Sesión III: Visualizando datos

Guillermo de Anda-Jáuregui y Cristóbal Fresno

Instituto Nacional de Medicina Genómica

2019

¿Por qué visualizamos los datos ?

¿Por qué visualizamos los datos ?

-> Explorar los datos

-> Reconocer patrones

-> Transmitir información

-> Presentar resultados

-> Diferentes primitivas a alto nivel (plot, barplot, hist, boxplot, etc.)

-> Primitivas de bajo nivel (points, lines)

-> Agregación por orden de la llamada a la función (abline, legend, etc.)

-> Gramática de gráficos (Pre-tidyverse)

-> Objetos + Transform + Persistencia

Ejemplo de dispersión

Hagamos el gráfico de dispersión usando iris para Petal.Length vs Petal.Width en versión R-base y con ggplot2:

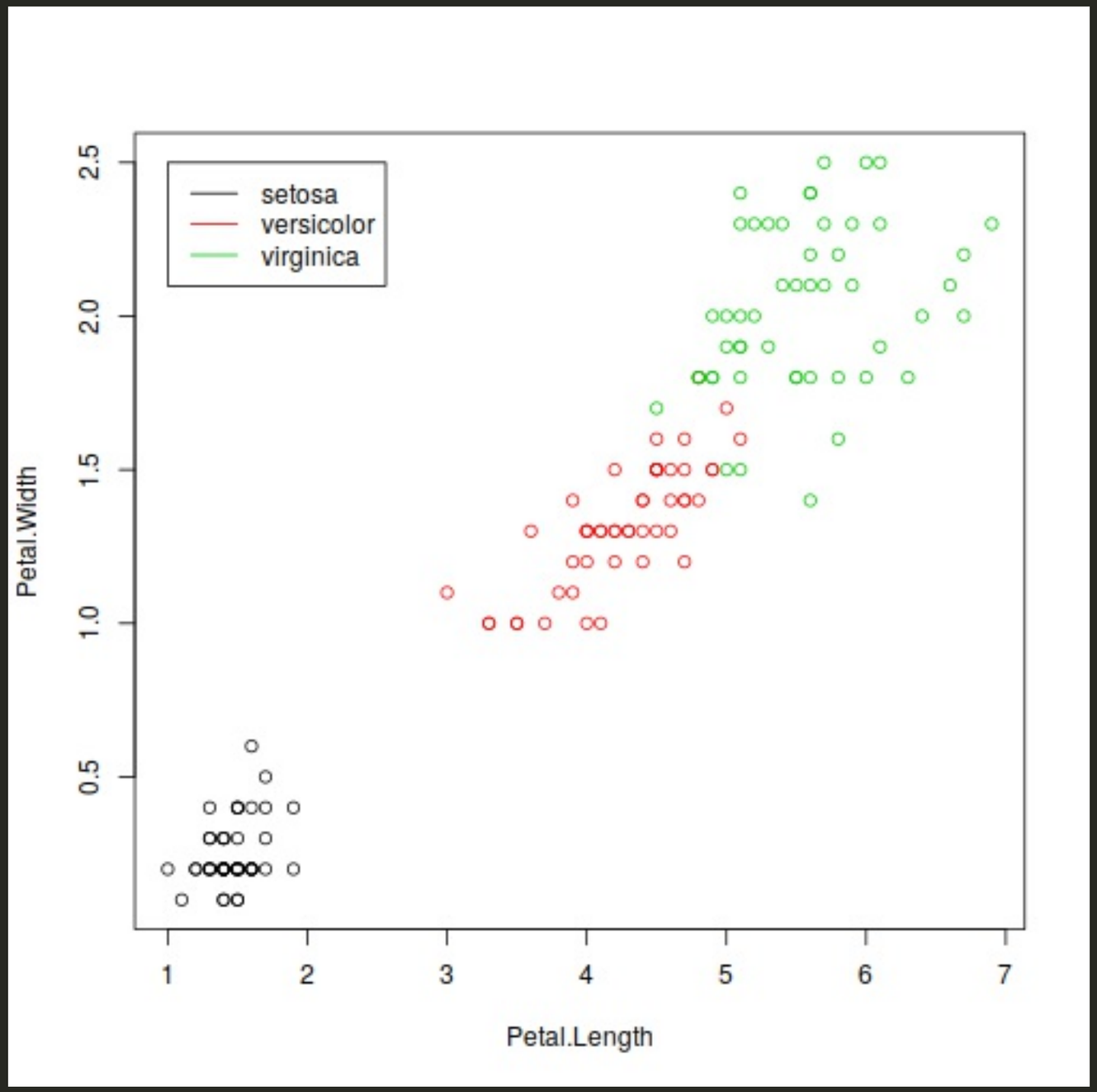
```
iris %>% head
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

Veamos la versión de R-base y la de ggplot2

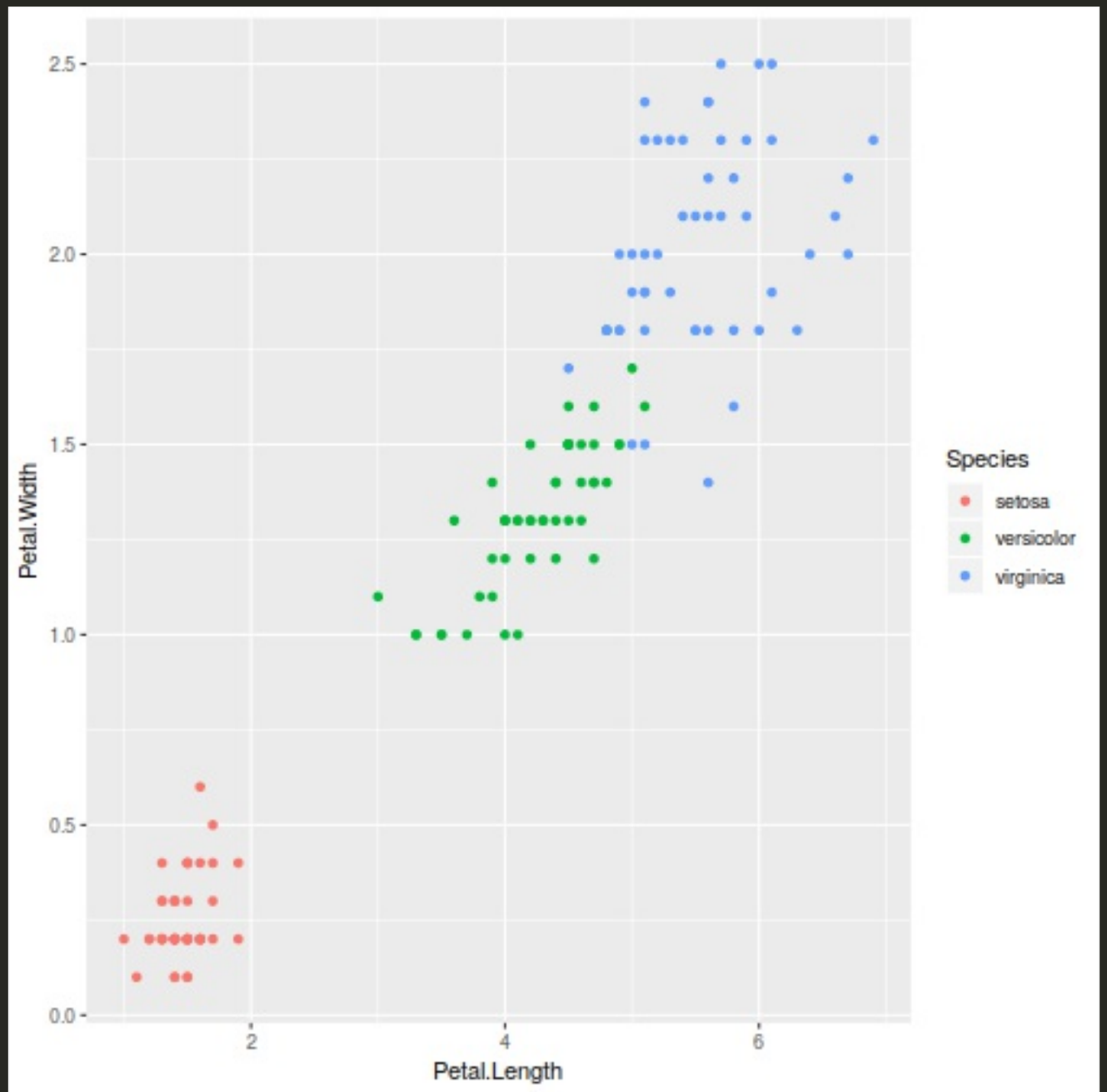
R version

```
plot(
x = iris$Petal.Length,
y = iris$Petal.Width,
col = iris$Species,
type = "p", #for points
xlab = "Petal.Length",
ylab = "Petal.Width"
)
legend(
legend = levels(iris$Species),
col = seq(levels(iris$Species)),
x = 1,
y = 2.5,
lty = 1
)
```


setosa
versicolor
virginica
Petal.Width
0.5
1.0
1.5
2.0
2.5
1
2
3
4
5
6
7
Petal.Length

ggplot version

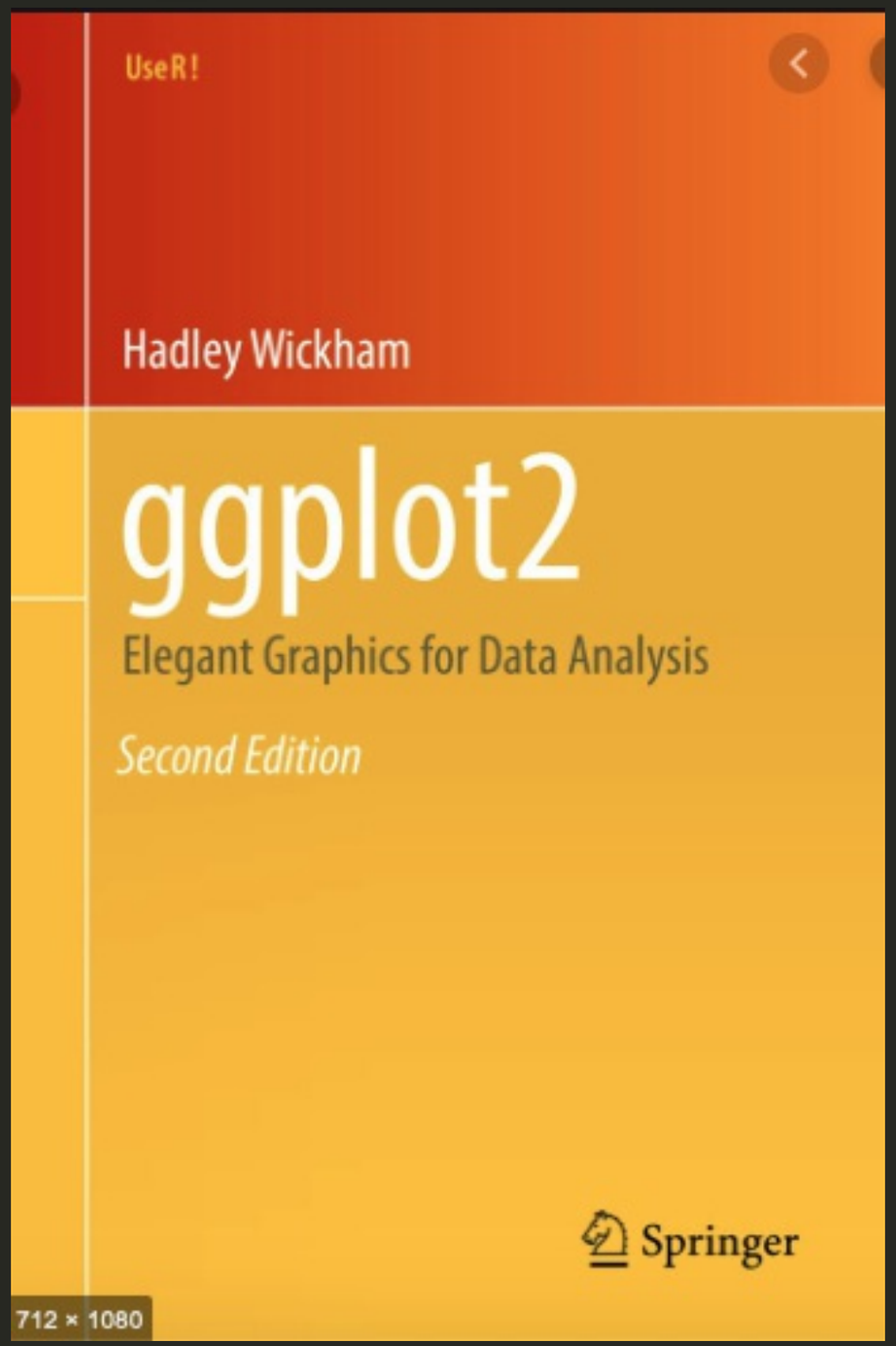
```
iris %>%
  ggplot(mapping = aes(x = Petal.Length,
                       y = Petal.Width,
                       colour = Species
                       )
                       ) +
  geom_point()
```


2.5
2.0
1.5
1.0
0.5
0.0
Petal.Width
2
4
6
Petal.Length
Species
setosa
versicolor
virginica

Hacemos un paréntesis y volvemos sobre este ejemplo ...

ggplot2: Filosofía

https://ggplot2.tidyverse.org/reference/



ggplot2: Filosofía

```
ggplot(data = <DATA>) +
     <GEOM_FUNCTION>(
  mapping = aes(<MAPPINGS>),
       stat = <STAT>,
   position = <POSITION>
          ) +
<COORDINATE_FUNCTION> +
   <FACET_FUNCTION> +
        <THEME>
```

OJO con la maldición de %>% vs "+"


ggplot2:
Build a data
MASTERPIECE
themes
geoms
aes
fill
color
ggAnimate
plotly
HORST '18

"

ggplot2: Gráfica

Debemos explicitar qué datos se utilizan

-> Desde una consecución de comandos

