

# Introdução à Visualização de Dados com o Pacote Ggplot2 do R

## Aula 01: Introdução ao ggplot2 e ao *grammar of graphics*



# Quem sou eu?



## Bruna Garbes

Cientista de dados no Hospital Albert Einstein  
Graduada em Ciência e Tecnologia (UFABC)  
Mestre em Bioinformática (IME-USP)

 /brunagarbes

 /mardedados

 /brunagarbes

# Agenda

- Avisos
- Objetivos do curso
- Ciclo da ciência de dados
- Relembrando conceitos de R e Rstudio
- Por que usar Ggplot2?
- Um pouco da história da visualização de dados
- O conceito de *grammar of graphics*.
  - Aspectos estéticos
  - Aspectos geométricos
  - Escalas
  - Faceamento
  - Coordenadas
  - Legendas
  - Temas
- Juntando tudo
- Descrevendo gráficos com a gramática
- Exercícios



# Avisos



- Recomendamos ter as telas da videochamada e do RStudio visíveis o tempo todo.
- O material da aula estará disponível na plataforma Cursos Extensão.
- Siga os códigos e os slides e execute os comandos ao longo da aula, conforme eles forem sendo abordados.
- Faça perguntas no chat para que os monitores possam ir tirando as suas dúvidas ao longo da aula. Nossos monitores são:
  - Alberto Rodrigues Ferreira
  - Alan da Silva
  - Geovana Lopes Batista
- Haverá atividades práticas e quizzes que deverão ser entregues, pois valem nota.
- Preencham a lista de chamada, pois para receberem certificados é necessário ter mais de 75% de presença!



# Objetivos deste curso

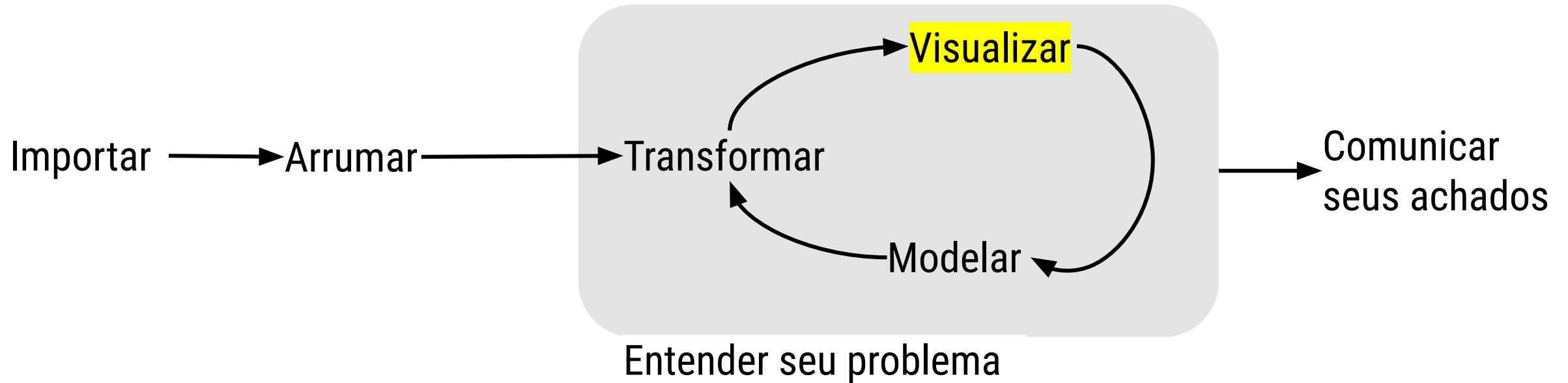
## Aprender Ggplot!

- Implementar a gramática de gráficos do ggplot2 usando a função `ggplot()` e construindo gráficos com o operador `+`;
- Incorporar elementos personalizados (cores, fontes, etc.) em suas visualizações ajustando os elementos do tema `ggplot2`;
- Construir gráficos mais limpos e que transmitam a nossa mensagem de forma clara e direta e
- Ganhar independência e capacidade para investigar o mundo do ggplot2 para expandir as habilidades aprendidas no curso.



# Ciclo da Ciência de Dados

Nosso foco será na visualização

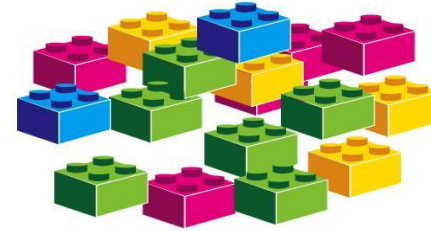


Fonte: [R for Data Science](#)

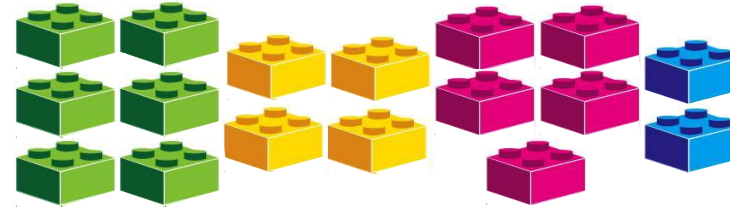
# Ciclo da Ciência de Dados

Nosso foco será na visualização

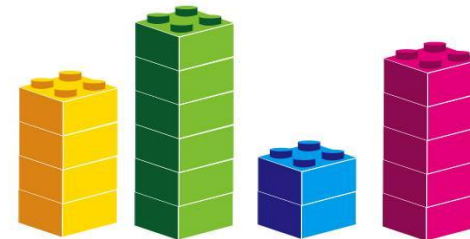
**Dados**



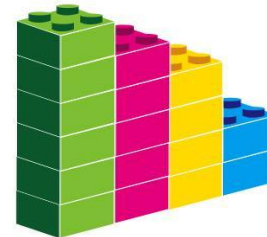
**Classificação**



**Organização**



**Apresentação visual**



# Relembrando conceitos de R e Rstudio

## O que é e por que usar o R?

- R é um ambiente de software livre para computação estatística e gráficos(<https://www.r-project.org/>);
- É uma linguagem de programação para análise de dados;
- É open-source;
- Possui uma comunidade ativa de desenvolvedores;
- Muito usado por cientistas de dados, estatísticos e pesquisadores;
- É flexível, permitindo desenvolver funções e pacotes para facilitar o trabalho;
- Está disponível em diferentes plataformas: Windows, Linux e Mac.



# Relembrando conceitos de R e Rstudio

## O que podemos fazer com o R?

- Análise de dados - Estatística, modelagem, etc.
- Visualização de dados
- Apresentações
- Relatórios dinâmicos
- Escrever livros
- Mineração de dados
- E muito mais!





# Relembrando conceitos de R e Rstudio

## Instalação do R e RStudio

- Instalação do R - <https://cloud.r-project.org/>
- Instalação do R Studio - <https://www.rstudio.com/products/rstudio/download/#download>



# Relembrando conceitos de R e Rstudio

## Rstudio

É o IDE da Linguagem R, ou seja, o ambiente que utilizamos para editar e executar os códigos em R.

The screenshot displays the RStudio interface with three main components highlighted:

- Editor:** Contains the following R code:

```
1 library(ggplot2)
2
3 ggplot(data = diamonds) +
4   geom_bar(mapping = aes(x = cut, fill = clarity))
5
```
- Console:** Shows the execution of the code:

```
Es/Oficina_R/ > library(ggplot2)
> ggplot(data = diamonds) +
+   geom_bar(mapping = aes(x = cut, fill = clarity))
> |
```
- Output:** Displays a stacked bar chart showing the count of diamonds for each cut category, grouped by clarity. The x-axis represents 'cut' (Fair, Good, Very Good, Premium, Ideal) and the y-axis represents 'count' (0 to 20000). The legend indicates clarity levels: I1, SI2, SI1, VS2, VS1, VVS2, VVS1, and IF.

# Relembrando conceitos de R e Rstudio

## Atalhos importantes:

Os atalhos facilitam. Vejam os principais:

- CTRL + ENTER: roda a linha selecionada no script
- ALT + -: (<-) sinal de atribuição
- CTRL + SHIFT + M: (%>%) operador pipe

# Relembrando conceitos de R e Rstudio

## Classes básicas ou atômicas do R:

- **Character:** texto
- **Integer:** números inteiros
- **Numeric:** números racionais
- **Complex:** números complexos (raramente usados para Análise de Dados)
- **Logical:** TRUE, FALSE ou NA
- **Factor:** variáveis categóricas

# Relembrando conceitos de R e Rstudio

## Tipos de objetos:

- **Vector:** armazena elementos de mesma classe
- **Matrix:** vetores de duas dimensões que armazenam elementos de mesma classe
- **List:** tipo especial de vetor que aceita elementos de classes diferentes
- **Data.frame:** são tabelas de dados com linhas e colunas, como uma tabela do Excel. Como são listas, essas colunas podem ser de classes diferentes

country	year	cases	population
Afghanistan	1999	745	19987071
Afghanistan	2000	2666	20095360
Brazil	1999	37737	172006362
Brazil	2000	80488	174504898
China	1999	210258	1272915272
China	2000	210766	128042583

variables

country	year	cases	population
Afghanistan	1999	745	19987071
Afghanistan	2000	2666	20095360
Brazil	1999	37737	172006362
Brazil	2000	80488	174504898
China	1999	210258	1272915272
China	2000	210766	128042583

observations

country	year	cases	population
Afghanistan	99	745	19987071
Afghanistan	00	2666	20095360
Brazil	99	37737	172006362
Brazil	00	80488	174504898
China	99	210258	1272915272
China	00	210766	128042583

values



# Relembrando conceitos de R e Rstudio

## Formato longo

Para o ggplot() funcionar, nossos dados precisam estar no formato longo. Aliás, todos os pacotes do tidyverse funcionam melhor com o dado nesse formato.

