

Introdução ao uso de dados geoespaciais no R

5 Visualização de dados

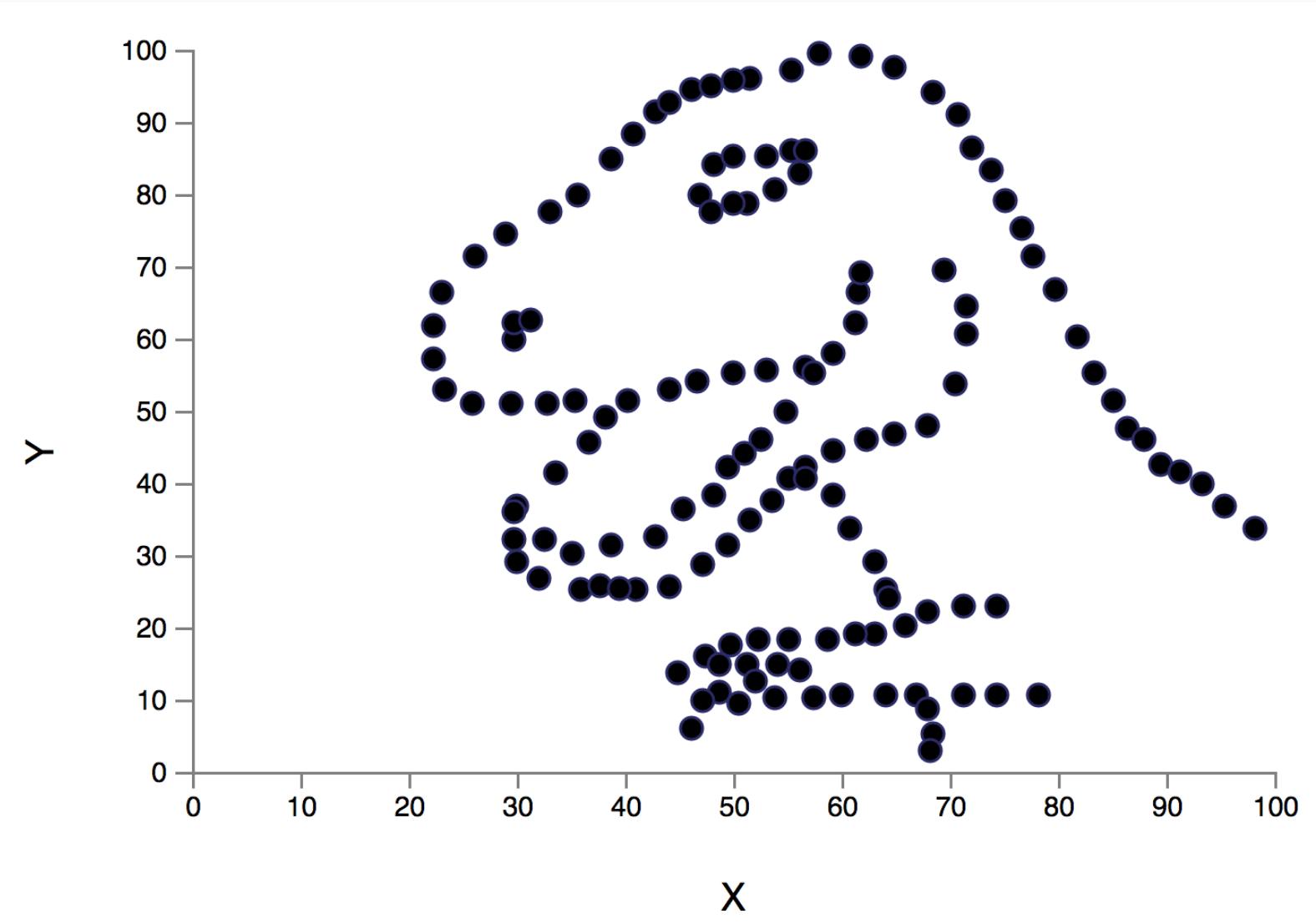
Maurício H. Vancine

Milton C. Ribeiro

UNESP - Rio Claro

Laboratório de Ecologia Espacial e Conservação (LEEC)

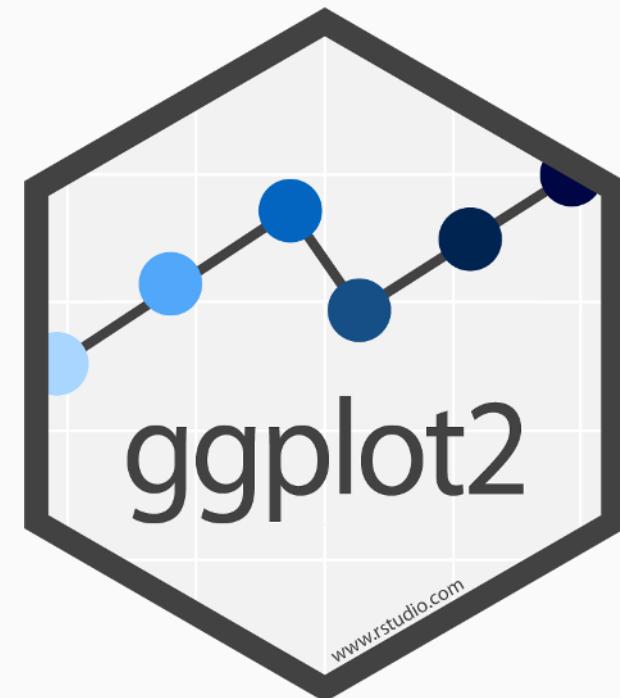
25/10/2021-05/11/2021



5 Visualização de dados

Conteúdo

1. Contextualização
2. Pacotes para produção de gráficos
3. Gramática dos gráficos
4. Principal material de estudo
5. Principais tipos de gráficos
6. Histograma e Densidade
7. Gráfico de setores
8. Gráfico de barras
9. Gráfico de caixas
10. Gráfico de dispersão
11. Gráfico pareado
12. Combinando gráficos
13. Gráficos animados
14. Gráficos interativos
15. Gráficos usando interface



5 Visualização de dados

Script

```
05_script_intro_geoespacial_r.R
```

1. Contextualização

Descrição

Melhor forma de **apresentar, sintetizar, discutir e interpretar** seus dados

Necessário em quase todas as **análises estatísticas**

Necessário em quase todas as **publicações**, trabalhos de consultoria, TCC, dissertação, tese, etc.

Existem vários **tipos de gráficos** para representar os padrões em seus dados para **diferentes tipos de finalidades**

De forma simplificada, os gráficos são **representações** dos nossos dados tabulares

2. Pacotes para produção de gráficos

Principais pacotes

[graphics](#): simples, porém útil para visualizações rápidas de quase todos os formatos de arquivos

[ggplot2](#): pacote integrado ao tidyverse, possui uma sintaxe própria baseada na gramática de gráficos por camadas (layers)

[ggplot2 extensions](#): conjunto de pacotes que adicionam diversas expansões ao pacote ggplot2. Exemplos: [GGally](#), [ggridge](#) e [esquisse](#)

[visdat](#): cria visualizações preliminares de um conjunto de dados inteiro para identificar problemas ou recursos inesperados usando `ggplot2`

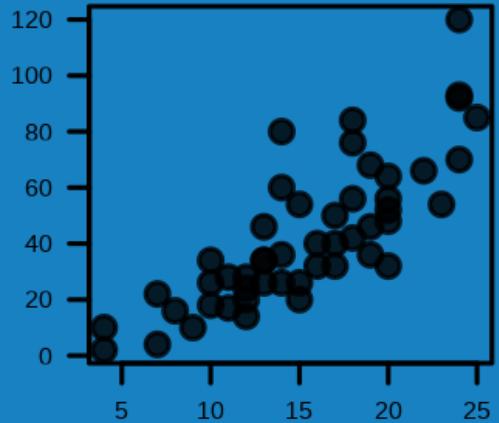
[ggpubr](#): fornece algumas funções simplificadas para criar gráficos para publicação, baseados no `ggplot2`

[cowplot](#): pacote que permite combinar vários gráficos de forma elaborada

[patchwork](#): pacote que permite combinar vários gráficos em um só de forma extremamente simples e com alta qualidade

[plotly](#): pacote para criar gráficos interativos da web por meio da biblioteca gráfica de JavaScript de código aberto plotly.js

graphics



2. Principais pacotes para gráficos

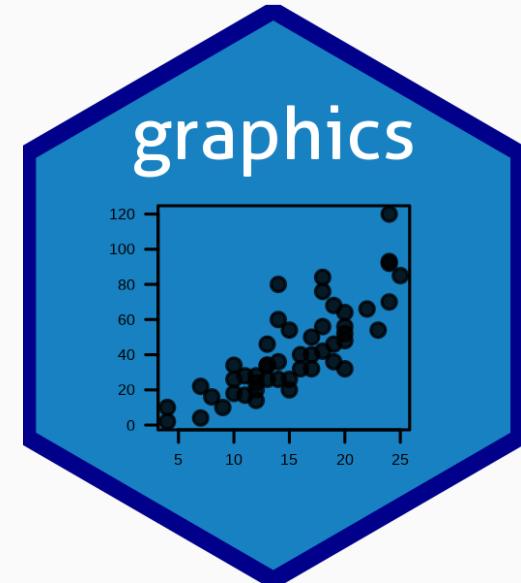
graphics

Default do R e mais simples

Pode ser utilizado para objetos de **diversas classes**

Possui funções como:

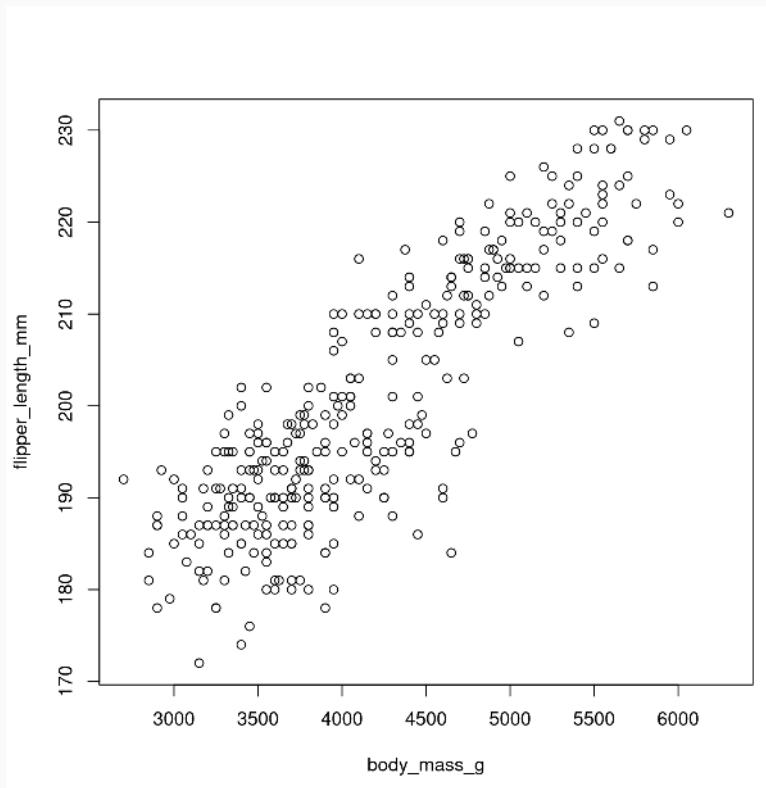
```
plot()  
hist()  
barplot()  
boxplot()  
abline()  
points()  
lines()  
polygon()
```

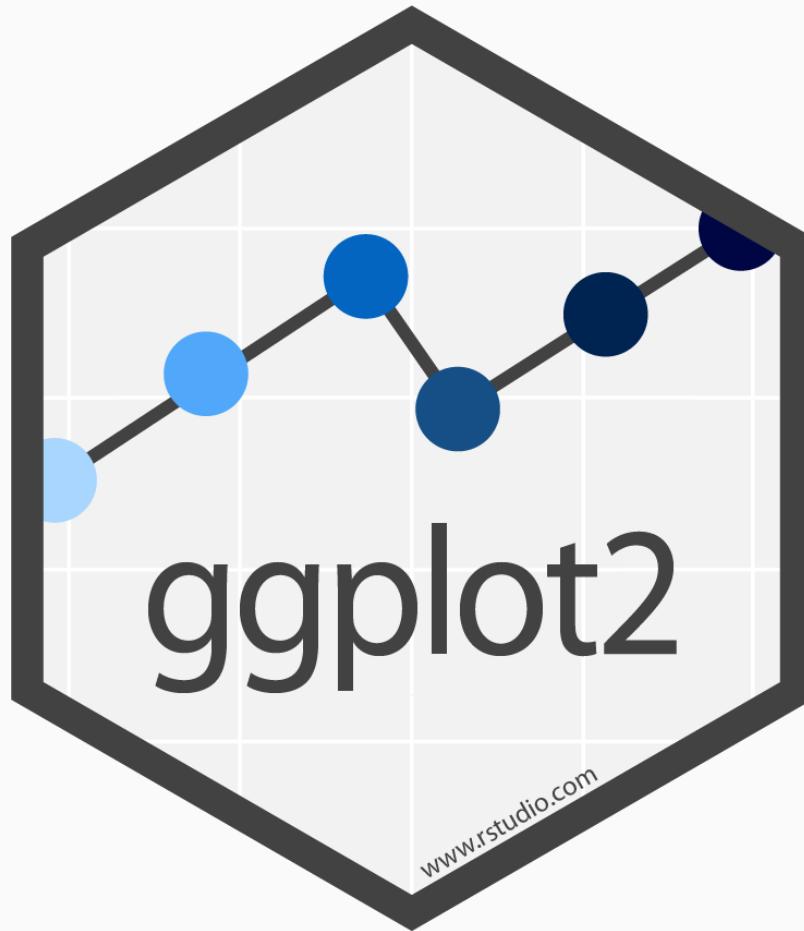


2. Principais pacotes para gráficos

graphics

```
# graphics  
plot(flipper_length_mm ~ body_mass_g, data = penguins)
```





[ggplot2](#)

2. Principais pacotes para gráficos

Data Visualization Cheatsheet

Data Visualization with ggplot2 :: CHEAT SHEET



Basics

ggplot2 is based on the **grammar of graphics**, the idea that you can build every graph from the same components: a **data set**, a **coordinate system**, and **geoms**—visual marks that represent data points.

To display values, map variables in the data to visual properties of the geom (**aesthetics**) like **size**, **color**, and **x** and **y** locations.

Completes the template below to build a graph.

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

Aesthetic mappings **data** **geom**

`ggplot(x = cty, y = hwy, data = mpg, geom = "point")`
Creates a complete plot with given data, geom, and mappings. Supplies many useful defaults.

`last_plot()` Returns the last plot

`ggsave("plot.png", width = 5, height = 5)` Saves last plot as 5' x 5' file named "plot.png" in working directory. Matches file type to file extension.

Geoms Use a geom function to represent data points, use the geom's aesthetic properties to represent variables. Each function returns a layer.

GRAPHICAL PRIMITIVES

- `a + geom_blank()` (Useful for expanding limits)
- `b + geom_curve(aes(yend = lat + 1, xend = lon + 1, curvature = 2))` x, yend, y, end, alpha, angle, color, curvature, linetype, size
- `a + geom_path(lineend = "butt", linejoin = "round")` x, y, alpha, color, group, linetype, size
- `a + geom_polygon(aes(group = group))` x, y, alpha, color, fil, group, linetype, size
- `b + geom_rect(aes(xmin = long, ymin = lat + 1), xmax, xmin, ymax = long + 1, ymax = lat + 1))` x, y, alpha, color, fill, group, linetype, size
- `a + geom_ribbon(aes(ymin = unemploy - 900, ymax = unemploy + 900))` x, y, alpha, color, fill, group, linetype, size

LINE SEGMENTS common aesthetics: x, y, alpha, color, linetype, size

- `b + geom_abline(aes(intercept = 0, slope = 1))`
- `b + geom_hline(aes(intercept = 0))`
- `b + geom_vline(aes(xintercept = 0))`
- `b + geom_segment(aes(yend = lat + 1, xend = lon + 1))`
- `b + geom_spoke(aes(angle = 1:115, radius = 1))`

ONE VARIABLE continuous

- `c + geom_area(stat = "bin")` x, y, alpha, color, fill, group, linetype, size
- `c + geom_density(kernel = "gaussian")` x, y, alpha, color, fill, group, linetype, size
- `c + geom_dotplot(binaxis = "y", stackdir = "center")` x, y, alpha, color, fill, group, linetype, size
- `f + geom_dotplot(binaxis = "area")` x, y, alpha, color, fill, group, linetype, size

discrete

- `d + geom_bar()` x, alpha, color, fill, linetype, size, weight

TWO VARIABLES

continuous x, continuous y

- `e + geom_label(aes(label = cyl, nudge_x = 1, nudge_y = 1, check_overlap = TRUE))` x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust
- `e + geom_jitter(height = 2, width = 2)` x, y, alpha, color, fill, shape, size
- `e + geom_point()` x, y, alpha, color, fill, shape, size, stroke
- `e + geom_quantile()` x, y, alpha, color, group, linetype, size, weight
- `e + geom_rug(sides = "bl")` x, y, alpha, color, linetype, size
- `e + geom_smooth(method = lm)` x, y, alpha, color, fill, group, linetype, size, weight
- `e + geom_text(aes(label = cyl, nudge_x = 1, nudge_y = 1, check_overlap = TRUE))` x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

discrete x, continuous y

- `f + geom_col()` x, y, alpha, color, fill, group, linetype, size
- `f + geom_boxplot(x, y, lower, middle, upper, ymax, ymin, alpha, color, fill, group, linetype, size, weight)`
- `f + geom_dotplot(binaxis = "center")` x, y, alpha, color, fill, group, linetype, size
- `f + geom_violin(scale = "area")` x, y, alpha, color, fill, group, linetype, size, weight

discrete x, discrete y

- `g + geom_count()` x, y, alpha, color, fill, shape, size, stroke

maps

- `d <- data.frame(murder = USArrests$Murder, state = tolower(trownames(USArrests)))`
- `map <- map_data("state")`
- `k <- ggplot(data, aes(fill = murder))`
- `k + geom_map(aes(map_id = state), map = map) + expand_limits(x = map$long, y = map$lat), map_id, alpha, color, fill, group, linetype, size`
- `l + geom_contour(aes(z = z))` x, y, z, alpha, colour, group, linetype, size, weight
- `l + geom_raster(aes(fill = z))` x, y, alpha, fill, interpolate = FALSE
- `l + geom_tile(aes(fill = z))` x, y, alpha, color, fill, linetype, size, width

RStudio® is a trademark of RStudio, Inc. • CC BY SA RStudio • info@rstudio.com • 844-448-1212 • rstudio.com • Learn more at: <http://ggplot2.tidyverse.org> • ggplot2 2.1.0 • Updated: 2016-11

2. Principais pacotes para gráficos

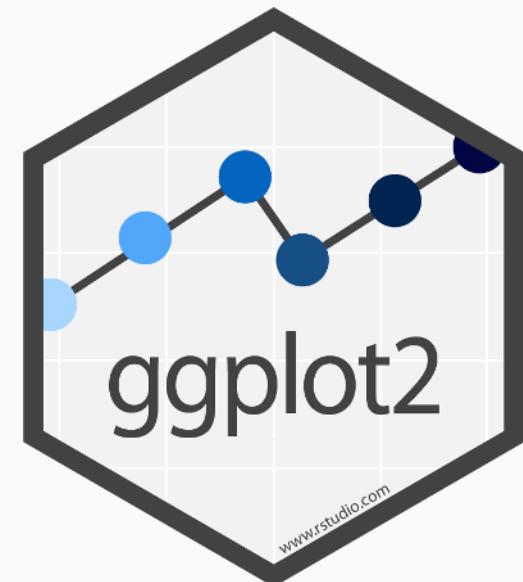
ggplot2

Integrado ao tidyverse, possui uma sintaxe própria

Necessita de funções específicas para objetos de **classes diferentes**

Estruturado dessa forma:

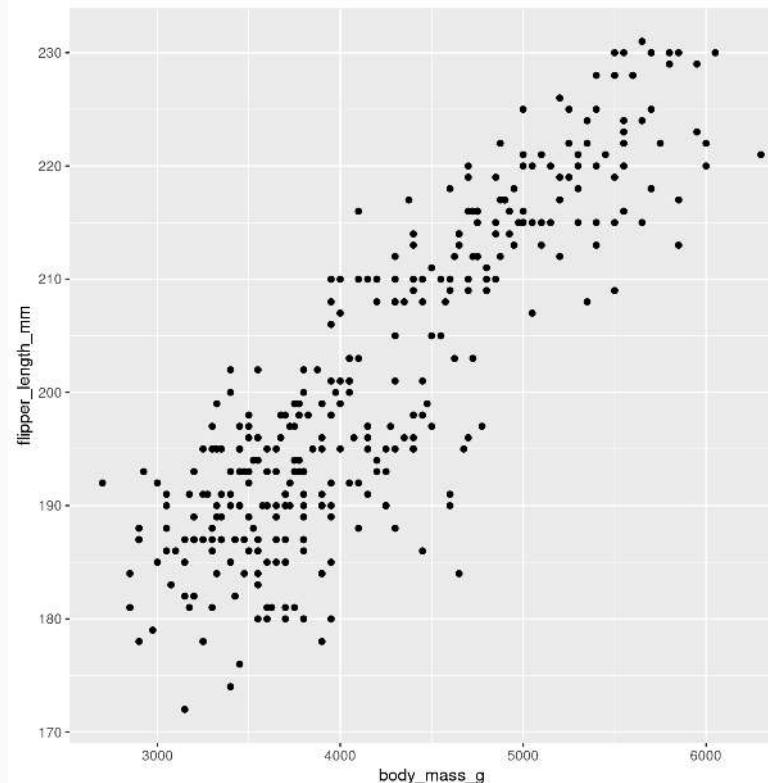
```
ggplot() +  
aes() +  
geom_() +  
facet_() +  
stats_() +  
coord_() +  
theme_()
```



2. Principais pacotes para gráficos

ggplot2

```
# ggplot2
library(ggplot2)
ggplot(data = penguins) + aes(x = body_mass_g, y = flipper_length_mm) + geom_point()
```





2. Principais pacotes para gráficos

ggpubr

Funções fáceis de usar para criar e personalizar plots para publicações baseadas no "ggplot2"

Necessita de funções específicas para gerar **gráficos específicos**

Funções:

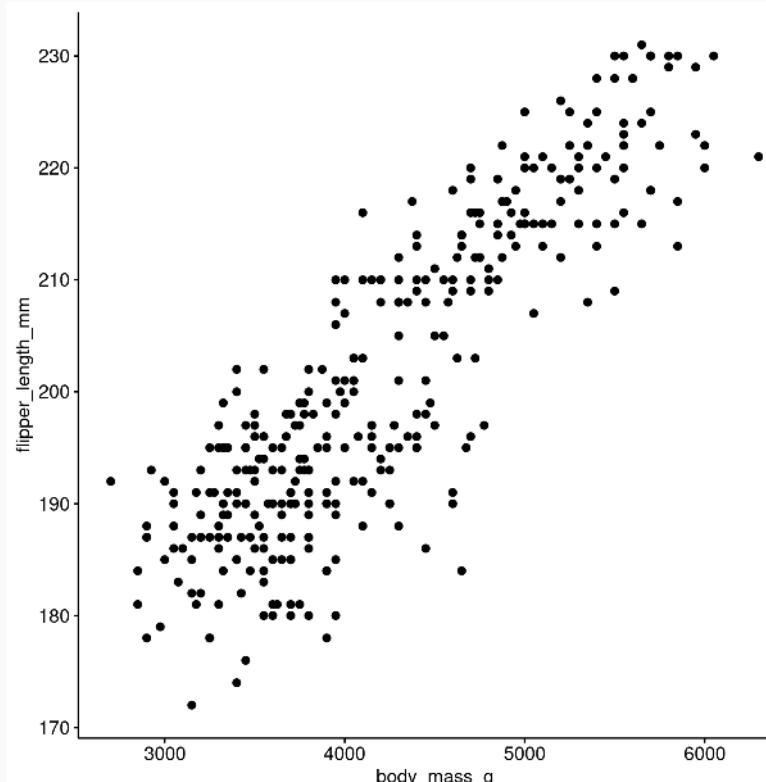
```
gghistogram()  
ggdensity()  
ggboxplot()  
ggviolin()  
ggbarplot()  
ggscatter()
```



2. Principais pacotes para gráficos

ggpubr

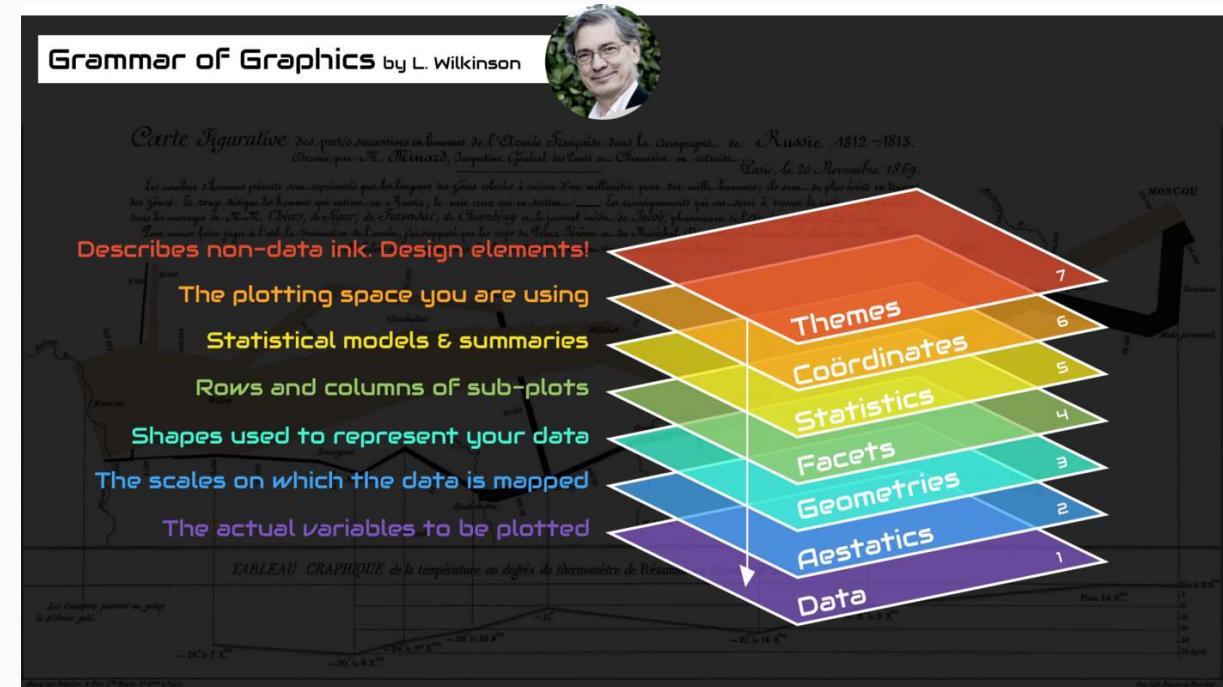
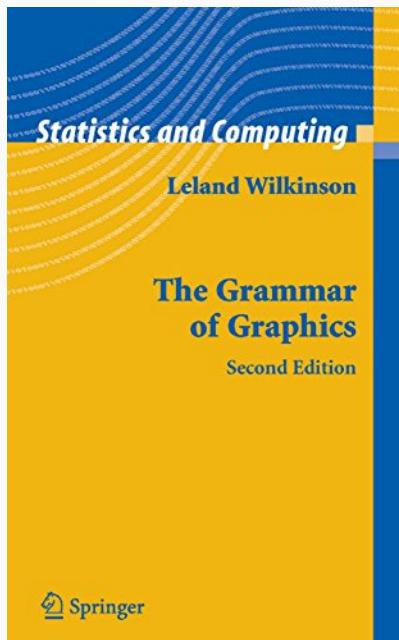
```
# ggpublisher
library(ggpubr)
ggscatter(penguins, x = "body_mass_g", y = "flipper_length_mm")
```



3. Gramática dos gráficos

Descrição

Wilkinson (2005) *Grammar of Graphics*: representação gráfica dos dados a partir de atributos estéticos (do inglês *aesthetic*)



3. Gramática dos gráficos

Descrição

Grammar of Graphics



xy, 3902, 29, 9,
4756, x, 72, 633,
647, 617, 827, 3,
1, 21, 45, tyu, 6,
987, 457, 283, 8,
4, 5, 671, 34, 67,
x, 981, hu, 89, 5

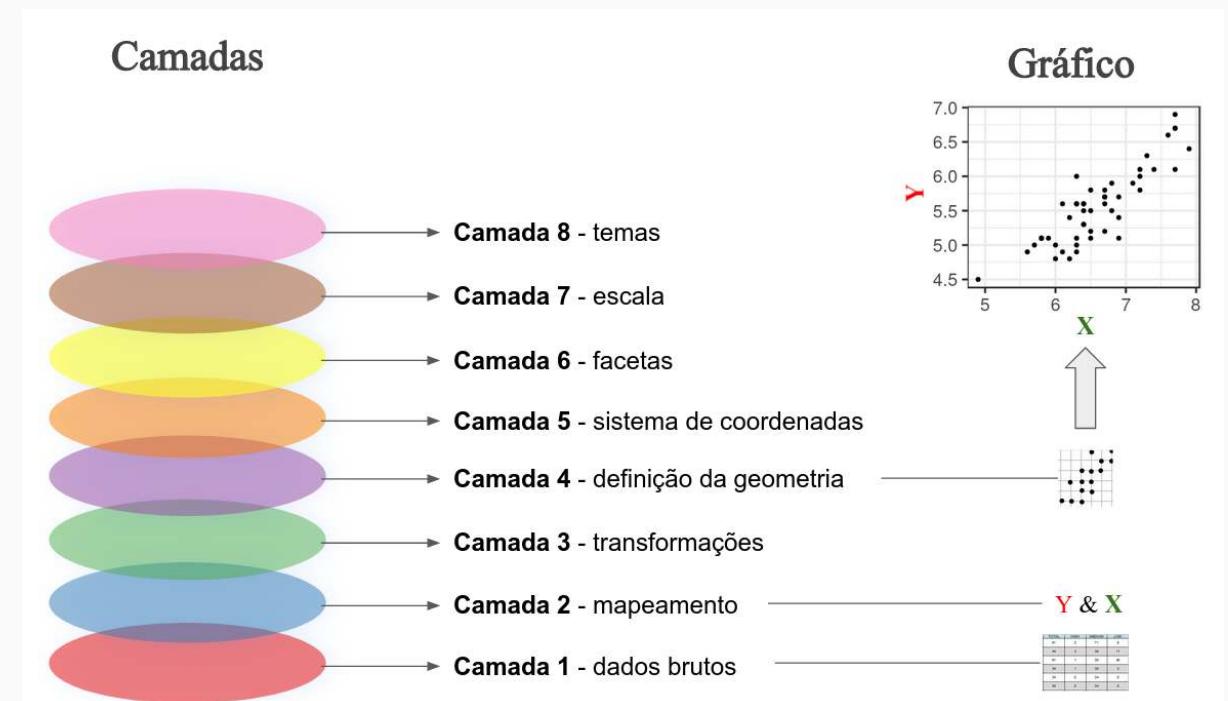
```
32 d <- ggplot(data) # Planning your 'Data' layer
33
34 # Before plotting (if not installed) install.packages("gridExtra")
35 # Then run library(gridExtra)
36 library(ggplot2)
37
38 # Create new variable for plot only x and y axis. (data" and "statistics" layer)
39 plot <- ggplot(data=new_data, aes(x=Genre, y=Gross...$G))
40
41 # Create new variables with ggplot2 layers
42 q <- plot + geom_boxplot(aes(fill=Studio, size=Budget...mill...),
43                           shape = 21, # this will shape a border around data points.
44                           colour = "black") # w with the border color of black.
45                           geom_boxplot(alpha=0.1, outlier.color = NA) # places the outlier on the data points
46
47 # Change axis and title if needed.
48 q <- q +
49   xlab("Genre") +
50   ylab("Gross") +
51   ggtitle("Domestic Gross N by Genre")
52
53 # Make your plot visually attractive and possibly with the 'Theme' function. (Theme layer)
54 q + theme(
55   axis.title = element_text(colour = "blue", size = 14),
56   axis.text = element_text(size = 12),
57   legend.title = element_text(size = 12),
58   legend.text = element_text(size = 12),
59   plot.title = element_text(size = 14, hjust = 0.5), # 'hjust' will center your text,
60   panel.background = element_rect(fill = "#E0E0E0"))
```

[Think About the Grammar of Graphics When Improving Your Graphs](#)

3. Gramática dos gráficos

Descrição

Wickham (2008) criou o pacote **ggplot2**, onde “gg” representa a contração de Grammar of Graphics



4. Principal material de estudo

Livros

O idealizador e mantenedor do pacote **ggplot2** foi de novo o **Hadley Wickham**

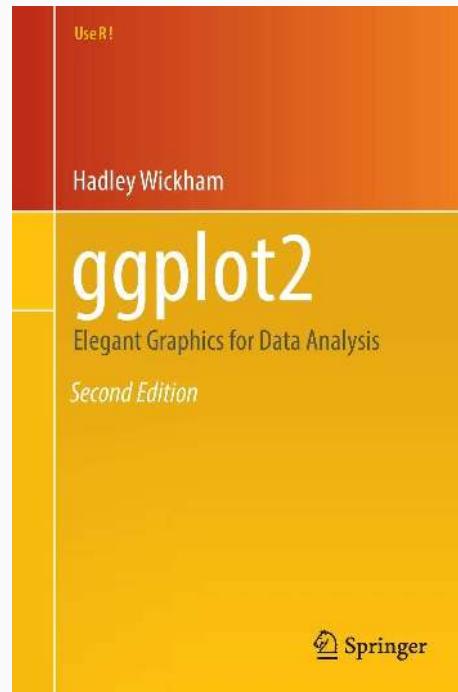


[Hadley Wickham](#)

4. Principal material de estudo

Livros

ggplot2 (2009, 2016)

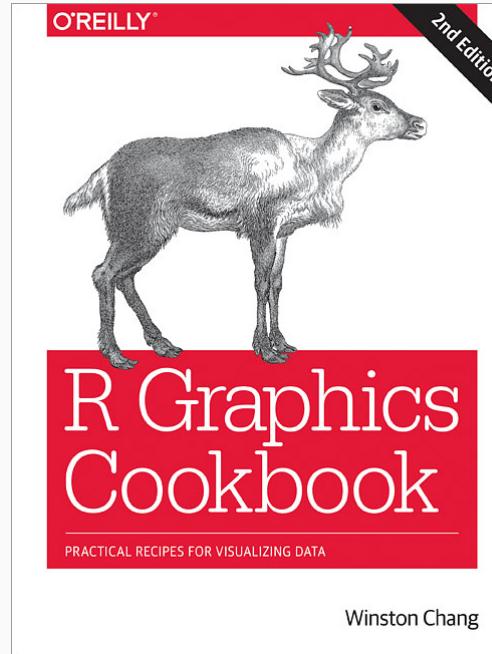


[Wickham \(2009, 2016\)](#)

4. Principal material de estudo

Livros

R Graphics Cookbook (2018)

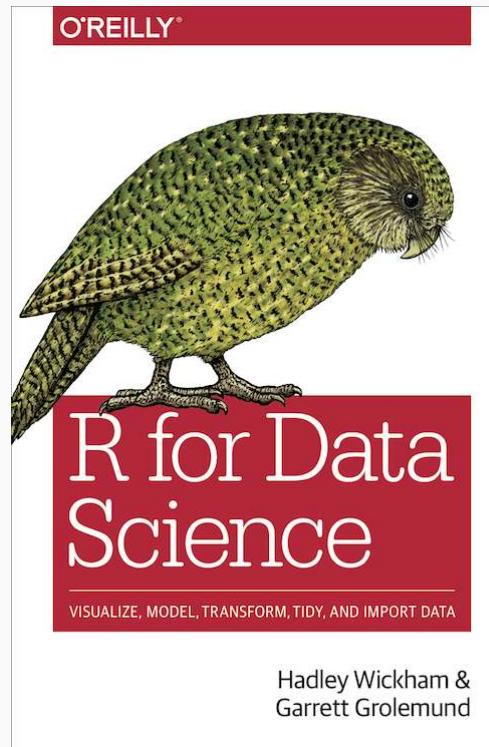


[Chang \(2018\)](#)

4. Principal material de estudo

Livros

R for Data Science (2017)

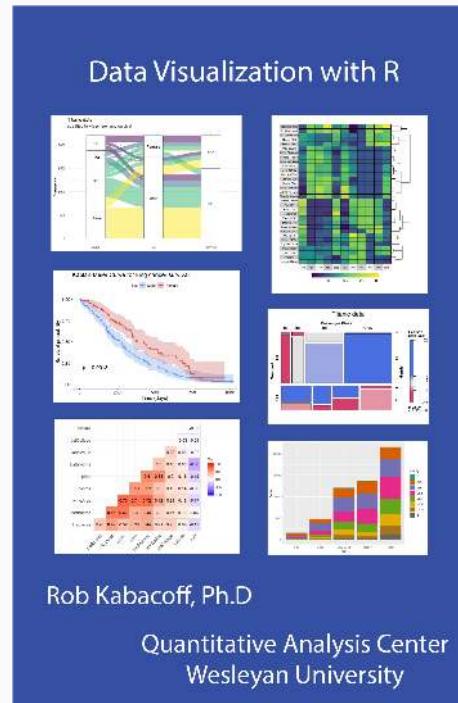


[Wickham & Grolemund \(2017\)](#)

4. Principal material de estudo

Livros

Data Visualization with R (2018)

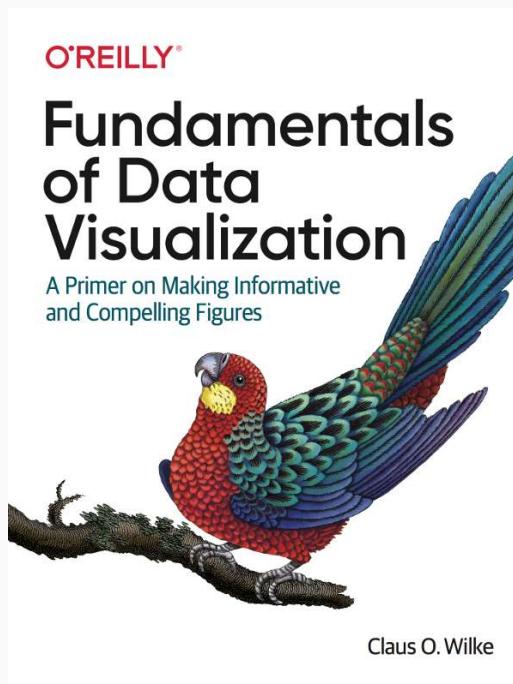


[Kabacoff \(2018\)](#)

4. Principal material de estudo

Livros

Fundamentals of Data Visualization (2019)