```
Nuestros_Datos %>%
    ggplot()
```

-> De forma explícita

```
ggplot(data = Nuestros_Datos)
```

ggplot2: Mapping

aes

Definen como las variables son mapeadas a las propiedades visuales (estéticas o aesthetic en inglés) de los geom.

ggplot2: PRIMITIVAS

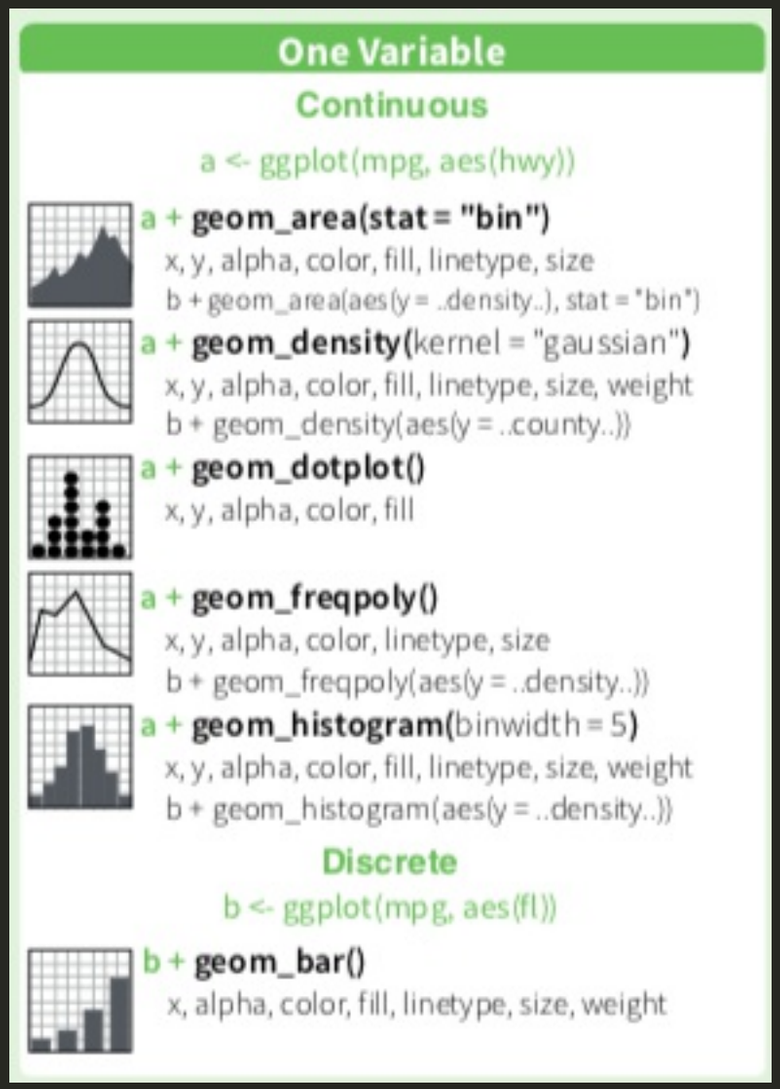
geom

Objetos geométricos que son la representación visual de las observaciones.

ggplot2:

Visual Data Exploration

ggplot2: PRIMITIVAS



Two Variables

Continuous X, Continuous Y

f <- ggplot(mpg, aes(cty, hwy))

f + **geom_blank()**

f + **geom_jitter()**
x, y, alpha, color, fill, shape, size

f + **geom_point()**
x, y, alpha, color, fill, shape, size

f + **geom_quantile()**
x, y, alpha, color, linetype, size, weight

f + **geom_rug(**sides = "bl"**)**
alpha, color, linetype, size

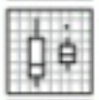
f + **geom_smooth(**model = lm**)**
x, y, alpha, color, fill, linetype, size, weight

f + **geom_text(**aes(label = cty)**)**
x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

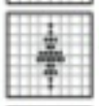
Discrete X, Continuous Y

g <- ggplot(mpg, aes(class, hwy))

g + **geom_bar(stat = "identity")**
x, y, alpha, color, fill, linetype, size, weight



g + **geom_boxplot()**
lower, middle, upper, x, ymax, ymin, alpha, color, fill, linetype, shape, size, weight

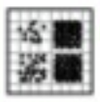


g + **geom_dotplot(**binaxis = "y", stackdir = "center"**)**
x, y, alpha, color, fill

g + **geom_violin(**scale = "area"**)**
x, y, alpha, color, fill, linetype, size, weight

Discrete X, Discrete Y

h <- ggplot(diamonds, aes(cut, color))



h + **geom_jitter()**
x, y, alpha, color, fill, shape, size

Continuous Bivariate Distribution

i <- ggplot(movies, aes(year, rating))

i + **geom_bin2d(**binwidth = c(5, 0.5)**)**
xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size, weight

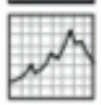
i + **geom_density2d()**
x, y, alpha, colour, linetype, size

i + **geom_hex()**
x, y, alpha, colour, fill size

Continuous Function

j <- ggplot(economics, aes(date, unemploy))

j + **geom_area()**
x, y, alpha, color, fill, linetype, size

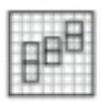


j + **geom_line()**
x, y, alpha, color, linetype, size

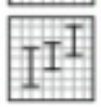
j + **geom_step(**direction = "hv"**)**
x, y, alpha, color, linetype, size

Visualizing error

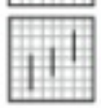
df <- data.frame(grp = c("A", "B"), fit = 4:5, se = 1:2)
k <- ggplot(df, aes(grp, fit, ymin = fit-se, ymax = fit+se))



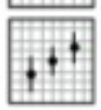
k + **geom_crossbar(**fatten = 2**)**
x, y, ymax, ymin, alpha, color, fill, linetype, size



k + **geom_errorbar()**
x, ymax, ymin, alpha, color, linetype, size, width (also **geom_errorbarh()**)



k + **geom_linerange()**
x, ymin, ymax, alpha, color, linetype, size



k + **geom_pointrange()**
x, y, ymin, ymax, alpha, color, fill, linetype, shape, size

Maps

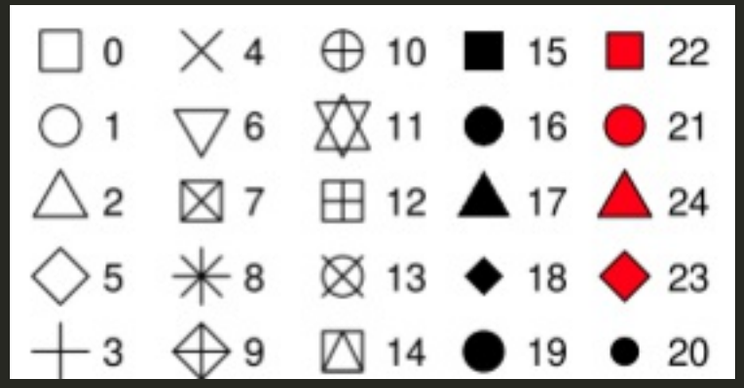
data <- data.frame(murder = USArrests$Murder,
  state = tolower(rownames(USArrests)))
map <- map_data("state")
l <- ggplot(data, aes(fill = murder))

l + **geom_map(**aes(map_id = state), map = map**) +**
expand_limits(x = map$long, y = map$lat**)**
map_id, alpha, color, fill, linetype, size

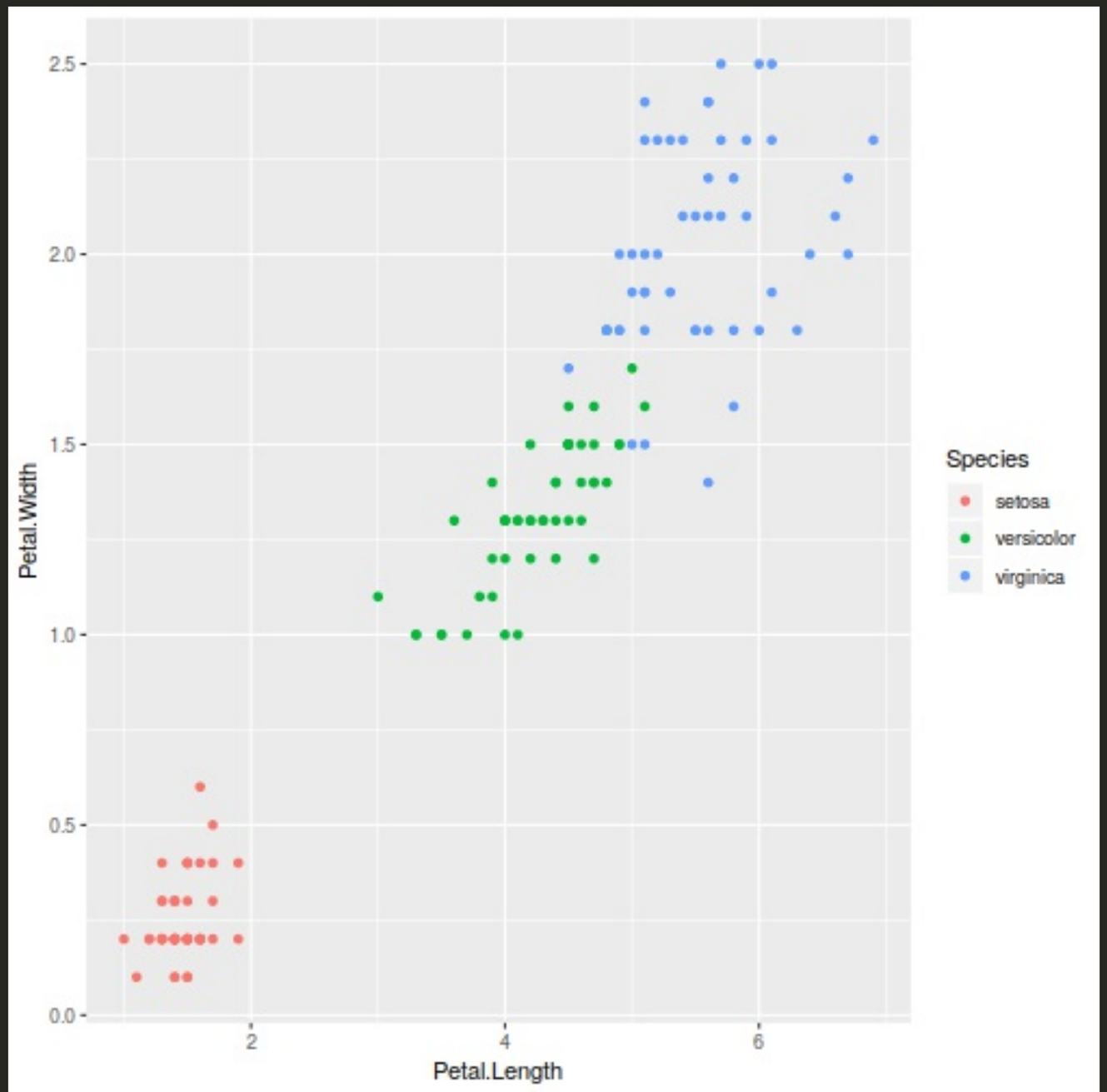
ggplot2: PRIMITIVAS

Tenemos GEOMs para 1, 2 y 3D

ggplot2: PRIMITIVAS - geom_point - shape



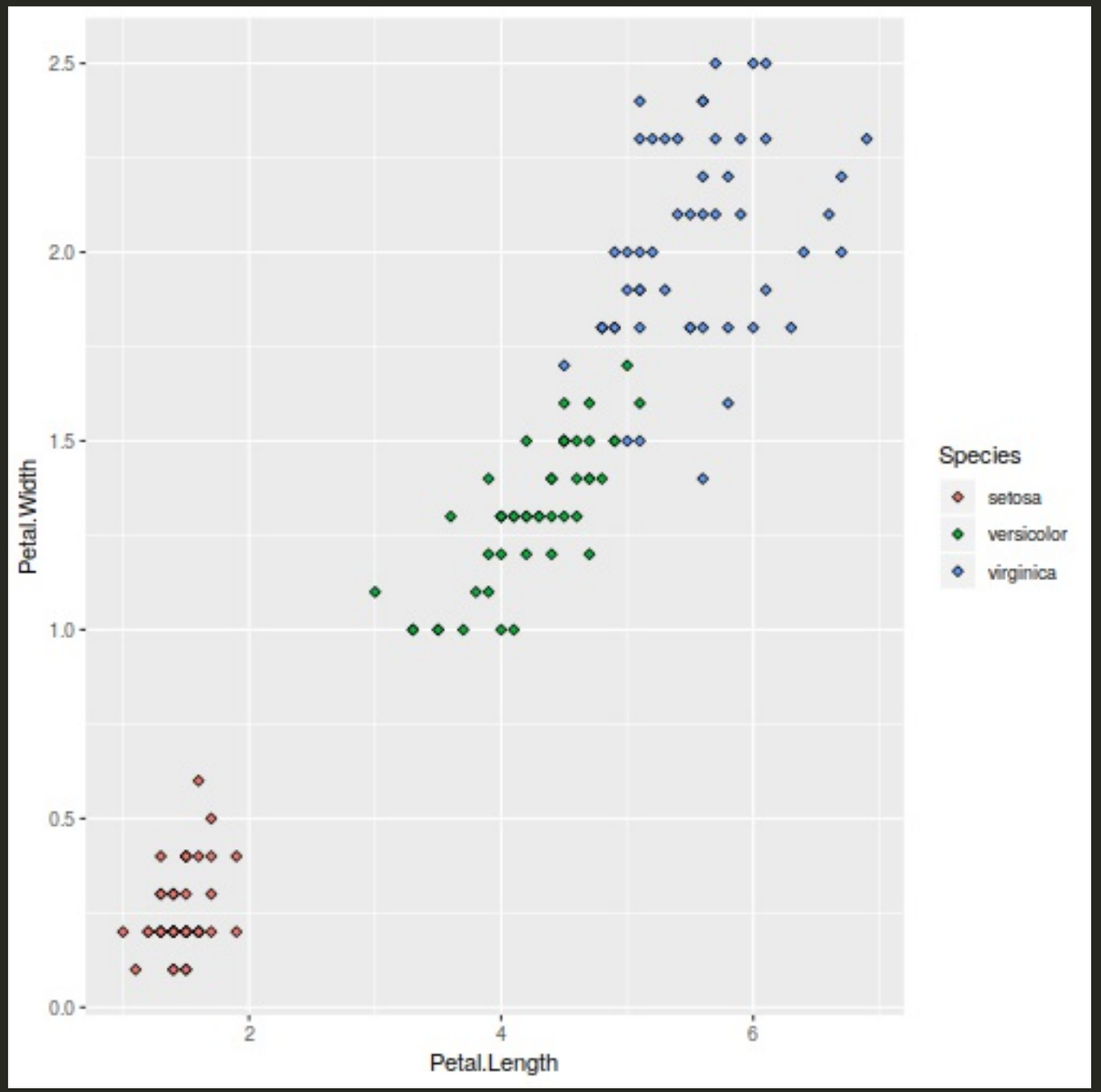
Terminado el paréntesis y volvemos sobre el ejemplo ...


2.5
2.0
1.5
1.0
0.5
0.0
Petal.Width
2
4
6
Petal.Length
Species
setosa
versicolor
virginica

EJERCICIOS:

- Cambie el punto de la gráfica de dispersión para que utilice un rombo relleno de color

¿Qué más le podemos hacer a la gráfica?



Pensemos en un...

-> Título y subtítulo...

-> Leyenda abajo...

-> Quitar el fondo gris...

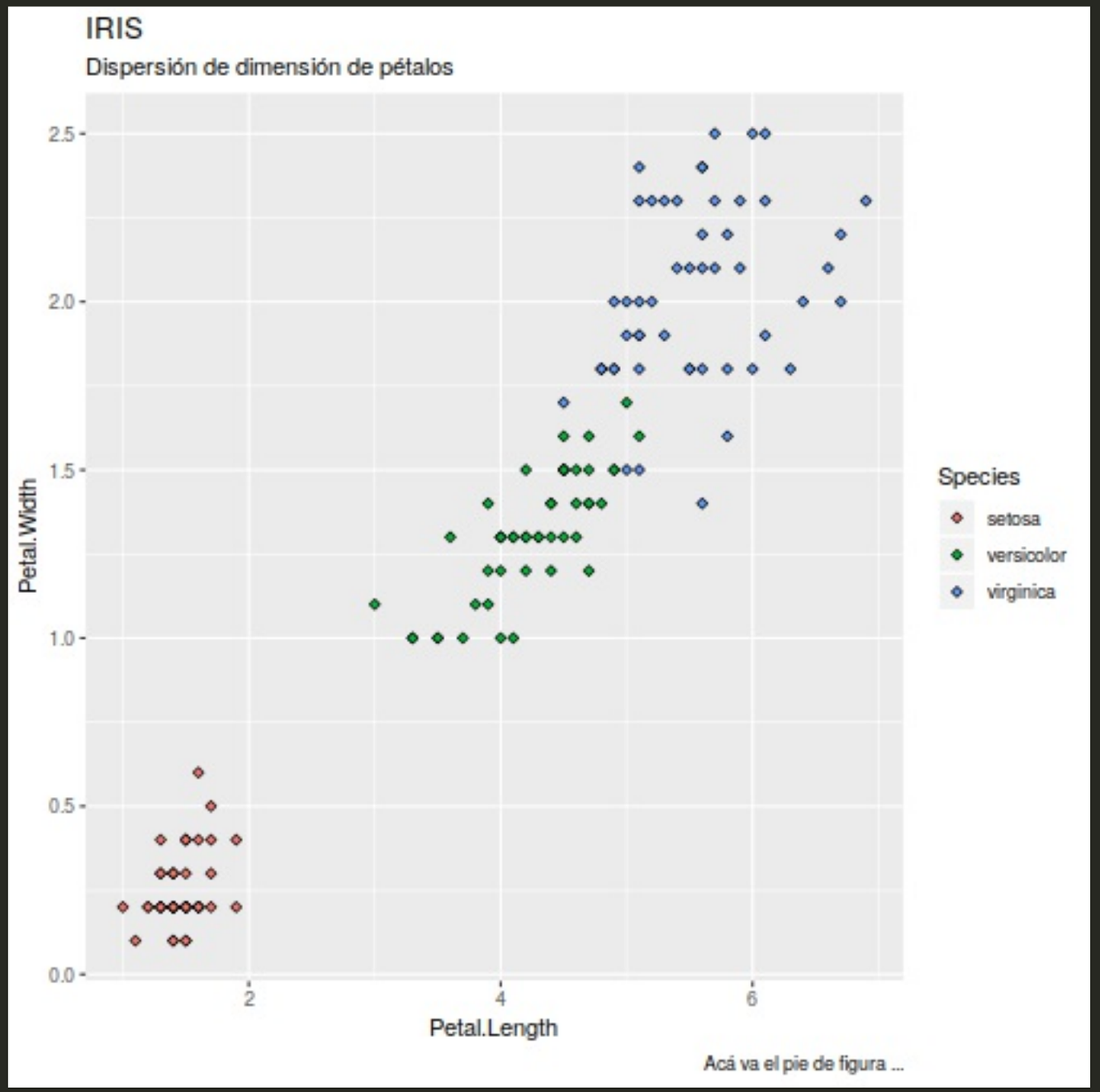
-> Cambiar la paleta de colores...

-> Separar en paneles por Specie...

ggplot2: Títulos y subtítulos

