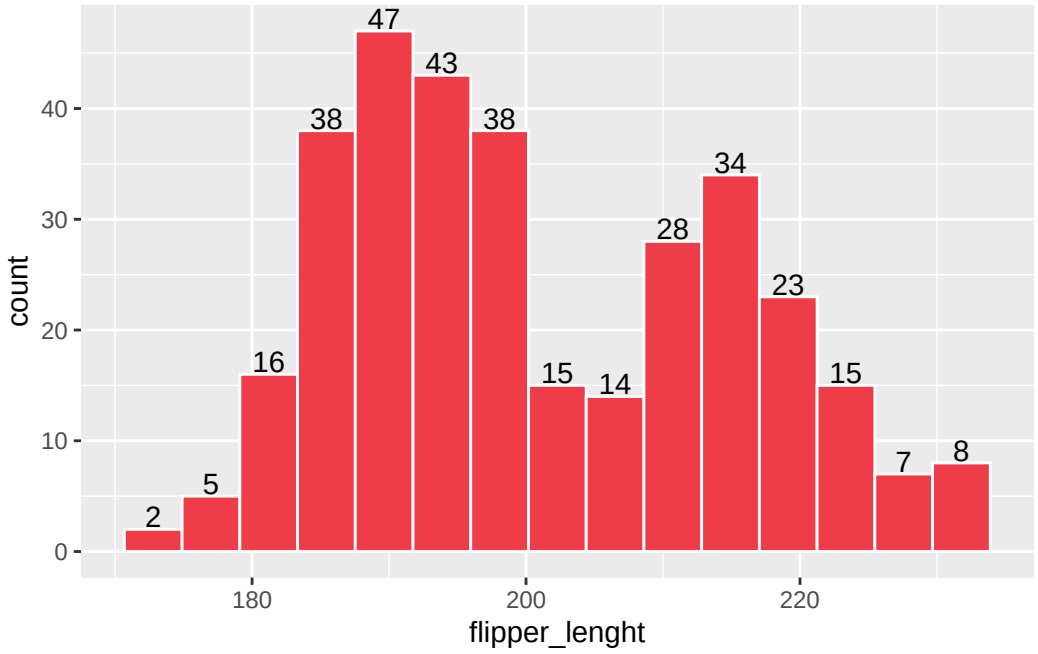
Es importante resaltar que el bins=30 es el valor predeterminado en un histograma, si el valor se modifica como en el ejemplo, este valor se debe tambien modificar en la función **stat_bin()** , sino los datos no coincidiran en el gráfico.

```
# Gráfico de histograma con bins =15
datos_pinguinos |>
  ggplot(aes(x= flipper_lenght)) +
  geom_histogram(color ="#ffffff" ,
                 fill = "#EF3C49",
                 bins=15)+
  stat_bin(aes(label=..count..),
           vjust=-0.1,
           hjust= 0.5,
           geom="text")
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
# Gráfico de histograma con bins =15 en el histograma y en el texto
datos_pinguinos |>
  ggplot(aes(x= flipper_lenght)) +
  geom_histogram(color ="#ffffff" ,
                 fill = "#EF3C49",
                 bins=15)+
  stat_bin(aes(label=..count..),
           vjust=-0.1,
           hjust= 0.5,
           geom="text",
           bins = 15)
```



4.7 Añadir info al gráfico

Para añadir una línea que intercepte o se dibuje en una posición x del gráfico se utiliza la función **geom_vline()**. xintercept = posición de intersección de la línea. color = color de la línea. size = tamaño de la línea.

Valor y tipo de linea,

- linetype = 0 **linetype = "blank"**
- linetype = 1 linetype = "solid"
- linetype = 2 linetype = "dashed"
- linetype = 3 linetype = "dotted"
- linetype = 4 linetype = "dotdash"
- linetype = 5 linetype = "longdash"
- linetype = 6 linetype = "twodash"

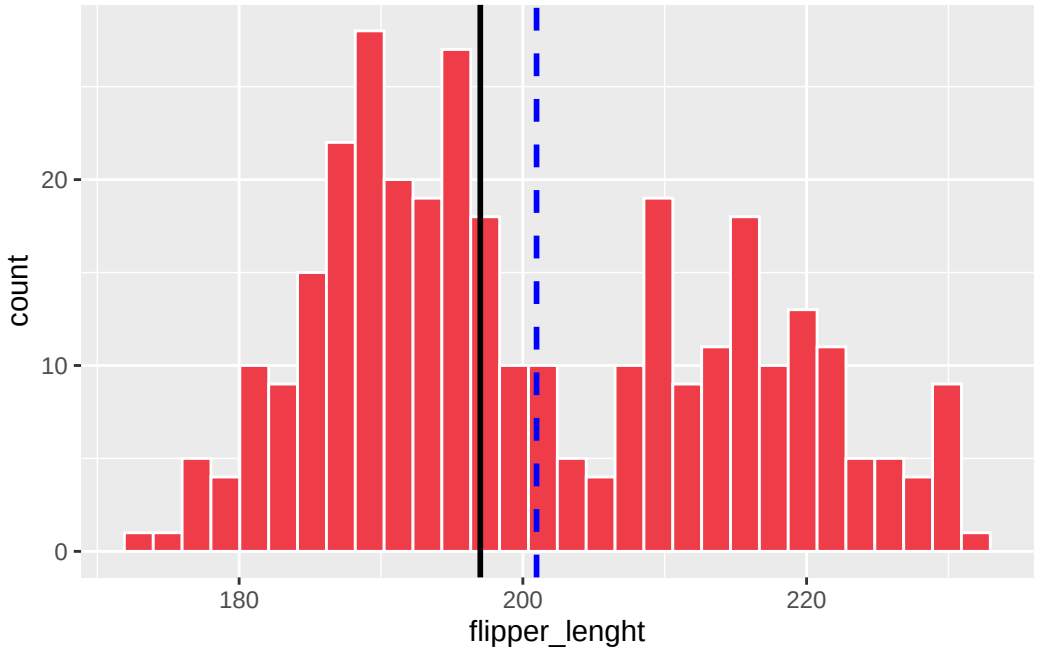
```
#calculo de la media y la mediana
media <- mean(datos_pinguinos$flipper_lenght)
media
```

```
[1] 200.967
```

```
mediana <- median(datos_pinguinos$flipper_lenght)
mediana
```

```
[1] 197
```

```
# Gráfico de histograma con colores en los bins para identificar
datos_pinguinos |>
  ggplot(aes(x= flipper_lenght)) +
  geom_histogram(color ="#ffffff",
                 fill = "#EF3C49",
                 bins=30)+
  geom_vline(xintercept = mediana, size = 1) +
  geom_vline(xintercept = media, size = 1, linetype = "dashed", color="blue")
```

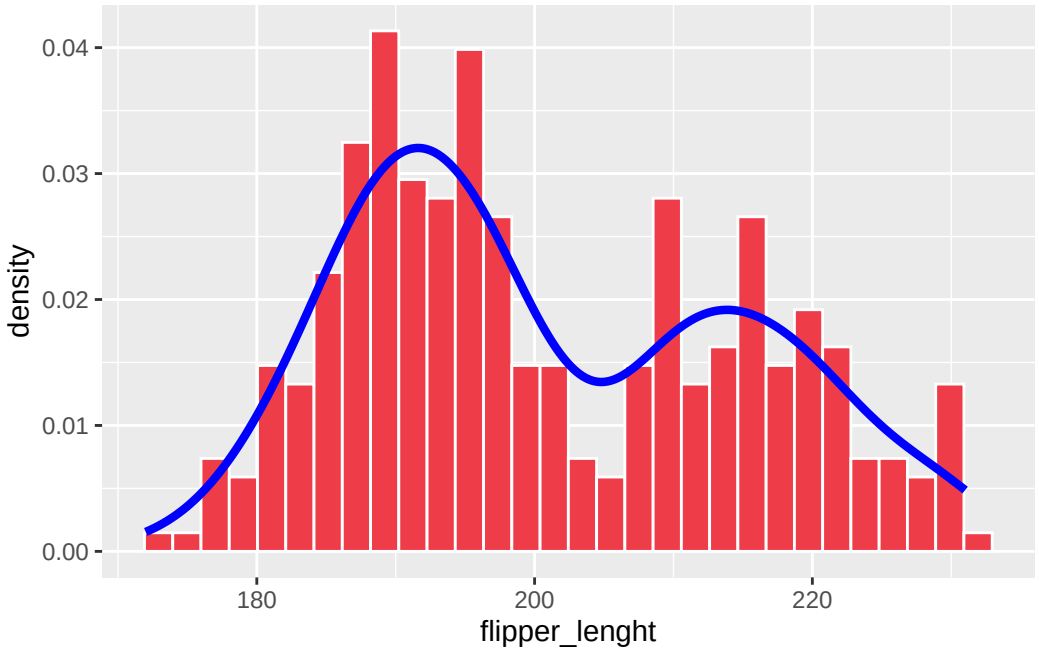



```
# Asimétrica, tiene sesgo a la Derecha (cola a la derecha)
```

4.8 Histograma y densidad

Podemos añadir un gráfico de densidad (poligono) junto al gráfico de histograma utilizando la función **geom_density()** en una nueva capa y la estética en el histograma **aes(y = ..density..)**, recordando que para añadir esta nueva geometria, hay que colocar el signo de + al final del cierre del paréntesis de **geom_histogram** .

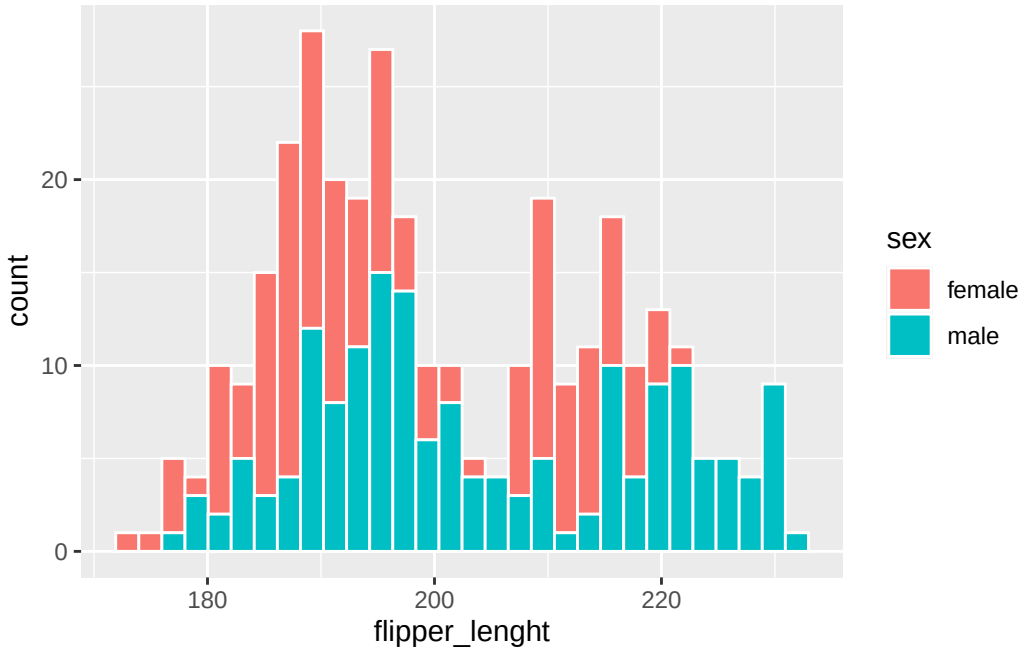
```
# Gráfico de histograma con grafico de densidad
# lwd=1.5 , representa el grosor del borde del
# grafico de densidad
datos_pinguinos |>
  ggplot(aes(x= flipper_lenght)) +
  geom_histogram(aes(y = ..density..),
                 color ="#ffffff",
                 fill = "#EF3C49",
                 bins=30)+
  geom_density(lwd=1.5, colour = "blue")
```



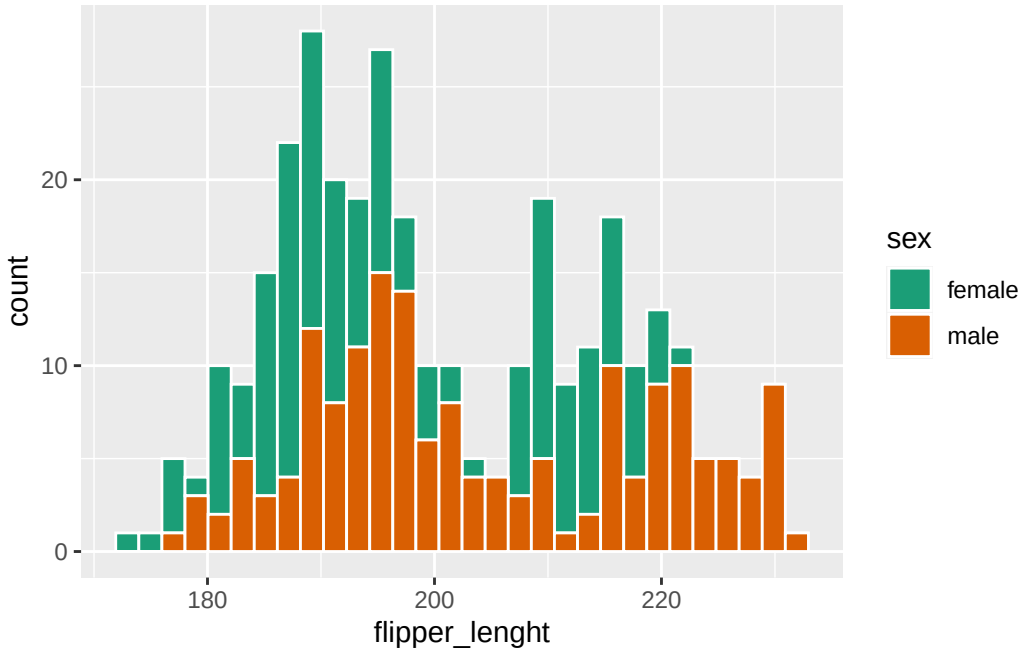
4.9 Histograma con variable categórica

Es posible utilizar una variable **categórica** en un histograma, en este caso la variable **sex** que contiene los valores del sexo de los pinguinos. Esta variable representa una estetica del gráfico, por lo tanto **debe incluirse dentro de la propiedad aes()** de la geometria **geom_histogram()** y la utilizaremos para modificar el color de relleno del histograma. Como la variable **sex** tiene dos categorias, ggplot genera dos histogramas de diferentes colores, uno por cada categoría.

```
# Gráfico de histograma con dos variables
datos_pinguinos |>
  ggplot(aes(x= flipper_lenght)) +
  geom_histogram(aes(fill=sex),
                 color ="#ffffff",
                 bins=30)
```




```
# colores personalizados
# paleta personalizada
colores <- c("#1B9E77", "#D95F02")

datos_pinguinos |>
  ggplot(aes(x= flipper_lenght)) +
  geom_histogram(aes(fill=sex),
                 color ="#ffffff",
                 bins=30)+
  scale_fill_manual(values = colores)
```

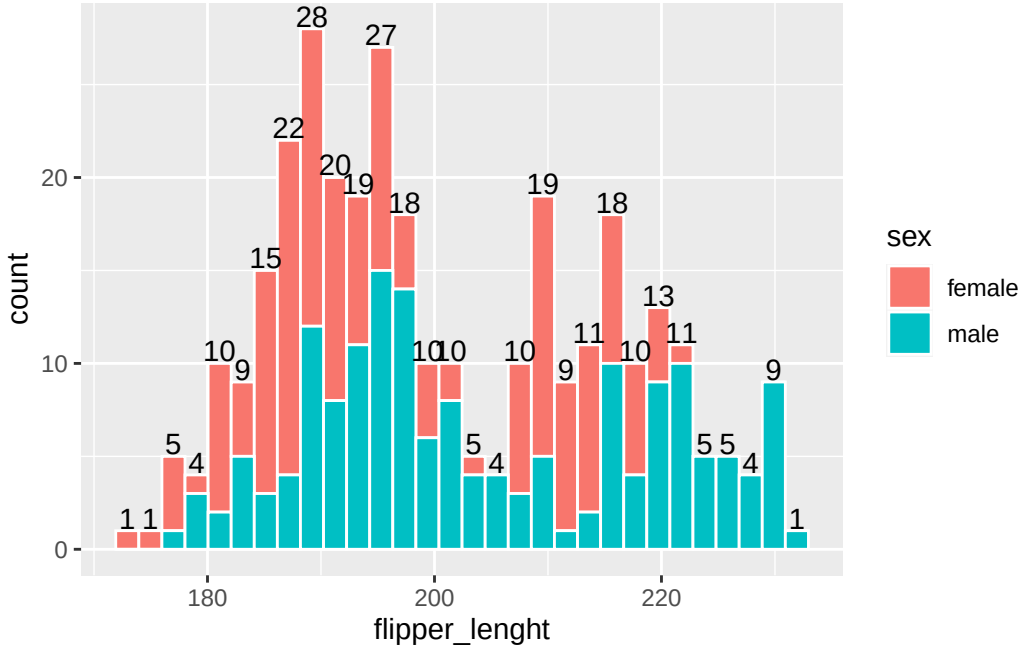


4.9 Añadir datos al histograma

Los códigos para colocar datos al histograma por categoría, es el mismo visto en los ejemplos anteriores donde se muestra texto en el gráfico de histograma..

```
# Gráfico de histograma dos variables
datos_pinguinos |>
  ggplot(aes(x= flipper_lenght)) +
  geom_histogram(aes(fill=sex),
                 color ="#ffffff",
                 bins=30)+
  stat_bin(aes(label=..count..),
           vjust=-0.1,
           hjust= 0.5,
           geom="text")
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

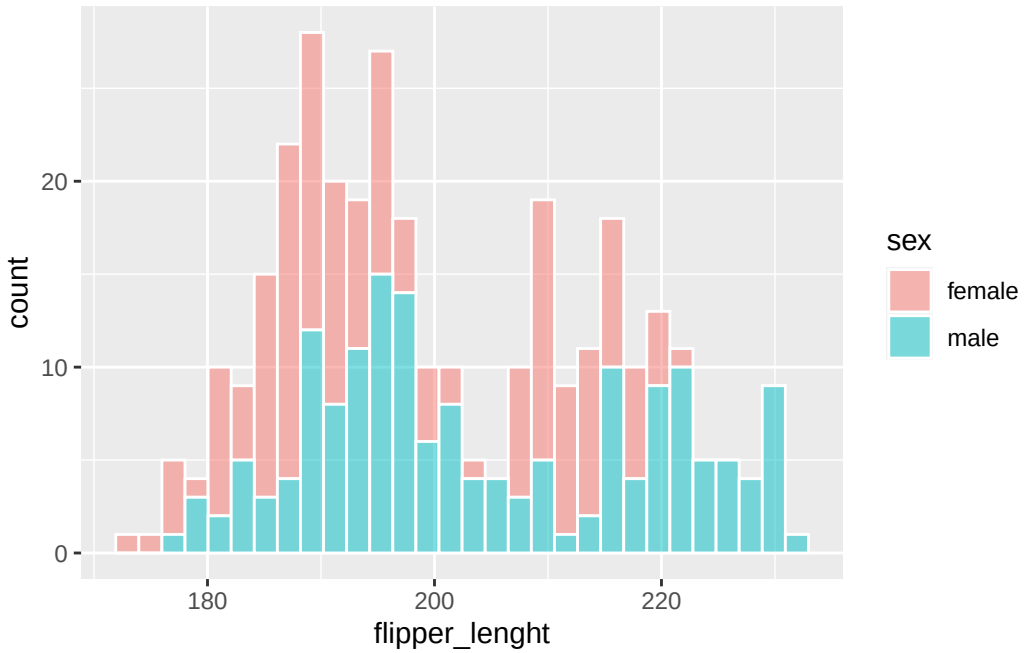


4.9 Histograma con transparencia

Podemos utilizar la propiedad alpha (transparencia) en le geometria **geom_histograma()** con valores de (0 a 1), 1 = 100%. **Para que usar la transparencia.** Cuando creamos este tipo de histogramas la naturaleza de ggplot, es crear los histograma una encima del otro, como si fueran un solo histograma pero separando la distribución de los datos según la variable **sex**, esto no permite comprar realmente la distribución de los datos de ambos histogramas.

```
# Gráfico de histograma con dos variables
datos_pinguinos |>
  ggplot(aes(x= flipper_lenght)) +
  geom_histogram(aes(fill=sex),
                 color ="#ffffff",
```

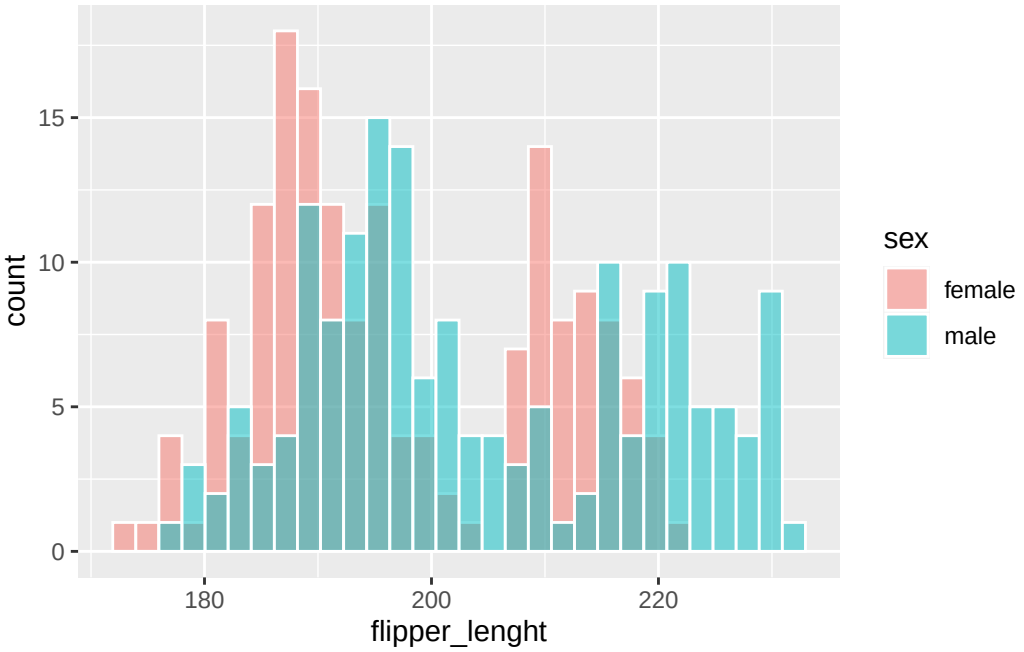
```
                bins=30,
                alpha=0.5)
```



4.9 Histograma con transparencia (position)

Si utilizamso la propiedad **position = "identity"** , en la geometria **geom_histogram()** podemos ver que ahora los histogramas se solapan, ya que uno esta detrás del otro, permitiendo comparar la distribución de los datos.

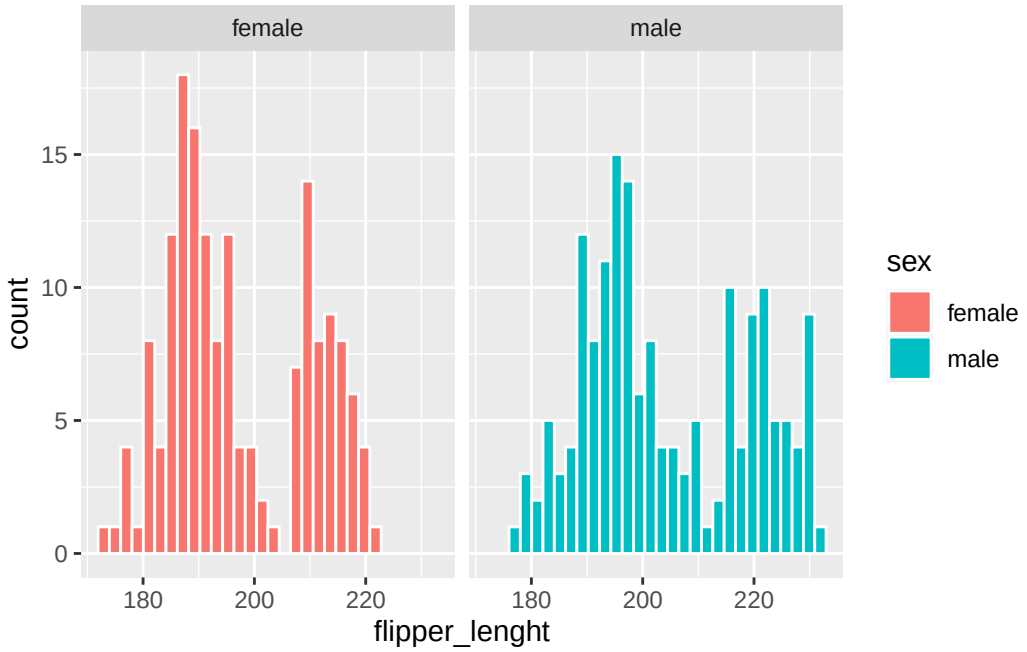
```
# Gráfico de histograma con dos variables
datos_pinguinos |>
  ggplot(aes(x= flipper_lenght)) +
  geom_histogram(aes(fill=sex),
                 color ="#ffffff",
                 bins=30,
                 alpha=0.5,
                 position ="identity")
```



4.10 Facetas

Las facetas son una forma más ordenada de mostrar los gráficos que muestran varias categorias, para ello debemos utilizar la función **facet_grid()** con el nombre de la variable. Los simbolos **.~** antes de la variable **sex** le indica a la faceta que debe colocar los gráficos categorizados en columnas, es decir uno al lado del otro.

```
datos_pinguinos |>
  ggplot(aes(x= flipper_lenght)) +
  geom_histogram(aes(fill=sex),
                 color ="#ffffff",
                 bins=30,
                 position ="identity")+
  facet_grid(.~sex)
```



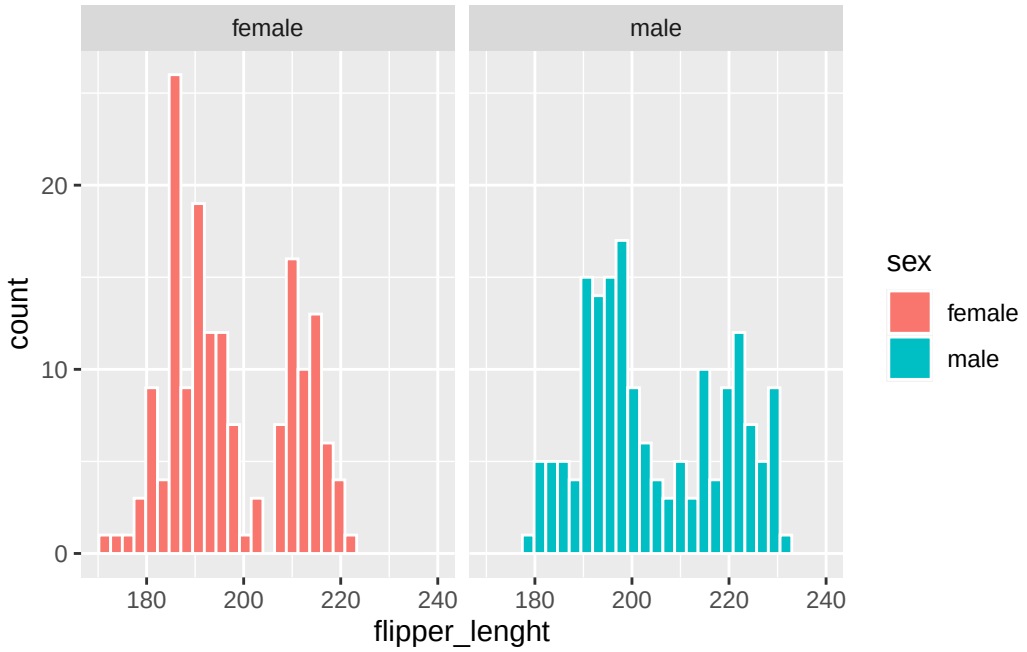
4.11 Recomendaciones (VIZ)

4.11 Recomendación 1

Cuando se utilizan facetas para mostrar distribuciones por grupo, permite comparar las distribuciones por una variable categórica, sin embargo, al separar los datos es importante mirar **el eje x** donde el rango de distribución del sexo **male** es mayor, por lo que es recomendable **mantener los ejes iguales** modificando los valores con la función **xlim()** colocando los rangos de la distribución **xlim(170,240)**.

```
datos_pinguinos |>
  ggplot(aes(x= flipper_lenght)) +
  geom_histogram(aes(fill=sex),
                 color ="#ffffff",
                 bins=30,
                 position ="identity")+
  facet_grid(.~sex)+
  xlim(170,240)
```

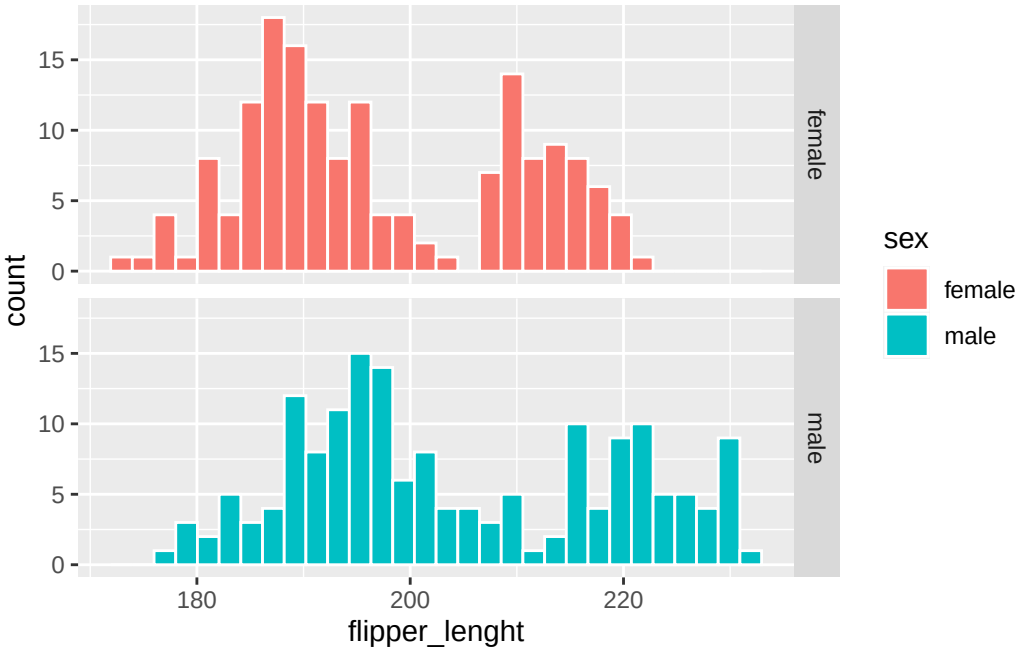
```
Warning: Removed 4 rows containing missing values (`geom_bar()`).
```



4.11 Recomendación 2

Otra opción en el uso de facetas para no modificar el eje x, es usar las facetas en formato de filas. Para generar las facetas en filas, debemos utilizar los simbolos **~.** después de la variable **sex** le indica a la faceta que debe colocar los gráficos categorizados en filas, uno debajo del otro. Esto permite una mejor comparación de los histogramas con las mismas coordenadas en el eje x.

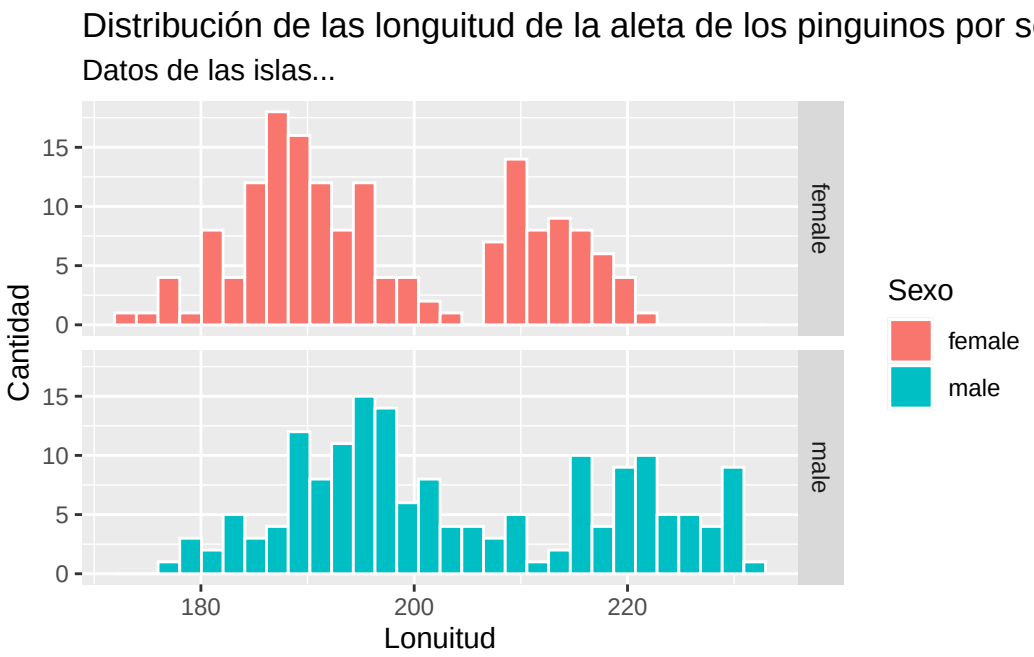
```
datos_pinguinos |>
  ggplot(aes(x= flipper_lenght)) +
  geom_histogram(aes(fill=sex),
                 color ="#ffffff",
                 bins=30,
                 position ="identity")+
  facet_grid(sex~.)
```



4.12 Titulos y nombre de ejes

Podemos añadir titulos al gráfico y modificar la etqiueta de los ejes x, y , como el de la leyenda através de la función **labs()**.

```
datos_pinguinos |>
  ggplot(aes(x= flipper_lenght)) +
  geom_histogram(aes(fill=sex),
                 color ="#ffffff",
                 bins=30,
                 position ="identity")+
  facet_grid(sex~.)+
  labs(title = "Distribución de las longuitud de la aleta de los pinguinos por sexo",
       subtitle = "Datos de las islas...",
       x="Lonuitud",
       y="Cantidad",
       fill="Sexo")
```



4.13 Práctica

Crear gráfico de histograma utilizando los datos **data_penguins_clean.csv** que se encuentran en la carpeta **data**. Al cargar los datos debe filtralos por la especie de **Gentoo**. Utilizar facetas basado en la variable **sex**.

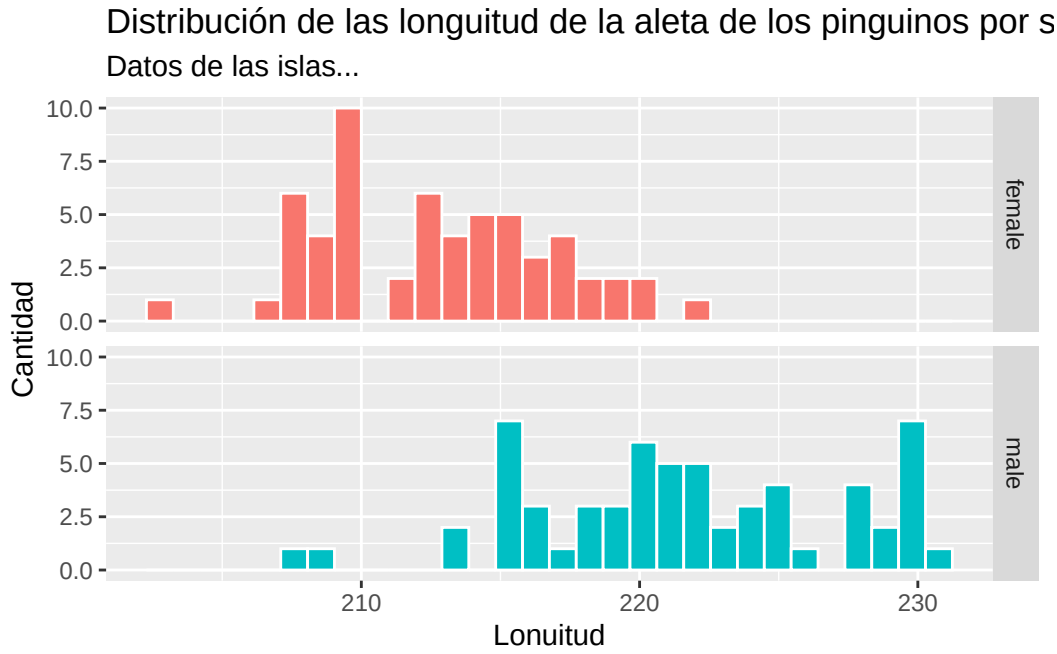
Gráfico de resultado:

```
datos_pinguinos <- read.csv("data/data_penguins_clean.csv")
datos_pinguinos |>
  glimpse()
```

```
Rows: 333
Columns: 8
$ X              <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, ~
$ species        <chr> "Adelie", "Adelie", "Adelie", "Adelie", "Adelie", "Adel~
$ island         <chr> "Torgersen", "Torgersen", "Torgersen", "Torgersen", "To~
$ bill_lenght    <dbl> 39.1, 39.5, 40.3, 36.7, 39.3, 38.9, 39.2, 41.1, 38.6, 3~
$ bill_depth     <dbl> 18.7, 17.4, 18.0, 19.3, 20.6, 17.8, 19.6, 17.6, 21.2, 2~
$ flipper_lenght <int> 181, 186, 195, 193, 190, 181, 195, 182, 191, 198, 185, ~
$ sex            <chr> "male", "female", "female", "female", "male", "female",~
$ body_mass_g    <int> 3750, 3800, 3250, 3450, 3650, 3625, 4675, 3200, 3800, 4~
```

```
datos_pinguinos$species <- as.factor(datos_pinguinos$species)
datos_pinguinos$island <- as.factor(datos_pinguinos$island)
datos_pinguinos$sex <- as.factor(datos_pinguinos$sex)


datos_pinguinos |>
  filter(species=="Gentoo") |>
  ggplot(aes(x= flipper_lenght)) +
```

```
  geom_histogram(aes(fill=sex),
                 color ="#ffffff",
                 bins=30,
                 position ="identity")+
  facet_grid(sex~.)+
  labs(title = "Distribución de las longuitud de la aleta de los pinguinos por sexo",
       subtitle = "Datos de las islas...",
       x="Lonuitud",
       y="Cantidad",
       fill="Sexo")+
  theme(legend.position = "none")
```



5Gráfico de boxplot

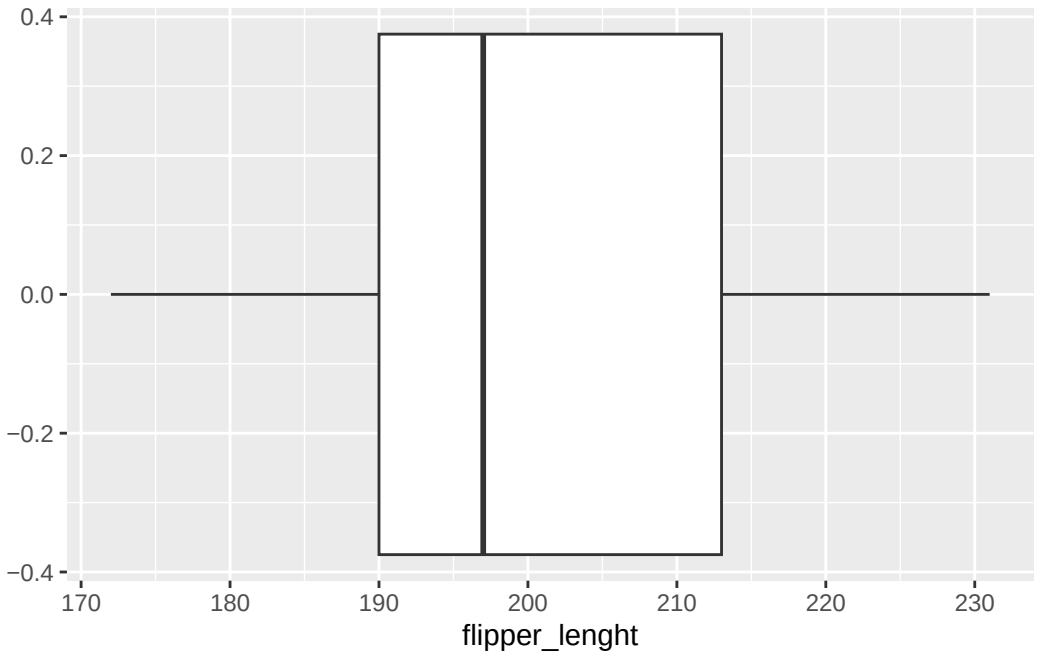
El Diagrama de Caja y bigotes (box and whisker plot en inglés) es un tipo de gráfico que muestra un resumen de la cantidad de datos en cinco medidas descriptivas, además de intuir su morfología y simetría.

Este tipo de gráficos nos permite identificar valores atípicos y comparar distribuciones de una variable. Además de conocer de una forma cómoda y rápida como el 50% de los valores centrales se distribuyen, para ello utilizamos la geometría **geom_boxplot()**.

Se pueden detectar los siguientes valores en el gráfico:

- Primer cuartil: representa el 25% de los datos.
- Tercer cuartil: representa el 75% de los datos acumulados..
- Mediana o Segundo Cuartil: Divide en dos partes iguales la distribución, acumula el 50% de los datos.
- Rango Intercuartílico (RIC): representa el 50% intermedio de los datos. Muestra la distancia entre el primer cuartil y el tercer cuartil (Q3-Q1).
- Valores mínimo y máximo.

```
datos_pinguinos |>
  ggplot(aes(x= flipper_lenght)) +
  geom_boxplot()
```




```
# valores de un boxplot
summary(datos_pinguinos$flipper_lenght)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    172     190     197     201     213     231
```

```
boxplot.stats(datos_pinguinos$flipper_lenght)
```

```
$stats
[1] 172 190 197 213 231

$n
[1] 333

$conf
[1] 195.0086 198.9914

$out
integer(0)
```

```
# REsumen de valores centrales y de variacion
library(summarytools)
```

```
Attaching package: 'summarytools'

The following object is masked from 'package:tibble':

    view
```

```
summarytools::descr(datos_pinguinos$flipper_lenght)
```

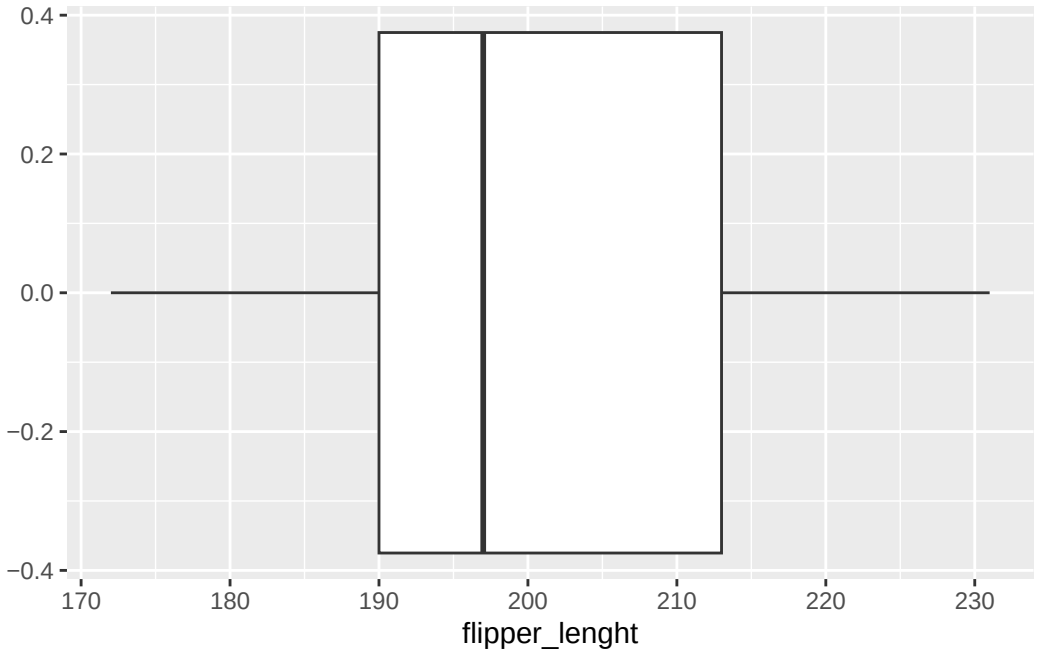
```
Descriptive Statistics
datos_pinguinos$flipper_lenght
N: 333

                     flipper_lenght
---------------- ----------------
            Mean           200.97
         Std.Dev            14.02
             Min           172.00
              Q1           190.00
          Median           197.00
              Q3           213.00
             Max           231.00
             MAD            16.31
             IQR            23.00
              CV             0.07
        Skewness             0.36
     SE.Skewness             0.13
        Kurtosis            -0.98
         N.Valid           333.00
       Pct.Valid           100.00
```

```
#Mostrar datos de datos con gráficos
# view(dfSummary(datos_pinguinos[,-1]))
```

5.1 Valores atipicos

Son observaciones cuyos valores son muy diferentes a las de otras observaciones del mismo grupo de datos. Los datos atípicos son ocasionados por: errores de procedimiento, acontecimientos extraordinarios, valores extremos, estos valores pueden distorsionar los resultados del análisis de datos.

```
#demo
library()
#Extarer datos de prueba
valoresA <-  datos_pinguinos |>
            slice_sample(n=3)
# sumarle 150 a los datos
valoresA$flipper_lenght <- valoresA$flipper_lenght + 80

# unir datos
datos_pinguinos_ <- rbind(datos_pinguinos,valoresA)

#mostrar gráfico
datos_pinguinos |>
  ggplot(aes(x= flipper_lenght)) +
  geom_boxplot()
```




```
boxplot.stats(datos_pinguinos_$flipper_lenght)
```

```
$stats
[1] 172.0 190.0 197.0 213.5 231.0

$n
[1] 336

$conf
[1] 194.9744 199.0256
```

```
$out
[1] 275 273 292
```

```
# $stats
# [1] 172 190 197 213 231

#194.9744 199.0256

(rango <- 231-172)
```

```
[1] 59
```

```
datos_pinguinos |>
  glimpse()
```

```
Rows: 333
Columns: 8
$ X              <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, ~
$ species        <fct> Adelie, Adelie, Adelie, Adelie, Adelie, Adelie, Adelie,~
$ island         <fct> Torgersen, Torgersen, Torgersen, Torgersen, Torgersen, ~
$ bill_lenght    <dbl> 39.1, 39.5, 40.3, 36.7, 39.3, 38.9, 39.2, 41.1, 38.6, 3~
$ bill_depth     <dbl> 18.7, 17.4, 18.0, 19.3, 20.6, 17.8, 19.6, 17.6, 21.2, 2~
$ flipper_lenght <int> 181, 186, 195, 193, 190, 181, 195, 182, 191, 198, 185, ~
$ sex            <fct> male, female, female, female, male, female, male, femal~
$ body_mass_g    <int> 3750, 3800, 3250, 3450, 3650, 3625, 4675, 3200, 3800, 4~
```

```
#desviavion estandar
sd(datos_pinguinos_$flipper_lenght)
```

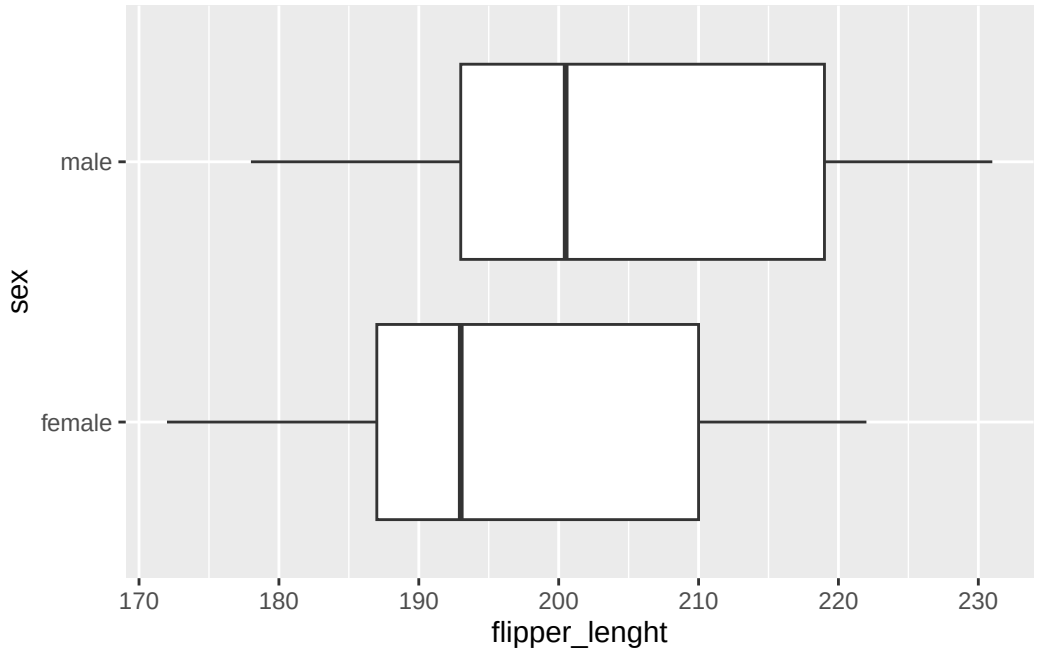
```
[1] 15.83572
```

```
datos_pinguinos$sex <- as.factor(datos_pinguinos$sex)

datos_pinguinos |>
  group_by(sex) |>
  summarise(media = mean(flipper_lenght),
            mediana = median(flipper_lenght))
```

```
# A tibble: 2 x 3
  sex    media mediana
  <fct>  <dbl>   <dbl>
1 female  197.     193
2 male    205.     200.
```

```
datos_pinguinos |>
  ggplot(aes(y= sex, x= flipper_lenght)) +
```

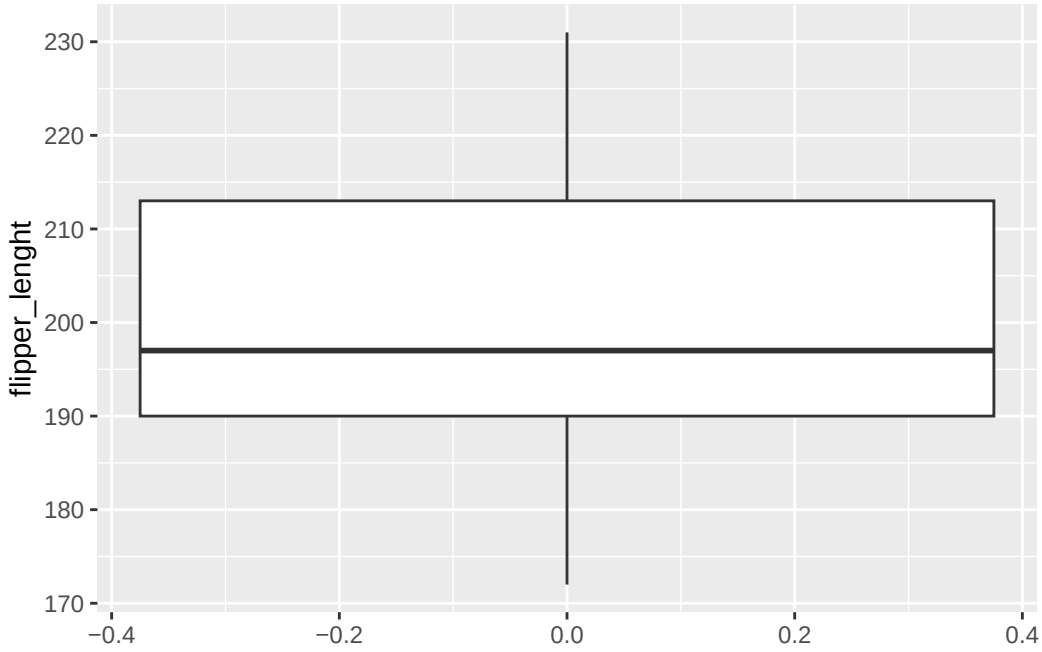
```
  geom_boxplot()
```



5.2 Boxplot vertical

Para girar el gráfico de boxplot en ggplot, solo debemos cambiar la asignación de la varible **x= flipper_lenght** a **y= flipper_lenght** en la estética **aes()**.

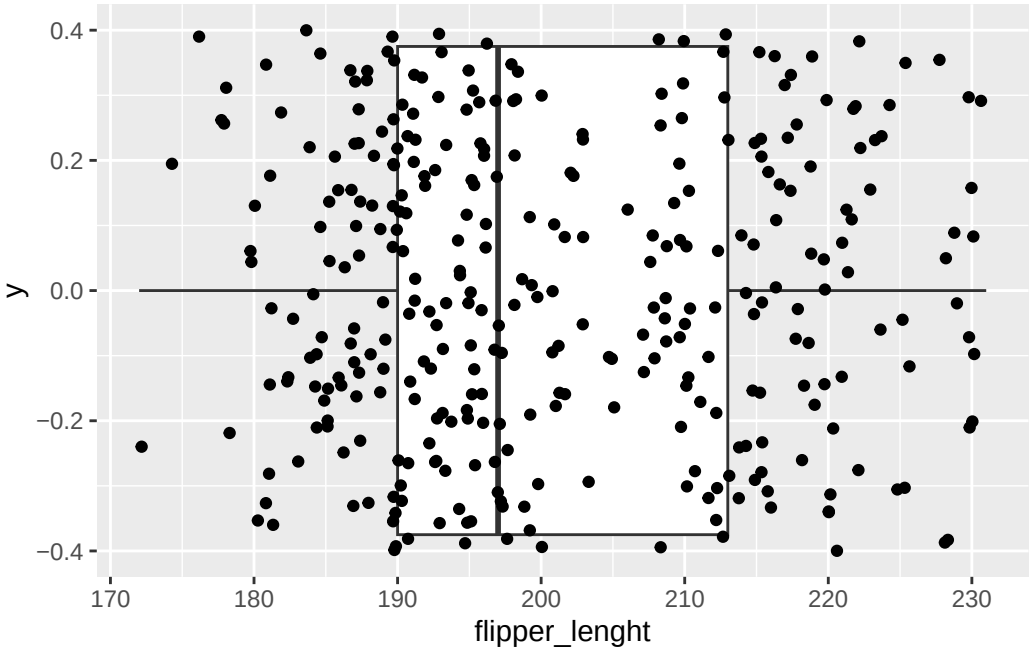
```
datos_pinguinos |>
  ggplot(aes(y= flipper_lenght)) +
  geom_boxplot()
```



5.3 Boxplot con distribución de puntos

Para añadir los puntos de distribución en el gráfico del boxplot debemos hacer dos pasos: 1. Añadir el eje y=0 en el **aes()** 2. Utrilizar le geometría **geom_jitter()**.