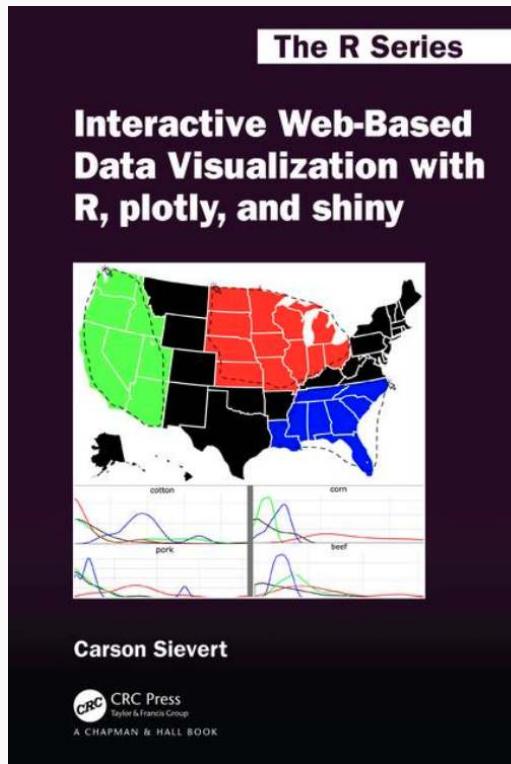


[Wilke \(2019\)](#)

4. Principal material de estudo

Livros

Interactive web-based data visualization with R, plotly, and shiny (2019)

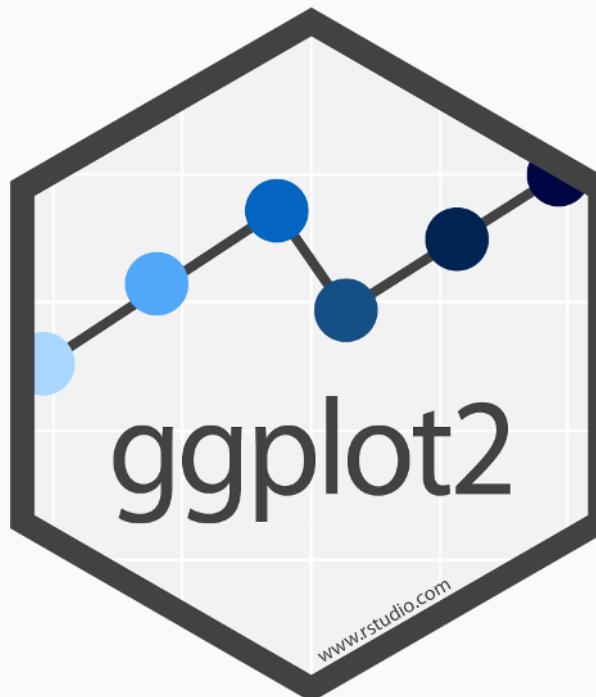


[Sievert \(2019\)](#)

4. Principal material de estudo

Sites

ggplot2: Reference



[ggplot2 reference](#)

4. Principal material de estudo

Sites

R Graph Gallery



[R Graph Gallery](#)

4. Principal material de estudo

Sites

from Data to Viz



from Data to Viz

4. Principal material de estudo

Sites

Statistical tools for high-throughput data analysis (STHDA)



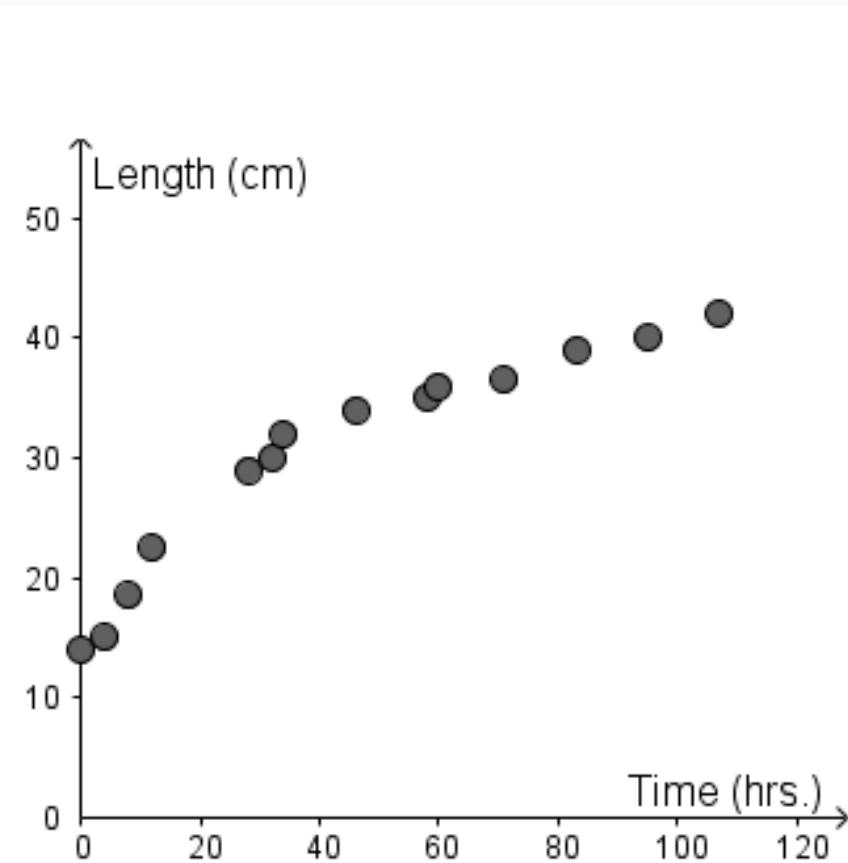
Dúvidas?

Os elementos de um gráfico são representações das colunas (eixos) e linhas (elementos) de nossas matrizes de dados

5. Principais tipos de gráficos

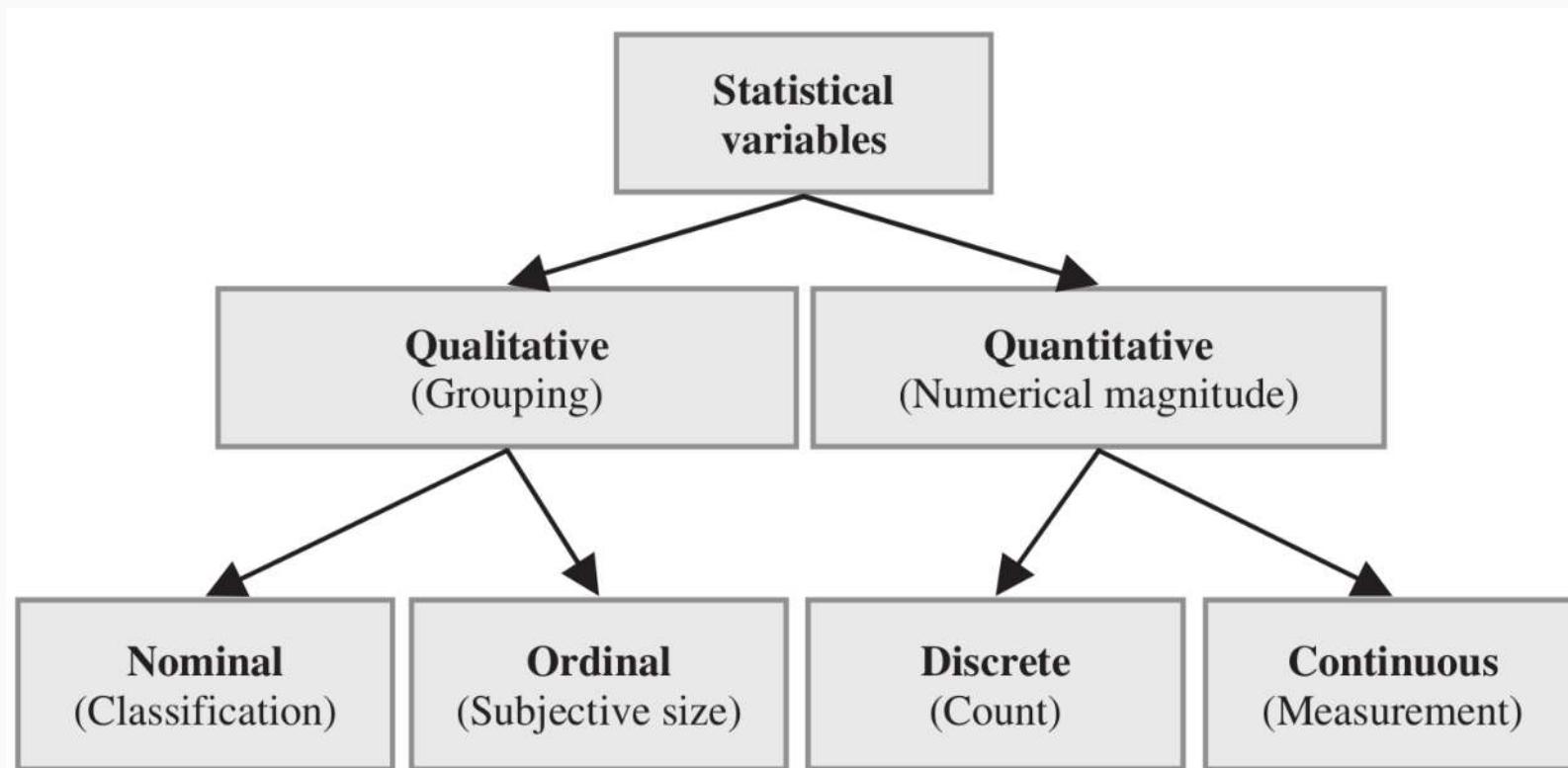
Colunas se tornam eixos e valores a representação

Amount of time from initial measurement (hrs.)	Length
0	14
4	15
8	18.5
12	22.5
28	29
32	30
34	32
46	34
58	35
60	36
71	36.5
83	39
95	40
107	42



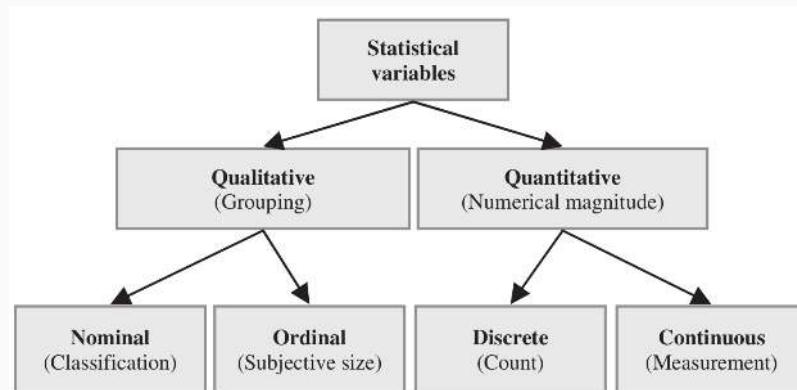
5. Principais tipos de gráficos

Tipos de variáveis



5. Principais tipos de gráficos

Tipos de variáveis

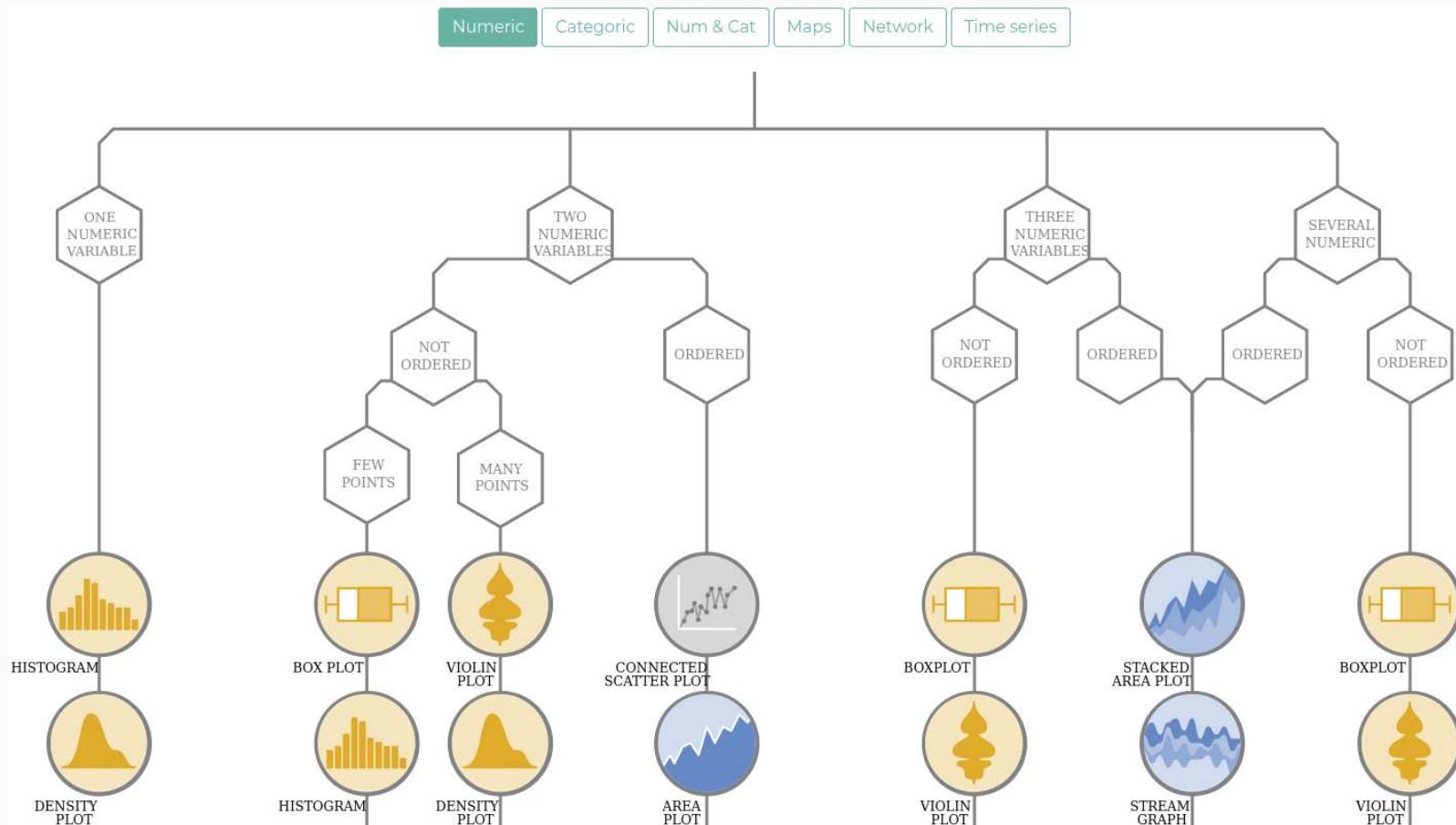


[Matthiopoulos \(2011\)](#), [@allison_horst](#)

O **tipo e quantidade das variáveis** indicará o melhor tipo de gráfico para **representar** os dados

5. Principais tipos de gráficos

from Data to Viz



from Data to Viz

Dúvidas?

E que dados vamos usar?

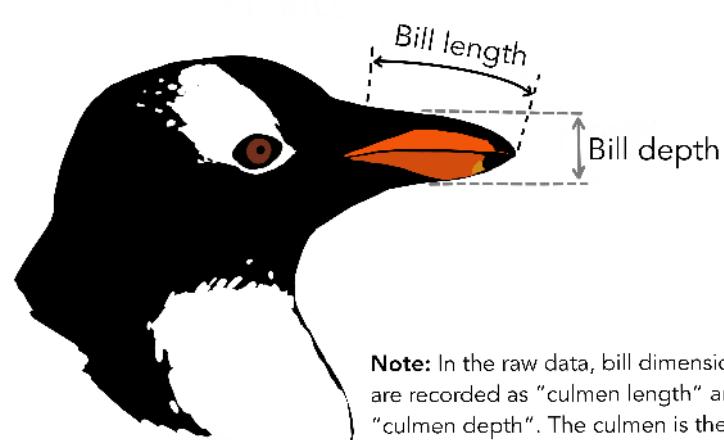


[palmerpenguins](https://palmerpenguins.com)

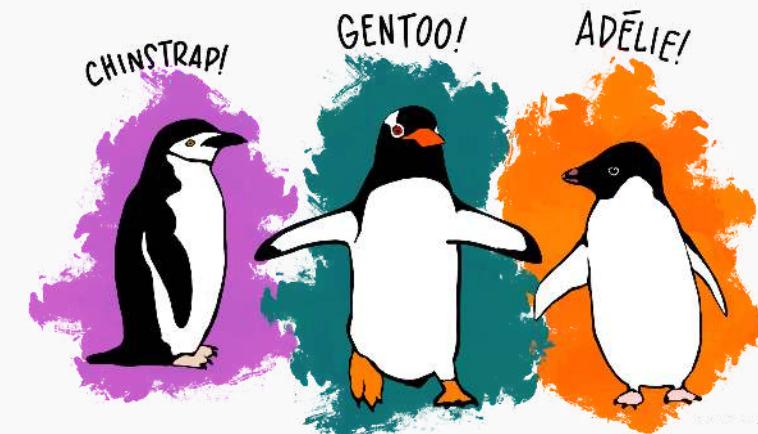
palmerpenguins

Dados de medidas de pinguins chamados palmerpenguins

```
# carregar  
library(palmerpenguins)  
  
# visualizar os dados  
penguins  
  
# glimpse  
tibble::glimpse(penguins)
```



[palmerpenguins](#)

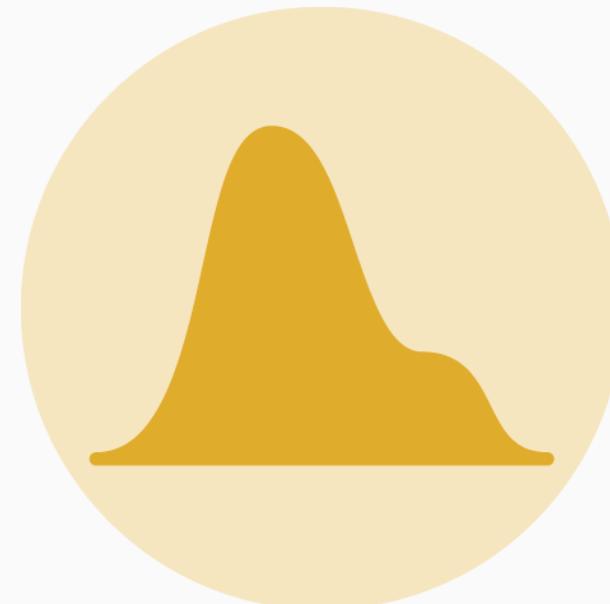


6. Histograma (*Histogram*)

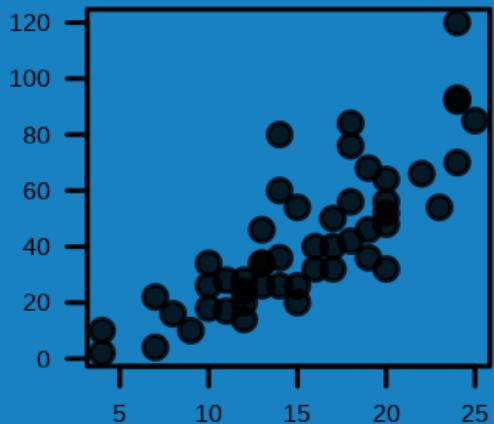
Representa os dados de: uma coluna

Tipo de dado: discreto ou contínuo

Distribuição de frequência e densidade de dados discretos ou contínuos



graphics

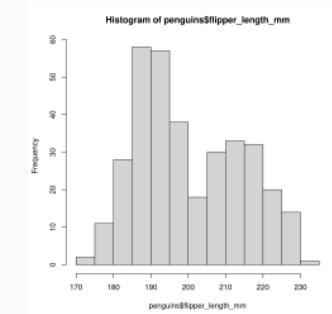


[graphics](#)

6. Histograma (*Histogram*)

graphics

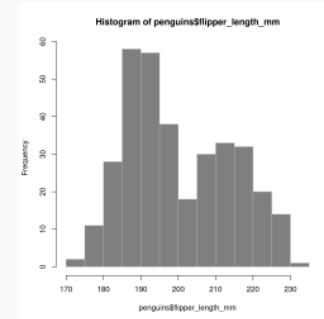
```
hist(penguins$flipper_length_mm)
```



6. Histograma (*Histogram*)

graphics

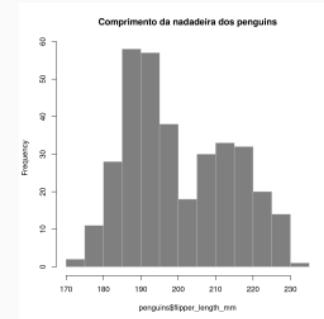
```
hist(penguins$flipper_length_mm,  
      col = "gray50",  
      border = "gray")
```