```
    labs(
      title = "Mi título",
   subtitle = "Mi subtítulo",
    caption = "Un pie de figura"
    )
```

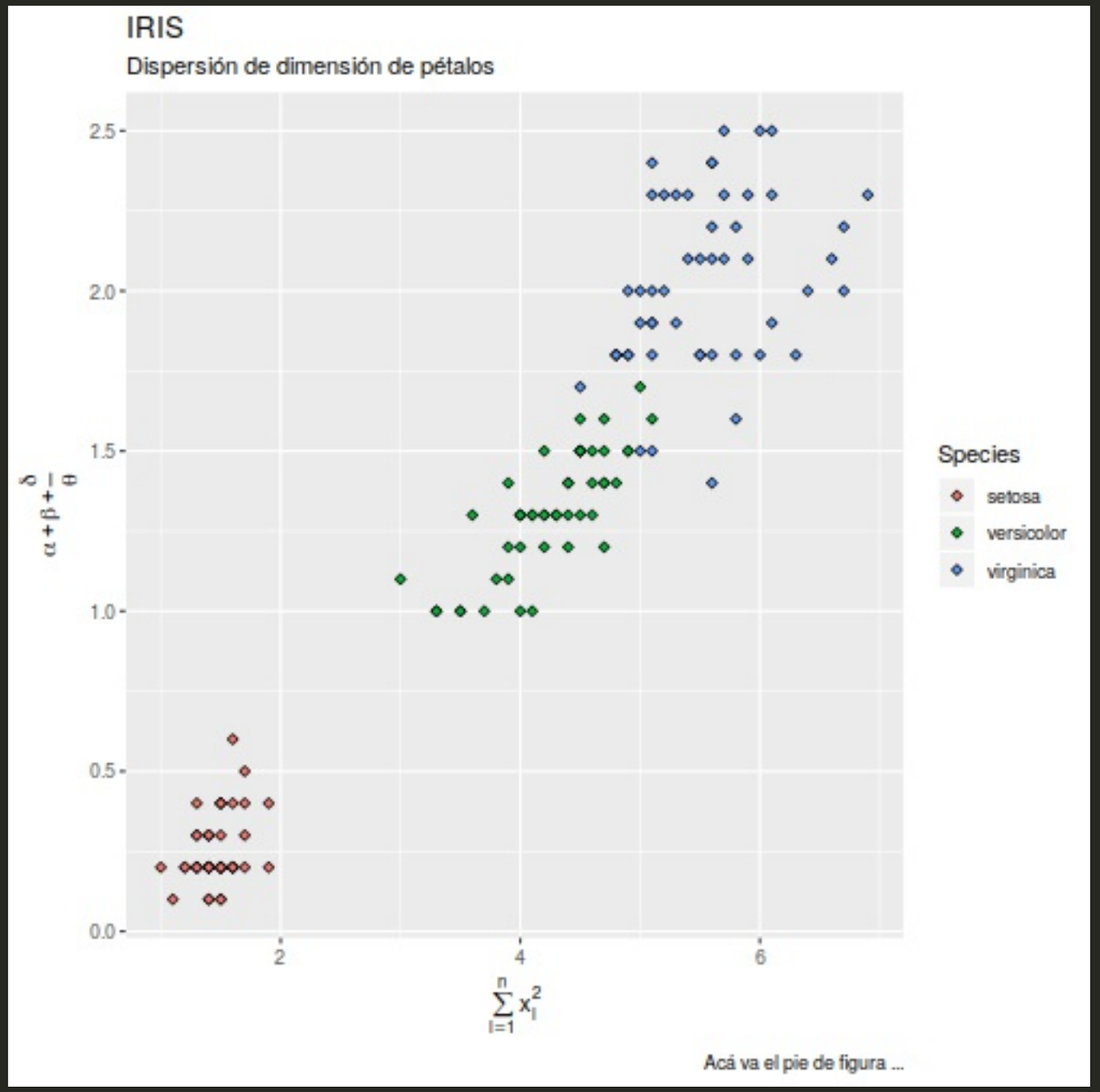
```
                iris %>%
ggplot(mapping = aes(x = Petal.Length,
                     y = Petal.Width,
                       fill = Species
                            )
                     ) +
          geom_point(shape=23) +
                labs(
             title = "IRIS",
subtitle = "Dispersión de dimensión de pétalos",
     caption = "Acá va el pie de figura ..."
                )
```


IRIS
Dispersión de dimensión de pétalos
Petal.Width
2.5
2.0
1.5
1.0
0.5
0.0
2
4
6
Petal.Length
Species
setosa
versicolor
virginica
Acá va el pie de figura ...

¿Y si tenemos que poner símbolos matemáticos en los ejes?

```
                labs(
    x = quote(Petal.Length=sum(x[i] ^ 2, i == 1, n)),
y = quote(Petal.Widht=alpha + beta + frac(delta, theta))
                )
```

```
                    iris %>%
  ggplot(mapping = aes(x = Petal.Length,
                       y = Petal.Width,
                         fill = Species
                                )
                       ) +
            geom_point(shape=23) +
                    labs(
                  title = "IRIS",
  subtitle = "Dispersión de dimensión de pétalos",
      caption = "Acá va el pie de figura ...",
        x = quote(sum(x[i] ^ 2, i == 1, n)),
    y = quote(alpha + beta + frac(delta, theta))
                    )
```


IRIS
Dispersión de dimensión de pétalos
2.5
2.0
1.5
1.0
0.5
0.0
2
4
6
$\sum_{i=1}^{n} x_i^2$
$\alpha + \beta + \gamma + \delta + \theta$
Species
setosa
versicolor
virginica
Acá va el pie de figura ...

Pensemos en un...

-> ~~Título y subtítulo...~~

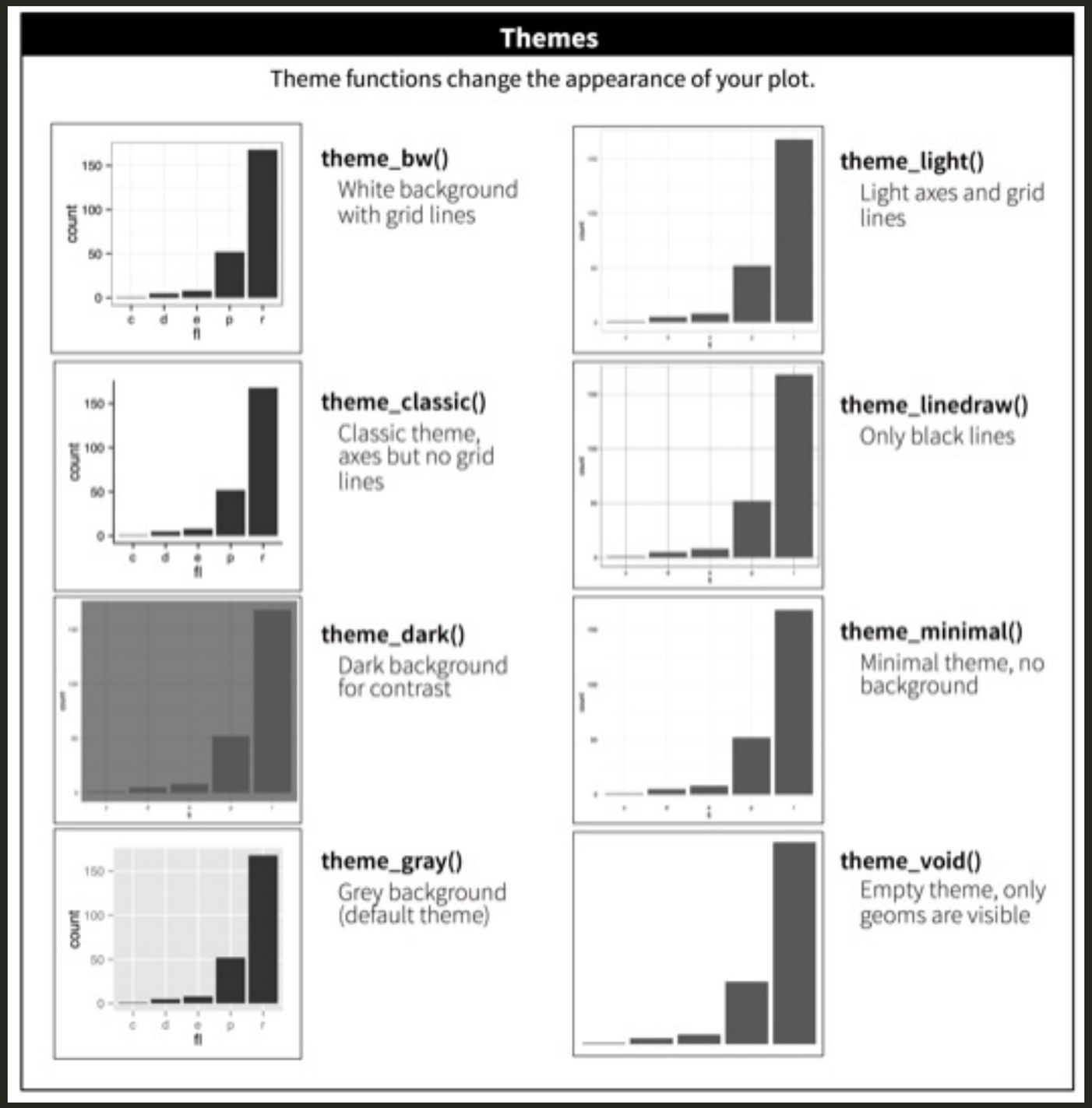
-> Leyenda abajo...

-> Quitar el fondo gris...

-> Cambiar la paleta de colores...

-> Separar en paneles por Specie...

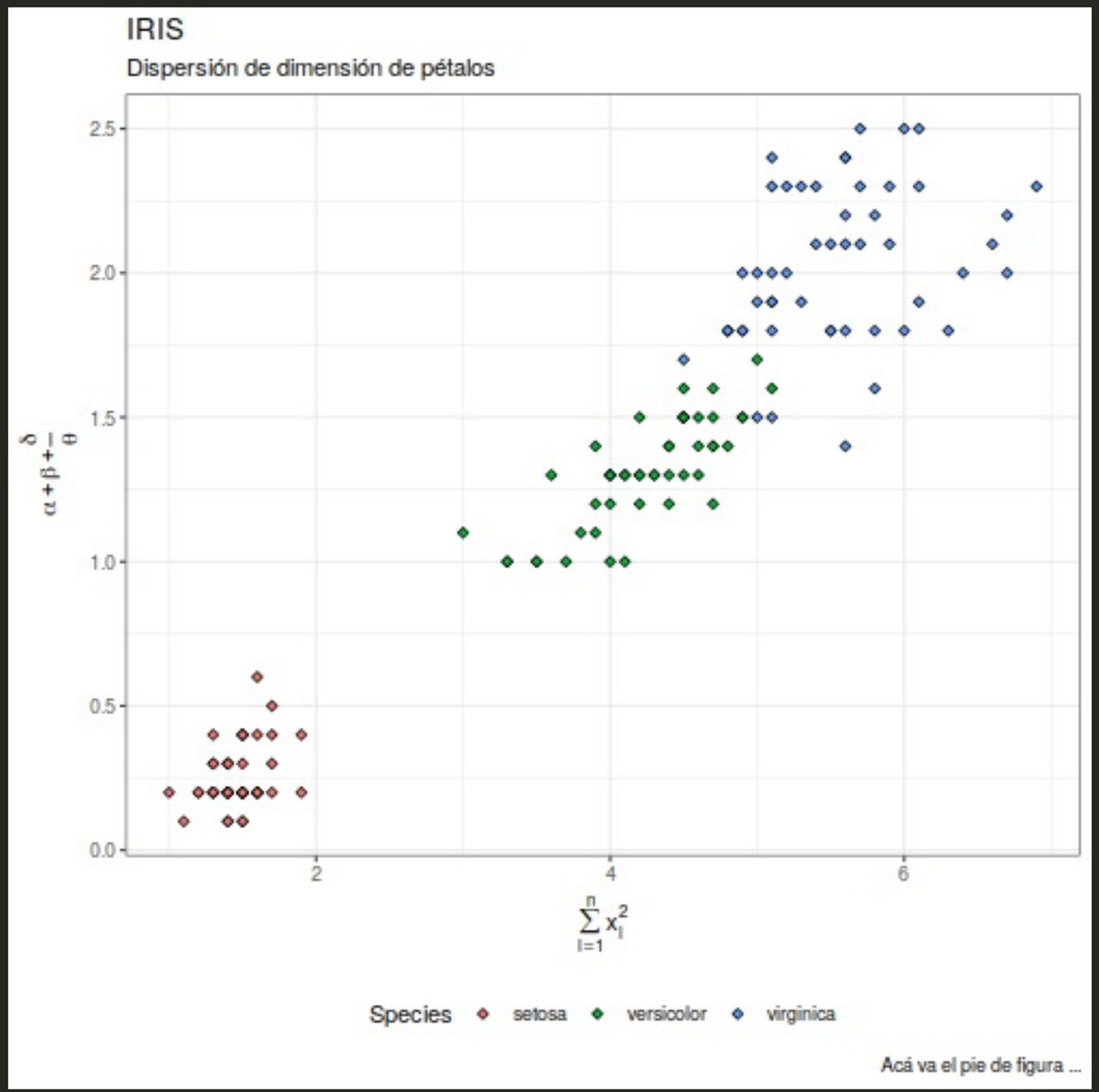
ggplot2: THEMES



ggplot2: THEMES - Legend