```
datos_pinguinos |>
  ggplot(aes(x= flipper_lenght, y=0)) +
  geom_boxplot()+
  geom_jitter()
```

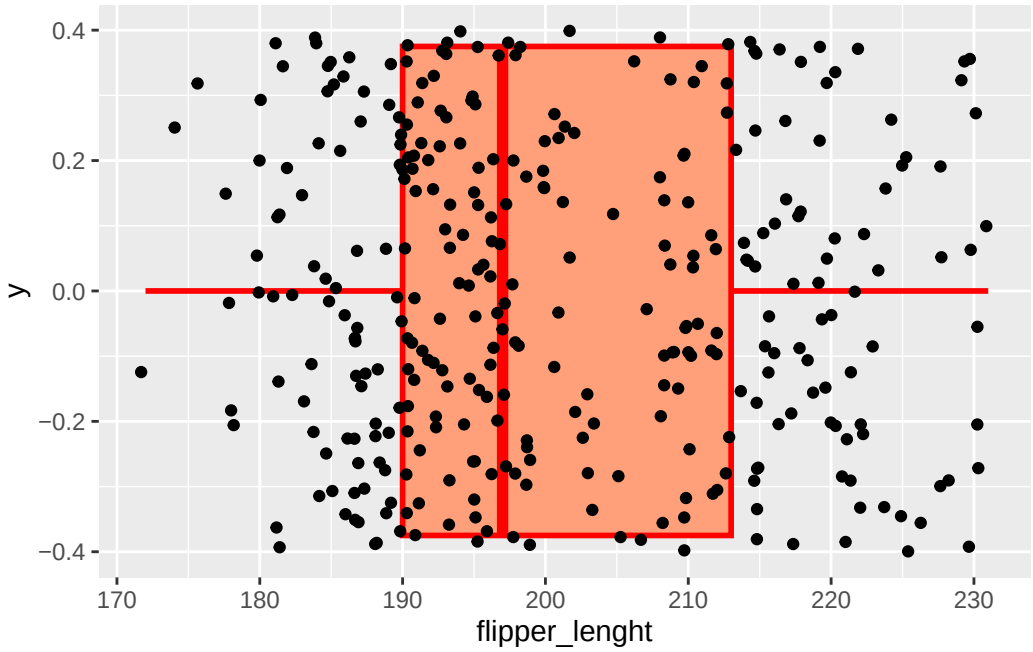


5.4 Boxplot con colores

Para modificar la apariencia del gráfico de boxplot, debemos de modificar las propiedades en la geomatría **geom_boxplot()**.

- *color = color de la línea del boxplot*
- fill = color de la caja, si utilizamos , fill="transparente" la caja es trasnparente.
- size = grosor de la linea del boxplot.

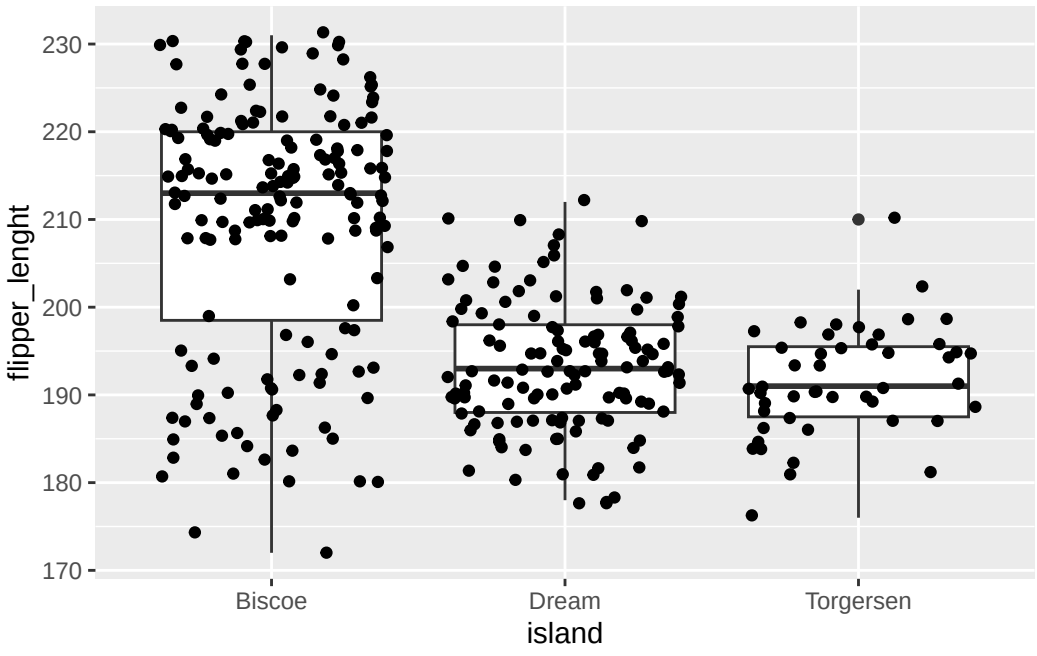
```
datos_pinguinos |>
  ggplot(aes(x= flipper_lenght, y=0)) +
  geom_boxplot(color = "red",
               fill = "lightsalmon",
               size = 1)+
  geom_jitter()
```



5.5 Boxplot con dos variables

Se queremos utilizar dos variables para comparar su distribución, debemos utilizar una variable categórica, en este caso la variable **island** en el valor de **X o Y,** en función de si queremos mostrar el boxplot horizontal o verical.

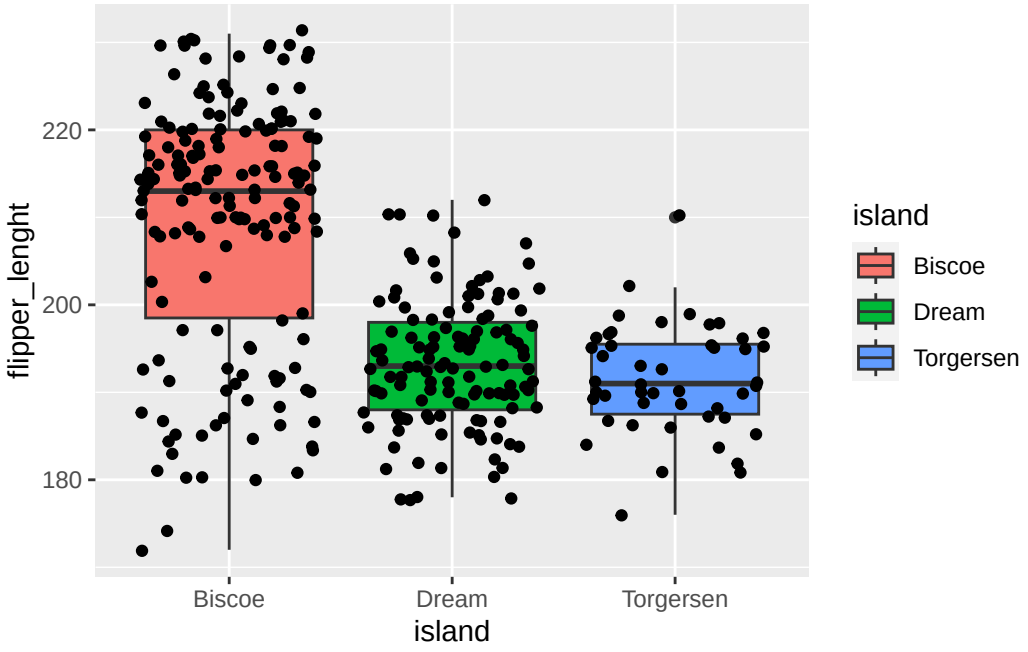
```
datos_pinguinos |>
  ggplot(aes(y= flipper_lenght, x=island)) +
  geom_boxplot()+
  geom_jitter()
```



5.6 Boxplot con dos variables - colores

Se queremos darle un color diferente a cada boxplot, colocamos en la geoametría del gráfico la propiedad **fill=variable categórica**, con ello se creará un color para cada boxplot.

```
datos_pinguinos |>
  ggplot(aes(y= flipper_lenght, x=island)) +
  geom_boxplot(aes(fill=island))+
  geom_jitter()
```

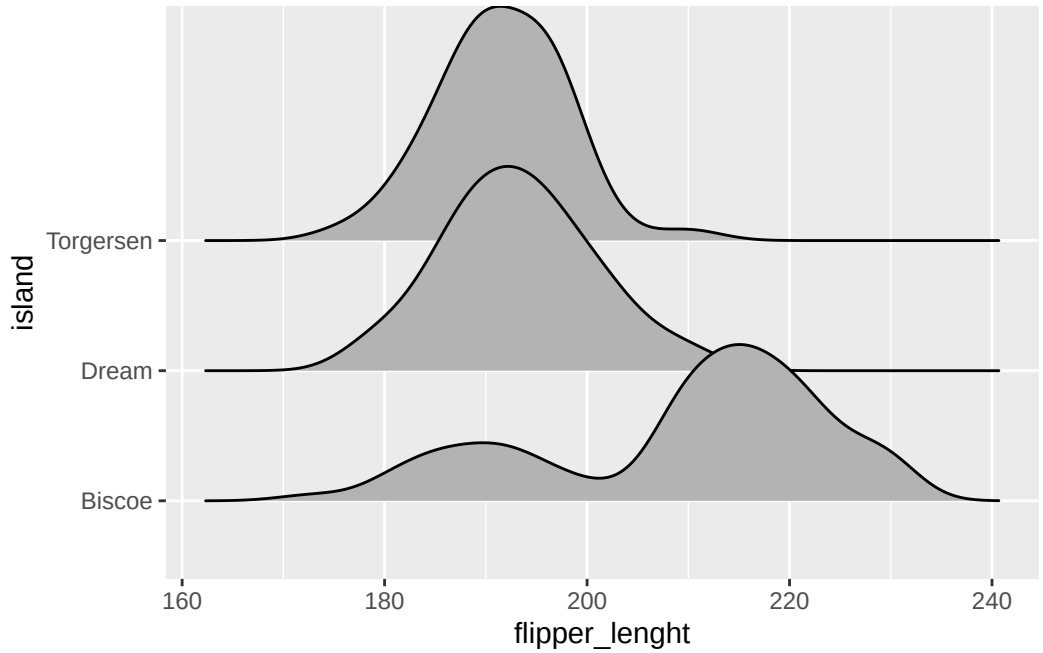


6Gráfico Ridge

Los diagramas de crestas **ridges** son diagramas de líneas parcialmente superpuestas que crean la impresión de una cadena montañosa. Pueden resultar muy útiles para visualizar cambios en las distribuciones a lo largo del tiempo o el espacio. Para su creación es necesario utilizar la geometria **geom_density_ridges()** que se encuentra en la libreria **ggridges**. A diferencia del diagrama de densidad e histograma, este tipo de gráfico necesita que se identifique los valores de **X y Y** en la estética **aes()** de ggplot.

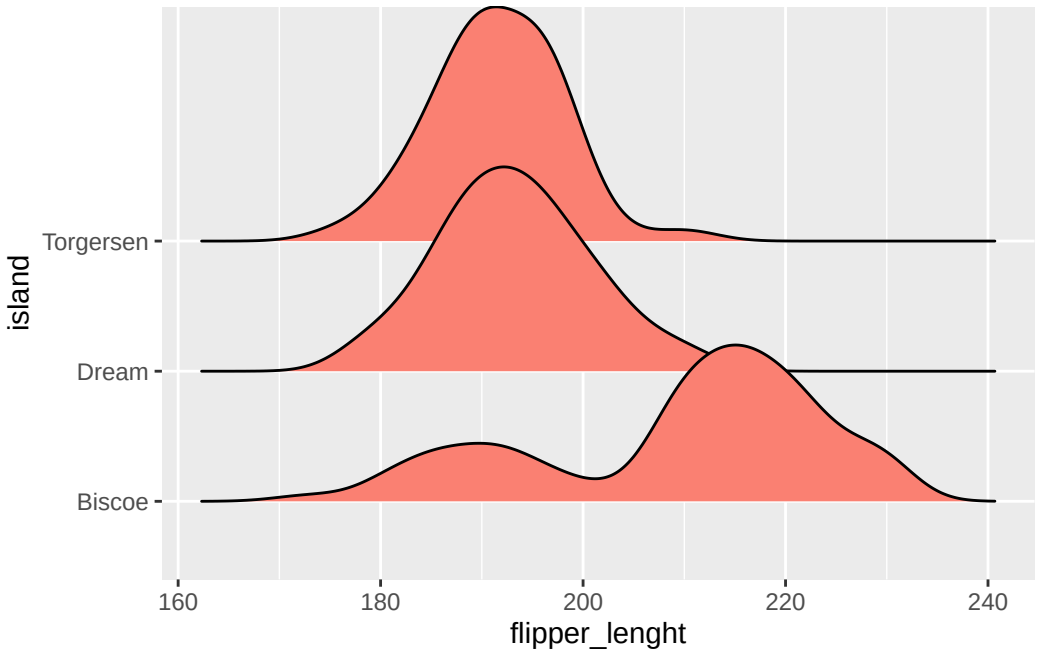
```
# Gráfico Ridge son color
datos_pinguinos |>
  ggplot(aes(x= flipper_lenght,
         y=island)) +
   geom_density_ridges()
```

```
Picking joint bandwidth of 3.23
```




```
# Gráfico Ridge  color relleno
datos_pinguinos |>
  ggplot(aes(x= flipper_lenght,
         y=island)) +
   geom_density_ridges(fill="salmon")
```

```
Picking joint bandwidth of 3.23
```

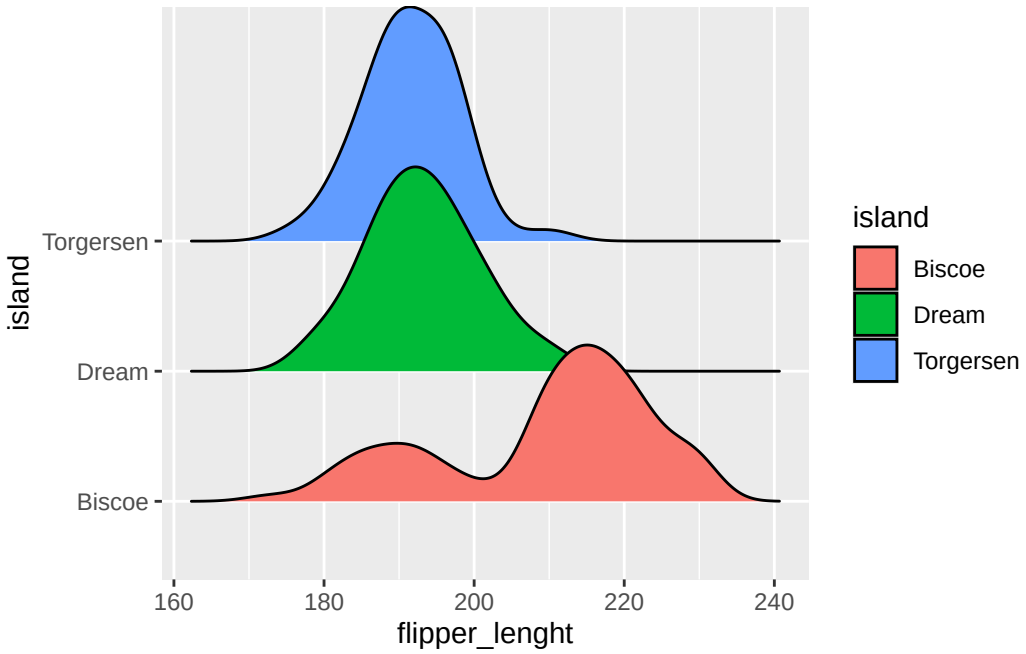


6.1 Gráfico Ridge - color

Para modificar el color del relleno en los gráficos tipo Ridge utilizamos la porpiedad **fill** y la variable **island** que indica el númeor de categorias, estas deben estar integradas en la estética aes() de la geoametría **geom_density_ridges**.

```
# dos variable en el color de la lineas
datos_pinguinos |>
  ggplot(aes(x= flipper_lenght,
         y=island)) +
   geom_density_ridges(aes(fill=island))
```

```
Picking joint bandwidth of 3.23
```

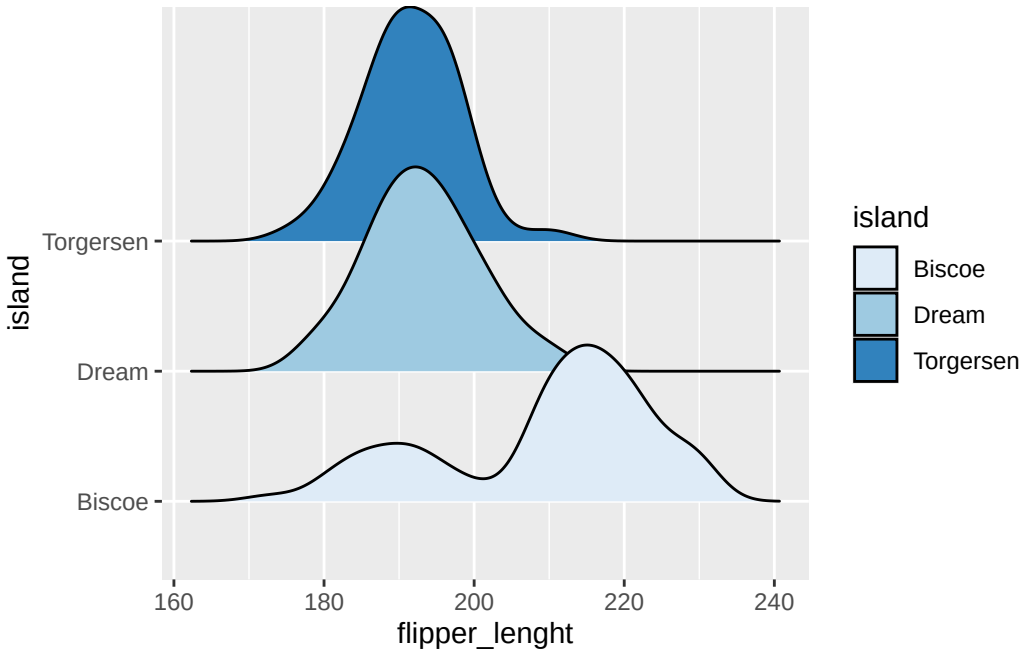


6.2 Gráfico Ridge - color palette

Podemos modificar el relleno en los gráficos tipo Ridge con colores diferentes a los colores predeterminados utilizando **scale_fill_brewer** y algún tipo de paleta de colores en ggplot. En otro de los temas veremos el uso de estas paletas de colores.

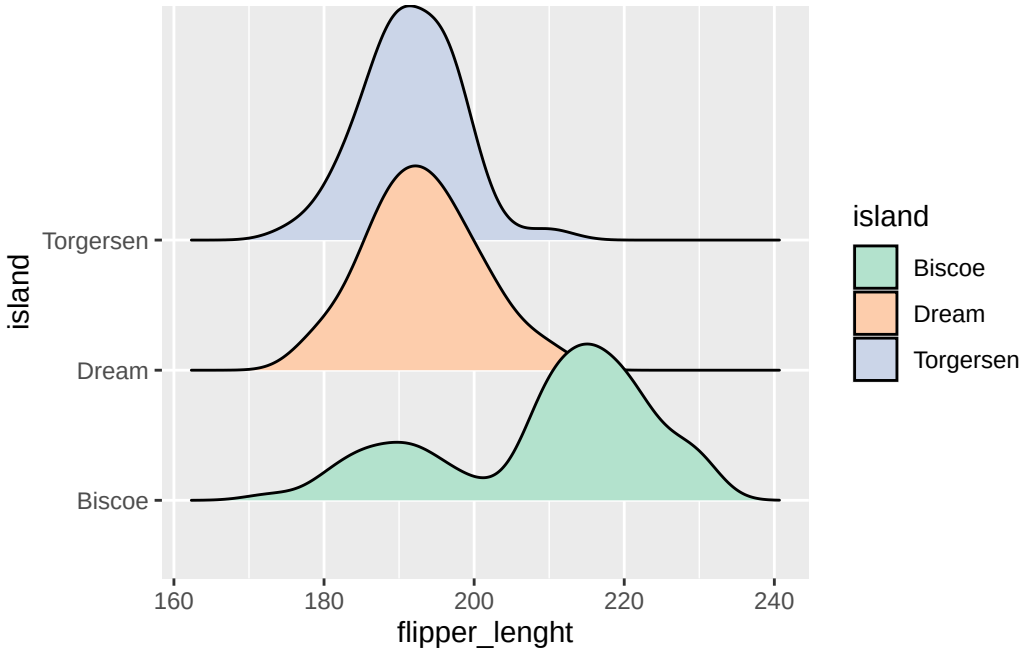
```
# colores con paleta de colores 1
datos_pinguinos |>
  ggplot(aes(x= flipper_lenght,
         y=island)) +
   geom_density_ridges(aes(fill=island))+
  scale_fill_brewer(palette = "Blues")
```

```
Picking joint bandwidth of 3.23
```




```
# colores con paleta de colores 2
datos_pinguinos |>
  ggplot(aes(x= flipper_lenght,
         y=island)) +
   geom_density_ridges(aes(fill=island))+
  scale_fill_brewer(palette = "Pastel2")
```

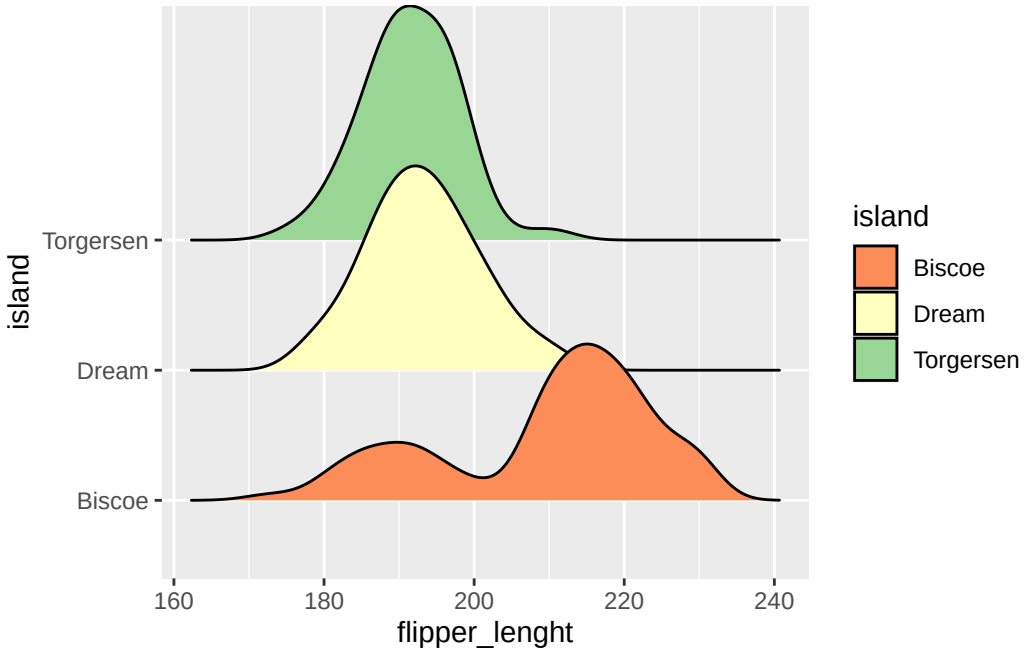
```
Picking joint bandwidth of 3.23
```




```
# colores con paleta de colores 8
datos_pinguinos |>
```

```
    ggplot(aes(x= flipper_lenght,
           y=island,
           fill=island)) +
     geom_density_ridges()+
    scale_fill_brewer(palette = "Spectral")
```

```
Picking joint bandwidth of 3.23
```



7Gráfico de Barra

7.1 Cargar librerias

Librerias necesarias que se deben haber instalado previamente con **install.packages(nombre del paquete)** o en el panel **Package**.

```
library(tidyverse) # incluye ggplot2
library(readxl) # funciones para importar xlsx
library(janitor) # funciones de limpieza
library(patchwork) #combinar gráficos de ggplot
library(ggExtra)
library(ggthemes) # nuevas temas para los
library(plotly) #gráficos interactivos # remotes::install_github("plotly/plotly")
library(tibble)
library(skimr) # reseumen numerico
library(modeest)
library(ggrepel) # añadir etiquetas a los gráficos
library(RColorBrewer) #paletas de colores
library(plotly) #graficos interactivos
```

7.2 Carga de datos

```
datos_pinguinos_clean <- read.csv("data/data_penguins_clean.csv")
datos_pinguinos_clean <- as_tibble(datos_pinguinos_clean)
datos_pinguinos_clean<- datos_pinguinos_clean[,-1]
head(datos_pinguinos_clean,5)
```

```
# A tibble: 5 x 7
  species island    bill_lenght bill_depth flipper_lenght sex    body_mass_g
  <chr>   <chr>           <dbl>      <dbl>          <int> <chr>        <int>
1 Adelie  Torgersen        39.1       18.7            181 male          3750
2 Adelie  Torgersen        39.5       17.4            186 female        3800
3 Adelie  Torgersen        40.3       18              195 female        3250
4 Adelie  Torgersen        36.7       19.3            193 female        3450
5 Adelie  Torgersen        39.3       20.6            190 male          3650
```

```
# Transformar variables de tipo caracter a factor
datos_pinguinos_clean$species <- as.factor(datos_pinguinos_clean$species)
datos_pinguinos_clean$island<- as.factor(datos_pinguinos_clean$island)
datos_pinguinos_clean$sex<- as.factor(datos_pinguinos_clean$sex)

# Verificar tipos de datos
datos_pinguinos_clean |>
  glimpse()
```

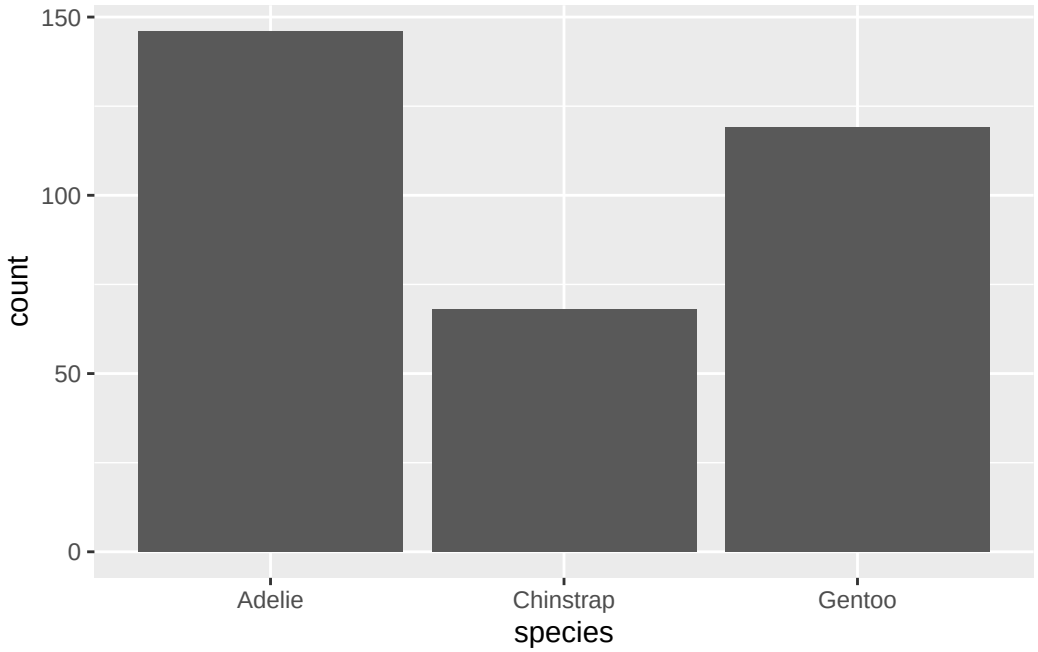
```
Rows: 333
Columns: 7
$ species        <fct> Adelie, Adelie, Adelie, Adelie, Adelie, Adelie, Adelie,~
$ island         <fct> Torgersen, Torgersen, Torgersen, Torgersen, Torgersen, ~
$ bill_lenght    <dbl> 39.1, 39.5, 40.3, 36.7, 39.3, 38.9, 39.2, 41.1, 38.6, 3~
$ bill_depth     <dbl> 18.7, 17.4, 18.0, 19.3, 20.6, 17.8, 19.6, 17.6, 21.2, 2~
$ flipper_lenght <int> 181, 186, 195, 193, 190, 181, 195, 182, 191, 198, 185, ~
$ sex            <fct> male, female, female, female, male, female, male, femal~
$ body_mass_g    <int> 3750, 3800, 3250, 3450, 3650, 3625, 4675, 3200, 3800, 4~
```

7.3 Gráfico de barra en R

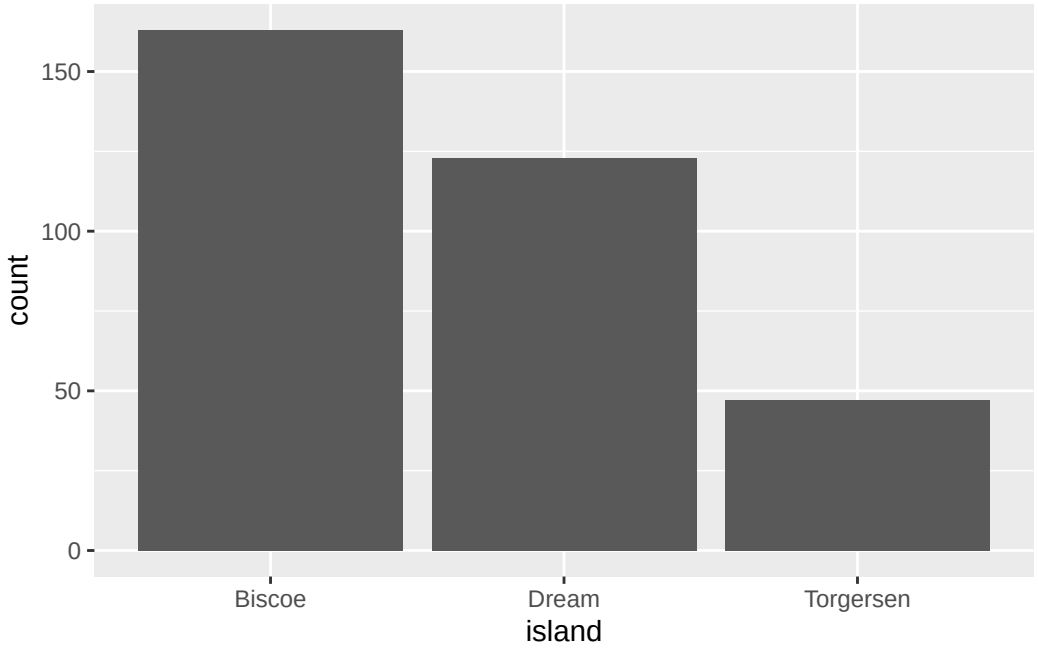
Conocido como gráfico de barras o gráfico de columnas, muestra las frecuencias absolutas y relativas en un sistema de coordenadas de dos ejes.Es utilizado para mostrar comparaciones numéricas discretas entre categorías, por lo tanto se utiliza como base una variable categórica. Un eje del gráfico muestra las categorías específicas que se están comparando y el otro eje representa una escala de valores discretos. Hay que mencionar que aunque utilizamos como sinónimo los gráficos de barras y columnas, un gráfico de columna es un gráfico con barras verticales y un gráfico de barras son gráficos en posición horizontal.

Para crear un gráfico de barra en ggplot utilizamos la geometría **geom_bar()** donde la variable en la estética **aes()** debe ser una variable categórica, como en este caso las variables **species, island, sex** del tibble **datos_pinguinos_clean**. Lo que hace ggplot, es agrupar el número de observaciones por cada categoría y el valor calculado corresponde al ancho o alto de la barra.

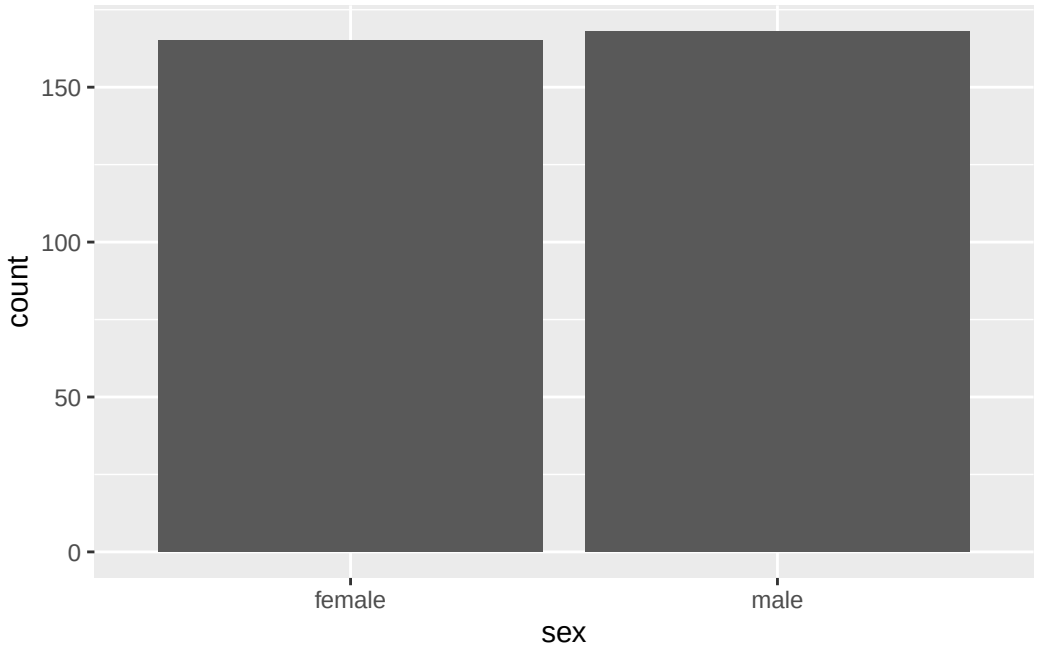
```
#Gráfico de barras de numero de observaciones por especies
datos_pinguinos_clean |>
  ggplot(aes(x=species)) +
  geom_bar()
```



```
#Gráfico de barras de numero de observaciones por isla
datos_pinguinos_clean |>
  ggplot(aes(x=island)) +
  geom_bar()
```



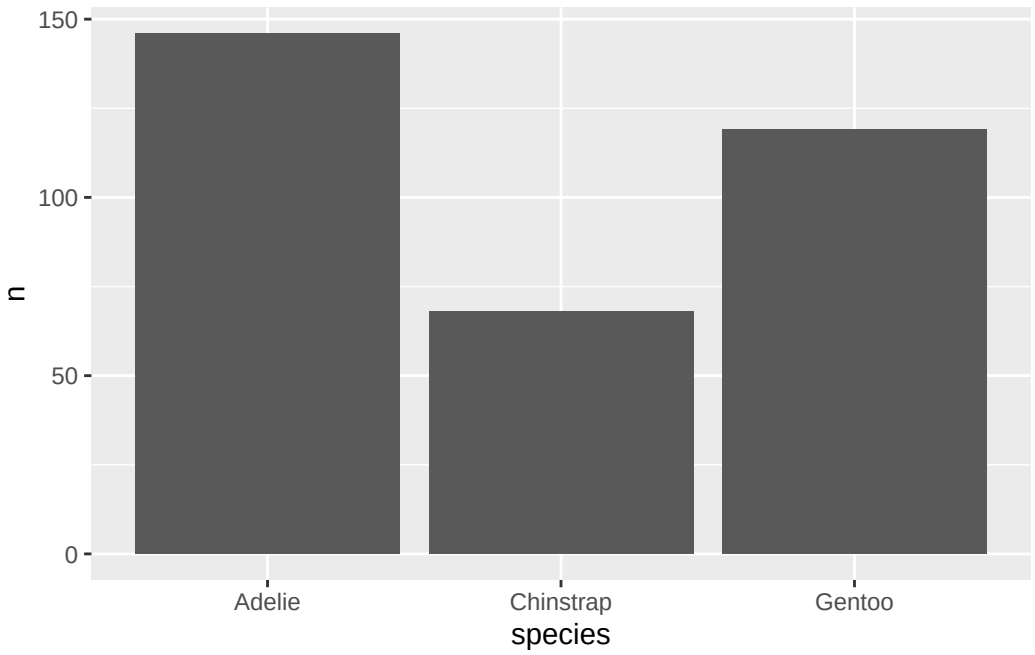

```
#Gráfico de barras de numero de observaciones por sexo
datos_pinguinos_clean |>
  ggplot(aes(x=sex)) +
  geom_bar()
```



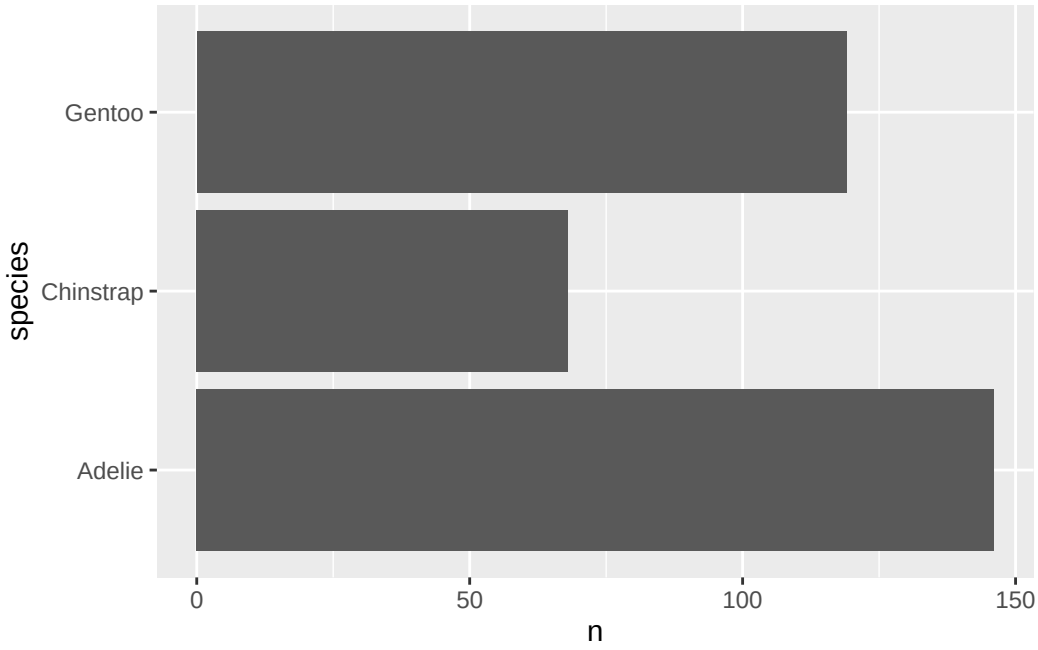
7.4 Gráfico de columnas

Ggplot tambien utiliza otra geometría para crear gráficos de columnas, la cual es **geom_col()**, sin embargo, su funcionamiento es un poco diferente ya que esta utiliza un valor categórico y un valor numérico para crear las columnas el cual debe ser calculado. Este gráfico tambien permite transformar un gráfico de barras o más bien girar el gráfico de columnas.

```
# Crear datos de conteo de observaciones por categoria
conteo <- datos_pinguinos_clean |>
  count(species)
# Crear gráfico de columnas
conteo |>
  ggplot(aes(x=species, y=n)) +
  geom_col()
```



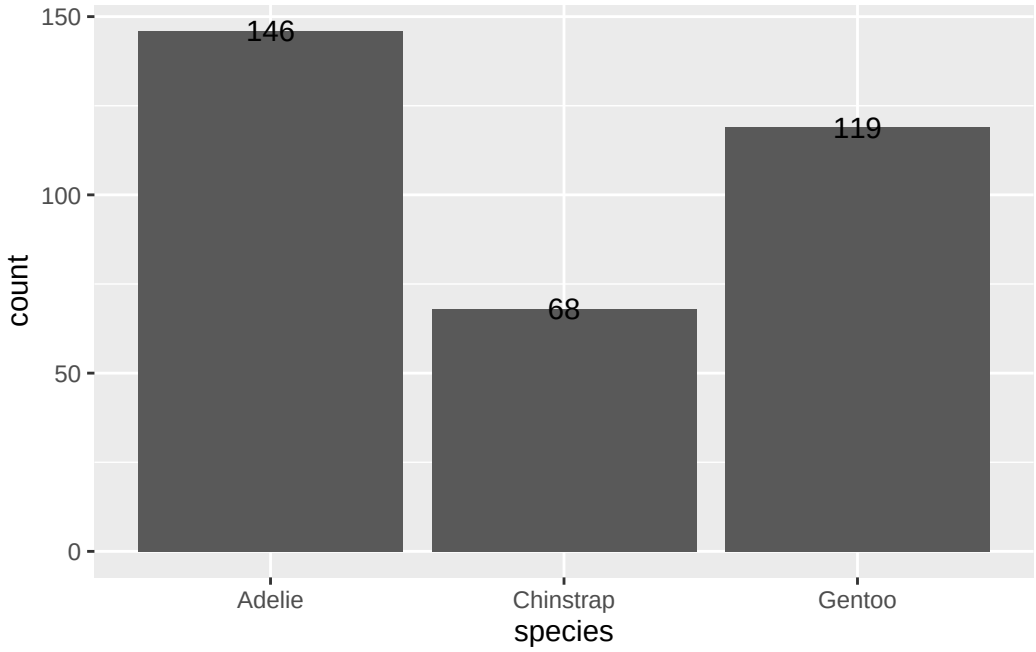

```
# Girar Gráfico modificando orden las variables X,Y
conteo |>
  ggplot(aes(y=species, x=n)) +
  geom_col()
```



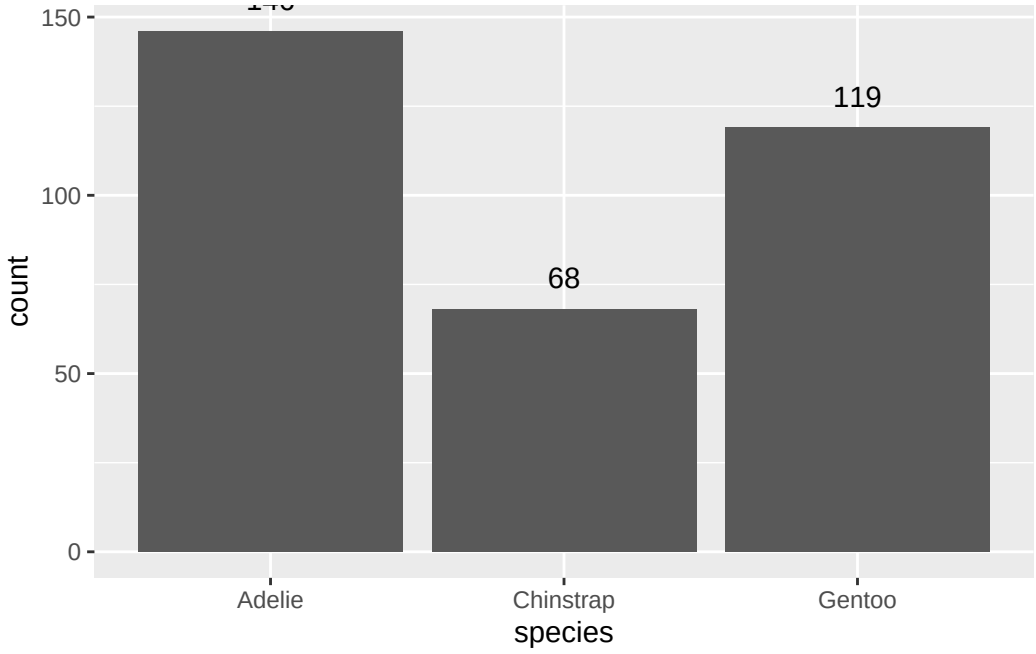
7.4 Insertar etiquetas de datos en el gráfico

Para añadir texto con los datos en cada columna se debe agregar la capa **geom_text()** y colocar las siguientes propiedades que cuenta el número de observaciones de la tabla **geom_text(aes(label = after_stat(count)),stat = "count")**.

```
# Observaciones por especies

datos_pinguinos_clean |>
  ggplot(aes(species)) +
  geom_bar()+
  geom_text(
    aes(label = after_stat(count)),
    stat = "count"
  )
```



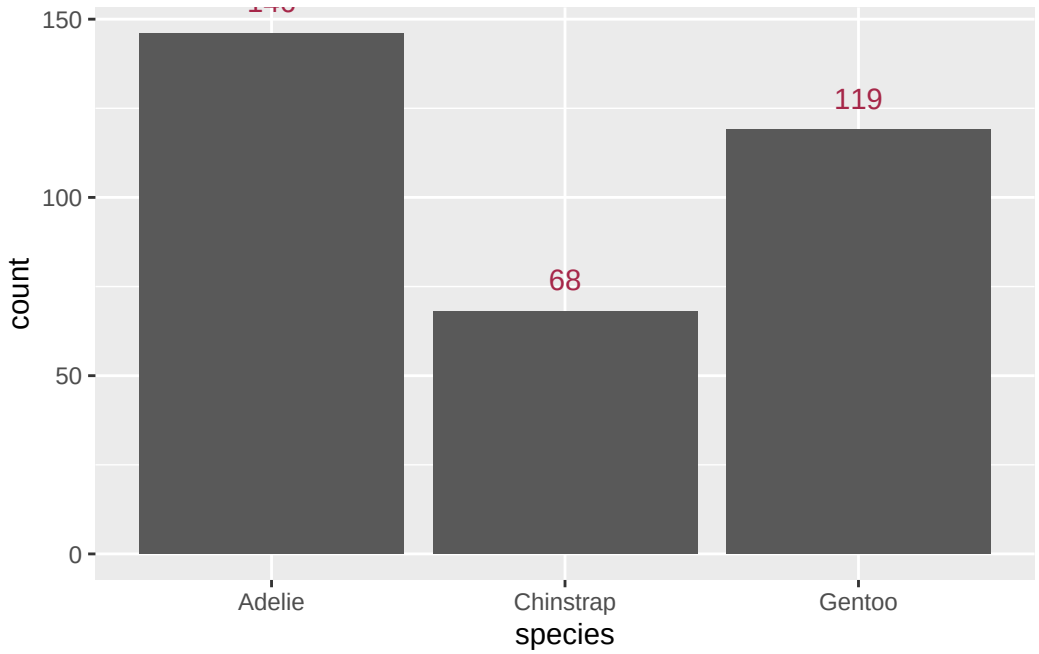

```
# Modificar posición de los numeros en capa vertical
# vjust = 0.0 (coloca el text por encima de la barra)
#  vjust = -1.0 (valores positivos para bajar texto, negativos para subir)

 datos_pinguinos_clean |>
  ggplot(aes(species)) +
  geom_bar()+
  geom_text(
    aes(label = after_stat(count)
        ),
    stat = "count" ,
    vjust = -1.0
  )
```



7.4 Modificar colores del texto de las barras

Usando la propiedad **colour** y el color del gráfico en hexadecimal o por su nombre, dentro de la función **geom_text()** podemos modificar el color del texto.

```
# modificar color del texto de etiqueta de datos
 datos_pinguinos_clean |>
  ggplot(aes(species)) +
  geom_bar()+
  geom_text(
    aes(label = after_stat(count)
        ),
    stat = "count" ,
    vjust = -1.0,
    colour="#a62749"
  )
```

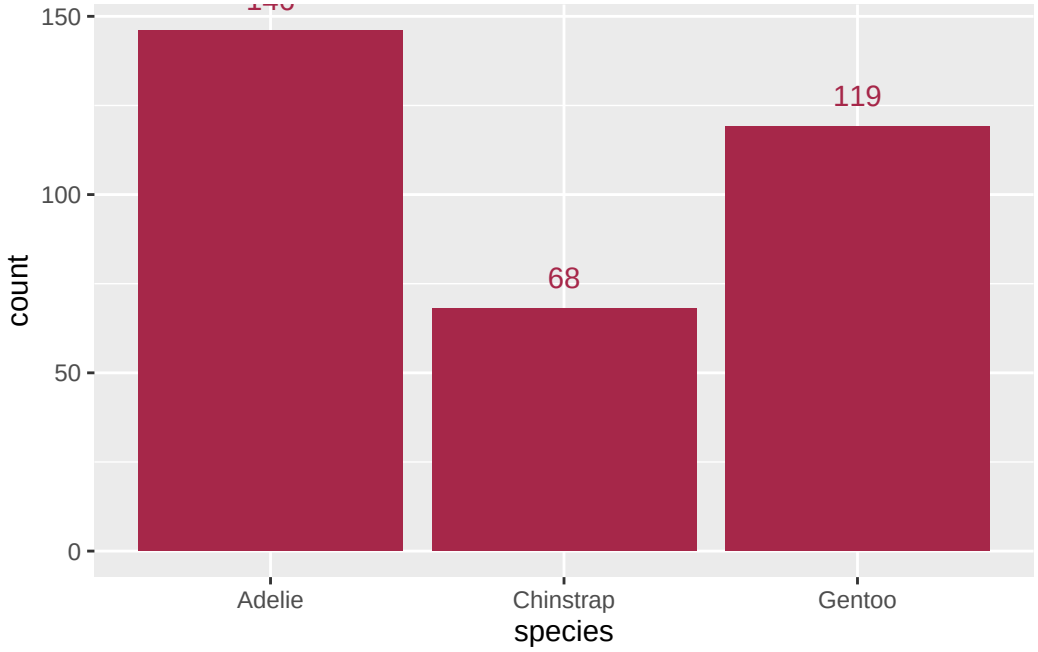


7.4 Modificar colores de las barras del gráfico

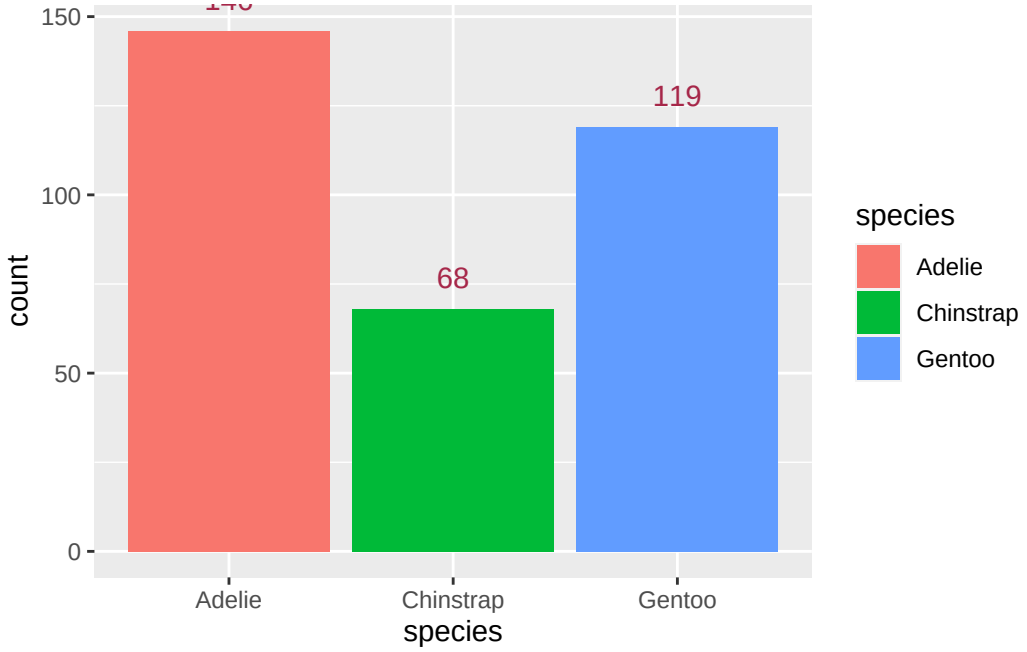
Para modificar el color de las barras se utiliza la propiedad **fill** en las propiedad de **aes(fill= "#color" / variable)** , se puede utilizar un solo color o asignar el color según variable categorica, en este ejemplo, utilizamos la variable **species**.

```
# Uno solo color en las barras

 datos_pinguinos_clean |>
  ggplot(aes(species,)) +
  geom_bar(fill="#a62749")+
  geom_text(
    aes(label = after_stat(count)
        ),
    stat = "count" ,
```

```
    vjust = -1.0,
    colour="#a62749"
  )
```



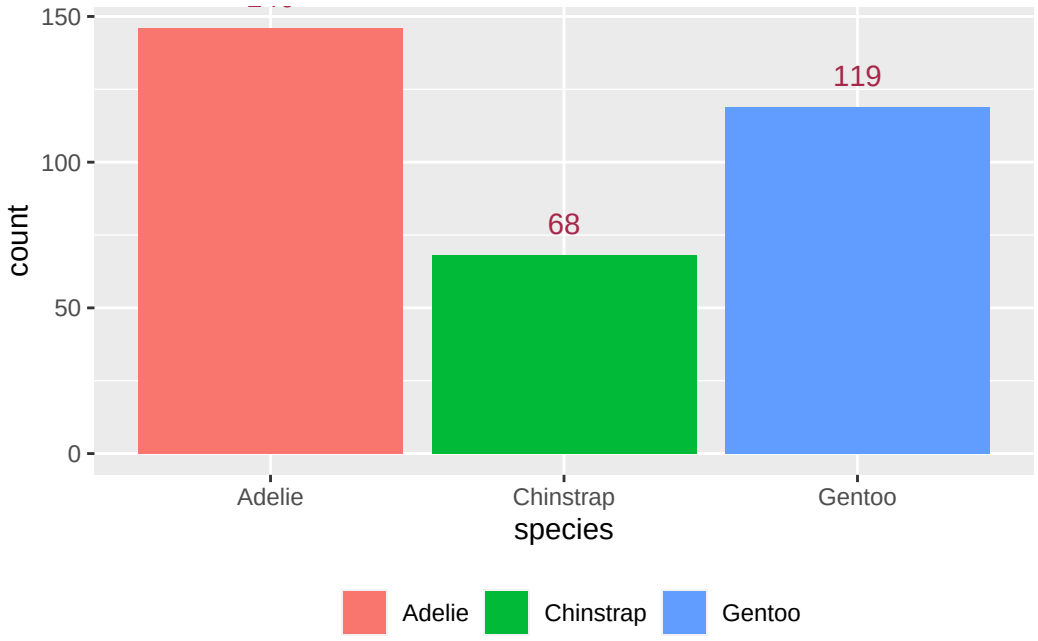

```
#Colores  en las barras según categorias
# Se crea una leyenda de los colores de la categoria
 datos_pinguinos_clean |>
  ggplot(aes(species,fill=species)) +
  geom_bar()+
  geom_text(
    aes(label = after_stat(count)
        ),
    stat = "count" ,
    vjust = -1.0,
    colour="#a62749"
  )
```



7.4 Modificar Posición de la leyenda de categorias

Para modificar la posición de la leyenda generada por ggplor, se utiliza la capa **theme()** y la propiedad **legend.position**, con los valores **“none”, “left”, “right”, “top”, “bottom”**. En este ejemplo ocultaremos la leyenda ya que el título de la leyenda no aporta nada, porque los colores y el eje x describen la categoría. Para eliminar el título de la leyenda debemos recordar que la leyenda se genera de la propiedad **fill**, para eliminarla utilizamos **labs(fill=“ “)**.

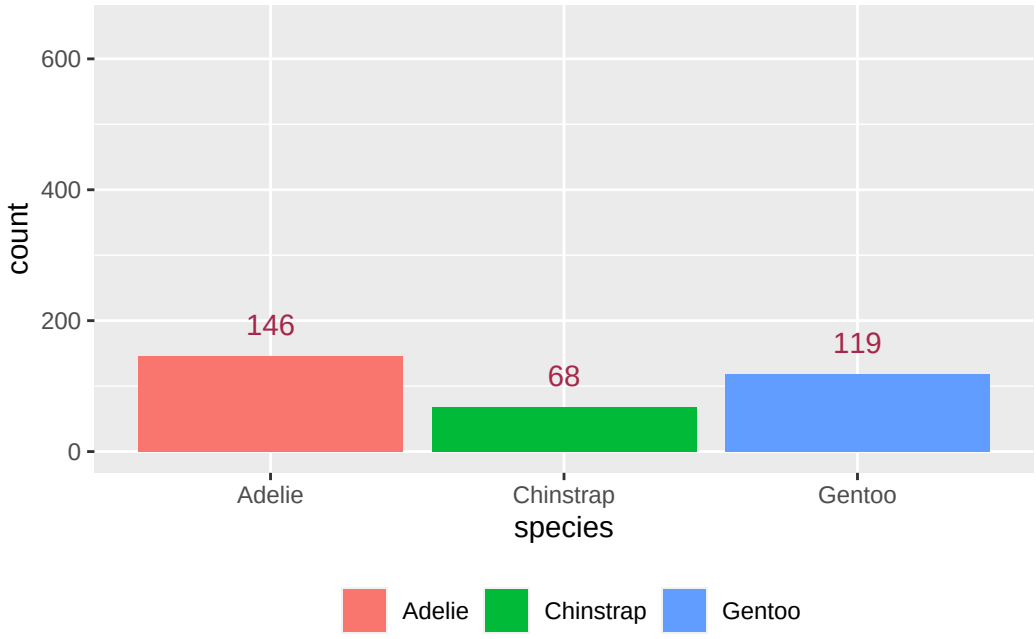
```
#Modificar la posición de la leyenda
 datos_pinguinos_clean |>
  ggplot(aes(species,fill=species)) +
  geom_bar()+
  geom_text(
    aes(label = after_stat(count)
        ),
    stat = "count" ,
    vjust = -1.0,
    colour="#a62749"
  )+
  theme(legend.position = "bottom")+
  labs(fill="")
```



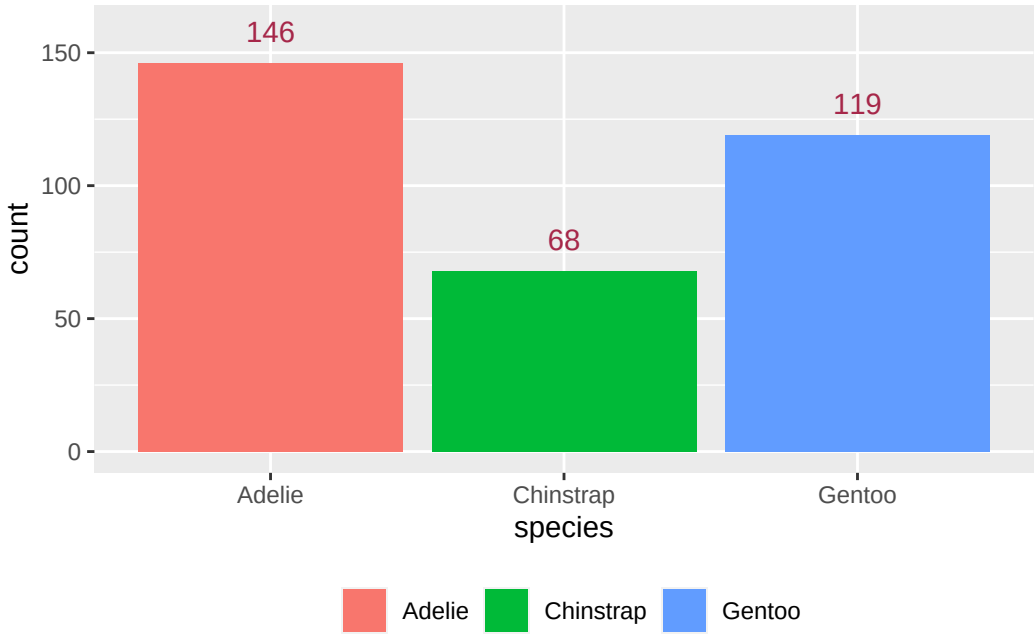
7.4 Modificar las coordenadas de los ejes

En el gráfico de ejemplo, los valores encima de las barras se cortan debido a que las etiquetas de los datos están por encima del valor del eje **y**. Para modificar las coordenadas **X y/o Y** en el gráfico se puede utilizar la capa de coordenadas **coord_cartesian(ylim=c(0,650))**. Se se desea modificar el eje **x** entonces se utilizaría **coord_cartesian(xlim=c(0,100))**, pero en este tipo de gráfico no es necesario.

