

Gráficos en R

Base y ggplot2

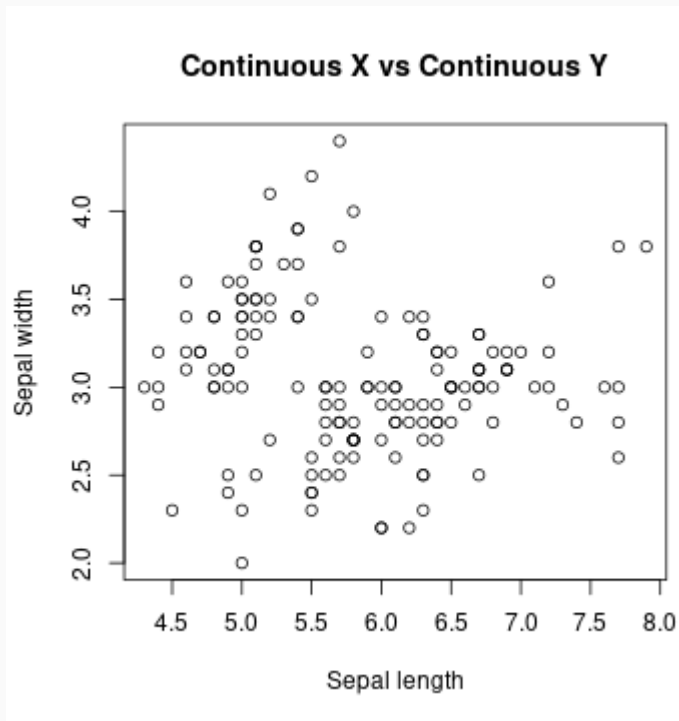
Dr. Samuel D. Gamboa Tuz

Base

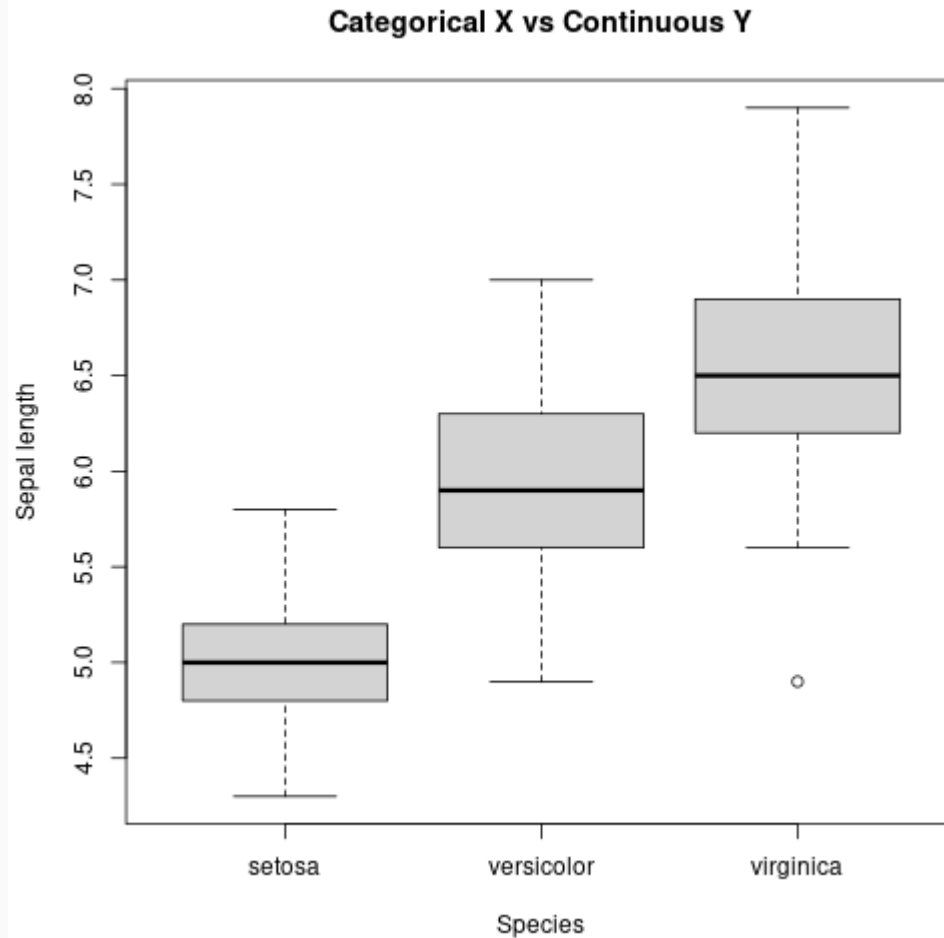
La función `plot()`

- La función `plot` se comporta de manera diferente dependiendo de los datos que le pasemos como argumento:

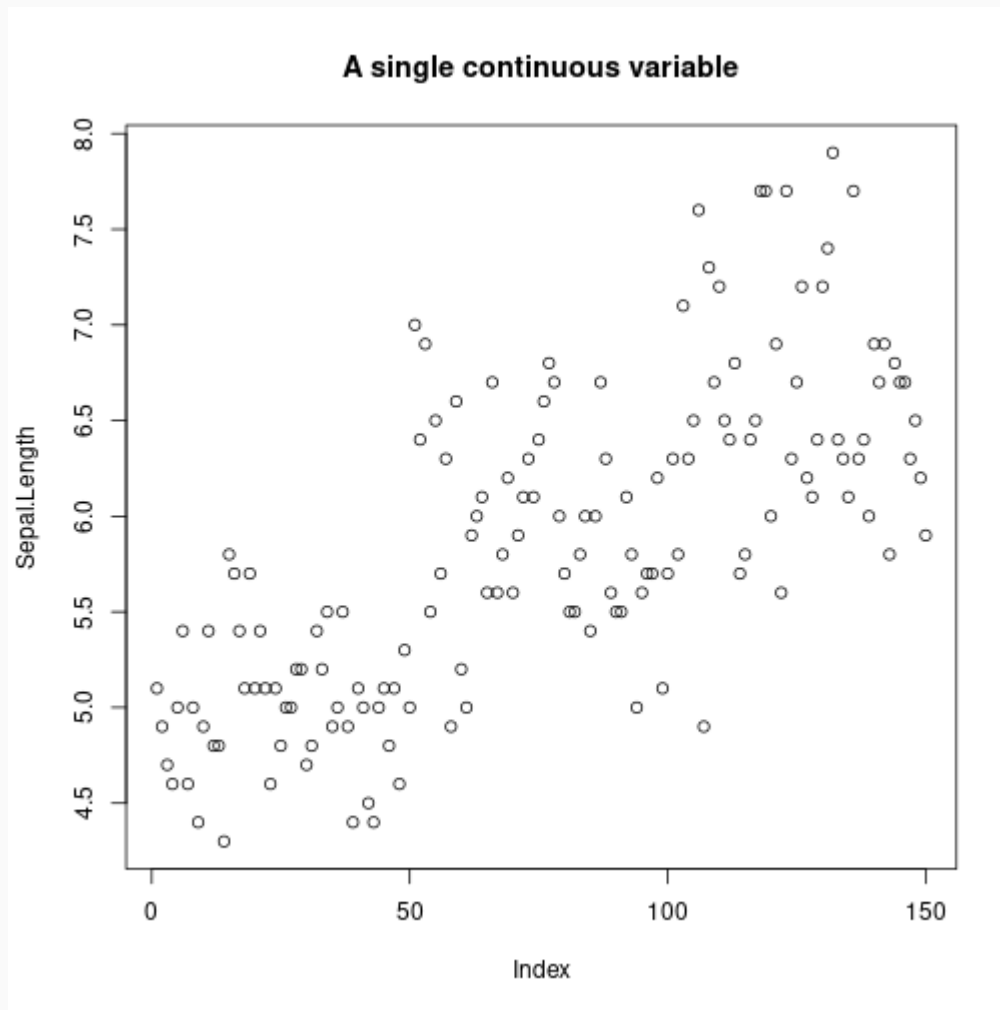
```
attach(iris)
plot(Sepal.Length, Sepal.Width,
     xlab = "Sepal length", ylab = "Sepal width",
     main = "Continuous X vs Continuous Y")
```