```
+ theme(legend.position = NULL) # remove the legend
        + theme(legend.position = "left")
         + theme(legend.position = "top")
       + theme(legend.position = "bottom")
 + theme(legend.position = "right") # the default
```

```
                iris %>%
ggplot(mapping = aes(x = Petal.Length,
                     y = Petal.Width,
                       fill = Species
                              )
                     ) +
          geom_point(shape=23) +
                labs(
              title = "IRIS",
subtitle = "Dispersión de dimensión de pétalos",
    caption = "Acá va el pie de figura ...",
      x = quote(sum(x[i] ^ 2, i == 1, n)),
  y = quote(alpha + beta + frac(delta, theta))
                     ) +
               theme_bw() +
          theme(legend.position = "bottom")
```


IRIS
Dispersión de dimensión de pétalos
2.5
2.0
1.5
1.0
0.5
0.0
2
4
6
$\alpha + \beta + \frac{\delta}{\theta}$
$\sum_{i=1}^{n} x_i^2$
Species
setosa
versicolor
virginica
Acá va el pie de figura ...

Pensemos en un...

-> ~~Título y subtítulo...~~

-> ~~Leyenda abajo...~~

-> ~~Quitar el fondo gris...~~

-> Cambiar la paleta de colores...

-> Separar en paneles por Specie...

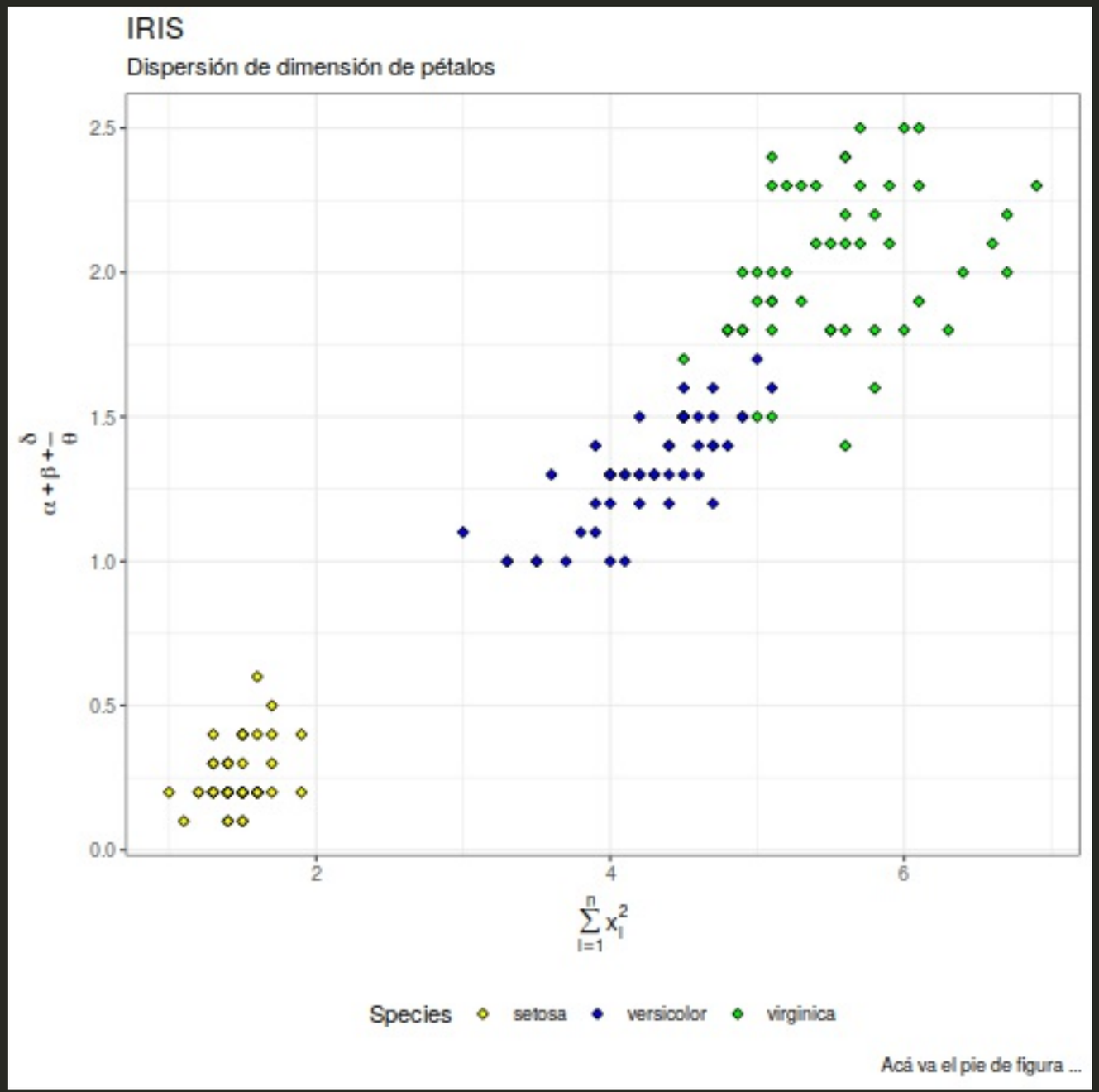
ggplot2: SCALES

Tenemos de todos los sabores!!!

```
scale_x/y/colour/fill_manual/continuos/discrete/log10
                scale_x_continuous()
                 scale_fill_manual()
                  scale_x_log10()
scale_x_continuous(limits = range(iris$Petal.Length))
    scale_y_continuous(limits = c(min=0, max=10))
        scale_colour_brewer(palette = "Set1")
          scale_fill_distiller("spectral")
```

ggplot2: SCALES colors

```
                    iris %>%
      ggplot(mapping = aes(x = Petal.Length,
                           y = Petal.Width,
                        fill = as.factor(Species)
                                         )
                           ) +
                 geom_point(shape=23) +
                          labs(
                       title = "IRIS",
    subtitle = "Dispersión de dimensión de pétalos",
       caption = "Acá va el pie de figura ...",
         x = quote(sum(x[i] ^ 2, i == 1, n)),
      y = quote(alpha + beta + frac(delta, theta))
                           ) +
                       theme_bw() +
        theme(legend.position = "bottom") +
                       scale_fill_manual(
      values = c("yellow", "blue", "green"),
                         name = "Species"
                         )
```


IRIS
Dispersión de dimensión de pétalos
2.5
2.0
1.5
1.0
0.5
0.0
2
4
6
$\frac{\delta}{\alpha+\beta+\theta}$
$\sum_{i=1}^{n} x_i^2$
Species
setosa
versicolor
virginica
Acá va el pie de figura ...

Pensemos en un...

-> ~~Título y subtítulo...~~

-> ~~Leyenda abajo...~~

-> ~~Quitar el fondo gris...~~

-> ~~Cambiar la paleta de colores...~~

-> Separar en paneles por Specie...

ggplot2: Facets

Facets

Define como los datos pueden ser agrupados en grillas (filas y/o columnas)

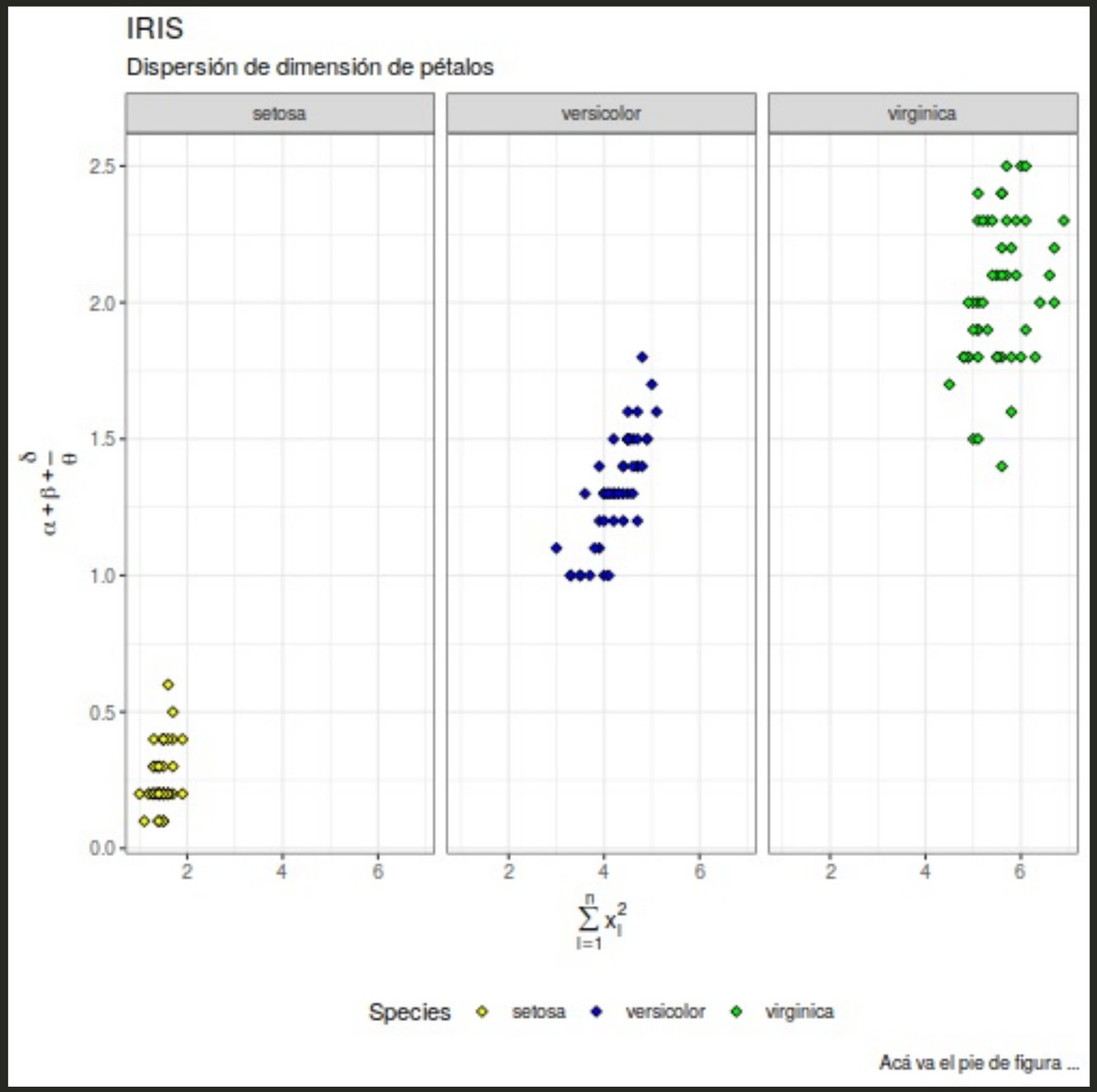
ggplot2: Facets: wrap vs grid

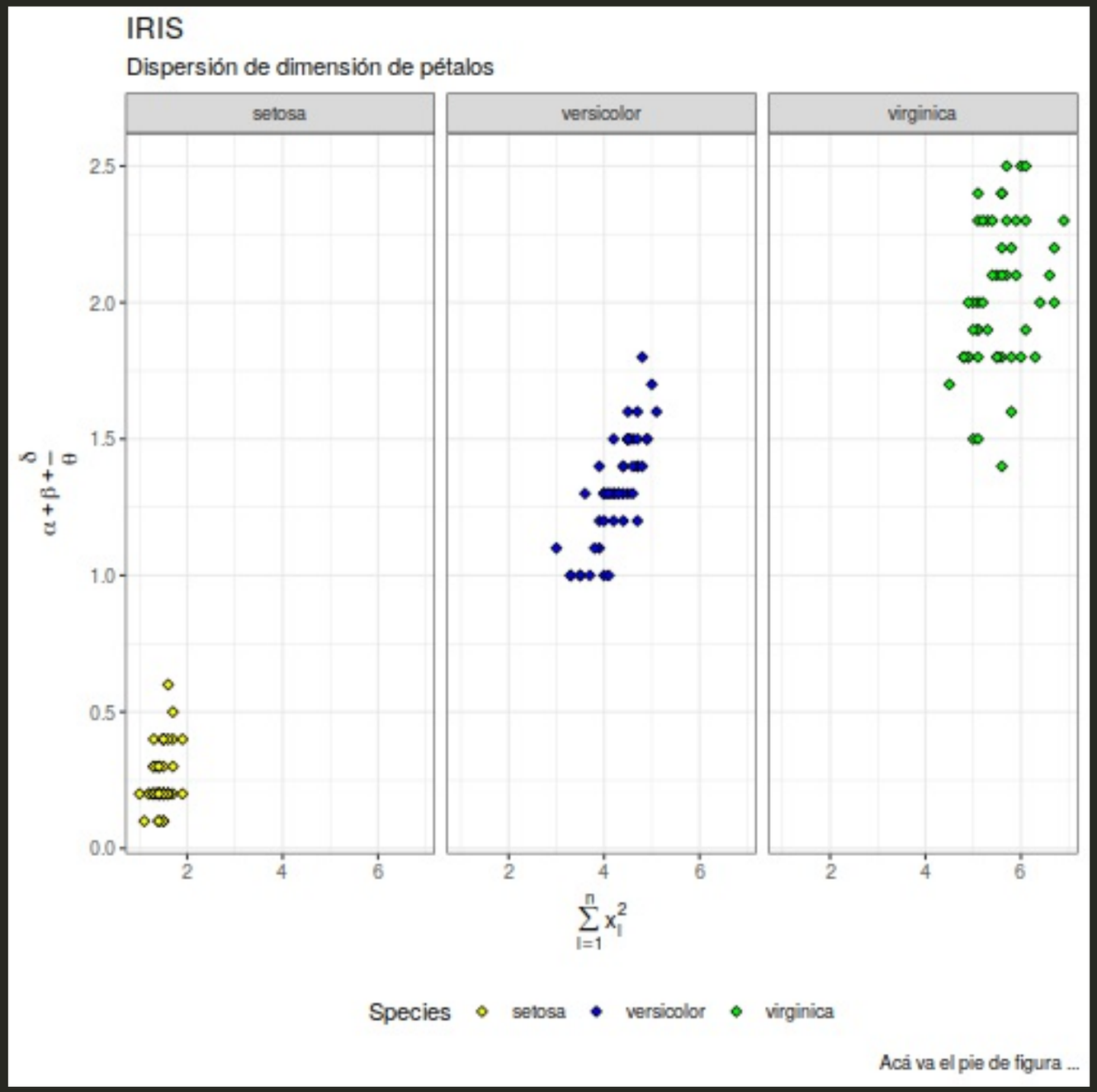
Tenemos de dos sabores

-> facet_wrap: Para una variable.

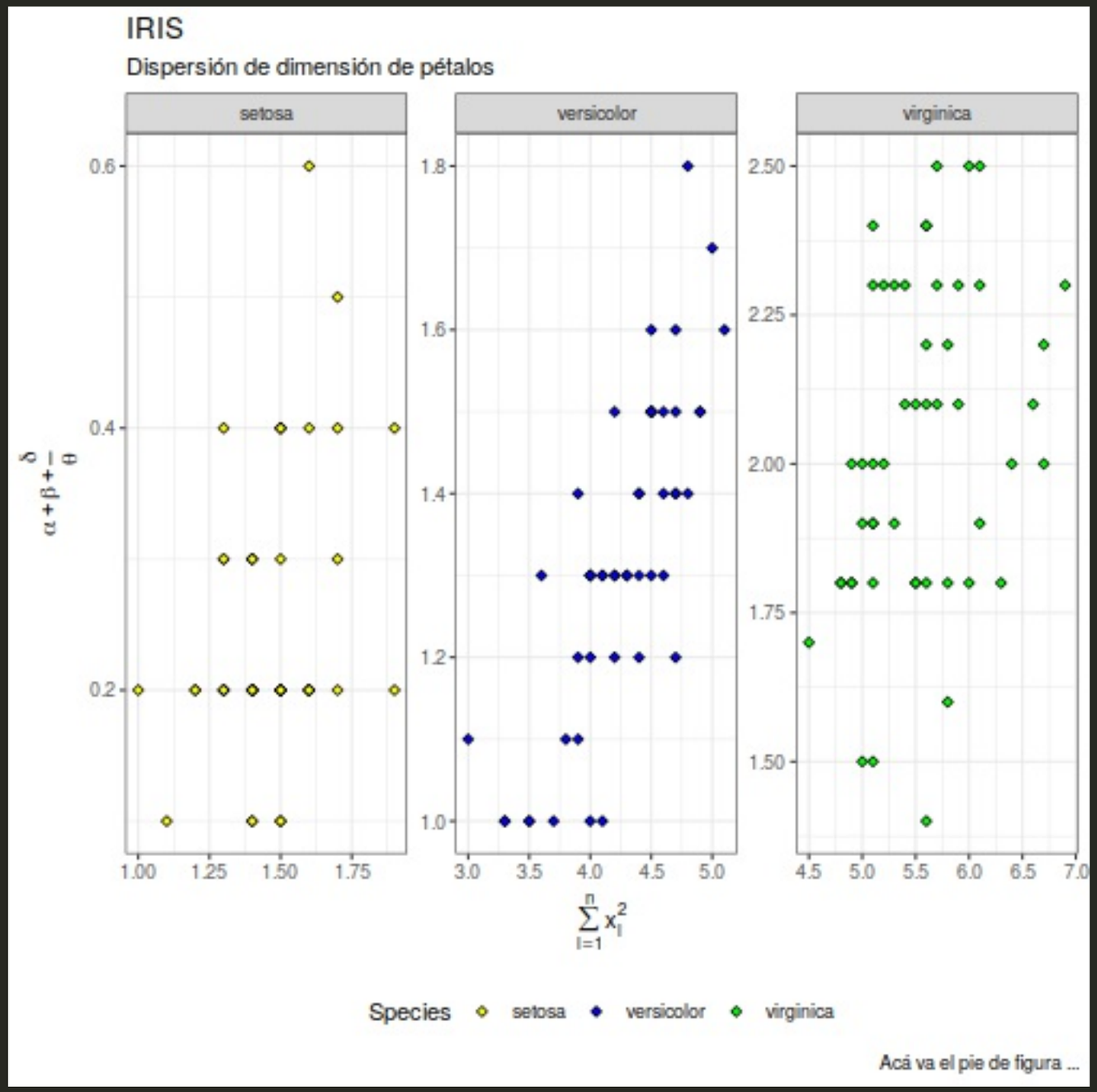
-> facet_grid: Una matriz de 2 variables.