```
#Se modifica la coordenada "Y" con los valores de 0 a 650  en un vector atómico
 datos_pinguinos_clean |>
  ggplot(aes(species,fill=species)) +
  geom_bar()+
  geom_text(
    aes(label = after_stat(count)
        ),
    stat = "count" ,
    vjust = -1.0,
    colour="#a62749"
  )+
  theme(legend.position = "bottom")+
  labs(fill="") +
  coord_cartesian(ylim=c(0,650))
```



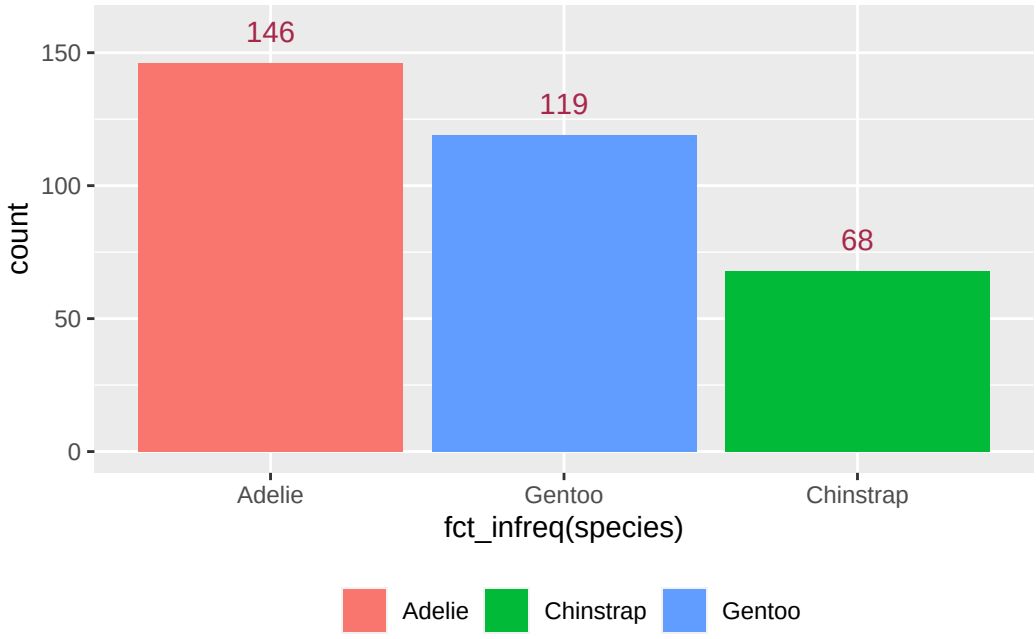

```
#Se modifica la coordenada "Y" con los valores de 0 a 160  en un vector atómico
 datos_pinguinos_clean |>
  ggplot(aes(species,fill=species)) +
  geom_bar()+
  geom_text(
    aes(label = after_stat(count)
        ),
    stat = "count" ,
    vjust = -1.0,
    colour="#a62749"
  )+
  theme(legend.position = "bottom")+
   labs(fill="") +
  coord_cartesian(ylim=c(0,160))
```



7.4 Ordenar barras por tamaño

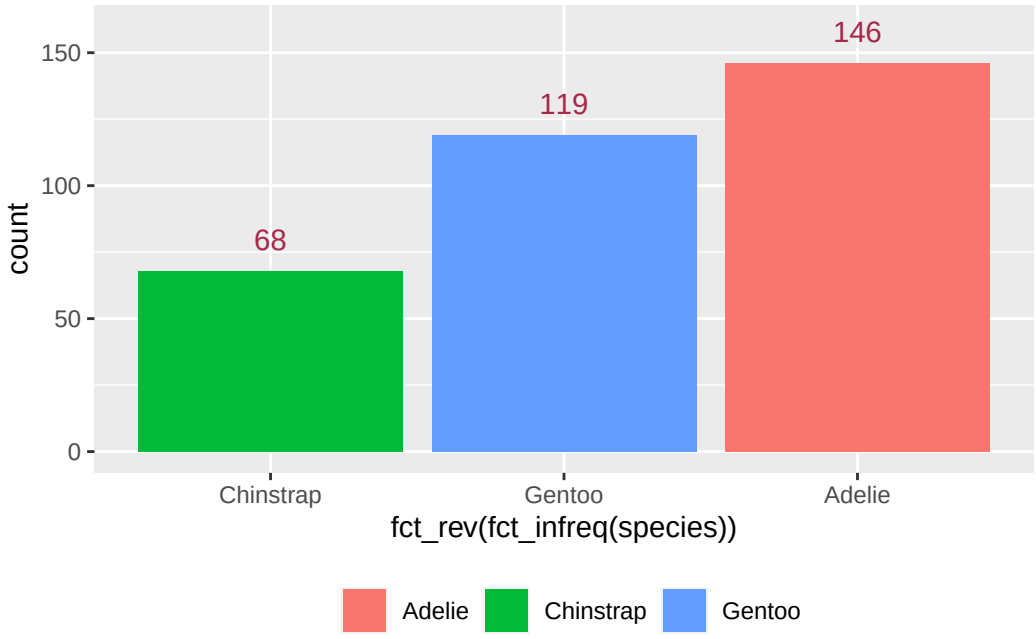
Algo común al utilizar el gráfico de barras es ordenar las barras según número de observaciones, para ello debemos utilizar la función **fct_infreq())** en la estetica **aes()** con la variable categórica **especies**,donde ggplot almacena el conteo por cada categoría.

```
#Se modifica la coordenada "Y" con los valores de 0 a 620  en un vector atómico
 datos_pinguinos_clean |>
  ggplot(aes(fct_infreq(species),
             fill=species)) +
  geom_bar()+
  geom_text(
    aes(label = after_stat(count)
        ),
    stat = "count" ,
    vjust = -1.0,
    colour="#a62749"
  )+
  theme(legend.position = "bottom")+
   labs(fill="") +
  coord_cartesian(ylim=c(0,160))
```



Invertir orden de las barras Para invertir el orden de las barras debemos utilizar la función **fct_rev()** en la estetica aes() que contiene la variable **especies** tomando en cuenta la función **fct_infreq()** que ordena el gráfico. Estas funciones de ordenamiento **solo** ordenan variables **categoricas** o de tipo **factor**.

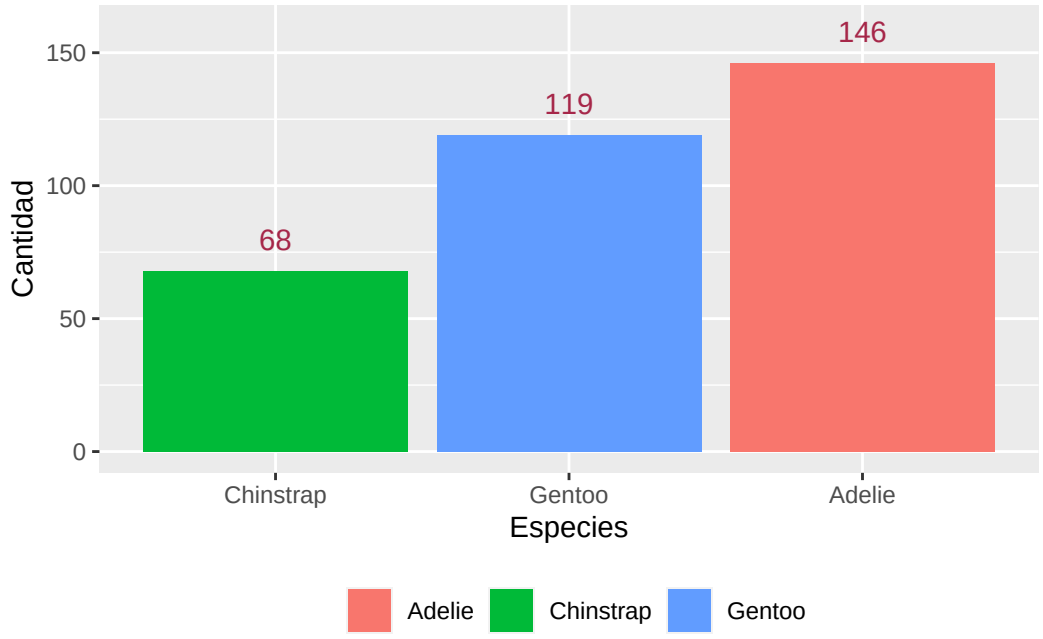
```
#Se modifica la coordenada "Y" con los valores de 0 a 620  en un vector atómico
 datos_pinguinos_clean |>
  ggplot(aes(fct_rev(fct_infreq(species)),
             fill=species)) +
  geom_bar()+
  geom_text(
    aes(label = after_stat(count)
        ),
    stat = "count" ,
    vjust = -1.0,
    colour="#a62749"
  )+
  theme(legend.position = "bottom")+
  labs(fill="") +
  coord_cartesian(ylim=c(0,160))
```



7.4 Modificar nombre de los ejes

Para modificar el nombre de los ejes debemos utilizar la función **lab()** e incluir los valores personalizados de **x / y**, los cuales se colocan dentro de la función de la siguiente forma **labs(x="Especies", y="Cantidad")**.

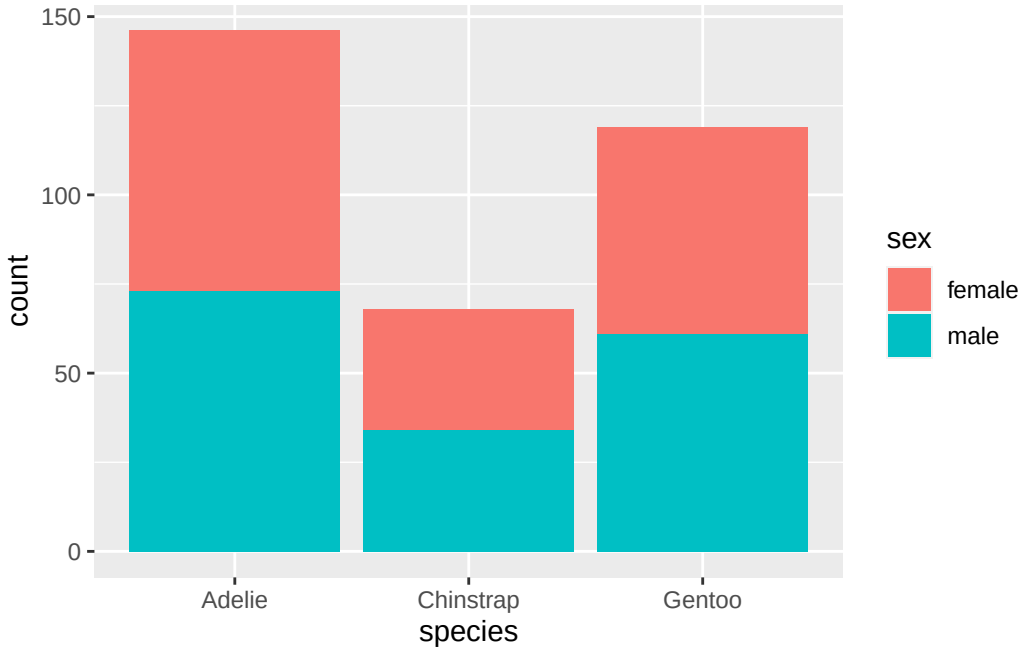
```
#Se modifica la coordenada "Y" con los valores de 0 a 620  en un vector atómico
 datos_pinguinos_clean |>
  ggplot(aes(fct_rev(fct_infreq(species)),
             fill=species)) +
  geom_bar()+
  geom_text(
    aes(label = after_stat(count)
        ),
    stat = "count" ,
    vjust = -1.0,
    colour="#a62749"
  )+
  theme(legend.position = "bottom")+
  labs(fill="",
       x="Especies",
       y="Cantidad") +
  coord_cartesian(ylim=c(0,160))
```



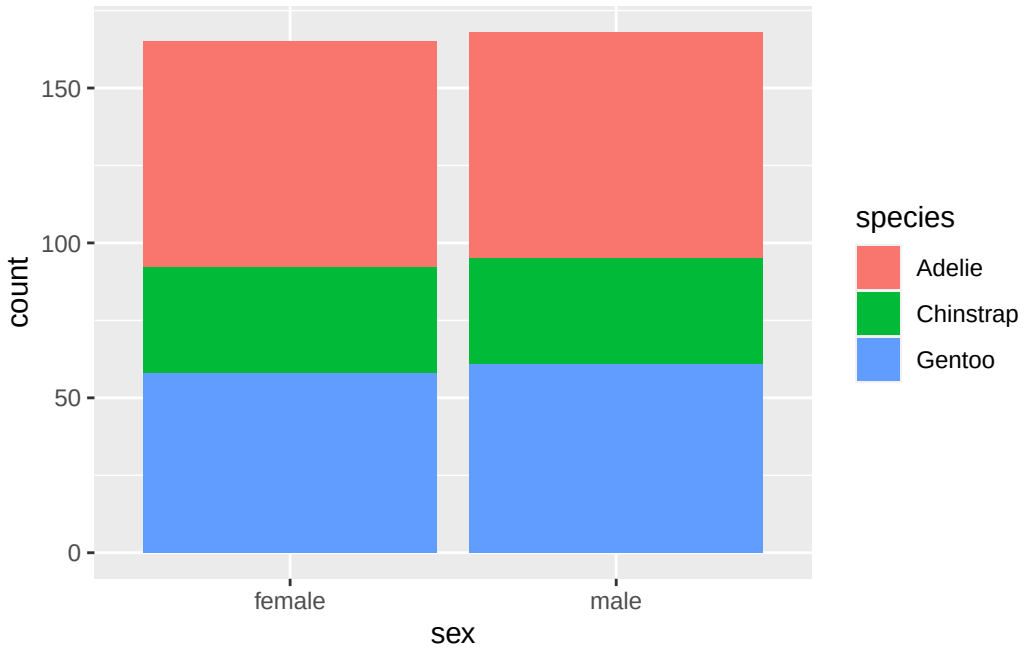
7.5 Barras apiladas

En ggplot podmeos hacer un gráfico utilizando dos variables categóricas en los gráficos de barras agrupando los datos por la estética de color **fill= variable categorica** de esta forma se creará lo que se conoce como **barras apiladas**.

```
# Dos variables categoricas
# species es la categoria en el eje x
# sex la categoria que apila las barras

datos_pinguinos_clean |>
  ggplot(aes(species, fill=sex)) +
  geom_bar()
```



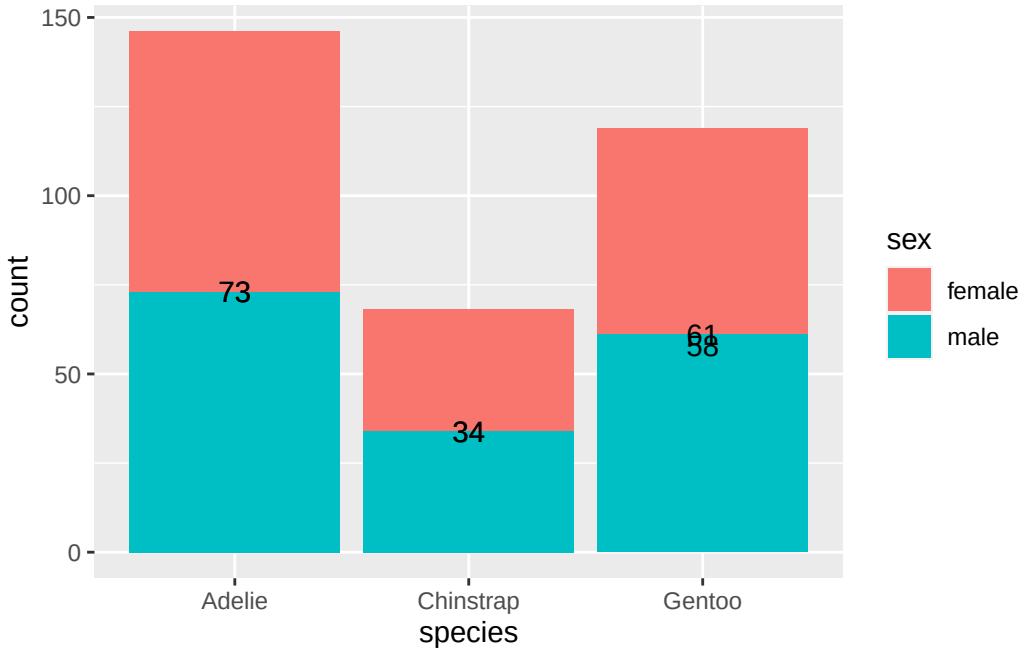

```
# Dos variables categoricas
# sex es la categoria en el eje x
# speciesla categoria que apila las barras
datos_pinguinos_clean |>
  ggplot(aes(sex, fill=species)) +
  geom_bar()
```



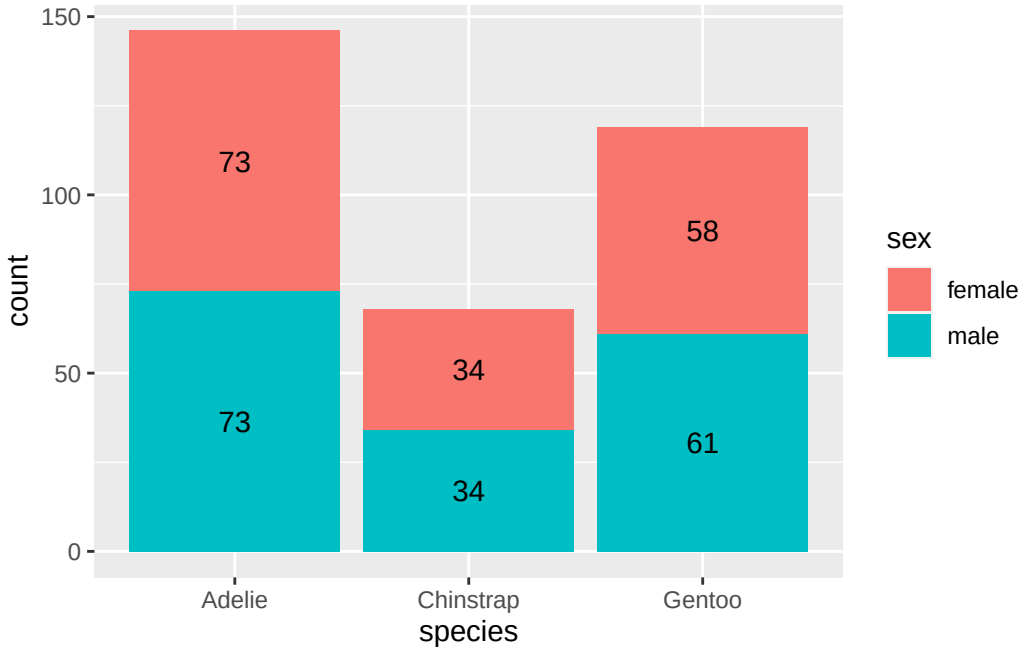
7.5 Etiqueta de datos en barras apiladas

Las barras apiladas tambien son conocidas como **stack bar** es por ello que para poder insertar los datos en barras apiladas debemos cambiar su posición predeterminada de los datos utilizando **position = position_stack(x,y)** en la geoametría de texto **eom_text()**.

```
#añadir capa geom_text()
# se insertan las etiquetas de datos , pero , no centralizadas
datos_pinguinos_clean |>
  ggplot(aes(species, fill=sex)) +
  geom_bar()+
  geom_text(
    aes(label = after_stat(count)
        ),
    stat = "count"
  )
```



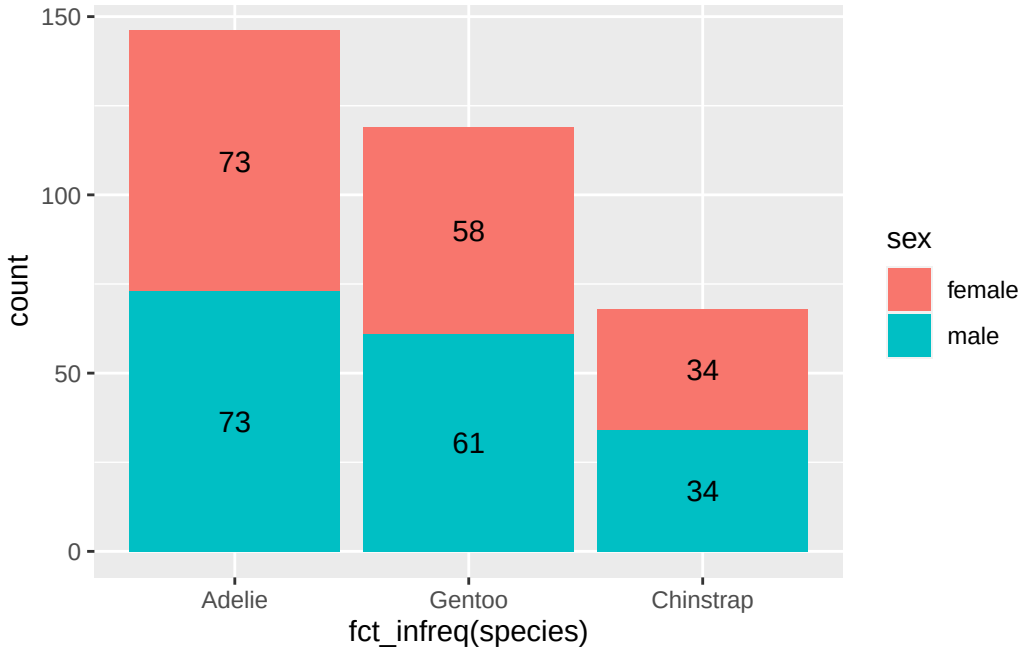

```
#para centrarlas se utiliza la propiedad position = position_stack(0.5, 0.0) dentro de goem_
datos_pinguinos_clean |>
  ggplot(aes(species, fill=sex)) +
  geom_bar()+
  geom_text(
    aes(label = after_stat(count)
        ),
    stat = "count",
    position = position_stack(0.5, 0.0)
  )
```



7.5 Ordenar barras apiladas

Para ordenar las barras aipladas utilizamos la función de ordenación **fct_infreq()** en las categoria del eje x y la varible categórica **species**.

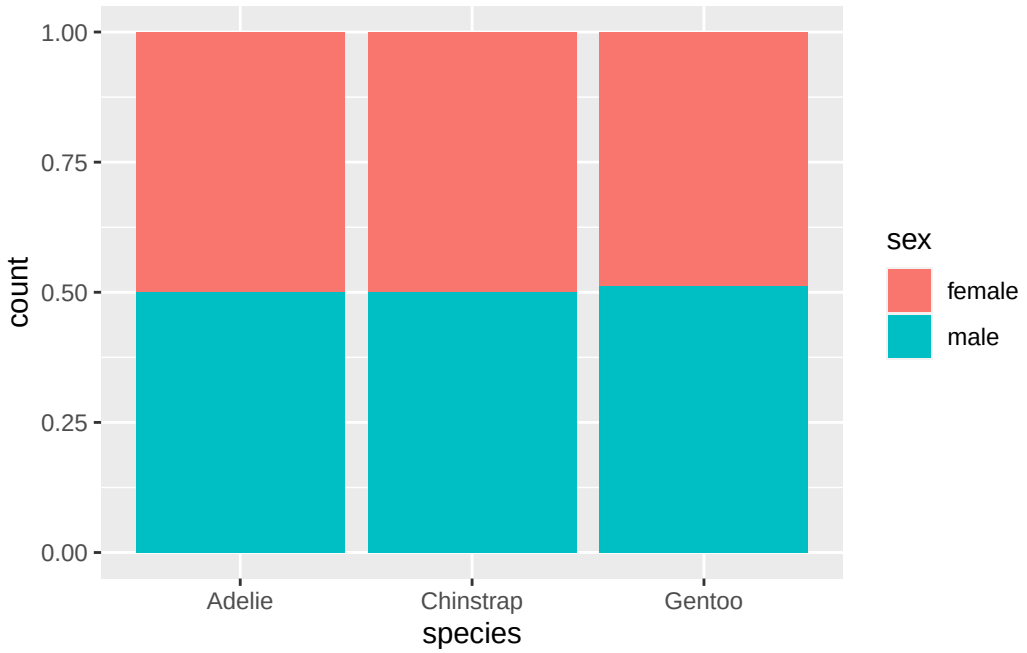
```
# Ordenar barras apiladas
datos_pinguinos_clean |>
  ggplot(aes(fct_infreq(species), fill=sex)) +
  geom_bar()+
  geom_text(
    aes(label = after_stat(count)
        ),
    stat = "count",
    position = position_stack(0.5, 0.0)
  )
```



7.5 Barras apiladas - datos relativos

Si queremos comparar las columnas con proporciones relativas, debemos añadir a la geometría la propiedad **geom_bar(position = "fill")**, con esto ggplot modificará los valores de la proporción de datos de la variable **sex** en cada columna **island**.

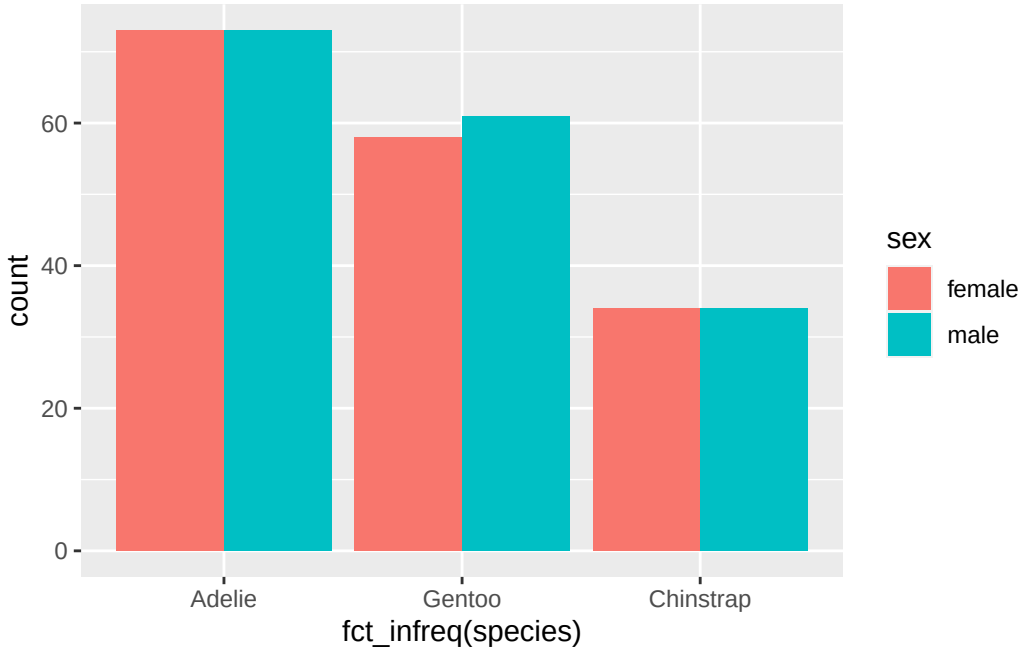
```
datos_pinguinos_clean |>
  ggplot(aes(species, fill=sex)) +
  geom_bar(position = "fill")
```



7.6 Barras No apiladas

Se queremos utilizar dos variables categóricas en el gráfico de barras, pero que no esten apiladas, sino una al lado de la otra, se debe utilizar en la geometría de la barra la **geom_bar(position = "dodge")** para que ggplot pueda reconocer que las barras no serán apiladas.

```
# grafico de dos variables categoricas agrupadas por color
# Ordenar barras no apiladas
datos_pinguinos_clean |>
  ggplot(aes(fct_infreq(species), fill=sex)) +
  geom_bar(position = "dodge")
```

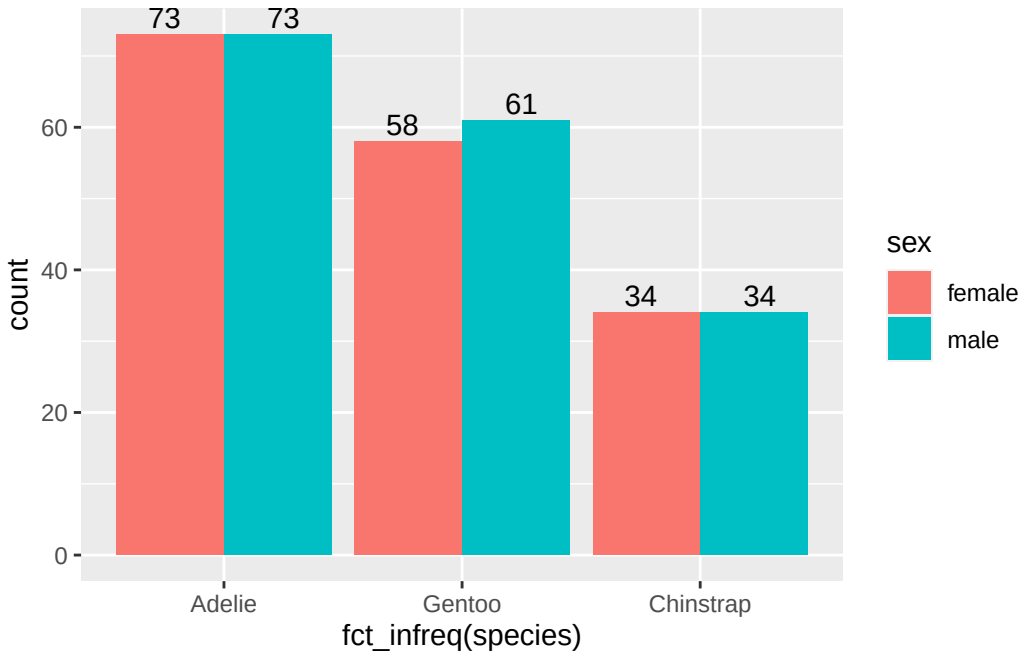


7.6 Modificar la posición de la etiqueta de datos en barras no apiladas

Al cambiar la propiedad **dodge** de barras no apiladas, se debe cambiar tambien la posición de los datos de texto con **position = position_dodge(1.0)** en la geometria de texto **geom_text()**. El valor de **1.0** permite colocar los datos encima de cada barra y centrado horizontalmente en cada barra. Para modificar la posición del texto en cada barra debemos ajustarlo con **vjust (posicion vertical)** y **hjust (posicion horizontal)**.

```
# Modificar posicion de la leyenda de datos con,
# p

datos_pinguinos_clean |>
  ggplot(aes(fct_infreq(species), fill=sex)) +
  geom_bar(position = "dodge")+
  geom_text(
    aes(label = after_stat(count)
        ),
    stat = "count",
    position = position_dodge(1.0),
```

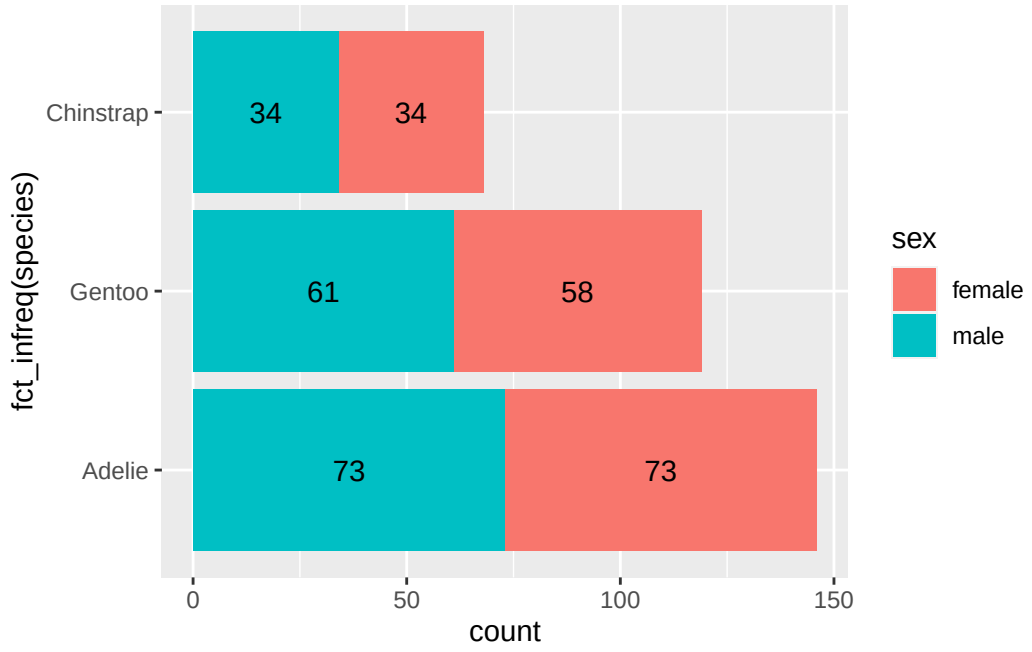
```
    vjust=-0.3,
    hjust=0.5
  )
```



7.6 Girar gráfico completo

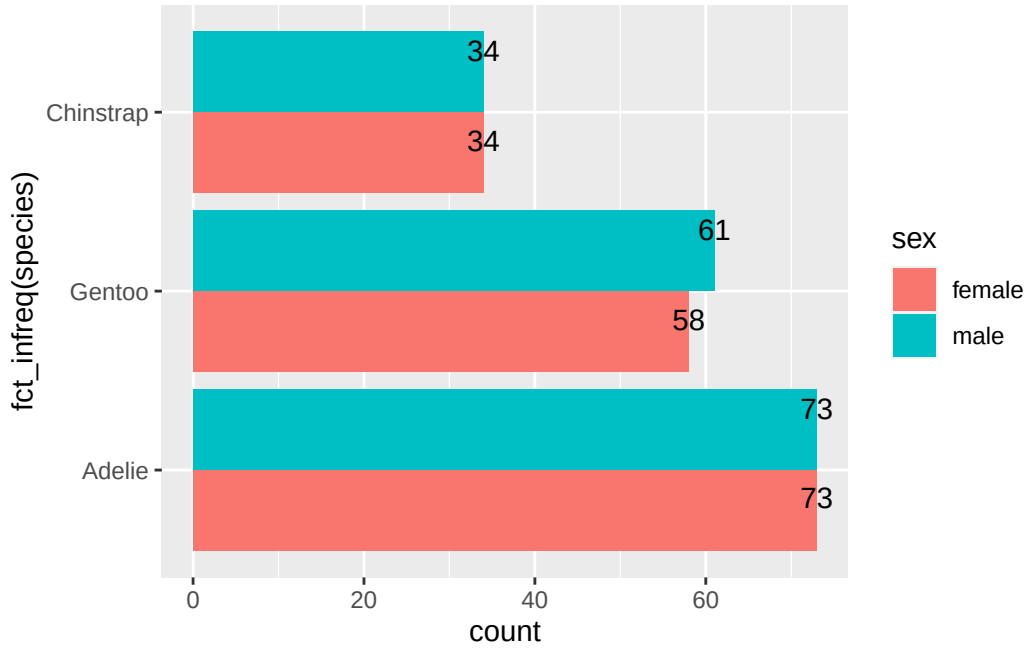
Para girar un gráfico completo utilizamos una nueva capa con la función **coord_flip()**. Solo giran las barras y las etiquetas. Las etiquetas de datos se le debe modificar su posición con **vjust y hjust**.

```
# Girar barras apiladas
datos_pinguinos_clean |>
  ggplot(aes(fct_infreq(species), fill=sex)) +
  geom_bar()+
  geom_text(
    aes(label = after_stat(count)
        ),
    stat = "count",
    position = position_stack(0.5, 0.0)
  )+
  coord_flip()
```



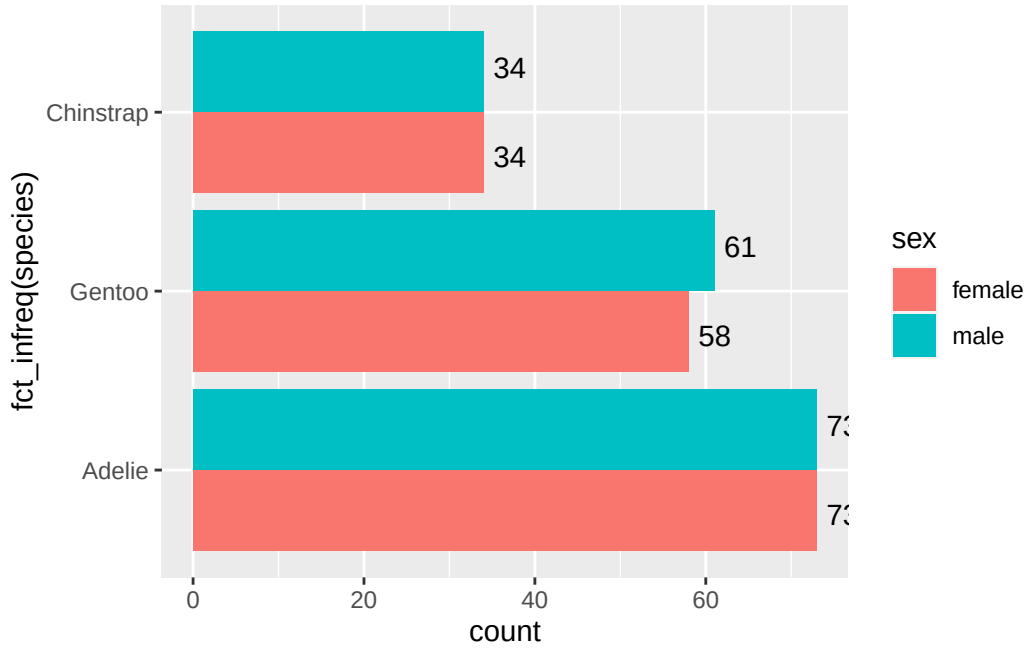
Giro de gráfico de barras no apiladas

```
# Girar barras no apiladas
# valores no ajustados
datos_pinguinos_clean |>
  ggplot(aes(fct_infreq(species), fill=sex)) +
  geom_bar(position = "dodge")+
  geom_text(
    aes(label = after_stat(count)
        ),
    stat = "count",
    position = position_dodge(1.0),
    vjust=-0.3,
    hjust=0.5
  )+
  coord_flip()
```



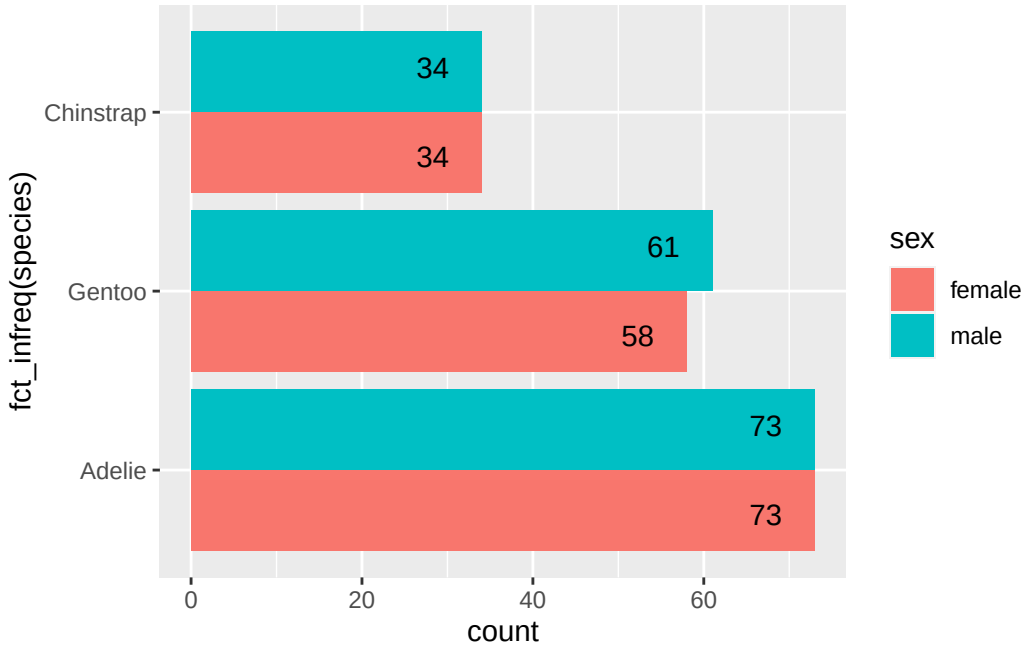
Girar barras no apiladas - valores ajustados fuera de la barra

```
datos_pinguinos_clean |>
  ggplot(aes(fct_infreq(species), fill=sex)) +
  geom_bar(position = "dodge")+
  geom_text(
    aes(label = after_stat(count)
        ),
    stat = "count",
    position = position_dodge(1.0),
    vjust=0.5,
    hjust=-0.3
  )+
  coord_flip()
```



Girar barras no apiladas - valores ajustados dentro de la barra

```
# Girar barras no apiladas
# valores ajustados dentro de la barra
datos_pinguinos_clean |>
  ggplot(aes(fct_infreq(species), fill=sex)) +
  geom_bar(position = "dodge")+
  geom_text(
    aes(label = after_stat(count)
        ),
    stat = "count",
    position = position_dodge(1.0),
    vjust=0.5,
    hjust= 2.0
  )+
  coord_flip()
```



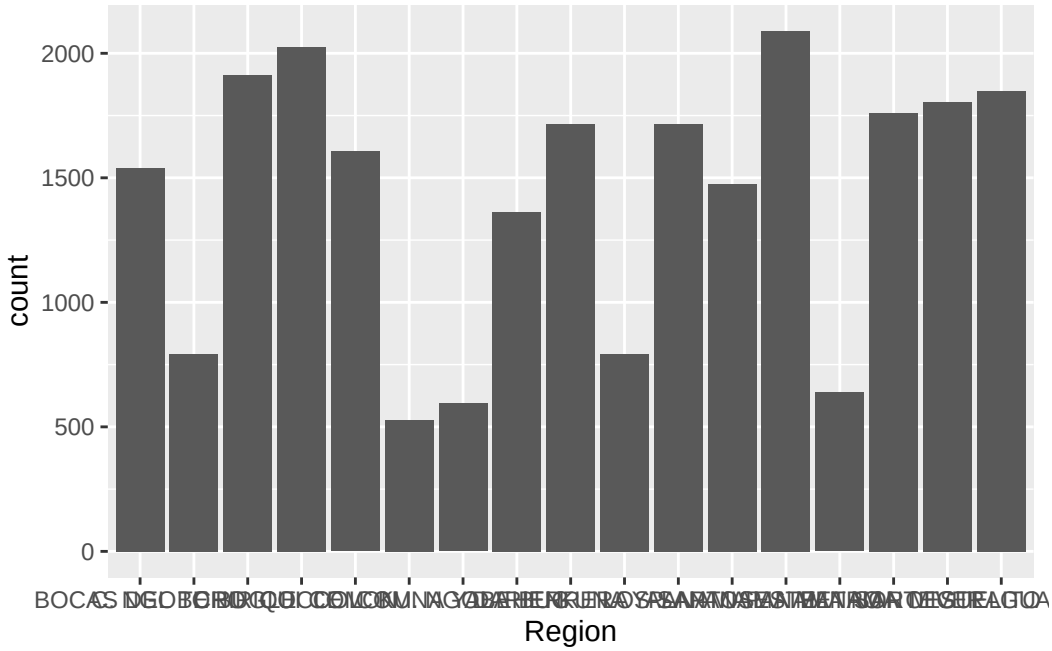
7.7 Recomendaciones

7.7 Recomendación 1

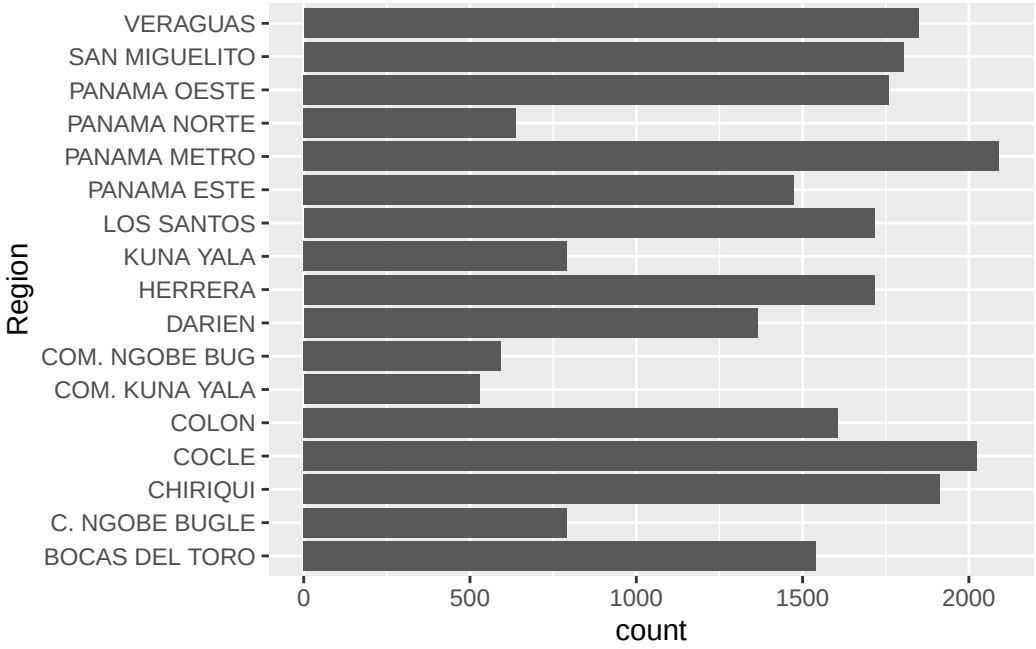
Muchas Categorias o etiquetas extensa

Los gráficos de barras son utilizamos con mayor frecuencia para generar **ranking de una categoria** o cuando el **número de categorías es muy extenso**. Para transformar las columnas a barras en ggplot es necesario girar todo el gráfico utilizando **coord_flip()** o modificando el eje de la variable de **X a Y**.

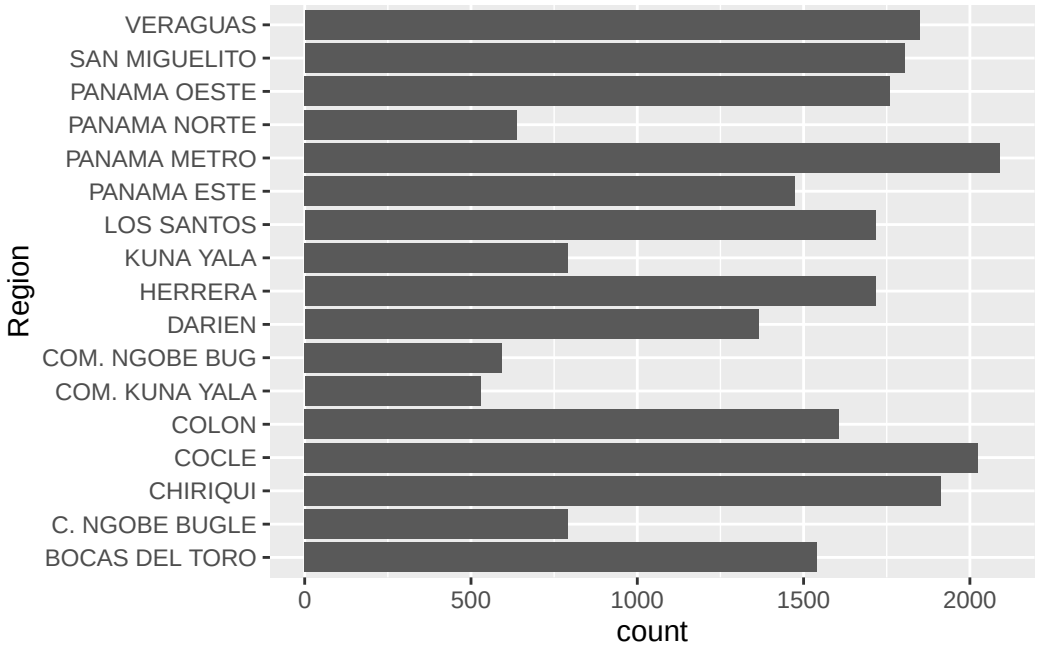
```
# Cargar datos
inec <- read.csv("data/INEC-morbilidad-2023-10-14.csv")
inec <- as_tibble(inec)

#Gráfico de columnas con un número considerables de categorias
# No se logra leer bien las etiquetas
inec |>
  ggplot(aes(Region)) +
  geom_bar()
```




```
#Gráfico de columnas transformado en barras, girando el gráfico
# en este caso se logra leer bien las etiquetas
inec |>
  ggplot(aes(Region)) +
  geom_bar() +
  coord_flip()
```




```
# Modificando el eje de la variable Region
# el valor predeterminado es X, lo cambiamos por Y
inec |>
  ggplot(aes(y=Region)) +
  geom_bar()
```

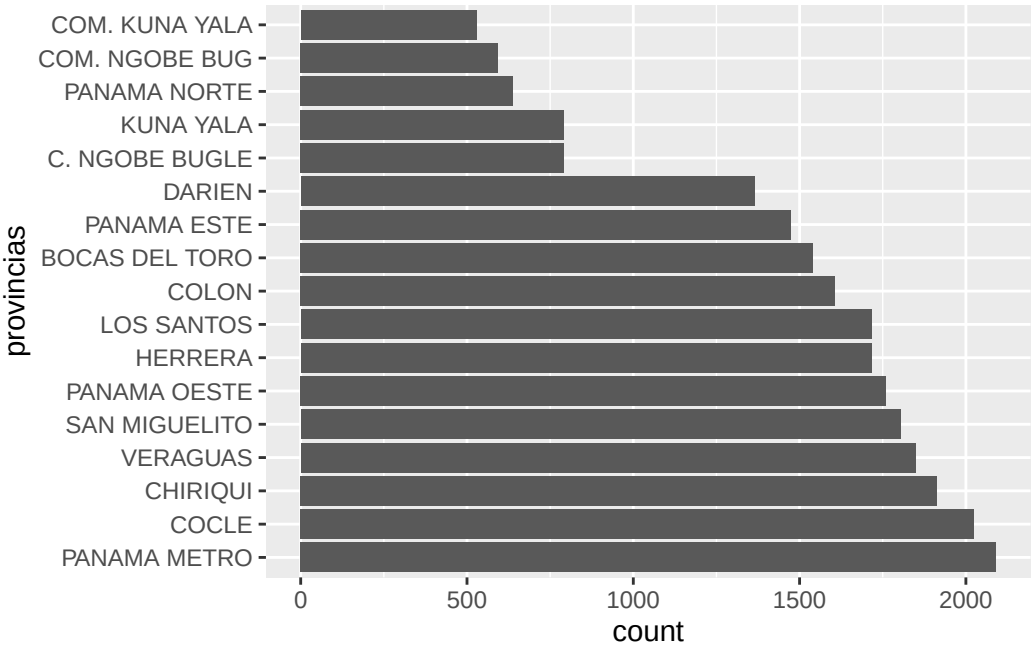


7.7 Recomendación 2

Ordenar barras o columnas Cuando creamos un gráfico de barras o columnas estás por naturaleza se ordenan según la categoría, principalmente en orden alfabético, sin embargo podemos modificar el orden de los datos según el conteo de los datos por categoría. Para ello podemos utiliza en la estética **aes()** la función **fct_infreq(variable categorica)** para ordenar los datos, estos se ordenarán de menor a mayor. Si deseamos invertir el orden, debemos utilizar la función **fct_rev()** que invierte el orden de las categorias, por lo que es importante utilizarlas combinadas **fct_rev(fct_infreq(Region))**.

Ordenar gráfico de barra de menor a mayor

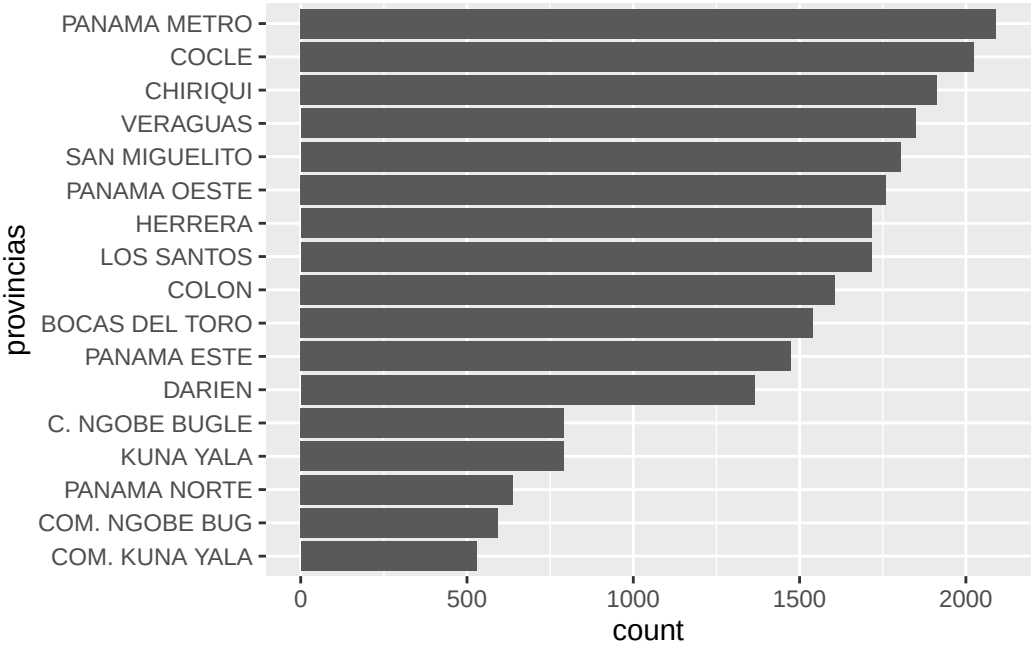
```
inec |>
  ggplot(aes(fct_infreq(Region))) +
  geom_bar() +
  coord_flip()+
  labs(x="provincias")
```



Ordenar gráfico de barra de mayor a menor

Añadir la función **fct_rev()** al ordenamiento de l avariable **Region**.

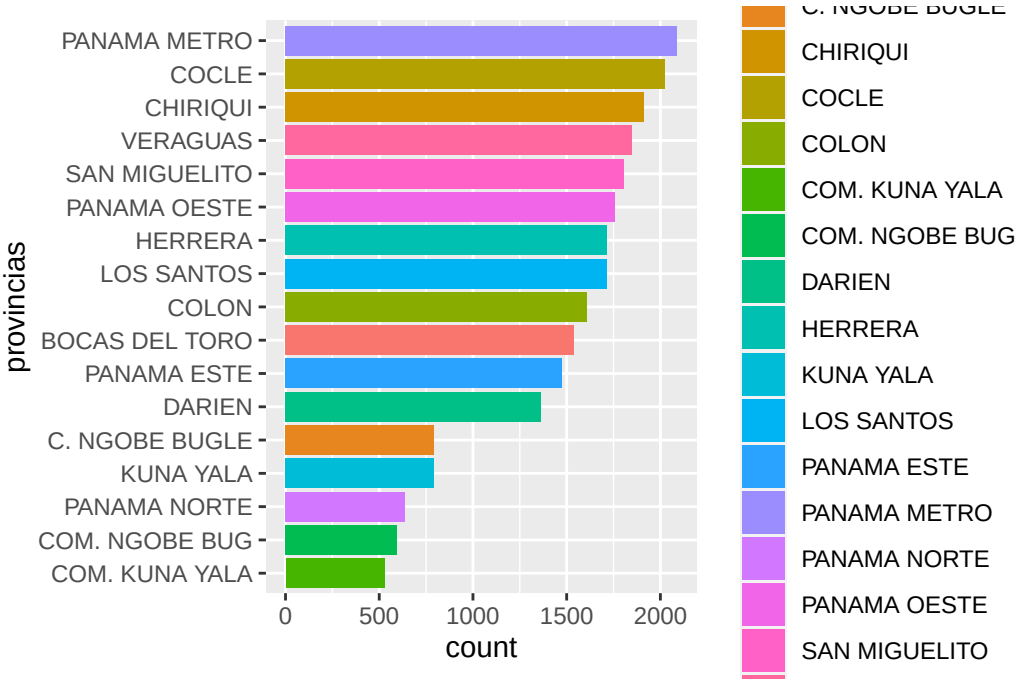
```
inec |>
  ggplot(aes(fct_rev(fct_infreq(Region)))) +
  geom_bar() +
  coord_flip()+
  labs(x="provincias")
```



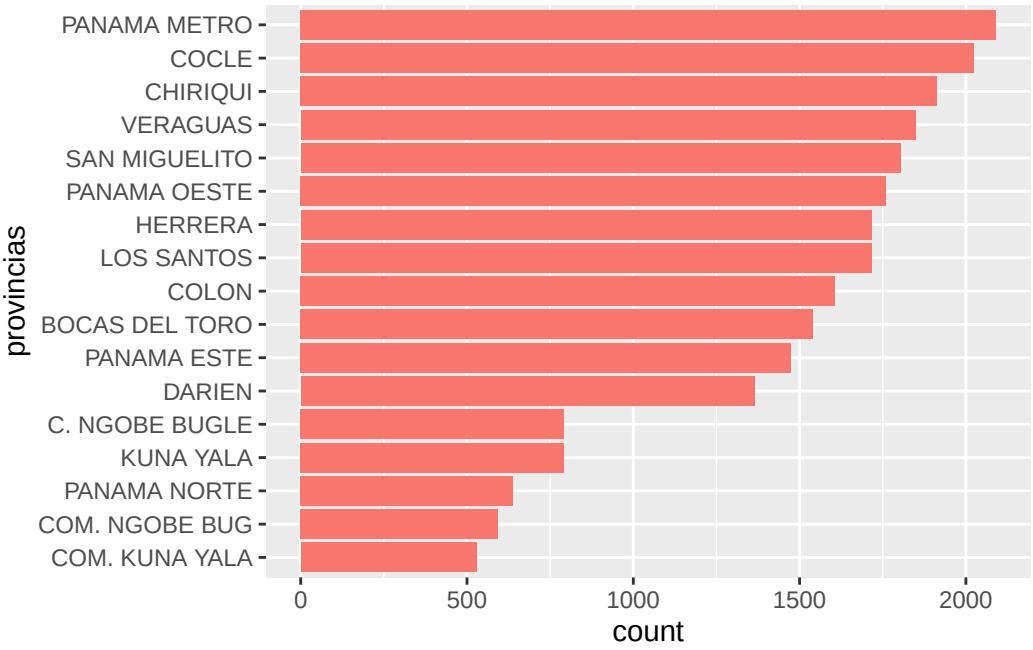
7.7 Recomendación 3

Qué aporta el color? Una características muy utilizada en los gráficos de barras o columnas, es colorear las barras cada una con diferentes colores, sin embargo, el color en la barra es una forma de categorizar los datos en el gráfico, por lo que si este no es el fin del color al utilizarlo, mejor utilizar un solo color en todas las barras, ya que probablemente sean todos de una misma categoria y lo que se resalta es el orden según la cantidad de observaciones de todas las categorías.