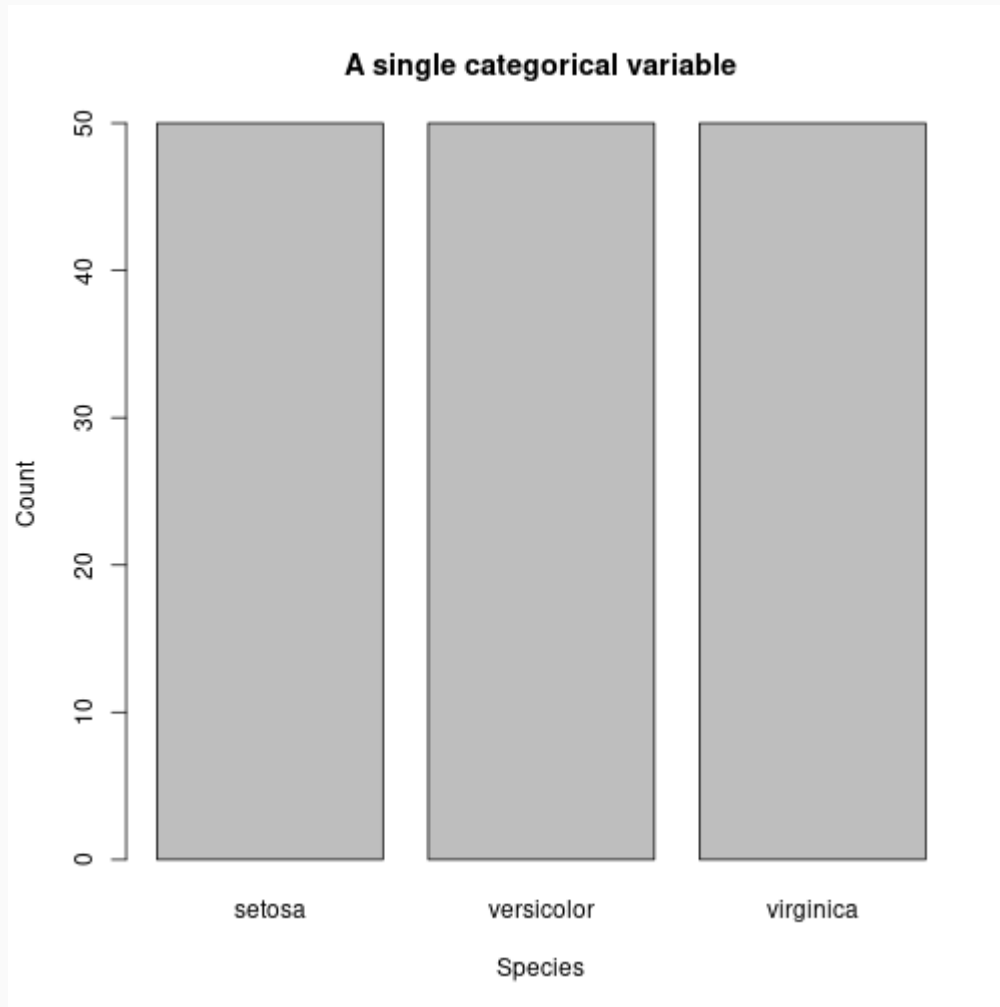
```
plot(Species, Sepal.Length,  
     xlab = "Species", ylab = "Sepal length",  
     main = "Categorical X vs Continuous Y")
```



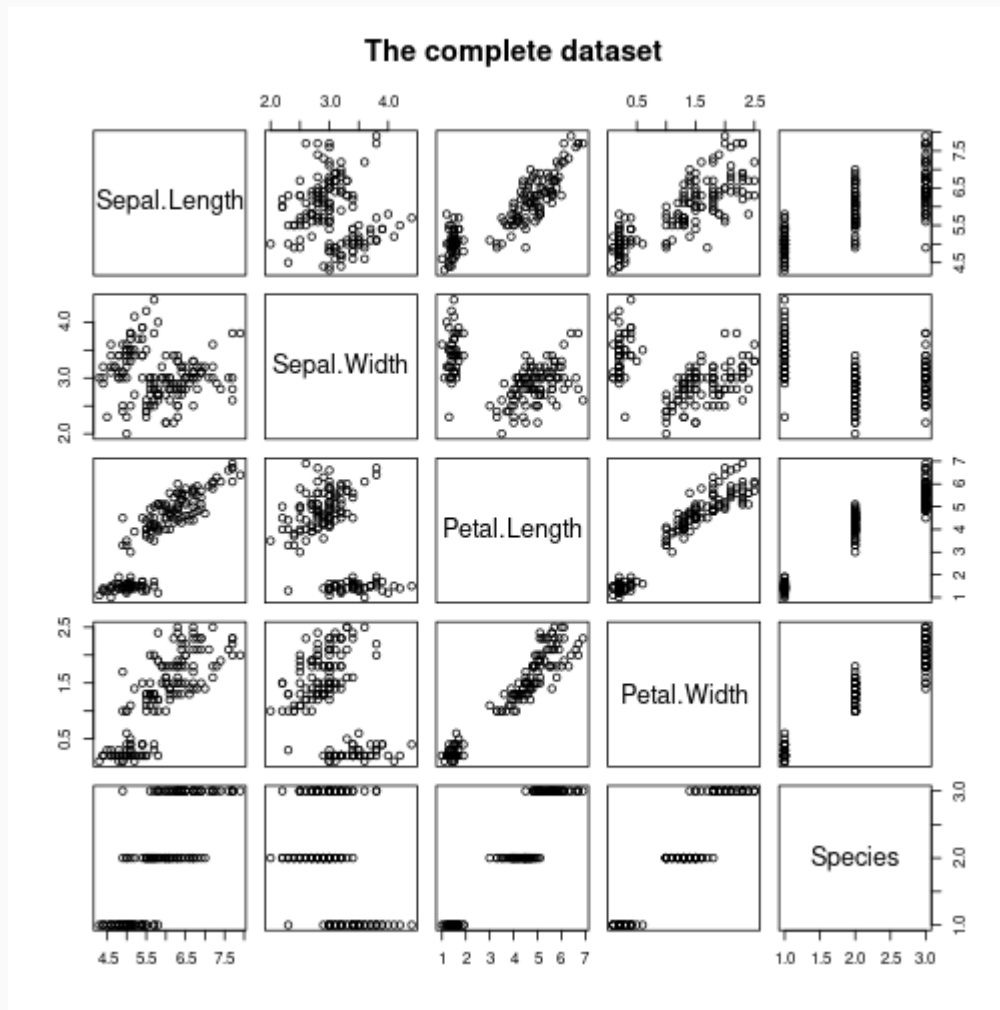
```
plot(Sepal.Length,  
     main = "A single continuous variable")
```



```
plot(Species,  
     ylab = "Count", xlab = "Species",  
     main = "A single categorical variable")
```

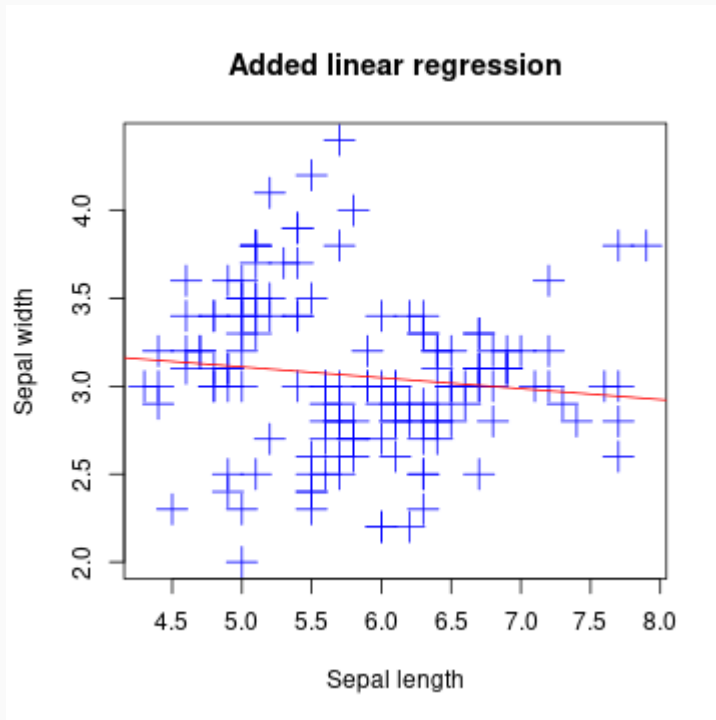


```
plot(iris,  
     main = "The complete dataset")
```



- Puedes modificar el gráfico con colores, formas, líneas, etc.

```
plot(Sepal.Length, Sepal.Width, pch = 3, cex = 2, col = "blue",  
     xlab = "Sepal length", ylab = "Sepal width",  
     main = "Added linear regression")  
abline(lm(Sepal.Width ~ Sepal.Length), col = "red")
```



- Para exportar el gráfico:
 - Se llama a una función con el tipo de archivo; ej. png, pdf, svg, etc.
 - Se corre el gráfico y todos sus componentes.
 - Se ejecuta `dev.off()`.