```
                    iris %>%
          ggplot(mapping = aes(x = Petal.Length,
                               y = Petal.Width,
                            fill = as.factor(Species)
                                 )
                               ) +
                      geom_point(shape=23) +
                             labs(
                           title = "IRIS",
        subtitle = "Dispersión de dimensión de pétalos",
            caption = "Acá va el pie de figura ...",
              x = quote(sum(x[i] ^ 2, i == 1, n)),
          y = quote(alpha + beta + frac(delta, theta))
                               ) +
                          theme_bw() +
                 theme(legend.position = "bottom") +
                          scale_fill_manual(
                values = c("yellow", "blue", "green"),
                            name = "Species"
                               ) +
                        facet_wrap(~Species)
```


IRIS
Dispersión de dimensión de pétalos
setosa
versicolor
virginica
2.5
2.0
1.5
1.0
0.5
0.0
2
4
6
$\alpha + \beta + \delta$
$\sum_{i=1}^{n} x_i^2$
Species
setosa
versicolor
virginica
Acá va el pie de figura ...


IRIS
Dispersión de dimensión de pétalos
setosa
versicolor
virginica
2.5
2.0
1.5
1.0
0.5
0.0
2
4
6
$\alpha + \beta + \delta$
$\sum_{i=1}^{n} x_i^2$
Species
setosa
versicolor
virginica
Acá va el pie de figura ...

```
                    iris %>%
          ggplot(mapping = aes(x = Petal.Length,
                               y = Petal.Width,
                            fill = as.factor(Species)
                                 )
                               ) +
                      geom_point(shape=23) +
                             labs(
                          title = "IRIS",
       subtitle = "Dispersión de dimensión de pétalos",
           caption = "Acá va el pie de figura ...",
             x = quote(sum(x[i] ^ 2, i == 1, n)),
          y = quote(alpha + beta + frac(delta, theta))
                             ) +
                           theme_bw() +
                theme(legend.position = "bottom") +
                           scale_fill_manual(
            values = c("yellow", "blue", "green"),
                             name = "Species"
                             ) +
           facet_wrap(~Species, scales = "free")
```


IRIS
Dispersión de dimensión de pétalos
setosa
versicolor
virginica
0.6
0.4
0.2
1.8
1.6
1.4
1.2
1.0
2.50
2.25
2.00
1.75
1.50
1.00
1.25
1.50
1.75
3.0
3.5
4.0
4.5
5.0
4.5
5.0
5.5
6.0
6.5
7.0
$\alpha + \beta + \frac{\delta}{\theta}$
$\sum_{i=1}^{n} x_i^2$
Species
setosa
versicolor
virginica
Acá va el pie de figura ...

Pensemos en un...

-> ~~Título y subtítulo...~~

-> ~~Leyenda abajo...~~

-> ~~Quitar el fondo gris...~~

-> ~~Cambiar la paleta de colores...~~

-> ~~Separar en paneles por Specie...~~

Ejercicios:

Utilizando MTCARS retome la gráfica de dispersión de la clase pasada donde logró agrupar por cantidad de cilindros.

- Agregue título a la gráfica.

- Utilice el theme de blanco y negro

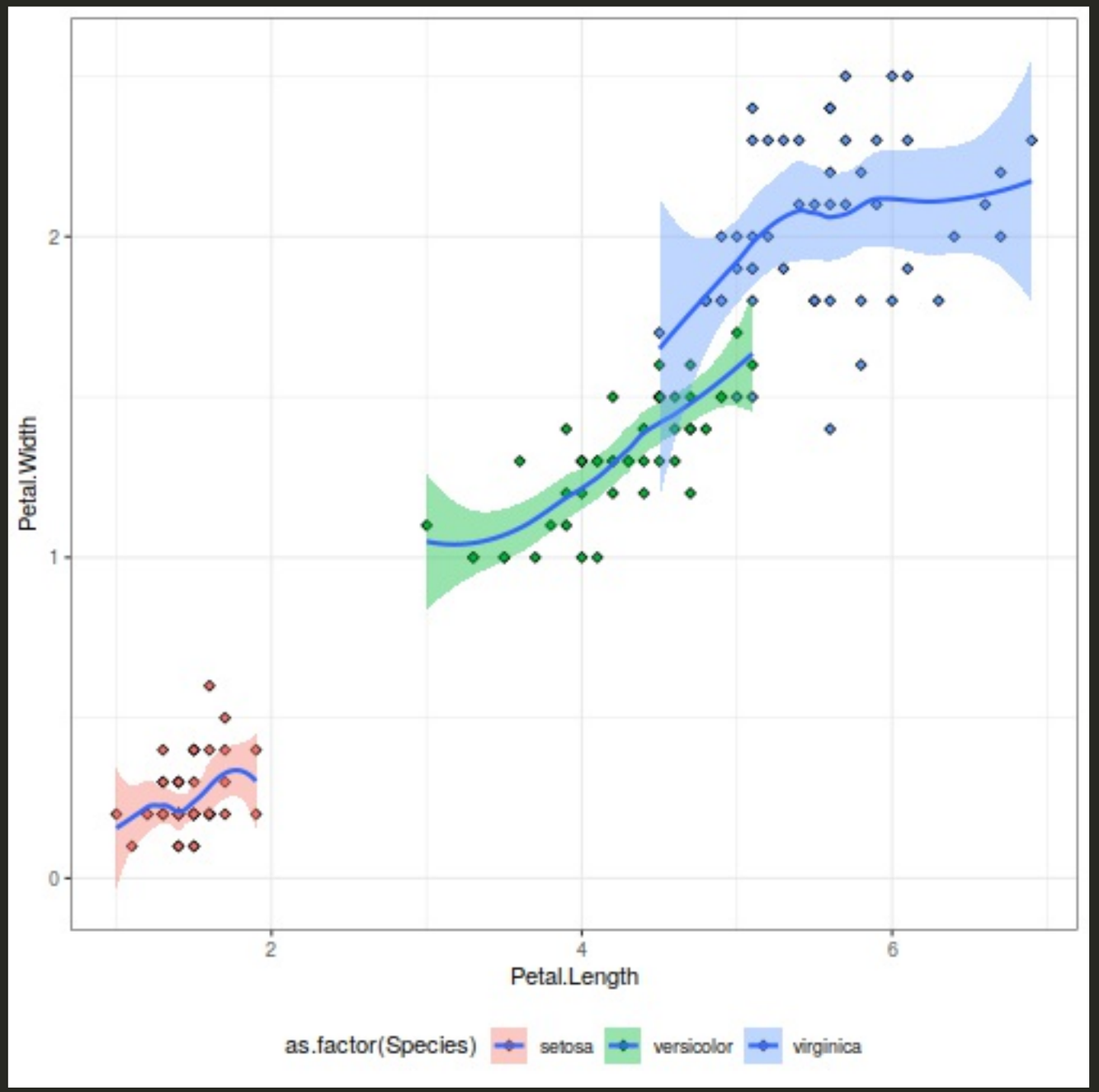
- Genere los paneles correspondinete a la cantidad de cilindros.

BONUS: geom_smooth

```
                    iris %>%
  ggplot(mapping = aes(x = Petal.Length,
                       y = Petal.Width,
                      fill = as.factor(Species)
                                       )
                      ) +
          geom_point(shape=23) +
               theme_bw() +
  theme(legend.position = "bottom") +
               geom_smooth()
```

BONUS: geom_smooth

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



Trabajemos con boxplots de ggplot2

geom_boxplot

-> Ojo con el formato de los datos

-> geom_boxplot los requiere en formato largo

```
iris %>% head
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

```
iris %>%
  tidyr::pivot_longer(cols = -Species, names_to = "variables", values_to = "
  head
```

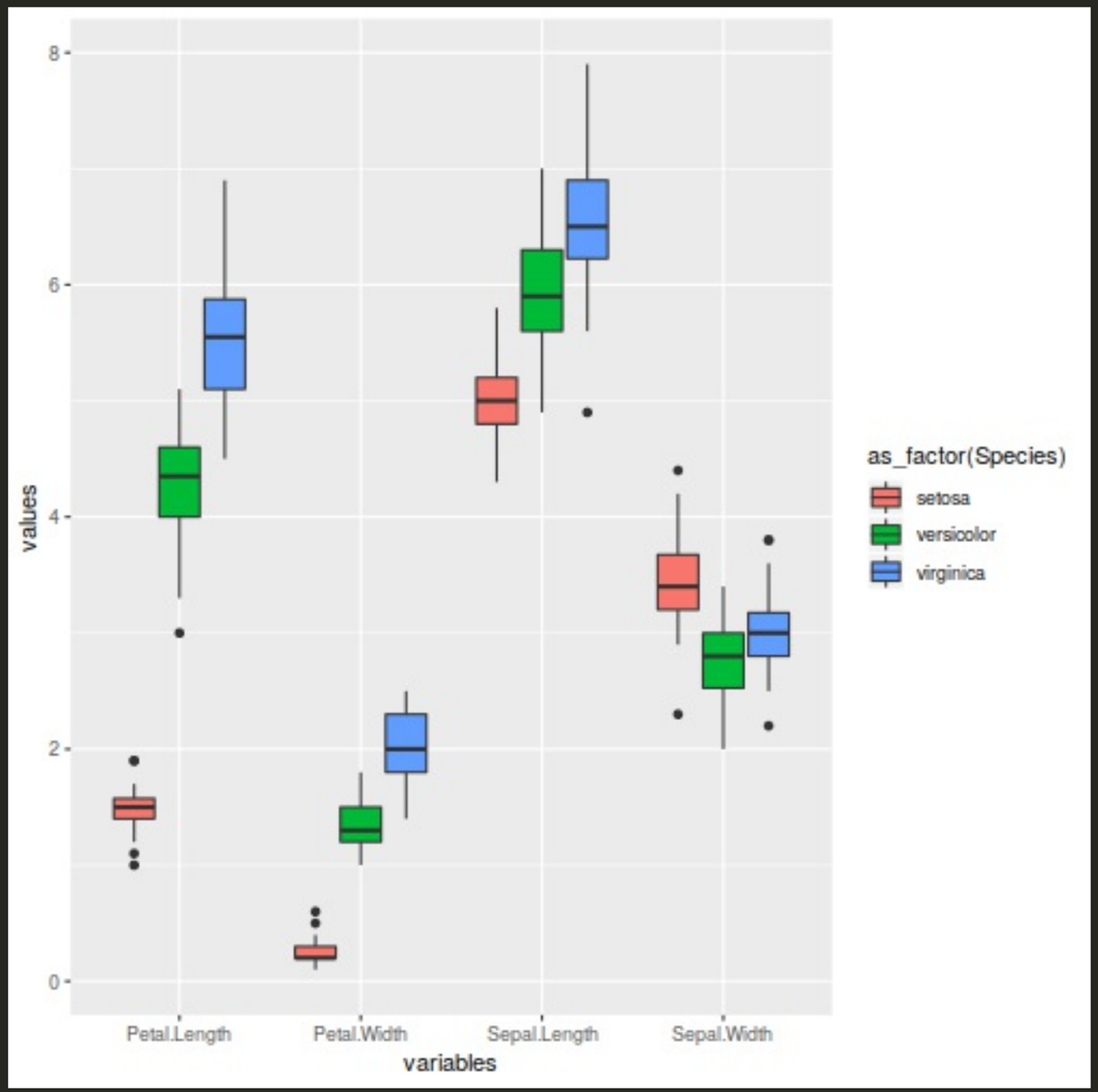
```
## # A tibble: 6 x 3
##   Species variables    values
##   <fct>   <chr>         <dbl>
## 1 setosa  Sepal.Length    5.1
## 2 setosa  Sepal.Width     3.5
## 3 setosa  Petal.Length    1.4
## 4 setosa  Petal.Width     0.2
## 5 setosa  Sepal.Length    4.9
## 6 setosa  Sepal.Width     3
```