```
# varios colores en las barras
inec |>
  ggplot(aes(fct_rev(fct_infreq(Region)))) +
  geom_bar(aes(fill=Region)) +
  coord_flip()+
  labs(x="provincias")
```



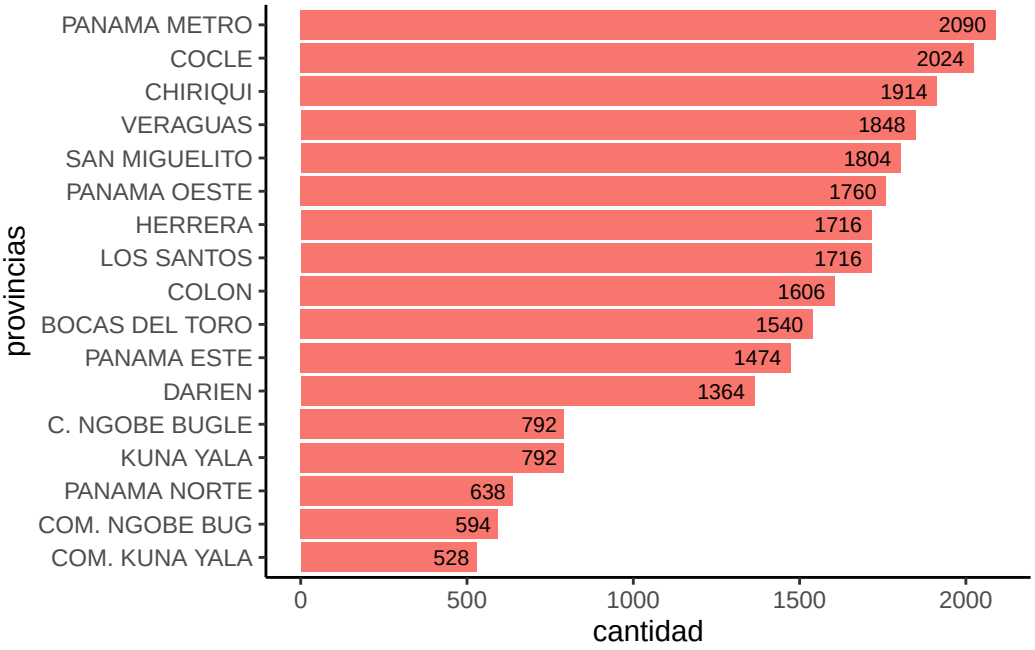

```
# un solo color en las barras
inec |>
  ggplot(aes(fct_rev(fct_infreq(Region)))) +
  geom_bar(aes(fill="salmon")) +
  coord_flip()+
  labs(x="provincias")+
  guides(fill="none") # eliminar leyenda
```



7.7 Recomendación 4

Menos es más Eliminar elementos que no aportan al gráfico como las leyendas de color, grillas y añadir valores, hacen del gráfico más simple pero con la información necesaria. Es por ello que es importante evaluar los elementos del gráfico que aportar a la visualización.

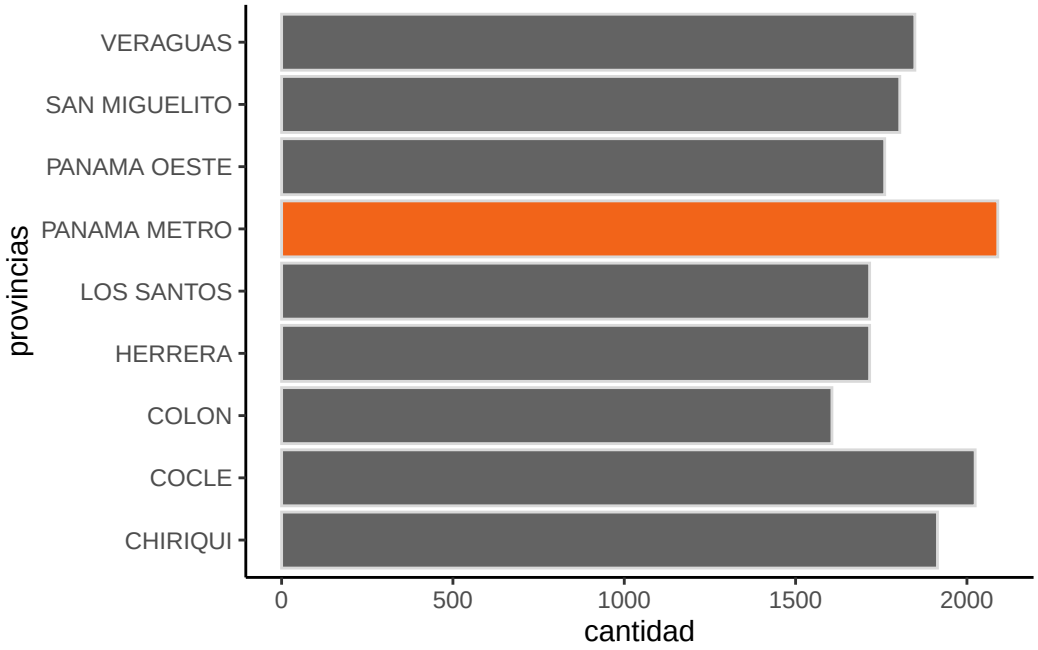
```
# un solo color en las barras
inec |>
  ggplot(aes(fct_rev(fct_infreq(Region)))) +
  geom_bar(aes(fill="salmon")) +
  coord_flip()+
  labs(x="provincias",
       y="cantidad")+
  guides(fill="none")+  # eliminar leyenda
  geom_text(
    aes(label = after_stat(count)
        ),
    stat = "count",
    vjust=0.5,
    hjust= 1.2,
    size=2.9
  )+
  theme_classic()
```



7.7 Recomendación 5

Énfasis en una barra Podemos ordenar el gráfico de barras según el número de observaciones, pero también podemos hacer énfasis en una categoría en particular la cual podemos resaltar o hacer énfasis por medio del color.

```
#El número de colores esta asociado al núemero de categorias
# En el caso de Región son 9 categorias
colores <- c("#636363","#636363", "#636363", "#636363" ,"#636363" ,"#636363" ,"#636363","#63
colores[1] <- "#F26419"

# un solo color en las barras

data <- inec |>
  count(Region) |>
  arrange(desc(n))

data |>
  filter(n>1540) |>
  ggplot(aes(x=Region, y=n)) +
  geom_col(fill=colores, color="#D9D9D9") +
  coord_flip()+
  labs(x="provincias",
       y="cantidad")+
  guides(fill="none")+  # eliminar leyenda
  theme_classic()
```



7.7 Reodernar barras - OP2

Otra forma de reordenar el gráfico de barras, es reordenar la variable categórica, esto se hace basado en que una variable categorica es de tipo **factor** en R, la cual contiene niveles **levels** de las diversas categoría y son los datos que utiliza ggplot para ordenar el gráfico con esa variable.

Gráfico de barra ordenado según Región (orden A-Z) Si creamos el gráfico, este ordena las barras en orden alfabetico de las categorias.

```
#Variable Región como factor
inec$Region <- as.factor(inec$Region)

#Niveles de los datos Region
levels(inec$Region)
```

```
 [1] "BOCAS DEL TORO" "C. NGOBE BUGLE" "CHIRIQUI"       "COCLE"
 [5] "COLON"          "COM. KUNA YALA" "COM. NGOBE BUG" "DARIEN"
 [9] "HERRERA"        "KUNA YALA"      "LOS SANTOS"     "PANAMA ESTE"
[13] "PANAMA METRO"   "PANAMA NORTE"   "PANAMA OESTE"   "SAN MIGUELITO"
[17] "VERAGUAS"
```

```
# un solo color en las barras
data <- inec |>
  count(Region) |>
  arrange(desc(n))


# Crear gráfico 1
data |>
  ggplot(aes(x=Region, y=n)) +
  geom_col(fill="salmon") +
  coord_flip()+
```

```
  labs(x="provincias",
       y="cantidad")+
  guides(fill="none")+  # eliminar leyenda
  theme_classic()
```

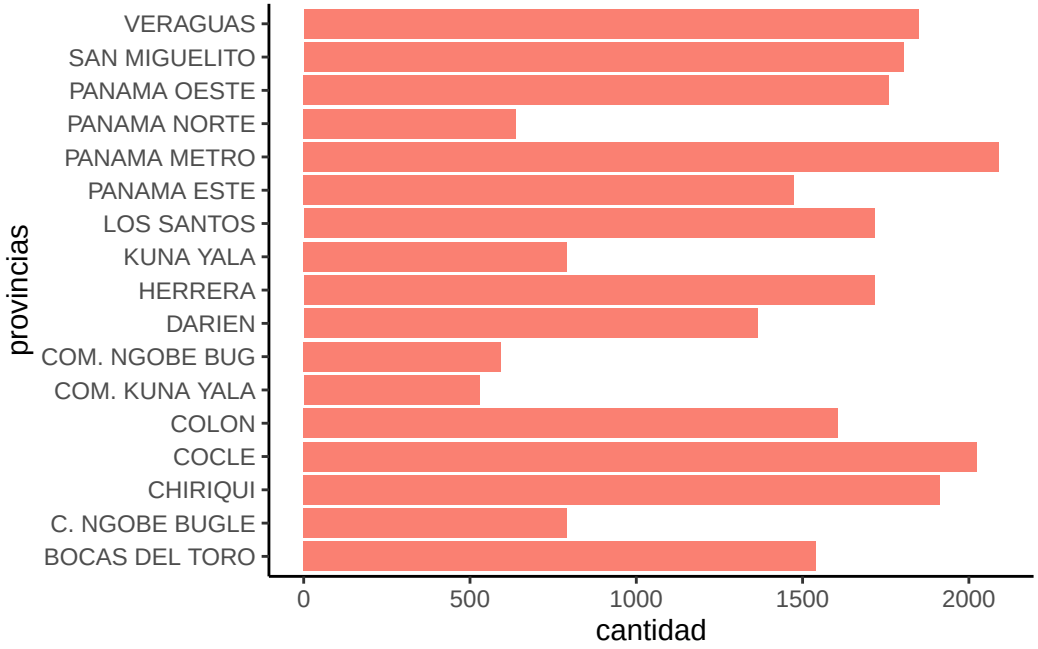


Gráfico de barra ordenado según levels() de la variable Region Si mostramos los datos del tibble **data** vemos que la variable **Region** no esta ordenada en orden alfabético sino por el valor de **n**, pero debemos inidcarle a esta variable tipo factor, que los valores **levels** de esa categoria deben tener el mismo orden del tibble **data**. Los datos del orden de la variable **REgion** la almacenaremos en un vector llamado **ordenREgion**.

```
# Imprimir tibble data
# datos ordenados por el valor de n
data |>
  head(5)
```

```
# A tibble: 5 x 2
  Region            n
  <fct>         <int>
1 PANAMA METRO   2090
2 COCLE          2024
3 CHIRIQUI       1914
4 VERAGUAS       1848
5 SAN MIGUELITO  1804
```

```
# Capturar datos del orden de la variable Region
# Region ordenado por valor de "n"
ordenRegion <- data$Region

# Ordenar niveles de variable categorica
data$Region<- factor(data$Region,
```

```
                      levels = ordenRegion)

#Niveles ordenados según tibble
levels(data$Region)
```

```
 [1] "PANAMA METRO"    "COCLE"           "CHIRIQUI"        "VERAGUAS"
 [5] "SAN MIGUELITO"   "PANAMA OESTE"    "HERRERA"         "LOS SANTOS"
 [9] "COLON"           "BOCAS DEL TORO"  "PANAMA ESTE"     "DARIEN"
[13] "C. NGOBE BUGLE"  "KUNA YALA"       "PANAMA NORTE"    "COM. NGOBE BUG"
[17] "COM. KUNA YALA"
```

```
# Crear gráfico 2
data |>
  ggplot(aes(x=Region, y=n)) +
  geom_col(fill="salmon") +
  coord_flip()+
  labs(x="provincias",
       y="cantidad")+
  guides(fill="none")+  # eliminar leyenda
  theme_classic()
```

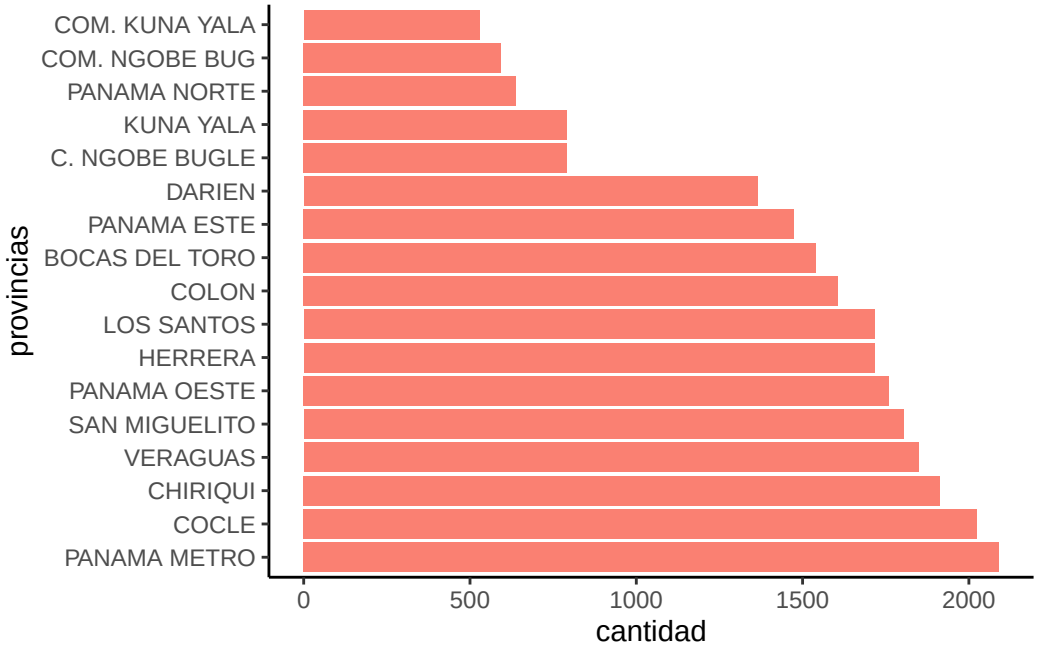


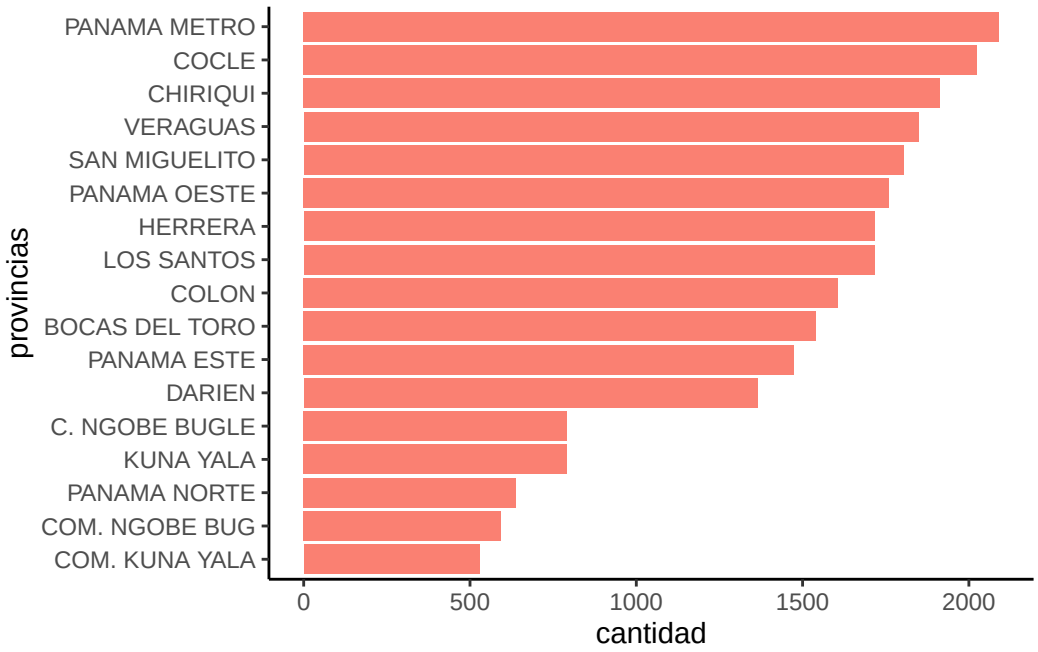
Gráfico de barra en ordenado INVERTIDO según levels() de la variable Region Para invertir el orden de los **levels()** de la variable región según orden del **tibble data**, debemos invertir los datos del vector **ordenRegion** utilizando la función **rev()**.

```
# Capturar datos del orden de la variable Region
# Invertir orden de vector Región, de menor a mayor
ordenRegion <- rev(data$Region)
# Ordenar niveles de variable categorica
data$Region<- factor(data$Region,
                      levels = ordenRegion)
```

```r
#Niveles ordenados a la inversa
levels(data$Region)
```

```
 [1] "COM. KUNA YALA" "COM. NGOBE BUG" "PANAMA NORTE"    "KUNA YALA"
 [5] "C. NGOBE BUGLE" "DARIEN"         "PANAMA ESTE"     "BOCAS DEL TORO"
 [9] "COLON"          "LOS SANTOS"     "HERRERA"         "PANAMA OESTE"
[13] "SAN MIGUELITO"  "VERAGUAS"       "CHIRIQUI"        "COCLE"
[17] "PANAMA METRO"
```

```r
# Crear gráfico 3
data |>
  ggplot(aes(x=Region, y=n)) +
  geom_col(fill="salmon") +
  coord_flip()+
  labs(x="provincias",
       y="cantidad")+
  guides(fill="none")+  # eliminar leyenda
  theme_classic()
```

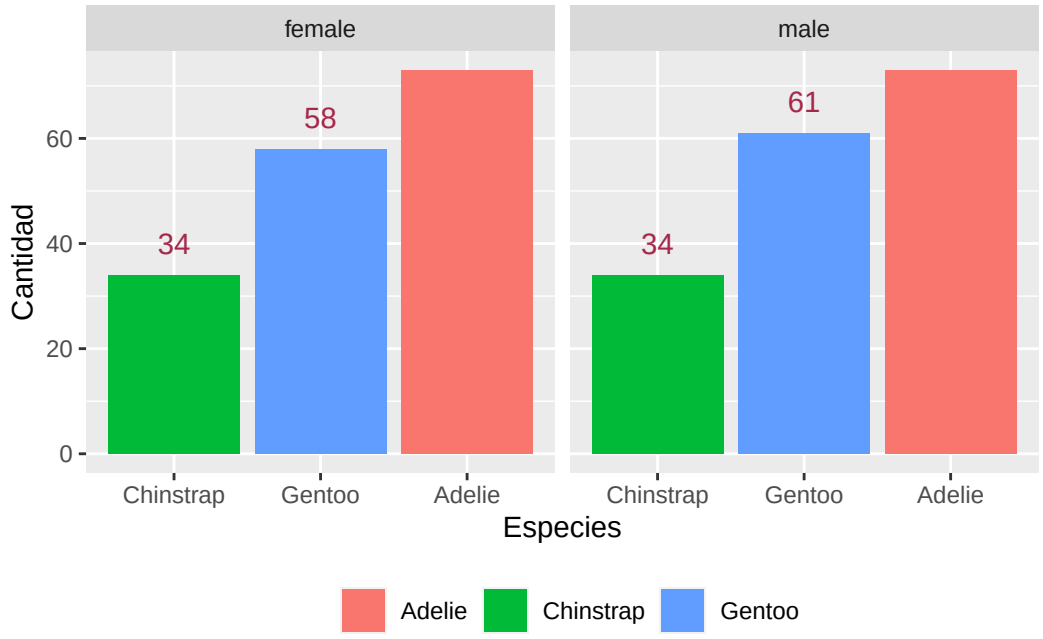


7.8 Facetas

7.8 Facetas de gráficos por columnas

Si queremos utilizar dos variables categóricas en el gráfico, sin utilizar barras apiladas o agrupadas podemos hacer uso de las **facetas**. Las facetas es una forma de crear gráficos del mismo tipo separados por una variable categórica, utilizando una nueva capa en ggplot llamada **facet_grid(variable)**, donde en este caso separaremso los gráficos e barras de **species** por la variable **sex** en la faceta. El formato de la faceta **facet_grid(.~sex)** indica al ggplot que los gráficos generados por la variable **sex** se muestren por columnas, esde decir , un gráfico al lado del otro.

```
datos_pinguinos_clean |>
 ggplot(aes(fct_rev(fct_infreq(species)),
            fill=species)) +
 geom_bar()+
 geom_text(
   aes(label = after_stat(count)
       ),
   stat = "count" ,
   vjust = -1.0,
   colour="#a62749"
 )+
 theme(legend.position = "bottom")+
 labs(fill="",
      x="Especies",
      y="Cantidad") +
 # coord_cartesian(ylim=c(0,160))+
 facet_grid(.~sex)
```

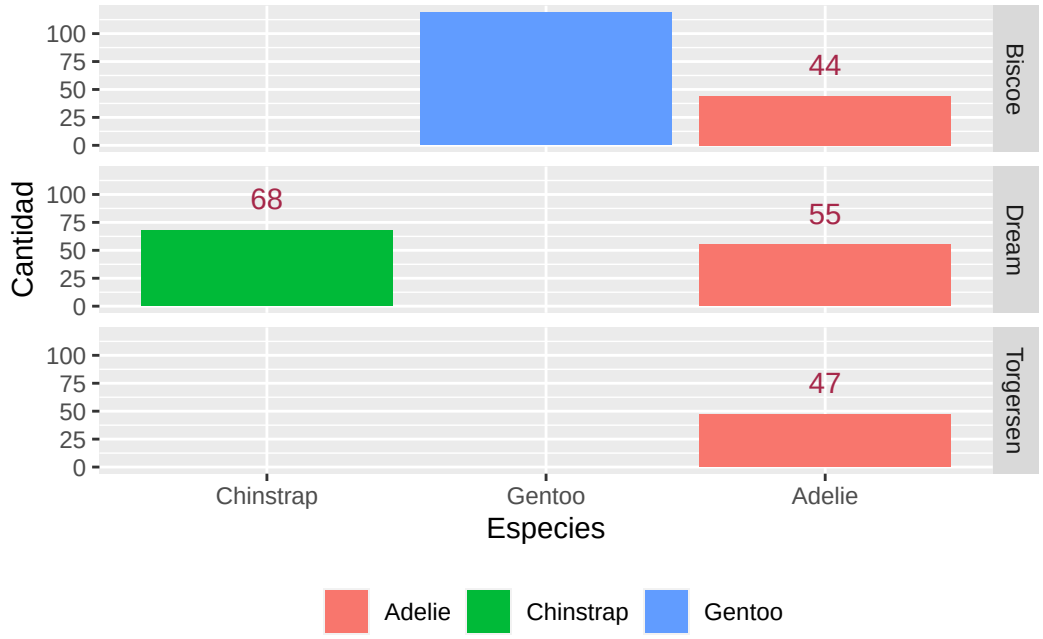


7.8 Faceta de gráficos por filas

El formato de la faceta **facet_grid(island~.)** indica al ggplot que los gráficos generados por la variable **island** se muestren por filas, es decir , un gráfico debajo del otro.

```
# Facetas de gráficos por filas
 datos_pinguinos_clean |>
  ggplot(aes(fct_rev(fct_infreq(species)),
             fill=species)) +
  geom_bar()+
  geom_text(
    aes(label = after_stat(count)
        ),
```

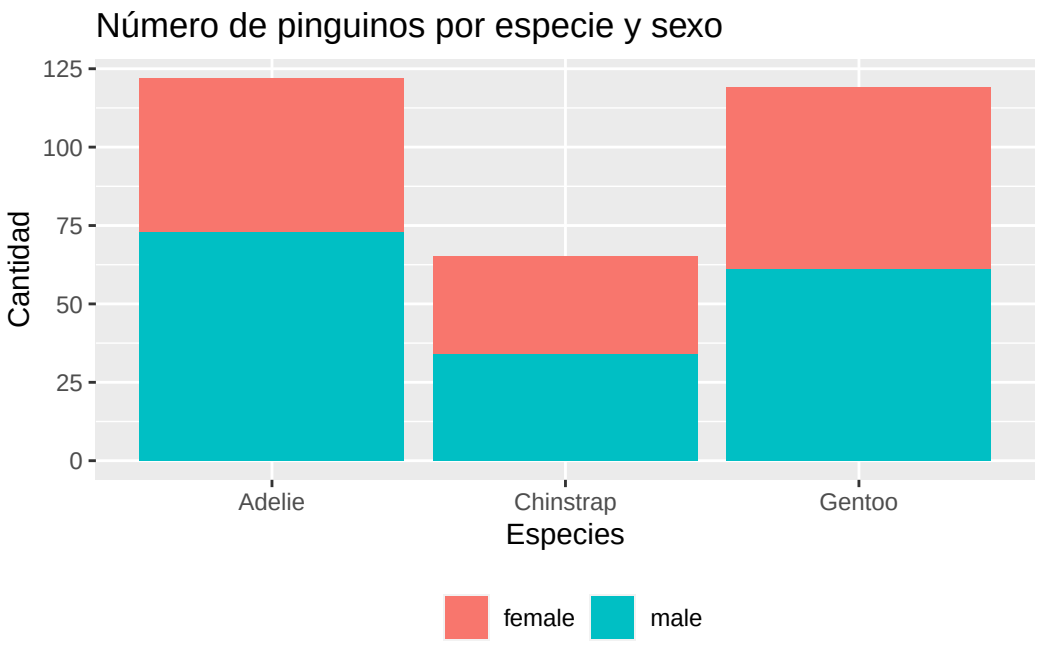
```
  stat = "count" ,
  vjust = -1.0,
  colour="#a62749"
)+
theme(legend.position = "bottom")+
labs(fill="",
     x="Especies",
     y="Cantidad") +
facet_grid(island~.)
```



7.9 Práctica

Crear gráfico de barras apiladas utilizando los datos de **data_penguins_clean.csv**. Seguir los pasos de** Crear gráfico de Pastel en R **de este documento.** TIPS**: En el gráfico debe hacer usos de la variables** sex y especies con **filter(condicion)** donde **body_mass_g >3200**.

Gráfico de resultado:


Número de pinguinos por especie y sexo
Cantidad
125
100
75
50
25
0
Adelie
Chinstrap
Gentoo
Especies
female
male

8Gráficos de Pastel

8.1 Cargar librerias

Librerias necesarias que se deben haber instalado previamente con **install.packages(nombre del paquete)** o el panel **Package**.

```
library(tidyverse) # incluye ggplot2
library(readxl) # funciones para importar xlsx
library(janitor) # funciones de limpieza
library(patchwork) #combinar gráficos de ggplot
library(ggExtra)
library(ggthemes) # nuevas temas para los
library(plotly) #gráficos interactivos # remotes::install_github("plotly/plotly")
library(tibble)
library(skimr) # reseumen numerico
library(modeest)
library(ggrepel) # añadir etiquetas a los gráficos
library(RColorBrewer) #paletas de colores
library(plotly) #graficos interactivos
```

8.2 Carga de datos

Se utilizará los datos de estudio de lectura y matematica de las escuelas de Miami en 2012 y 2013 por grado y Distrito llamado **SchoolsMiamiDade**.

Significado de las variables:

SchoolName : Nombre de la Escuela BoardDistrict : Núemero de Dsitrito donde esta la escuela SchoolGrade : Grado al que se le aplico la prueba Reading2012 : puntaje de 1 a 100 en Lectura 2012 Reading2013: puntaje de 1 a 100 en Lectura 2013 ReadingDifference: diferencia entre 2013 y 2012 en lectura Math2012: puntaje de 1 a 100 en Matemática 2012 Math2013: puntaje de 1 a 100 en Matemática 2013 MathDifference: diferencia entre 2013 y 2012 en matemática

```
SchoolsMiamiDade<- read.csv("data/SchoolsMiamiDade.csv")
SchoolsMiamiDade <- as_tibble(SchoolsMiamiDade)
head(SchoolsMiamiDade,5)
```

```
# A tibble: 5 x 9
  SchoolName BoardDistrict SchoolGrade Reading2012 Reading2013 ReadingDifference
  <chr>      <chr>         <chr>              <int>       <int>             <int>
1 0041 AIR ~ (9)           A                     82          80                -2
2 0070 CORA~ (9)           A                     71          73                 2
3 0071 EUGE~ (5)           A                     69          69                 0
4 0072 SUMM~ (9)           B                     57          50                -7
5 0073 MAND~ (9)           C                     34          32                -2
# i 3 more variables: Math2012 <int>, Math2013 <int>, MathDifference <int>
```

```r
# Verificar tipos de datos
SchoolsMiamiDade |>
  glimpse()
```

```
Rows: 460
Columns: 9
$ SchoolName         <chr> "0041 AIR BASE ELEMENTAR", "0070 CORAL REEF MONT AC"~
$ BoardDistrict      <chr> "(9)", "(9)", "(5)", "(9)", "(9)", "(2)", "(4)", "(3~
$ SchoolGrade        <chr> "A", "A", "A", "B", "C", "F", "A", "A", "A", "C", "D~
$ Reading2012        <int> 82, 71, 69, 57, 34, 28, 68, 73, 68, 39, 38, 45, 53, ~
$ Reading2013        <int> 80, 73, 69, 50, 32, 29, 70, 72, 68, 32, 41, 35, 51, ~
$ ReadingDifference <int> -2, 2, 0, -7, -2, 1, 2, -1, 0, -7, 3, -10, -2, -1, --
$ Math2012           <int> 71, 64, 66, 50, 38, 26, 68, 78, 73, 41, 43, 59, 56, ~
$ Math2013           <int> 75, 56, 64, 54, 39, 47, 66, 77, 76, 39, 47, 50, 55, ~
$ MathDifference     <int> 4, -8, -2, 4, 1, 21, -2, -1, 3, -2, 4, -9, -1, -3, --
```

8.3 Gráficos de Pastel en R

Los gráficos de pastel, pie o circulares ayudan a mostrar proporciones y porcentajes entre categorías al dividir un círculo en segmentos proporcionales. Cada longitud de arco representa una proporción de cada categoría, mientras que el círculo completo representa el 100%.

Este tipo de gráficos son ideales para dar al lector una idea rápida de la distribución proporcional de los datos. Sin embargo tiene varias desventajas:

- No pueden mostrar muchos valores, porque a medida que aumenta el número de valores, el tamaño de cada segmento es más pequeño y dificil de interpretar.
- Ocupan más espacio que sus alternativas y la interpretación del gráfico no siempre es fácil de realizar, como el gráfico de barra.
- No son buenos para realizar comparaciones entre grupos de gráficos circulares.

El uso del Gráfico de Pastel puede ser un problema ya que las personas se les dificulta leer ángulos, por lo que al cerebro le cuesta interpretar el gráfico de forma rápida. El comentario no está relacionado ha no utilizarlo, sino a la idea de evaluar la necesidad de utilizarlo y tomar en cuenta recomendaciones de cuando utilizarlo.

8.4 Crear gráfico de Pastel en R

En **ggplot** no hay una geometría para realizar un gráfico de pastel, sin embargo, aunque el proceso de su creación resulte un poco tedioso, es una forma de entender de donde se deriva el gráfico de barra y su posible sustituto en algunos casos **Los pasos para hacer un gráfico de pastel son los siguientes:**

8.4 Crear una tabla de datos ordenados

```r
tb_distritos<- SchoolsMiamiDade |>
               count(BoardDistrict) |>
               arrange(desc(n))
```

```
tb_distritos_<- SchoolsMiamiDade |>
             group_by(BoardDistrict, SchoolGrade) |>
            summarise(n = n())
```

```
`summarise()` has grouped output by 'BoardDistrict'. You can override using the
`.groups` argument.
```

```
# Verificar tipos de datos
tb_distritos |>
  glimpse()
```

```
Rows: 11
Columns: 2
$ BoardDistrict <chr> "(9)", "(2)", "(1)", "(6)", "(4)", "(8)", "(5)", "(7)", ~
$ n             <int> 76, 75, 53, 53, 47, 47, 41, 37, 28, 2, 1
```

8.4 Transformar tabla a tibble (opcional)

El conjunto de datos a utilziar en el gráfico tambien puede ser un dataframe,

```
tb_distritos <- as_tibble(tb_distritos)
tb_distritos |>
  glimpse()
```

```
Rows: 11
Columns: 2
$ BoardDistrict <chr> "(9)", "(2)", "(1)", "(6)", "(4)", "(8)", "(5)", "(7)", ~
$ n             <int> 76, 75, 53, 53, 47, 47, 41, 37, 28, 2, 1
```

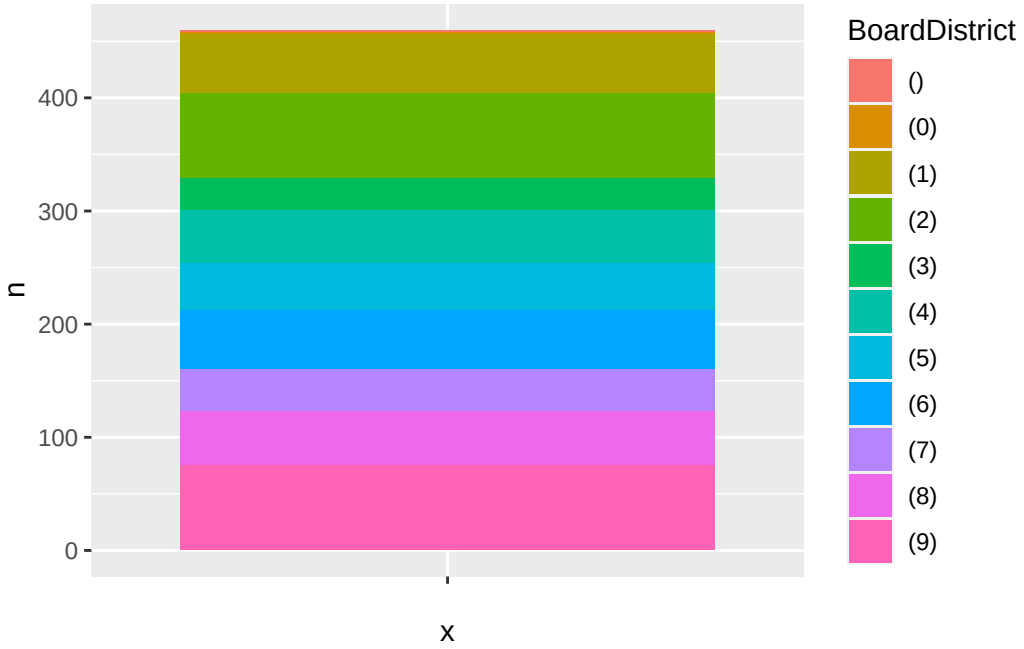
```
# se generan dos variables una cuantitativa y una cualitativa
# La varibale cualitativa debe ser categorica tipo factor()

tb_distritos$BoardDistrict <- as.factor(tb_distritos$BoardDistrict)

tb_distritos |>
  glimpse()
```

```
Rows: 11
Columns: 2
$ BoardDistrict <fct> (9), (2), (1), (6), (4), (8), (5), (7), (3), (), (0)
$ n             <int> 76, 75, 53, 53, 47, 47, 41, 37, 28, 2, 1
```

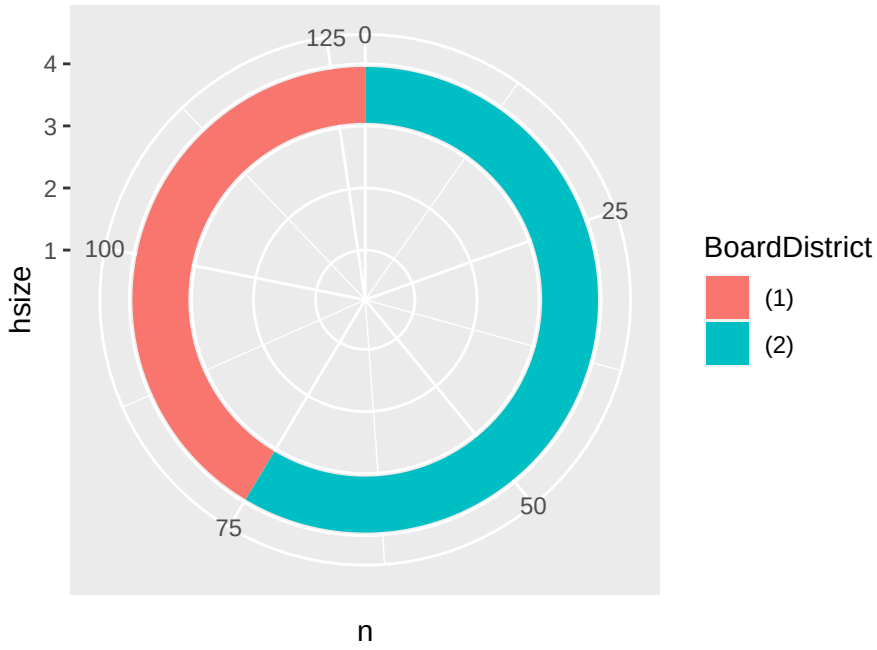
8.4 Crear gráfico de barra

Partimos creando un gráfico de barras con dos variables, **x=“ “** que es la posición en X en el gráfico, pero que se coloca solo para poder generar una sola columna, **y = n** , muestran los calores cuantitativo del conteo de cada **Distrito** apilados, el valor de **fill = BoardDistrict** es una Variable categorica que genera el color de cada pila de la columna.

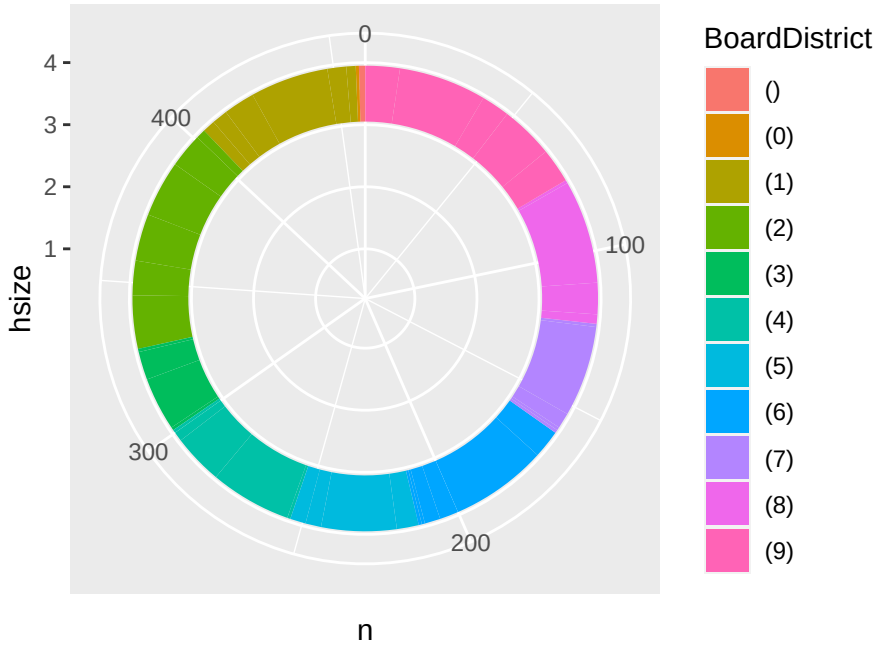
```
tb_distritos |>
ggplot(aes(x="", y= n, fill=BoardDistrict)) +
    geom_col()
```




```
#ejemplo circular - anillo
# Tamaño del agujero central
hsize <- 3.5
# Crear tamaño del agujero del anillo en el tibble
tb_distritos <- tb_distritos |>
                mutate(x = hsize)

tb_distritos[c(2,3),] |>
ggplot(aes(x=hsize, y= n, fill=BoardDistrict)) +
    geom_col() +
    coord_polar("y", start=0) +
    xlim(c(0.2, hsize+0.5))
```



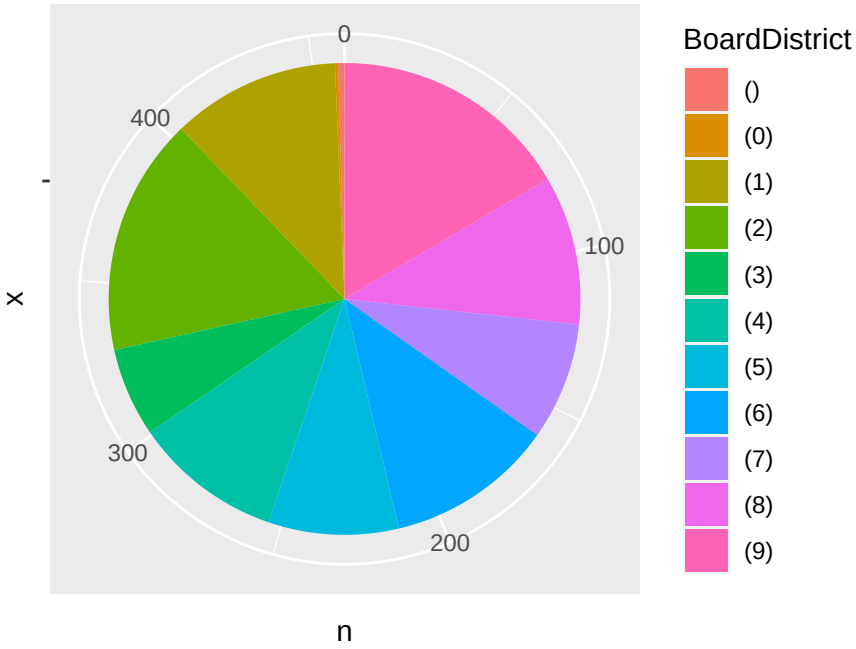

```
# otra tabla
tb_distritos_ |>
ggplot(aes(x=hsize, y= n, fill=BoardDistrict)) +
    geom_col() +
    coord_polar("y", start=0) +
    xlim(c(0.2, hsize+0.5))
```



8.4 Añadir coordenadas polares al gráfico

Una vez creado el gráfico de barras, generamos la forma circular utilizando coordenadas polares a través de la función **coord_polar("y", start=0)** , la cual debe inciar con el ángulo **0** y terminará en el ángulo **360**..

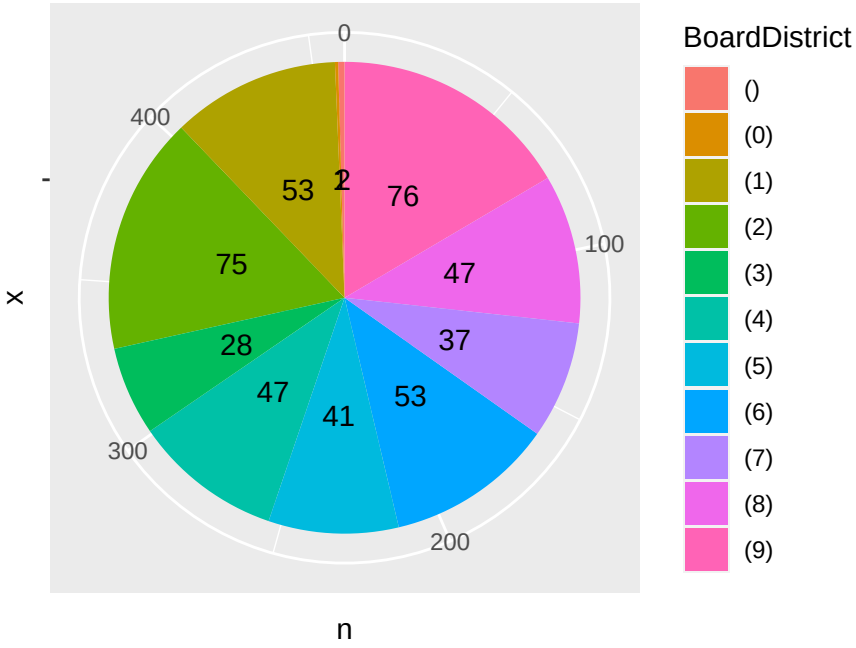
```
tb_distritos |>
ggplot(aes(x="", y= n, fill=BoardDistrict)) +
    geom_col()+
    coord_polar("y", start=0)
```



8.4 Añadimos los datos al grafico

Con la geometria **geom_text()** añadimos los textos de cada variable **n** y ajustamos la posición de los datos con **vjust()**.

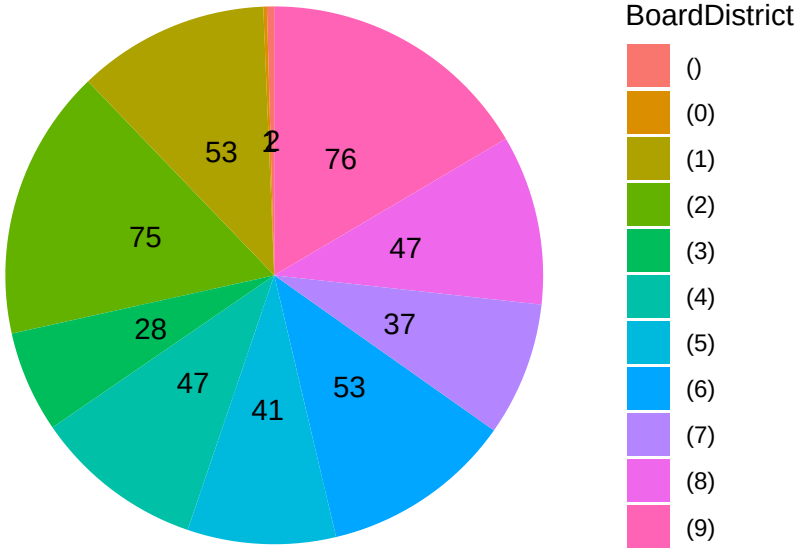
```
tb_distritos |>
ggplot(aes(x="", y= n, fill=BoardDistrict)) +
    geom_col()+
    coord_polar("y", start=0) +
    geom_text(aes(label =n),position = position_stack(vjust = 0.5))
```



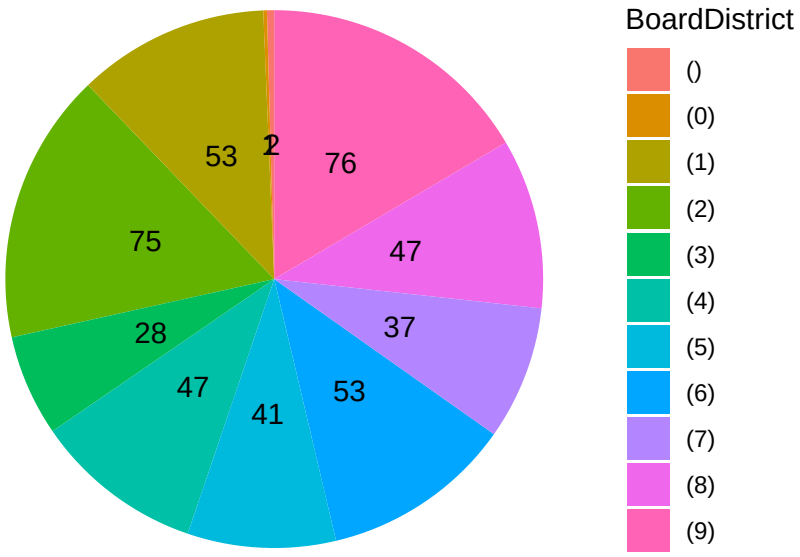
8.4 Eliminamos elementos generados de las coordenadas polares

Con la función **theme_void()** eliminamos los componentes visuales generados por las coordenadas polares, que no aportan nada al gráfico.

```
tb_distritos |>
ggplot(aes(x="", y= n, fill=BoardDistrict)) +
    geom_col()+
    coord_polar("y", start=0) +
    geom_text(aes(label =n),position = position_stack(vjust = 0.5)) +
    theme_void()
```



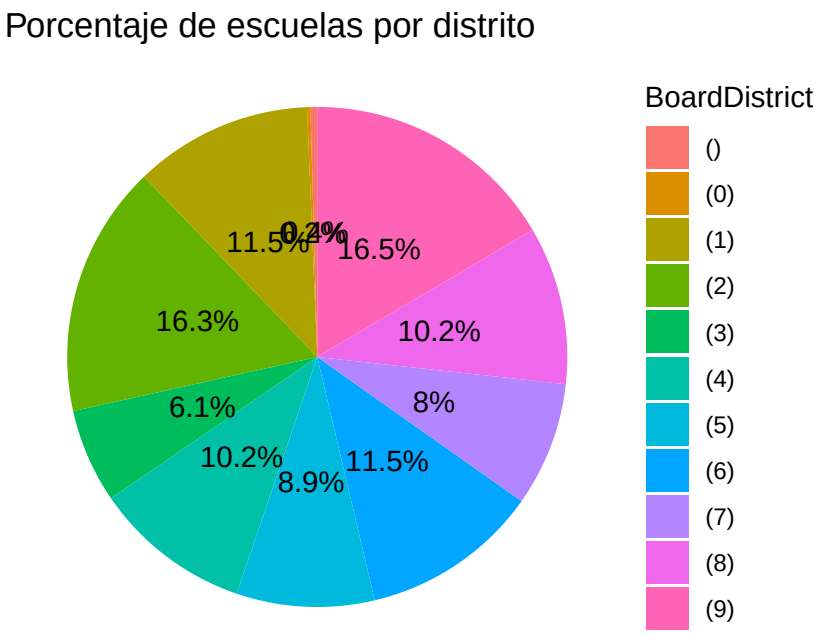
```
tb_distritos |>
ggplot(aes(x="", y= n, fill=BoardDistrict)) +
    geom_col()+
    coord_polar("y", start=0) +
    geom_text(aes(label =n),position = position_stack(vjust = 0.5)) +
    theme_void()
```



8.5 Gráfico de pastel con porcentaje

Para calcaluar el porcentaje de cada **slide** del gráfico de pastel se debe conocer el total de valores de la variable **n**, el cual calcularemos en la variable **suma**. Este valor lo utilizaremos para dividir cada valor de la variaqble **n** y multiplicado por **100**. Utilizamos la función **round()**, para redondear y usamos un solo digito decima. Creamos una nueva variable en el tibble llamado **porcentaje** la cual añadimos al tibble **tb_distritos** utilizando la función **mutate()**.

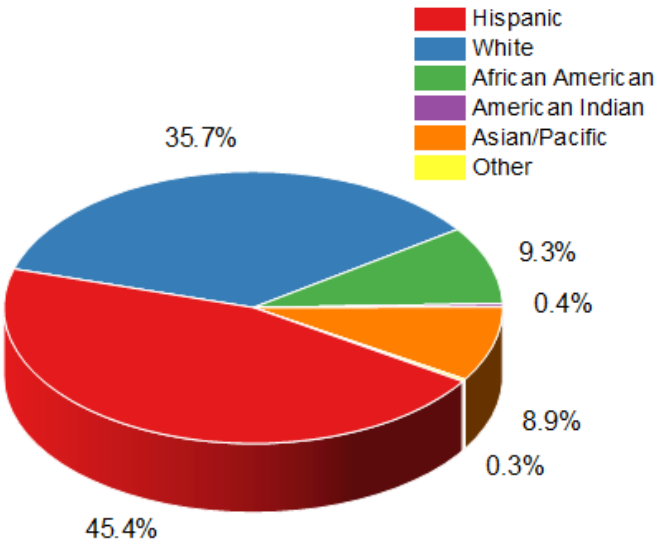
```
#calcular variable porcentaje
suma <- sum(tb_distritos$n)
tb_distritos <- tb_distritos |>
                mutate(porcentaje = round((n /suma) * 100,1))

# gráfico de pie
tb_distritos |>
ggplot(aes(x="", y= n, fill=BoardDistrict)) +
    geom_col()+
    coord_polar("y", start=0) +
    geom_text(aes(label = paste0(porcentaje, "%")),
              position = position_stack(vjust = 0.5)) +
    theme_void()+
    labs(title = "Porcentaje de escuelas por distrito") #opcional
```



8.6 Recomendaciones

8.6 recomendación 0

El objetivo de los gráficos es trasmitir información a través de este, no hay que adornar los datos, **NO utilice efectos 3D** en los gráficos, menos en el gráfico de pie, a las personas le cuestat interpretar ángulos, se hace más complicado añadir interpretar figuras en perspectiva.

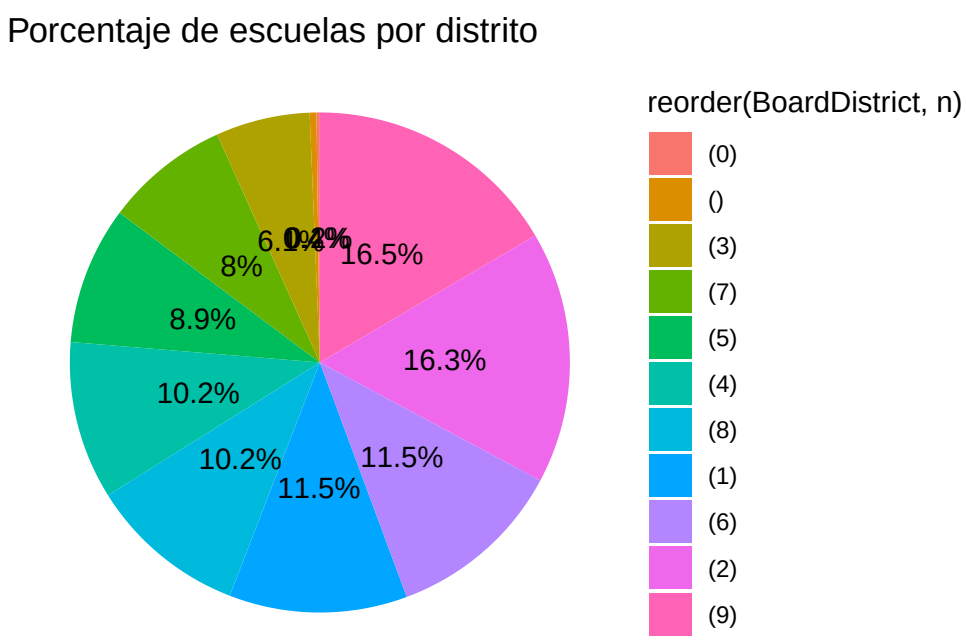


8.6 Recomendación 1

Orden de los slide Si bien las etiquetas de los datos se solapan, es un tema que veremos posteriormente. El **error común** en los gráficos de pastel es que **los pedazos del pastel o slide aveces no están en orden**. Un gráfico de pastel **debiera iniciar con el slide más grande en la posición**

12 del reloj, seguido de los slide por orden de tamaño. En este caso aunque inicia en la posición 12 del reloj, es el slide de la posición 2 con el valor 10.2% el que esta de primero, y no debe estar en esa posición. Para order los slide en el gráfico, debemos ordenar la variable categórica **Board_district**, uitilizando la función **reorder()** en esa variable, la cual ordenara la variable categorica **BoardDistrict** por el valor de **n**, el cual esta ordenado de mayor a menor.