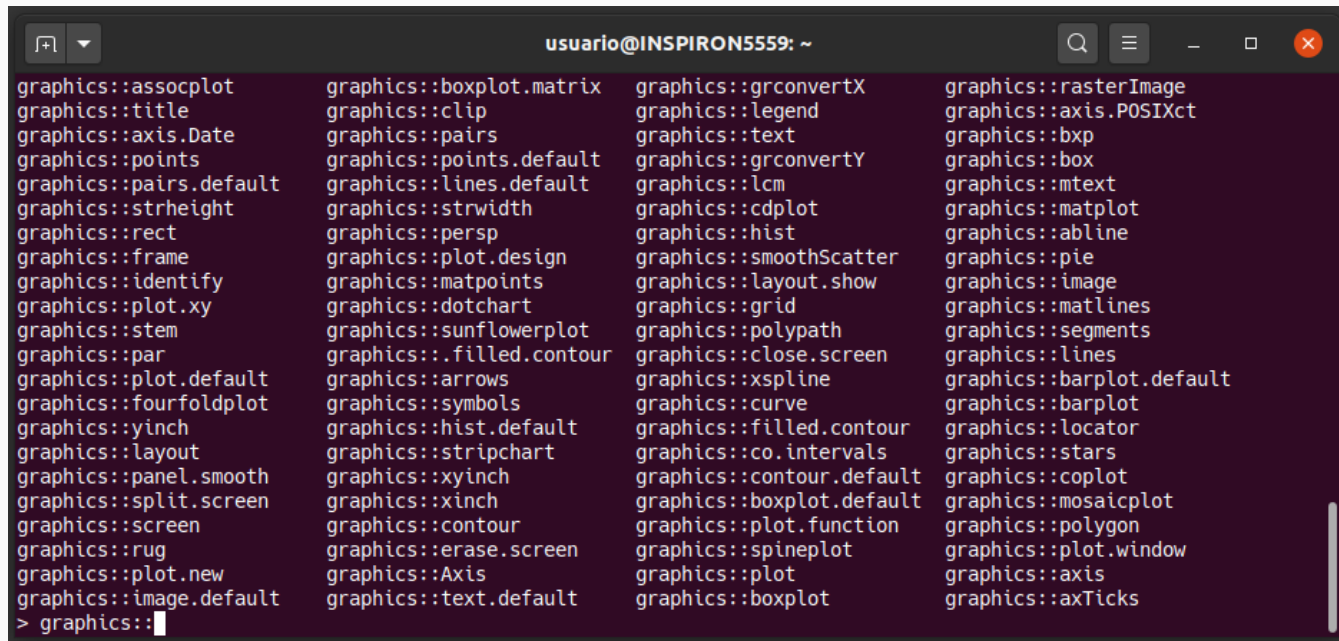
```
# Tipo de archivo de salida
png("test_plot.png", width = 5, height = 5, units = "in", res = 300)

# Elementos del gráfico (plot y abline)
plot(Sepal.Length, Sepal.Width, pch = 3, cex = 2, col = "blue",
      xlab = "Sepal length", ylab = "Sepal width",
      main = "Added linear regression")
abline(lm(Sepal.Width ~ Sepal.Length), col = "red")

# Finalizar con dev.off()
dev.off()
```

```
## png
## 2
```

- Además de `plot()`, hay varias funciones en el paquete **graphics** que sirven para crear diferentes tipos de gráficos.

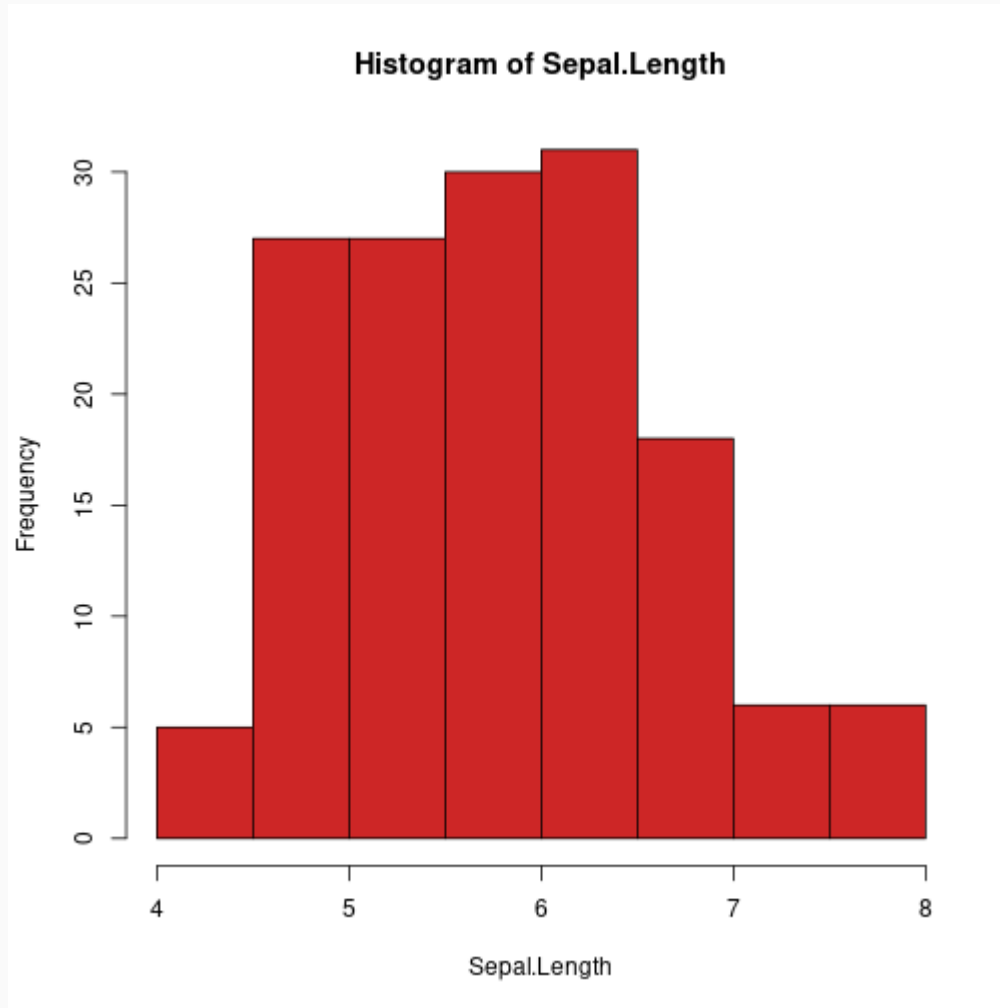


A terminal window titled 'usuario@INSPIRON5559: ~' displays the contents of the `graphics` package. The functions are listed in four columns. The first column contains functions like `assocplot`, `title`, `axis.Date`, `points`, `pairs.default`, `strheight`, `rect`, `frame`, `identify`, `plot.xy`, `stem`, `par`, `plot.default`, `fourfoldplot`, `yinch`, `layout`, `panel.smooth`, `split.screen`, `screen`, `rug`, `plot.new`, `image.default`, and `graphics::`. The second column contains `boxplot.matrix`, `clip`, `pairs`, `points.default`, `lines.default`, `strwidth`, `persp`, `plot.design`, `matpoints`, `dotchart`, `sunflowerplot`, `filled.contour`, `arrows`, `symbols`, `hist.default`, `stripchart`, `xyinch`, `xinch`, `contour`, `erase.screen`, `Axis`, and `text.default`. The third column contains `grconvertX`, `legend`, `text`, `grconvertY`, `lcm`, `cdplot`, `hist`, `smoothScatter`, `layout.show`, `grid`, `polypath`, `close.screen`, `xspline`, `curve`, `filled.contour`, `co.intervals`, `contour.default`, `boxplot.default`, `plot.function`, `spineplot`, `plot`, and `boxplot`. The fourth column contains `rasterImage`, `axis.POSIXct`, `bxp`, `box`, `mtext`, `matplot`, `abline`, `pie`, `image`, `matlines`, `segments`, `lines`, `barplot.default`, `barplot`, `locator`, `stars`, `coplot`, `mosaicplot`, `polygon`, `plot.window`, `axis`, and `axTicks`. The prompt `> graphics::` is at the bottom left.

```
usuario@INSPIRON5559: ~  
graphics::assocplot    graphics::boxplot.matrix  graphics::grconvertX    graphics::rasterImage  
graphics::title        graphics::clip            graphics::legend        graphics::axis.POSIXct  
graphics::axis.Date    graphics::pairs          graphics::text          graphics::bxp  
graphics::points       graphics::points.default  graphics::grconvertY    graphics::box  
graphics::pairs.default graphics::lines.default   graphics::lcm           graphics::mtext  
graphics::strheight    graphics::strwidth       graphics::cdplot        graphics::matplot  
graphics::rect         graphics::persp          graphics::hist          graphics::abline  
graphics::frame        graphics::plot.design    graphics::smoothScatter graphics::pie  
graphics::identify     graphics::matpoints      graphics::layout.show   graphics::image  
graphics::plot.xy      graphics::dotchart       graphics::grid          graphics::matlines  
graphics::stem         graphics::sunflowerplot  graphics::polypath     graphics::segments  
graphics::par          graphics::filled.contour graphics::close.screen   graphics::lines  
graphics::plot.default graphics::arrows         graphics::xspline       graphics::barplot.default  
graphics::fourfoldplot graphics::symbols        graphics::curve         graphics::barplot  
graphics::yinch        graphics::hist.default   graphics::filled.contour graphics::locator  
graphics::layout       graphics::stripchart     graphics::co.intervals  graphics::stars  
graphics::panel.smooth graphics::xyinch         graphics::contour.default graphics::coplot  
graphics::split.screen graphics::xinch          graphics::boxplot.default graphics::mosaicplot  
graphics::screen       graphics::contour        graphics::plot.function  graphics::polygon  
graphics::rug          graphics::erase.screen   graphics::spineplot     graphics::plot.window  
graphics::plot.new     graphics::Axis          graphics::plot          graphics::axis  
graphics::image.default graphics::text.default   graphics::boxplot       graphics::axTicks  
> graphics::
```