Cada variável tem sua própria coluna

country	year	cases	population
Afghanistan	1999	745	19987071
Afghanistan	2000	2666	20095360
Brazil	1999	37737	17200362
Brazil	2000	80488	174604898
China	1999	210258	1272915272
China	2000	210766	128042583

variables

Cada observação tem a sua própria linha

country	year	cases	population
Afghanistan	1999	745	19987071
Afghanistan	2000	2666	20095360
Brazil	1999	37737	17200362
Brazil	2000	80488	174604898
China	1999	210258	1272915272
China	2000	210766	128042583

observations

Cada valor tem a sua própria célula

country	year	cases	population
Afghanistan	99	745	19987071
Afghanistan	00	2666	20095360
Brazil	99	37737	17200362
Brazil	00	80488	174604898
China	99	210258	1272915272
China	00	210766	128042583

values

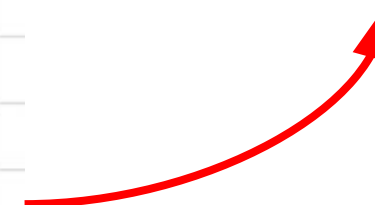
# Relembrando conceitos de R e Rstudio

## Formato longo

Porém, é muito mais comum encontrar dados fora desse formato.

	A	B	C	D	E
1	Office	Year	Incidents	Needs Verification	Different Source
2	Utah County	2015	134	FALSE	FALSE
3	Salt Lake County	2015	302	TRUE	FALSE
4	Davis County	2015	254	FALSE	FALSE
5	Juab County	2015	78	FALSE	FALSE
6	Utah County	2016	145	FALSE	TRUE
7	Salt Lake County	2016	334	FALSE	FALSE
8	Davis County	2016	288	FALSE	FALSE
9	Juab County	2016	82	TRUE	TRUE
10	Utah County	2017	167	TRUE	FALSE

Agora assim os dados são plotáveis!



# Relembrando conceitos de R e Rstudio

## Formato longo

Atualmente, o comando `gather()` é chamado de `pivot_longer()` e o comando `spread()` é chamado de `pivot_wider()`.

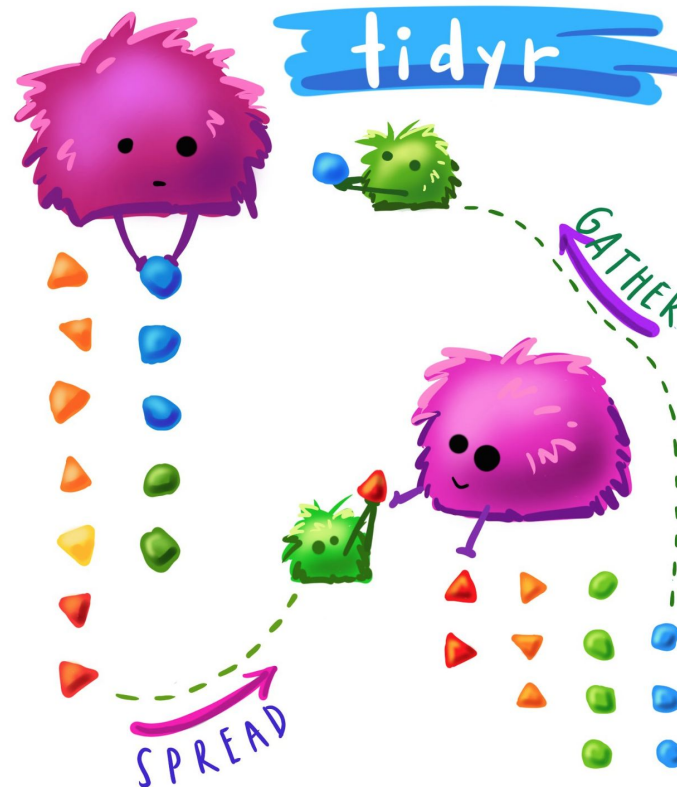
wide

id	x	y	z
1	a	c	e
2	b	d	f

# Relembrando conceitos de R e Rstudio

## Formato longo

Atualmente, o comando `gather()` é chamado de `pivot_longer()` e o comando `spread()` é chamado de `pivot_wider()`.



# Relembrando conceitos de R e Rstudio

## Pacotes no R

Pacotes são coleções de funções, dados e documentação que estendem as capacidades do R básico. Eles precisam ser instalados e carregados.





# Relembrando conceitos de R e Rstudio

## O pacote Tidyverse

É uma coleção de pacotes R projetados para a ciência de dados. Todos os pacotes compartilham uma mesma filosofia de desenvolvimento, sintaxe e estruturas de dados.



# Relembrando conceitos de R e Rstudio

## Instalar e carregar pacotes

- Via CRAN: `install.packages("nome-do-pacote")`.

```
install.packages("tidyverse")
```

- Via Github: `devtools::install_github("nome-do-repo/nome-do-pacote")`.

```
devtools::install_github("tidyverse/dplyr")
```

- `library(nome-do-pacote)`

```
library(tidyverse)
```

# Relembrando conceitos de R e Rstudio

## Os pacotes do Tidyverse

- **ggplot2**: cria gráficos
- **dplyr**: manipulação de dados
- **tidyr**: arruma os dados
- **readr**: leitura dos dados
- **purrr**: ferramentas para programação funcional, trabalha com funções e vetores
- **tibble**: dataframes moderno, mais simples de manipular
- **magrittr**: facilita a escrita e leitura de código
- **stringr**: trabalha com strings
- **forcats**: trabalha com fatores
- **lubridate**: trabalha com datas



# Relembrando conceitos de R e Rstudio

## O operador pipe %>%

Imagine uma receita que tenha as seguintes instruções: junte os ingredientes, misture e leve ao forno. Na forma usual do R, teríamos:

- **forno(misture(junte(ingredientes)))**

Dessa forma temos de pensar "de dentro para fora". O primeiro comando que vemos é forno, sendo que essa é a última operação que será realizada. Já com o operador pipe temos algo assim:

- **ingredientes %>% junte %>% misture %>% forno**

Podemos pensar no pipe %>% como um operador que efetua as operações à direita nos valores que estão à esquerda, ou ainda, o operador %>% passa o que está à esquerda como argumento para a operação da direita.

- **Atalho: CTRL + SHIFT + M**

# Relembrando conceitos de R e Rstudio

## Os 7 principais verbos do Dplyr

A ideia do pacote dplyr é tornar a manipulação de dados explícita utilizando verbos que indicam a ação a ser realizada. O dplyr foi desenhado para trabalhar com o operador pipe `%>%` do pacote magrittr.

- **filter()**: seleciona linhas
- **arrange()**: ordena de acordo com uma ou mais colunas
- **select()**: seleciona colunas
- **mutate()**: cria/modifica colunas
- **summarise()**: sumariza/agrega colunas
- **group\_by()**: agrupa colunas
- **join()**: junta dois conjuntos de dados por meio de um ou mais campos em comum (chaves/keys)