geom_boxplot

Ahora podemos hacer el boxplot

```
                iris %>%
             tidyr::pivot_longer(
                cols = -Species,
             names_to = "variables",
           values_to = "values") %>%
    ggplot(mapping = aes(x = variables,
                         y = values,
                  fill = as_factor(Species))) +
               geom_boxplot()
```

geom_boxplot



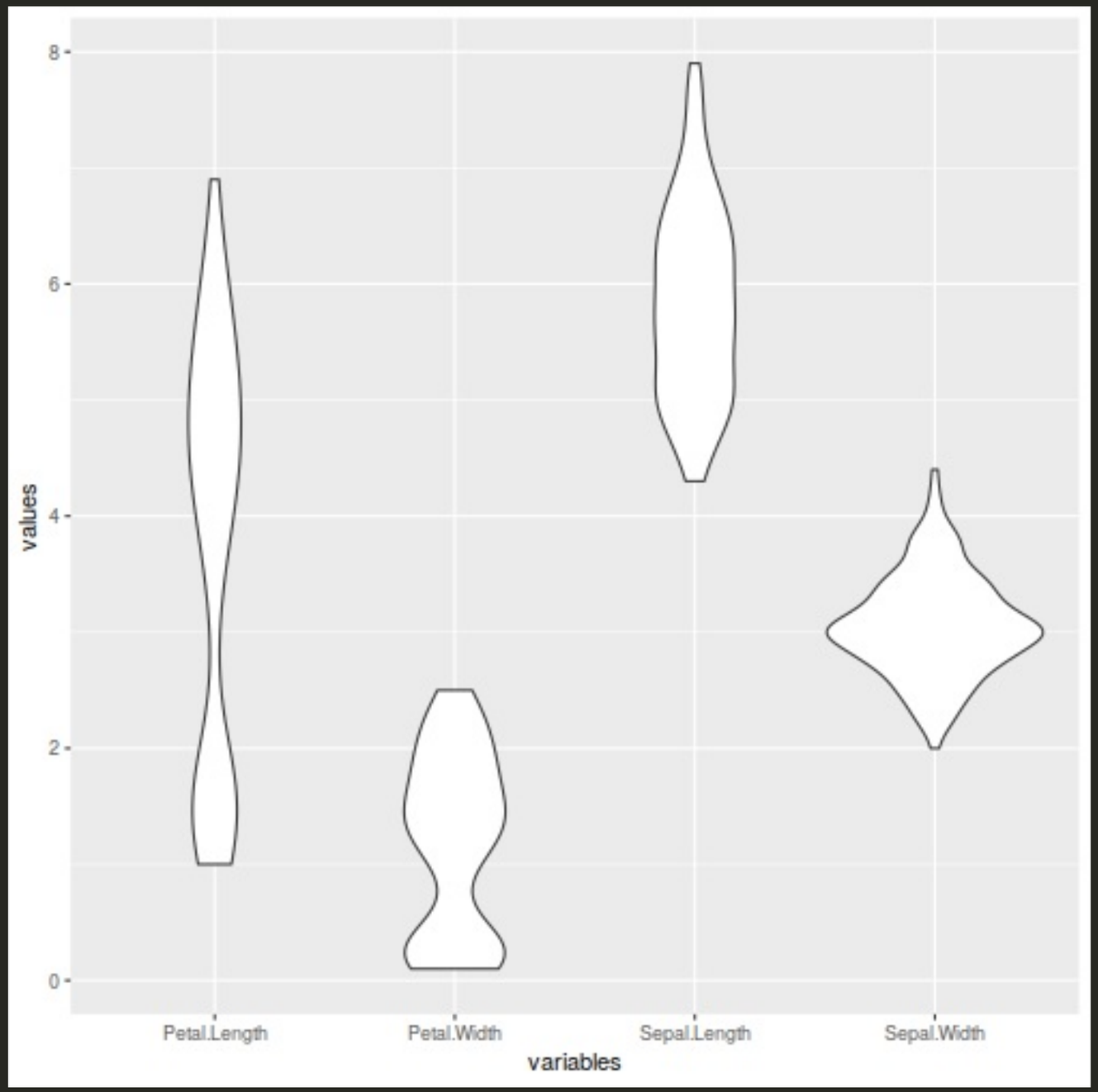
Ejercicio

- Hacer un ggplot's boxplot con MTCARS para las variables "disp" y "hp"

Sugerencia:

- Cambie la representación de los datos
- Utilice verbos para seleccionar o filtrar, dependiendo de su razonamiento.

geom_violin


8
6
4
2
0
values
Petal.Length
Petal.Width
Sepal.Length
Sepal.Width
variables

geom_density

Pensemos en datos simulados

```
datos <- tibble(
valores = c(rnorm(100), rnorm(n = 100, mean = 10)))
datos %>%
ggplot(aes(x = valores))+
geom_density()
```

geom_density

Pensemos en datos simulados

EJERCICIO

Genere el boxplot de los datos anteriores

Genere el violín de los datos anteriores

¿Qué observa en ambos casos? ¿Es correcto?

ggplot2: Statistical transformations

1. **`geom_bar()`** begins with the **`diamonds`** data set

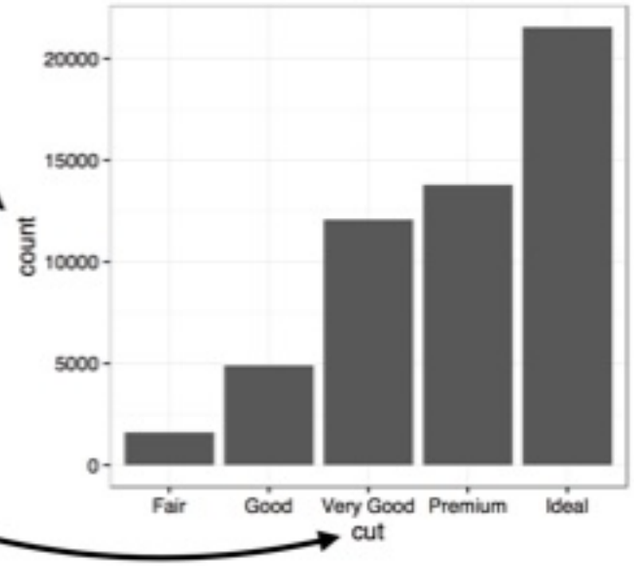
2. **`geom_bar()`** transforms the data with the "count" stat, which returns a data set of cut values and counts.

3. **`geom_bar()`** uses the transformed data to build the plot. cut is mapped to the x axis, count is mapped to the y axis.

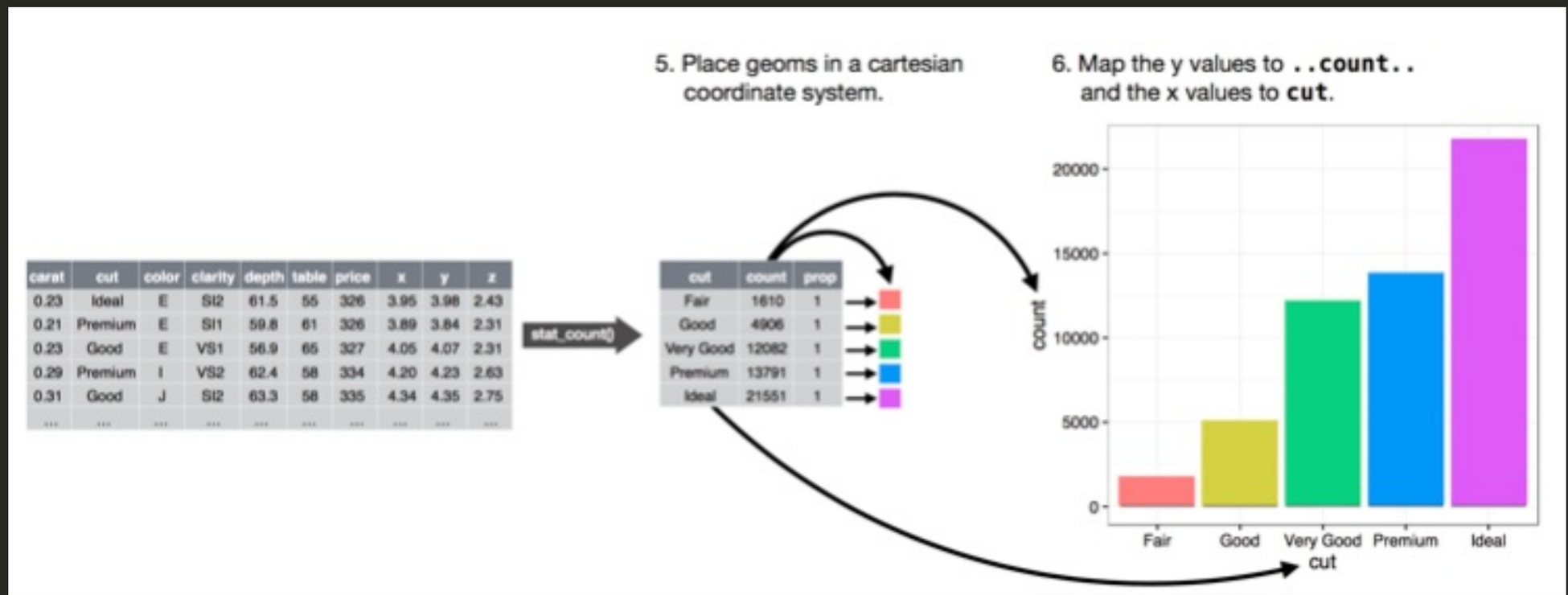
carat	cut	color	clarity	depth	table	price	x	y	z
0.23	Ideal	E	SI2	61.5	55	326	3.95	3.98	2.43
0.21	Premium	E	SI1	59.8	61	326	3.89	3.84	2.31
0.23	Good	E	VS1	56.9	65	327	4.05	4.07	2.31
0.29	Premium	I	VS2	62.4	58	334	4.20	4.23	2.63
0.31	Good	J	SI2	63.3	58	335	4.34	4.35	2.75
...	...	...	...	...	...	...	...	...	...

stat_count()

cut	count	prop
Fair	1610	1
Good	4906	1
Very Good	12082	1
Premium	13791	1
Ideal	21551	1