- Por ejemplo, un histograma puede crearse con `hist()`:

```
hist(Sepal.Length, col = "firebrick3"); detach(iris)
```



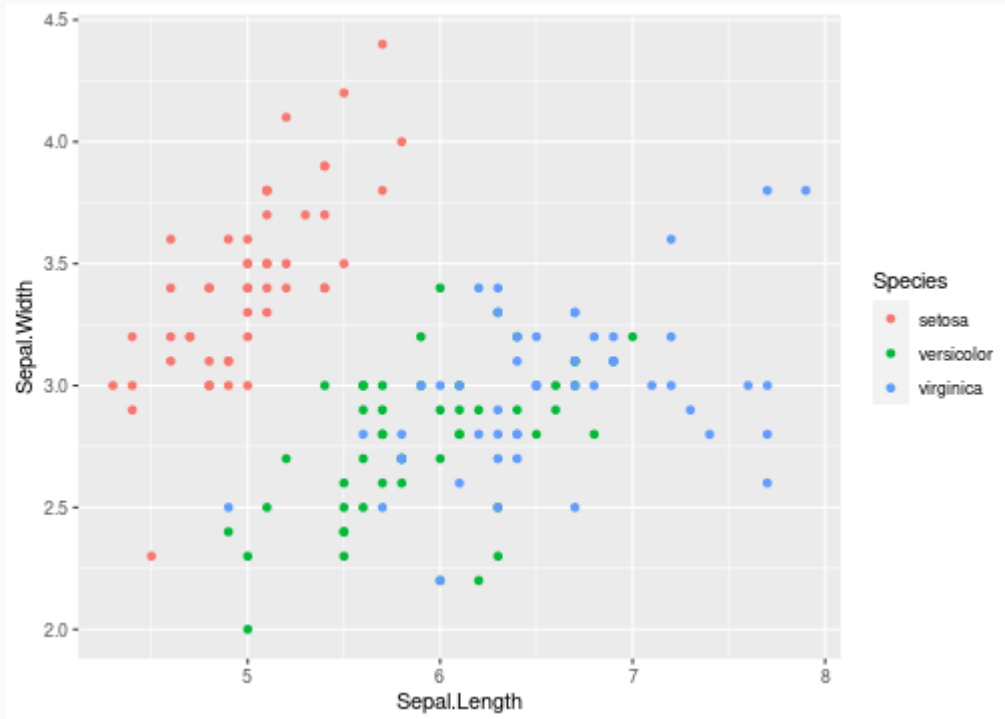
ggplot2

Gramática de gráficos

Gramática de gráficos y ggplot2

- **ggplot2** está basado en la gramática de gráficos.
- Un gráfico mapea (**mapping**) los **datos** a un atributo estético (**aesthetics**; color, forma, tamaño, etc.) de un objeto geométrico (**geom**; líneas, barras, puntos, etc.).
- Carguemos el paquete ggplot2 y creemos un gráfico de dispersión (siguiente diapositiva):

```
library(ggplot2)
ggplot( # call ggplot2
  data = iris, # data
  mapping = aes(x = Sepal.Length, y = Sepal.Width, color = Species) # aesthetic
) +
  geom_point() # geom
```