6. Histograma (*Histogram*)

graphics

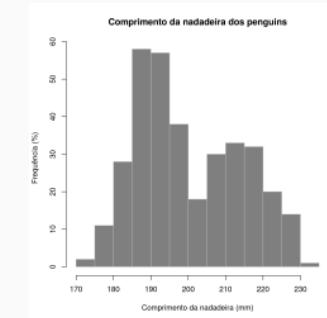
```
hist(penguins$flipper_length_mm,  
      col = "gray50",  
      border = "gray",  
      main = "Comprimento da nadadeira dos penguins")
```



6. Histograma (*Histogram*)

graphics

```
hist(penguins$flipper_length_mm,  
      col = "gray50",  
      border = "gray",  
      main = "Comprimento da nadadeira dos pinguins",  
      xlab = "Comprimento da nadadeira (mm)",  
      ylab = "Frequência (%)")
```



6. Histograma (*Histogram*)

graphics

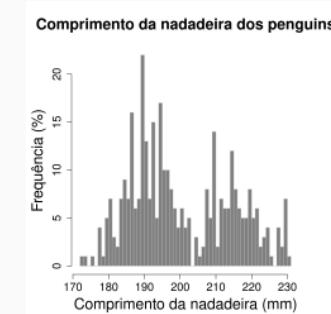
```
hist(penguins$flipper_length_mm,  
      col = "gray50",  
      border = "gray",  
      main = "Comprimento da nadadeira dos pinguins",  
      xlab = "Comprimento da nadadeira (mm)",  
      ylab = "Frequência (%)",  
      br = 50)
```



6. Histograma (*Histogram*)

graphics

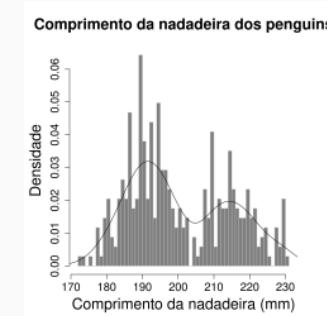
```
par(mar = c(5, 5, 5, 5))
hist(penguins$flipper_length_mm,
  col = "gray50",
  border = "gray",
  main = "Comprimento da nadadeira dos penguins",
  xlab = "Comprimento da nadadeira (mm)",
  ylab = "Frequência (%)",
  br = 50,
  cex.main = 2,
  cex.lab = 2,
  cex.axis = 1.5)
```



6. Densidade (*Density*)

graphics

```
par(mar = c(5, 5, 5, 5))
hist(penguins$flipper_length_mm,
  col = "gray50",
  border = "gray",
  main = "Comprimento da nadadeira dos penguins",
  xlab = "Comprimento da nadadeira (mm)",
  ylab = "Densidade",
  br = 50,
  cex.main = 2,
  cex.lab = 2,
  cex.axis = 1.5,
  prob = TRUE)
lines(density(na.omit(penguins$flipper_length_mm)))
```



6. Densidade (*Density*)

graphics

```
par(mar = c(5, 5, 5, 5))
plot(density(na.omit(penguins$flipper_length_mm)),
  col = "gray50",
  main = "Comprimento da nadadeira dos penguins",
  xlab = "Comprimento da nadadeira (mm)",
  ylab = "Densidade",
  cex.main = 2,
  cex.lab = 2,
  cex.axis = 1.5)
polygon(density(na.omit(penguins$flipper_length_mm))
  col = "gray50")
```



6. Densidade (*Density*)

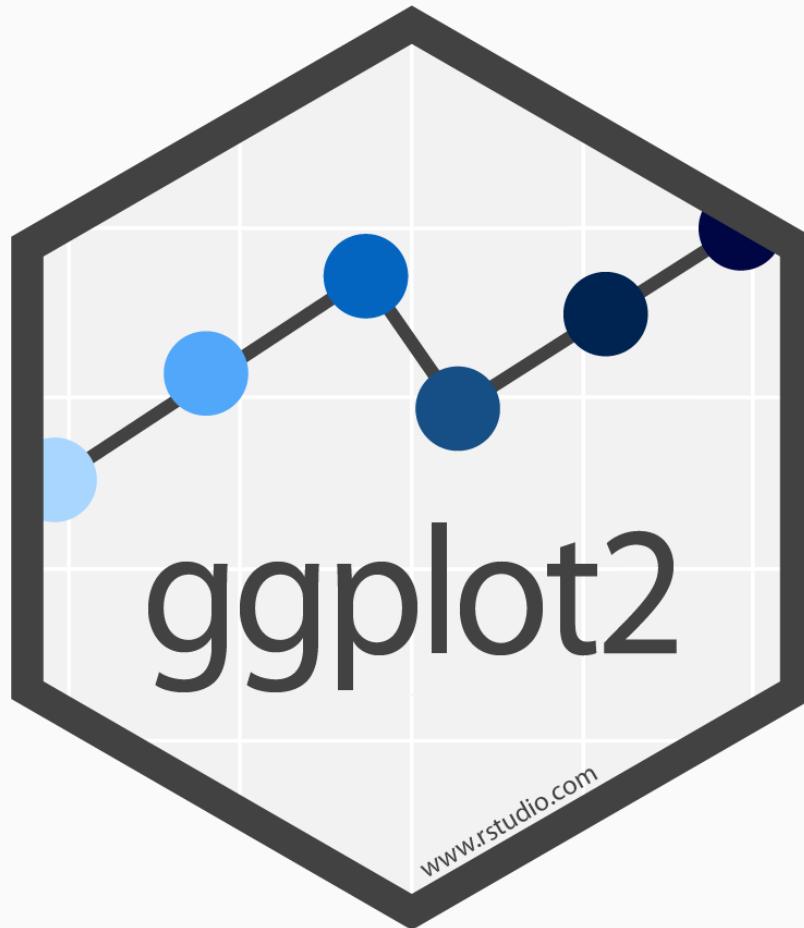
graphics

Exportar

```
png(here::here("03_dados", "graficos", "plot_densidade.png"),
    wi = 15, he = 15, un = "cm", res = 300)
```

```
par(mar = c(5, 5, 5, 5))
plot(density(na.omit(penguins$flipper_length_mm)),
    col = "gray50",
    main = "Comprimento da nadadeira dos penguins",
    xlab = "Comprimento da nadadeira (mm)",
    ylab = "Frequência (%)",
    cex.main = 2, cex.lab = 2, cex.axis = 1.5)
polygon(density(na.omit(penguins$flipper_length_mm)), col = "gray50")
```

```
dev.off()
```

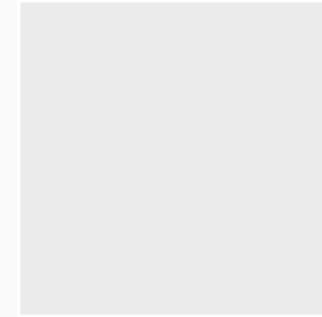


[ggplot2](#)

6. Histograma (*Histogram*)

ggplot2

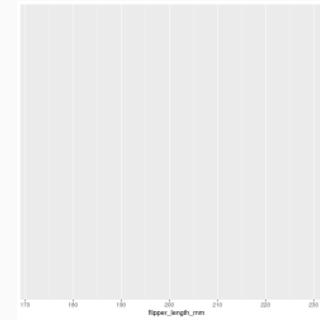
```
ggplot(data = penguins)
```



6. Histograma (*Histogram*)

ggplot2

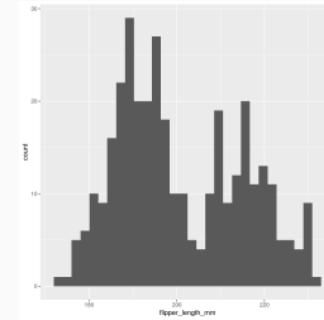
```
ggplot(data = penguins, aes(x = flipper_length_mm))
```



6. Histograma (*Histogram*)

ggplot2

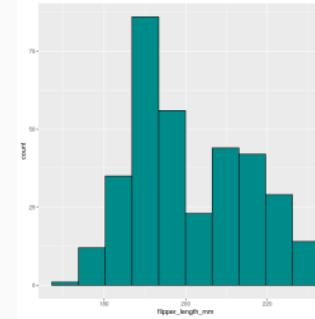
```
ggplot(data = penguins, aes(x = flipper_length_mm)) +  
  geom_histogram()
```



6. Histograma (*Histogram*)

ggplot2

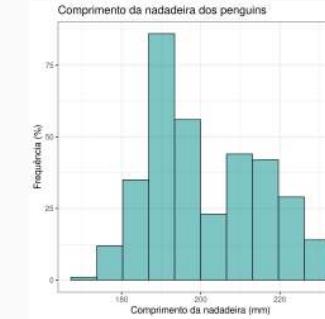
```
ggplot(data = penguins, aes(x = flipper_length_mm)) +  
  geom_histogram(color = "black",  
                 fill = "cyan4",  
                 bins = 10)
```



6. Histograma (*Histogram*)

ggplot2

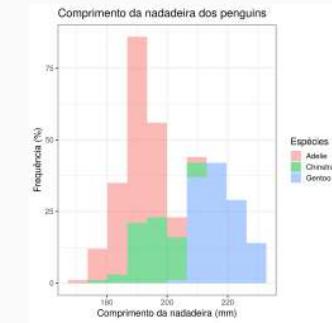
```
ggplot(data = penguins, aes(x = flipper_length_mm)) +  
  geom_histogram(color = "black",  
                 fill = "cyan4",  
                 bins = 10,  
                 alpha = .5) +  
  labs(title = "Comprimento da nadadeira dos pinguins",  
       x = "Comprimento da nadadeira (mm)",  
       y = "Frequência (%))") +  
  theme_bw(base_size = 15)
```



6. Histograma (*Histogram*)

ggplot2

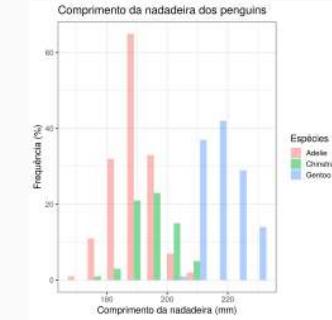
```
ggplot(data = penguins,
       aes(x = flipper_length_mm, fill = species)) +
  geom_histogram(bins = 10, alpha = .5) +
  labs(title = "Comprimento da nadadeira dos pinguins",
       fill = "Espécies",
       x = "Comprimento da nadadeira (mm)",
       y = "Frequência (%)") +
  theme_bw(base_size = 15)
```



6. Histograma (*Histogram*)

ggplot2

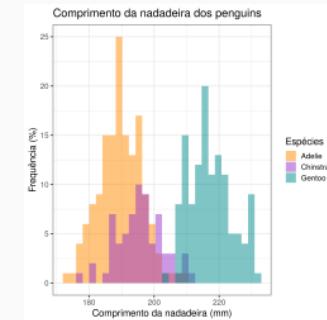
```
ggplot(data = penguins,
       aes(x = flipper_length_mm, fill = species)) +
  geom_histogram(bins = 10, alpha = .5,
                 position = "dodge") +
  labs(title = "Comprimento da nadadeira dos pinguins",
       fill = "Espécies",
       x = "Comprimento da nadadeira (mm)",
       y = "Frequência (%)") +
  theme_bw(base_size = 15)
```



6. Histograma (*Histogram*)

ggplot2

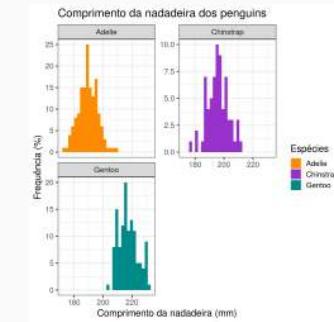
```
ggplot(data = penguins,  
       aes(x = flipper_length_mm, fill = species)) +  
  geom_histogram(alpha = .5, position = "identity") +  
  scale_fill_manual(values = c("darkorange",  
                               "darkorchid", "cyan4")) +  
  labs(title = "Comprimento da nadadeira dos pinguins",  
       fill = "Espécies",  
       x = "Comprimento da nadadeira (mm)",  
       y = "Frequência (%))" ) +  
  theme_bw(base_size = 15)
```



6. Histograma (*Histogram*)

ggplot2

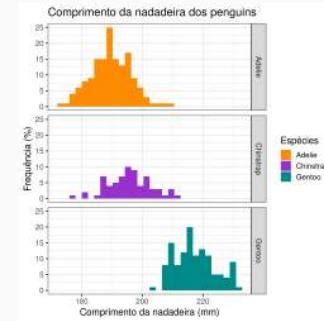
```
ggplot(data = penguins,
        aes(x = flipper_length_mm, fill = species)) +
  geom_histogram() +
  scale_fill_manual(values = c("darkorange",
                               "darkorchid", "cyan4"))
  facet_wrap(~ species, ncol = 2, scale = "free_y") +
  labs(title = "Comprimento da nadadeira dos pinguins",
       fill = "Espécies",
       x = "Comprimento da nadadeira (mm)",
       y = "Frequência (%)") +
  theme_bw(base_size = 15)
```



6. Histograma (*Histogram*)

ggplot2

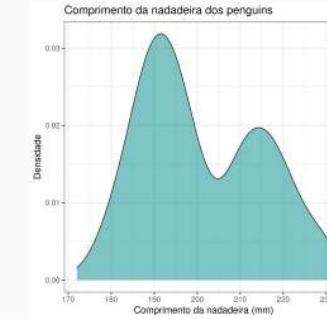
```
ggplot(data = penguins,
       aes(x = flipper_length_mm, fill = species)) +
  geom_histogram() +
  scale_fill_manual(values = c("darkorange",
                               "darkorchid", "cyan4"))
  facet_grid(species ~ .) +
  labs(title = "Comprimento da nadadeira dos pinguins",
       fill = "Espécies",
       x = "Comprimento da nadadeira (mm)",
       y = "Frequência (%)") +
  theme_bw(base_size = 15)
```



6. Densidade (*Density*)

ggplot2

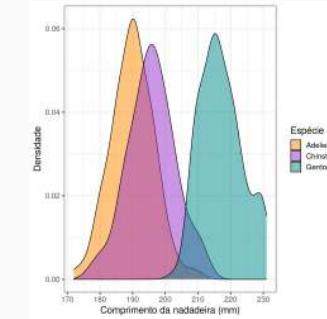
```
ggplot(data = penguins) +  
  aes(x = flipper_length_mm) +  
  geom_density(color = "black",  
               fill = "cyan4", alpha = .5) +  
  labs(title = "Comprimento da nadadeira dos pinguins",  
        x = "Comprimento da nadadeira (mm)",  
        y = "Densidade") +  
  theme_bw(base_size = 15)
```



6. Densidade (*Density*)

ggplot2

```
ggplot(data = penguins,
       aes(x = flipper_length_mm,
            fill = species)) +
  geom_density(alpha = .5) +
  scale_fill_manual(values = c("darkorange",
                               "darkorchid", "cyan4"))
  labs(x = "Comprimento da nadadeira (mm)",
       y = "Densidade",
       fill = "Espécie") +
  theme_bw(base_size = 15)
```



6. Histograma (*Histogram*)

ggplot2

Exportar

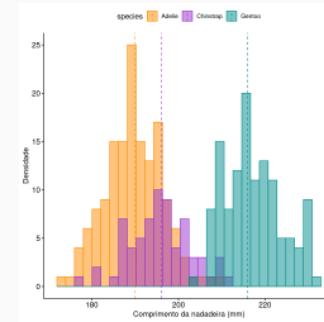
```
ggplot_densidade ← ggplot(data = penguins, aes(x = flipper_length_mm, fill = species)) +  
  geom_density(alpha = .5) +  
  scale_fill_manual(values = c("darkorange", "darkorchid", "cyan4")) +  
  labs(x = "Comprimento da nadadeira (mm)", y = "Densidade", fill = "Espécie") +  
  theme_bw(base_size = 15)  
  
ggsave(filename = here::here("03_dados", "graficos", "histogram_ggplot2.png"),  
       plot = ggplot_densidade, wi = 20, he = 15, un = "cm", dpi = 300)
```



6. Histograma (*Histogram*)

ggpubr

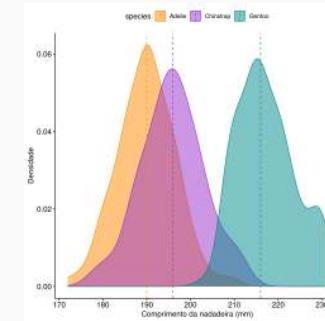
```
gghistogram(data = penguins,  
            x = "flipper_length_mm",  
            add = "median",  
            color = "species",  
            fill = "species",  
            palette = c("darkorange", "darkorchid",  
            xlab = "Comprimento da nadadeira (mm)",  
            ylab = "Densidade")
```



6. Densidade (*Density*)

ggpubr

```
ggdensity(data = penguins,  
          x = "flipper_length_mm",  
          add = "median",  
          color = "species",  
          fill = "species",  
          palette = c("darkorange", "darkorchid", "darktealgreen"),  
          xlab = "Comprimento da nadadeira (mm)",  
          ylab = "Densidade")
```



6. Densidade (*Density*)

ggpubr

Exportar

```
ggpubr_densidade ← ggdensity(data = penguins,
                               x = "flipper_length_mm",
                               add = "median",
                               fill = "cyan4",
                               rug = TRUE,
                               add_density = TRUE,
                               xlab = "Comprimento da nadadeira (mm)",
                               ylab = "Densidade")

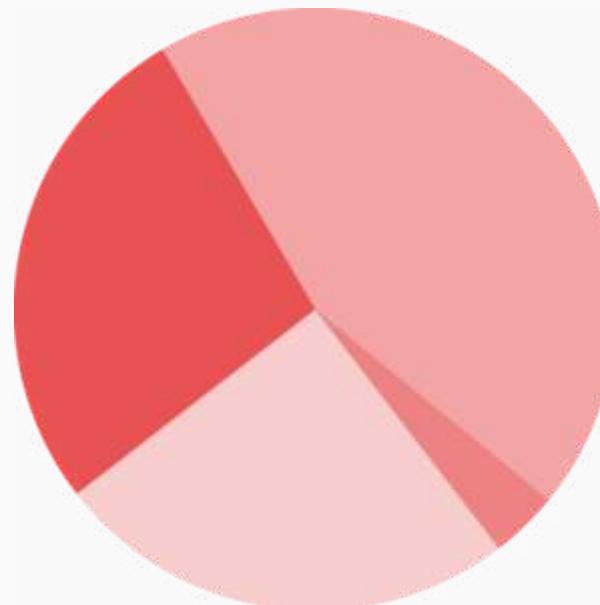
ggsave(filename = here::here("03_dados", "graficos", "densidade_ggpubr.png"),
       plot = ggpubr_densidade, wi = 20, he = 15, un = "cm", dpi = 300)
```

7. Gráfico de setores (*Pie chart*)

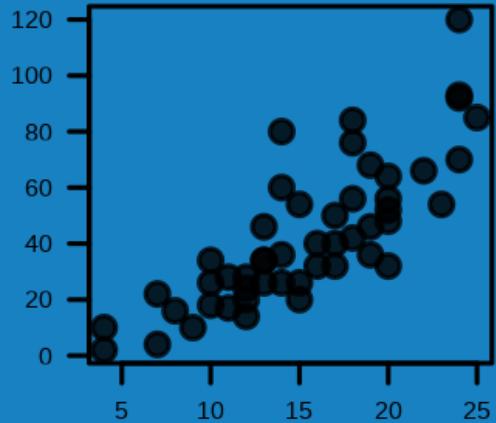
Representa os dados de: uma coluna

Tipo de dado: categórico

Proporção ou porcentagem de dados categóricos



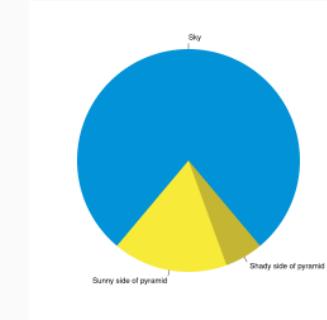
graphics



7. Gráfico de setores (*Pie chart*)

Clássico

```
par(mar = c(0, 1, 0, 1))
pie(c(280, 60, 20),
    c("Sky", "Sunny side of pyramid", "Shady side of
    col = c("#0292D8", "#F7EA39", "#C4B632"),
    init.angle = -50, border = NA)
```