ggplot2: Statistical transformations



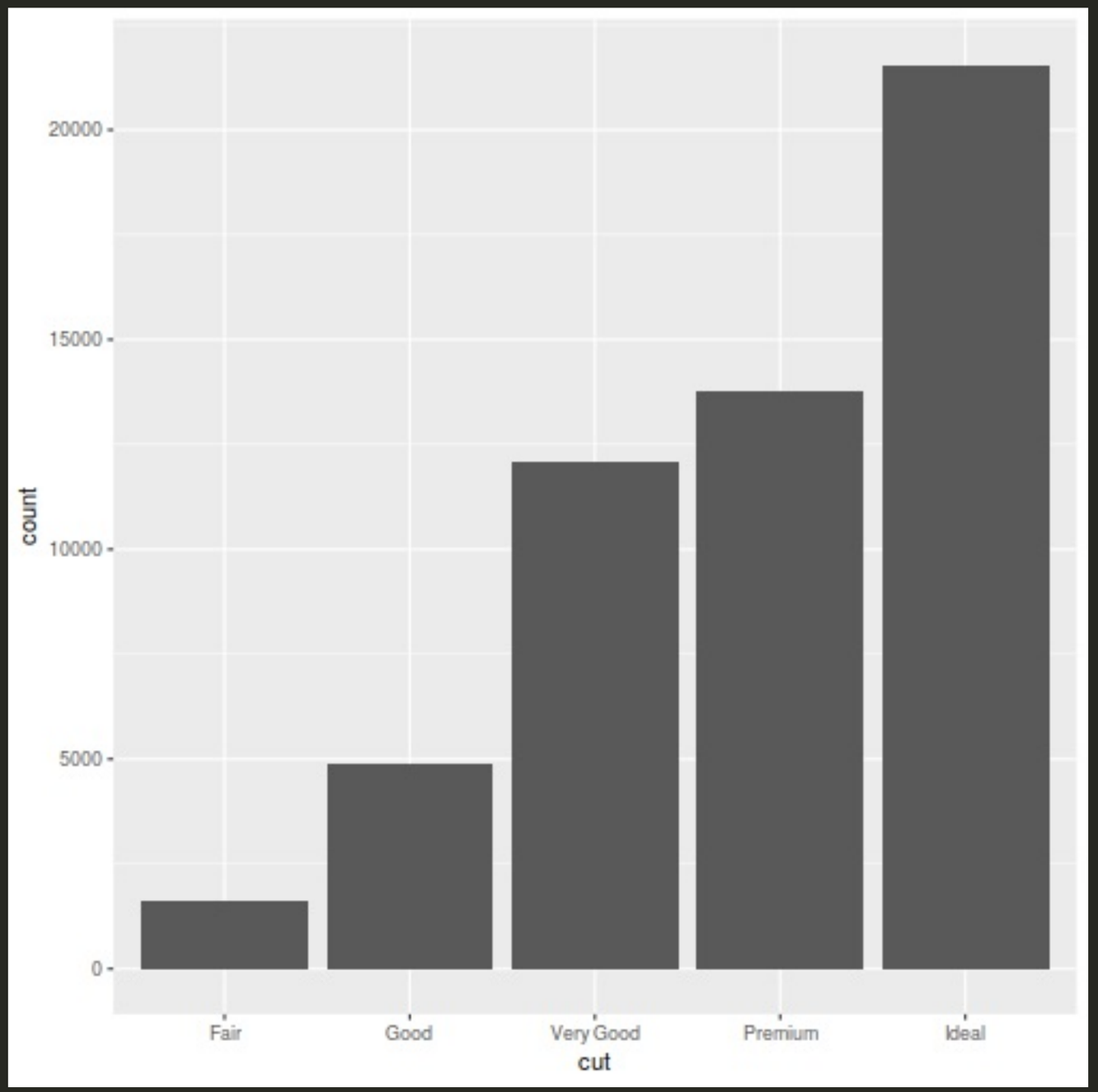
ggplot2: geom_bar

Trabajemos con diamods

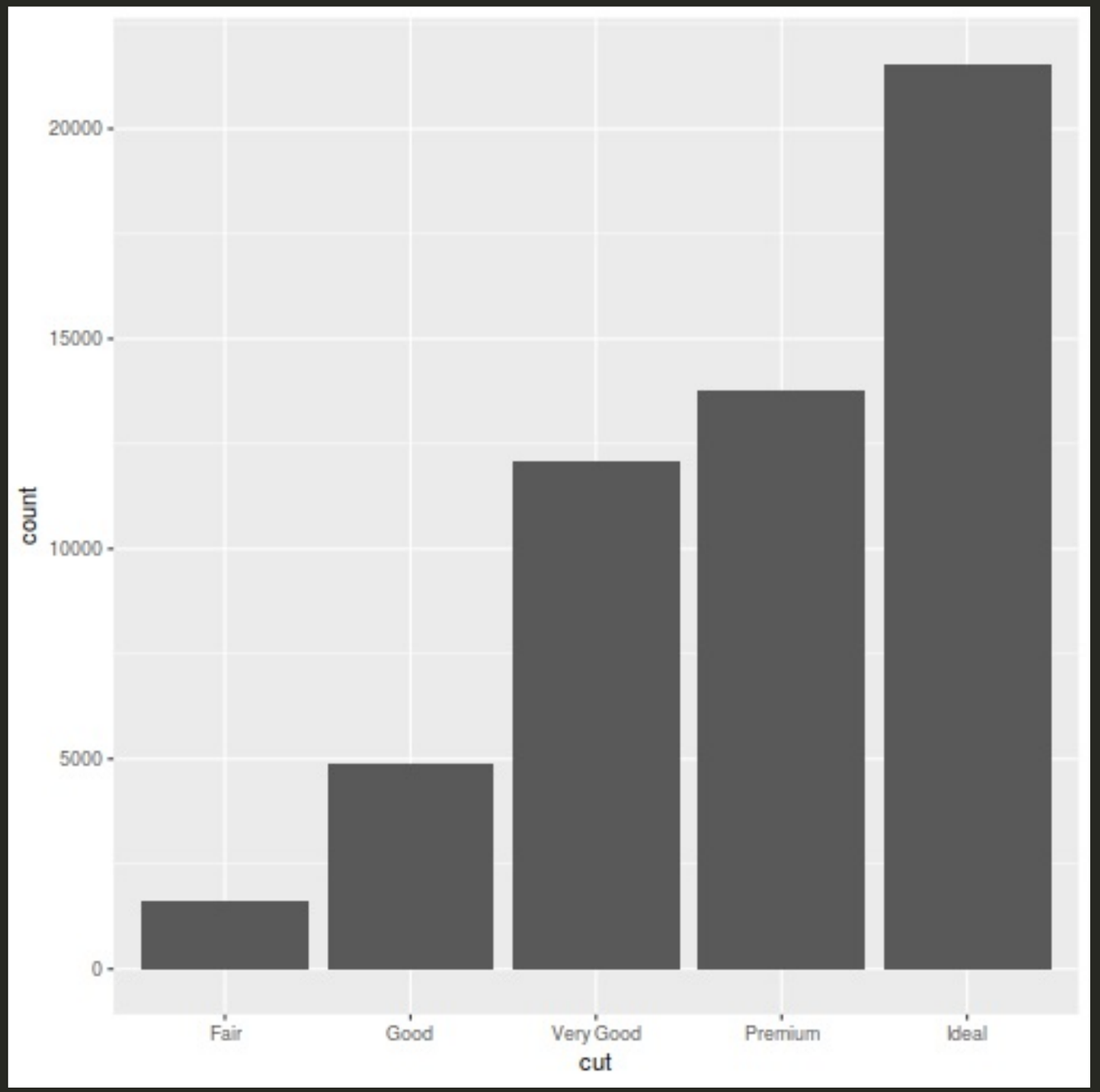
```
diamonds %>% head
```

```
## # A tibble: 6 x 10
##   carat cut       color clarity depth table price     x     y     z
##   <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
## 1 0.23  Ideal     E     SI2      61.5    55   326  3.95  3.98  2.43
## 2 0.21  Premium   E     SI1      59.8    61   326  3.89  3.84  2.31
## 3 0.23  Good      E     VS1      56.9    65   327  4.05  4.07  2.31
## 4 0.290 Premium   I     VS2      62.4    58   334  4.2   4.23  2.63
## 5 0.31  Good      J     SI2      63.3    58   335  4.34  4.35  2.75
## 6 0.24  Very Good J     VVS2     62.8    57   336  3.94  3.96  2.48
```

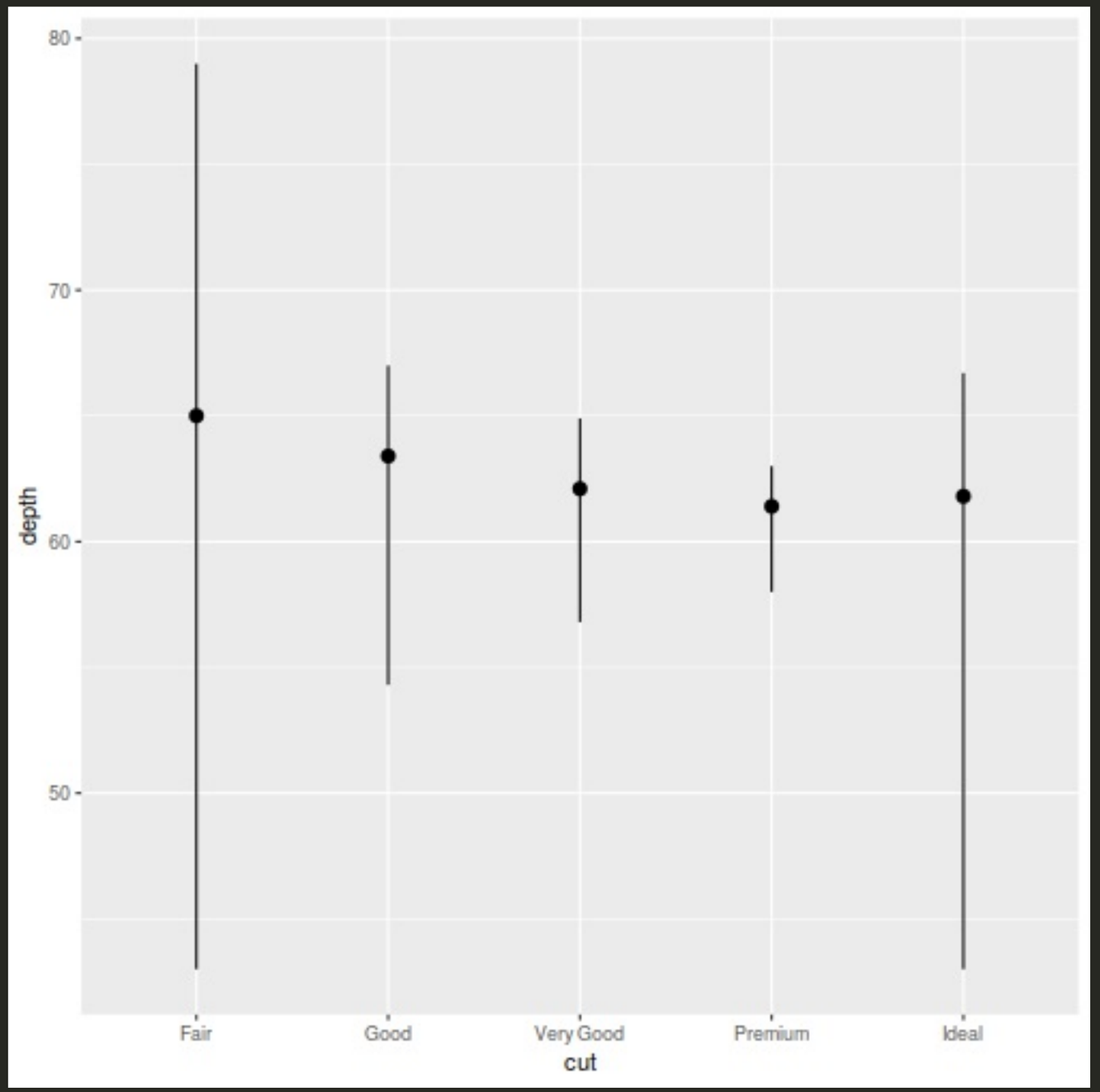
```
diamonds %>%
 ggplot() +
  geom_bar(
mapping = aes(x = cut)
)
```



```
ggplot(data = diamonds) +
stat_count(mapping = aes(x = cut))
```



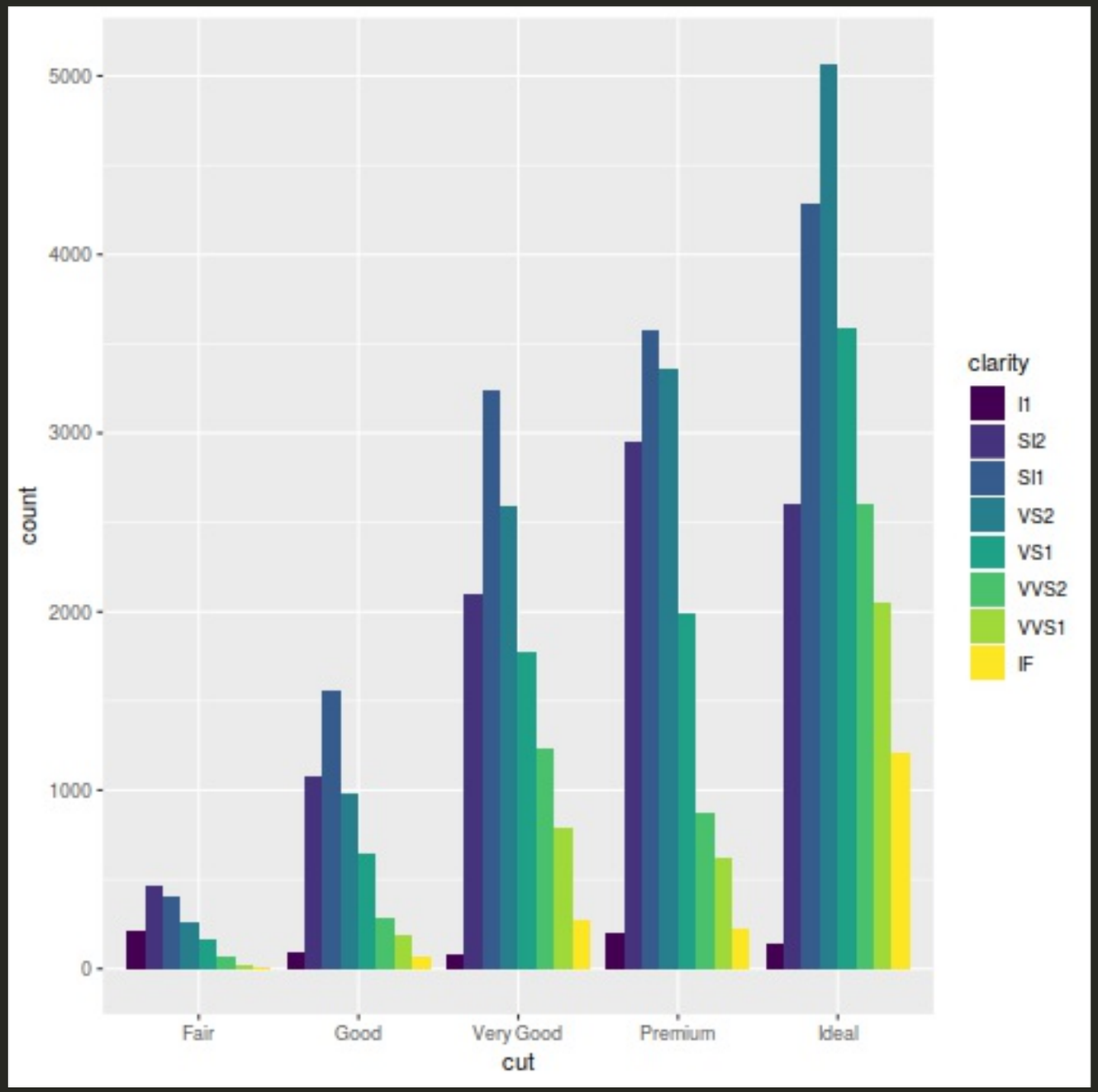
```
ggplot(data = diamonds) +
stat_summary(mapping = aes(x = cut, y = depth),
  fun.ymin = min,fun.ymax = max,fun.y = median)
```



ggplot2: position

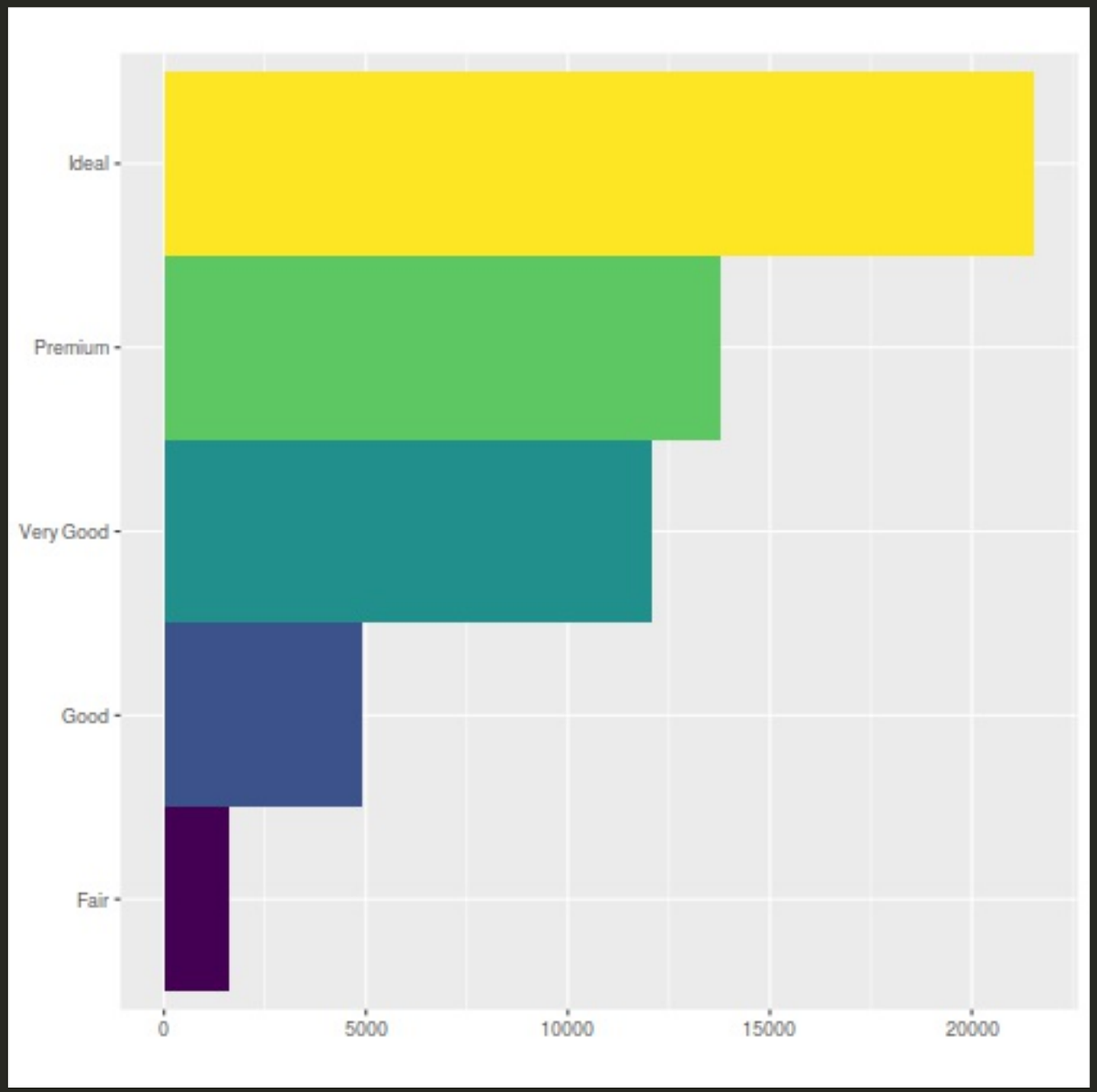
```
ggplot(data = diamonds) +
  geom_bar(
mapping = aes(x = cut, fill = clarity),
    position = "dodge"
  )
```