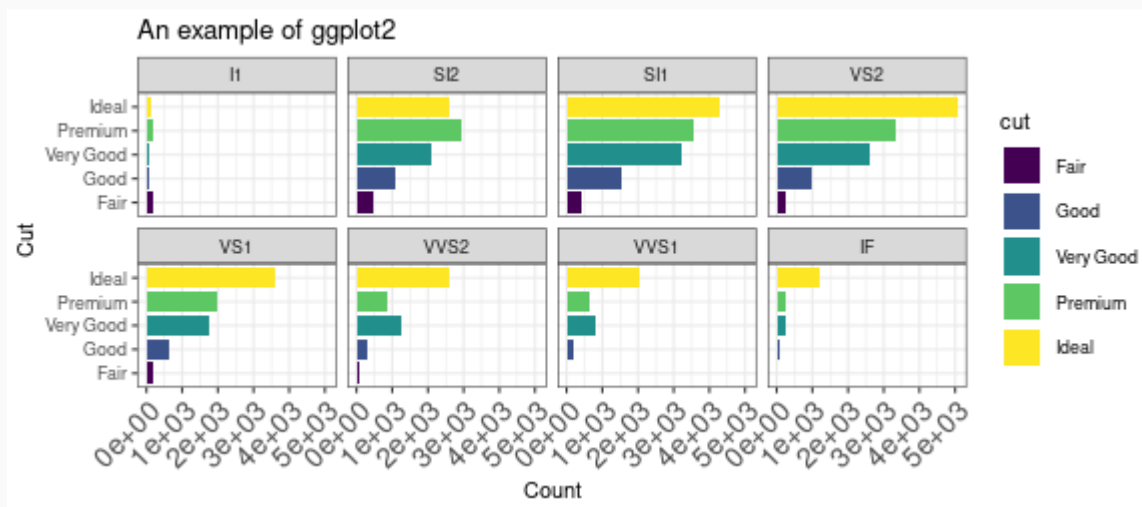
Componentes de un gráfico en ggplot2

| Elemento | Descripción |
|-------------|---|
| ggplot call | La función de llamado a ggplot2. Puede contener al mapping y aesthetics . También puede estar vacío. |
| layers | Colección de figuras geométricas (geoms) y transformaciones estadísticas (stats). Puede contener el <i>mapping</i> y <i>aesthetics</i> o estar vacío. |
| scales | Es el link entre datos y aesthetics (x, y, fill, size, shape, etc.). |
| guides | Controla el tamaño, color, etc. de los símbolos de las leyendas. |
| labels | Título de la gráfica y los ejes. |
| coordinates | Describen como se mapean las coordenadas del gráfico en el plano (cartesiano, polar, etc). |
| facets | Especifica como partir y mostrar subconjuntos de datos. |
| theme | Decoración de líneas, ejes, texto, fondo, etc (no confundir con aesthetics). Hay temas predefinidos que pueden ser modificados. |

```

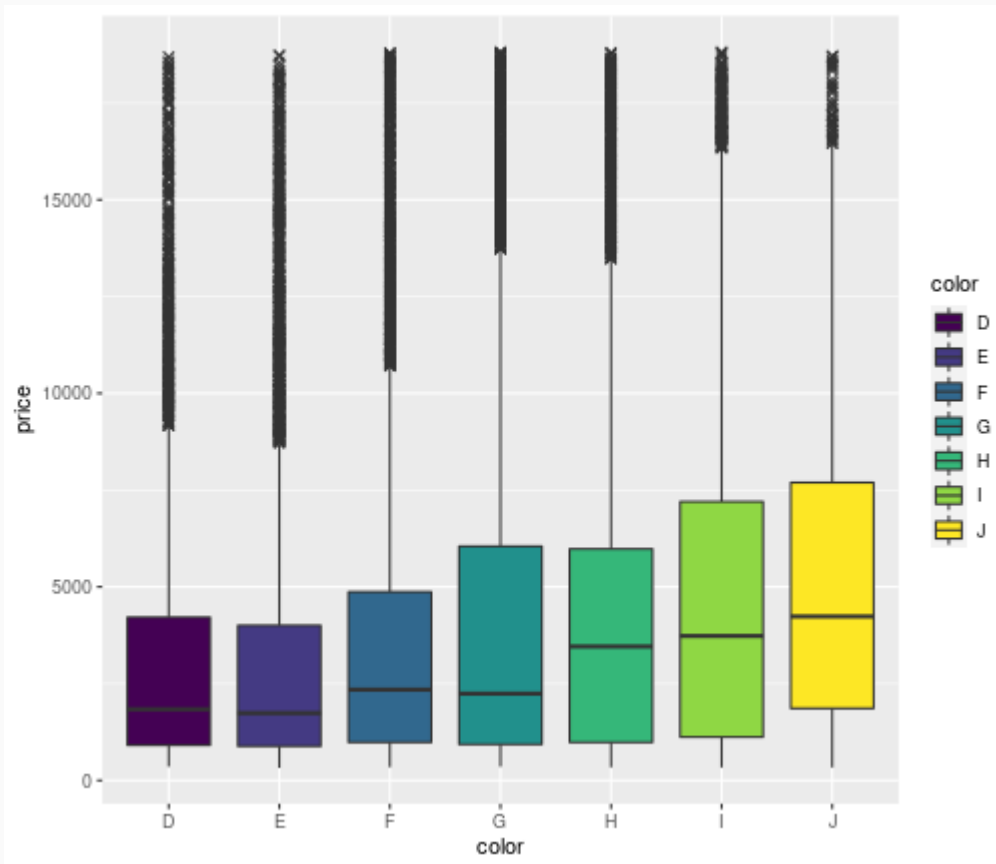
ggplot( # ggplot call
  data = diamonds, # data
  mapping = aes(x = cut, fill = cut) # mapping, aesthetics
) +
  geom_bar(stat = "count") + # geom + stat
  scale_y_continuous( # scales
    labels = function(x) format(x, scientific = TRUE)
  ) +
  guides(fill = guide_legend(override.aes = list(size = 8))) + # guides
  labs(x = "Cut", y = "Count", title = "An example of ggplot2") + # labels
  coord_flip() + # coordinates
  facet_wrap(~clarity, nrow = 2) + # facet
  theme_bw() + # theme (predefined)
  theme( # theme (custom)
    axis.text.x = element_text(size = 14, angle = 45, hjust = 1)
  )

```



- A diferencia de `plot()`, puedes salvar tus gráficos creados en `ggplot2` como un objeto:

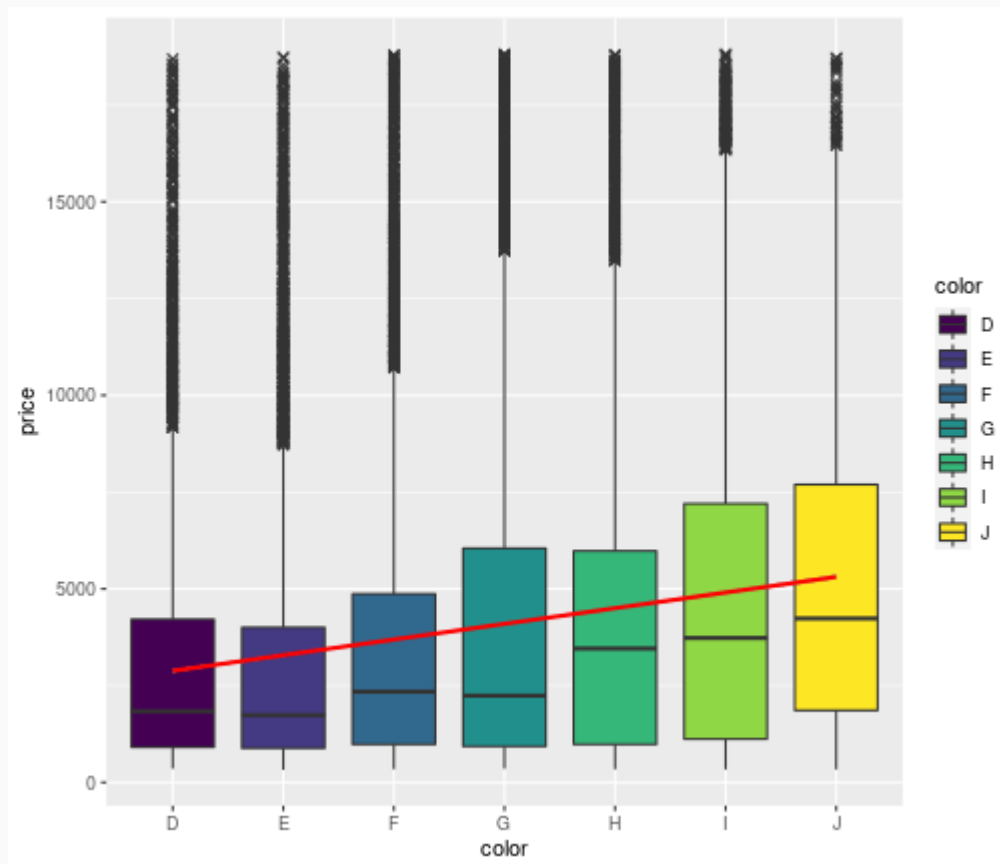
```
p <- ggplot(diamonds,aes(color, price)) +  
  geom_boxplot(aes(fill = color),  
              outlier.shape = 4, outlier.size = 2)  
p
```



- Puedes agregarle más geoms y componentes a tu gráfico guardado.

```
p <- p +  
  geom_smooth(aes(group = 1), method = "lm", color = "red")  
p
```

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



- ggplot2 tiene su propia función para exportar gráficos a png, svg, pdf, etc.

```
# Salvar como png  
ggsave("test_ggplot.png", p)
```

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

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

```
# Salvar como pdf  
ggsave("test_ggplot.pdf", p)
```

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

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

```
# Salvar como svg  
ggsave("test_ggplot.svg", p)
```

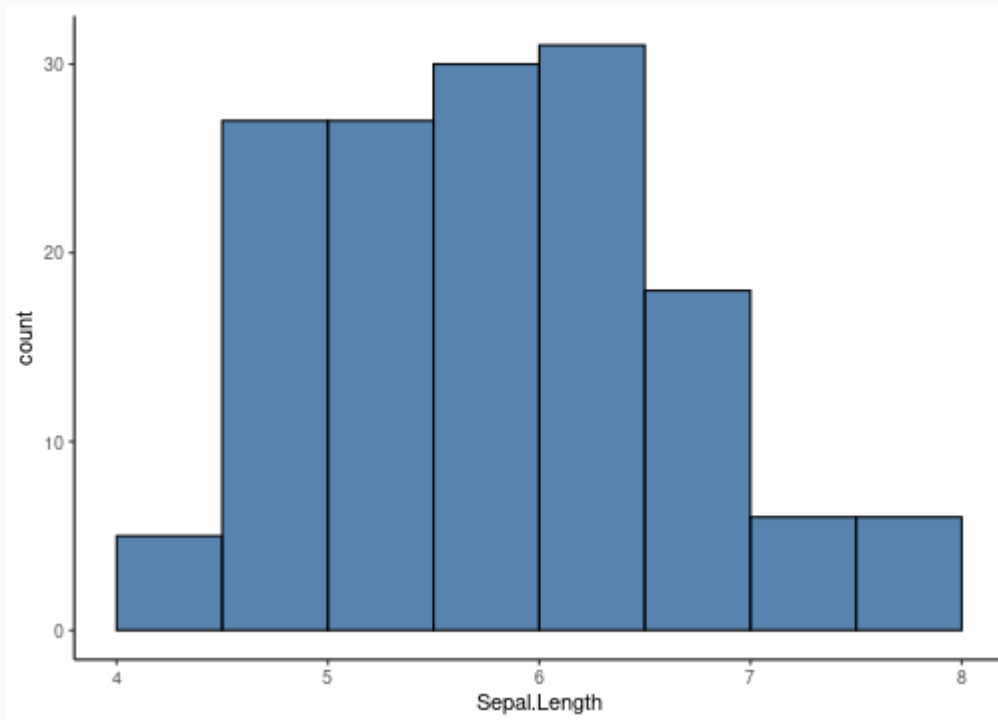
```
## Saving 7 x 7 in image
```

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

- Recomiendo salvarlo en svg para poder modificarlo luego (de ser necesario) sin perder resolución, por ejemplo, con Inkscape.
- Recomiendo los paquetes **Cairo** y **SVGLite**, pero debes salvar tu gráficos como los creados con plot, no con `ggsave()`.