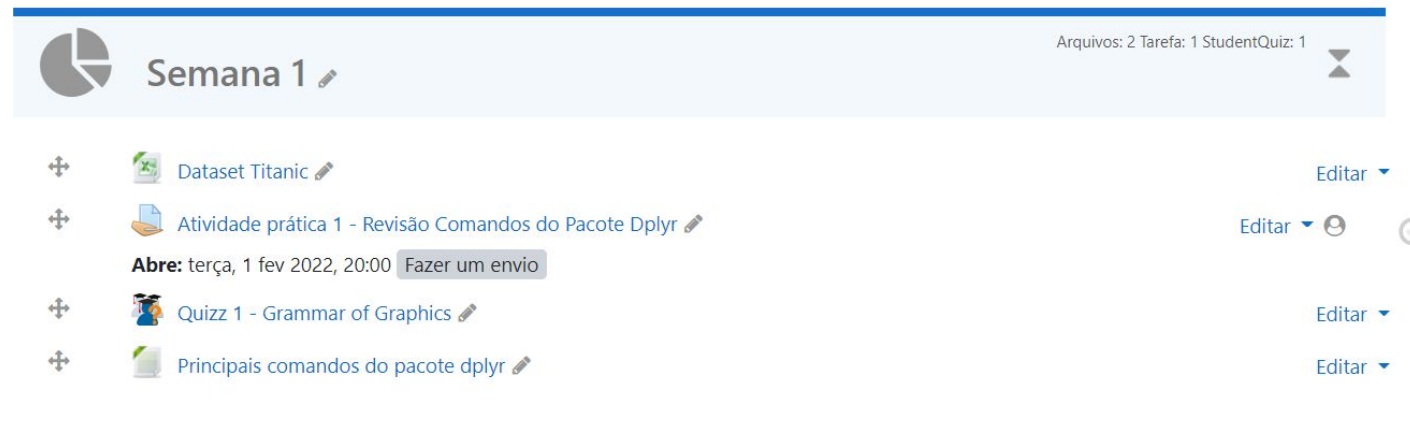
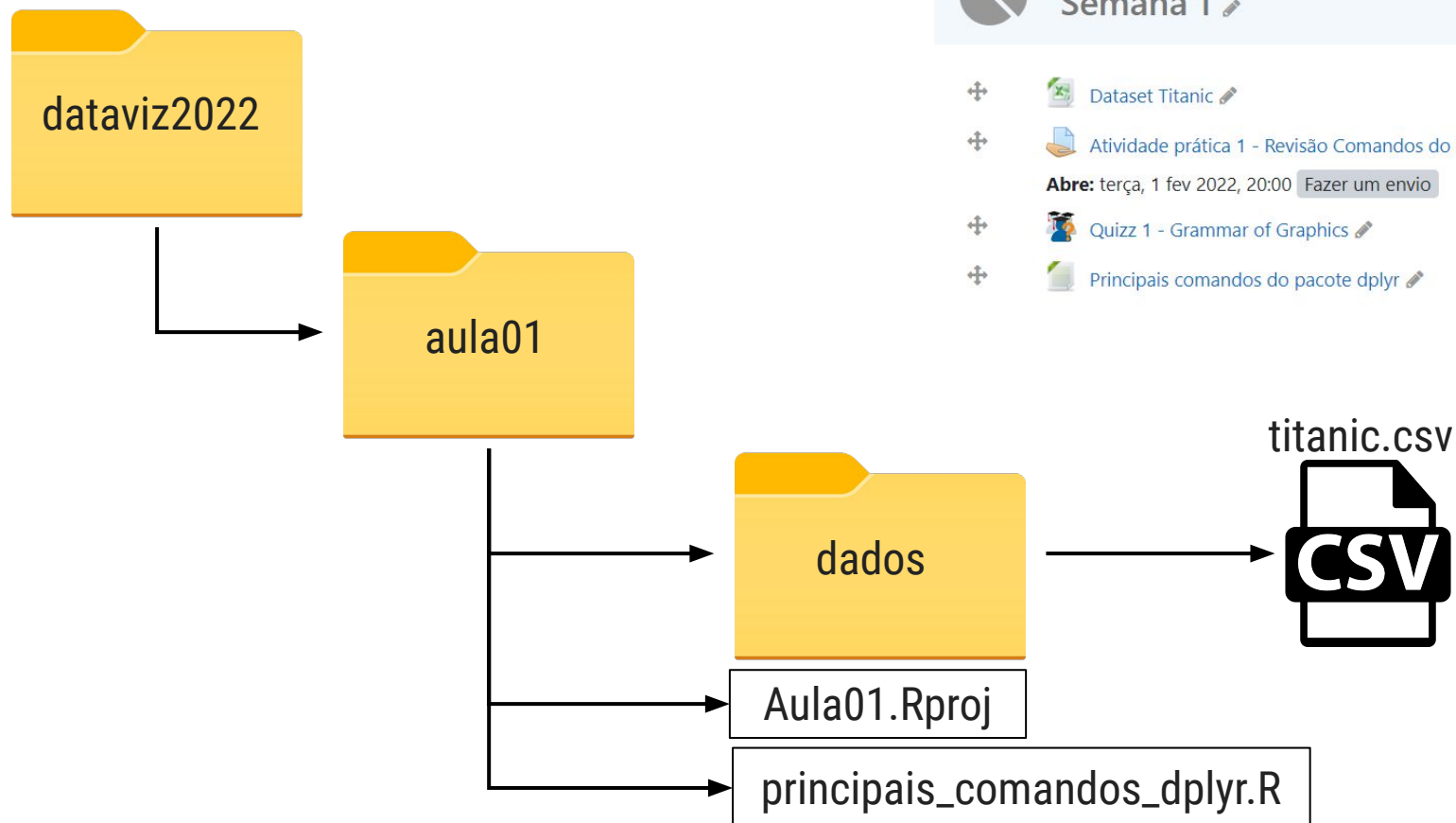




# Relembrando conceitos de R e Rstudio

## Preparando nosso ambiente

Vamos criar a seguinte estrutura de pastas:



# Relembrando conceitos de R e Rstudio

## Importando os dados

```
library(tidyverse)
```

```
# Uma outra opção é carregar somente o(s) pacote(s) que irá utilizar.
```

```
#library(dplyr)
```

```
# Importa o arquivo csv para o objeto df_titanic
```

```
df_titanic <- read_csv("data/titanic.csv")
```

# Relembrando conceitos de R e Rstudio

## filter(): filtra linhas da base de dados

```
# Seleciona os sobreviventes.  
df_titanic %>% filter(sobreviveu == "sim")
```

```
## # A tibble: 342 x 12  
##   id_passageiro sobreviveu classe nome  sexo  idade irmaos_conjuge  
##         <int> <chr>         <int> <chr> <chr> <dbl>         <int>  
## 1             2 sim             1 Cumi... femi...   38             1  
## 2             3 sim             3 Heik... femi...   26             0  
## 3             4 sim             1 Futr... femi...   35             1  
## 4             9 sim             3 John... femi...   27             0  
## 5            10 sim             2 Nass... femi...   14             1  
## 6            11 sim             3 Sand... femi...    4             1  
## 7            12 sim             1 Bonn... femi...   58             0  
## 8            16 sim             2 Hewl... femi...   55             0  
## 9            18 sim             2 Will... masc...   NA             0  
## 10           20 sim             3 Mass... femi...   NA             0  
## # ... with 332 more rows, and 5 more variables: pais_crianças <int>,  
## #   passagem <chr>, tarifa <dbl>, cabine <chr>, embarque <chr>
```

**Tibble** é uma releitura moderna do data.frame.

# Relembrando conceitos de R e Rstudio

## filter(): filtra linhas da base de dados

```
# Cria um objeto e atribui a ele as linhas com os sobreviventes.
sobreviventes <- df_titanic %>% filter(sobreviveu == "sim")

# Crianças com menos de 12 anos que sobreviveram.
criancas_sobreviventes <- df_titanic %>% filter(sobreviveu == "sim" & idade < 12)

# Embarque realizado nos locais: Southampton ou Queenstow.
embarque <- df_titanic %>% filter(embarque == "Southampton" |
                                embarque == "Queenstow")

# A instrução acima pode ser reescrita com o operador %in%:
embarque <- df_titanic %>% filter(embarque %in% c("Southampton", "Queenstow"))

# Pessoas sem informação de local de embarque.
# is.na() - função que retorna TRUE se o valor for NA e FALSE se não for.
sem_embarque <- df_titanic %>% filter(is.na(embarque))

# Pessoas que tem "Elizabeth" em qualquer posição do campo nome.
# str_detect - função que retorna TRUE se detectou o valor dado e
# FALSE, caso não tenha encontrado.
nome <- df_titanic %>% filter(str_detect(nome, "Elizabeth"))
```

# Relembrando conceitos de R e Rstudio

`arrange()`: ordena as linhas da base de dados de acordo com uma ou mais colunas

```
# Ordena por ordem crescente da coluna nome.  
passageiros_ordenados <- df_titanic %>% arrange(nome)  
passageiros_ordenados
```

```
## # A tibble: 891 x 12  
##   id_passageiro sobreviveu classe nome  sexo  idade irmaos_conjuge  
##         <int> <chr>         <int> <chr> <chr> <dbl>         <int>  
## 1         846 nao             3 Abbi... masc...  42             0  
## 2         747 nao             3 Abbo... masc...  16             1  
## 3         280 sim             3 Abbo... femi...  35             1  
## 4         309 nao             2 Abel... masc...  30             1  
## 5         875 sim             2 Abel... femi...  28             1  
## 6         366 nao             3 Adah... masc...  30             0  
## 7         402 nao             3 Adam... masc...  26             0  
## 8          41 nao             3 Ahli... femi...  40             1  
## 9         856 sim             3 Aks,... femi...  18             0  
## 10        208 sim             3 Albi... masc...  26             0  
## # ... with 881 more rows, and 5 more variables: pais_crianças <int>,  
## #   passagem <chr>, tarifa <dbl>, cabine <chr>, embarque <chr>
```



# Relembrando conceitos de R e Rstudio

## select(): seleciona colunas

```
# Seleciona as colunas indicadas.  
df_titanic %>% select(nome, idade, classe, embarque)
```

```
## # A tibble: 891 x 4  
##   nome                               idade classe embarque  
##   <chr>                             <dbl>  <int> <chr>  
## 1 Braund, Mr. Owen Harris             22      3 Southampt...  
## 2 Cumings, Mrs. John Bradley (Florence Briggs Th... 38      1 Cherbourg  
## 3 Heikkinen, Miss. Laina              26      3 Southampt...  
## 4 Futrelle, Mrs. Jacques Heath (Lily May Peel)     35      1 Southampt...  
## 5 Allen, Mr. William Henry            35      3 Southampt...  
## 6 Moran, Mr. James                    NA      3 Queenstow  
## 7 McCarthy, Mr. Timothy J            54      1 Southampt...  
## 8 Palsson, Master. Gosta Leonard        2      3 Southampt...  
## 9 Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Be... 27      3 Southampt...  
## 10 Nasser, Mrs. Nicholas (Adele Achem)          14      2 Cherbourg  
## # ... with 881 more rows
```