7. Gráfico de setores (*Pie chart*)

Tabela de frequência

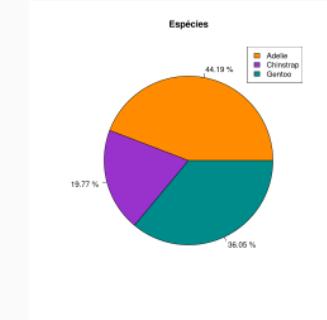
```
# calculo da proporcao
penguins_prop ← penguins %>%
  dplyr::count(species) %>%
  dplyr::mutate(prop = round(n/sum(n), 4)*100)
penguins_prop
```

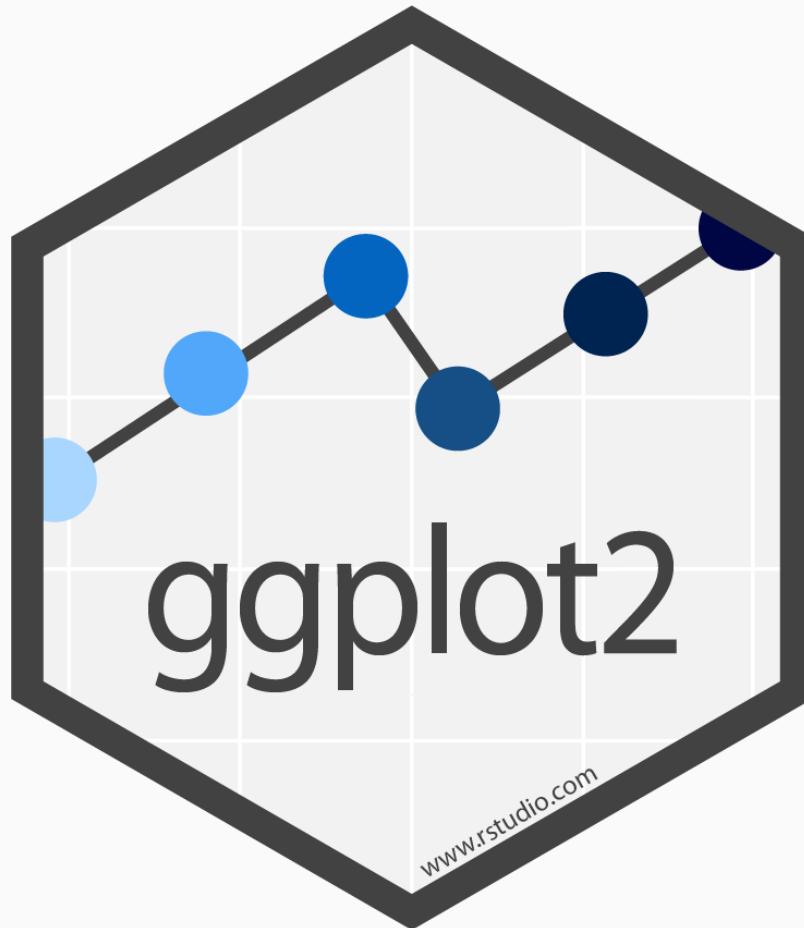
```
## # A tibble: 3 × 3
##   species     n   prop
##   <fct>   <int>   <dbl>
## 1 Adelie     152   44.2
## 2 Chinstrap   68   19.8
## 3 Gentoo    124   36.0
```

7. Gráfico de setores (*Pie chart*)

graphics

```
par(mar = c(5, 5, 5, 5))
pie(penguins_prop$prop,
    labels = paste(penguins_prop$prop, "%"),
    main = "Espécies",
    col = c("darkorange", "darkorchid", "cyan4"))
legend("topright", legend = penguins_prop$species,
       fill = c("darkorange", "darkorchid", "cyan4"))
```



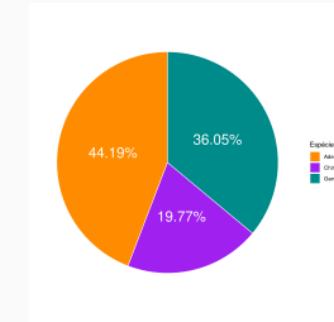


[ggplot2](#)

7. Gráfico de setores (*Pie chart*)

ggplot2

```
ggplot(data = penguins_prop,  
       aes(x = "", y = prop, fill = species)) +  
  geom_bar(stat = "identity", color = "white") +  
  coord_polar("y", start = 0) +  
  geom_text(aes(label = paste0(prop, "%")),  
            color = "white",  
            position = position_stack(vjust = .5),  
            size = 3)  
  scale_fill_manual(values = c("darkorange", "purple",  
                             "teal")) +  
  theme_void() +  
  labs(fill = "Espécie")
```

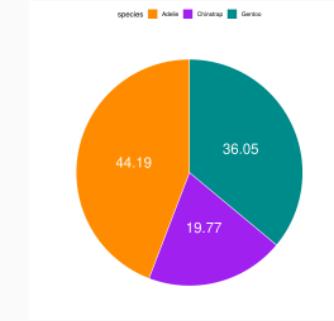




7. Gráfico de setores (*Pie chart*)

ggpubr

```
ggp pie(penguins_prop,  
        "prop",  
        label = "prop",  
        lab.pos = "in",  
        lab.font = c(8, "white"),  
        fill = "species",  
        color = "white",  
        palette = c("darkorange", "purple", "cyan4"))
```



Mas todos dizem para jogar o gráfico de pizza fora e
pedir uma pizza...

... e eu concordo...

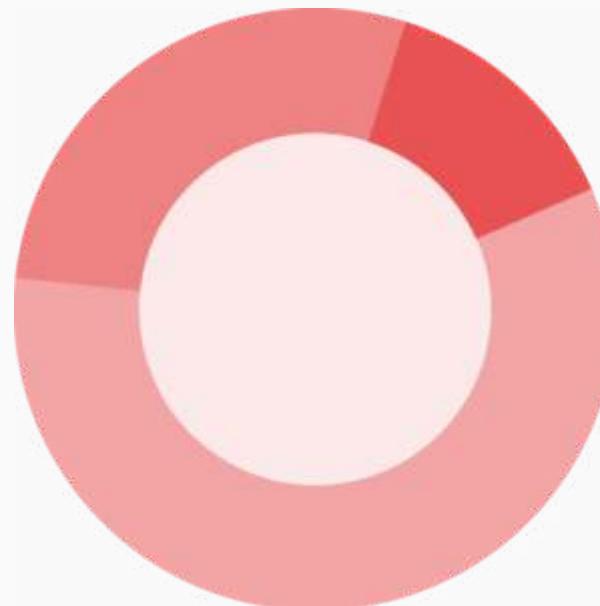
Então vamos usar algo mais saudável: vamos retirar o recheio da pizza...

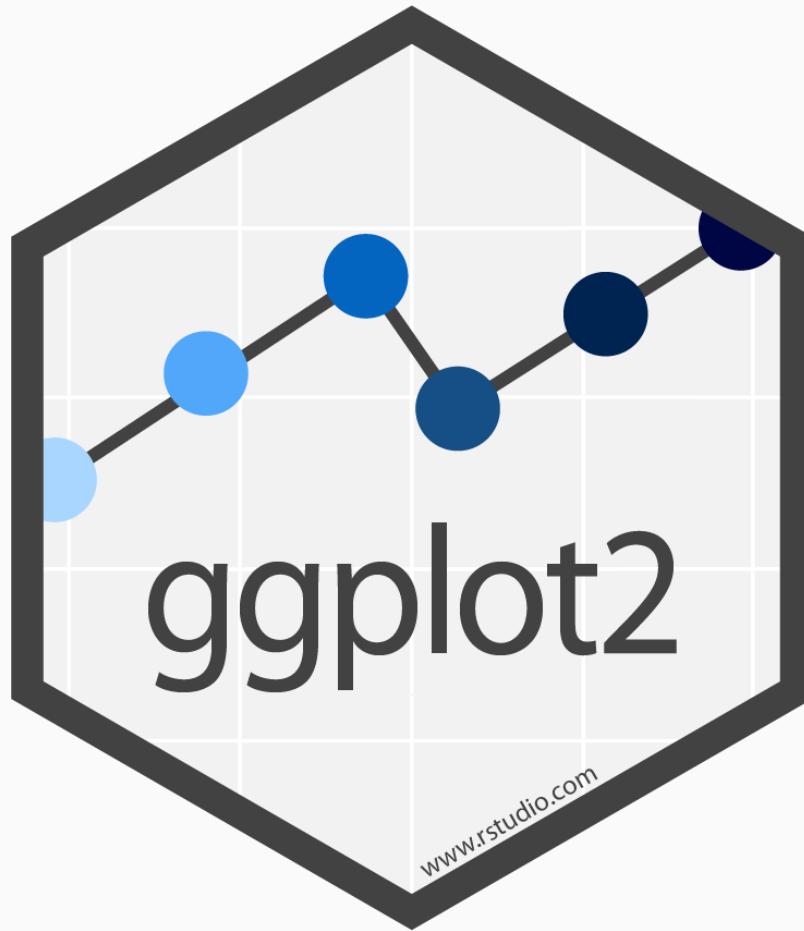
7. Gráfico de setores (*Donut chart*)

Representa os dados de: uma coluna

Tipo de dado: categórico

Proporção ou porcentagem de dados categóricos



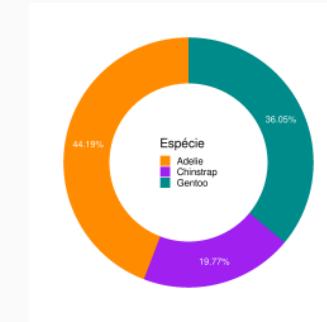


[ggplot2](#)

7. Gráfico de setores (*Donut chart*)

ggplot2

```
ggplot(data = penguins_prop,  
       aes(x = 2, y = prop, fill = species)) +  
  geom_bar(stat = "identity") +  
  coord_polar(theta = "y", start = 0) +  
  geom_text(aes(label = paste0(prop, "%")),  
            color = "white",  
            position = position_stack(vjust = .5),  
            size = 15) +  
  scale_fill_manual(values = c("darkorange", "purple",  
                             "teal")) +  
  xlim(0, 2.5) +  
  theme_void() +  
  theme(legend.position = c(.5, .5),  
        legend.title = element_text(size = 20),  
        legend.text = element_text(size = 15)) +  
  labs(fill = "Espécie")
```

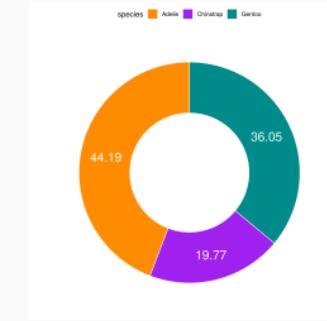




7. Gráfico de setores (*Donut chart*)

ggpubr

```
ggdonutchart(penguins_prop,  
             "prop",  
             label = "prop",  
             lab.pos = "in",  
             lab.font = c(7, "white"),  
             fill = "species",  
             color = "white",  
             palette = c("darkorange", "purple", "cyan"))
```



8. Gráfico de barras (*Bar plot*)

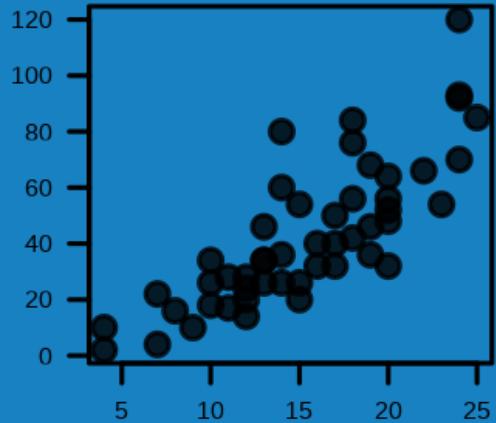
Representa os dados de: duas colunas

Modo das colunas: X = categórico e Y = categórico

Resume dados de contagens para uma coluna com diversos tipos de dados



graphics



8. Gráfico de barras (*Bar plot*)

Tabela de frequênciа

```
# numero de individuos coletados
penguins_count ← penguins %>%
  dplyr::count(species)
penguins_count
```

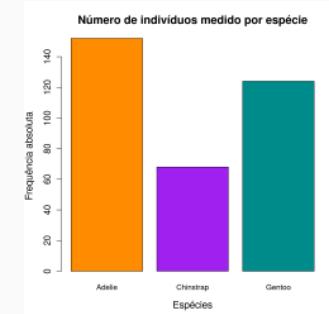


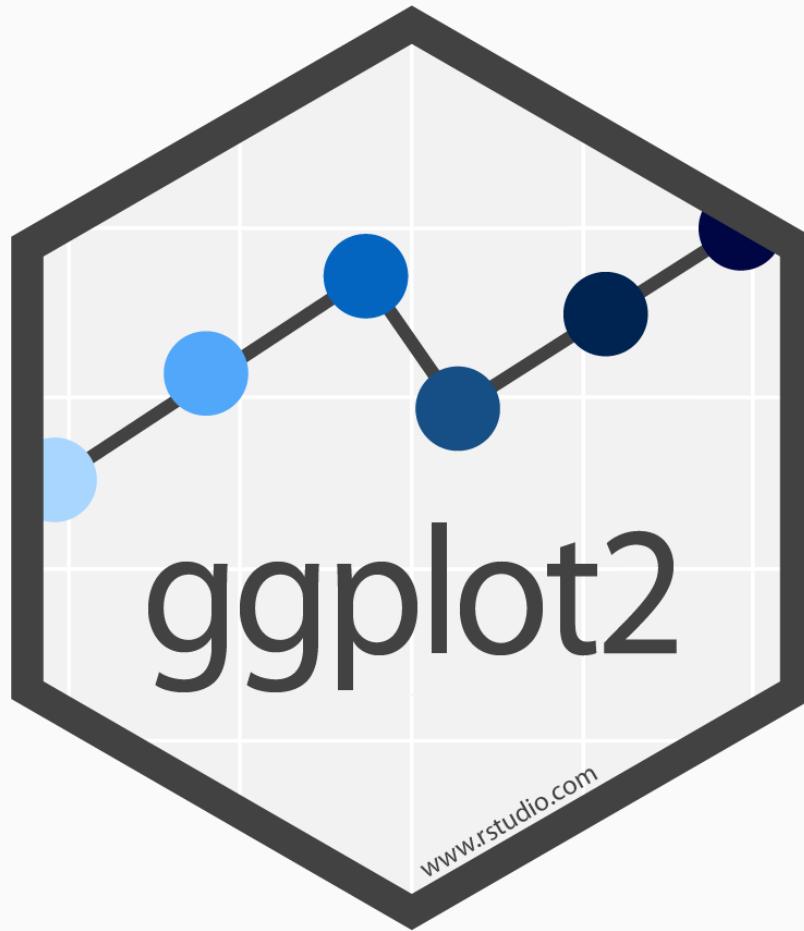
```
## # A tibble: 3 × 2
##   species     n
##   <fct>    <int>
## 1 Adelie     152
## 2 Chinstrap   68
## 3 Gentoo    124
```

8. Gráfico de barras (*Bar plot*)

graphics

```
barplot(n ~ species,
        data = penguins_count,
        col = c("darkorange", "purple", "cyan4"),
        main = "Número de indivíduos medido por espécie",
        xlab = "Espécies",
        ylab = "Frequência absoluta",
        cex.main = 1.5,
        cex.lab = 1.3,
        cex.axis = 1.2)
```

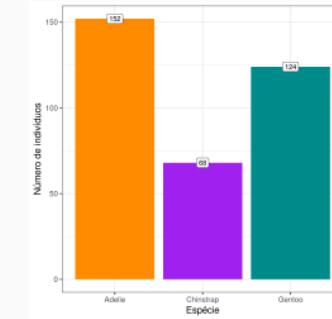




8. Gráfico de barras (*Bar plot*)

ggplot2

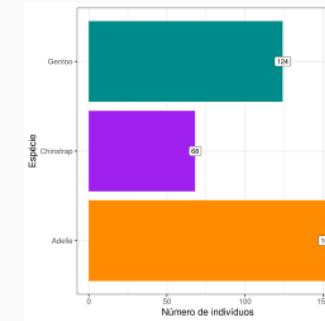
```
ggplot(data = penguins_count,  
       aes(x = species, y = n, fill = species)) +  
  geom_bar(stat = "identity") +  
  geom_label(aes(label = n), fill = "white") +  
  scale_fill_manual(values = c("darkorange", "purple")) +  
  theme_bw(base_size = 15) +  
  theme(legend.position = "none") +  
  labs(x = "Espécie",  
       y = "Número de indivíduos",  
       fill = "Espécie")
```



8. Gráfico de barras (*Bar plot*)

ggplot2

```
ggplot(data = penguins_count,  
       aes(x = species, y = n, fill = species)) +  
  geom_bar(stat = "identity") +  
  geom_label(aes(label = n), fill = "white") +  
  scale_fill_manual(values = c("darkorange", "purple")) +  
  coord_flip() +  
  theme_bw(base_size = 15) +  
  theme(legend.position = "none") +  
  labs(x = "Espécie",  
       y = "Número de indivíduos",  
       fill = "Espécie")
```

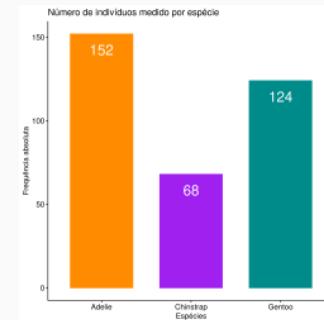




9. Gráfico de barras (*Bar plot*)

ggpubr

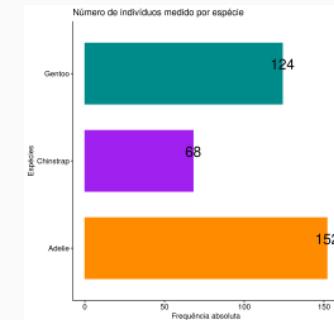
```
ggbarrplot(penguins_count,
  x = "species",
  y = "n",
  fill = "species",
  color = "species",
  palette = c("darkorange", "purple", "cyan4"),
  label = TRUE,
  lab.pos = "in",
  lab.col = "white",
  lab.size = 8,
  main = "Número de indivíduos medido por espécie",
  xlab = "Espécies",
  ylab = "Frequência absoluta",
  legend = "none")
```



9. Gráfico de barras (*Bar plot*)

ggpubr

```
ggbarrplot(penguins_count,
            x = "species",
            y = "n",
            fill = "species",
            color = "species",
            palette = c("darkorange", "purple", "cyan4"),
            label = TRUE,
            lab.pos = "out",
            lab.col = "black",
            lab.size = 8,
            main = "Número de indivíduos medido por espécie",
            xlab = "Espécies",
            ylab = "Frequência absoluta",
            legend = "none",
            orientation = "horiz")
```