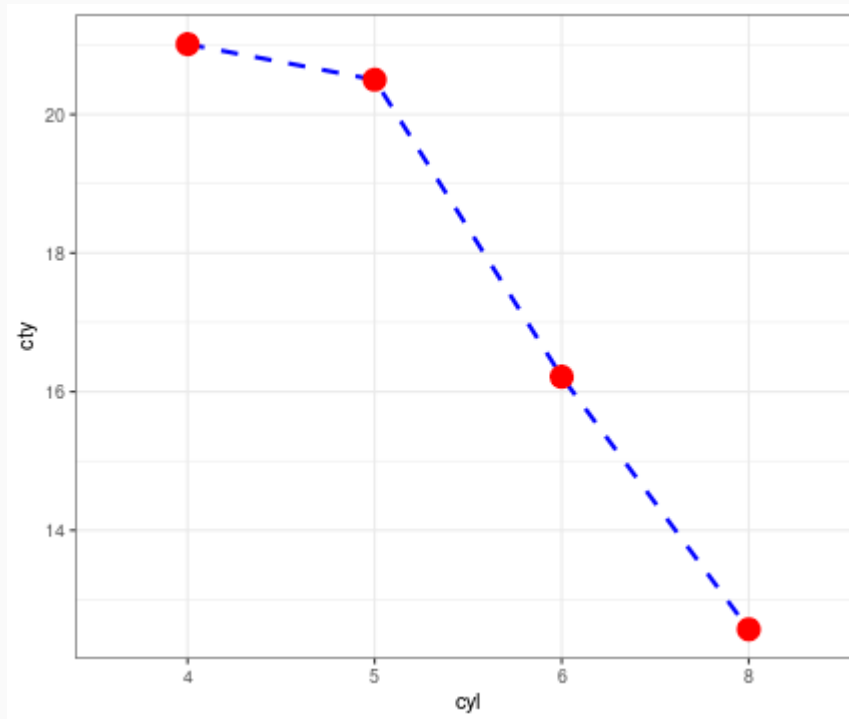
- Un histograma de frecuencias con ggplot2:

```
ggplot(iris, aes(Sepal.Length)) +  
  geom_histogram(binwidth = 0.5, center = 0.25, alpha = 0.7,  
                 fill = "dodgerblue4", color = "black") +  
  theme_classic()
```



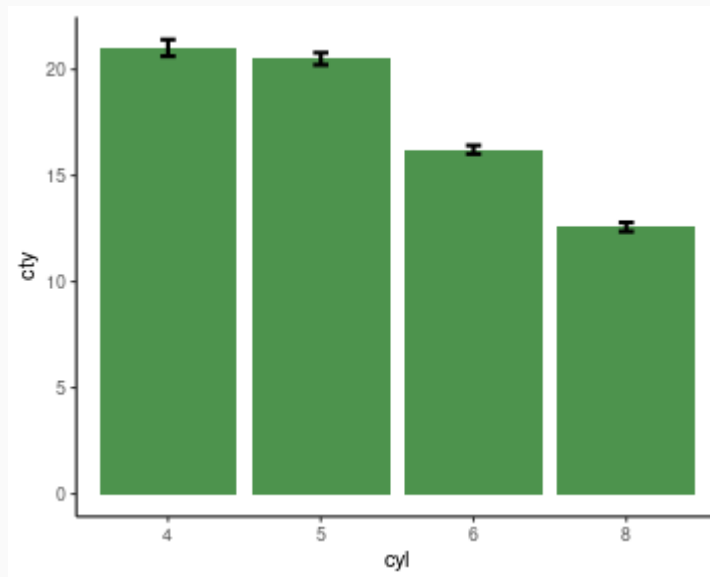
- Podemos indicar otros *stats* al momento de crear los gráficos. Por ejemplo, con `geom_line()` y `geom_point()`:

```
mpg_copy <- mpg
mpg_copy$cyl <- factor(mpg_copy$cyl)
ggplot(mpg_copy, aes(cyl, cty)) +
  geom_line(aes(group = 1), stat = "summary",
            fun = "mean", color = "blue", size = 1, linetype = "dashed") +
  geom_point(stat = "summary", fun = "mean", color = "red", size = 5) +
  theme_bw()
```



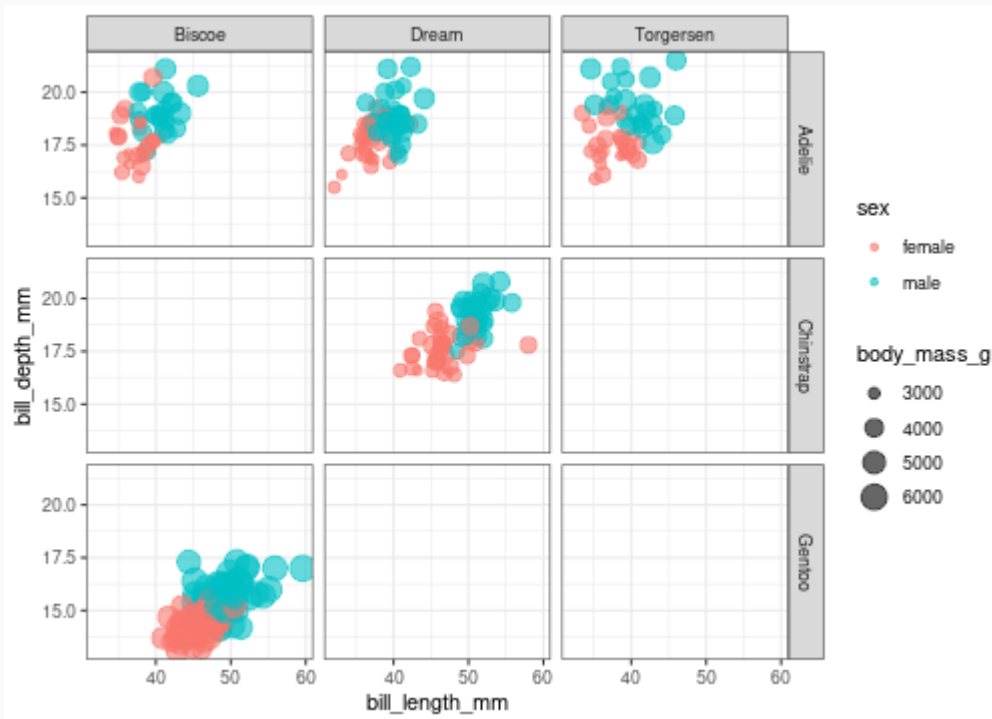
- Con `geom_bar()` y `geom_errorbar()`:

```
ggplot(mpg_copy , aes(cyl, cty)) +  
  geom_bar(stat = "summary", fun = "mean",  
           fill = "darkgreen", alpha = 0.7) +  
  geom_errorbar(stat = "summary", fun.data = "mean_se",  
               width = 0.1, size = 1) +  
  theme_classic()
```



- Probemos `facet_grid()` con *palmer penguins*.

```
penguin_filename <- paste0("https://raw.githubusercontent.com/",  
  "rfordatascience/tidytuesday/master/data/",  
  "2020/2020-07-28/penguins.csv")  
penguins <- readr::read_csv(penguin_filename)  
ggplot(na.omit(penguins), aes(bill_length_mm, bill_depth_mm)) +  
  geom_point(aes(color = sex, size = body_mass_g), alpha = 0.6) +  
  facet_grid(species ~ island) +  
  theme_bw()
```



Cambiar el orden de variables con factores

- En la figura anterior, los nombres de las islas y de las especies aparecen en orden alfabético. Podemos ponerlas en el orden que queramos con factores.