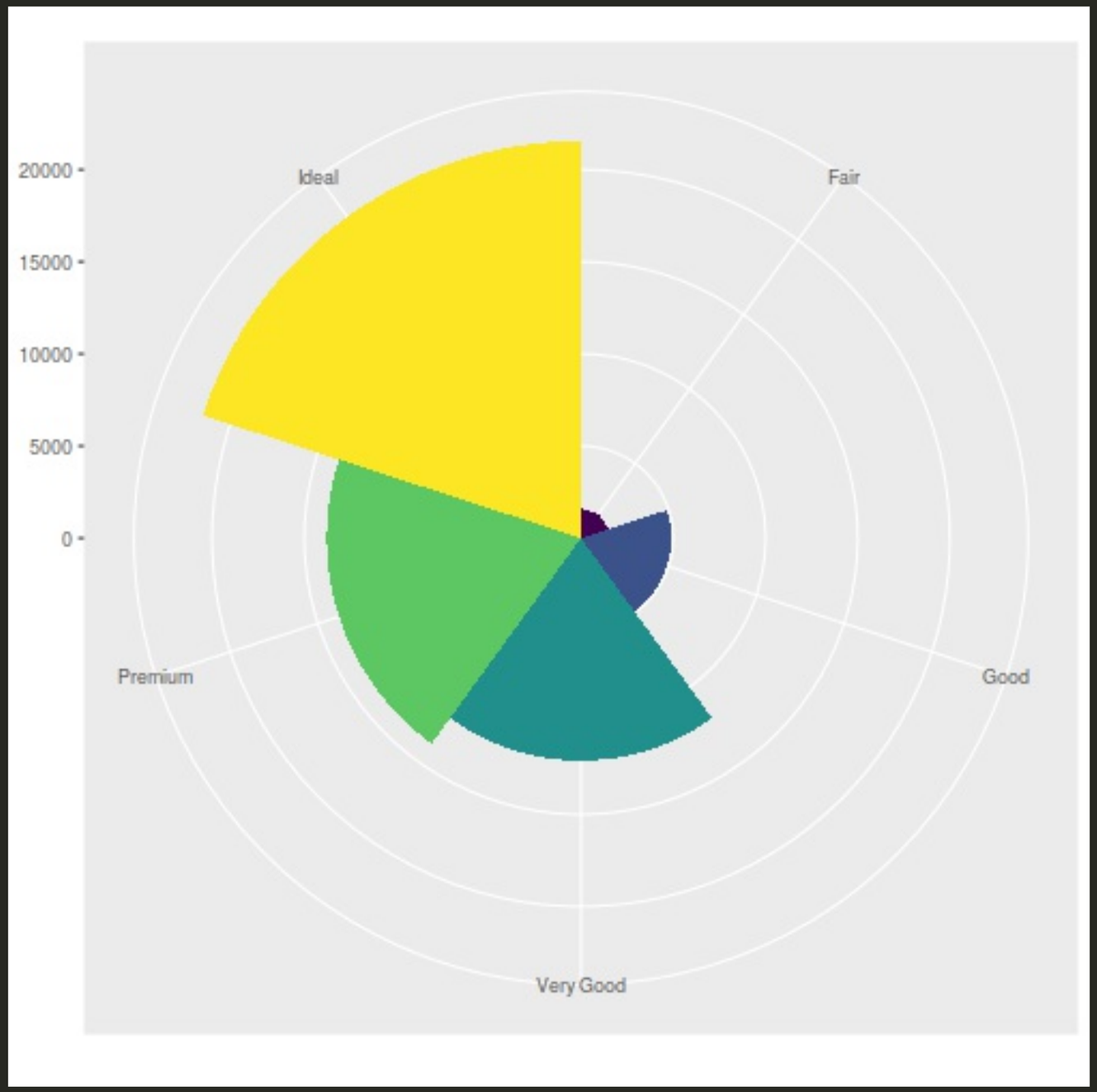
ggplot2: position



coor_flip vs coord_polar

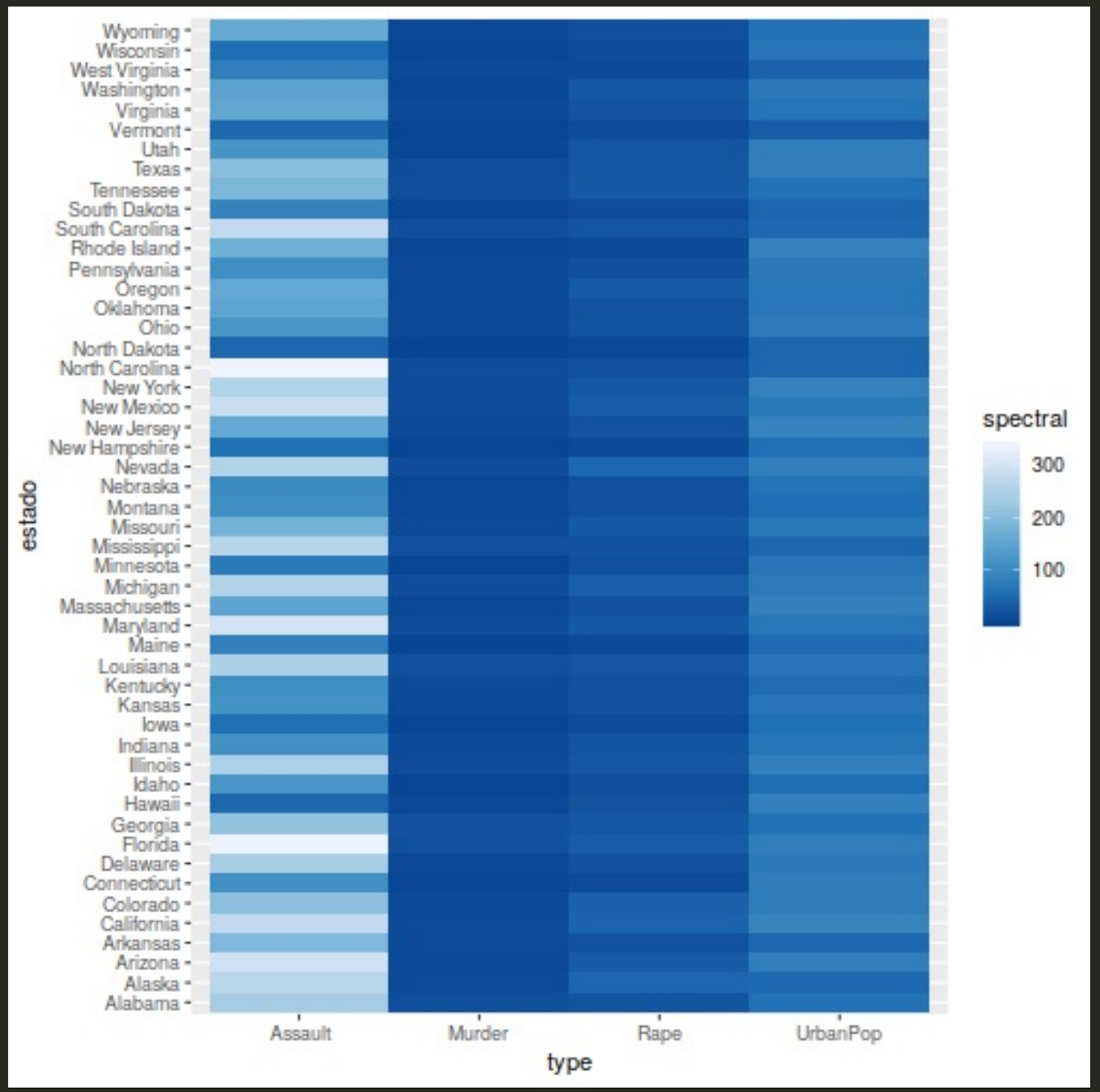
```
bar <- ggplot(data = diamonds) +
                geom_bar(
 mapping = aes(x = cut, fill = cut),
          show.legend = FALSE,
                width = 1
                ) +
       theme(aspect.ratio = 1) +
       labs(x = NULL, y = NULL)

         bar + coord_flip()
         bar + coord_polar()
```


Ideal
Premium
Very Good
Good
Fair
0
5000
10000
15000
20000


20000
15000
10000
5000
0
Ideal
Fair
Premium
Good
Very Good

geom_tile

```
USArrests %>%
  rownames_to_column(var = "estado") %>%
  pivot_longer(cols = -estado, names_to = "type", values_to = "value", value
  ggplot(mapping = aes(x = type, y = estado, fill = value)) +
  geom_tile() +
  scale_fill_distiller("spectral")
```


Wyoming
Wisconsin
West Virginia
Washington
Virginia
Vermont
Utah
Texas
Tennessee
South Dakota
South Carolina
Rhode Island
Pennsylvania
Oregon
Oklahoma
Ohio
North Dakota
North Carolina
New York
New Mexico
New Jersey
New Hampshire
Nevada
Nebraska
Montana
Missouri
Mississippi
Minnesota
Michigan
Massachusetts
Maryland
Maine
Louisiana
Kentucky
Kansas
Iowa
Indiana
Illinois
Idaho
Hawaii
Georgia
Florida
Delaware
Connecticut
Colorado
California
Arkansas
Arizona
Alaska
Alabama
estado
Assault
Murder
Rape
UrbanPop
type
spectral
300
200
100

Ejercicio:

-> Cambie la tonalidad para que la barra de color quede en verde, amarillo y rojo.

ggplot2:Saving

plots tradicional

-> ~~NO se pueden GUARDAR~~

-> ~~NO se pueden Cargar~~

ggplot2:Saving

plots de ggplot son Objetos

-> Se pueden almacenar .RData, RDS, etc.

-> Se pueden cargar

-> Siempre se pueden recuperar los datos

-> Siempre se puede modificar la gráfica

-> Se puede exportar a pdf, png, etc

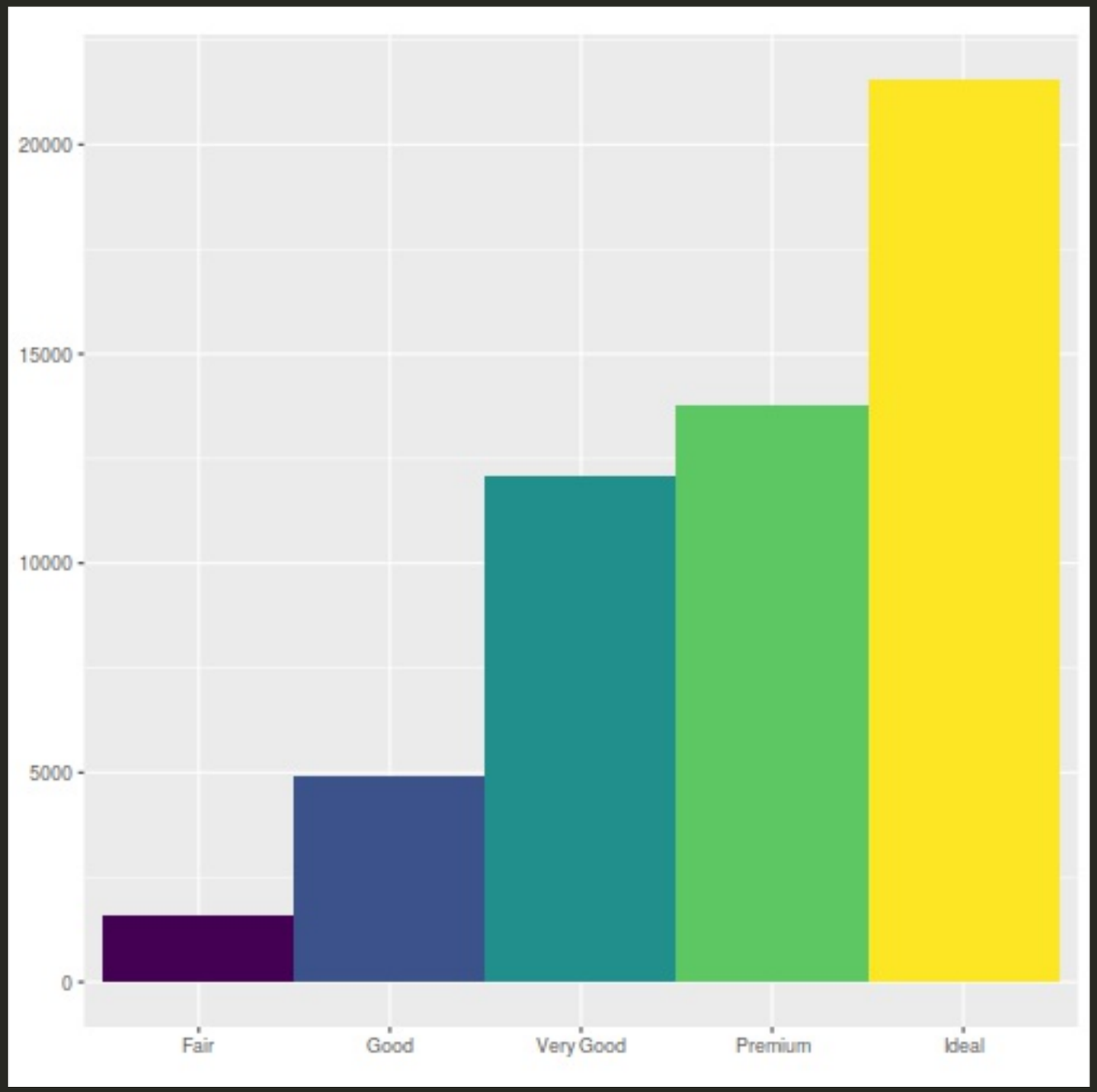
ggplot2:Saving

```
bar <- ggplot(data = diamonds) +
                geom_bar(
 mapping = aes(x = cut, fill = cut),
               show.legend = FALSE,
                    width = 1
                  ) +
              theme(aspect.ratio = 1) +
               labs(x = NULL, y = NULL)

ggsave(plot = bar, filename = "results/bar.pdf")
```

```
## Saving 7 x 7 in image
```

```
save(bar, file = "results/bar.RData")
```

```
rm(list=ls())
load("results/bar.RData")
bar
```



EJERCICIO de TAREA!!!