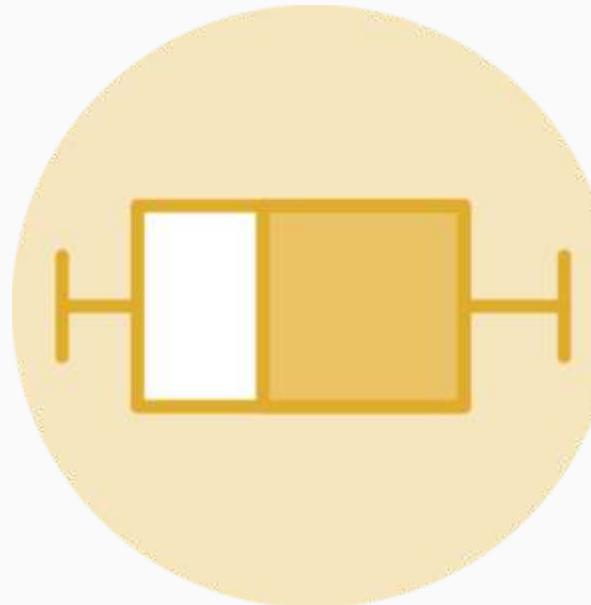


9. Gráfico de caixa (*Box plot*)

Representa os dados de: duas colunas

Modo das colunas: X = categórico e Y = contínuo

Resume informações de medidas contínuas para dois ou mais fatores categóricos

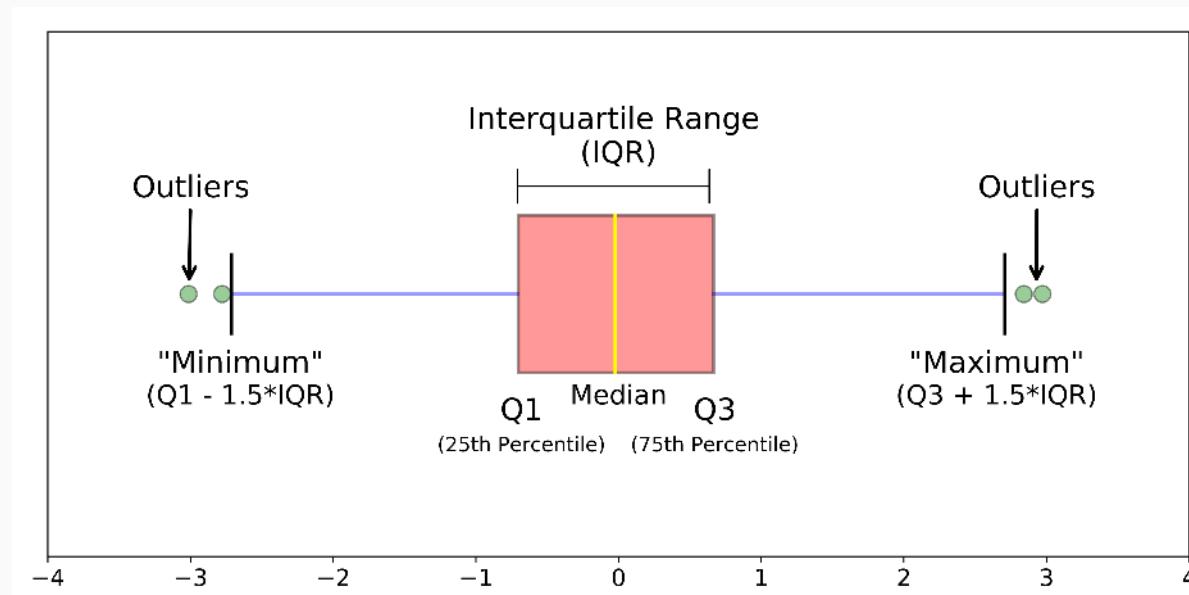


9. Gráfico de caixa (*Box plot*)

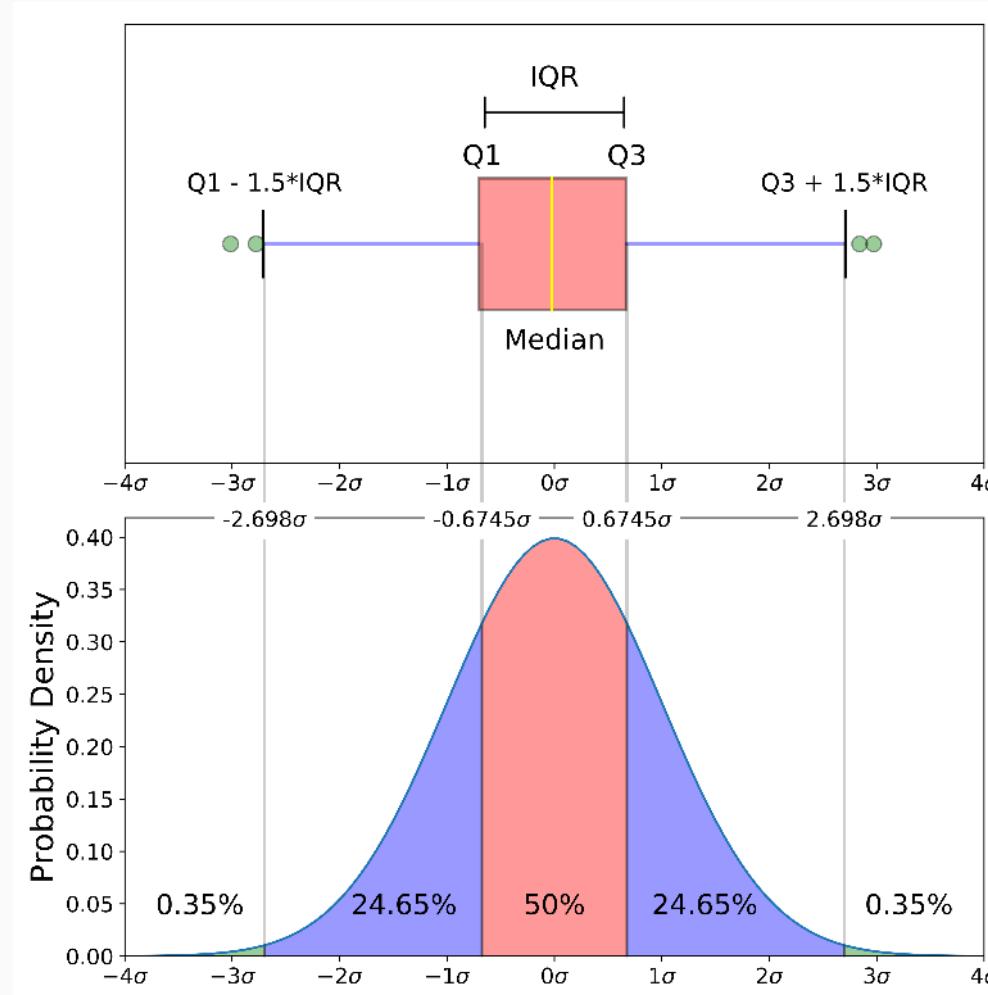
Intervalo inter-quartil (*interquartile range - IQR*)

Límite inferior e limite superior ($1.5 \times \text{IQR}$)

Valores exteriores (*outliers*)

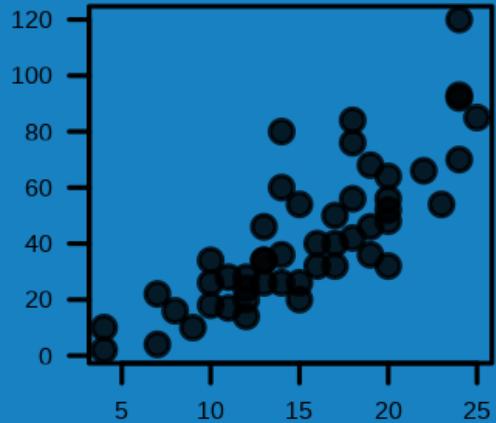


9. Gráfico de caixa (*Box plot*)



[Understanding Boxplots](#)

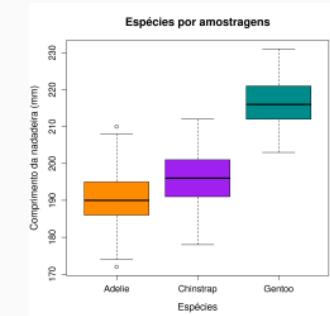
graphics

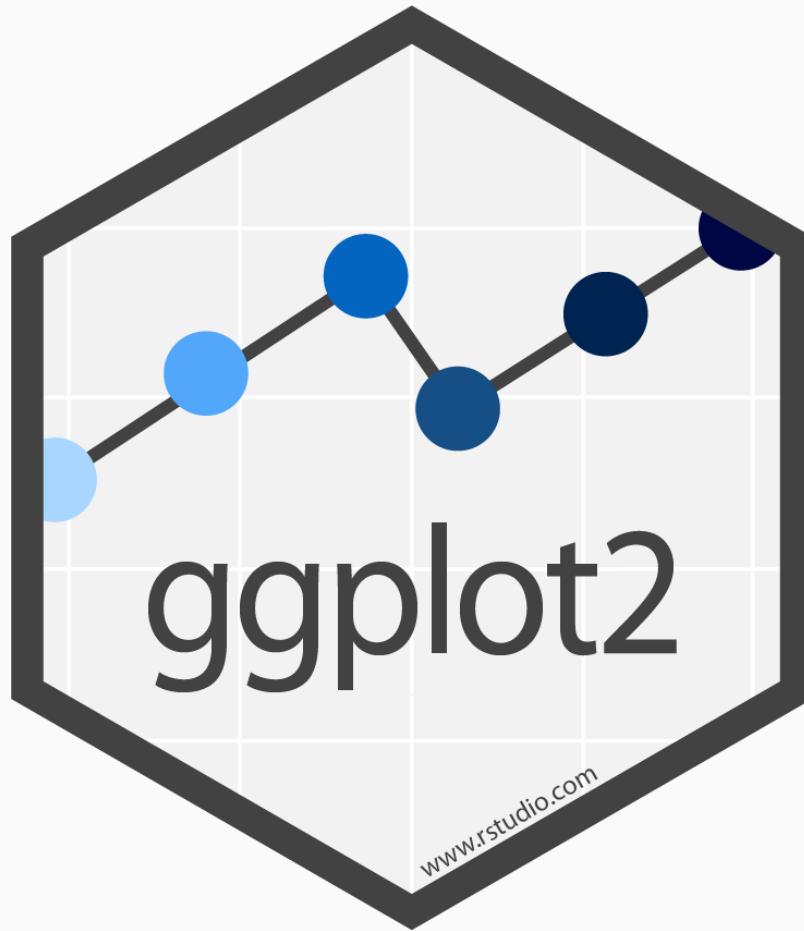


9. Gráfico de caixa (*Box plot*)

graphics

```
boxplot(flipper_length_mm ~ as.factor(species),  
        data = penguins,  
        col = c("darkorange", "purple", "cyan4"),  
        border = "black",  
        main = "Espécies por amostragens",  
        xlab = "Espécies",  
        ylab = "Comprimento da nadadeira (mm)",  
        cex.main = 1.5,  
        cex.lab = 1.3,  
        cex.axis = 1.2)
```

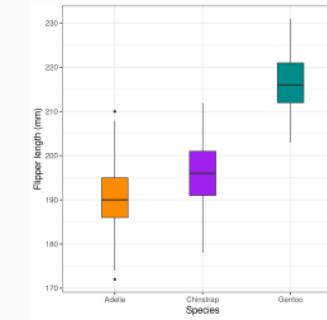




9. Gráfico de caixa (*Box plot*)

ggplot2

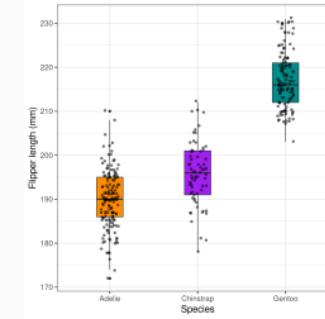
```
ggplot(data = penguins,  
       aes(x = species, y = flipper_length_mm, fill = species)) +  
  geom_boxplot(width = .3,  
               show.legend = FALSE) +  
  scale_fill_manual(values = c("darkorange", "purple", "teal")) +  
  theme_bw(base_size = 15) +  
  labs(x = "Species", y = "Flipper length (mm)")
```



9. Gráfico de caixa (Box plot)

ggplot2

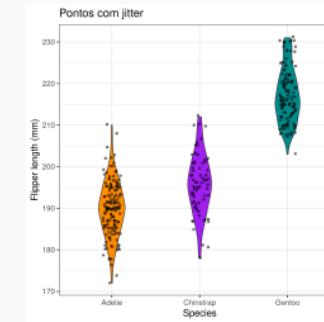
```
ggplot(data = penguins,  
       aes(x = species, y = flipper_length_mm, fill = species)) +  
  geom_boxplot(width = .3,  
               show.legend = FALSE) +  
  geom_jitter(alpha = .5,  
              show.legend = FALSE,  
              position = position_jitter(width = .1,  
              scale_fill_manual(values = c("darkorange", "purple", "teal")) +  
  theme_bw(base_size = 15) +  
  labs(x = "Species", y = "Flipper length (mm)")
```



10. Gráfico de caixa (*Violin plot*)

ggplot2

```
ggplot(data = penguins,  
       aes(x = species, y = flipper_length_mm, fill = species)) +  
  geom_violin(width = .3,  
              show.legend = FALSE) +  
  geom_jitter(alpha = .5,  
              show.legend = FALSE,  
              position = position_jitter(width = .1,  
              scale_fill_manual(values = c("darkorange", "purple", "teal")) +  
  theme_bw(base_size = 15) +  
  labs(title = "Pontos com jitter", x = "Species", y = "Flipper length (mm)")
```

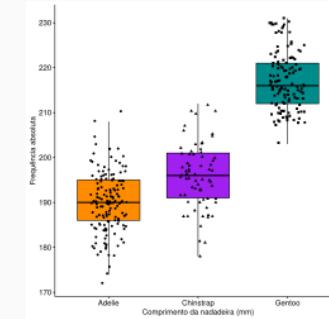




9. Gráfico de caixa (Box plot)

ggpubr

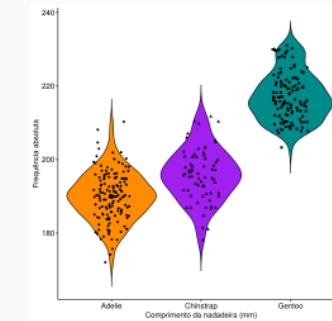
```
ggboxplot(data = penguins,
  x = "species",
  y = "flipper_length_mm",
  add = "jitter",
  shape = "species",
  fill = "species",
  color = "black",
  palette = c("darkorange", "purple", "cyan4",
  xlab = "Comprimento da nadadeira (mm)",
  ylab = "Frequência absoluta",
  legend = "none")
```



10. Gráfico de caixa (*Violin plot*)

ggpubr

```
ggviolin(data = penguins,
  x = "species",
  y = "flipper_length_mm",
  add = "jitter",
  shape = "species",
  fill = "species",
  color = "black",
  palette = c("darkorange", "purple", "cyan4"
xlab = "Comprimento da nadadeira (mm)",
ylab = "Frequência absoluta",
legend = "none")
```

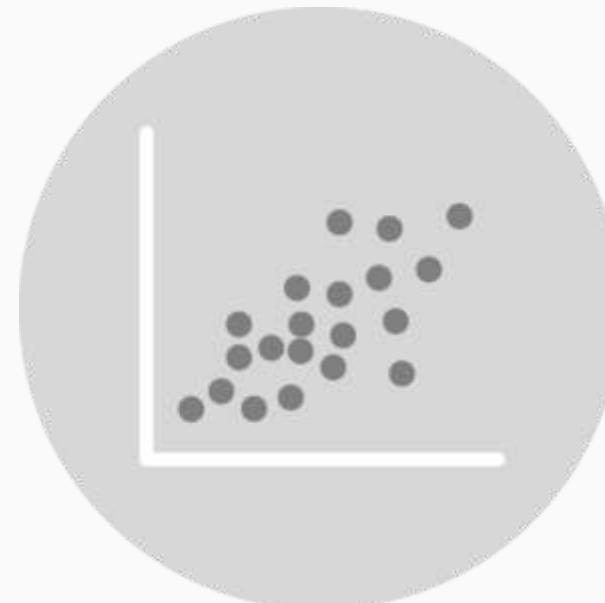


10. Gráfico de dispersão (*Scatter plot*)

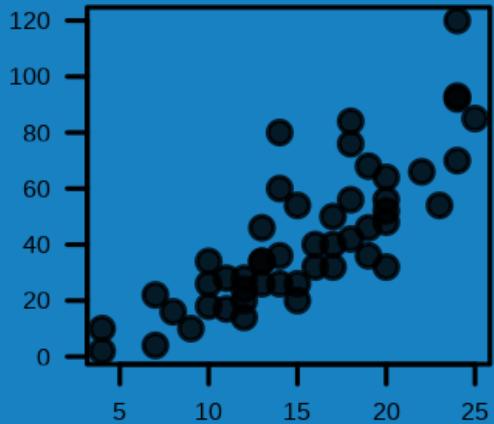
Representa os dados de: duas colunas

Modo das colunas: X = numérico e Y = numérico

Plota a relação entre duas variáveis contínuas



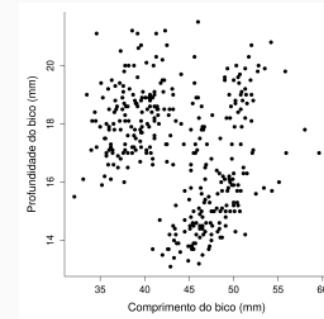
graphics

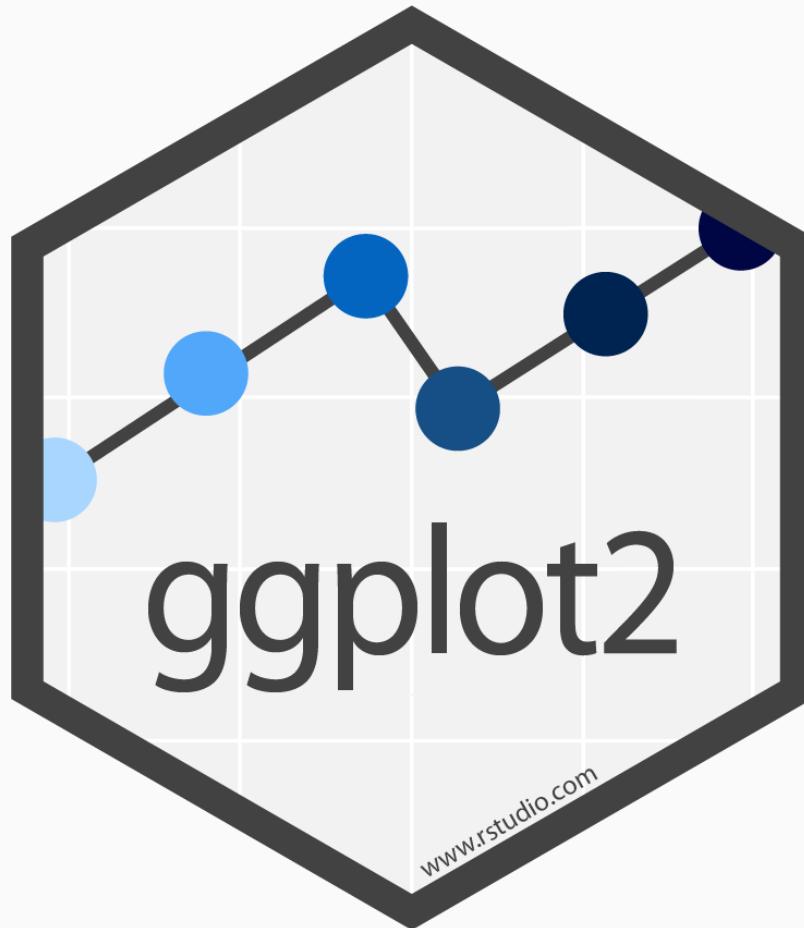


10. Gráfico de dispersão (*Scatter plot*)

graphics

```
par(mar = c(5, 5, 1, 1))
plot(bill_depth_mm ~ bill_length_mm,
     data = penguins,
     pch = 20,
     cex = 1.5,
     xlab = "Comprimento do bico (mm)",
     ylab = "Profundidade do bico (mm)",
     cex.lab = 1.5,
     cex.axis = 1.3,
     bty = "l")
```