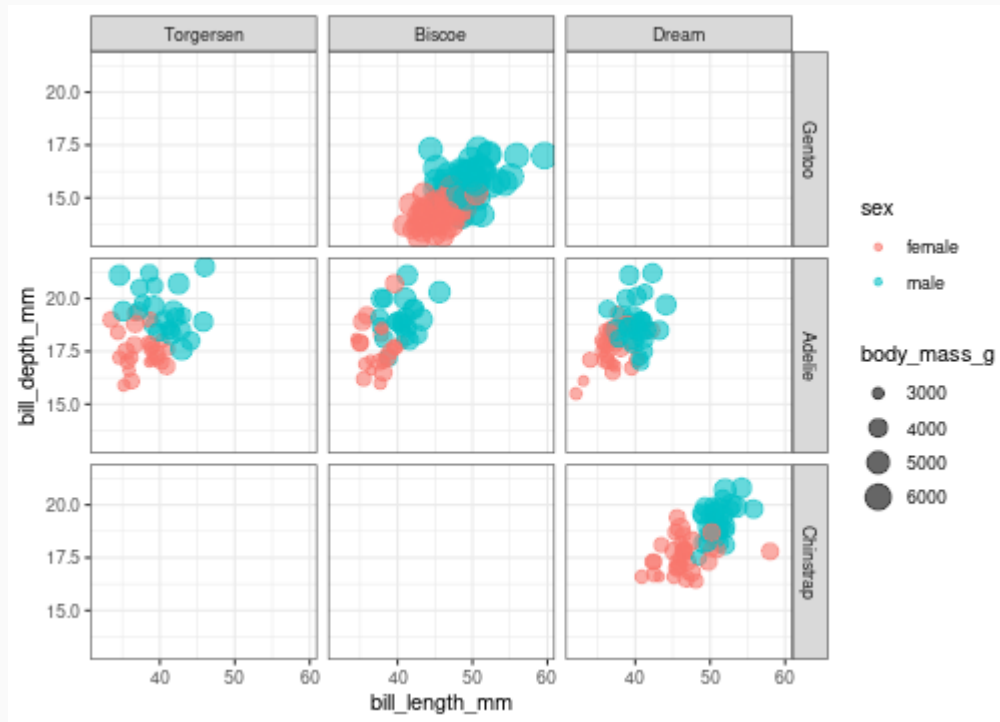
```
ordered_species <- c("Gentoo", "Adelie", "Chinstrap")
ordered_islands <- c("Torgersen", "Biscoe", "Dream")
penguins$species <- factor(penguins$species, levels = ordered_species)
penguins$island <- factor(penguins$island, levels = ordered_islands)
penguins
```

```
## # A tibble: 344 x 8
##   species island bill_length_mm bill_depth_mm flipper_length_...
##   <fct>   <fct>         <dbl>         <dbl>         <dbl>
## 1 Adelie  Torge...           39.1           18.7           181
## 2 Adelie  Torge...           39.5           17.4           186
## 3 Adelie  Torge...           40.3           18            195
## 4 Adelie  Torge...           NA            NA            NA
## 5 Adelie  Torge...           36.7           19.3           193
## 6 Adelie  Torge...           39.3           20.6           190
## 7 Adelie  Torge...           38.9           17.8           181
## 8 Adelie  Torge...           39.2           19.6           195
## 9 Adelie  Torge...           34.1           18.1           193
## 10 Adelie Torge...           42            20.2           190
## # ... with 334 more rows, and 3 more variables: body_mass_g <dbl>,
## #   sex <chr>, year <dbl>
```

- Ahora las columnas *species* e *island* son factores.

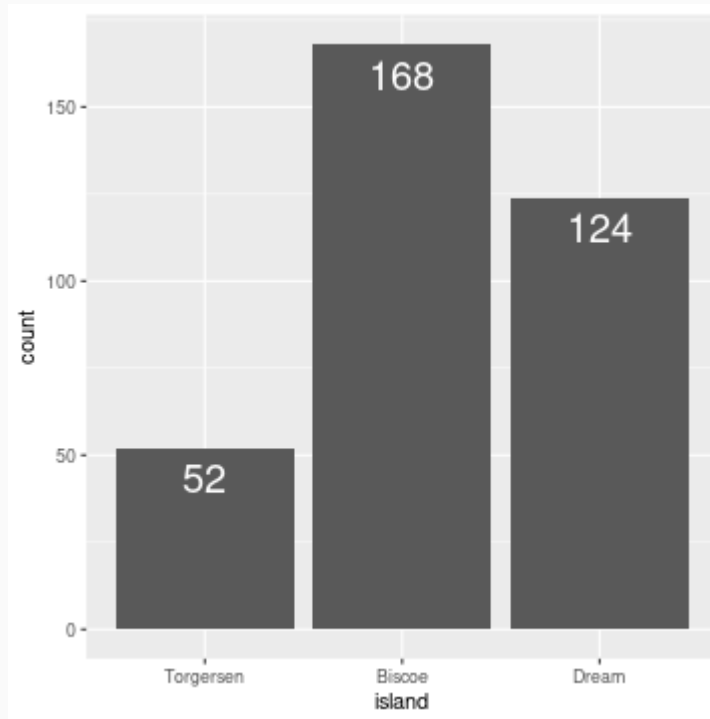
- Al graficar, los nombres de las islas y especies aparecen en el orden indicado en los factores:

```
ggplot(na.omit(penguins), aes(bill_length_mm, bill_depth_mm)) +  
  geom_point(aes(color = sex, size = body_mass_g), alpha = 0.6) +  
  facet_grid(species ~ island) +  
  theme_bw()
```



- También podemos cambiar el orden de los factores en función a otra variable.

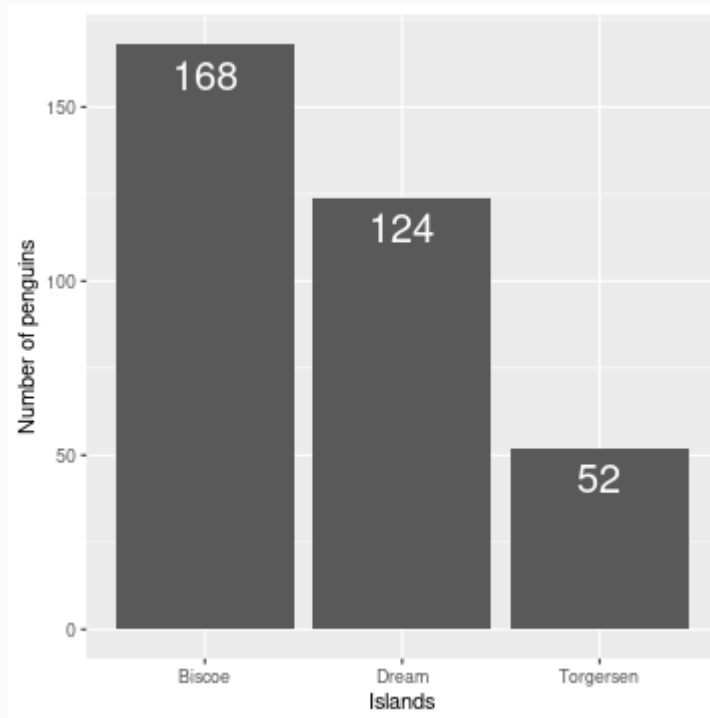
```
# Calcular el número de pingüinos por isla  
islands_count <- dplyr::count(penguins, island, name = "count")  
ggplot(islands_count, aes(island, count)) +  
  geom_col() +  
  geom_text(aes(label = count), color = "white", vjust = 1.5, size = 7)
```



- Aquí, las islas aparecen en el orden que habíamos indicado.

- Podemos ordenar las islas de acuerdo al número de pingüinos, de mayor a menor:

```
ggplot(islands_count, aes(reorder(island, -count), count)) +  
  geom_col() +  
    geom_text(aes(label = count), color = "white", vjust = 1.5, size = 7) +  
  labs(x = "Islands", y = "Number of penguins")
```