# Relembrando conceitos de R e Rstudio

mutate(): cria/modifica colunas

```
# Altera a coluna tarifa para o valor da tarifa em reais.  
tarifa_conversao <- df_titanic %>% mutate(tarifa = tarifa * 5,37 )
```

```
# Retorna a coluna tarifa para o valor da época.  
tarifa_conversao <- df_titanic %>% mutate(tarifa = tarifa / 5,37 )
```

# Relembrando conceitos de R e Rstudio

summarize(): sumariza/agrega columnas da base de dados

```
# Calcula a média da variável idade  
# na.rm = TRUE remove os NAs  
df_titanic %>% summarize(mean(idade, na.rm=TRUE))
```

```
## # A tibble: 1 x 1  
##   `mean(idade, na.rm = TRUE)`  
##           <dbl>  
## 1           29.7
```

```
# Calcula: número de mulheres, mediana geral da tarifa e número de passageiros.  
# No caso abaixo a função sum() retorna o número de mulheres.  
# A função n() mostra o número de linhas (em cada grupo) e  
# costuma ser bastante usada com o summarize.  
df_titanic %>%  
  summarize(  
    mulheres = sum(sexo == "feminino", na.rm = TRUE),  
    mediana_tarifa = median(tarifa, na.rm = TRUE),  
    num_passageiros = n()  
  )
```

```
## # A tibble: 1 x 3  
##   mulheres mediana_tarifa num_passageiros  
##   <int>         <dbl>         <int>  
## 1     314           14.5           891
```



# Relembrando conceitos de R e Rstudio

## group\_by() %>% summarize()

```
# Agrupa pela variável sobreviveu e calcula  
# o número de passageiros por grupo (sim/nao).  
df_titanic %>%  
  group_by(sobreviveu) %>%  
  summarize(num_passageiros = n())
```

```
## # A tibble: 2 x 2  
##   sobreviveu num_passageiros  
##   <chr>          <int>  
## 1 nao             549  
## 2 sim             342
```

```
# Agrupa pelo local de embarque e calcula a mediana da tarifa de cada grupo.  
df_titanic %>%  
  group_by(embarque) %>%  
  summarize(media_tarifa = median(tarifa, na.rm = TRUE))
```

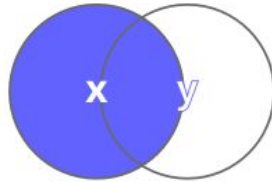
```
## # A tibble: 4 x 2  
##   embarque   media_tarifa  
##   <chr>         <dbl>  
## 1 Cherbourg    29.7  
## 2 Queenstow    7.75  
## 3 Southampton   13  
## 4 <NA>         80
```

# Relembrando conceitos de R e Rstudio

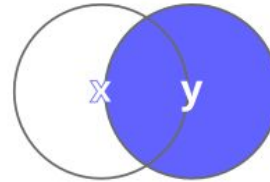
Dplyr::join()

## dplyr joins

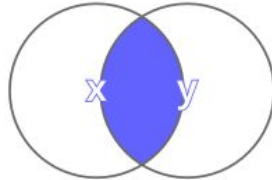
left\_join(x, y)



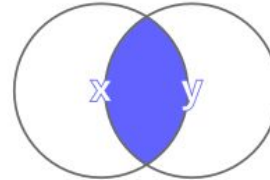
right\_join(x, y)



inner\_join(x, y)

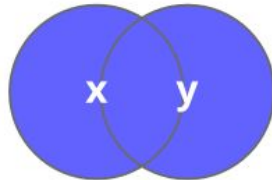


semi\_join(x, y)

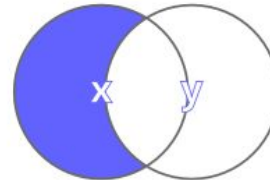


(never duplicate rows of x)

full\_join(x, y)

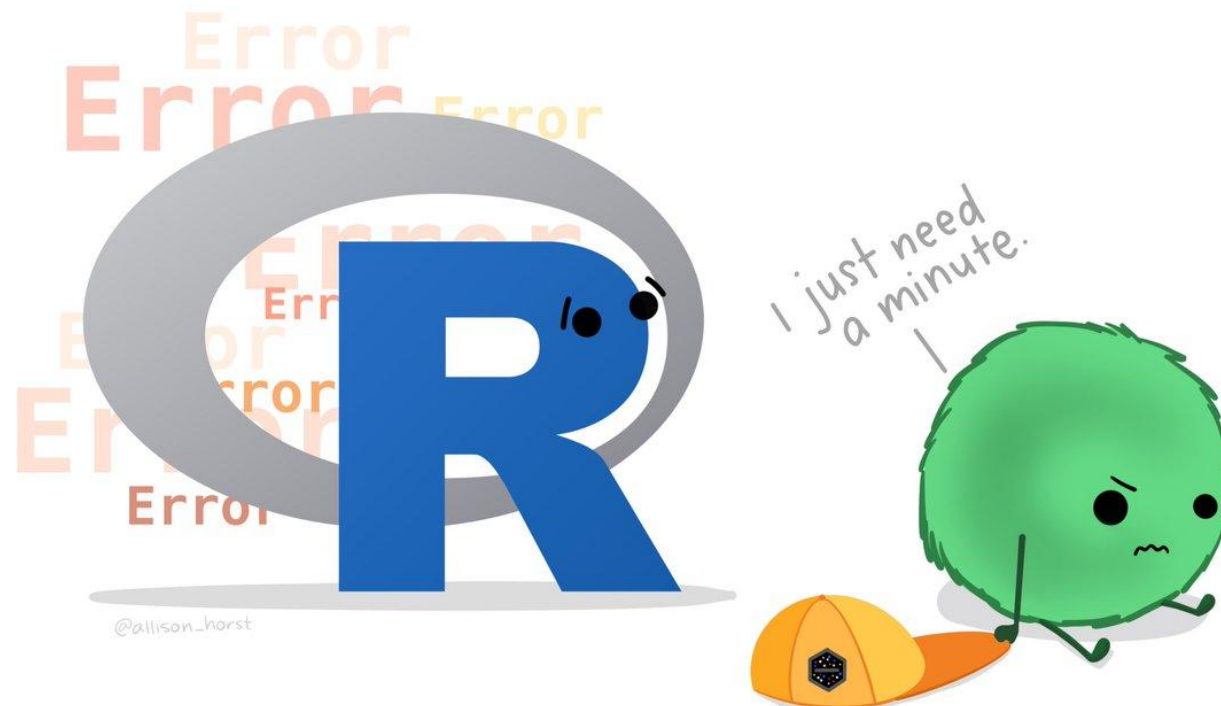


anti\_join(x, y)



# Agora é com vocês!

- Acessem a atividade prática 1 contida na aba da semana 1, baixem o arquivo `revisao_comandos_dplyr.R` e responda os seguintes desafios descritos na atividade. Vocês terão 30min para realizar esta atividade.



# Por que usar o Ggplot2?

Resposta rápida: porque ele é maravilhoso!

- Porque ele permite criar gráficos de forma programática ou automatizada.
- Só pequenas mudanças precisam ser feitas se nossos dados mudarem ou decidirmos mudar o tipo de gráfico.
- Isso ajuda na criação de gráficos com qualidade para publicação feitos a partir de quantidades mínimas de ajustes.