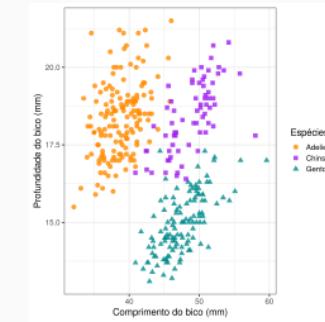




10. Gráfico de dispersão (*Scatter plot*)

ggplot2

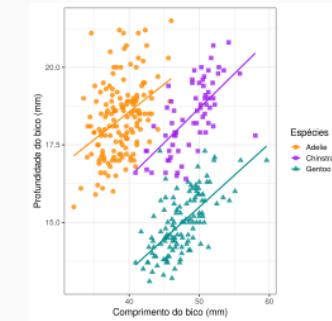
```
ggplot(data = penguins,
       aes(x = bill_length_mm,
           y = bill_depth_mm,
           color = species,
           shape = species)) +
  geom_point(size = 3, alpha = .8) +
  scale_shape_manual(values = c(19, 15, 17)) +
  scale_color_manual(values = c("darkorange", "purple",
                                "teal")) +
  theme_bw(base_size = 15) +
  labs(x = "Comprimento do bico (mm)",
       y = "Profundidade do bico (mm)",
       color = "Espécies", shape = "Espécies")
```



10. Gráfico de dispersão (*Scatter plot*)

ggplot2

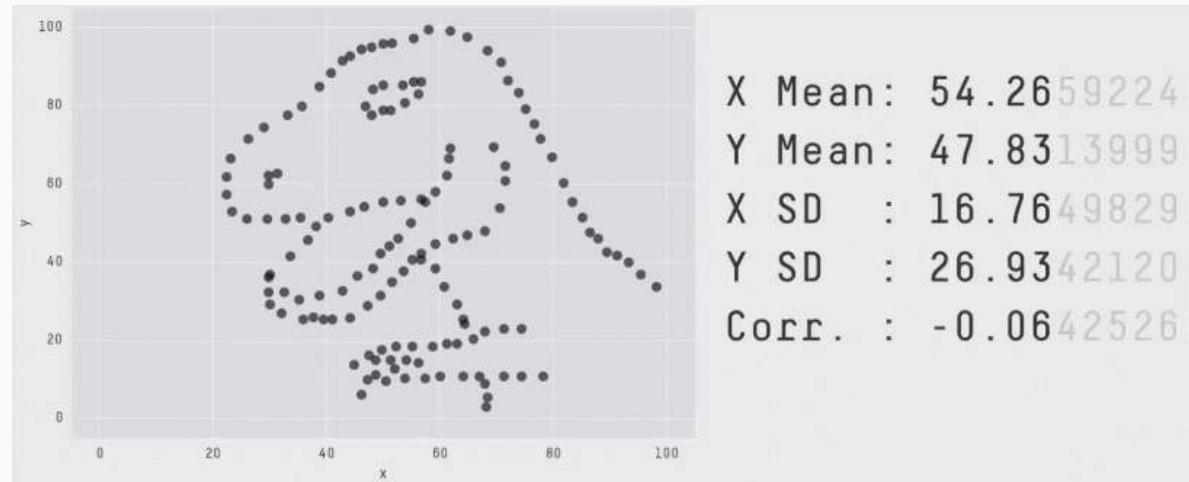
```
ggplot(data = penguins,
       aes(x = bill_length_mm,
           y = bill_depth_mm,
           color = species,
           shape = species)) +
  geom_point(size = 3, alpha = .8) +
  geom_smooth(method = "lm", se = FALSE) +
  scale_shape_manual(values = c(19, 15, 17)) +
  scale_color_manual(values = c("darkorange", "purple", "teal")) +
  theme_bw(base_size = 15) +
  labs(x = "Comprimento do bico (mm)",
       y = "Profundidade do bico (mm)",
       color = "Espécies", shape = "Espécies")
```



10. Gráfico de dispersão (*Scatter plot*)

Same Stats, Different Graphs: Generating Datasets with Varied Appearance and Identical Statistics through Simulated Annealing

Justin Matejka, George Fitzmaurice

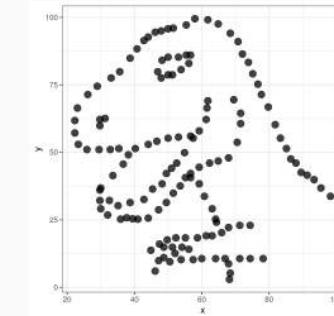


10. Gráfico de dispersão (*Scatter plot*)

dino

```
# package
library(datasauRus)

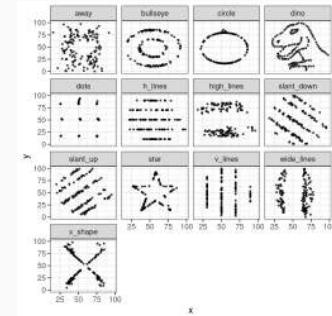
# data + plot
datasaurus_dozen %>%
  dplyr::filter(dataset = "dino") %>%
  ggplot() +
  aes(x = x, y = y) +
  geom_point(colour = "black", fill = "black",
             size = 5, alpha = .75, pch = 21) +
  theme_bw(base_size = 15)
```



10. Gráfico de dispersão (*Scatter plot*)

Todos os plots

```
datasaurus_dozen %>%
  ggplot() +
  aes(x = x, y = y) +
  geom_point(colour = "black", fill = "black",
             size = 1, alpha = .75, pch = 21) +
  facet_wrap(~dataset) +
  theme_bw(base_size = 15)
```

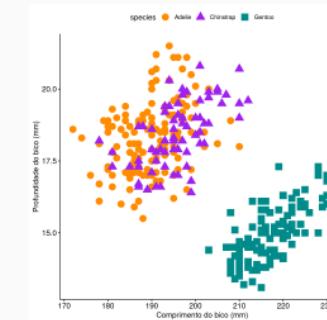




10. Gráfico de dispersão (*Scatter plot*)

ggpubr

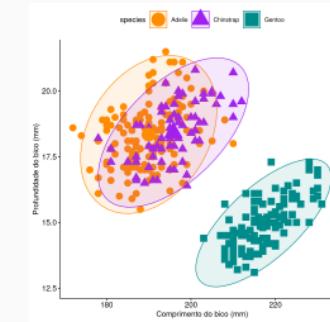
```
ggscatter(data = penguins,
          x = "flipper_length_mm",
          y = "bill_depth_mm",
          color = "species",
          fill = "species",
          palette = c("darkorange", "purple", "cyan4"),
          shape = "species",
          size = 5,
          xlab = "Comprimento do bico (mm)",
          ylab = "Profundidade do bico (mm)")
```



10. Gráfico de dispersão (*Scatter plot*)

ggpubr

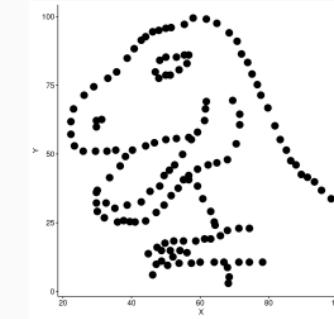
```
ggscatter(data = penguins,
          x = "flipper_length_mm",
          y = "bill_depth_mm",
          color = "species",
          fill = "species",
          palette = c("darkorange", "purple", "cyan4"),
          shape = "species",
          size = 5,
          xlab = "Comprimento do bico (mm)",
          ylab = "Profundidade do bico (mm)",
          ellipse = TRUE,
          mean.point = TRUE)
```



10. Gráfico de dispersão (*Scatter plot*)

ggpubr

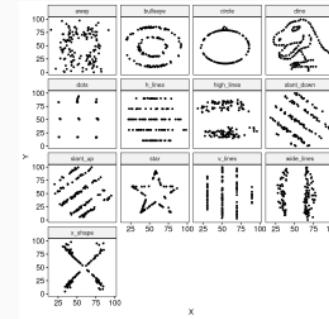
```
ggscatter(data = datasaurus_dozen %>%
  dplyr::filter(dataset == "dino"),
  x = "x",
  y = "y",
  size = 5,
  xlab = "X",
  ylab = "Y")
```



10. Gráfico de dispersão (*Scatter plot*)

ggpubr

```
ggscatter(data = datasaurus_dozen,
          x = "x",
          y = "y",
          size = 1,
          xlab = "X",
          ylab = "Y",
          facet.by = "dataset")
```

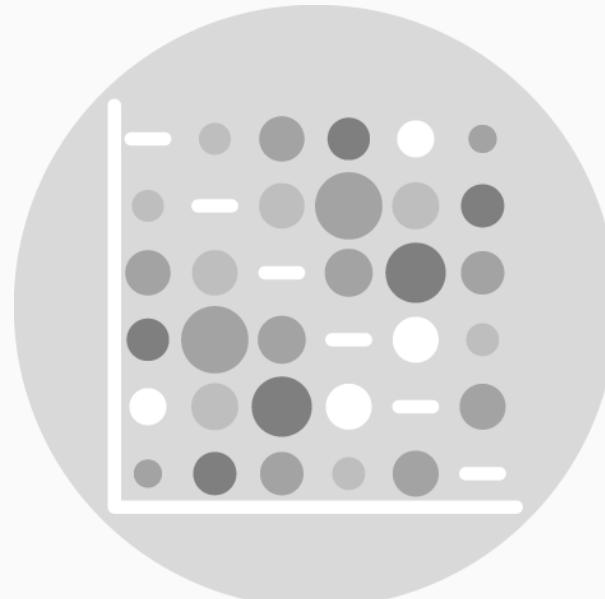


11. Gráfico pareado (*Pairs plot*)

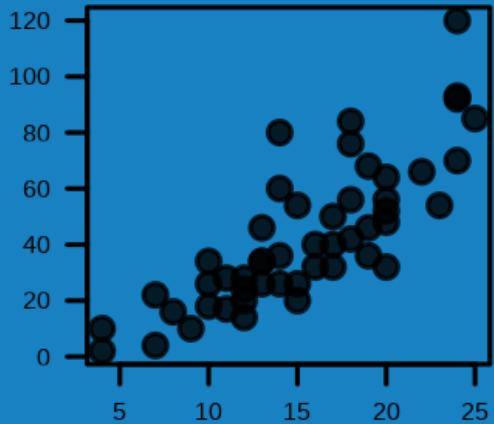
Representa os dados de: muitas colunas

Modo das colunas: X = numérico e Y = numérico

Plota a relação entre duas variáveis contínuas, mas para várias colunas



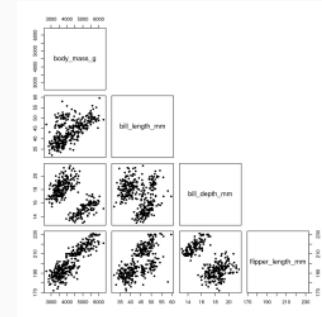
graphics

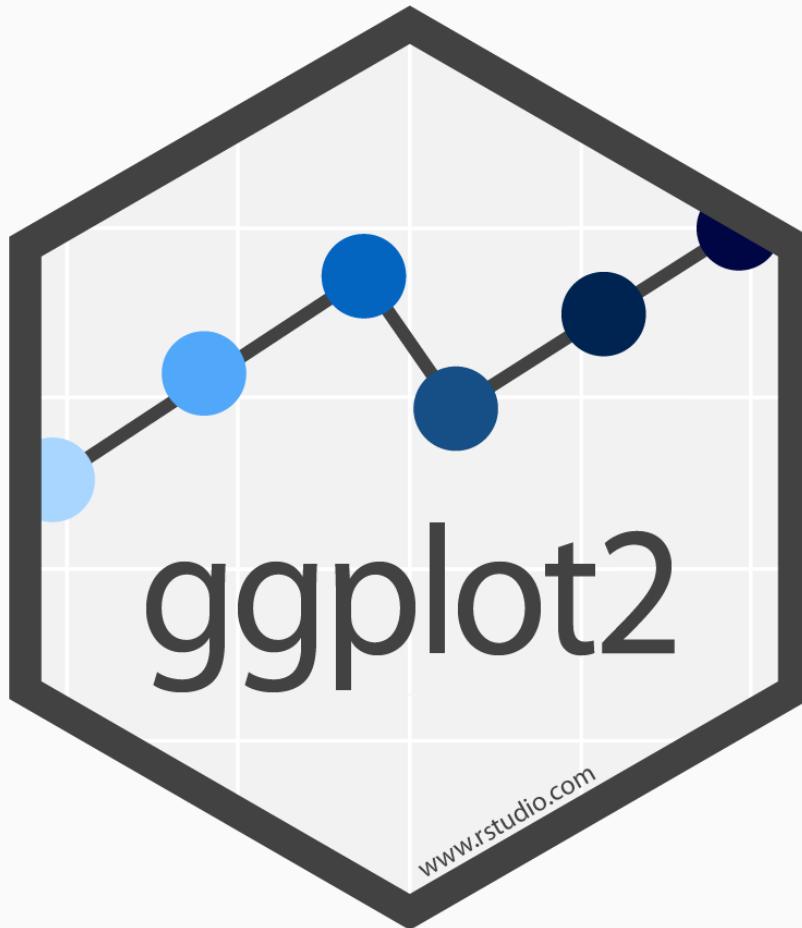


11. Gráfico pareado (*Pairs plot*)

graphics

```
penguins %>%
  dplyr::select(body_mass_g, ends_with("_mm")) %>%
  pairs(pch = 20,
        upper.panel = NULL)
```



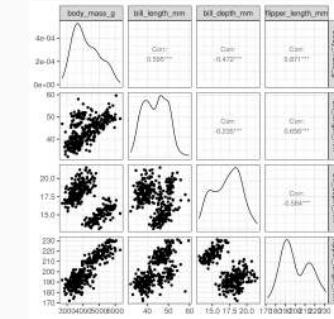


11. Gráfico pareado (*Pairs plot*)

GGally

```
# pacote
# install.packages("GGally")
library(GGally)

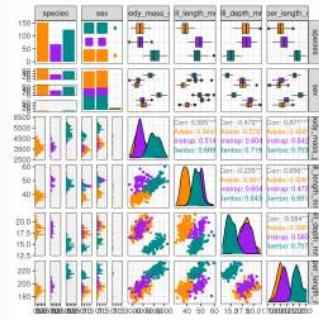
penguins %>%
  dplyr::select(body_mass_g, ends_with("_mm")) %>%
  ggpairs() +
  theme_bw(base_size = 15)
```



11. Gráfico pareado (*Pairs plot*)

GGally

```
penguins %>%
  dplyr::select(species, sex, body_mass_g, ends_with("m"))
GGally::ggpairs(aes(color = species)) +
  scale_colour_manual(values = c("darkorange", "purple")) +
  scale_fill_manual(values = c("darkorange", "purple")) +
  theme_bw(base_size = 15)
```

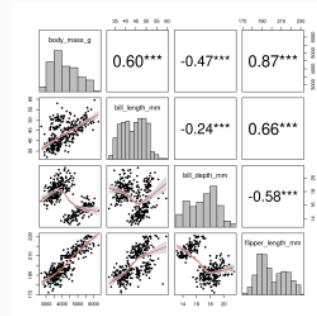


11. Gráfico pareado (*Pairs plot*)

psych

```
# pacote
# install.packages("psych")
library(psych)

penguins %>%
  dplyr::select(body_mass_g, ends_with("_mm")) %>%
  pairs.panels(pch = 20,
               ellipses = FALSE,
               density = FALSE,
               stars = TRUE,
               hist.col = "gray",
               digits = 2,
               rug = FALSE,
               breaks = 10,
               ci = TRUE)
```



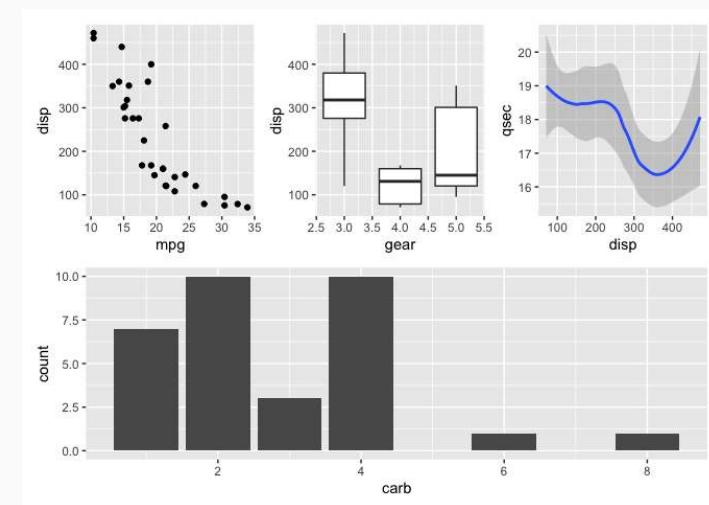
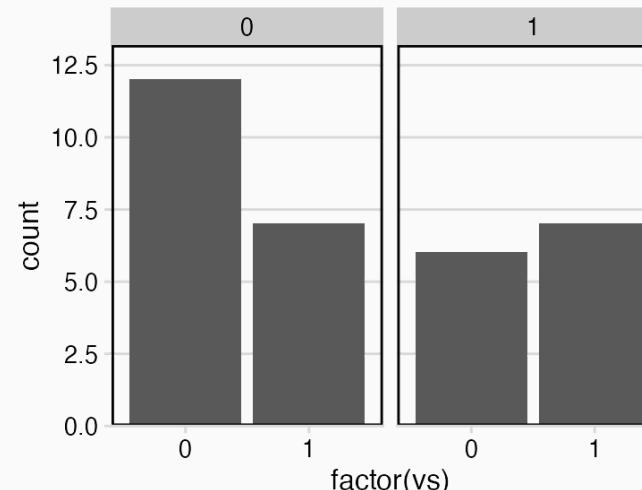
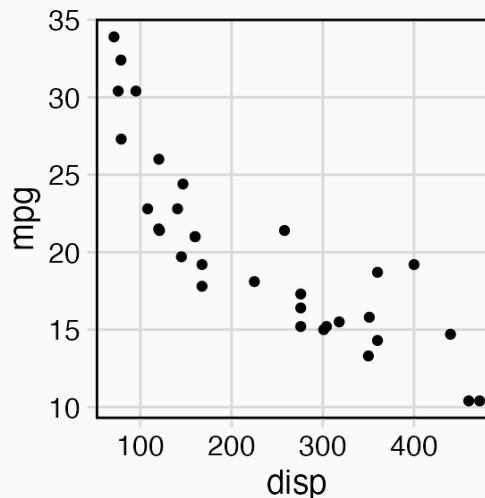
12. Combinando gráficos

Descrição

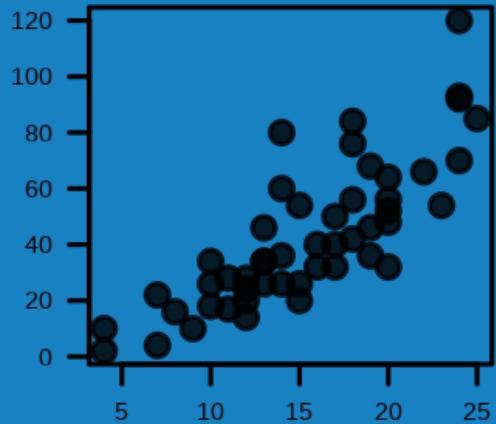
Muitas vezes precisamos **combinar gráficos** para representar nossos dados

Essa tarefa pode ser realizada depois da criação dos gráficos em aplicativos como [Inkscape](#) ou [GIMP](#)