Forcats

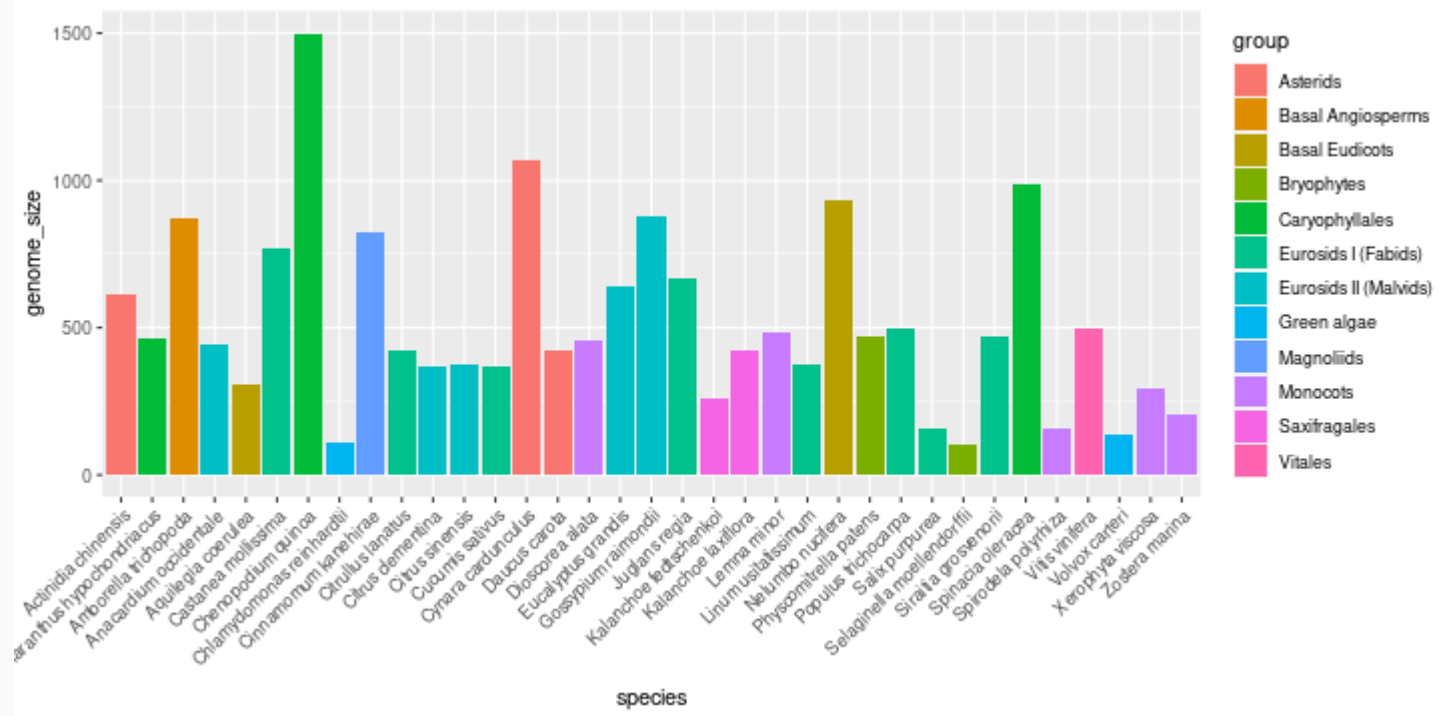
- El paquete **forcats** provee un conjunto de funciones que pueden hacer más fácil ordenar o cambiar los valores de un vector tipo *factor*.
- En el siguiente ejemplo, las especies de plantas están ordenadas de acuerdo a su grupo taxonómico y quiero conservar ese orden. Además, los colores para cada grupo taxonómico están indicados en una columna:

```
(plants <- readxl::read_excel("datasets.xlsx", sheet = 2))
```

```
## # A tibble: 35 x 4
##   group          group_color species          genome_size
##   <chr>          <chr>      <chr>          <dbl>
## 1 Green algae    #8f9a9b    Chlamydomonas reinhar...    110
## 2 Green algae    #8f9a9b    Volvox carteri            138
## 3 Bryophytes     #008933    Physcomitrella patens      472
## 4 Bryophytes     #008933    Selaginella moellendo...    100
## 5 Basal Angiospe... #a52a2a    Amborella trichopoda       870
## 6 Magnoliids     #ff8330    Cinnamomum kanehirae       823
## 7 Monocots       #00d958    Zostera marina             202
## 8 Monocots       #00d958    Lemna minor                481
## 9 Monocots       #00d958    Spirodela polyrhiza        158
## 10 Monocots      #00d958    Xerophyta viscosa          295
## # ... with 25 more rows
```

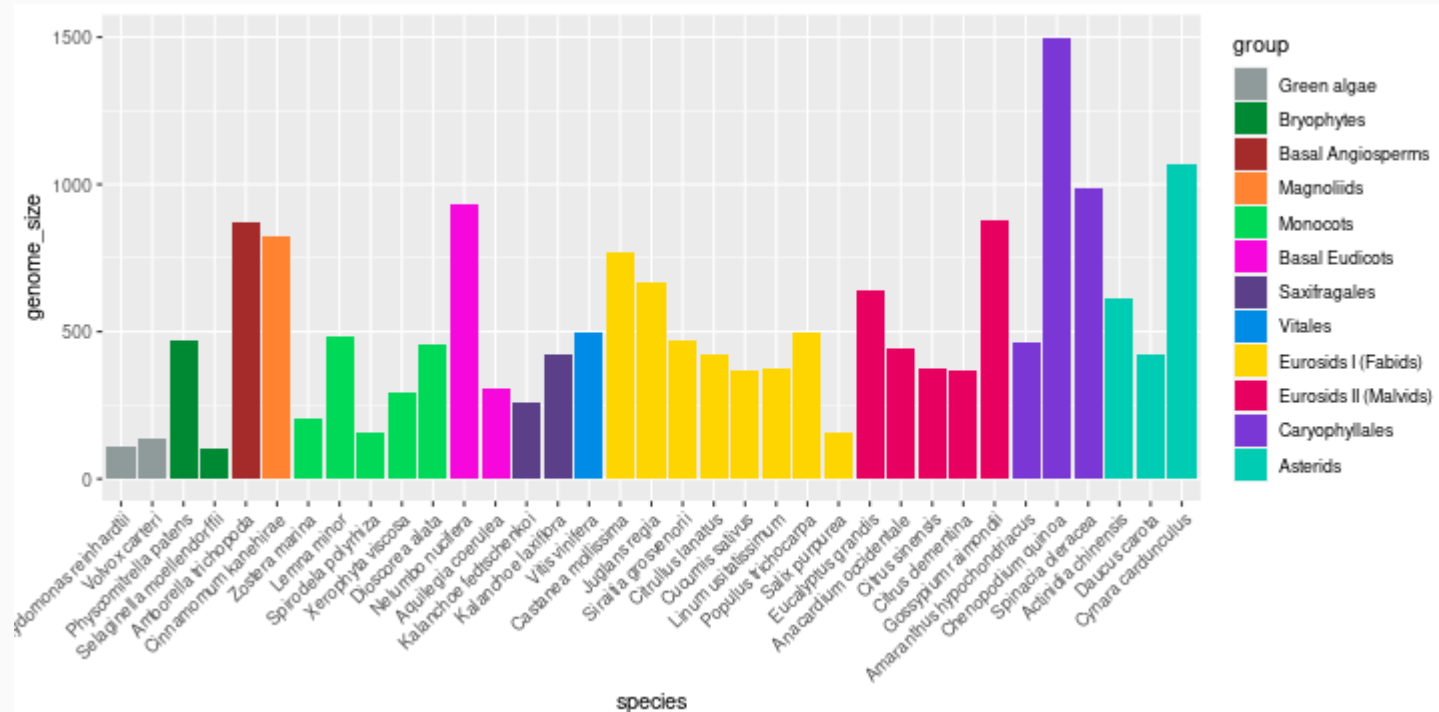
- Al graficar, las especies y grupos aparecen en orden alfabético y los colores los define R de acuerdo a su paleta default:

```
ggplot(plants, aes(species, genome_size, fill = group)) +  
  geom_col() +  
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



- Puedo conservar el orden de especies y grupos con `forcats::fct_inorder()`, y utilizar la columna de colores para usar mi propia paleta:

```
plants$species <- forcats::fct_inorder(plants$species)
plants$group <- forcats::fct_inorder(plants$group)
group_colors <- unique(plants$group_color)
ggplot(plants, aes(species, genome_size, fill = group)) +
  geom_col() +
  scale_fill_manual(values = group_colors) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Actividades recomendadas

- ¿Cuál es la diferencia entre `geom_bar()` y `geom_col()`?
- ¿Cómo harías una gráfica de barras agrupadas? ¿Y una de barras apiladas? Checa el argumento "position" de `geom_col`. No olvides tomar en cuenta las barras de error.
- ¿Cuál es la diferencia entre `geom_line()` y `geom_path()`.
- ¿Cómo pondrías varias gráficas en una sola figura? Checa la función `ggarrange()` del paquete `ggpubr()`.
- El paquete `ggrepel()` te puede ser útil cuando quieras usar `geom_text()` con `geom_point()`.
- ¿Cómo pondrías sub- y superíndices en los títulos de los ejes x y y? ¿Cómo pondrías símbolos griegos?

Bibliografía y links recomendados

- Capítulos 2 y 15 de *R for Data Science* (<https://r4ds.had.co.nz/data-visualisation.html>).
- Wilkinson, Leland. 2005. *The Grammar of Graphics*. 2nd ed. Statistics and Computing. Springer).
- ggplot2: Elegant Graphics for Data Analysis Hadley Wickham, Danielle Navarro, and Thomas Lin Pedersen (<https://ggplot2-book.org/>).
- Claus Wilke. *Fundamentals of data visualization* (<https://clauswilke.com/dataviz/index.html>).
- Una galería con gráficos de R: <https://www.r-graph-gallery.com>.
- #tidytuesday en twitter (<https://github.com/rfordatascience/tidytuesday/tree/master/data>).
- Ordenar factores en facets: <https://juliasilge.com/blog/reorder-within>.