```
tb_distritos |>
ggplot(aes(x="", y= n, fill= reorder(BoardDistrict,n))) +
    geom_col()+
    coord_polar("y", start=0) +
    geom_text(aes(label = paste0(porcentaje, "%")),
              position = position_stack(vjust = 0.5)) +
    theme_void()+
    labs(title = "Porcentaje de escuelas por distrito") #opcional
```



8.6 Recomendación 2

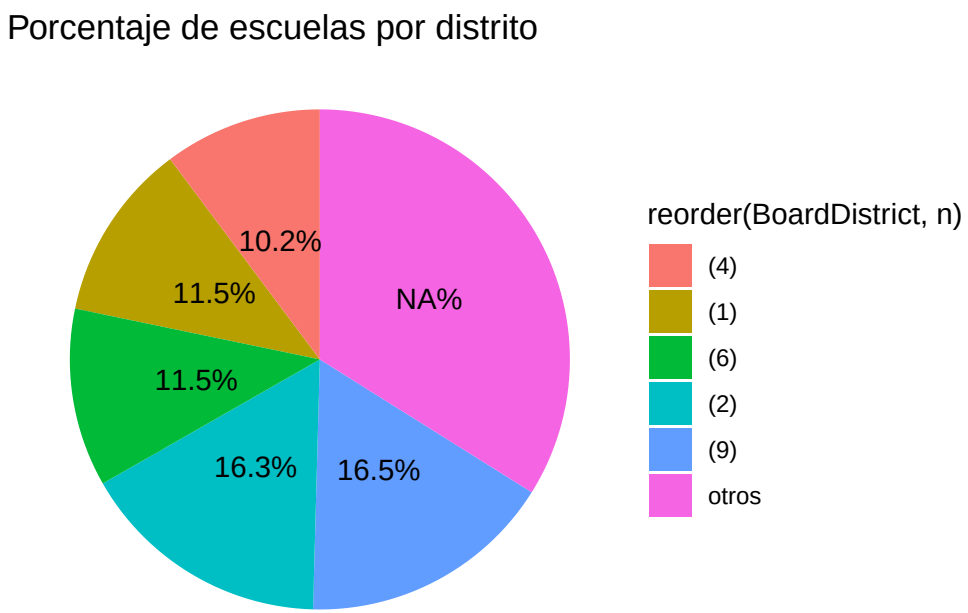
Número de slide en el gráfico de pastel Según expertos en VIZ al utilizar un gráfico de pastel, es recomendable que el número de slide no sea mayor a **seis (6)**. En caso de ser así, se recomienda unir los valores más pequeños en un solo slide llamado **otros / others**. En este ejemplo el número de slide del pie de escuela por distritos en USA, es de **11**, así que trabajaremos en reducir este número a solo 6. **Nota:** *Esta es solo una RECOMENDACIÓN en la visualización de datos al utilizar gráfico de pie*

```
# Crear nueva fila de valor "otros" seleccionando las filas
# de las 6 a  la 11 para luego sumar los valores de n

n <- sum(tb_distritos$n[6:11])
suma <- sum(tb_distritos$n)
BoardDistrict <- "otros"
x <- round(n/suma *100,1)
# Creamos un dataframe con los valores calculados
```

```r
other_row <- data.frame(BoardDistrict,n,x)
other_row$BoardDistrict <- as.factor(other_row$BoardDistrict)

# Crear nueva tabla con los seis valores seleccionados
# de los distritos con más escuelas, inlcuyendo "otros"
other_distritos <- tb_distritos[1:5,]
# other_distritos |>
#   glimpse()

tb_distritos_ <- bind_rows(other_distritos, other_row)
```

Gráfico de pastel con slide ordenados

```r
tb_distritos_ |>
ggplot(aes(x="", y= n, fill= reorder(BoardDistrict,n))) +
    geom_col()+
    coord_polar("y", start=0) +
    geom_text(aes(label = paste0(porcentaje, "%")),
              position = position_stack(vjust = 0.5)) +
    theme_void()+
    labs(title = "Porcentaje de escuelas por distrito") #opcional
```

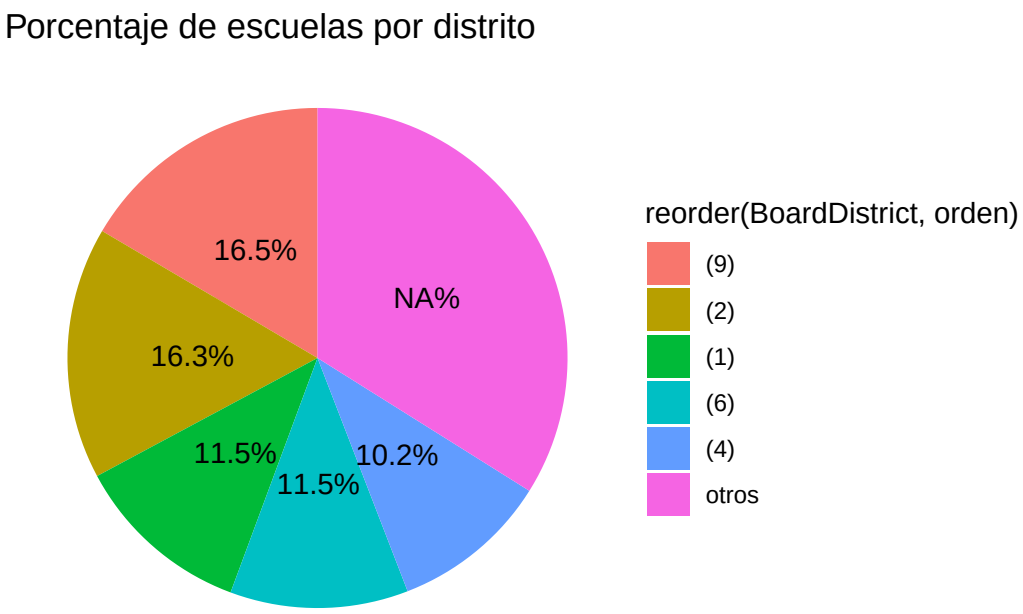


Ordenar slide según variable

El gráfico generado de los valores más altos, ahora coloca **otros** al inicio del gráfico, es debido a que contiene el porcentaje con mayorvalor que es **33.9%**, sin embargo, al ser el slide de datos restante es **recomendable** que este al final del resto de los slide. Existen diferentes formas de ordenarlo, en este lo haremos creando una nueva variable llamada **orden** que contiene los valores del 1 al 6 (cantidad de slide) en el orden en que queremos aparezcan los slides.

```r
# Crear varible orden con valores del 1 al 6
 tb_distritos_$orden <- 1:6
```

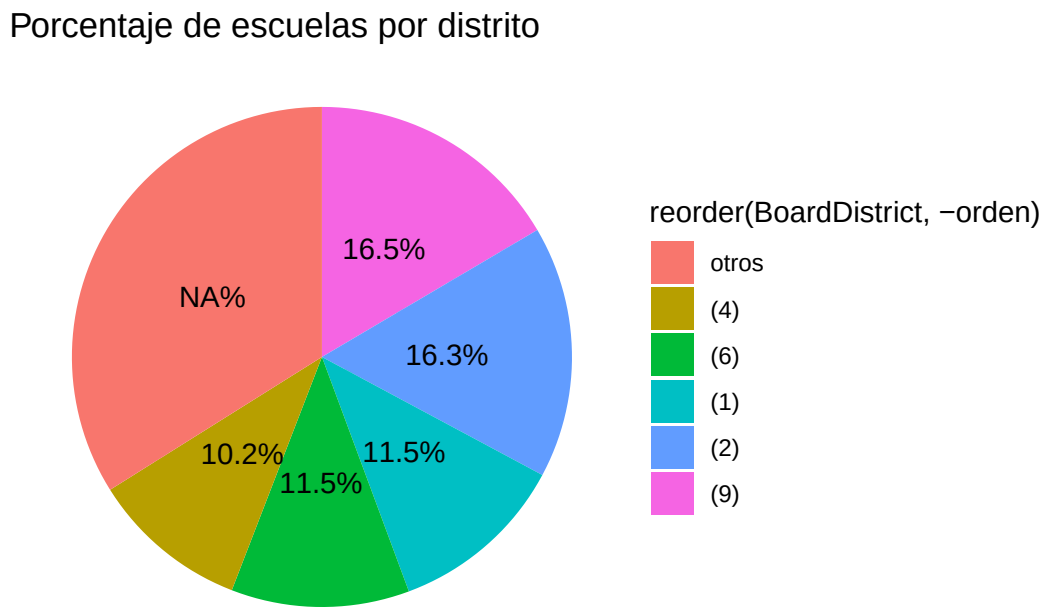
```
# En lugar de ordenar la función reorden por la variable "n"
# lo haremos con la variable "orden"
tb_distritos_ |>
ggplot(aes(x="", y= n, fill= reorder(BoardDistrict,orden))) +
    geom_col()+
    coord_polar("y", start=0) +
    geom_text(aes(label = paste0(porcentaje, "%")),
              position = position_stack(vjust = 0.5)) +
    theme_void()+
    labs(title = "Porcentaje de escuelas por distrito") #opcional
```



Invertir orden de slide en según variable

El gráfico ha sido ordenado por la variable **orden** pero a iniciado en las posición de las 12 de la manecilla del reloj, pero, en sentido contario a las manecillas, para modificarlo, debemos decirle a la variable **orden** que invierta su orden amteponiendo el signo negativo (-) en esa variable **-orden**. De esa forma obtenemos el orden del gráfico.

```
# Invertir orden de los valores según la posición negativa
# de la variable "orden"
tb_distritos_ |>
ggplot(aes(x="", y= n, fill= reorder(BoardDistrict, -orden))) +
    geom_col()+
    coord_polar("y", start=0) +
    geom_text(aes(label = paste0(porcentaje, "%")),
                  position = position_stack(vjust = 0.5)) +
    theme_void()+
    labs(title = "Porcentaje de escuelas por distrito")  #opcional
```



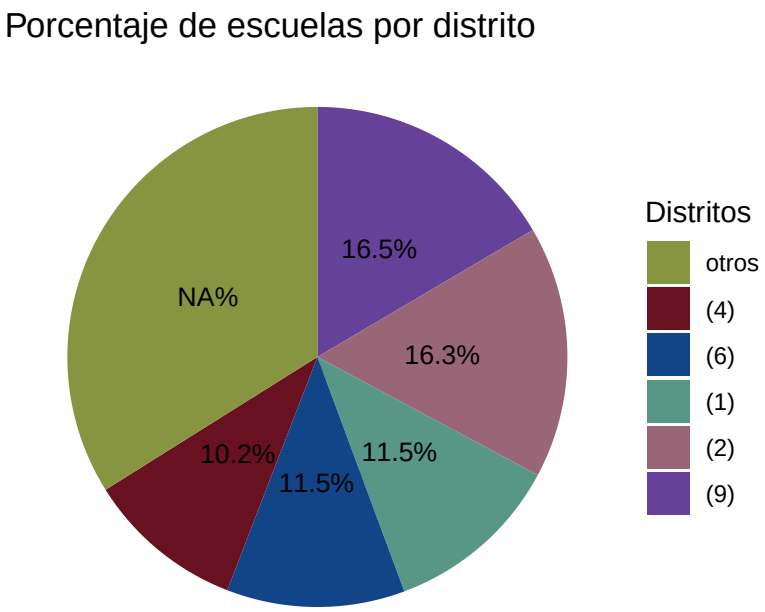
8.6 Recomendación 3

Colores no normalizados

Este error puede ser un poco relativo (diseñar es un arte, pero hay reglas), al utilizar la variable **BoardDistrict** en la propiedad **fill** en la estética de **aes() de ggplot**, este nos genera diversos colores por cada distrito, estos colores son un estandar de ggplot. El problema surge cuando creamos un gráfico con colores personalizados, sin contemplar que existes paletas de colores definidas.

A manera de ejemplo, crearemos un vector llamado **colores**, con diversos colores sin evaluar su combinación, los cuales se cargaran a traves de la función **scale_fill_manual(values = colores)**.

```
colores <- c("#879541","#681221","#114588","#589785","#996677","#664199")


tb_distritos_ |>
ggplot(aes(x="", y= n, fill= reorder(BoardDistrict, -orden))) +
    geom_col()+
    coord_polar("y", start=0) +
    geom_text(aes(label = paste0(porcentaje, "%")),
                  position = position_stack(vjust = 0.5),
                  size= 3.4) +
    theme_void()+
    labs(title = "Porcentaje de escuelas por distrito",
         fill="Distritos") + #opcional
    scale_fill_manual(values = colores)
```



8.6 Paleta de colores personalizada

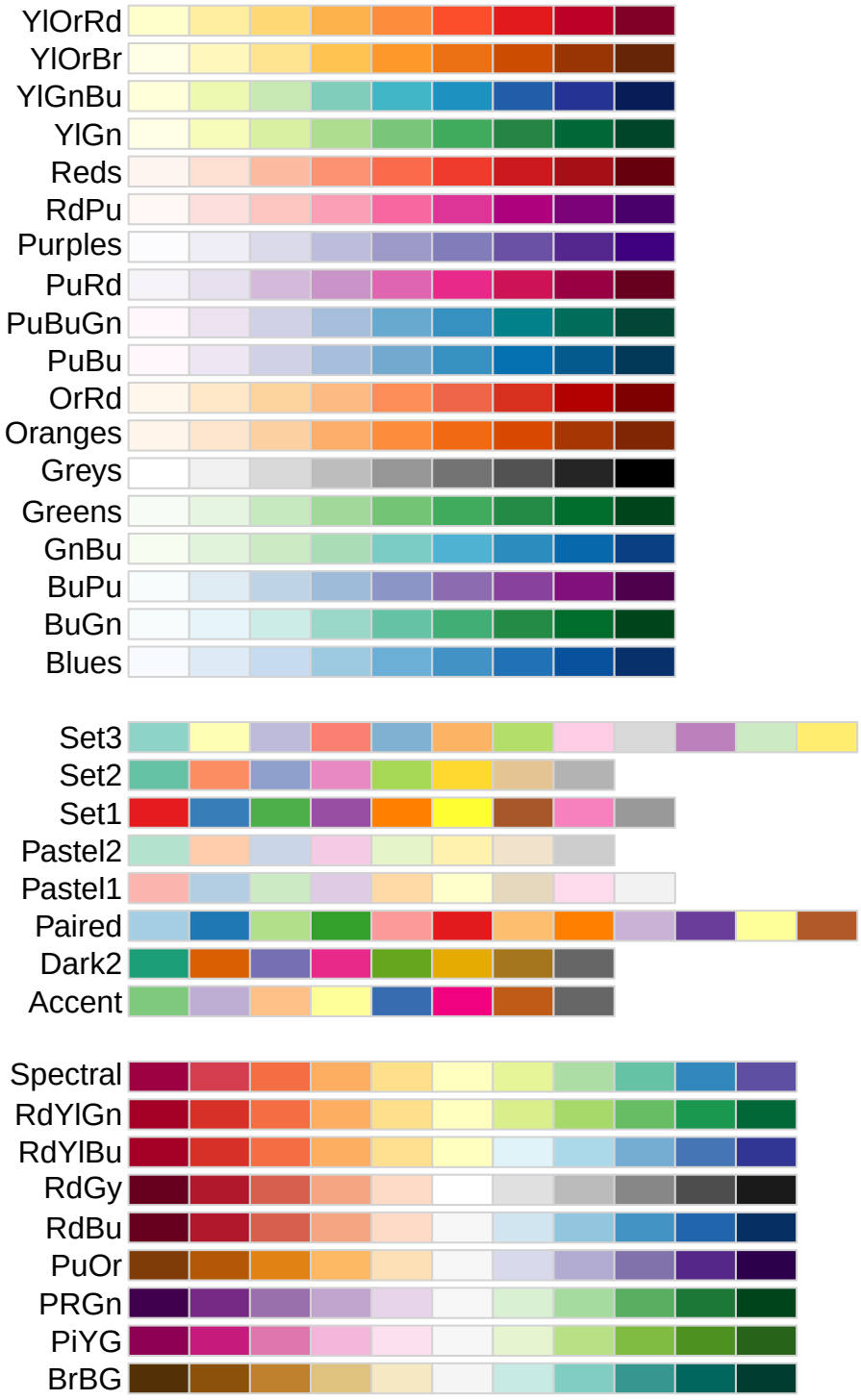
Existen diferentes tipo de paletas de colores en ggplot:

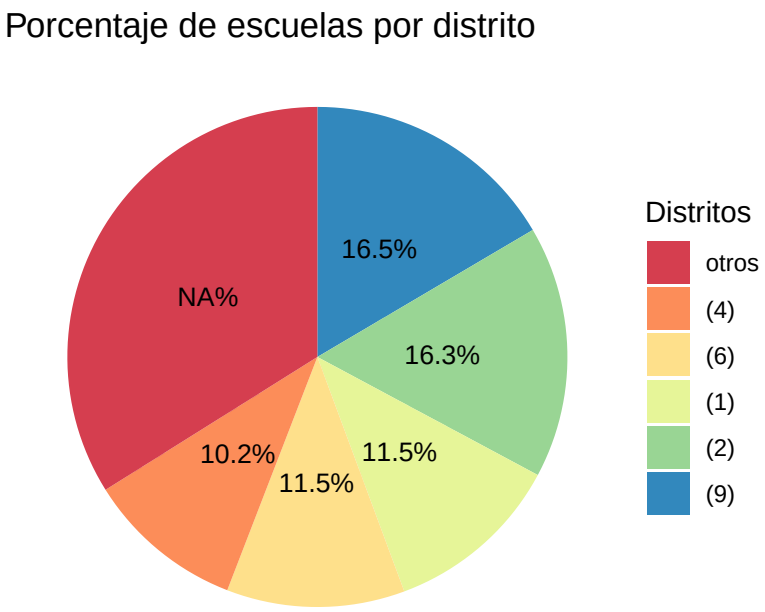
- *Diverging*: BrBG, PiYG, PRGn, PuOr, RdBu, RdGy, RdYlBu, RdYlGn, Spectral
- *Qualitative*: Accent, Dark2, Paired, Pastel1, Pastel2, Set1, Set2, Set3
- *Sequential*: Blues, BuGn, BuPu, GnBu, Greens, Greys, Oranges, OrRd, PuBu, PuBuGn, PuRd, Purples, RdPu, Reds, YlGn, YlGnBu, YlOrBr, YlOrRd

Mostrar paletas de colores

Colores divergentes Utilizamos la paleta **palette = "Spectral"** en la función **scale_fill_brewer**.

```
tb_distritos_ |>
ggplot(aes(x="", y= n, fill= reorder(BoardDistrict, -orden))) +
    geom_col()+
    coord_polar("y", start=0) +
    geom_text(aes(label = paste0(porcentaje, "%")),
                  position = position_stack(vjust = 0.5),
                  size= 3.4) +
    theme_void()+
       labs(title = "Porcentaje de escuelas por distrito",
         fill="Distritos") + #opcional
    scale_fill_brewer(palette = "Spectral")
```

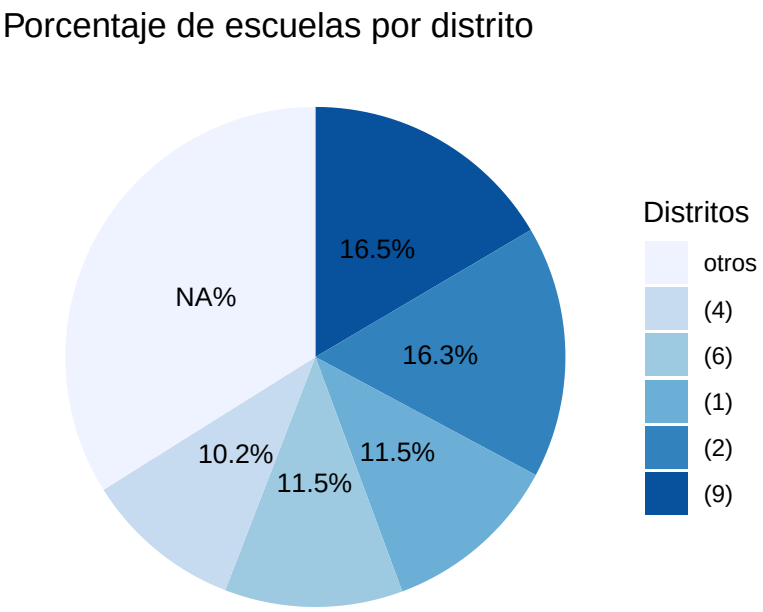

YlOrRd
YlOrBr
YlGnBu
YlGn
Reds
RdPu
Purples
PuRd
PuBuGn
PuBu
OrRd
Oranges
Greys
Greens
GnBu
BuPu
BuGn
Blues
Set3
Set2
Set1
Pastel2
Pastel1
Paired
Dark2
Accent
Spectral
RdYlGn
RdYlBu
RdGy
RdBu
PuOr
PRGn
PiYG
BrBG




```
# display.brewer.all(colorblindFriendly = TRUE)  # paleta de colores
# ?display.brewer.all(type="qual") # paleta de colores cualitativo
# brewer.pal(6,"Dark2") # paleta y numero de colores
```

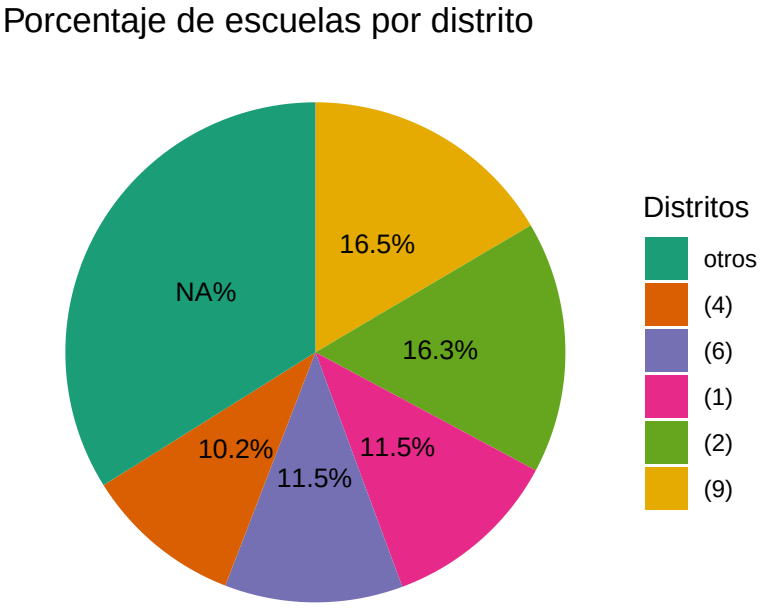
Colores sequenciales Utilizamos la paleta **palette = "Blues"** en la función **scale_fill_brewer**.

```
tb_distritos_ |>
ggplot(aes(x="", y= n, fill= reorder(BoardDistrict, -orden))) +
    geom_col()+
    coord_polar("y", start=0) +
    geom_text(aes(label = paste0(porcentaje, "%")),
                  position = position_stack(vjust = 0.5),
                  size= 3.4) +
    theme_void()+
       labs(title = "Porcentaje de escuelas por distrito",
         fill="Distritos") + #opcional
    scale_fill_brewer(palette = "Blues")
```



Colores Cualitativos Utilizamos la paleta **palette = "Dark2"** en la función **scale_fill_brewer**.

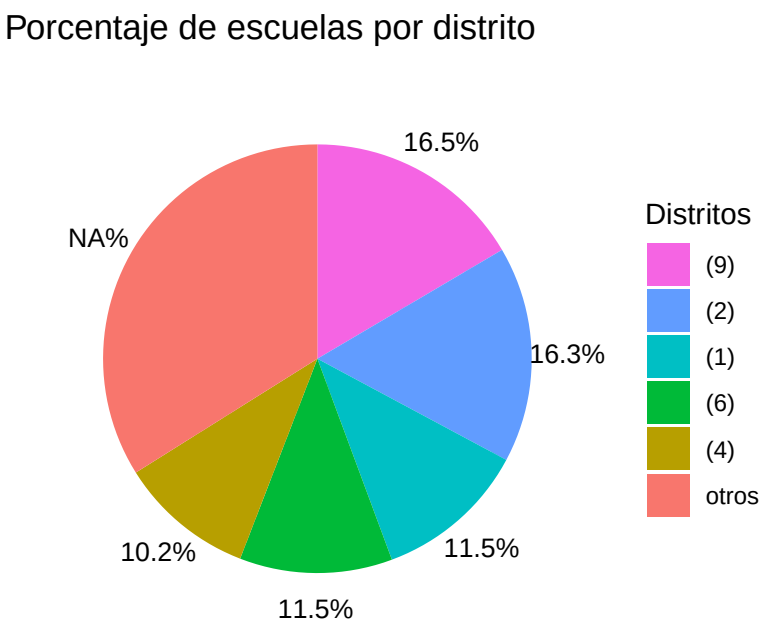
```
tb_distritos_ |>
ggplot(aes(x="", y= n, fill= reorder(BoardDistrict, -orden))) +
    geom_col()+
    coord_polar("y", start=0) +
    geom_text(aes(label = paste0(porcentaje, "%")),
                position = position_stack(vjust = 0.5),
                size= 3.4) +
    theme_void()+
       labs(title = "Porcentaje de escuelas por distrito",
         fill="Distritos") + #opcional
    scale_fill_brewer(palette = "Dark2")
```



8.6 Modificar apariencia del gráfico

Antes de ver otra recomendación en los colores del gráfico, modificaremos algunos elementos al gráfico como, añadir el valor de **x=1.6** en **goem_text()** para colocar los valores del texto fuera del gráfico. Utilizando **guides(fill = guide_legend(reverse = TRUE))** podemos modificar el orden de los items de la leyenda.

```
tb_distritos_ |>
ggplot(aes(x="", y= n, fill= reorder(BoardDistrict, -orden))) +
    geom_col()+
    coord_polar("y", start=0) +
    geom_text(aes(x=1.6,
                  label = paste0(porcentaje, "%")),
                  position = position_stack(vjust = 0.5),
                  size= 3.4) +
    theme_void()+
       labs(title = "Porcentaje de escuelas por distrito",
         fill="Distritos") + #opcional
   guides(fill = guide_legend(reverse = TRUE))
```

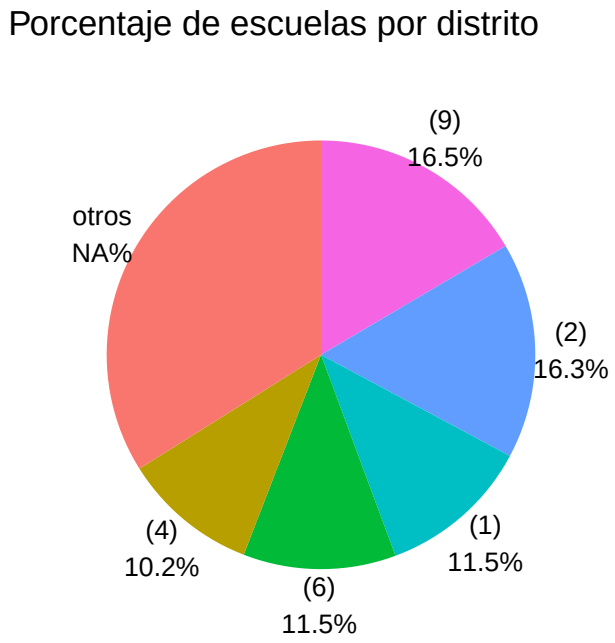


Eliminar leyenda, añadir info a datos

Podemos añadir el nombre del distrito **BoardDistrict** en la etiqueta de cada slide. Al crear esta información, la leyenda estaría de más por lo que la eliminaremos utilizando **theme(legend.position="none").**

```
tb_distritos_ |>
ggplot(aes(x="", y= n, fill= reorder(BoardDistrict, -orden))) +
    geom_col()+
    coord_polar("y", start=0) +
    geom_text(aes(x=1.6,
                  label = paste0(BoardDistrict,"\n",porcentaje, "%")),
                  position = position_stack(vjust = 0.5),
```

```
                size= 3.4) +
    theme_void()+
       labs(title = "Porcentaje de escuelas por distrito",
         fill="Distritos") + #opcional
   theme(legend.position="none")
```



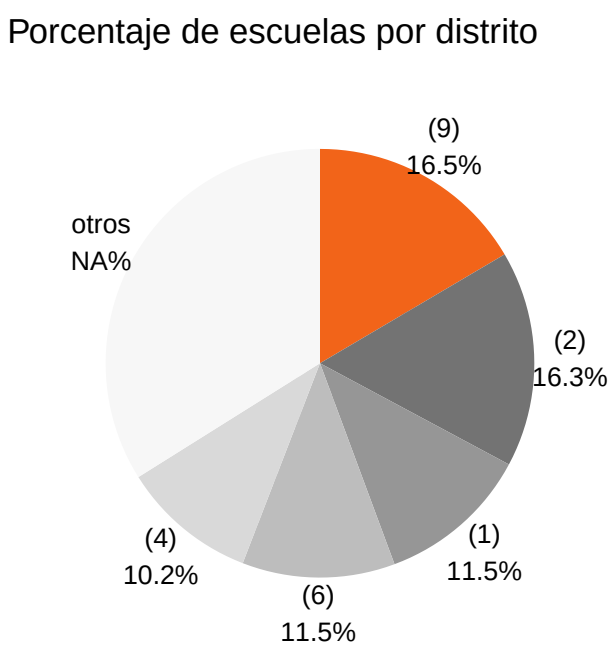
8.6 Recomendación 4

Énfasis en un slide, resaltar slide con mayor porcentaje

Una de las recomendaciones al realizar visualizaciones es guiar al lector en el aspecto a resaltar del gráfico, ya sea que queramos resaltar el slide con mayor porcentaje o el slide con menor porcentaje. Para ello podemos hacer uso del color para resaltar el slide de interés.

```
# Crear paleta de colores secuancial  gris
# brewer.pal permite capturar lista de colores de Greysorder
colores <- brewer.pal(7,"Greys")
# Crear color personalizado a posición de slide énfasis
colores[6] <- "#F26419"

# Invertir orden de los valores según la posición negativa
# de la variable "orden"
tb_distritos_ |>
ggplot(aes(x="", y= n, fill= reorder(BoardDistrict, -orden))) +
    geom_col()+
    coord_polar("y", start=0) +
    geom_text(aes(x=1.6,
                  label = paste0(BoardDistrict,"\n",porcentaje, "%")),
                  position = position_stack(vjust = 0.5),
                  size= 3.4) +
    theme_void()+
       labs(title = "Porcentaje de escuelas por distrito",
```

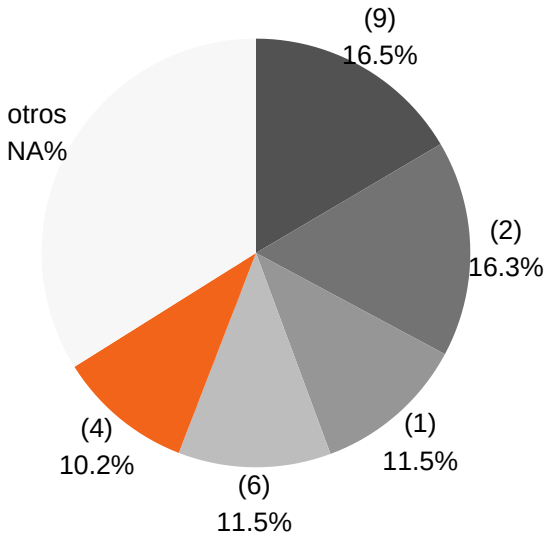
```
        fill="Distritos") + #opcional
    theme(legend.position="none") +
    scale_fill_manual(values = colores)
```



Énfasis en un slide, resaltar slide con menor porcentaje

```
#
colores <- brewer.pal(7,"Greys")
colores[2] <- "#F26419"

# Invertir orden de los valores según la posición negativa
# de la variable "orden"
tb_distritos_ |>
ggplot(aes(x="", y= n, fill= reorder(BoardDistrict, -orden))) +
    geom_col()+
    coord_polar("y", start=0) +
    geom_text(aes(x=1.6,
                  label = paste0(BoardDistrict,"\n",porcentaje, "%")),
                  position = position_stack(vjust = 0.5),
                  size= 3.4) +
    theme_void()+
       labs(title = "Porcentaje de escuelas por distrito",
         fill="Distritos") + #opcional
    theme(legend.position="none") +
    scale_fill_manual(values = colores)
```

Porcentaje de escuelas por distrito



9Gráficos de puntos o dispersión

9.1 Cargar librerias

Librerias necesarias que se deben haber instalado previamente con **install.packages(nombre del paquete)** o wn el panel **Package**.

```
library(tidyverse) # incluye ggplot2
library(readxl) # funciones para importar xlsx
library(janitor) # funciones de limpieza
library(patchwork) #combinar gráficos de ggplot
library(ggExtra)
library(ggthemes) # nuevas temas para los
library(plotly) #gráficos interactivos # remotes::install_github("plotly/plotly")
library(tibble)
library(skimr) # reseumen numerico
library(modeest)
library(ggrepel) # añadir etiquetas a los gráficos
library(RColorBrewer) #paletas de colores
library(plotly) #graficos interactivos
library(corrplot)
```

9.2 Carga de datos

Se utilizará los datos de estudio de lectura y matemática de las escuelas de Miami en 2012 y 2013 por grado y Distrito llamado **SchoolsMiamiDade**.

Significado de las variables del dataframe SchoolsMiamiDade Escuelas de Miami con notas promedio en las materias de lectura y matemática en los años 2012 y 2013.

SchoolName : Nombre de la Escuela BoardDistrict : Número de Distrito donde esta la escuela SchoolGrade : Grado al que se le aplicó la prueba Reading2012 : puntaje de 1 a 100 en Lectura 2012 Reading2013: puntaje de 1 a 100 en Lectura 2013 ReadingDifference: diferencia entre 2013 y 2012 en lectura Math2012: puntaje de 1 a 100 en Matemática 2012 Math2013: puntaje de 1 a 100 en Matemática 2013 MathDifference: diferencia entre 2013 y 2012 en matemática

```
SchoolsMiamiDade<- read.csv("data/SchoolsMiamiDade.csv")
SchoolsMiamiDade <- as_tibble(SchoolsMiamiDade)
head(SchoolsMiamiDade,5)
```

```
# A tibble: 5 x 9
  SchoolName BoardDistrict SchoolGrade Reading2012 Reading2013 ReadingDifference
  <chr>      <chr>         <chr>              <int>       <int>             <int>
1 0041 AIR ~ (9)           A                     82          80                -2
2 0070 CORA~ (9)           A                     71          73                 2
3 0071 EUGE~ (5)           A                     69          69                 0
4 0072 SUMM~ (9)           B                     57          50                -7
5 0073 MAND~ (9)           C                     34          32                -2
# i 3 more variables: Math2012 <int>, Math2013 <int>, MathDifference <int>
```

```
# Verificar tipos de datos
SchoolsMiamiDade |>
  glimpse()
```

```
Rows: 460
Columns: 9
$ SchoolName        <chr> "0041 AIR BASE ELEMENTAR", "0070 CORAL REEF MONT AC"~
$ BoardDistrict     <chr> "(9)", "(9)", "(5)", "(9)", "(9)", "(2)", "(4)", "(3~
$ SchoolGrade       <chr> "A", "A", "A", "B", "C", "F", "A", "A", "A", "C", "D~
$ Reading2012       <int> 82, 71, 69, 57, 34, 28, 68, 73, 68, 39, 38, 45, 53, ~
$ Reading2013       <int> 80, 73, 69, 50, 32, 29, 70, 72, 68, 32, 41, 35, 51, ~
$ ReadingDifference <int> -2, 2, 0, -7, -2, 1, 2, -1, 0, -7, 3, -10, -2, -1, --
$ Math2012          <int> 71, 64, 66, 50, 38, 26, 68, 78, 73, 41, 43, 59, 56, ~
$ Math2013          <int> 75, 56, 64, 54, 39, 47, 66, 77, 76, 39, 47, 50, 55, ~
$ MathDifference    <int> 4, -8, -2, 4, 1, 21, -2, -1, 3, -2, 4, -9, -1, -3, --
```

```
SchoolsMiamiDade$BoardDistrict <- as.factor(SchoolsMiamiDade$BoardDistrict)
SchoolsMiamiDade$SchoolGrade <- as.factor(SchoolsMiamiDade$SchoolGrade)

# ver resumen de datos
SchoolsMiamiDade |>
  summary()
```

```
 SchoolName        BoardDistrict SchoolGrade  Reading2012      Reading2013
Length:460         (9)    : 76    : 55       Min.   :  0.00   Min.   : 0.00
Class :character   (2)    : 75   A:204       1st Qu.: 36.00   1st Qu.:37.00
Mode  :character   (1)    : 53   B: 85       Median : 51.00   Median :52.50
                   (6)    : 53   C: 77       Mean   : 50.98   Mean   :52.08
                   (4)    : 47   D: 31       3rd Qu.: 67.00   3rd Qu.:68.00
                   (8)    : 47   F:  7       Max.   :100.00   Max.   :97.00
                   (Other):109   I:  1       NA's   :17
ReadingDifference    Math2012         Math2013      MathDifference
Min.   :-17.000   Min.   :  0.00   Min.   :  0.00   Min.   :-25.0000
1st Qu.: -2.000   1st Qu.: 41.00   1st Qu.: 40.00   1st Qu.: -5.0000
Median :  1.000   Median : 53.00   Median : 52.50   Median : -1.0000
Mean   :  1.113   Mean   : 53.26   Mean   : 52.29   Mean   : -0.7046
3rd Qu.:  4.000   3rd Qu.: 69.00   3rd Qu.: 67.00   3rd Qu.:  4.0000
Max.   : 33.000   Max.   :100.00   Max.   :100.00   Max.   : 25.0000
NA's   :17        NA's   :91       NA's   :78       NA's   :91
```

```
# pairs(SchoolsMiamiDade[, c(4:8)])
# cor(na.omit(SchoolsMiamiDade[, c(4:8)]))
```

9.3 Gráficos de Dispersión y puntos en R

Los gráficos de dispersión (gráfico de puntos, diagrama de XY) utilizan dos variables cuantitativas para generar una colección de puntos usando coordenadas cartesianas para mostrar estos valores. Al mostrar una variable en cada eje, se puede detectar si existe una relación o correlación entre las dos variables.

9.4 Gráfico de puntos

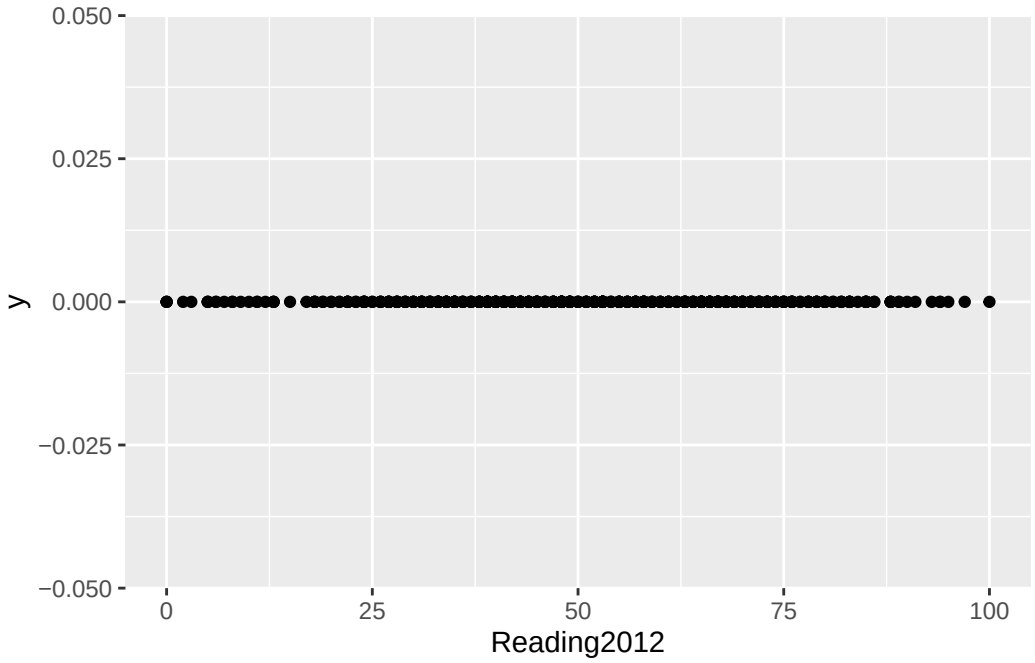
Se pueden construir gráfico de dispersión con ggplot2 utilizando la geometría **geom_point()** el cual requiere de 2 variables, por lo regular cuantitativa. Sin embargo, tambien es posible generar este tipo de gráficos con una variable númerica **Reading2012** y una variable **BoardDistrict** tipo categorica.

9.4 Una variable cuantitativa y una cualitativa

El gráfico muestra la posición de cada escuela según puntaje, por cada distrito **Reading2012**.

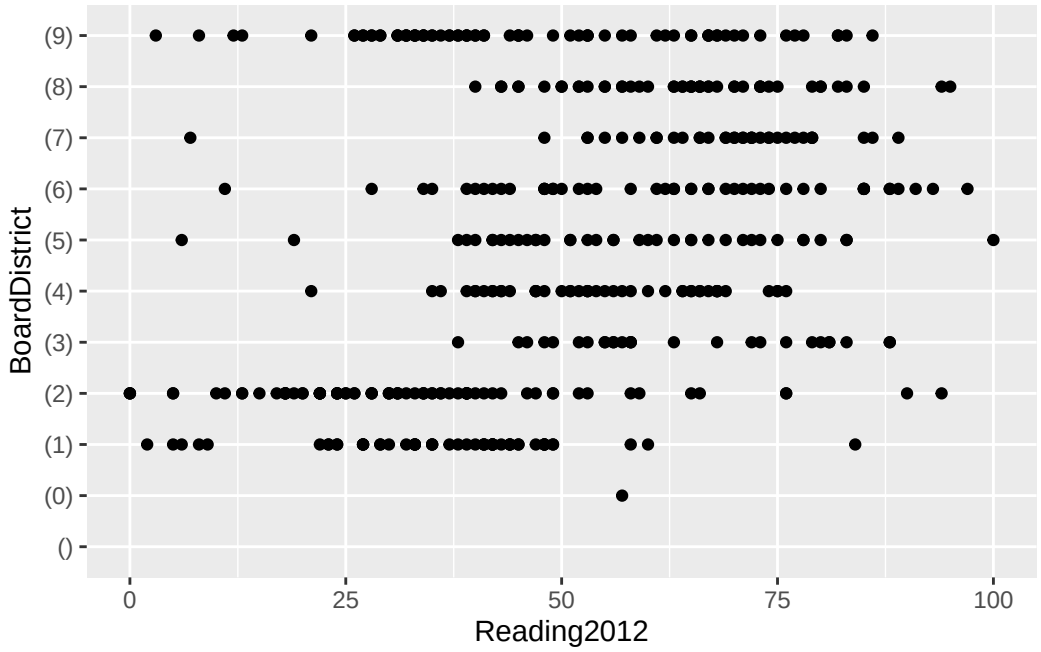
```
# grafico de punto con una sola variable
SchoolsMiamiDade |>
  ggplot(aes(Reading2012,y=0))+
  geom_point()
```

```
Warning: Removed 17 rows containing missing values (`geom_point()`).
```



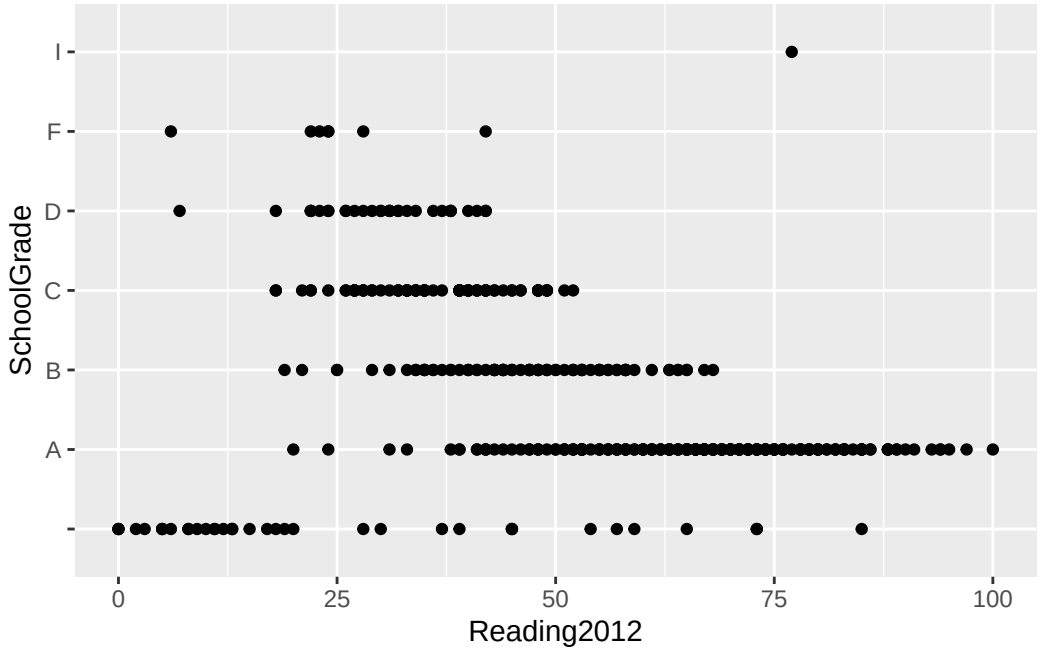

```
#grafico de puntos con dos variables una cuantitativa y otra cualitativa
# Materia de lectura 2012 en los diferentes distritos
SchoolsMiamiDade |>
  ggplot(aes(Reading2012,BoardDistrict))+
  geom_point()
```

```
Warning: Removed 17 rows containing missing values (`geom_point()`).
```




```
# Materia de lectura 2012 en los diferentes grados
SchoolsMiamiDade |>
ggplot(aes(Reading2012,SchoolGrade))+
  geom_point()
```

```
Warning: Removed 17 rows containing missing values (`geom_point()`).
```

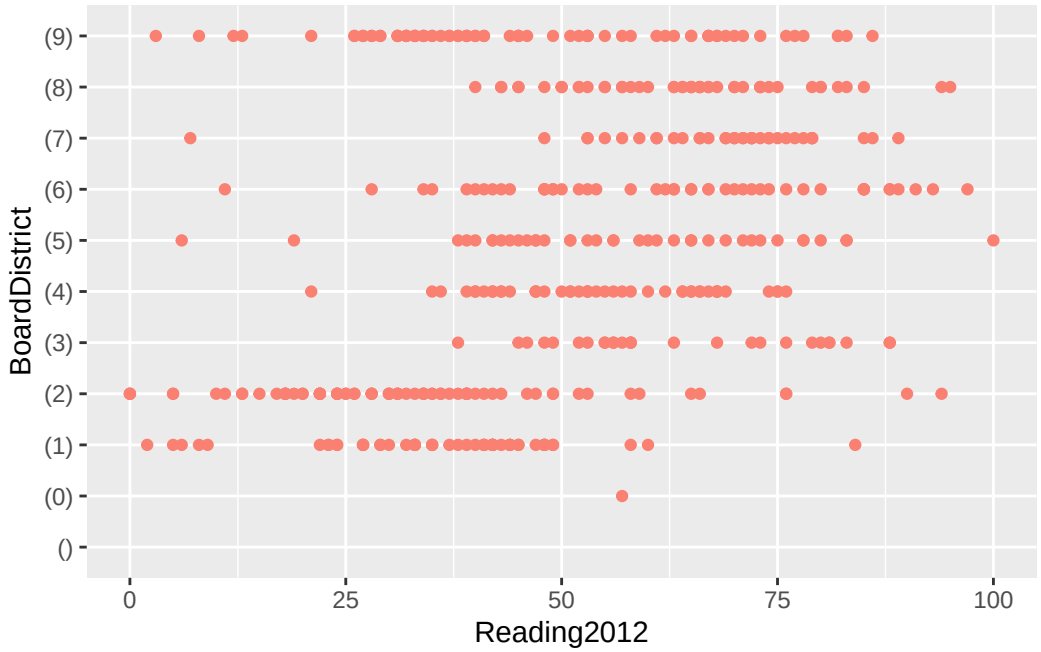


9.4 Color en los puntos del gráfico

Para modificar el color de los puntos en el gráfico podemos utilizar la propiedad **fill** en la geometria de puntos.

```
SchoolsMiamiDade |>
  ggplot(aes(Reading2012,BoardDistrict))+
  geom_point(color="salmon")
```

```
Warning: Removed 17 rows containing missing values (`geom_point()`).
```



¿Cuántas escuelas están por encima de de la nota minima de 71 (como ejemplo) en lectura y matemática 2012

Para responde esta pregunta utilizaremos la función **filter(Reading2012 >= 71)** para crear el filtro de las escuelas con puntaje mayor o igual a 71. El resultado es, lectura = 87, matemática = 74.

```
# filtro y conteo de datos en lectura 2012
SchoolsMiamiDade |>
  filter(Reading2012 >= 71) |>
  nrow()
```

```
[1] 87
```

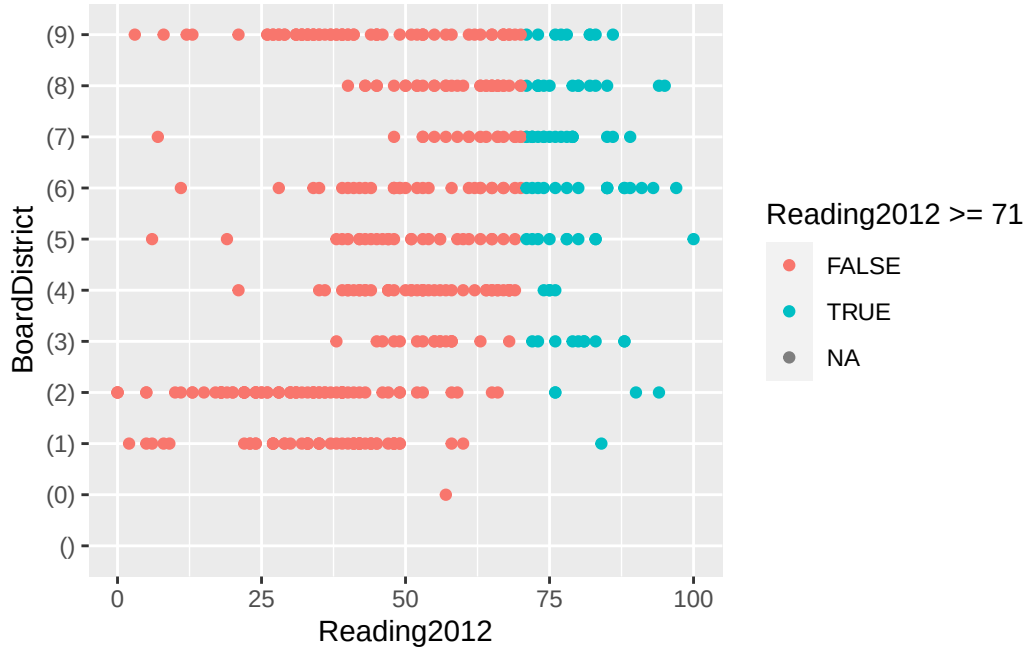
```
# filtro y conteo de datos en matematica 2012
SchoolsMiamiDade |>
  filter(Math2012 >= 71) |>
  nrow()
```

```
[1] 74
```

Representación de gráfico de dispersión a la pregunta anterior Para colorear las escuelas según puntaje mayor a 71 podemos utilizar la misma condición del filtro **Reading2012 >= 71** usado en el código anterior para buscar los valores, este filtro lo podemos colocar en la propiedad **color** de la geometria **geom_point(aes(color=Reading2012 >= 71))** . De esta forma **ggplot** hará la separación de colores por nosotros.

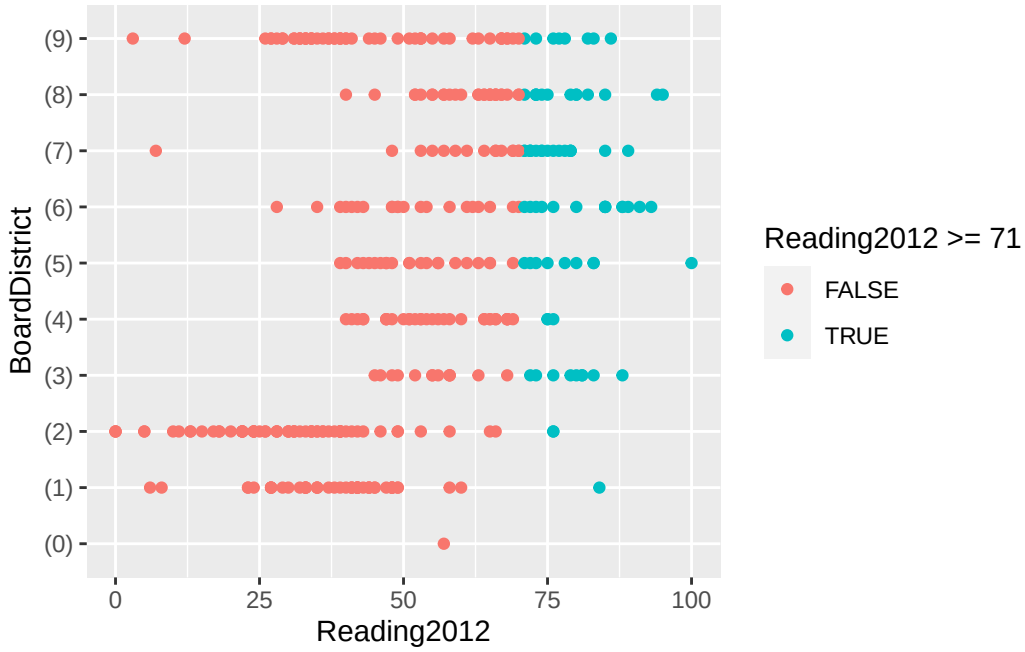
```
# Gráfico con dos variables y colorear con filtro
# Lectura 2012
SchoolsMiamiDade |>
ggplot(aes(Reading2012,BoardDistrict))+
  geom_point(aes(color=Reading2012 >= 71))
```

```
Warning: Removed 17 rows containing missing values (`geom_point()`).
```



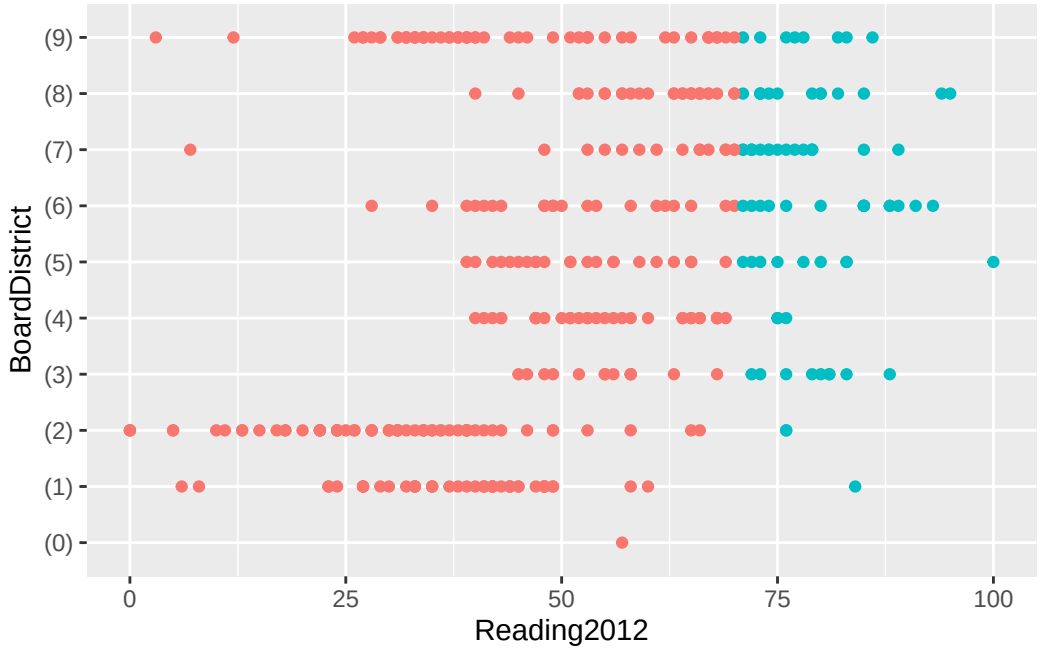
Eliminar valor vacios en el gráfico El gráfico anterior coloca en la leyenda los colores de los valores mayores a 71 como **TRUE**, los menores a 71 como **FALSE**, pero tambien coloca el valor de **NA** que representan los valores nulos. Para eliminar estos valores podemos filtrar los datos previos al uso de ggplot utilizando **na.omit(Reading2012)** otra opción es utilizar **filter(Reading2012 !=" ")**.