Outras vezes, podemos usar códigos e pacotes do R para fazer essa tarefa



graphics



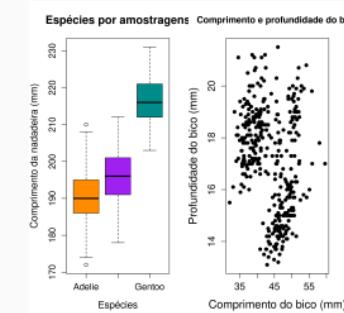
12. Combinando gráficos

graphics

```
par(mfrow = c(1, 2))

boxplot(flipper_length_mm ~ as.factor(species),
        data = penguins,
        col = c("darkorange", "purple", "cyan4"),
        main = "Espécies por amostragens",
        xlab = "Espécies",
        ylab = "Comprimento da nadadeira (mm)",
        cex.main = 1.5, cex.lab = 1.3, cex.axis = 1.3)

plot(bill_depth_mm ~ bill_length_mm,
      data = penguins,
      pch = 20,
      cex = 1.5,
      main = "Comprimento e profundidade do bico",
      xlab = "Comprimento do bico (mm)",
      ylab = "Profundidade do bico (mm)",
      cex.lab = 1.5, cex.axis = 1.3)
```



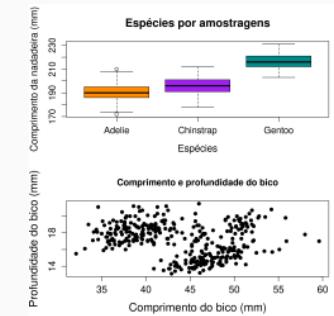
12. Combinando gráficos

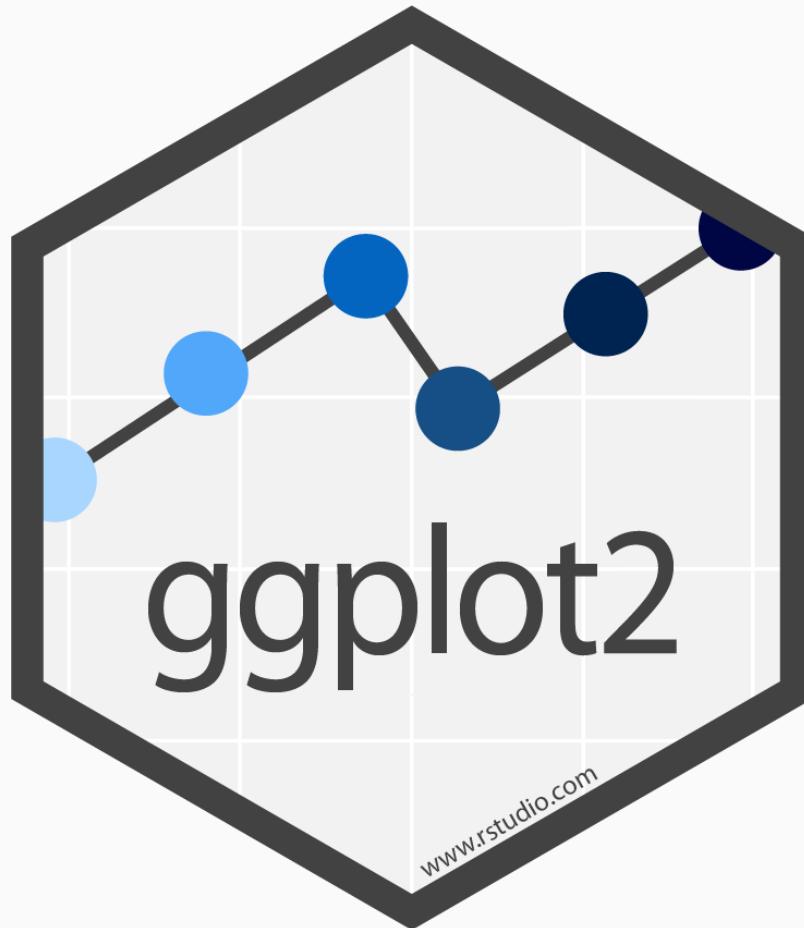
graphics

```
par(mfrow = c(2, 1))

boxplot(flipper_length_mm ~ as.factor(species),
        data = penguins,
        col = c("darkorange", "purple", "cyan4"),
        main = "Espécies por amostragens",
        xlab = "Espécies",
        ylab = "Comprimento da nadadeira (mm)",
        cex.main = 1.5, cex.lab = 1.3, cex.axis = 1.3

plot(bill_depth_mm ~ bill_length_mm,
      data = penguins,
      pch = 20,
      cex = 1.5,
      main = "Comprimento e profundidade do bico",
      xlab = "Comprimento do bico (mm)",
      ylab = "Profundidade do bico (mm)",
      cex.lab = 1.5, cex.axis = 1.3)
```



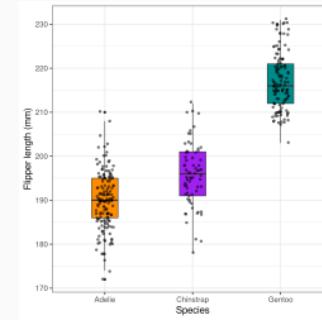


[ggplot2](#)

12. Combinando gráficos

ggplot2

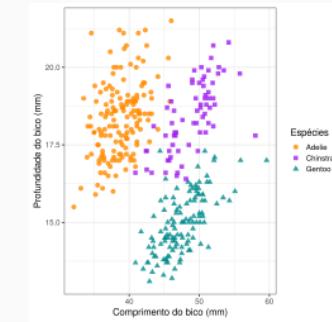
```
ggplot_boxplot ← ggplot(data = penguins,
                         aes(x = species, y = flipper_length_mm))
  geom_boxplot(width = .3,
                show.legend = FALSE) +
  geom_jitter(alpha = .5,
              show.legend = FALSE,
              position = position_jitter(width = .1,
                                          scale_fill_manual(values = c("darkorange", "purple",
                                          "teal")))) +
  theme_bw(base_size = 15) +
  labs(x = "Species", y = "Flipper length (mm)")
ggplot_boxplot
```

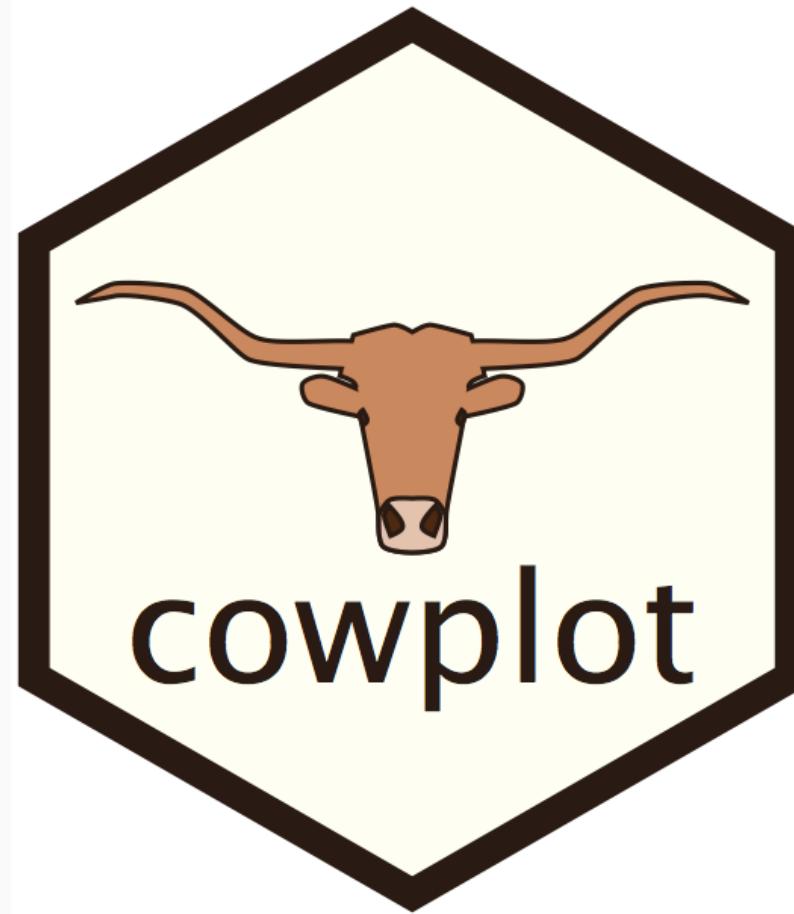


12. Combinando gráficos

ggplot2

```
ggplot_scatterplot ← ggplot(data = penguins,
                           aes(x = bill_length_mm,
                               y = bill_depth_mm,
                               color = species,
                               shape = species)) +
  geom_point(size = 3, alpha = .8) +
  scale_shape_manual(values = c(19, 15, 17)) +
  scale_color_manual(values = c("darkorange", "purple",
                                "teal")) +
  theme_bw(base_size = 15) +
  labs(x = "Comprimento do bico (mm)",
       y = "Profundidade do bico (mm)",
       color = "Espécies", shape = "Espécies")
ggplot_scatterplot
```





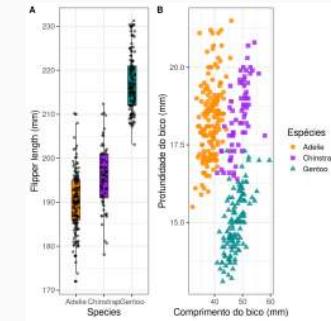
[cowplot](#)

12. Combinando gráficos

cowplot

```
# pacote
# install.packages("cowplot")
library(cowplot)

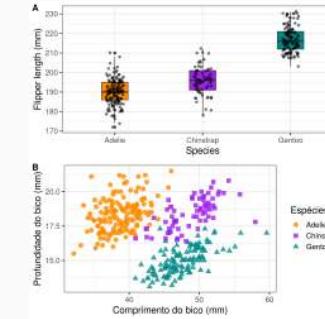
# combinacao horizontal
plot_grid(ggplot_boxplot, ggplot_scatterplot,
          align = "h", rel_widths = c(1, 1.5),
          labels = "AUTO")
```



12. Combinando gráficos

cowplot

```
# combinacao vertical  
plot_grid(ggplot_boxplot, ggplot_scatterplot,  
          ncol = 1, align = "v",  
          labels = "AUTO")
```



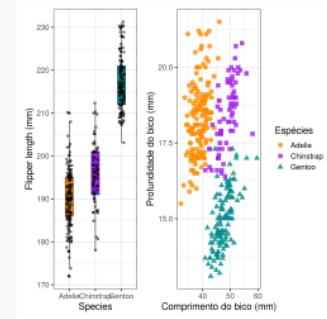


[patchwork](#)

12. Combinando gráficos

patchwork

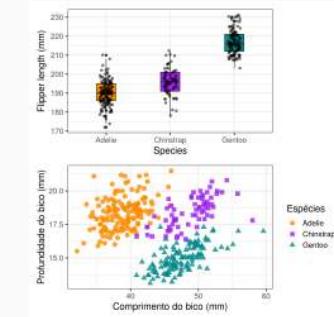
```
# pacote  
# install.packages("patchwork")  
library(patchwork)  
  
# combinacao horizontal  
ggplot_boxplot + ggplot_scatterplot
```

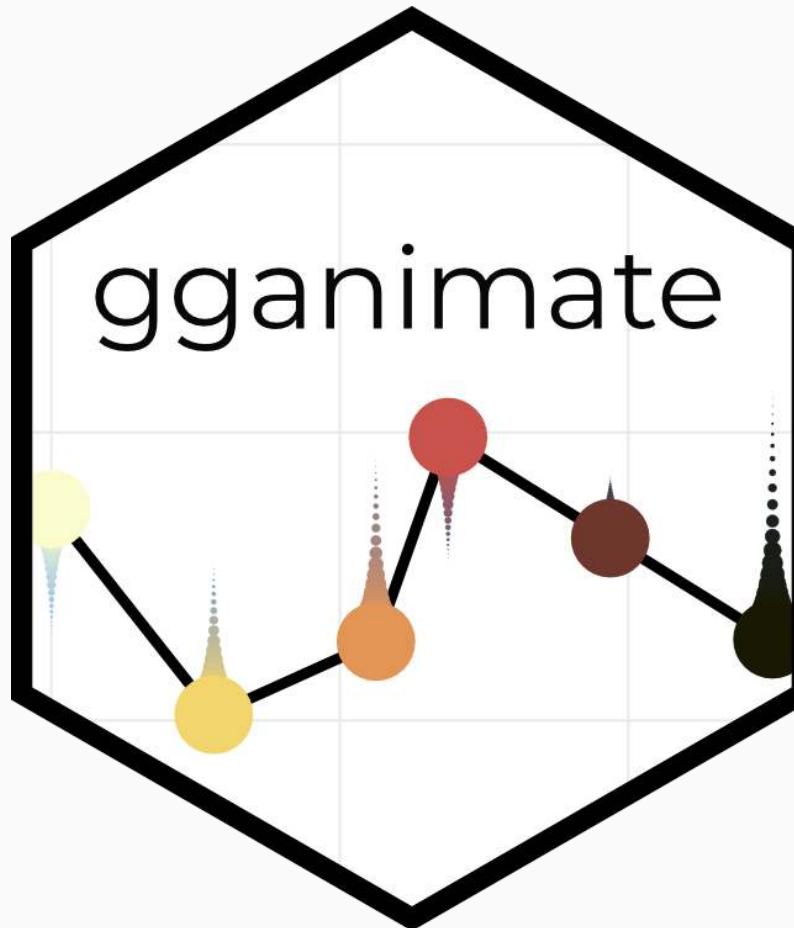


12. Combinando gráficos

patchwork

```
# combinacao vertical  
ggplot_boxplot / ggplot_scatterplot
```





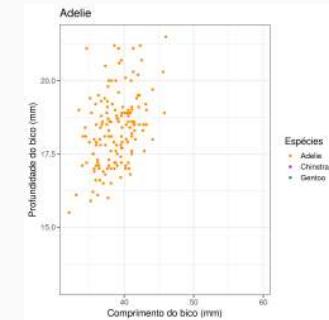
[ganimate](#)

13. Gráficos animados

gganimate

```
# pacote
# devtools::install_github('thomasp85/gganimate')
library(gganimate)

plot_animate ← ggplot(data = penguins,
                      aes(x = bill_length_mm,
                          y = bill_depth_mm,
                          color = species)) +
  geom_point() +
  scale_color_manual(values = c("darkorange", "purple",
    "blue", "red")) +
  theme_bw(base_size = 15) +
  labs(x = "Comprimento do bico (mm)",
       y = "Profundidade do bico (mm)",
       color = "Espécies", shape = "Espécies") +
  labs(title = "{closest_state}") +
  transition_states(species) +
  enter_grow() +
  exit_fade()
plot_animate
```



13. Gráficos animados

gganimate

Exportar

```
gganimate::anim_save(filename = here::here("03_dados",
                                         "graficos" ,
                                         "plot_animate.gif"),
                      animation = plot_animate)
```



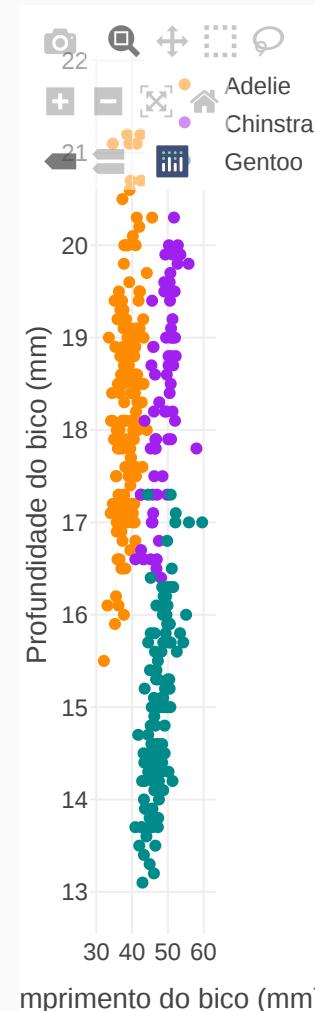
plotly

14. Gráficos interativos

plotly

```
# pacote
# install.packages("plotly")
library(plotly)

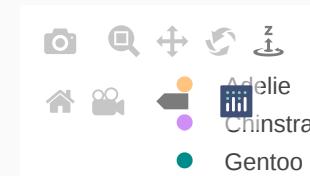
plot_ly(data = penguins,
        x = ~bill_length_mm,
        y = ~bill_depth_mm,
        type = "scatter",
        color = ~species,
        colors = c("darkorange", "purple", "cyan4"))
layout(xaxis = list(title = "Comprimento do bico (mm)",
                     ticks = c(30, 40, 50, 60)),
       yaxis = list(title = "Profundidade do bico (mm)",
                     ticks = c(13, 14, 15, 16, 17, 18, 19, 20, 21, 22)))
```



14. Gráficos interativos

plotly

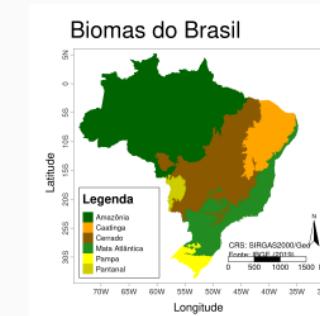
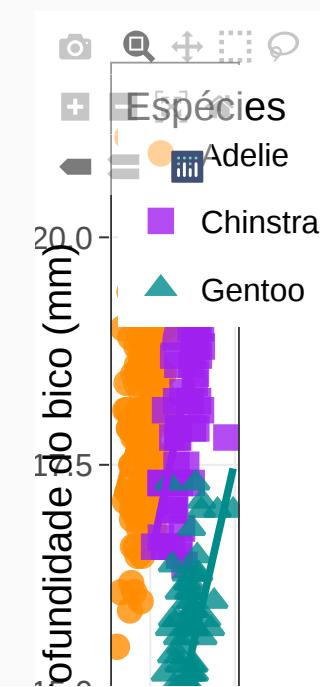
```
plot_ly(data = penguins,
        x = ~bill_length_mm,
        y = ~bill_depth_mm,
        z = ~body_mass_g,
        type = "scatter3d",
        color = ~species,
        colors = c("darkorange", "purple", "cyan4"))
layout(scene = list(xaxis = list(title = "Comprimento da Bico (mm)"),
                    yaxis = list(title = "Profundidade da Bico (mm)"),
                    zaxis = list(title = "Massa (g)")))
```



14. Gráficos interativos

plotly

```
plot_penguins_scatter_int ← ggplotly(  
  ggplot(data = penguins,  
    aes(x = bill_length_mm,  
        y = bill_depth_mm,  
        color = species,  
        shape = species)) +  
  geom_point(size = 3, alpha = .8) +  
  geom_smooth(method = "lm", se = FALSE) +  
  scale_shape_manual(values = c(19, 15, 17)) +  
  scale_color_manual(values = c("darkorange", "purple")) +  
  theme_bw(base_size = 15) +  
  labs(x = "Comprimento do bico (mm)",  
       y = "Profundidade do bico (mm)",  
       color = "Espécies", shape = "Espécies"))  
plot_penguins_scatter_int
```



14. Gráficos interativos

plotly

Exportar

```
# pacote
# install.packages("htmlwidgets")
library(htmlwidgets)

# export
htmlwidgets::saveWidget(widget = plot_penguins_scatter_int,
                        file = here::here("03_dados", "graficos", "plot_penguins_scatter_int.html"))
```



15. Gráficos usando interface esquisse

```
# pacote  
#install.packages("esquisse")  
library(esquisse)  
  
# iniciar  
esquisse :: esquisser(iris)  
esquisse :: esquisser(palmerpenguins :: penguins)
```



Dúvidas?

Leitura

I O que é Geoprocessamento?

Conceito não pode ser confundido com todo o conjunto das geotecnologias, como o Sensoriamento Remoto, a Cartografia e os Sistemas de Posicionamento Global (GPS).

Geógrafo Jorge Xavier da Silva

Coordenador do Laboratório de Geoprocessamento (LAGEOP) da UFRJ

[Xavier-da-Silva \(2009\)](#)

Dúvidas?

Maurício Vancine

Contatos:

✉ mauricio.vancine@gmail.com

🐦 [@mauriciovancine](https://twitter.com/mauriciovancine)

/github [mauriciovancine](https://github.com/mauriciovancine)

🔗 mauriciovancine.github.io



Slides criados via pacote [xaringan](#) e tema [Metropolis](#). Animação dos sapos por [@probzz](#).