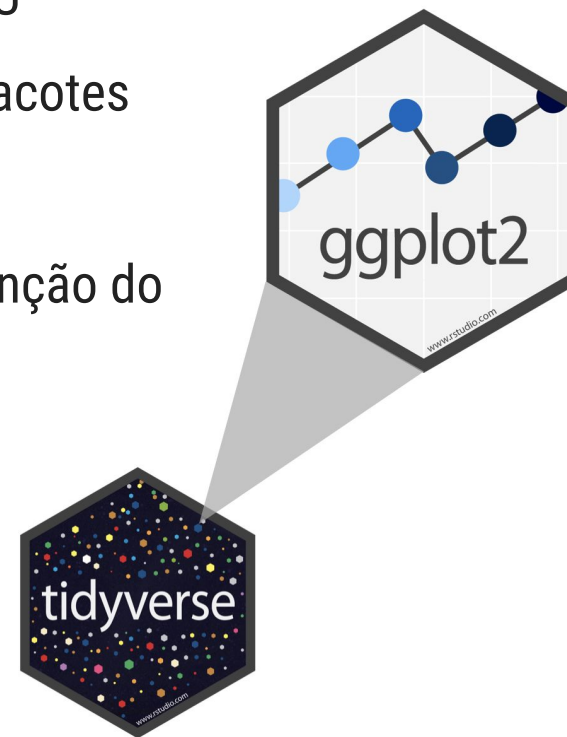


# Por que usar o Ggplot2?

## Como instalar o Ggplot2()?

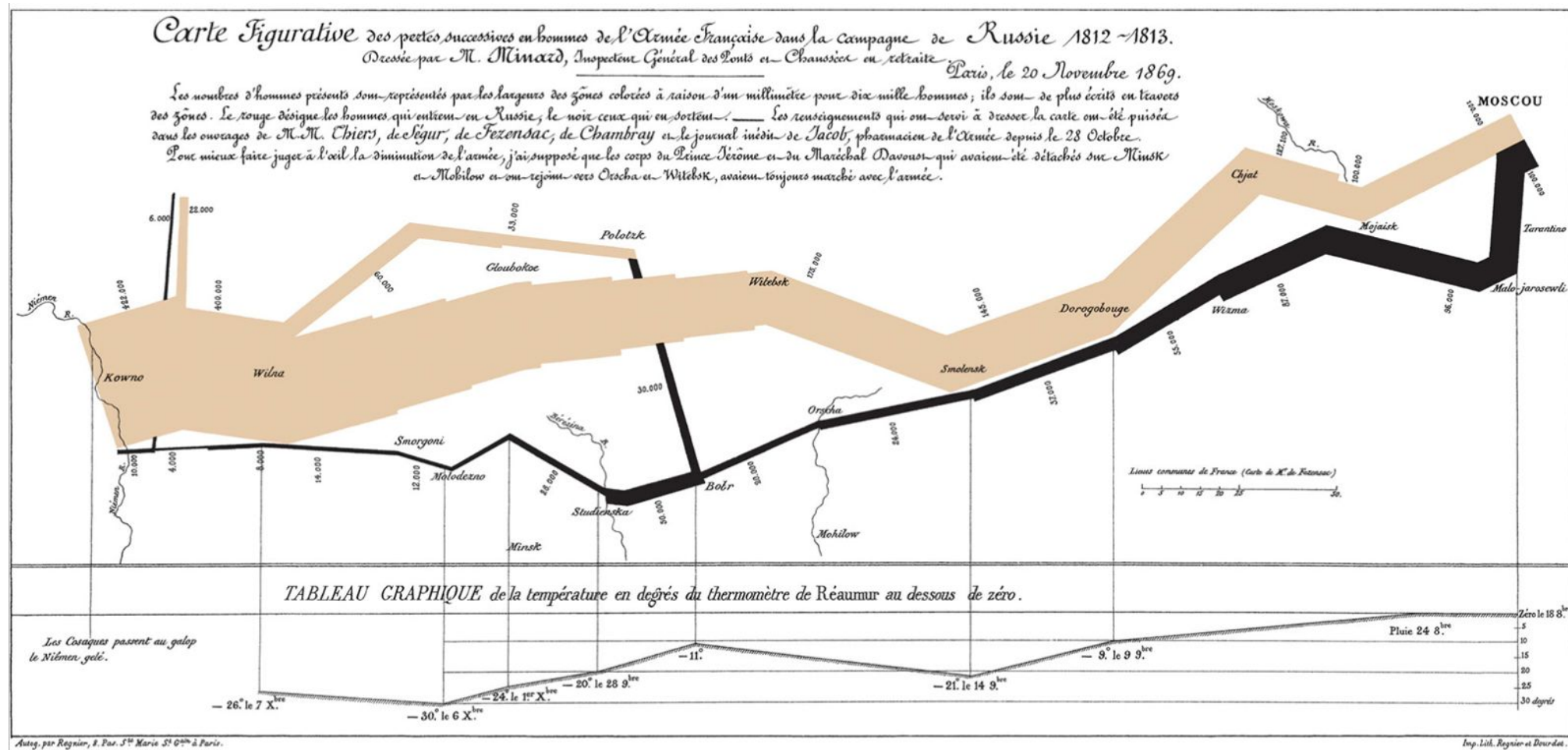
Podemos instalar o pacote Ggplot2 de duas formas:

- Instalando o próprio pacote Ggplot2 com o comando `install.packages("ggplot2")` OU
- Instalando o metapacote do tidyverse, o qual possui o Ggplot2 como um de seus pacotes
- **Dica:** ggplot2 refere-se ao nome do pacote, enquanto ggplot refere-se à principal função do pacote



# Um pouco da história da visualização de dados

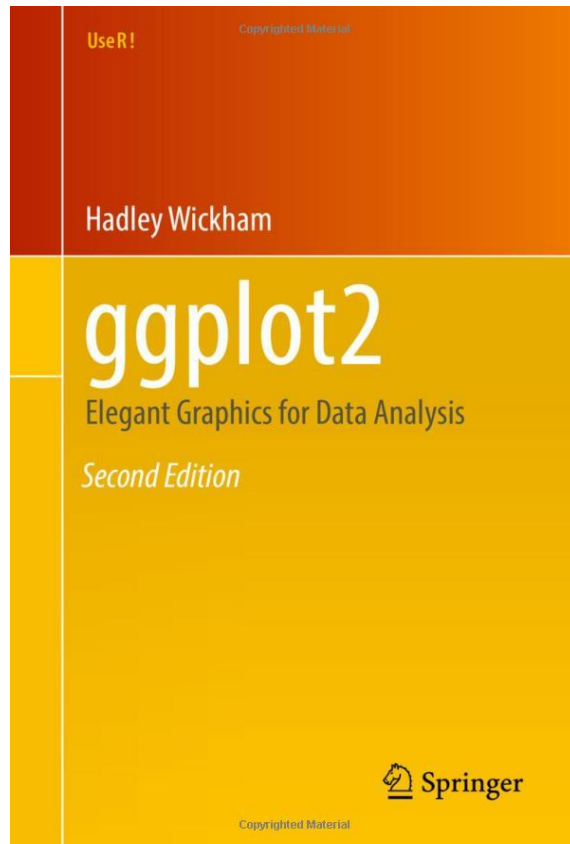
## Gráficos de Minnard





# O conceito de grammar of graphics

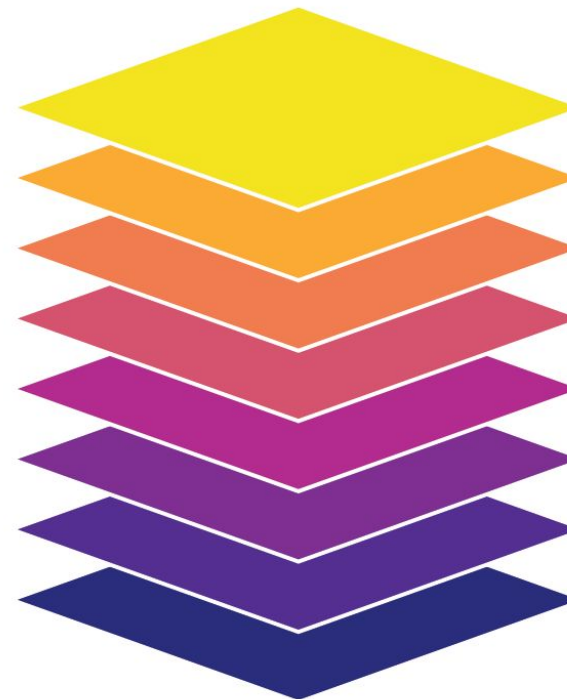
Procura mapear dados aos aspectos estéticos de um gráfico