```
# Gráfico - se eliminan datos vacios
SchoolsMiamiDade |>
  na.omit(Reading2012) |>
  ggplot(aes(Reading2012,BoardDistrict))+
    geom_point(aes(color=Reading2012 >= 71))
```



Eliminar leyenda Si deseamos eliminar la leyenda de color (que no aporta nada), podemos colocar al final la capa **guides(color = FALSE)**.

```
# Gráfico - se elimina leyenda de colores de datos
SchoolsMiamiDade |>
  na.omit(Reading2012) |>
  ggplot(aes(Reading2012,BoardDistrict))+
    geom_point(aes(color=Reading2012 >= 71))+
  guides(color = "none")
```



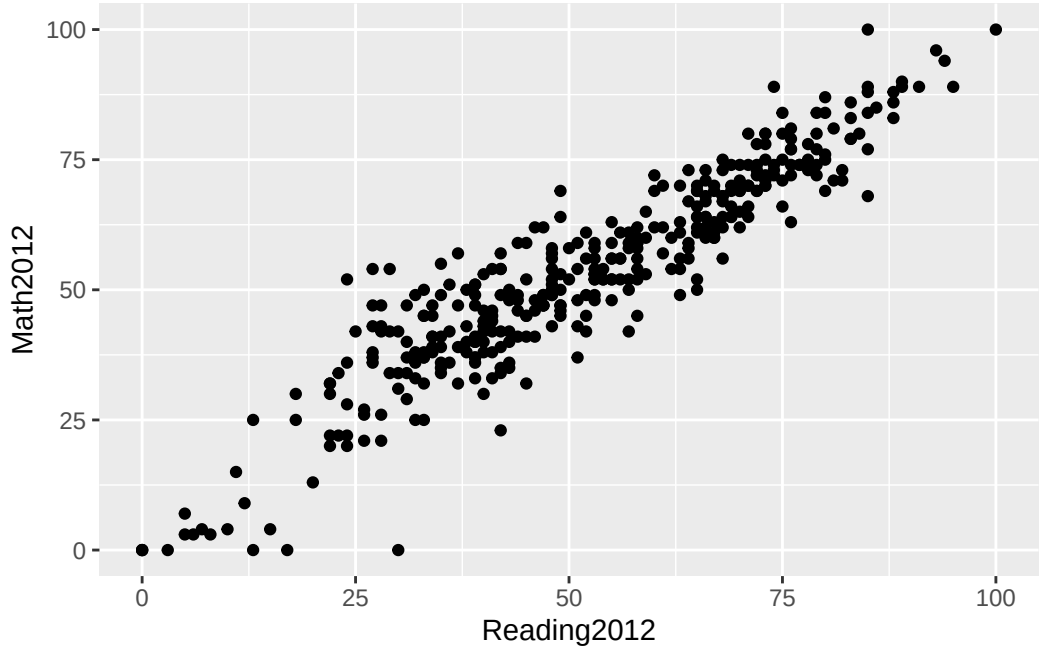
9.5 Gráfico de dispersión

9.5 Dos variables numéricas

Utilizamos las variables relacionadas con los puntajes de matemática 2012 y lectura 2012, ambos datos cuantitativos. El gráfico se puede interpretar una correlación fuerte positiva (los valores de ambas variables van en aumento).

```
# Gráfico con dos variables numericas
SchoolsMiamiDade |>
  ggplot(aes(Reading2012,Math2012))+
  geom_point()
```

```
Warning: Removed 91 rows containing missing values (`geom_point()`).
```



Modificar color, forma y tamaño en los puntos

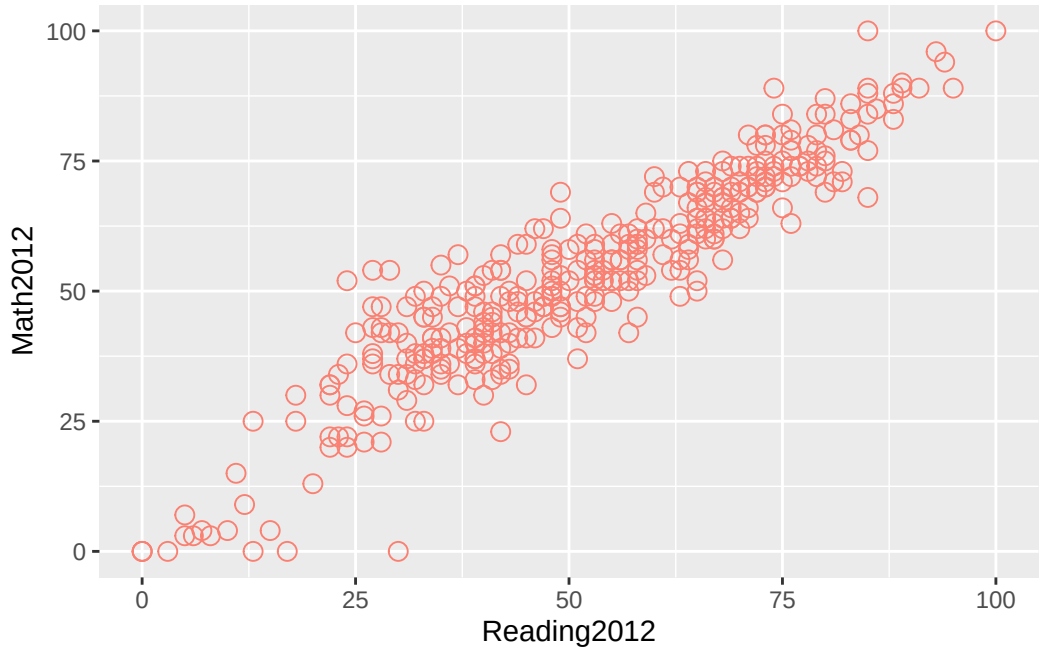
Las formas **shape** del los puntos pueden ser valor de 1 a 25.

- shape = 0, square
- shape = 1, circle
- shape = 2, triangle point up
- shape = 3, plus
- shape = 4, cross
- shape = 5, diamond
- shape = 6, triangle point down
- shape = 7, square cross
- shape = 8, star

- shape = 9, diamond plus
- shape = 10, circle plus
- shape = 11, triangles up and down
- shape = 12, square plus
- shape = 13, circle cross
- shape = 14, square and triangle down
- shape = 15, filled square
- shape = 16, filled circle
- shape = 17, filled triangle point-up
- shape = 18, filled diamond
- shape = 19, solid circle
- shape = 20, bullet (smaller circle)
- shape = 21, filled circle blue
- shape = 22, filled square blue
- shape = 23, filled diamond blue
- shape = 24, filled triangle point-up blue
- shape = 25, filled triangle point down blue

```
# Gráfico con dos variables numericas y un solo color en los puntos, forma =1
SchoolsMiamiDade |>
  ggplot(aes(Reading2012,Math2012))+
  geom_point(color="salmon",
             size=3,
             shape=1)
```

```
Warning: Removed 91 rows containing missing values (`geom_point()`).
```



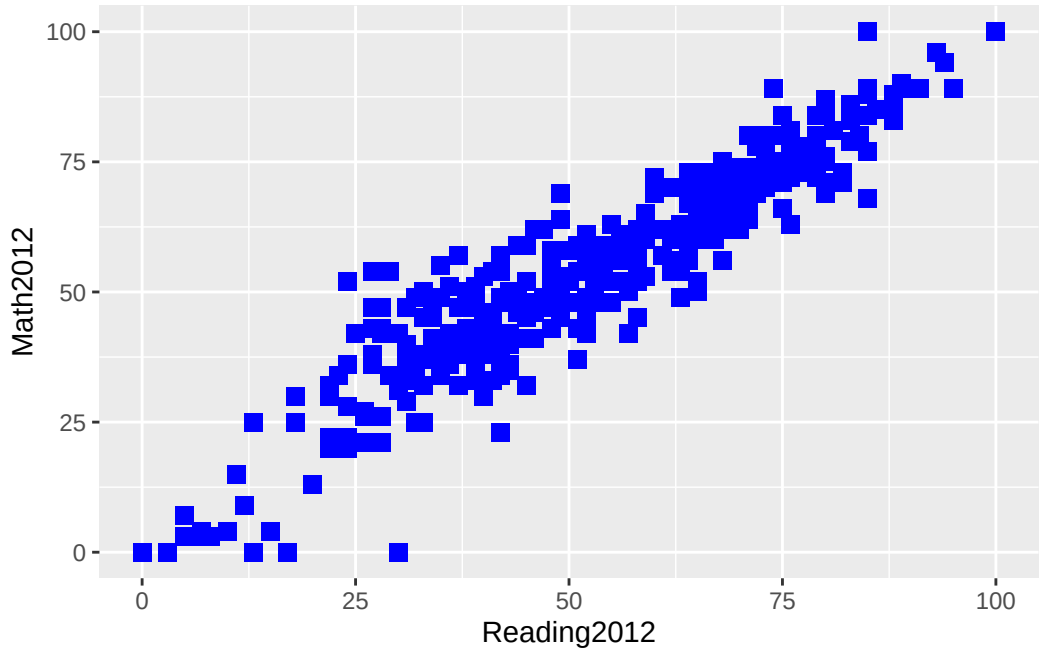

```
# Gráfico con dos variables numericas y un solo color en los puntos, forma =5
SchoolsMiamiDade |>
  ggplot(aes(Reading2012,Math2012))+
  geom_point(color= "red",
             size=3,
             shape=5)
```

```
Warning: Removed 91 rows containing missing values (`geom_point()`).
```


```
# Gráfico con dos variables numericas y un solo color en los puntos, forma =15
SchoolsMiamiDade |>
```

```
  ggplot(aes(Reading2012,Math2012))+
  geom_point(color="blue",
             size=3,
             shape=15)
```

```
Warning: Removed 91 rows containing missing values (`geom_point()`).
```

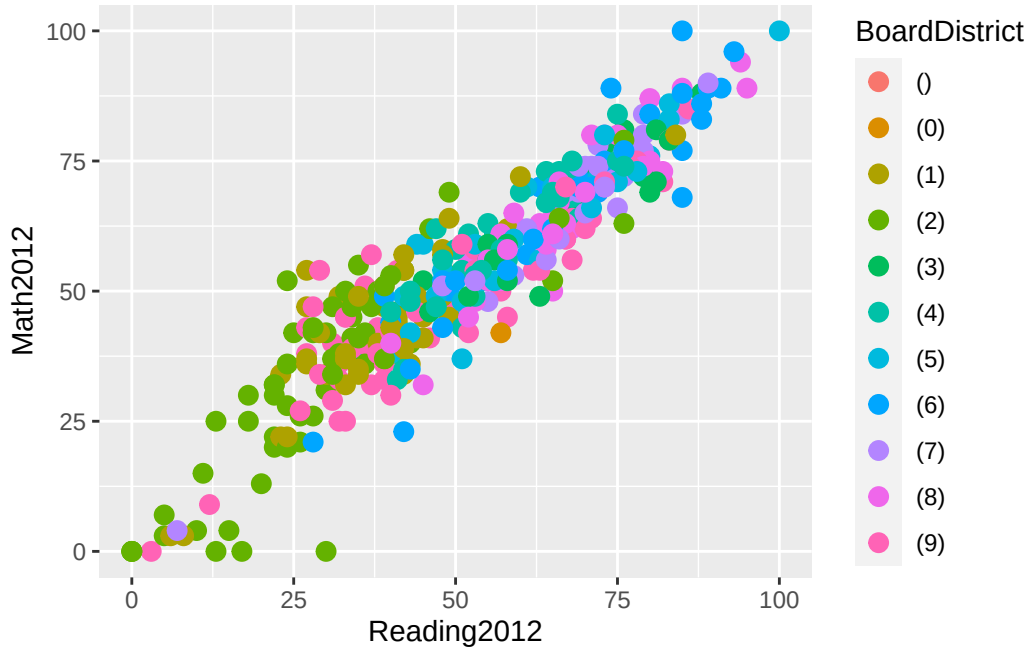


9.5 Tres variables en el gráfico

En los diagramas de dispersión en ggplot, es posible añadir una tercera variable, en este caso la variable categórica **BoardDistrict** en la propiedad **color** de la geometria. De esta forma es posible identificar las escuelas con mayor puntaje en ambas materia y a que distrito pertenecen,

```
# Gráfico con dos variables numéricas y colores según los distritos
SchoolsMiamiDade |>
  ggplot(aes(Reading2012,Math2012))+
  geom_point(aes(
             color=BoardDistrict),
             size=3)
```

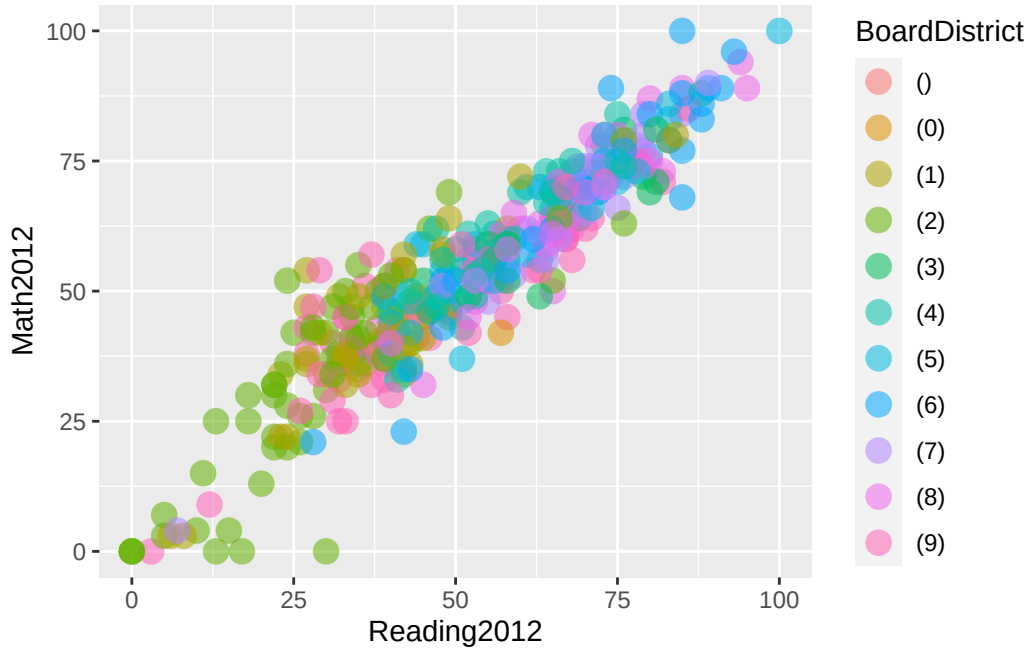
```
Warning: Removed 91 rows containing missing values (`geom_point()`).
```



Modificar el tamaño de los puntos y la transparencia (valores de 0 a 1) Uno de los inconvenientes comunes en los gráficos de dispersión con muchos puntos, es el solapamiento de estos, por lo que aveces no se puede apreciar de forma correcta el número de puntos, Para ello podemos utilizar la propiedad **alpha** para la transparencia. Se aumenta el tamaño de los puntos a **4** a manera de ejemplo para mostrar mejor el solapamiento.

```
# Tamaño de los puntos fijos utilizando size = 4
SchoolsMiamiDade |>
ggplot(aes(Reading2012,Math2012))+
  geom_point(aes(
             color=BoardDistrict),
              alpha=0.55,
              size=4)
```

```
Warning: Removed 91 rows containing missing values (`geom_point()`).
```

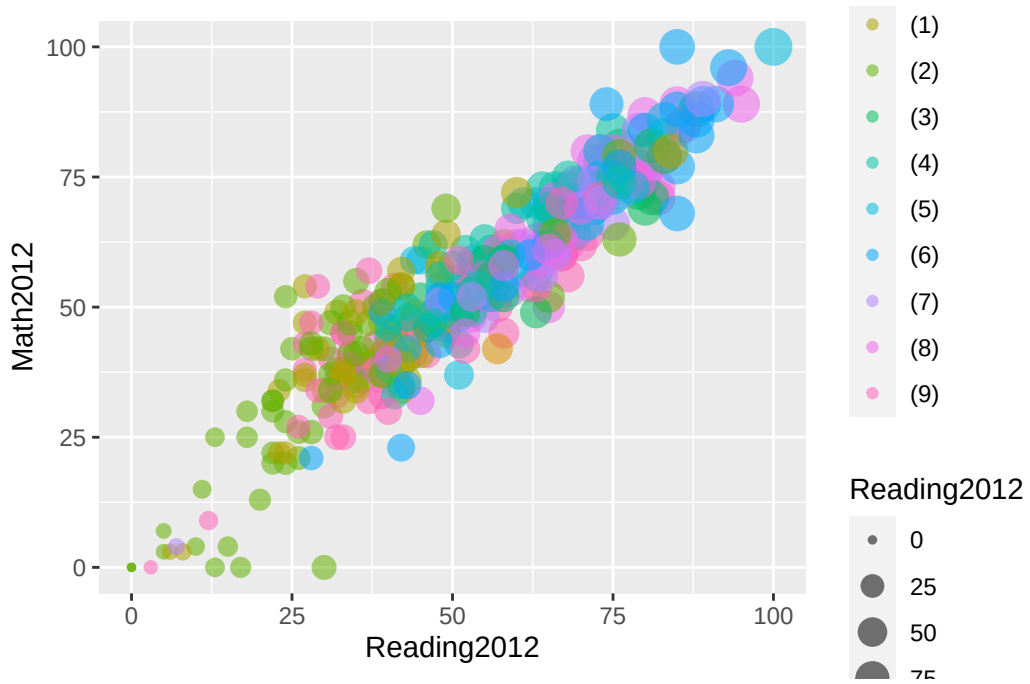


9.5 Cuatro variables

Podemos utilizar una cuarta variable en este tipo de gráfico modificando el tamañó de los puntos de forma dinámica, en este caso la variable a utilizar **Reading2012** debe ser de tipo numérica. Al realizar esto se generan dos leyendas en el gráfico, uno de **color** y otra se **size**.

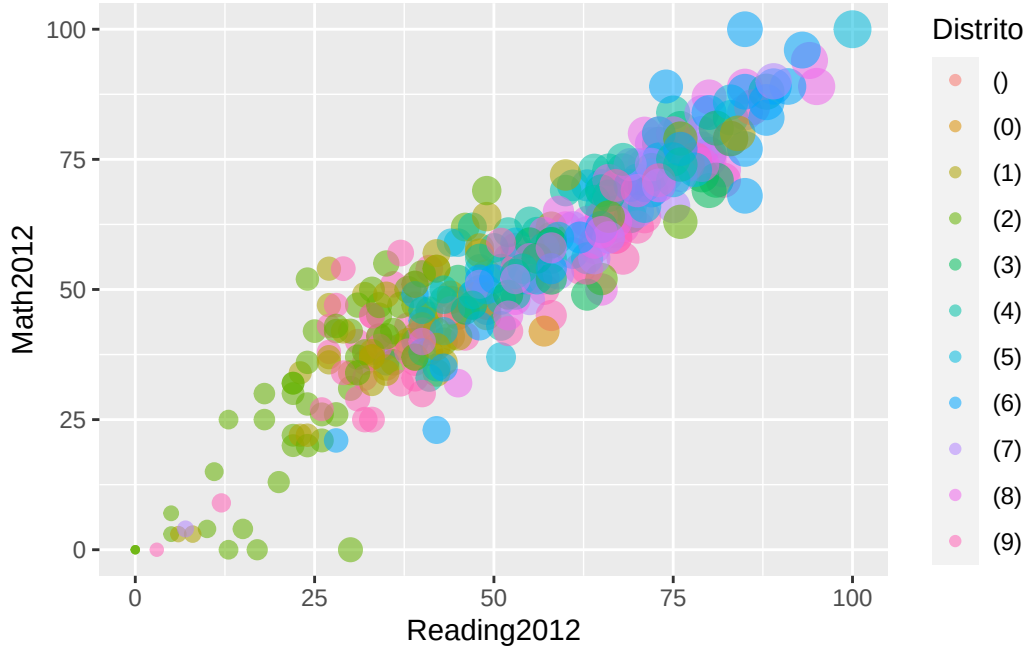
```
# Tamaño de los puntos dinámicos utilizando size = Reading2012
SchoolsMiamiDade |>
  ggplot(aes(Reading2012,Math2012))+
  geom_point(aes(
                color=BoardDistrict,
                size=Reading2012),
                alpha=0.55)
```

```
Warning: Removed 91 rows containing missing values (`geom_point()`).
```




```
# eliminar leyenda de size
SchoolsMiamiDade |>
  ggplot(aes(Reading2012,Math2012))+
  geom_point(aes(
                 color=BoardDistrict,
                 size=Reading2012),
                 alpha=0.55)+
  guides(size="none")+
  labs(color="Distrito")
```

```
Warning: Removed 91 rows containing missing values (`geom_point()`).
```



9.5 Cinco variables

A manera de ejemplo (muchas variables pu7eden sobrecargar el gráfico), es posible incluir una quinta variable (puede generar carga en la visualización) en el gráfico modificando la propiedad **shape** en el geometria, la variable que modifca este parámetro debe ser **categorica**, y la diversida de formas **shape** será en función de la cantidad de categorías de esa variable.

```
# Tamaño de los puntos dinámicos y la forma del punto según el Distrito
SchoolsMiamiDade |>
  ggplot(aes(Reading2012,Math2012))+
  geom_point(aes(
                 color=BoardDistrict,
                 size=Reading2012,
                 shape=SchoolGrade),
                 alpha=0.45)+
  guides(size="none")+
  labs(color="Distrito")
```

```
Warning: The shape palette can deal with a maximum of 6 discrete values because more
than 6 becomes difficult to discriminate
i you have requested 7 values. Consider specifying shapes manually if you need
  that many have them.
```

```
Warning: Removed 92 rows containing missing values (`geom_point()`).
```

9.5 Eliminar datos vacíos

Si bien evaluar si se eliminan o no los datos **nulos o vacios** es un tema previo a la visualización de datos, es posible filtrar esos datos para que no causen ruido al generar el gráfico, para ello utilizamos la función **filter(SchoolGrade!=" ")** previo a la función ggplot. Podemos verificar la visualización con los datos de las tablas de frecuencia.

```
#tabla de frecuencia
# Una forma de mostrar los datos categóricos
# y el numero de valores sin categoria
table(SchoolsMiamiDade$BoardDistrict)
```

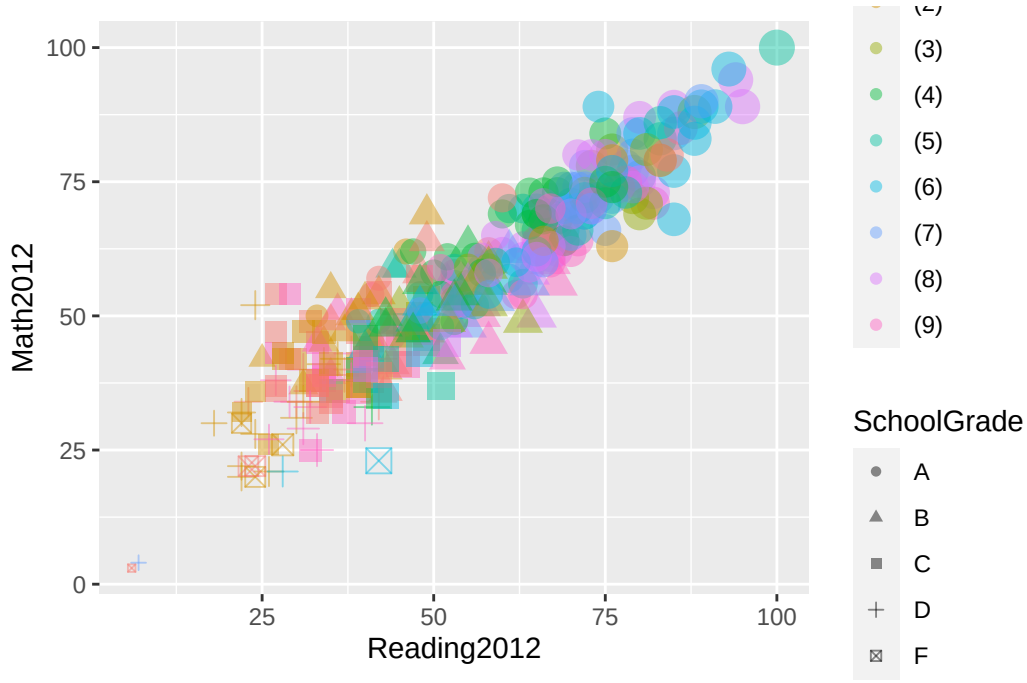
```
() (0) (1) (2) (3) (4) (5) (6) (7) (8) (9)
 2   1  53  75  28  47  41  53  37  47  76
```

```
table(SchoolsMiamiDade$SchoolGrade)
```

```
     A   B   C   D   F   I
55 204  85  77  31   7   1
```

```
# summary(SchoolsMiamiDade)


SchoolsMiamiDade |>
  filter(SchoolGrade!=" ") |>
  ggplot(aes(Reading2012,Math2012))+
  geom_point(aes(
                 color=BoardDistrict,
                 size=Reading2012,
                 shape=SchoolGrade),
                 alpha=0.45)+
  guides(size="none")+
  labs(color="Distrito")
```

```
Warning: Removed 64 rows containing missing values (`geom_point()`).
```



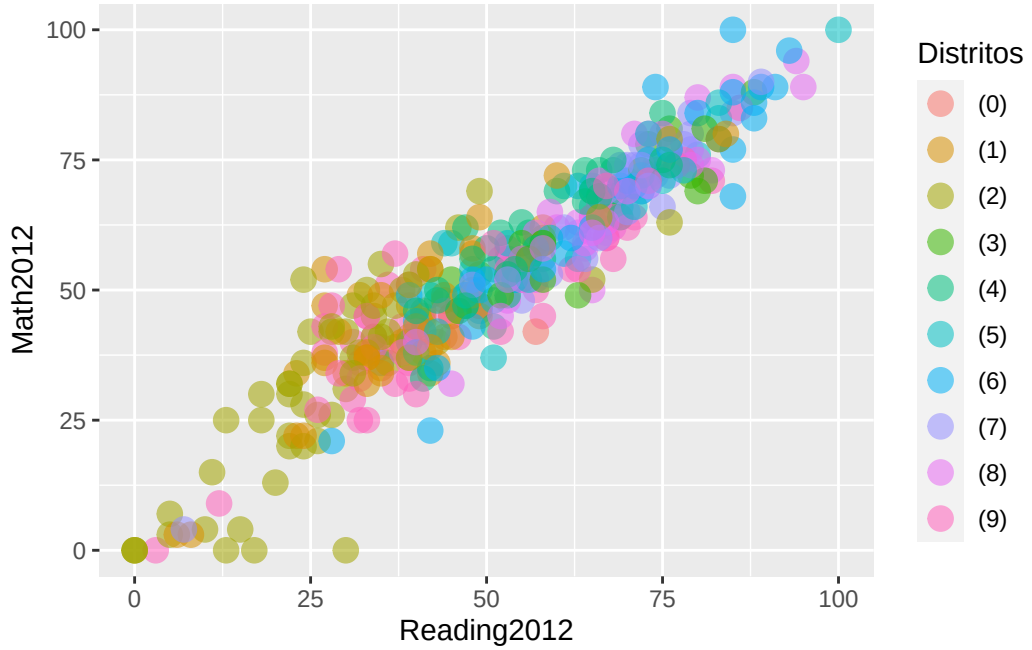
Eliminar Distritos vacios o con minimo de escuelas (opcional) Eliminar escuelas con valores () y (0) en los distritos que no aportan información al gráfico utilizando **filter(BoardDistrict!=“()”)**. Se verifican los datos con una tabla de frecuencia.

```
#tabla de frecuencia de distritos
table(SchoolsMiamiDade$BoardDistrict)
```

```
() (0) (1) (2) (3) (4) (5) (6) (7) (8) (9)
 2   1  53  75  28  47  41  53  37  47  76
```

```
#eliminar datos de distritos con valor ()
SchoolsMiamiDade |>
  filter(BoardDistrict!="()")  |>
  ggplot( aes(Reading2012,Math2012))+
  geom_point(aes(color=BoardDistrict),
             alpha=0.55,
             size=4)+
  guides(color = guide_legend("Distritos"))
```

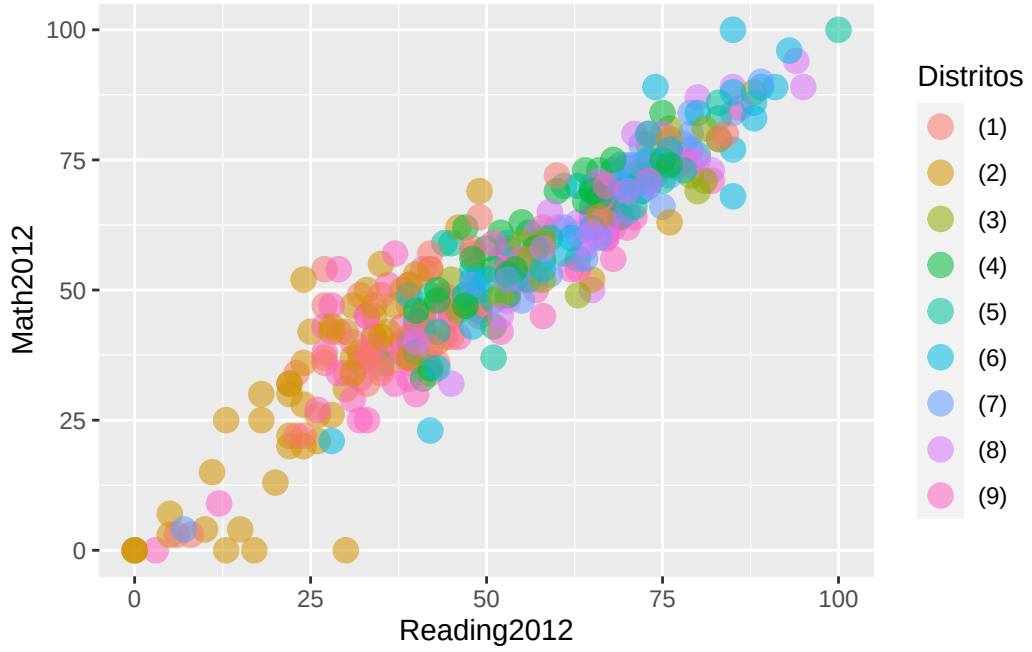
```
Warning: Removed 89 rows containing missing values (`geom_point()`).
```




```
#eliminar datos de distritos con valores vacios (), (0)
SchoolsMiamiDade |>
  filter(BoardDistrict!=c("()","(0)"))  |>
  ggplot( aes(Reading2012,Math2012))+
  geom_point(aes(
             color=BoardDistrict),
             alpha=0.55,
             size=4)+
```

```
  guides(color = guide_legend("Distritos"))
```

```
Warning: Removed 89 rows containing missing values (`geom_point()`).
```



9.6 Énfasis en los gráficos

9.6 Enfasis colores en puntos

En un gráfico de dispersión también podemos hacer énfasis en una categoría en particular,asignando colores de forma manual. En este ejemplo contabilizamso el número de categorías de la variable **BoardDistrict** que son **11** eliminando las categorias **"()","(0)"**, por lo que serían 9 categorías a utilizar en el gráfico, por lo tanto deberíamos crear nueve colores, uno por cada categoría, almacenados en el vector **colores** y modificando solo el color de la posición del distrito de interés.

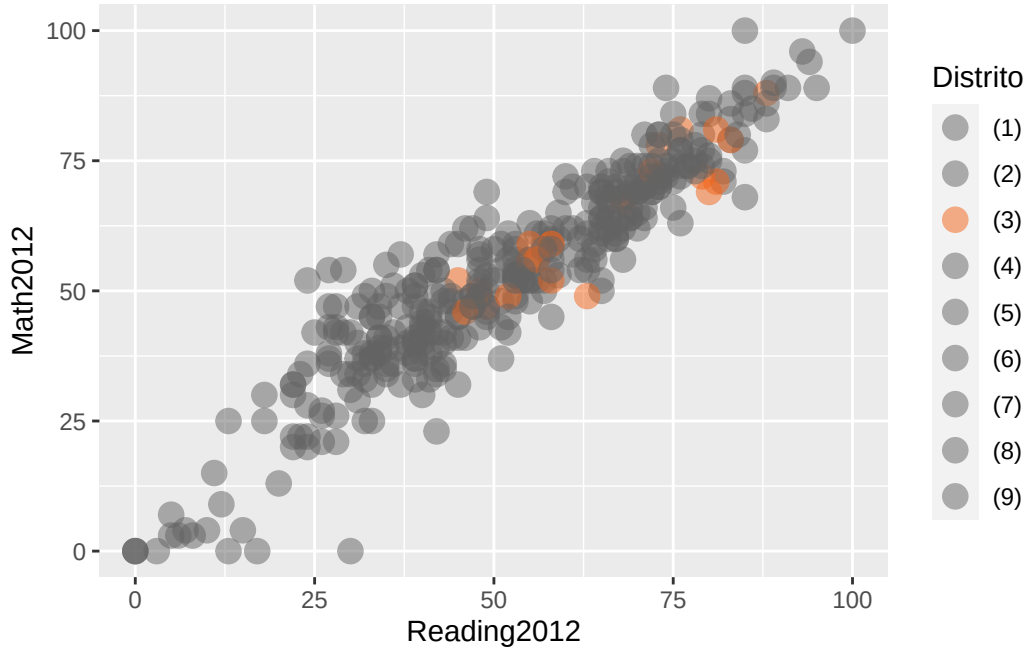
```
# Calcular Cantidad de categorias
table(SchoolsMiamiDade$BoardDistrict) |>
  length()
```

```
[1] 11
```

```
#El número de colores esta asociado al número de categorias
# En el caso de Región son 9 categorias
colores <- c("#636363","#636363", "#636363", "#636363" ,"#636363" ,
             "#636363" ,"#636363","#636363", "#636363")

# Modificar color del distrito en la posición 3
colores[3] <- "#F26419"
```

```
SchoolsMiamiDade |>
  filter(BoardDistrict!=c("()","(0)"))  |>
  ggplot(aes(Reading2012,Math2012))+
  geom_point(aes(
             color=BoardDistrict),
             alpha=0.5,
             size=4)+
  guides(size="none")+
  labs(color="Distrito")+
  scale_color_manual(values = colores)
```

```
Warning: Removed 89 rows containing missing values (`geom_point()`).
```



9.6 Enfasis transparencia en puntos

Otra forma de hacer énfasis en el gráfico es a través de la transparencia **alpha**, para ello debemos crear un vector **transparencia**con los valores del valor **alpha** de cada valor de la variable **BoardDistrict** y utilizando **scale_alpha_manual()**. Es importante resaltar que la propiedad **alfa** debe estar en la estética **aes()** ya que sus datos pertencerán a la variable **BoardDistrict**.

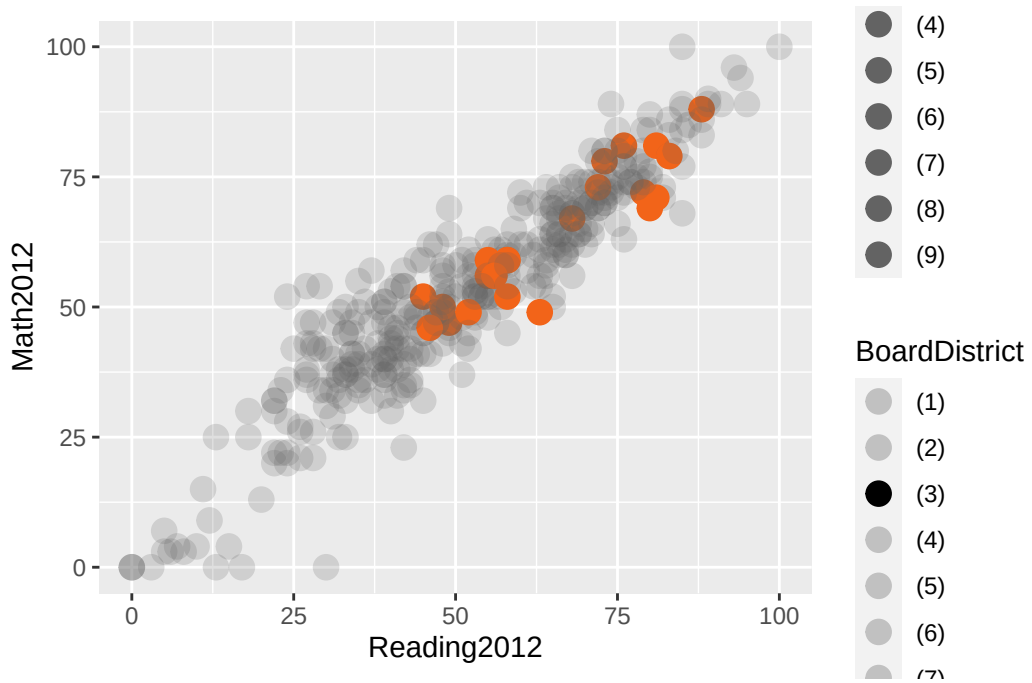
```
# Calcular Cantidad de categorias
table(SchoolsMiamiDade$BoardDistrict) |>
  length()
```

```
[1] 11
```

```
# El número de colores esta asociado al número de categorias
# En el caso de Región son 9 categorias
colores <- c("#636363","#636363", "#636363", "#636363" ,"#636363" ,
             "#636363" ,"#636363","#636363", "#636363")
```

```
colores[3] <- "#F26419"

# valores de trasnparencia
transparencia <- sample(0.2,9, replace = TRUE)
transparencia[3] <- 1.0

SchoolsMiamiDade |>
  filter(BoardDistrict!=c("()","(0)"))  |>
  ggplot(aes(Reading2012,Math2012))+
  geom_point(aes(
             color=BoardDistrict,
             alpha=BoardDistrict),
             size=4)+
  guides(size="none")+
  labs(color="Distrito")+
  scale_color_manual(values = colores)+
  scale_alpha_manual(values = transparencia)
```

```
Warning: Removed 89 rows containing missing values (`geom_point()`).
```



9.6 Etiquetas en los puntos

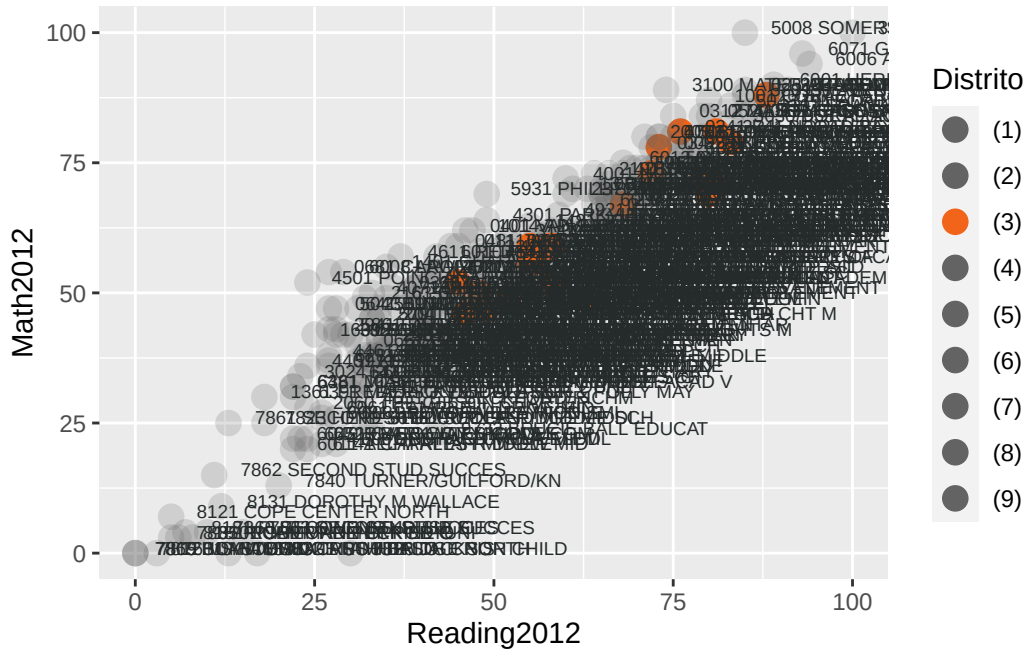
Para mostarr las etiquetas o texto en los puntos del gráfico debemos utilizar la geoametria **geom_text()** y el valor de la etiqueta a utilizar **label=variable**.

```
# El número de colores esta asociado al número de categorias
# En el caso de Región son 9 categorias
colores <- c("#636363","#636363", "#636363", "#636363" ,"#636363" ,
             "#636363" ,"#636363","#636363", "#636363")
colores[3] <- "#F26419"
```

```
# valores de trasnparencia
transparencia <- sample(0.2,9, replace = TRUE)
transparencia[3] <- 1.0

SchoolsMiamiDade |>
  filter(BoardDistrict!=c("()","(0)"))  |>
  ggplot(aes(Reading2012,Math2012))+
  geom_point(aes(
             color=BoardDistrict,
             alpha=BoardDistrict),
             size=4)+
  guides(size="none",
         alpha ="none")+
  labs(color="Distrito")+
  scale_color_manual(values = colores)+
  scale_alpha_manual(values = transparencia) +
  geom_text(aes(
          label=SchoolName),
          size=2.5,
          hjust=-0.1,
          vjust=0.1,
          color="#252b2b"
           )
```

```
Warning: Removed 89 rows containing missing values (`geom_point()`).
```

```
Warning: Removed 89 rows containing missing values (`geom_text()`).
```



9.6 Énfasis en el texto

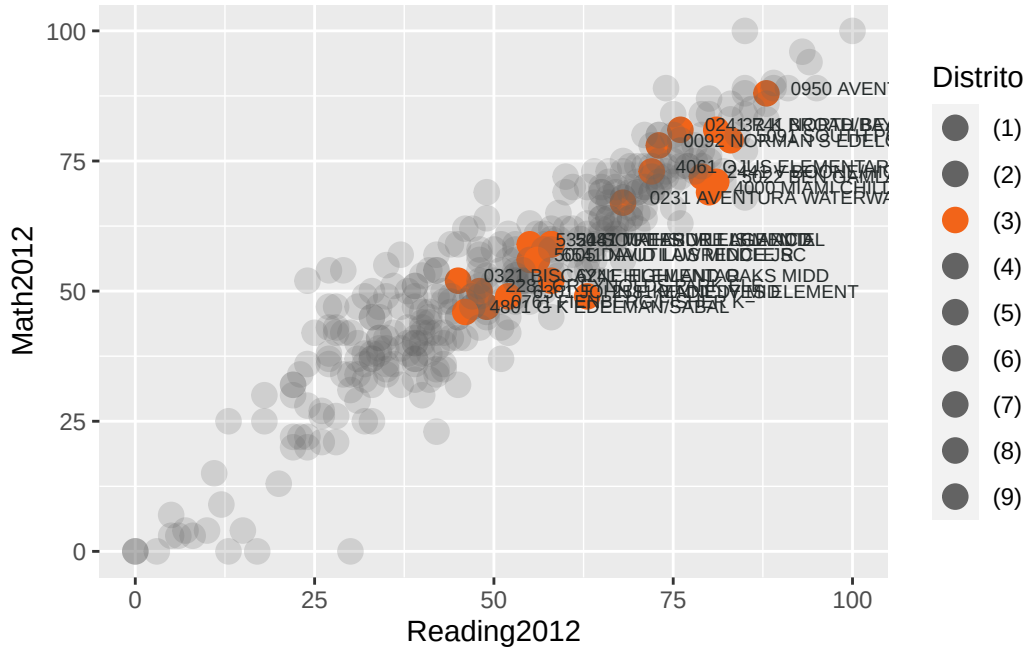
POdemos también hacer énfasis en las etiquetas de texto, colocando solo las etiquetas de los punto que queremos, para ello es necesario crear una variable que contenga esos datos. **etiquetas <-**

(ifelse(SchoolsMiamiDade$BoardDistrict == "(3)", SchoolsMiamiDade$SchoolName," ")) SchoolsMiamiDade<- etiquetas Si el distrito es"(3)" asignar a la etiqueta el nombre de la escuela de la variable** SchoolName*** **, sino le asigna espacio en blanco a la etiqueta. Almacenamos este vector al tibble** SchoolsMiamiDade**.

```
# El número de colores esta asociado al número de categorias
# En el caso de Región son 9 categorias
colores <- c("#636363","#636363", "#636363", "#636363" ,"#636363" ,
             "#636363" ,"#636363","#636363", "#636363")
colores[3] <- "#F26419"

# valores de trasnparencia
transparencia <- sample(0.2,9, replace = TRUE)
transparencia[3] <- 1.0

#valores de etiqueta
# SI el distrito es "(3)" asignar a la etiqueta el nombre de la escuela
# sino le asigna espacio en blanco a la etiqueta
etiquetas <- (ifelse(SchoolsMiamiDade$BoardDistrict=="(3)",
                     SchoolsMiamiDade$SchoolName," "))
SchoolsMiamiDade$label_ <- etiquetas

SchoolsMiamiDade |>
  filter(BoardDistrict!=c("()","(0)"))  |>
  ggplot(aes(Reading2012,Math2012))+
  geom_point(aes(
             color=BoardDistrict,
             alpha=BoardDistrict),
             size=4)+
  guides(size="none",
         alpha ="none")+
  labs(color="Distrito")+
  scale_color_manual(values = colores)+
  scale_alpha_manual(values = transparencia) +
  geom_text(aes(
          label=label_),
          size=2.5,
          hjust=-0.1,
          vjust=0.1,
          color="#252b2b"
           )
```

```
Warning: Removed 89 rows containing missing values (`geom_point()`).
```

```
Warning: Removed 89 rows containing missing values (`geom_text()`).
```



Recortar texto de la etiqueta que contine los nombres de las escuelas Para recortar el nombre de las etiquetas, podemos utilizar el siguiente código **str_sub(etiquetas,1,4)**, este selecciona de cada etiqueta la posición 1 al 4 de cada texto y luego se lo asigna a la variable **label** del dataframe.

```
# El número de colores esta asociado al número de categorias
# En el caso de Región son 9 categorias
colores <- c("#636363","#636363", "#636363", "#636363" ,"#636363" ,
             "#636363" ,"#636363","#636363", "#636363")
colores[3] <- "#F26419"

# valores de trasnparencia
transparencia <- sample(0.2,9, replace = TRUE)
transparencia[3] <- 1.0

#valores de etiqueta
# SI el distrito es "(3)" asignar a la etiqueta el nombre de la escuela
# sino le asigna espacio en blanco a la etiqueta
etiquetas <- (ifelse(SchoolsMiamiDade$BoardDistrict=="(3)",
                     SchoolsMiamiDade$SchoolName," "))

# Selecciona solo los primeros cuatros digitos de la etiqueta
# que contiene los nombres
SchoolsMiamiDade$label_ <- str_sub(etiquetas,1,4)

SchoolsMiamiDade |>
  filter(BoardDistrict!=c("()","(0)"))  |>
  ggplot(aes(Reading2012,Math2012))+
  geom_point(aes(
             color=BoardDistrict,
             alpha=BoardDistrict),
             size=4)+
  guides(size="none",
         alpha ="none")+
```

```
  labs(color="Distrito")+
  scale_color_manual(values = colores)+
  scale_alpha_manual(values = transparencia) +
  geom_text(aes(
          label=label_),
          size=2.5,
          hjust=-0.15,
          vjust=0.1,
          color="#252b2b"
           )
```

```
Warning: Removed 89 rows containing missing values (`geom_point()`).
```

```
Warning: Removed 89 rows containing missing values (`geom_text()`).
```

9.7 Facetas

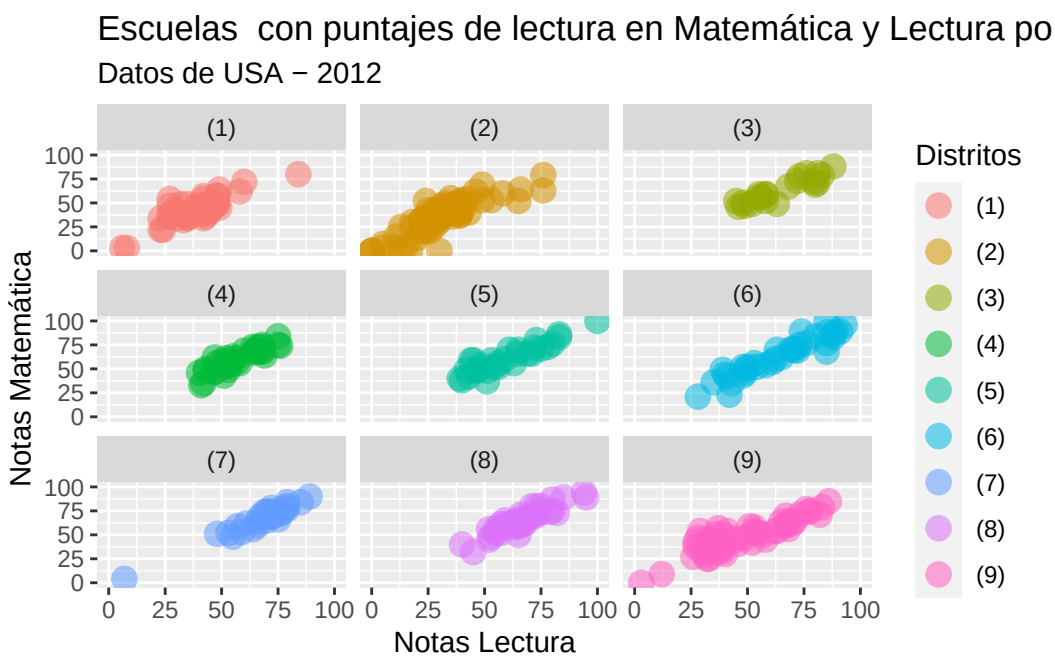
9.7 Facetas para clasificar gráficos

Podemos utilizar facetas para separar los gráficos de dispersión por una variable categórica, através de las propiedades **facet_wrap()** o **facet_grid()** , aunque funcionan para lo mismo la forma de dsitribución de los gráficos utilizando estas funciones es diferente.

```
SchoolsMiamiDade |>
  filter(BoardDistrict!=c("()","(0)"))  |>
  ggplot(aes(Reading2012,Math2012))+
  geom_point(aes(
                color=BoardDistrict),
                alpha=0.55,
                size=4) +
```

```
  labs(title="Escuelas  con puntajes de lectura en Matemática y Lectura por Distrito",
       subtitle="Datos de USA - 2012") +
  ylab("Notas Matemática") +
  xlab("Notas Lectura")+
  guides(color = guide_legend("Distritos")) +
  facet_wrap(~BoardDistrict)
```

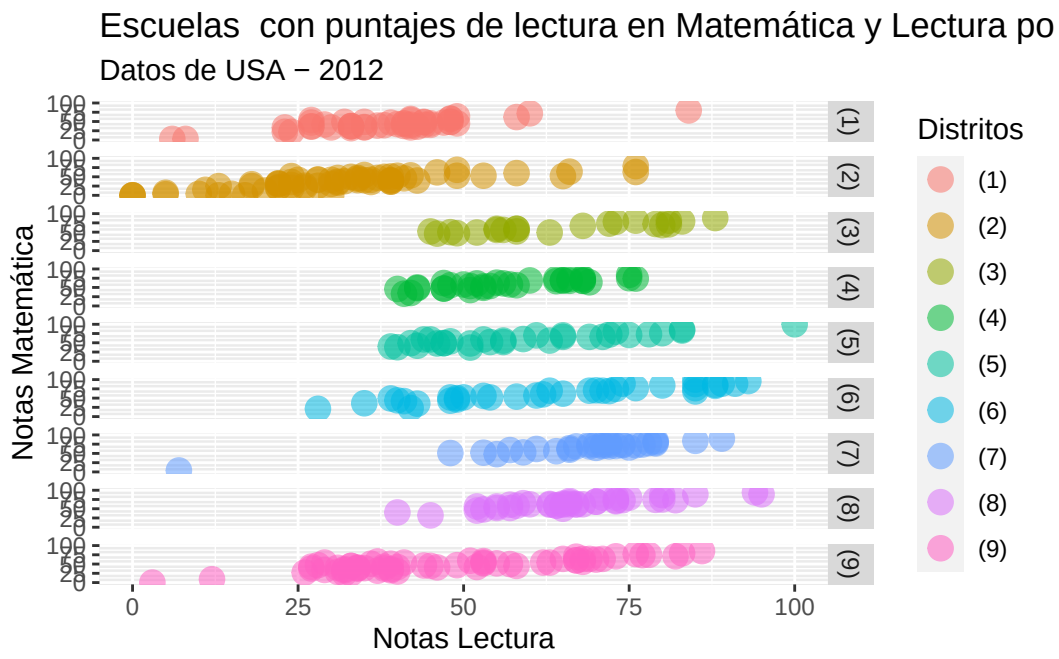
```
Warning: Removed 89 rows containing missing values (`geom_point()`).
```



Uso de la función facet_grid()

```
SchoolsMiamiDade |>
  filter(BoardDistrict!=c("()","(0)"))  |>
  ggplot(aes(Reading2012,Math2012))+
  geom_point(aes(
                 color=BoardDistrict),
                 alpha=0.55,
                 size=4) +
  labs(title="Escuelas  con puntajes de lectura en Matemática y Lectura por Distrito",
       subtitle="Datos de USA - 2012") +
  ylab("Notas Matemática") +
  xlab("Notas Lectura")+
  guides(color = guide_legend("Distritos")) +
  facet_grid(BoardDistrict~.)
```

```
Warning: Removed 89 rows containing missing values (`geom_point()`).
```

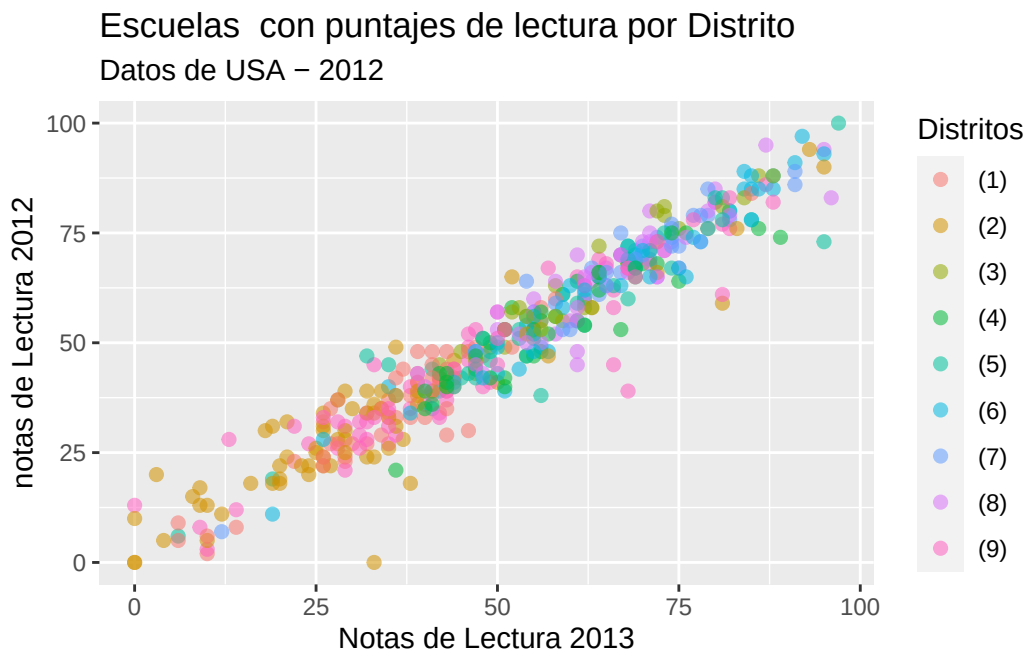


9.7 Evaluación de las variables de lectura en los años 2012 y 2013

Ambos datos cuantitativos **Reading2013,Reading2012**.