Implementado  
por Hadley  
Wickham no  
pacote Ggplot2



**Theme**  
**Labels**  
**Coordinates**  
**Facets**  
**Scales**  
**Geometries**  
**Aesthetics**  
**Data**

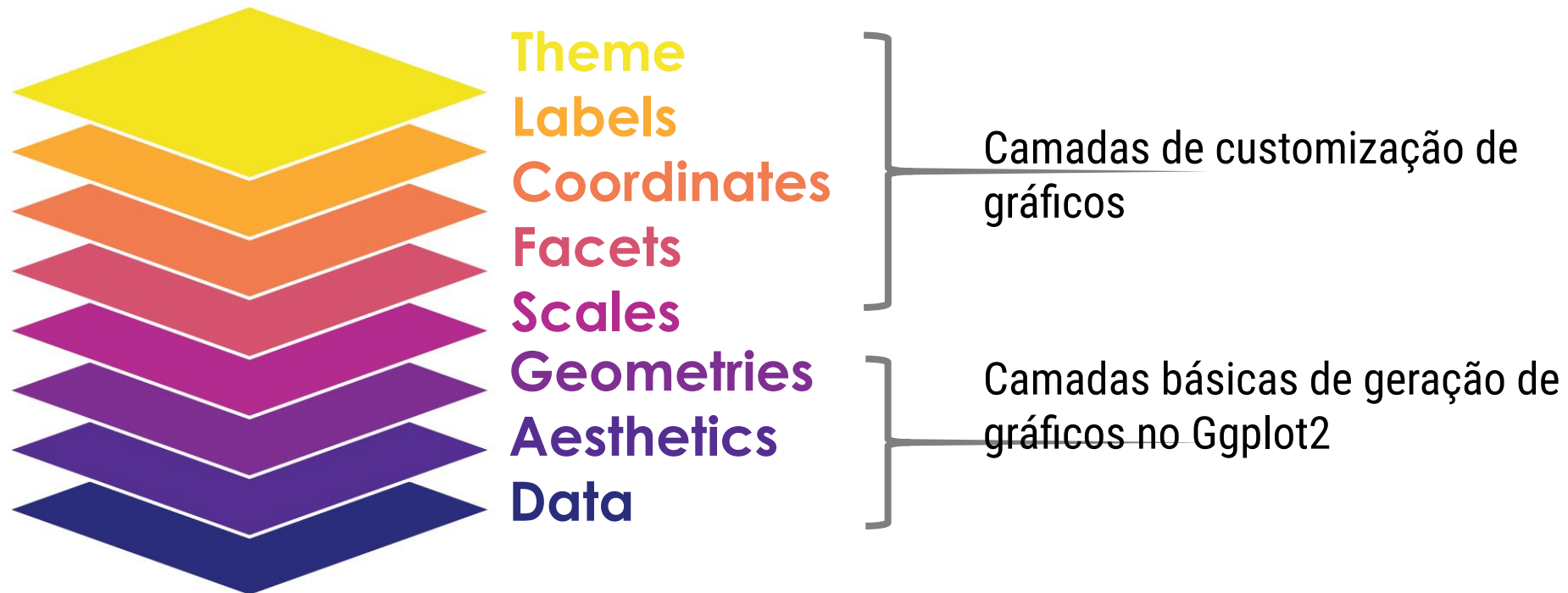






# O conceito de grammar of graphics

Pensando os componentes do gráfico como camadas que se acumulam

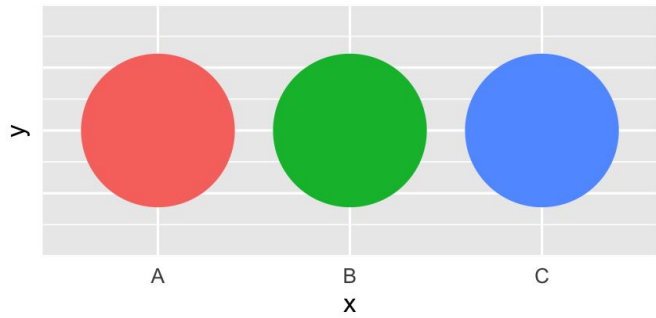


# Aspectos estéticos

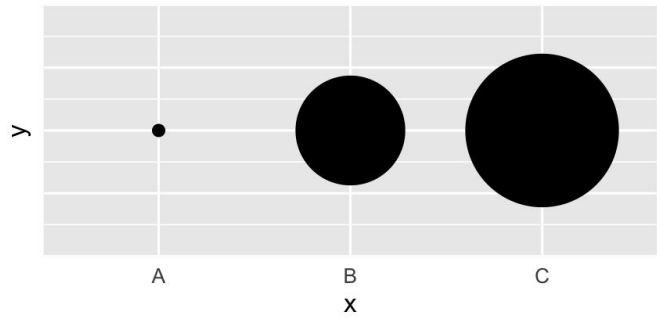
Tem a ver com a forma com que os nossos dados são representados



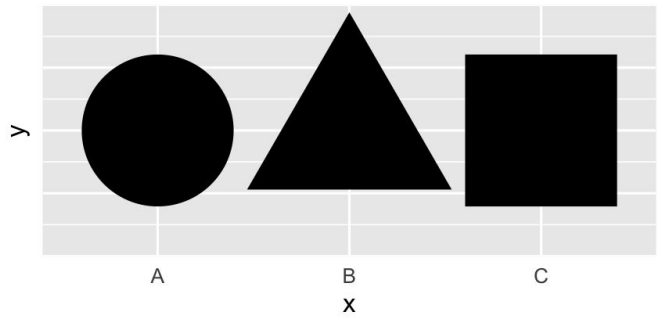
color(discrete)



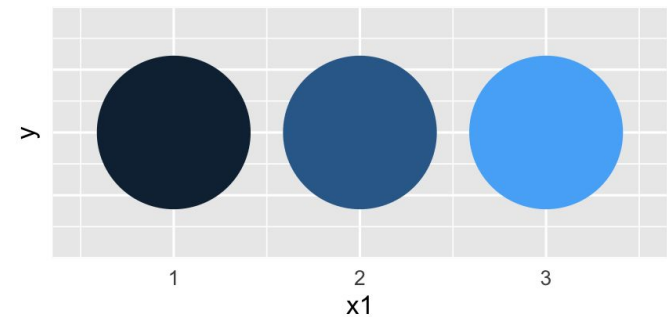
size



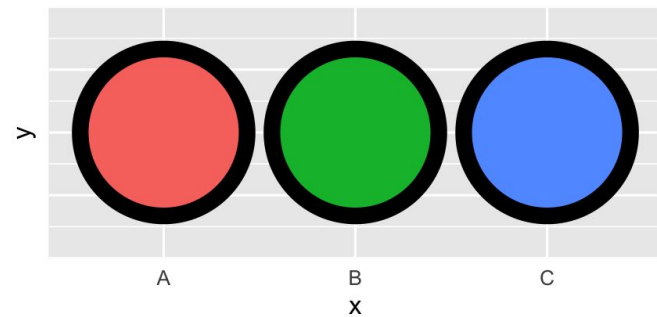
shape



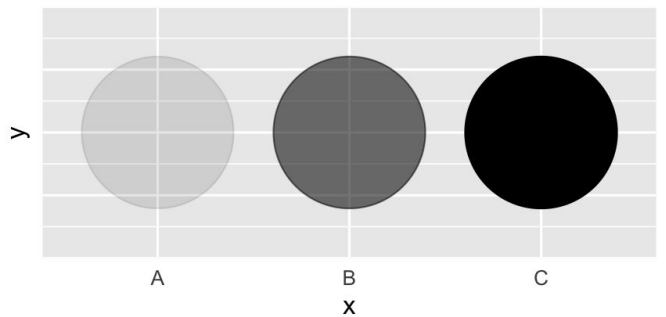
color(continuous)



fill




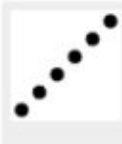


alpha



# Aspectos geométricos

Tem a ver com o tipo de gráfico a ser usado



	Example geom	What it makes
	<code>geom_col()</code>	Bar charts
<code>text</code>	<code>geom_text()</code>	Text
	<code>geom_point()</code>	Points
	<code>geom_boxplot()</code>	Boxplots
	<code>geom_sf()</code>	Maps

# Escalas

Mudam as propriedades das variáveis que estamos mapeando



## Example layer

```
scale_x_continuous()
```

```
scale_x_continuous(breaks = 1:5)
```

```
scale_x_log10()
```

```
scale_color_gradient()
```

```
scale_fill_viridis_d()
```