```
SchoolsMiamiDade |>
  filter(BoardDistrict!=c("()","(0)"))  |>
  ggplot(aes(Reading2013,Reading2012))+
  geom_point(size=2, aes(color=BoardDistrict), alpha=0.55) +
  labs(title="Escuelas  con puntajes de lectura por Distrito",
       subtitle="Datos de USA - 2012") +
   ylab("notas de Lectura 2012") + xlab("Notas de Lectura 2013")+
  guides(color = guide_legend("Distritos"))
```

```
Warning: Removed 15 rows containing missing values (`geom_point()`).
```



9.8 Recta de regresión lineal

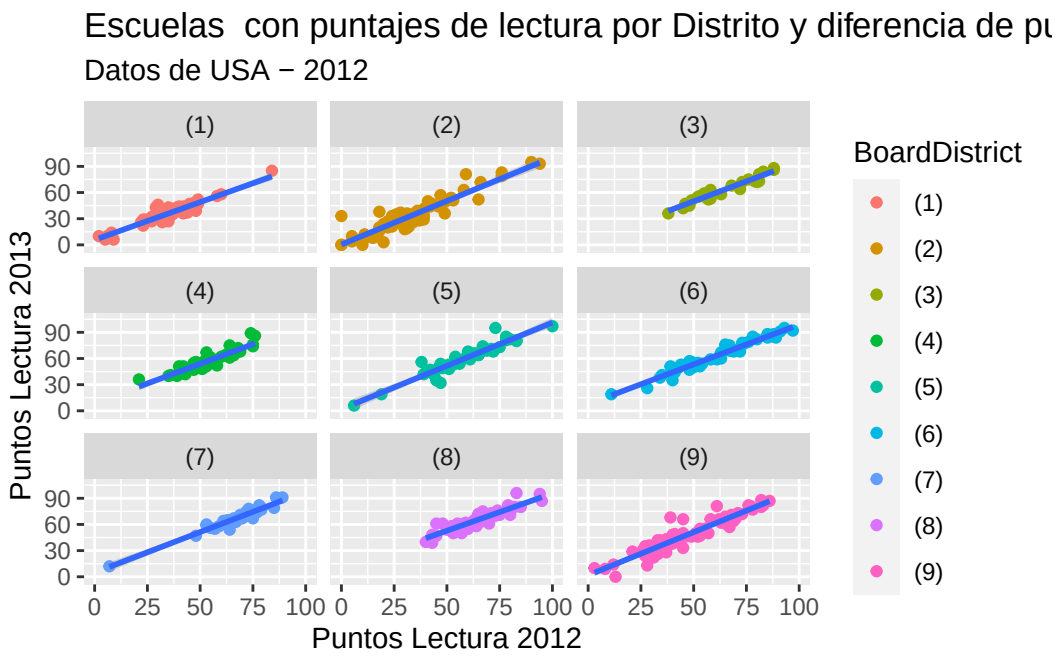
Se puede crear una recta de regresión lineal a través de la geometría **geom_smooth()** y sus métodos **geom_smooth(method='lm')** : - lm *modelo lineal* - glm **- gam** - loess *ajuste de regresión local suave*

```
#metodol lm
SchoolsMiamiDade |>
  filter(BoardDistrict!=c("()","(0)"))  |>
  ggplot(aes(Reading2012,Reading2013))+
  geom_point(aes(color=BoardDistrict)) +
  labs(title="Escuelas  con puntajes de lectura por Distrito y diferencia de puntos por año'
  ylab("Puntos Lectura 2013") + xlab("Puntos Lectura 2012") +
  geom_smooth(method='lm', se=TRUE) +
  facet_wrap(BoardDistrict~.)
```

```
`geom_smooth()` using formula = 'y ~ x'
```

```
Warning: Removed 15 rows containing non-finite values (`stat_smooth()`).
```

```
Warning: Removed 15 rows containing missing values (`geom_point()`).
```



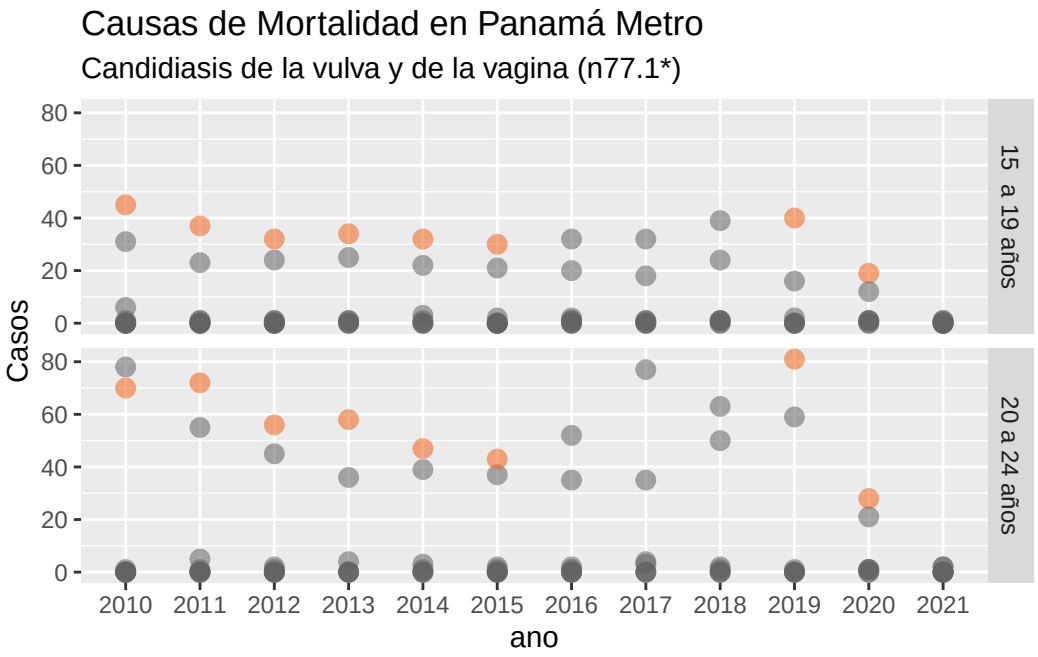
9.9 Práctica

Crear gráfico de puntos utilizando los datos **INEC-morbilidad-2023-10-14.csv**. Es necesario filtrar los datos seleccionando solo: **Region="PANAMA METRO" Genero="M" Edad="15 a 19 años"** o igual a **Edad="20 a 24 años"** Se debe usar como en los colores la variable **Causas** y hacer énfasis en los colores en la **Causa=Candidiasis de la vulva y de la vagina (n77.1*)**. El valor de alpha=035 (opcional) y la faceta se debe hacer utilizando la variable** edad**.

Gráfico de resultado:

```
inec <- read.csv("data/INEC-morbilidad-2023-10-14.csv")
inec <- as_tibble(inec)

# table(inec$Region)
# #
# inec |>
#   glimpse()

inec_4 <- inec  |>
          filter(Region=="PANAMA METRO" &
                   Genero=="M" &
                   (Edad=="15  a 19 años" |  Edad=="20 a 24 años"))

inec_4$ano <- as.factor(inec_4$ano)

# El número de colores esta asociado al número de categorias
# En el caso de Región son 9 categorias
colores <- c("#636363","#636363", "#636363", "#636363" ,"#636363" ,
             "#636363" ,"#636363","#636363", "#636363" , "#636363",
             "#636363" ,"#636363","#636363", "#636363" , "#636363")
colores[4] <- "#F26419" #Candidiasis de la vulva y de la vagina (n77.1*)
# colores[4] <- "#F26419"
```

```
# table(inec_4$Edad)
# table(inec_4$Causas)

#grafico
inec_4 |>
ggplot(aes(ano, Casos))+
         geom_point(aes(
             color=Causas,
             alpha=0.5
         ),
         size=3.0)+
  guides(color="none",
         alpha="none")+
  labs(title="Causas de Mortalidad en Panamá Metro",
       subtitle = "Candidiasis de la vulva y de la vagina (n77.1*)")+
  scale_color_manual(values = colores)+
  theme_gray()+
  facet_grid(Edad~.)
```




```
    # theme(legend.position="bottom") +
```

10Gráficos de Línea

Los gráficos de líneas se utilizan para mostrar valores cuantitativos durante un intervalo o período de tiempo continuo. Este tipo de gráfico se utiliza con mayor frecuencia para mostrar tendencias y analizar cómo han cambiado los datos a lo largo del tiempo.

Los gráficos de líneas se dibujan trazando primero los puntos de datos en una cuadrícula de coordenadas cartesianas y luego conectando a línea entre todos estos puntos.

10.0 Cargar librerias

Librerias necesarias que se deben haber instalado previamente con **install.packages(nombre del paquete)** o en el panel **Package**.

```
library(tidyverse) # incluye ggplot2
library(readxl) # funciones para importar xlsx
library(janitor) # funciones de limpieza
library(patchwork) #combinar gráficos de ggplot
library(ggExtra)
library(ggthemes) # nuevas temas para los
library(tibble)
library(skimr) # reseumen numerico
library(ggrepel) # añadir etiquetas a los gráficos
library(RColorBrewer) #paletas de colores
library(ggthemes)
```

10.0 Carga de datos

```
# leer datos de SchoolsMiamiDade
data_cases_tiroides<- read.csv("data/data_cases_tiroides.csv",
                  header = TRUE,
                  sep = ";")

# estructura de  los datos
data_cases_tiroides |>
    glimpse()
```

```
Rows: 76
Columns: 3
$ year  <int> 1975, 1976, 1977, 1978, 1979, 1980, 1981, 1982, 1983, 1984, 1985~
$ cases <chr> "NewCases", "NewCases", "NewCases", "NewCases", "NewCases", "New~
$ rate  <dbl> 4.8, 4.8, 5.4, 5.1, 4.5, 4.3, 4.4, 4.6, 4.7, 4.8, 5.1, 5.3, 5.0,~
```

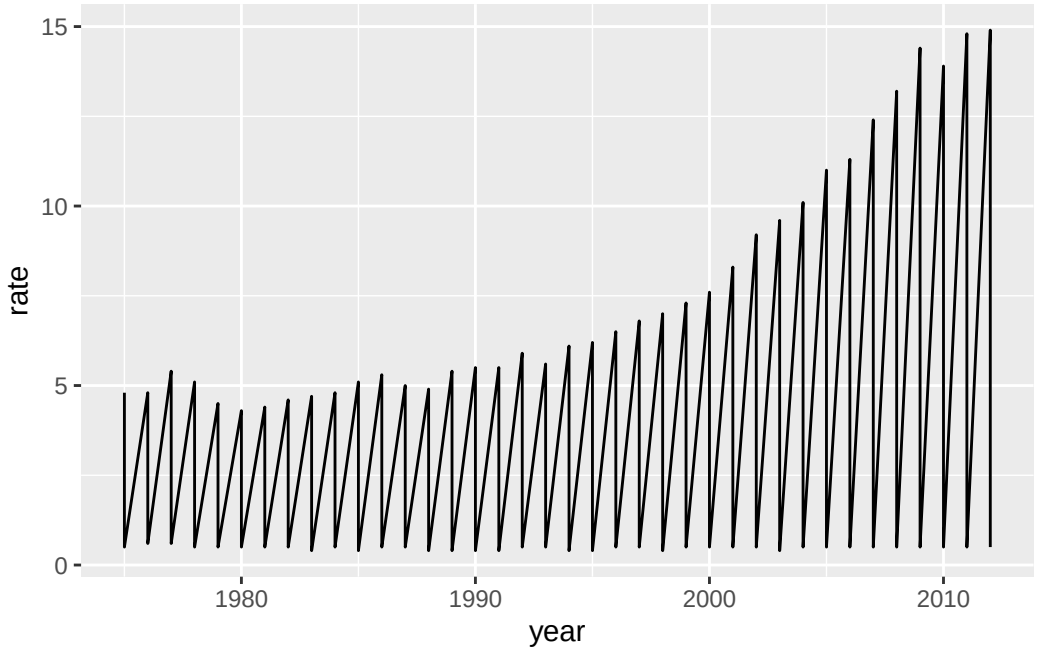
10.0 Significado de las variables del dataframe data_cases_tiroides

Year : año de cases : casos de estudio Rate : número de frecuencia

10.1 Crear gráfico de línea en R

El Gráfico de línea se puede realizar utilizando dos variables, relacionada con el intervalo o tiempo y la variable cuantitativa. Para crearlo en ggplot2 se utiliza la geometría **geom_line()**.

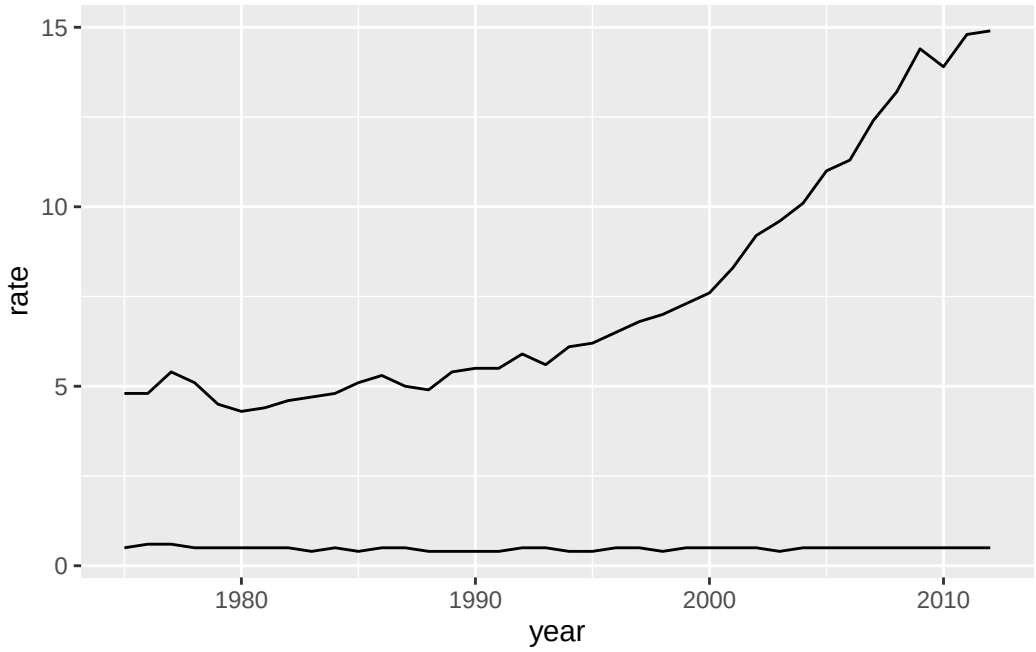
```
data_cases_tiroides |>
 ggplot(aes(year,rate)) +
   geom_line()
```



10.1 Agrupar datos por variable categórica

Al crear el gráfico en ggplot2 se muestra un gráfico cuyos valores generan una forma que muestra una especie de variación en los datos por cada año, estos se debe a que se están graficando todos los casos de el dataframe, sin embargo, los datos contienen una variable categórica llamada **cases**, la cual podemos colocar en la estética utilizandola como grupo **group=cases**, esto permitirá gráficar los datos separados por grupo.

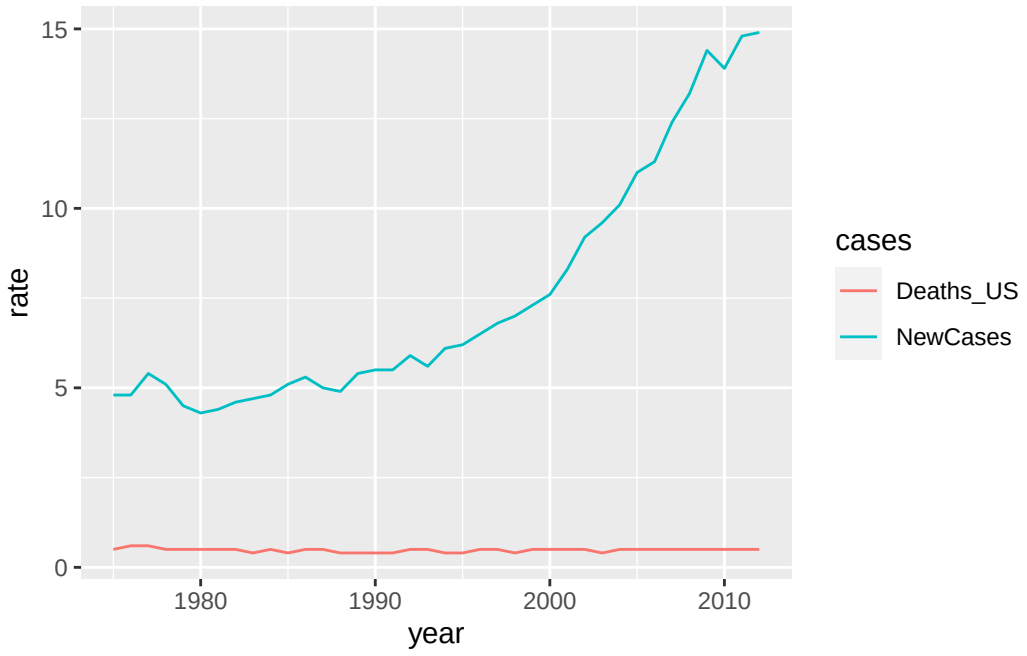
```
data_cases_tiroides |>
  ggplot(aes(year,
             rate,
             group=cases)) +
   geom_line()
```



10.1 Modificar color de línea del gráfico

Para modificar el color de línea de cada gráfico utilizaremos la propiedad color y la misma variable categórica que usamos para separar los grupos, **color=case**, el cual no solo mostrará diferentes colores de línea sino una leyenda de los grupos.

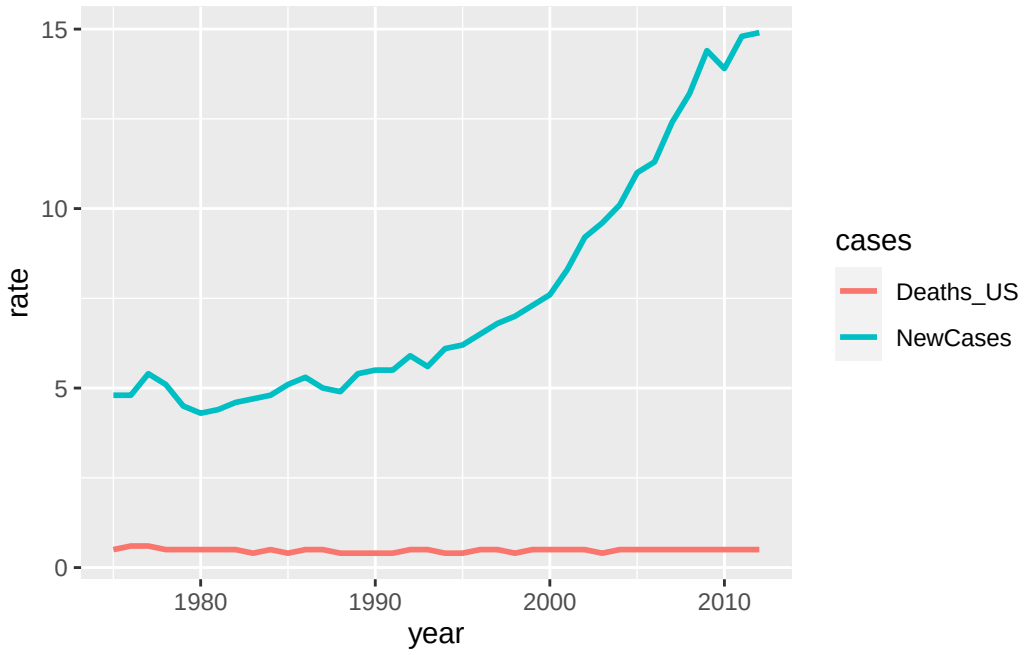
```
data_cases_tiroides |>
 ggplot(aes(year,
            rate,
            group=cases,
            color=cases)) +
   geom_line()
```



10.1 Modificar el grosor de línea

Podemos modificar el grosor de las líneas del gráfico colocando en la propiedad **linewidth=1**.

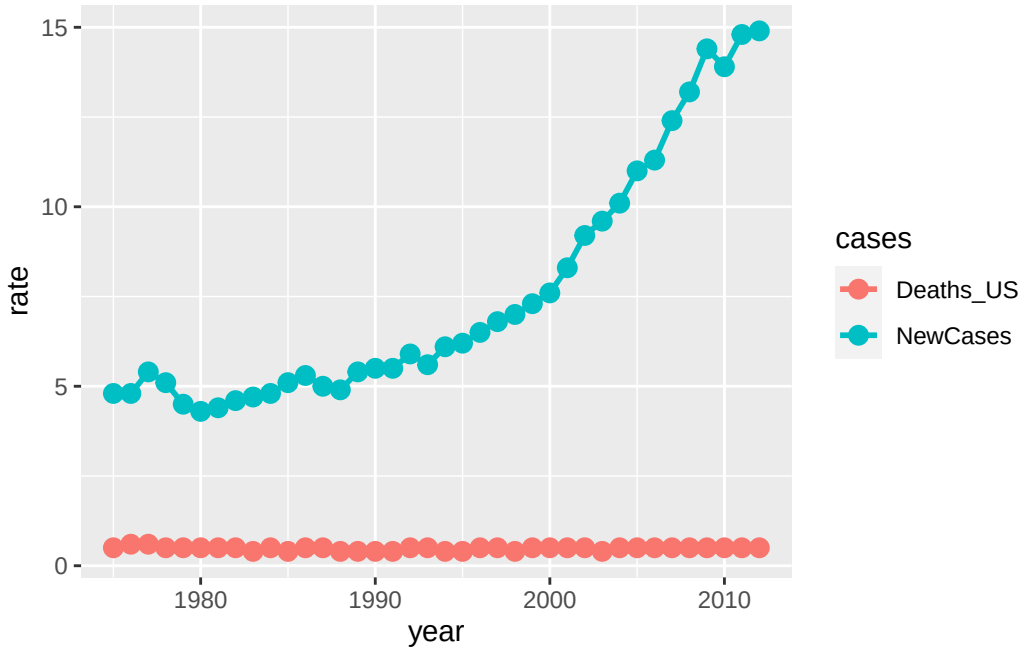
```
data_cases_tiroides |>
  ggplot(aes(year,
          rate,
          group=cases,
          color=cases)) +
    geom_line(linewidth=1)
```



10.1 Añadir geometría de puntos

Podemos añadir otra geometría al gráfico, en particular, la geometría de puntos utilizando **geom_point()** antes o después de **geom_line()**. A la geometría de punto le colocamos solo el tamaño con valor a **size=3**.

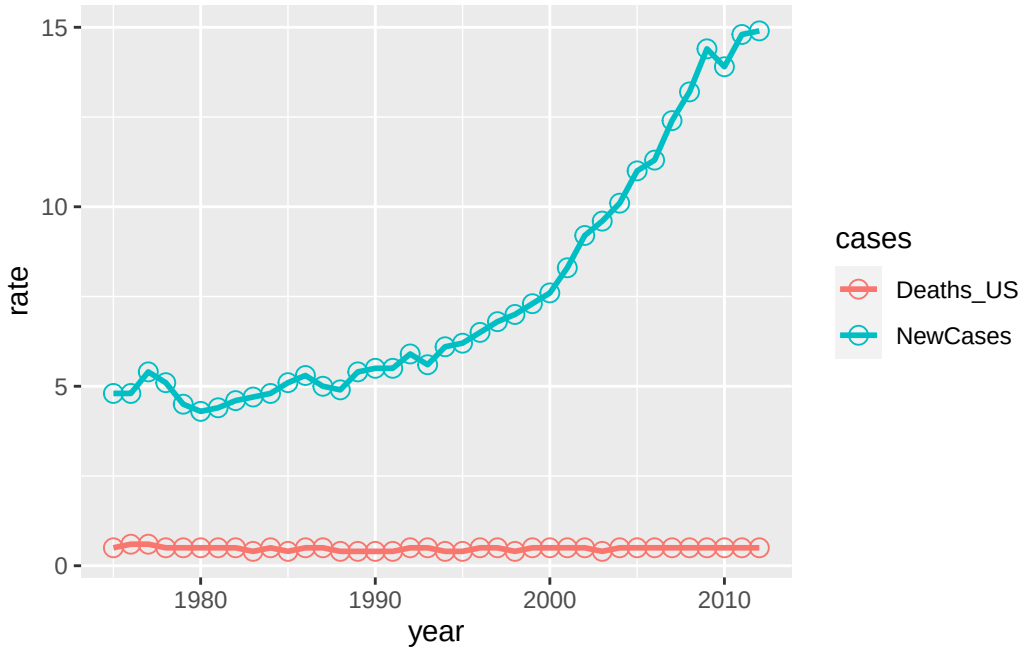
```
data_cases_tiroides |>
  ggplot(aes(year,
             rate,
             group=cases,
             color=cases)) +
    geom_line(linewidth=1)+
    geom_point(size=3)
```



10.1 Añadir geometría de puntos, shape

Podemos modificar la forma de los puntos utilizando valores entre 1 al 25, en este caso la forma 21 es un círculo sin relleno.

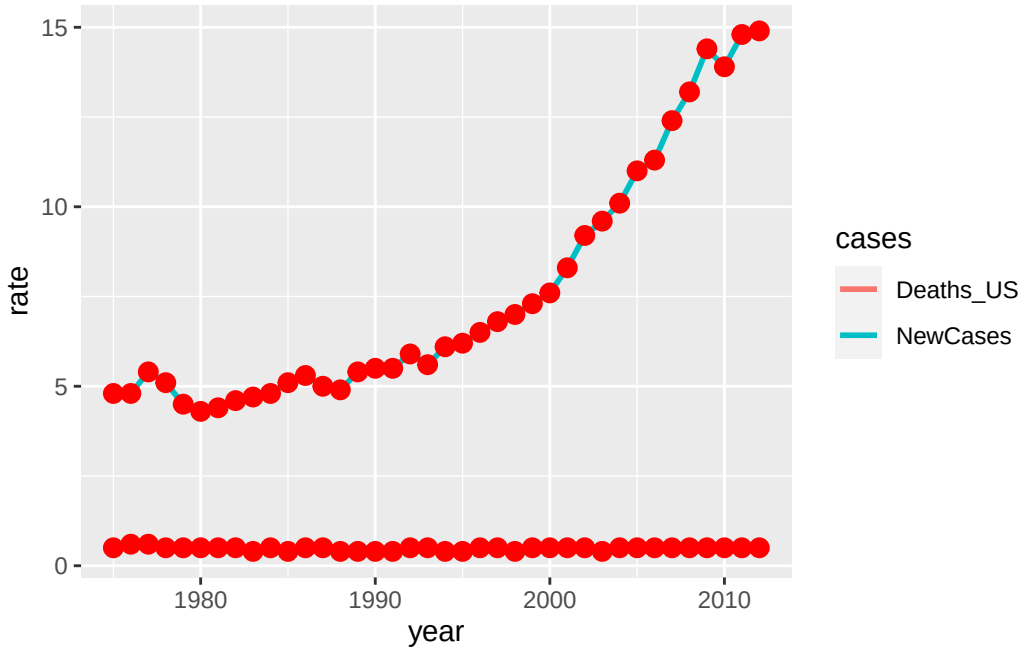
```
# Grafico de lineas y puntos con circulo sin relleno
data_cases_tiroides |>
  ggplot(aes(year,
             rate,
             group=cases,
             color=cases)) +
    geom_line(linewidth=1)+
    geom_point(size=3,
               shape=21)
```



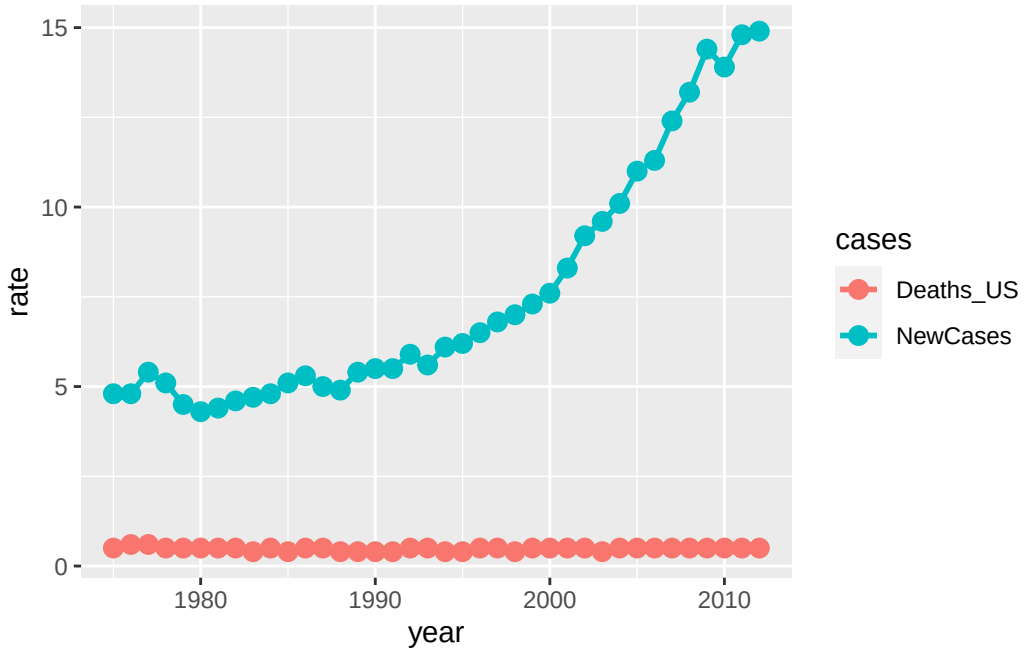
10.1 Añadir geometría de puntos, shape ,color

Si queremos rellenar los puntos de la forma **shape**, podemos hacerlo utilizando colores personalizados (pueden ser diferente tanto en el relleno **fill** como en el color de la línea de la forma **color**). Si queremos utilizar el mismo color de las líneas en el punto entonces debemos utilizar en **fill y color** la variable categórica que separa los grupos de líneas **cases**.

```
# Grafico de lineas y puntos con circulo con relleno
# Color personalizado
data_cases_tiroides |>
  ggplot(aes(year,
             rate,
             group=cases,
             color=cases)) +
    geom_line(linewidth=1)+
    geom_point(size=3,
               shape=21,
               color="red",
               fill="red")
```




```
# Grafico de lineas y puntos con circulo con relleno
# Color según variables categóricas
data_cases_tiroides |>
  ggplot(aes(year,
             rate,
             group=cases,
             color=cases)) +
    geom_line(linewidth=1)+
    geom_point(size=3,
               shape=21,
               aes(color=cases,
                   fill=cases))
```

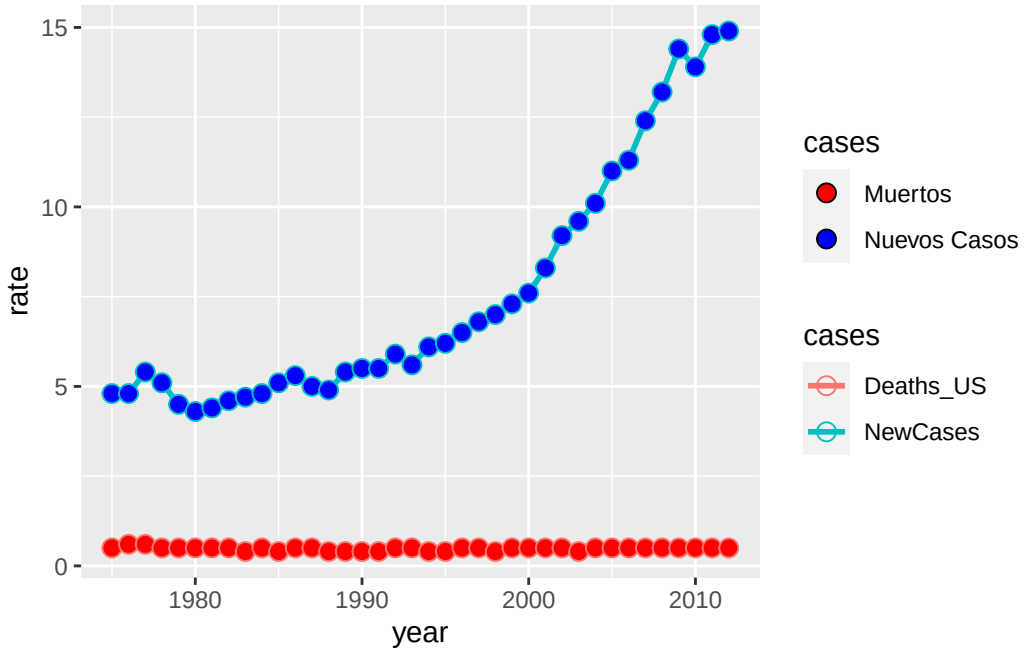


10.1 Modificar leyenda del gráfico

Modificamos las etiquetas de los datos de la leyenda de forma manual con **scale_fill_manual**. En este caso el color esta dado por dos valores, asi que le decimos a ggplot que los dos grupos de valores serán: **labels = c("Muertos", "Nuevos Casos")** para los textos de la leyenda y para los colores de las líneas y circulos **c("red", "blue")**. Ahora se mostrarán dos leyendas de **color" y fill**, por lo que debemos eliminar una de las dos.

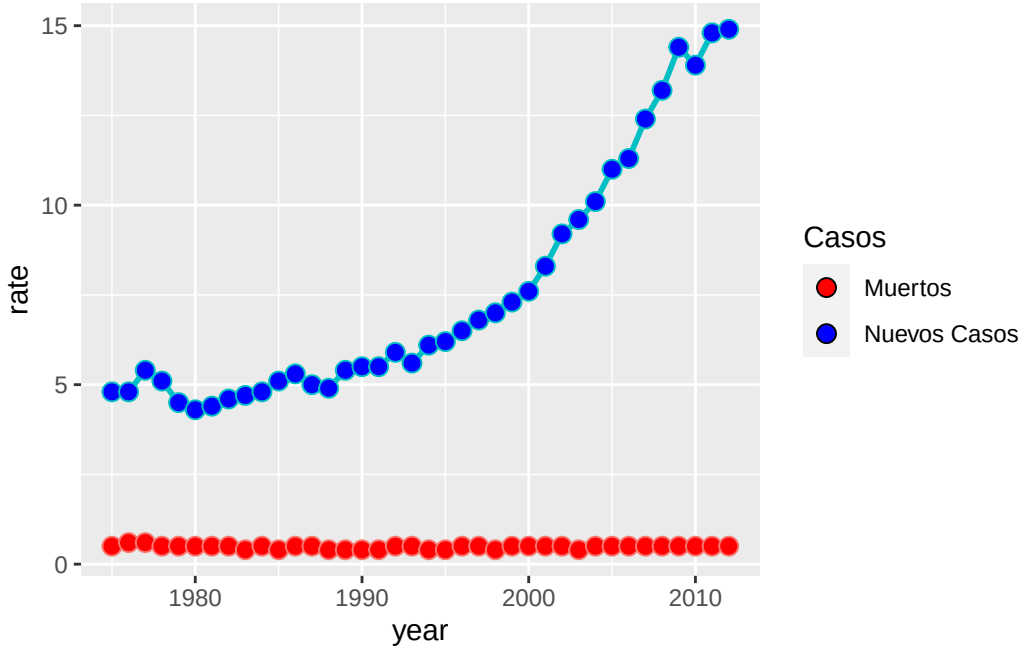
```
data_cases_tiroides |>
  ggplot(aes(year,
             rate,
             group=cases,
             color=cases)) +
    geom_line(size=1)+
    geom_point(size=3,
               shape=21,
               aes(color=cases,
                   fill=cases))+
   scale_fill_manual(labels = c("Muertos", "Nuevos Casos"),
                     values = c("red", "blue"))
```

```
Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
i Please use `linewidth` instead.
```



Eliminamos una de las leyendas con **color = "none"** y modificamos el título de la leyenda que dejamos con **fill = guide_legend("Casos")**. Es importante resaltar que el código **aes(color=cases, fill=cases)** genera los colores según el número de líneas, no puede eliminarse aunque hayamos asignados nuevos colores, esto es porque le dicen al ggplot a través de la variable **cases** asignada que son dos colores que hay que generar, sin esta identificación no se cambiarióa los colores **values = c("red", "blue")**.

```
data_cases_tiroides |>
  ggplot(aes(year, rate,
             group=cases,
             color=cases)) +
    geom_line(size=1)+
    geom_point(size=3,
               shape=21,
               aes(color=cases,
                   fill=cases))+
  scale_fill_manual(labels = c("Muertos", "Nuevos Casos"), values = c("red", "blue"))+
  guides(color = "none",
         fill = guide_legend("Casos"))
```



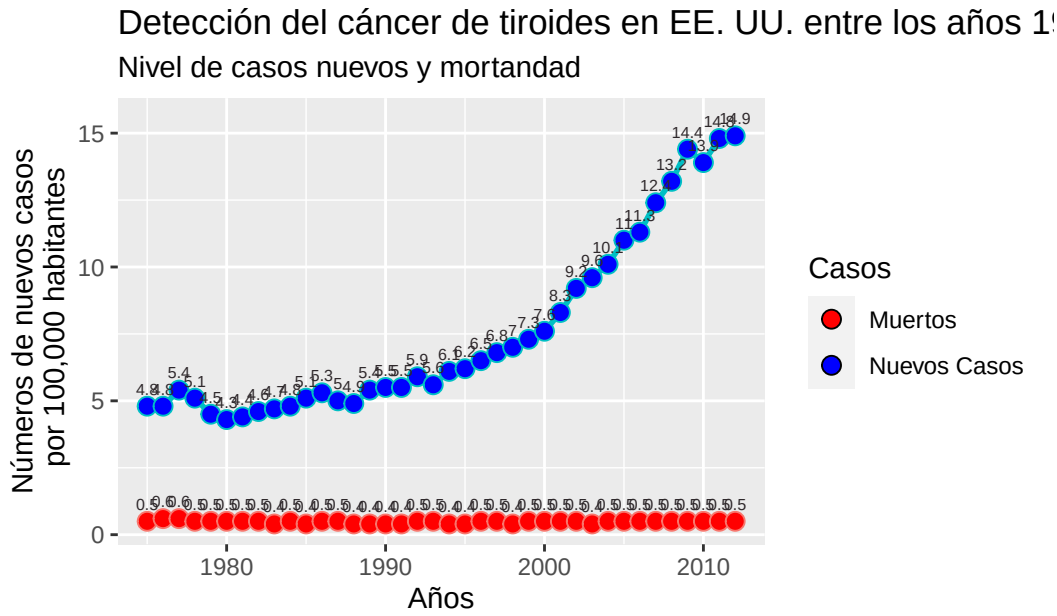
10.1 Añadir componentes al gráfico

Utilizando la función **labs()** podemos añadir título al gráfico, subtitulo, etiquetas a las coordenadas, texto de la fuente de datos y valores en los puntos.

```
data_cases_tiroides |>
  ggplot(aes(year,
             rate,
             group=cases,
             color=cases)) +
    geom_line(size=1)+
    geom_point(size=3,
               shape=21,
               aes(color=cases,
                   fill=cases))+
    scale_fill_manual(labels = c("Muertos", "Nuevos Casos"),
                      values = c("red", "blue"))+
    guides(color = FALSE,
```

```
          fill = guide_legend("Casos")) +
    labs(title="Detección del cáncer de tiroides en EE. UU. entre los años 1975 y 2012",
         subtitle="Nivel de casos nuevos y mortandad",
         caption="fuente: C. I. Bliss (1952) The Statistics of Bioassay. Academic Press.",
         y="Números de nuevos casos\n por 100,000 habitantes",
         x="Años")+
    geom_text(aes(label =  rate),
              nudge_y = 0.65,
              size=2,
              color="#2e282b")
```

```
Warning: The `<scale>` argument of `guides()` cannot be `FALSE`. Use "none" instead as
of ggplot2 3.3.4.
```



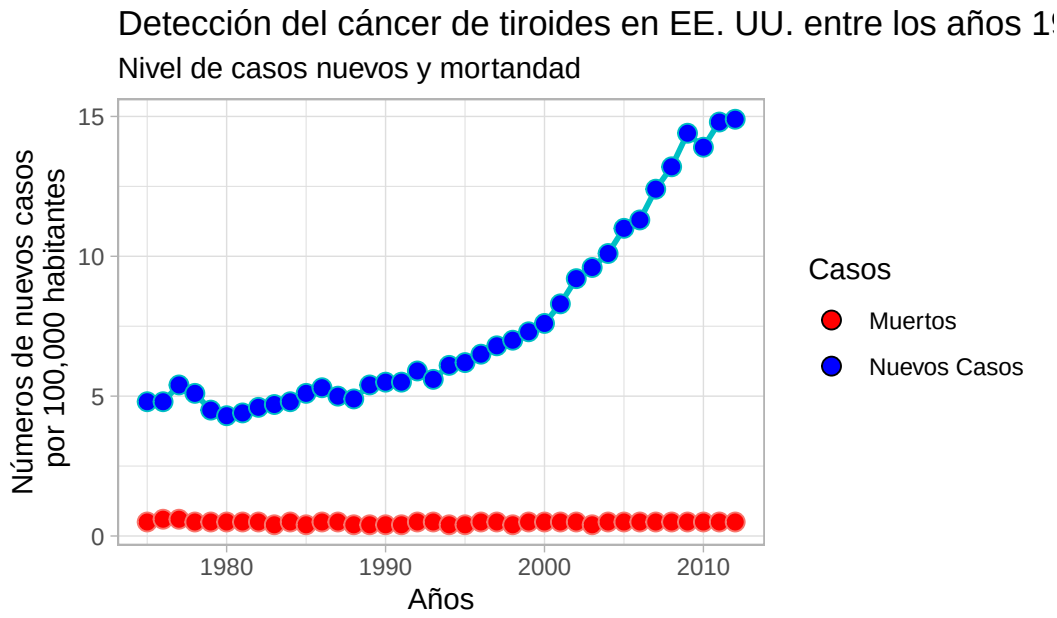
fuente: C. I. Bliss (1952) The Statistics of Bioassay. Academic Press.

10.1 Añadir Themes a los graficos en ggplot

Podemos personalizar el gráfico utilizando algunos de los temas de ggplot. en este ejemplo theme_light().

```
data_cases_tiroides |>
  ggplot(aes(year,
             rate,
             group=cases,
             color=cases)) +
    geom_line(size=1)+
    geom_point(size=3,
               shape=21,
               aes(color=cases,
                   fill=cases))+
    scale_fill_manual(labels = c("Muertos", "Nuevos Casos"),
                      values = c("red", "blue"))+
```

```
guides(color = FALSE,
       fill = guide_legend("Casos")) +
labs(title="Detección del cáncer de tiroides en EE. UU. entre los años 1975 y 2012",
     subtitle="Nivel de casos nuevos y mortandad",
     caption="fuente: C. I. Bliss (1952) The Statistics of Bioassay. Academic Press.",
     y="Números de nuevos casos\n por 100,000 habitantes",
     x="Años")+
theme_light()
```



fuente: C. I. Bliss (1952) The Statistics of Bioassay. Academic Press.

10.1 PRACTICA 1

Genere dos gráfico de líneas uno para cada caso de la variable CASES. Para ello debe hacer dos subgrupo filtrado por cases = =Deaths_US y cases == NewCases , utilizar cun subgrupo en cada gráfico.

10.1 PRACTICA 2

Cree un gráfico de los casos igual al mostrado en este documento, modificando , el color , la forma de los puntos y seleccionando otro theme.

11Gráficos de densidad y Area

Un gráfico de área o densidad visualiza la distribución de datos cuantitativos en un intervalo o período de tiempo continuo. Este gráfico es una variación de un histograma que usa suavizado de kernel para trazar valores, lo que permite distribuciones más suaves suavizando el ruido. Los picos de un gráfico de densidad ayudan a mostrar dónde se concentran los valores durante el intervalo.

11.0 Cargar librerias

Librerias necesarias que se deben haber instalado previamente con **install.packages(nombre del paquete)** o en el panel **Package**.

```
library(tidyverse) # incluye ggplot2
library(readxl) # funciones para importar xlsx
library(janitor) # funciones de limpieza
library(patchwork) #combinar gráficos de ggplot
library(ggExtra)
library(ggthemes) # nuevas temas para los
library(tibble)
library(skimr) # reseumen numerico
library(ggrepel) # añadir etiquetas a los gráficos
library(RColorBrewer) #paletas de colores
library(ggthemes)
```

11.0 Carga de datos

```
# leer datos de titanic
titanic <- read.csv("data/titanic.csv")

titanic |>
  glimpse()
```

```
Rows: 1,309
Columns: 14
$ Pclass    <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ~
$ Survived  <int> 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, ~
$ Name      <chr> "Allen, Miss. Elisabeth Walton", "Allison, Master. Hudson Tr~
$ Sex       <chr> "female", "male", "female", "male", "female", "male", "femal~
$ Age       <dbl> 29.0000, 0.9167, 2.0000, 30.0000, 25.0000, 48.0000, 63.0000,~
$ Sibsp     <int> 0, 1, 1, 1, 1, 0, 1, 0, 2, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, ~
$ Parch     <int> 0, 2, 2, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, ~
$ Ticket    <chr> "24160", "113781", "113781", "113781", "113781", "19952", "1~
$ Fare      <dbl> 211.3375, 151.5500, 151.5500, 151.5500, 151.5500, 26.5500, 7~
$ Cabin     <chr> "B5", "C22 C26", "C22 C26", "C22 C26", "C22 C26", "E12", "D7~
$ Embarked  <chr> "S", "S", "S", "S", "S", "S", "S", "S", "S", "C", "C", "C", ~
$ Boat      <chr> "2", "11", "", "", "", "3", "10", "", "D", "", "", "4", "9",~
$ body      <int> NA, NA, NA, 135, NA, NA, NA, NA, NA, 22, 124, NA, NA, NA, NA~
$ home.dest <chr> "St Louis, MO", "Montreal, PQ / Chesterville, ON", "Montreal~
```

11.0 Significado de las variables del dataframe titanic

survival - Survival (0 = No; 1 = Yes) class - Passenger Class (1 = 1st; 2 = 2nd; 3 = 3rd) name - Name sex - Sex age - Age sibsp - Number of Siblings/Spouses Aboard parch - Number of Parents/Children Aboard ticket - Ticket Number fare - Passenger Fare cabin - Cabin embarked - Port of Embarkation (C = Cherbourg; Q = Queenstown; S = Southampton) boat - Lifeboat (if survived) body - Body number (if did not survive and body was recovered)

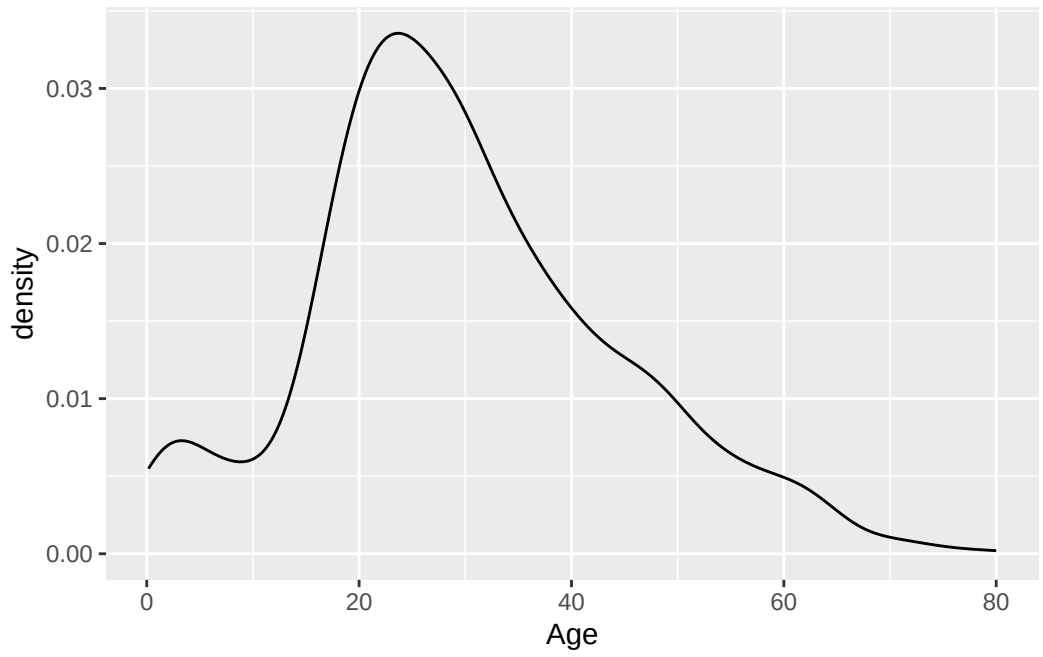
11.1 Gráfico de área en R

Se pueden construir gráfico de área en ggplot2 utilizando la geometría **geom_density()** el cual requiere solo 1 variable numérica como entrada. En este ejemplo se utiliza la variable númerica **Age** y se muestra una distribución de la edad de los pasajeros del Titanic. La mayor densidad de datos está en el rango de edad de 15 a 45.

11.1 Gráfico de histograma de la edad de los pasajeros del titanic

```
# grafico de área
titanic |>
  ggplot(aes(Age))+
  geom_density()
```

```
Warning: Removed 263 rows containing non-finite values (`stat_density()`).
```



Al crear este gráfico de área R envia el mensaje: Warning Removed 263 rows containing non-finite values (stat_bin). Indica que se han omitido 263 valores de la variable Age por ser valores nulos.

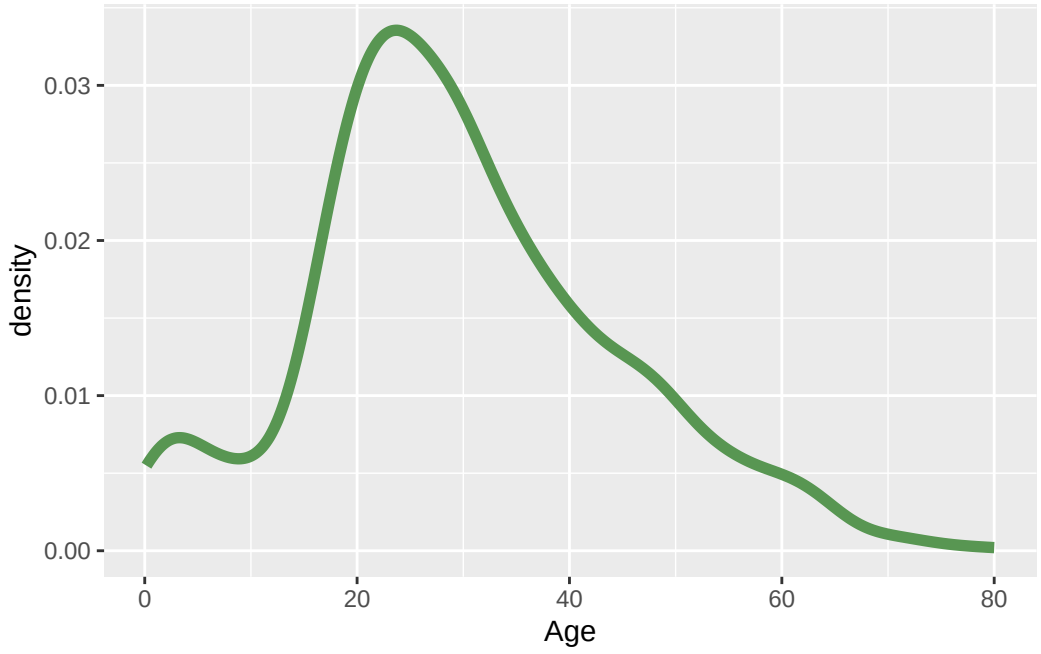
11.1 Modificar color y tamaño de la línea del gráfico

Podemos cambiar el color de la línea del grafico y el tamaño de la línea en la geometría.

```
#graficos de area

titanic |>
  ggplot(aes(Age))+
  geom_density(color="#589652",
               size=2)
```

```
Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
i Please use `linewidth` instead.
```

```
Warning: Removed 263 rows containing non-finite values (`stat_density()`).
```



11.1 Color de relleno en el gráfico

POdemos modificar el color de relleno del gráfico de densidad utilizando nombre de los colores o colores hexadecimales y la propiedad **fill** en la **geom_density()**.

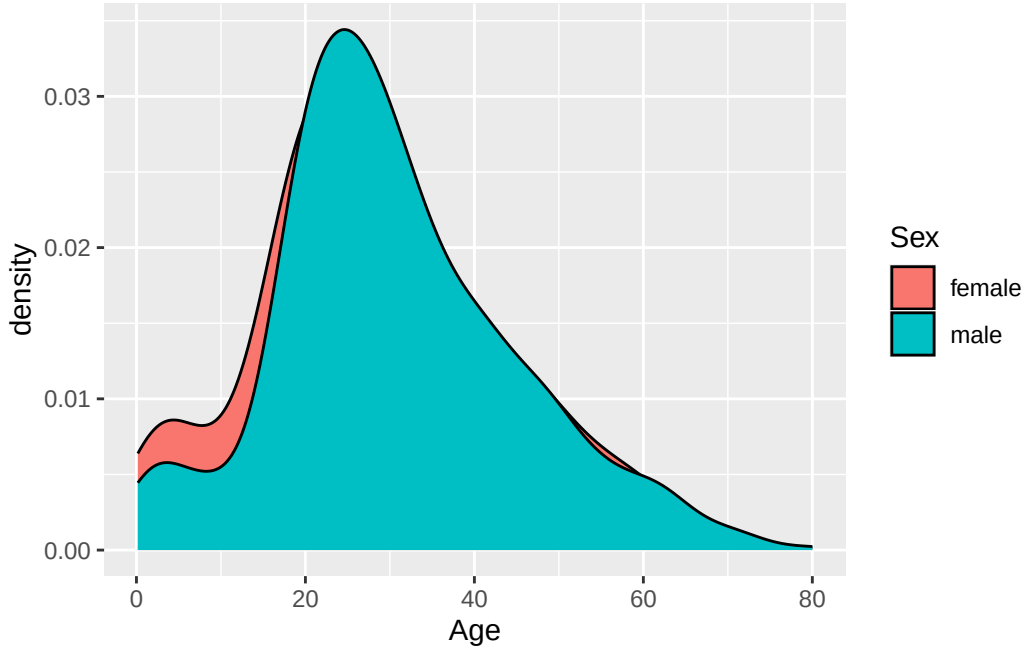
```
#graficos de area
titanic |>
  ggplot(aes(Age))+
  geom_density(color= "red",
               fill="#589652")
```

11.1 Gráfico con variable cuantitativa Age y categorica Sex.

Se utiliza la variable categórica **sex** para crear grupos de datos en las estética de datos a través del relleno del gráfico **fill =Sex**, esto permite agrupar los datos por la variable categórica y generar dos gráficos de área.

```
titanic |>
  ggplot(aes(Age,
             fill=Sex))+
  geom_density()
```

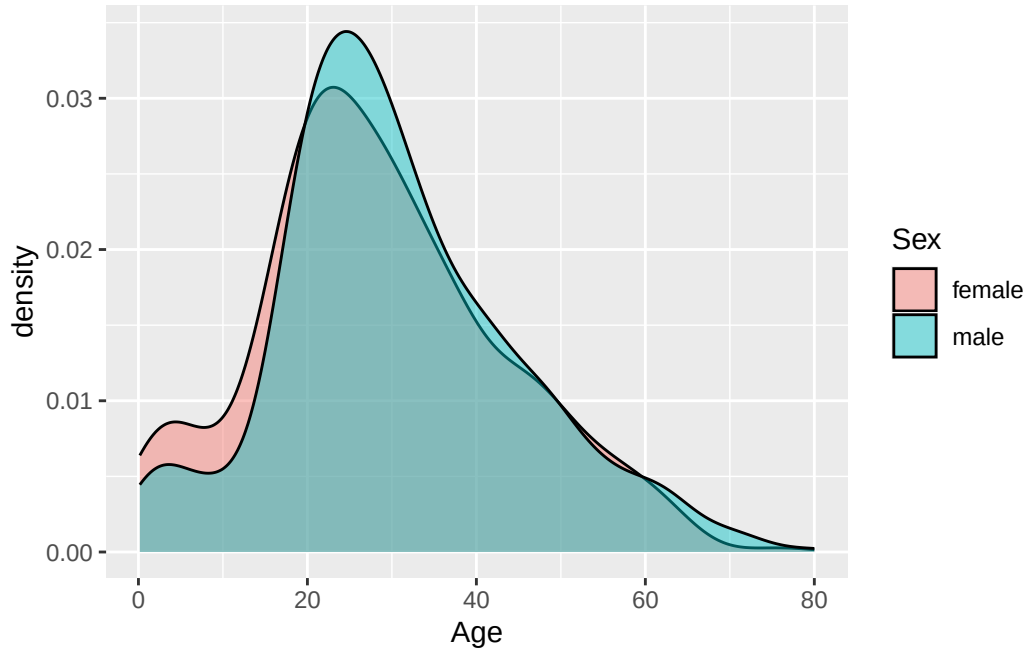
```
Warning: Removed 263 rows containing non-finite values (`stat_density()`).
```



11.2 Superponer gráficos de área

Se puden superponer los gráficos y mostrar ambos utilizando la estética **alpha** en la geometría. En el gráfico de densidad se puede ver que existe una dispersión de datos (ancho del grafico) similar aunque los datos del grupo femenino están ligeramente más dispersos y una centralidad de datos, que indica la media de datos, ligeramente mayor en los hombres. Esto puede indicar que la media de edad de los varones es mayor que el de las mujeres.

```
titanic |>
  ggplot(aes(Age,
             fill=Sex))+
    geom_density(alpha=0.45)
```



11.2 Extrae subgrupo de datos por sexo y calculo de la media

Al crear subgrupo de datos por sexo, obtenemos dos nuevos dataFrames, female y male. Si utilizamos la función **mean()**podemos calcular la media de los datos de la variable Age de la siguiente forma mean(female$Age).