## What it does

Make the x-axis continuous

Manually specify axis ticks

Log the x-axis

Use a gradient

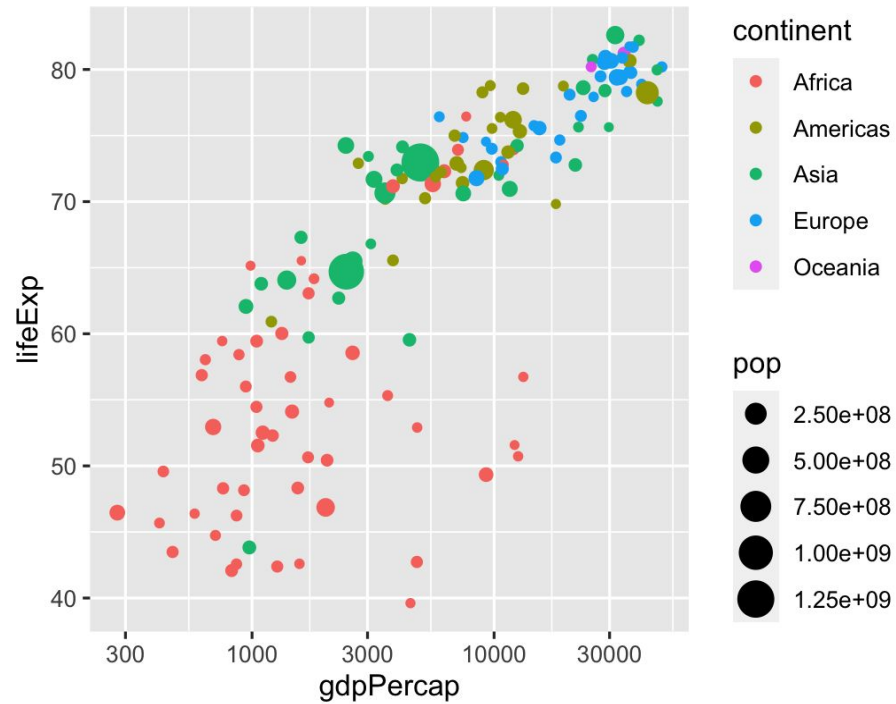
Fill with discrete viridis colors

# Escalas

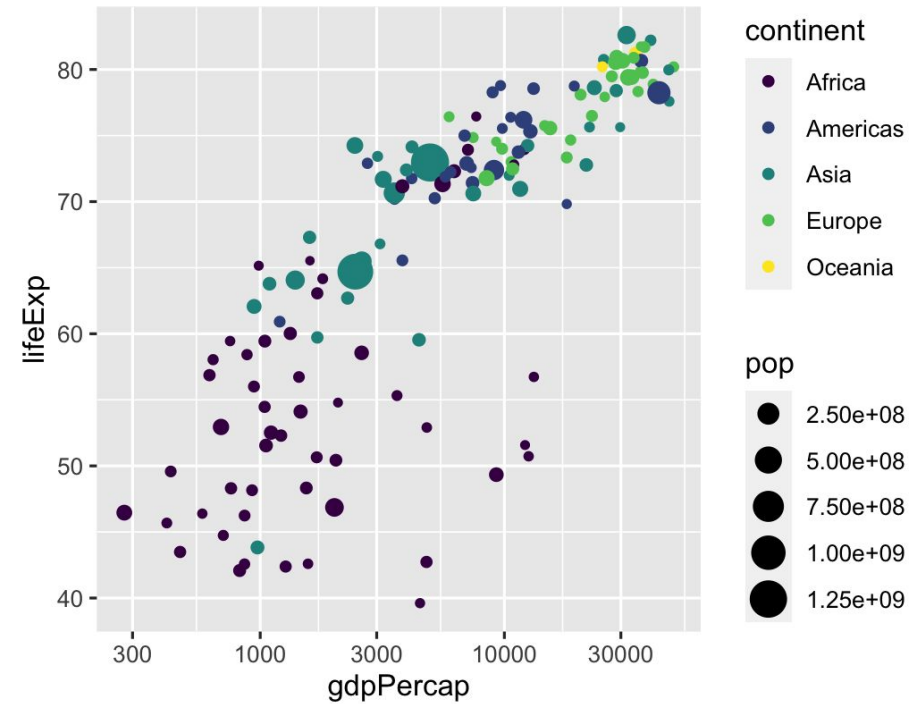
Mudam as propriedades das variáveis que estamos mapeando



`scale_x_log10()`



`scale_color_viridis_d()`



# Faceamento

Facets mostram subplots de acordo com algum tipo de variável



## Example layer

```
facet_wrap(vars(continent))
```

## What it does

Plot for each continent

```
facet_wrap(vars(continent, year))
```

Plot for each continent/year

```
facet_wrap(..., ncol = 1)
```

Put all facets in one column

```
facet_wrap(..., nrow = 1)
```

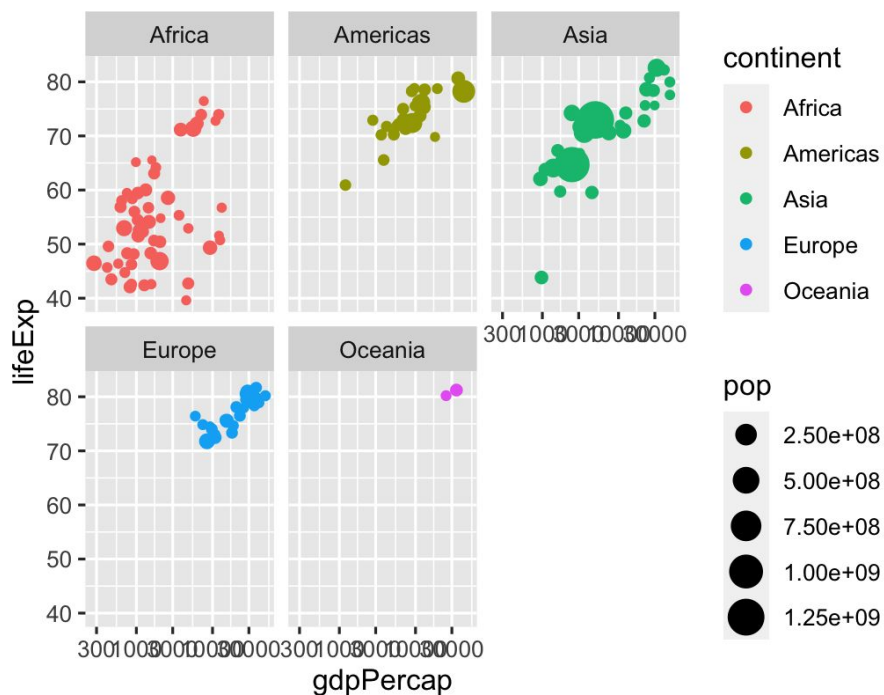
Put all facets in one row

# Faceamento

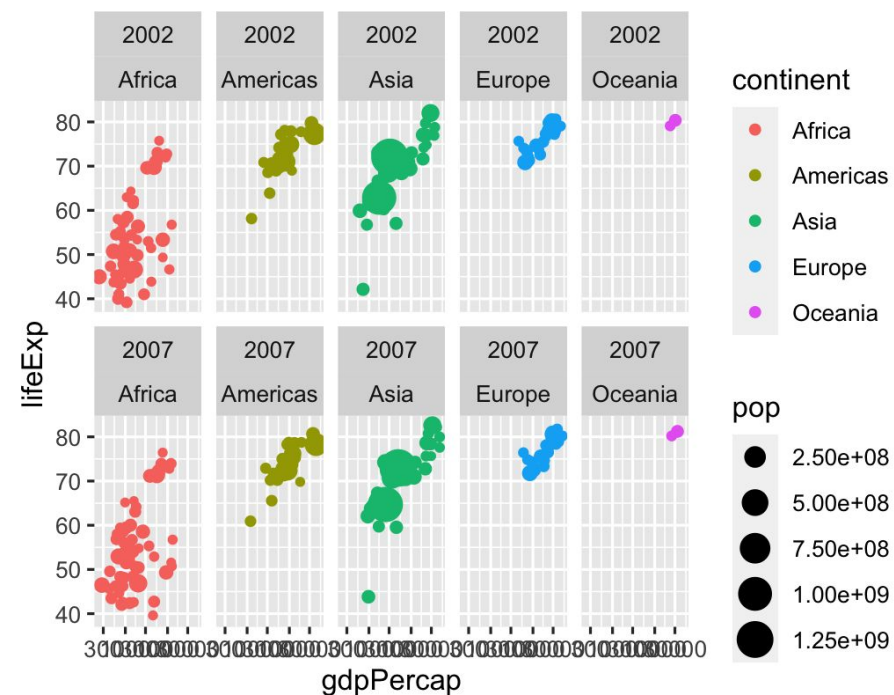
Facets mostram subplots de acordo com algum tipo de variável



`facet_wrap(vars(continent))`



`facet_wrap(vars(continente, year))`





# Coordenadas

Mudam o sistema de coordenadas



## Example layer

```
coord_cartesian()
```

```
coord_cartesian(ylim = c(1, 10))
```

```
coord_flip()
```

```
coord_polar()
```