```
#subgrupo de datos por sexo
female <- subset(titanic, titanic$Sex=="female")
male <- subset(titanic, titanic$Sex=="male")
nrow(female)
```

```
[1] 466
```

```
nrow(male)
```

```
[1] 843
```

```r
#media de subgrupo de datos son valores nulos
# el resultado será NA
mean(female$Age)
```

```
[1] NA
```

```r
mean(male$Age)
```

```
[1] NA
```

Si ejecutamos la función **summary()**> podemos ver un resumen de los datos, donde en la variable **Age** indica 263 valores nulos, este impide que podamos utilizar funciones aritméticas como mean() para hacer cálculos.

```r
#Resumen de datos donde la variable AGE tiene valores nulos o NA
summary(titanic)
```

```
    Pclass          Survived          Name                Sex
 Min.   :1.000   Min.   :0.000   Length:1309        Length:1309
 1st Qu.:2.000   1st Qu.:0.000   Class :character   Class :character
 Median :3.000   Median :0.000   Mode  :character   Mode  :character
 Mean   :2.295   Mean   :0.382
 3rd Qu.:3.000   3rd Qu.:1.000
 Max.   :3.000   Max.   :1.000

     Age              Sibsp            Parch          Ticket
 Min.   : 0.1667   Min.   :0.0000   Min.   :0.000   Length:1309
 1st Qu.:21.0000   1st Qu.:0.0000   1st Qu.:0.000   Class :character
 Median :28.0000   Median :0.0000   Median :0.000   Mode  :character
 Mean   :29.8811   Mean   :0.4989   Mean   :0.385
 3rd Qu.:39.0000   3rd Qu.:1.0000   3rd Qu.:0.000
 Max.   :80.0000   Max.   :8.0000   Max.   :9.000
 NA's   :263
     Fare            Cabin             Embarked            Boat
 Min.   :  0.000   Length:1309        Length:1309        Length:1309
 1st Qu.:  7.896   Class :character   Class :character   Class :character
 Median : 14.454   Mode  :character   Mode  :character   Mode  :character
 Mean   : 33.295
 3rd Qu.: 31.275
 Max.   :512.329
 NA's   :1
     body        home.dest
 Min.   :  1.0   Length:1309
 1st Qu.: 72.0   Class :character
 Median :155.0   Mode  :character
 Mean   :160.8
 3rd Qu.:256.0
 Max.   :328.0
 NA's   :1188
```

11.2 Eliminar valores nulos NA y calculo de la media de grupos Sex

Podemos utilizar dentro de la funciones aritméticas la función na.omit(variable) para eliminar los valores nulos de la variable Age. De esta forma calculamos la media de datos para identificar cual grupo tiene un valor central mayor: female: 28.68 male 30.58

```
#media de subgrupo de datos sin valores nulos
mean(na.omit(female$Age))
```

```
[1] 28.68707
```

```
mean(na.omit(male$Age))
```

```
[1] 30.58523
```

```
#mediana
median(na.omit(female$Age))
```

```
[1] 27
```

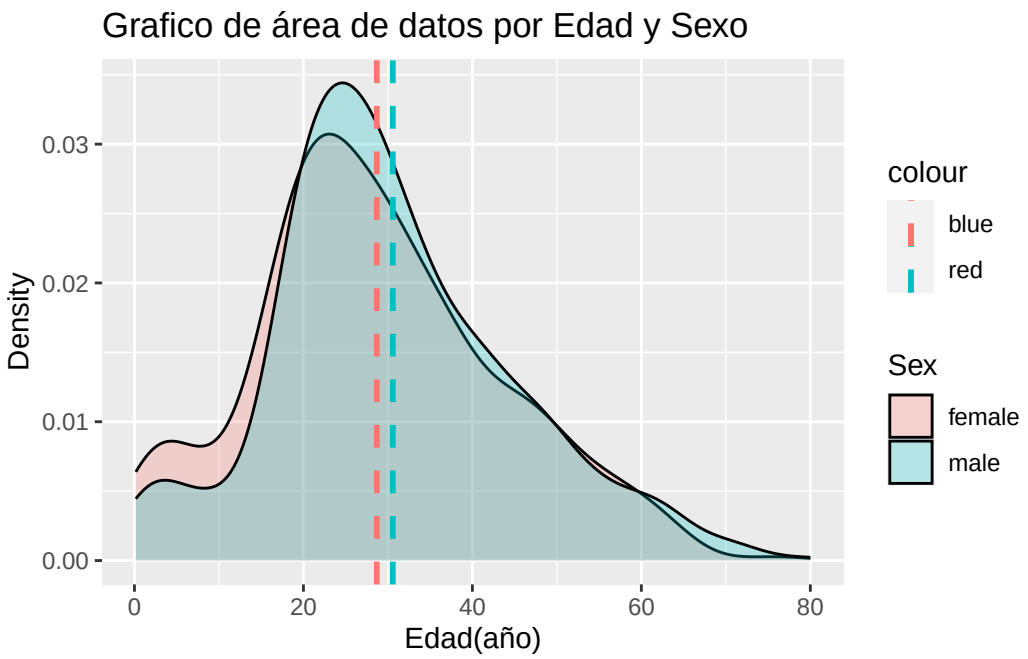
```
median(na.omit(male$Age))
```

```
[1] 28
```

11.2 Datos de la media en el gráfico

Mostrar la media de los datos por grupo **sex** en el gráfico de densidad con la capa **geom_vline()**. Se añadio el título y se cambio los valores de la etiqueta del eje x, como tambien se modificó la etiqueta de la leyenda.

```
titanic |>
  ggplot(aes(Age,
             fill=Sex))+
  geom_density(alpha=0.25)+
  geom_vline(aes(xintercept=mean(na.omit(female$Age)),
                 color="blue"),
             linetype="dashed", size=1) +
  geom_vline(aes(xintercept=mean(na.omit(male$Age)),
                 color="red"),
             linetype="dashed", size=1) +
  labs(title="Grafico de área de datos por Edad y Sexo",
       x="Edad(año)",
       y = "Density")
```



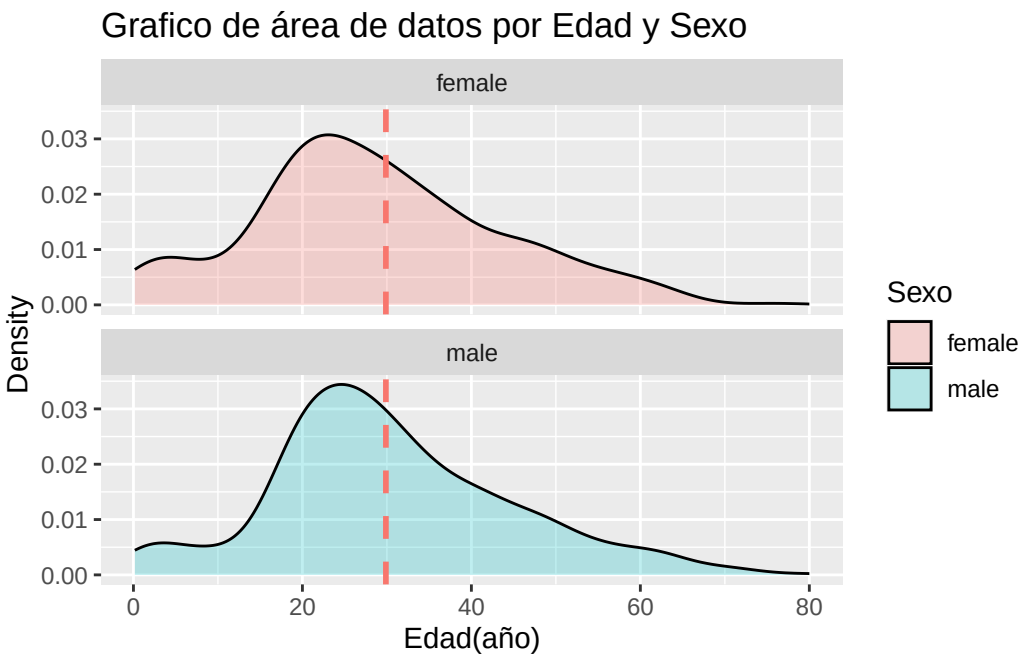
11.2 Gráfico de área por edad en faceta de la variable Sexo

En el siguiente ejemplo utilizaremos para separar los gráficos de área las facetas, particularmente la función **facet_wrap()** con la variable categórica **sex** de la siguiente forma, **facet_wrap(~Sex, ncol=1)**, esto generá las facetas por fila, pero debemos utilizar el valor **n=col** para indicar que solo tendrá una columna.

```
titanic |>
  ggplot(aes(Age,
             fill=Sex))+
  geom_density(alpha=0.25)+
  geom_vline(aes(xintercept=mean(na.omit(Age)),
                 color="blue"),
             linetype="dashed",
             size=1) +
  labs(title="Grafico de área de datos por Edad y Sexo",
       x="Edad(año)",
       y = "Density")+
  guides(fill = guide_legend("Sexo"),
         color = FALSE)+
  facet_wrap(~Sex, ncol=1)
```

```
Warning: The `<scale>` argument of `guides()` cannot be `FALSE`. Use "none" instead as
of ggplot2 3.3.4.
```

```
Warning: Removed 263 rows containing non-finite values (`stat_density()`).
```



11.2 Gráfico de área por edad en faceta de la variable Sexo - opcion 2

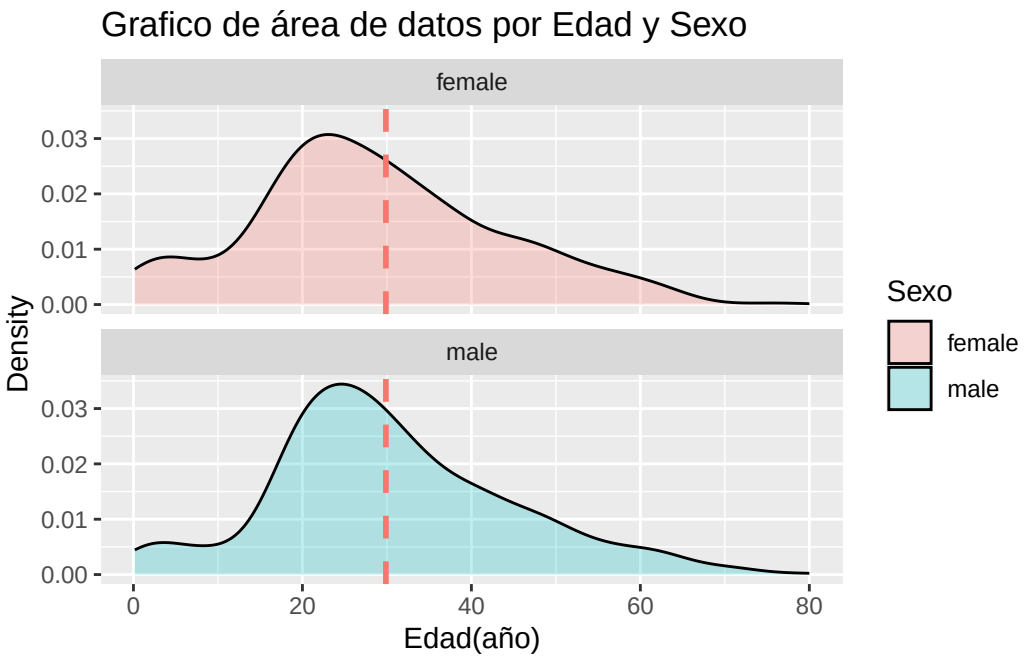
Utilizaremos una sola función **geom_vline()**, adicional que el cálculo de la media no se realizará por cada grupo, sino solo de la variable seleccionada **mean(na.omit(Age)**. El resultado muestra en apariencia que la línea de la **media** del sexo femenino es igual a la del sexo masculino, y esto es un error en el gráfico.

```
media_ <- titanic |>
  group_by(Sex) |>
  summarise(media=mean(na.omit(Age)))

media_
```

```
# A tibble: 2 x 2
  Sex    media
  <chr>  <dbl>
1 female  28.7
2 male    30.6
```

```
titanic |>
  ggplot(aes(Age,
             fill=Sex))+
  geom_density(alpha=0.25)+
  geom_vline(aes(xintercept=mean(na.omit(Age)),
                 color="red"),
             linetype="dashed",
             size=1) +
  labs(title="Grafico de área de datos por Edad y Sexo",
       x="Edad(año)",
       y = "Density")+
  guides(fill = guide_legend("Sexo"),
         color = FALSE)+
```

```
  facet_wrap(~Sex, ncol=1)
```



Este error lo podemos corregir creando un nuevo dataframe que calcula la media por cada categoria de la variable **sex**, este dataframe llamado **mean_**, lo utilzaremos con la función **geom_vline(data=media_, aes(xintercept=media, color=Sex))** , donde los valores de **data** representan el dataframe calculado, xintercept, contiene el cálculo de la media por **sex y color**.

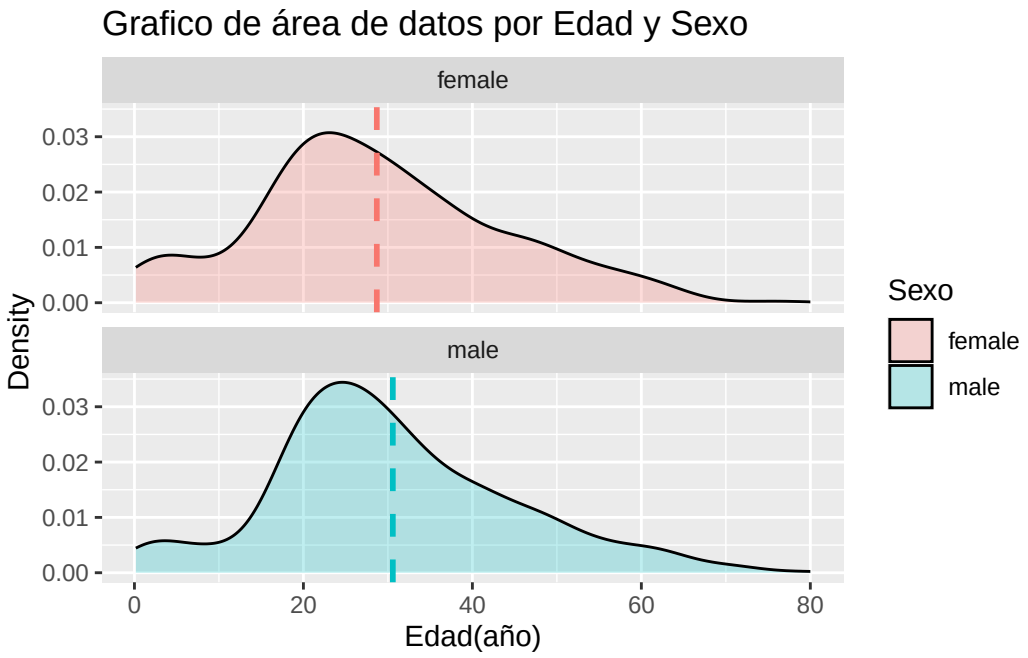
```
# calculo de la media por sexo
media_ <- titanic |>
  group_by(Sex) |>
  summarise(media=mean(na.omit(Age)))

media_
```

```
# A tibble: 2 x 2
  Sex    media
  <chr>  <dbl>
1 female  28.7
2 male    30.6
```

```
titanic |>
  ggplot(aes(Age,
             fill=Sex))+
  geom_density(alpha=0.25)+
  geom_vline(data=media_,
             aes(xintercept=media,
                 color=Sex),
             linetype="dashed",
             size=1) +
  labs(title="Grafico de área de datos por Edad y Sexo",
       x="Edad(año)",
```

```
       y = "Density")+
  guides(fill = guide_legend("Sexo"),
         color = FALSE)+
  facet_wrap(~Sex, ncol=1)
```

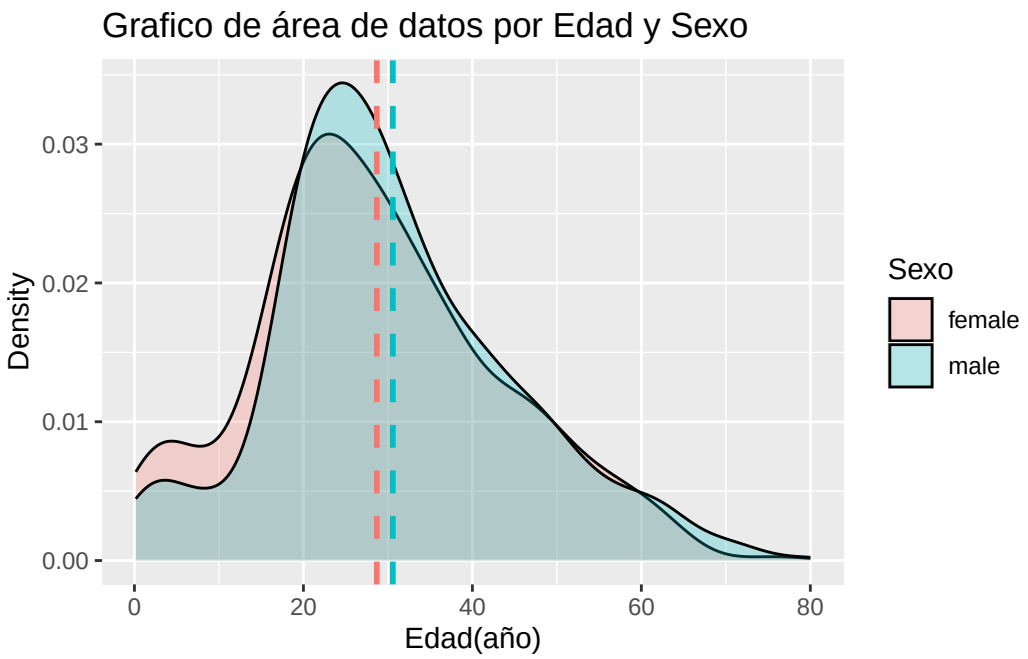
```
Warning: Removed 263 rows containing non-finite values (`stat_density()`).
```



11.2 Modificar etiqueta SEX y eliminar etiqueta color.

Al generar las líneas de las media de cada grupo, se crea otra leyenda que indican los **colores** de la línea, esta leyenda la omitimos utilizando **guides(color = "none")** .

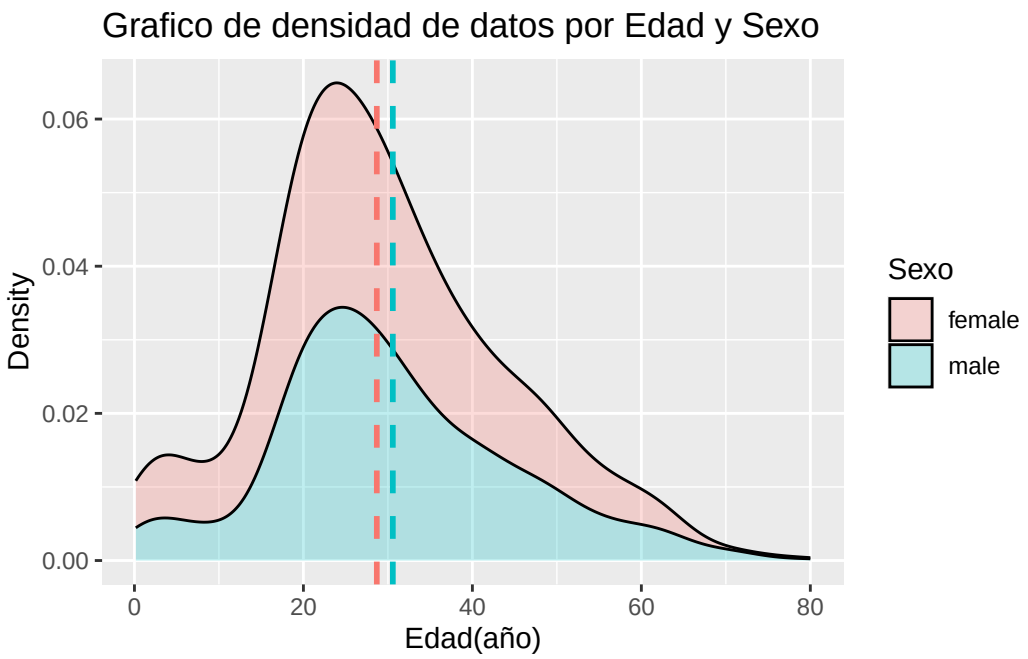
```
titanic |>
  ggplot(aes(Age,
             fill=Sex))+
  geom_density(alpha=0.25)+
  geom_vline(aes(xintercept=mean(na.omit(female$Age)),
                 color="blue"),
             linetype="dashed", size=1) +
  geom_vline(aes(xintercept=mean(na.omit(male$Age)),
                 color="red"),
             linetype="dashed", size=1) +
  labs(title="Grafico de área de datos por Edad y Sexo",
       x="Edad(año)",
       y = "Density")+
  guides(fill = guide_legend("Sexo"),
         color = FALSE)
```



11.2 Agrupar gráficos de densidad

Para agrupar graficos de densidad se debe utilizar la función **position = "stack"** en la geometría, este permite apilar un gráfico sobre otro.

```
titanic |>
  ggplot(aes(Age, fill=Sex))+
  geom_density(alpha=0.25 ,
               position = "stack")+
  geom_vline(aes(xintercept=mean(na.omit(female$Age)),
                 color="blue"),
             linetype="dashed", size=1) +
  geom_vline(aes(xintercept=mean(na.omit(male$Age)),
                 color="red"),
             linetype="dashed", size=1) +
  labs(title="Grafico de densidad de datos por Edad y Sexo",
       x="Edad(año)",
       y = "Density")+
  guides(fill = guide_legend("Sexo"),
         color = "none")
```



11.2 PRÁCTICA 1

Crear un gráfico de densidad con la variable FARE y categorica Sex

11.2 PRACTICA 2

Creo un gráfico de densidad con la variable FARE y categorica Sex Utilice en la geometria la estetica kernel = c(kernel="gaussian").

11.2 PRACTICA 3

Calcule la media y los datos de pasajeros del sexo masculino, mayores y menores que la media. Muestre los 10 primeros datos.

12Temas para gráficos en ggplot

12.1 Cargar librerias

Librerias necesarias que se deben haber instalado previamente con **install.packages(nombre del paquete)** o en el panel **Package**.

```
library(tidyverse) # incluye ggplot2
library(readxl) # funciones para importar xlsx
library(janitor) # funciones de limpieza
library(patchwork) #combinar gráficos de ggplot
library(ggExtra)
library(ggthemes) # nuevas temas para los
library(plotly) #gráficos interactivos # remotes::install_github("plotly/plotly")
library(tibble)
library(skimr) # reseumen numerico
library(modeest)
library(ggrepel) # añadir etiquetas a los gráficos
library(RColorBrewer) #paletas de colores
library(ggthemes)
```

12.2 Carga de datos

Se utilizará los datos de estudio de lectura y matemática de las escuelas de Miami en 2012 y 2013 por grado y Distrito llamado **SchoolsMiamiDade**.

Significado de las variables del dataframe SchoolsMiamiDade Escuelas de Miami con notas promedio en las materias de lectura y matematica en los años 2012 y 2013.

```
SchoolsMiamiDade<- read.csv("data/SchoolsMiamiDade.csv")
SchoolsMiamiDade <- as_tibble(SchoolsMiamiDade)
head(SchoolsMiamiDade,5)
```

```
# A tibble: 5 x 9
  SchoolName BoardDistrict SchoolGrade Reading2012 Reading2013 ReadingDifference
  <chr>      <chr>         <chr>             <int>       <int>              <int>
1 0041 AIR ~ (9)           A                    82          80                 -2
2 0070 CORA~ (9)           A                    71          73                  2
3 0071 EUGE~ (5)           A                    69          69                  0
4 0072 SUMM~ (9)           B                    57          50                 -7
5 0073 MAND~ (9)           C                    34          32                 -2
# i 3 more variables: Math2012 <int>, Math2013 <int>, MathDifference <int>
```

12.3 Temas (Themes) en ggplot

Los temas, son una forma de personalizar los componentes que no son datos en los gráficos, componentes como, títulos, etiquetas, fuentes, fondo, líneas de cuadrícula y leyendas. Se pueden utilizar temas para dar a las tramas un aspecto personalizado coherente. Si bien es posible personalizar los

temas a través de las propiedades de la función **theme()**, ggplot2 integra siete temas que permiten personalizar el gráfico, añadiendo una nueva capa del tema seleccionado al gráfico.

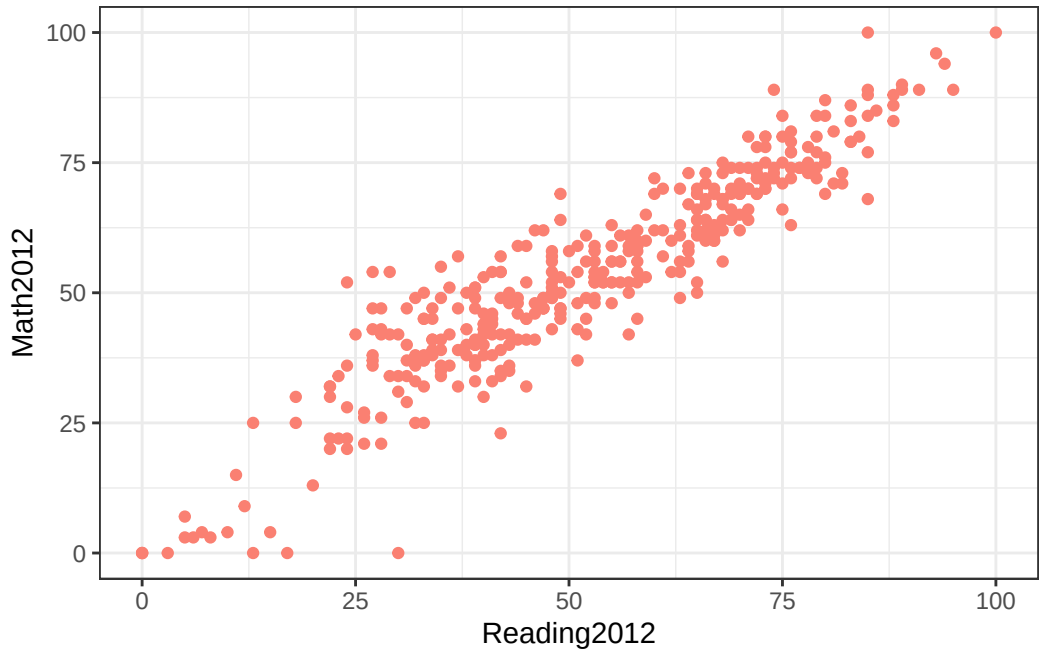
A continuación se mostrará los temas predeterminados de ggplot y otros temas de otras librerias.

12.3 theme_bw()

Es una variación theme_grey() que utiliza un fondo blanco y finas líneas de cuadrícula grises.

```
SchoolsMiamiDade |>
  ggplot(aes(Reading2012,Math2012))+
  geom_point(color="salmon") +
  theme_bw()
```

```
Warning: Removed 91 rows containing missing values (`geom_point()`).
```

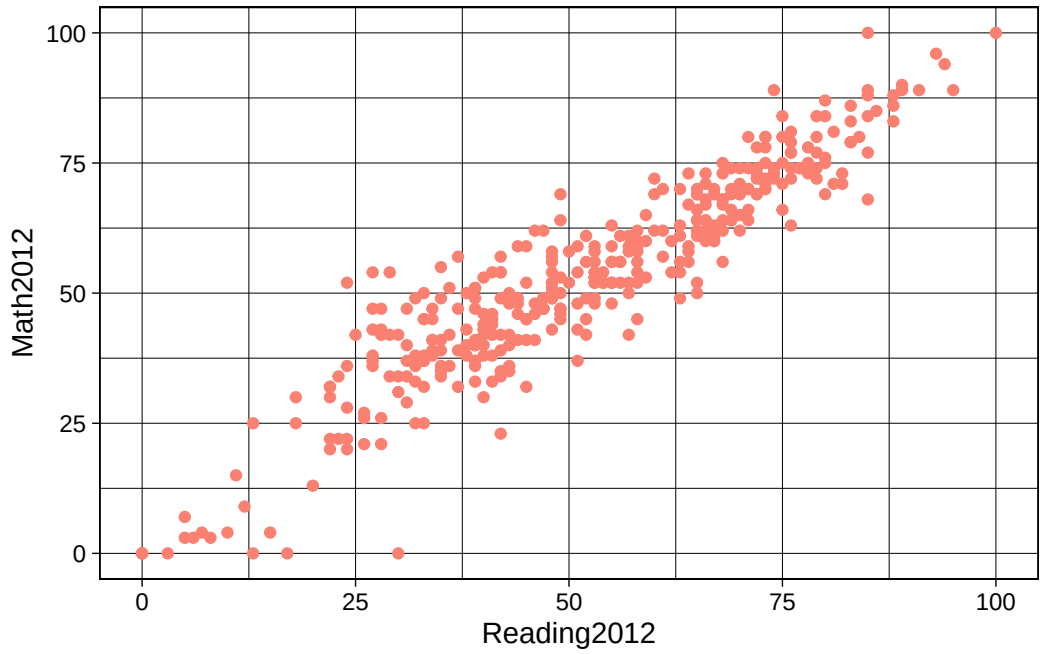


12.3 theme_linedraw()

Tema con solo líneas negras de varios anchos sobre fondos blancos, que recuerda a un dibujo lineal.

```
SchoolsMiamiDade |>
  ggplot(aes(Reading2012,Math2012))+
  geom_point(color="salmon") +
  theme_linedraw()
```

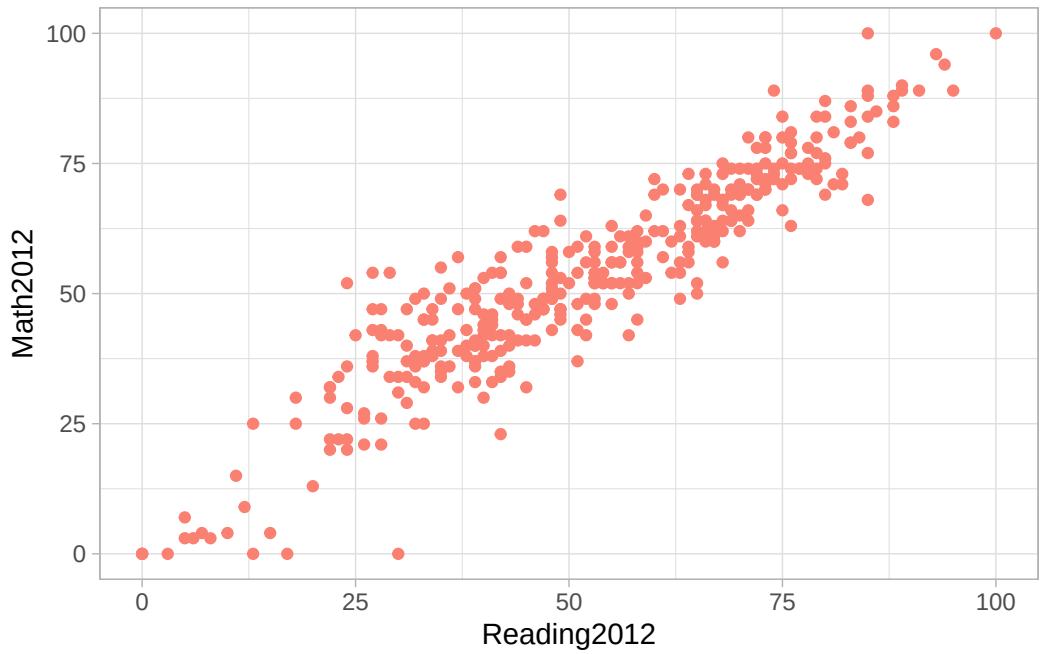
```
Warning: Removed 91 rows containing missing values (`geom_point()`).
```



12.3 theme_light()

Similar theme_linedraw()pero con líneas y ejes de color gris claro, para dirigir más atención a los datos.

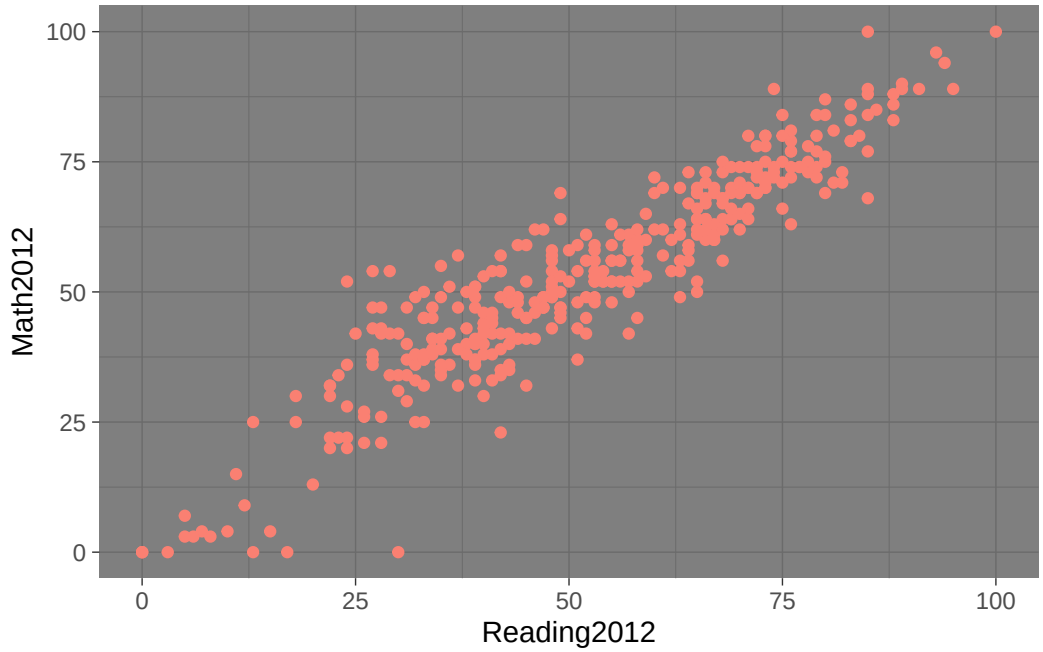
```
SchoolsMiamiDade |>
  ggplot(aes(Reading2012,Math2012))+
  geom_point(color="salmon") +
  theme_light()
```



12.3 theme_dark()

Versión oscura de theme_light(), con tamaños de línea similares pero un fondo oscuro. Útil para hacer resaltar líneas finas de colores.

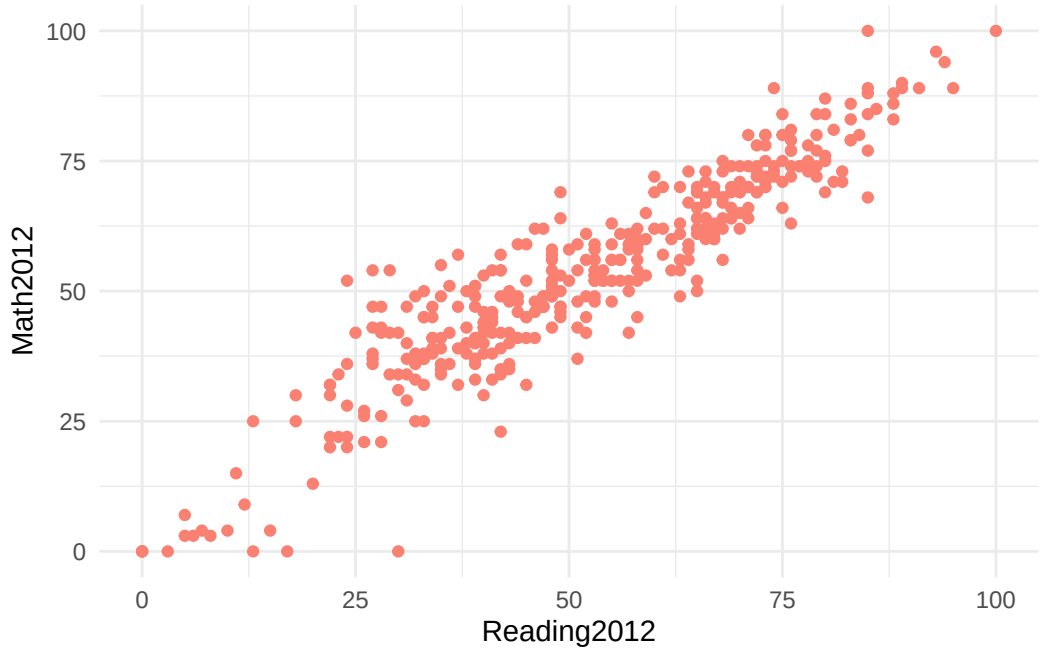
```
SchoolsMiamiDade |>
  ggplot(aes(Reading2012,Math2012))+
  geom_point(color="salmon") +
  theme_dark()
```



12.3 theme_minimal()

Un tema minimalista sin anotaciones de fondo.

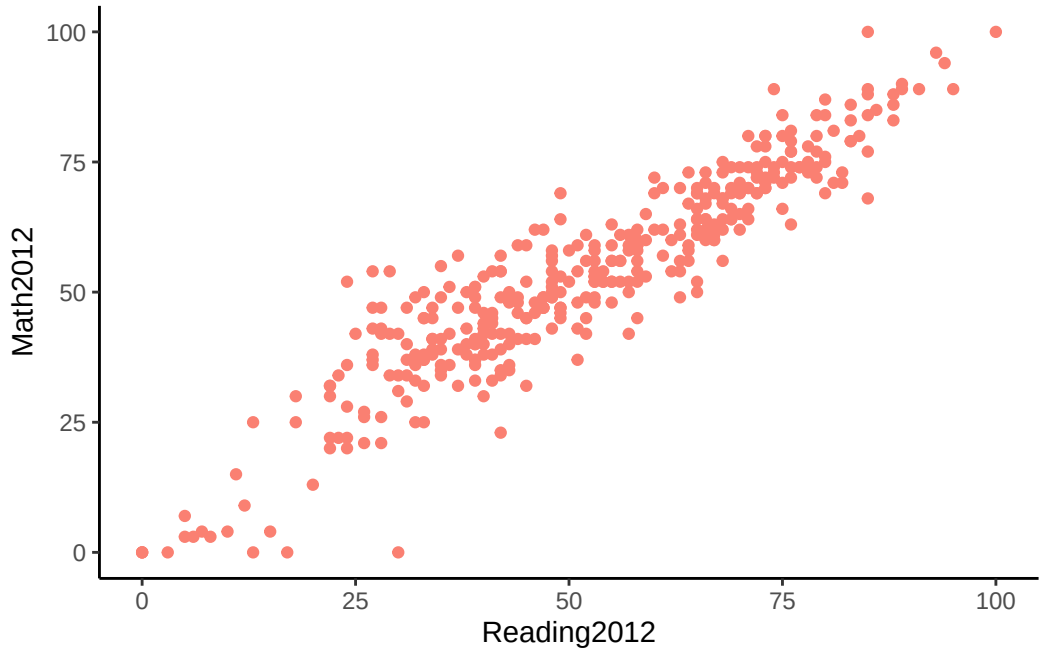
```
SchoolsMiamiDade |>
  ggplot(aes(Reading2012,Math2012))+
  geom_point(color="salmon") +
 theme_minimal()
```



12.3 theme_classic()

Un tema de aspecto clásico, con líneas de eje x e y y sin líneas de cuadrícula.

```
SchoolsMiamiDade |>
  ggplot(aes(Reading2012,Math2012))+
  geom_point(color="salmon") +
  theme_classic()
```



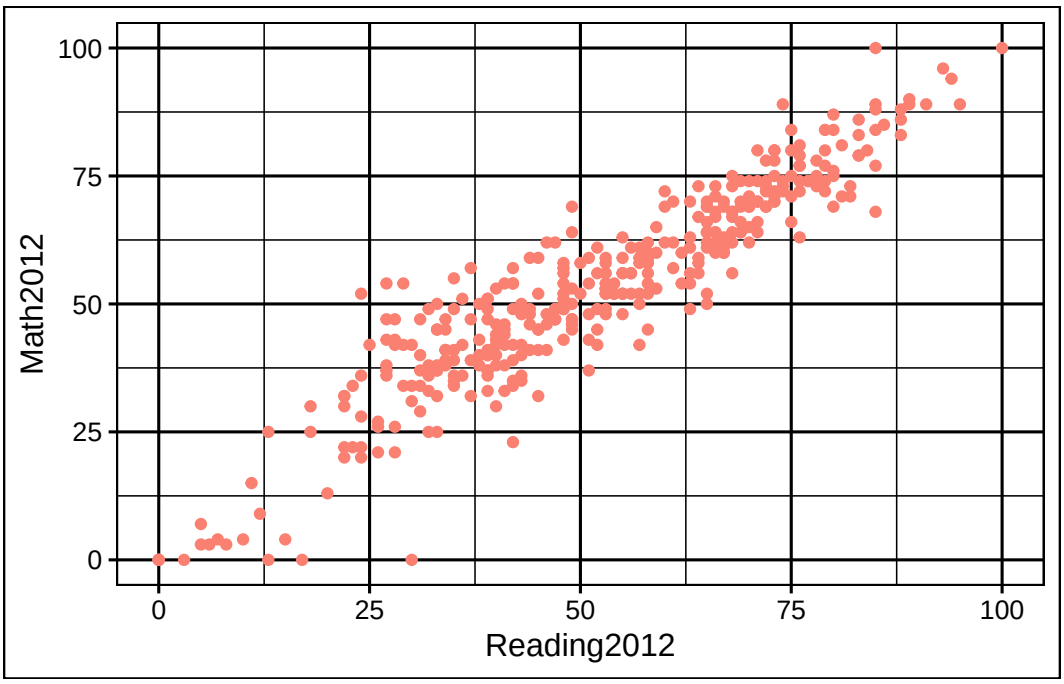
12.4 ggthemes

ggthemes es una libreria que contiene un conjunto de 20 temas diferentes que tienen el mismo funcionamiento de los temas predeterminado de ggplot.

12.4 Ejemplos de temas en ggthemes

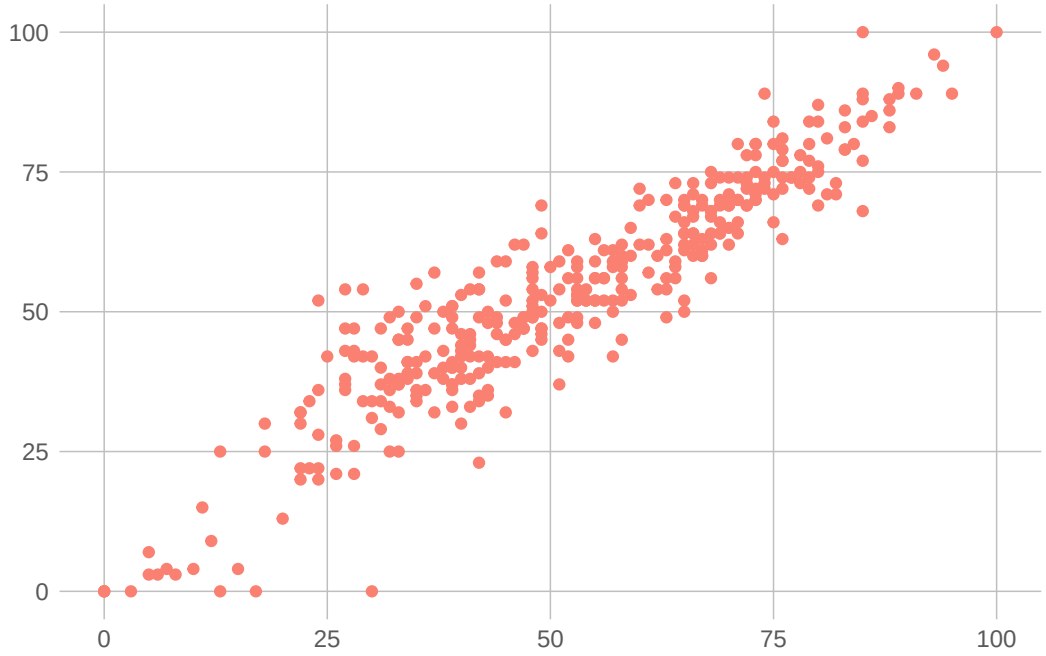
```
SchoolsMiamiDade |>
  ggplot(aes(Reading2012,Math2012))+
  geom_point(color="salmon") +
  theme_foundation()
```

```
Warning: Removed 91 rows containing missing values (`geom_point()`).
```



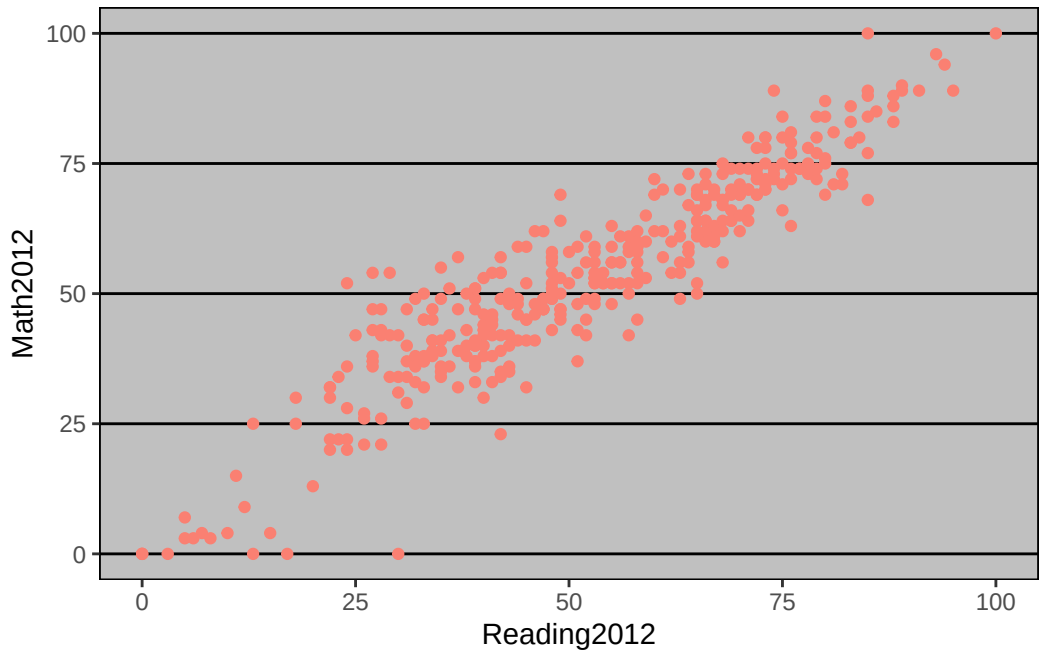

```
SchoolsMiamiDade |>
  ggplot(aes(Reading2012,Math2012))+
  geom_point(color="salmon") +
  theme_excel_new()
```

```
Warning: Removed 91 rows containing missing values (`geom_point()`).
```



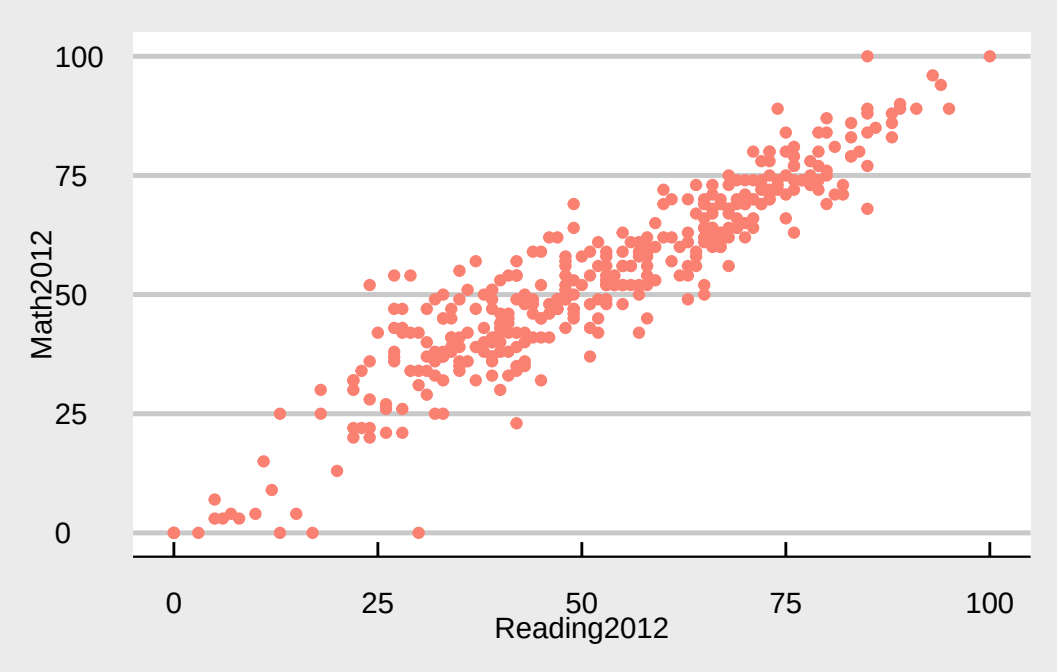

```
SchoolsMiamiDade |>
  ggplot(aes(Reading2012,Math2012))+
  geom_point(color="salmon") +
  theme_excel()
```

```
Warning: Removed 91 rows containing missing values (`geom_point()`).
```



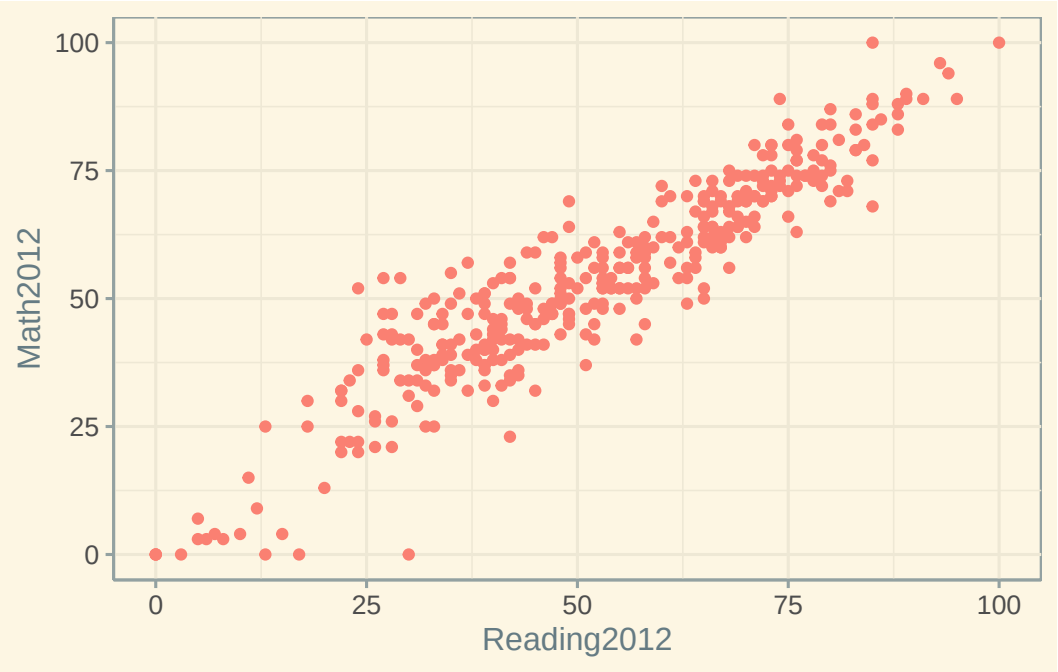

```
SchoolsMiamiDade |>
  ggplot(aes(Reading2012,Math2012))+
  geom_point(color="salmon") +
  theme_economist_white()
```

```
Warning: Removed 91 rows containing missing values (`geom_point()`).
```




```
SchoolsMiamiDade |>
  ggplot(aes(Reading2012,Math2012))+
  geom_point(color="salmon") +
  theme_solarized()
```

```
Warning: Removed 91 rows containing missing values (`geom_point()`).
```

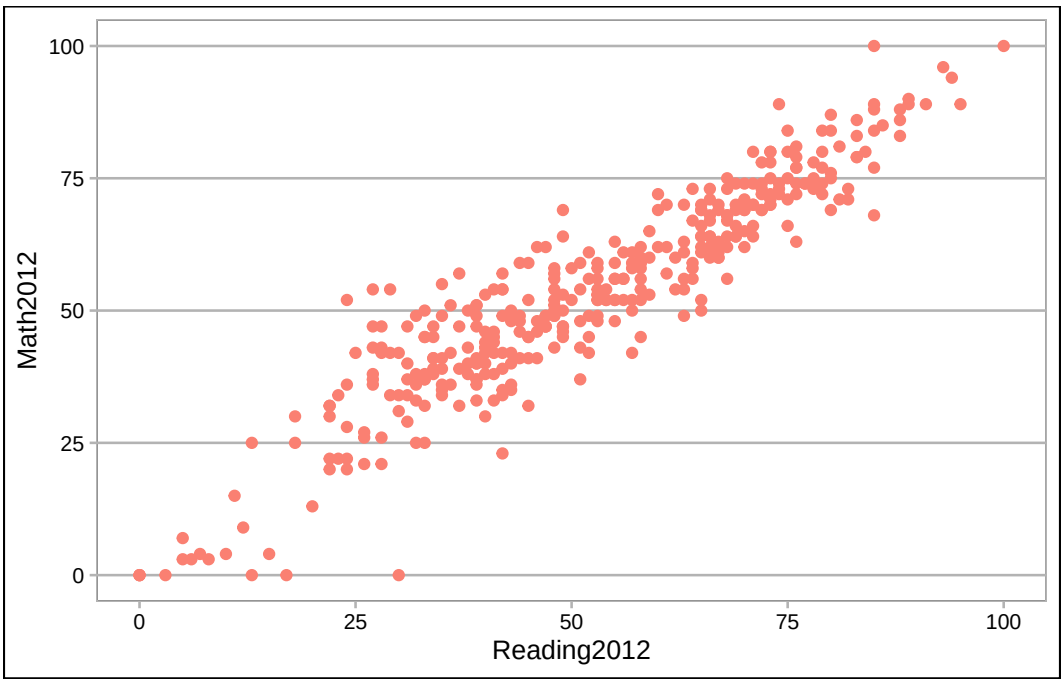


```
SchoolsMiamiDade |>
  ggplot(aes(Reading2012,Math2012))+
  geom_point(color="salmon") +
  theme_economist()
```

```
Warning: Removed 91 rows containing missing values (`geom_point()`).
```

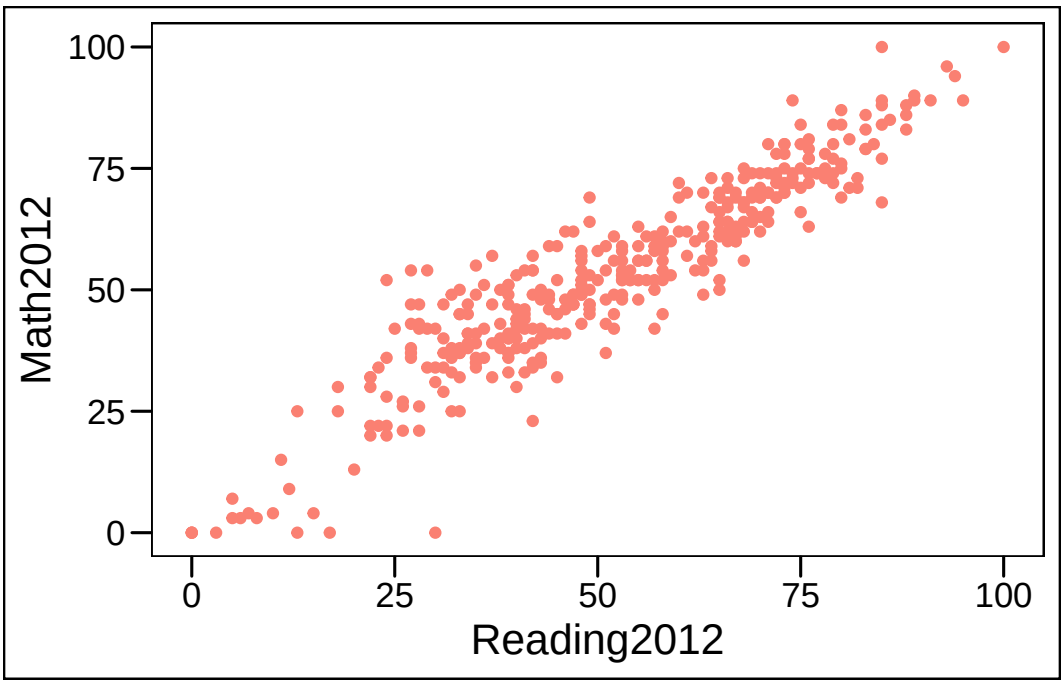

```
SchoolsMiamiDade |>
  ggplot(aes(Reading2012,Math2012))+
  geom_point(color="salmon") +
  theme_calc()
```

```
Warning: Removed 91 rows containing missing values (`geom_point()`).
```




```
SchoolsMiamiDade |>
  ggplot(aes(Reading2012,Math2012))+
  geom_point(color="salmon") +
  theme_base()
```

```
Warning: Removed 91 rows containing missing values (`geom_point()`).
```



Bibliografía

- Quarto
- R for Data Science
- Rmarkdown
- PalmerPenguins
- Rcharts
- Rgraph
- Gallery Book
- ggplot Book
- Awesome ggplot2
- Video Datos pinguinos