## What it does

Plot for each continent

Zoom in where y is 1–10

Switch x and y

Use circular polar system

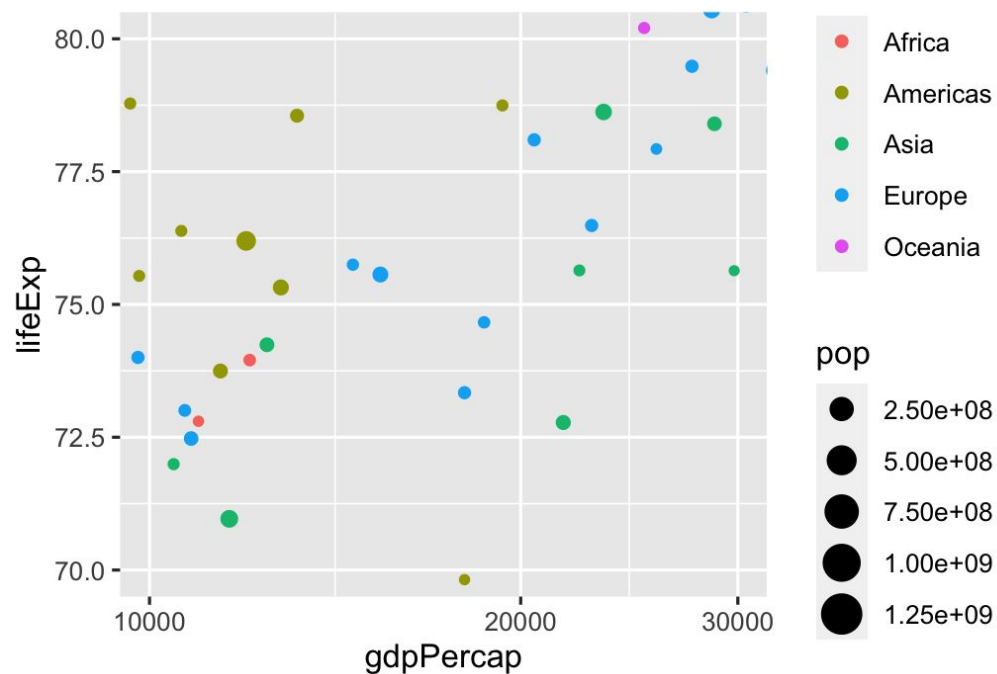


# Coordenadas

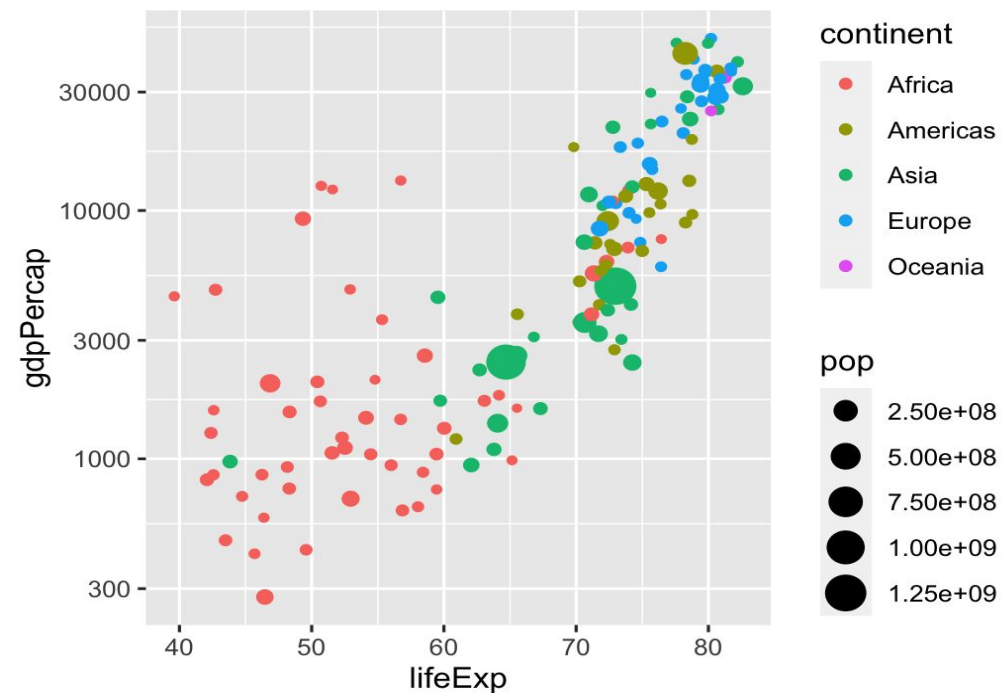
Mudam o sistema de coordenadas



```
coord_cartesian(ylim = c(70, 80),  
               xlim = c(10000, 30000))
```



```
coord_flip()
```



# Legendas

Adiciona legendas, títulos e subtítulos

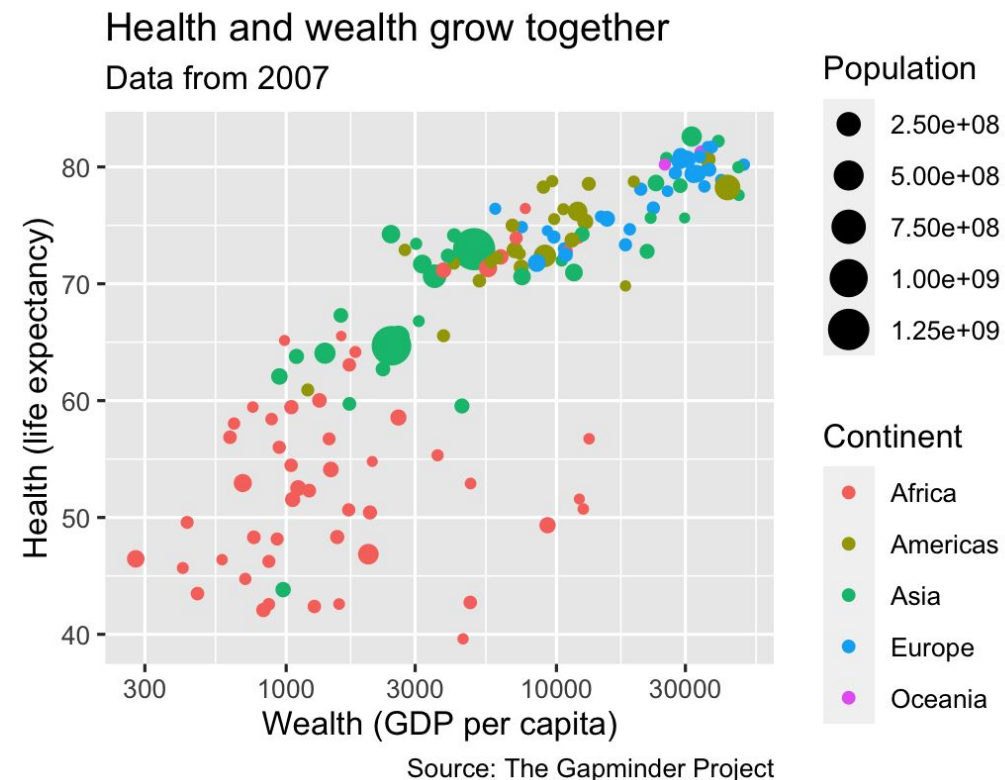
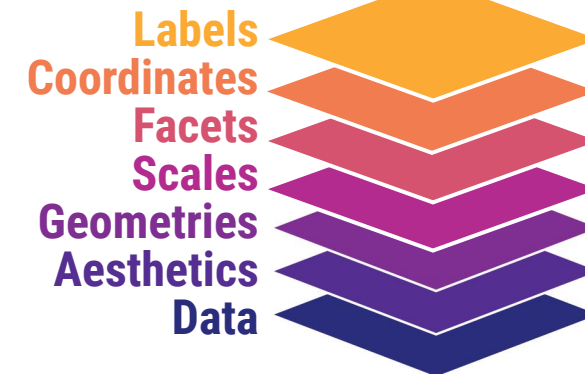


Example layer	What it does
<code>labs(title = "Neat title")</code>	Title
<code>labs(caption = "Something")</code>	Caption
<code>labs(y = "Something")</code>	y-axis
<code>labs(size = "Population")</code>	Title of size legend

# Legendas

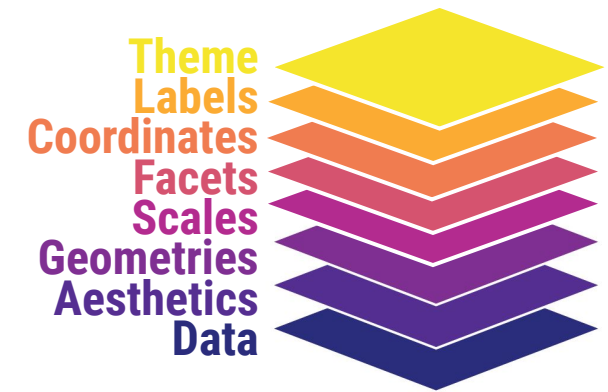
Adiciona legendas, títulos e subtítulos

```
ggplot(gapminder_2007,  
       aes(x = gdpPercap, y = lifeExp,  
           color = continent, size = pop)) +  
  geom_point() +  
  scale_x_log10() +  
  labs(title = "Health and wealth grow together",  
        subtitle = "Data from 2007",  
        x = "Wealth (GDP per capita)",  
        y = "Health (life expectancy)",  
        color = "Continent",  
        size = "Population",  
        caption = "Source: The Gapminder Project")
```



# Temas

Muda a aparência de vários itens do gráfico.



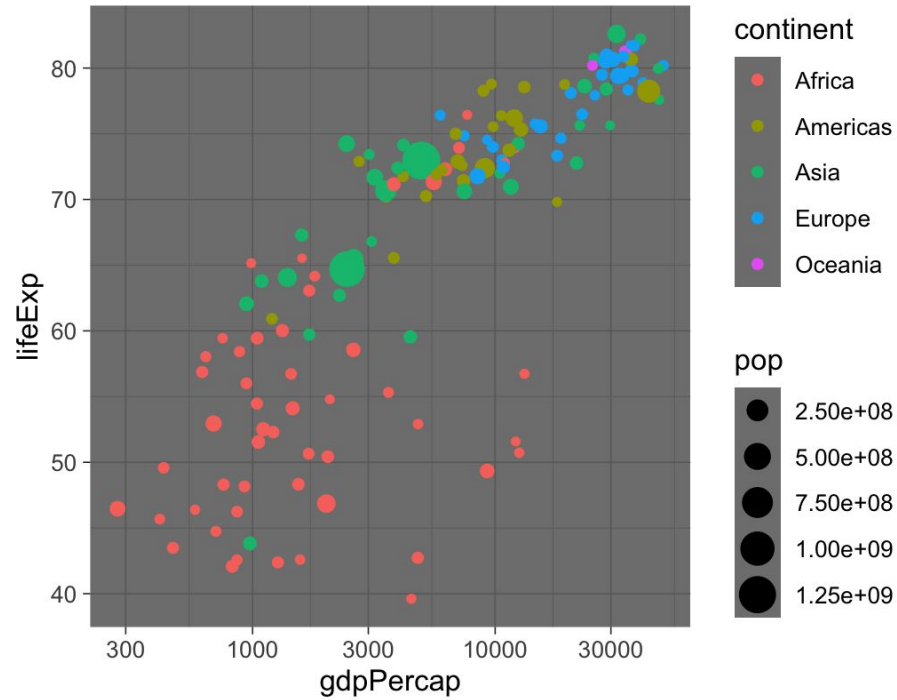
<b>Example layer</b>	<b>What it does</b>
<code>theme_grey()</code>	Default grey background
<code>theme_bw()</code>	Black and white
<code>theme_dark()</code>	Dark
<code>theme_minimal()</code>	Minimal

# Temas

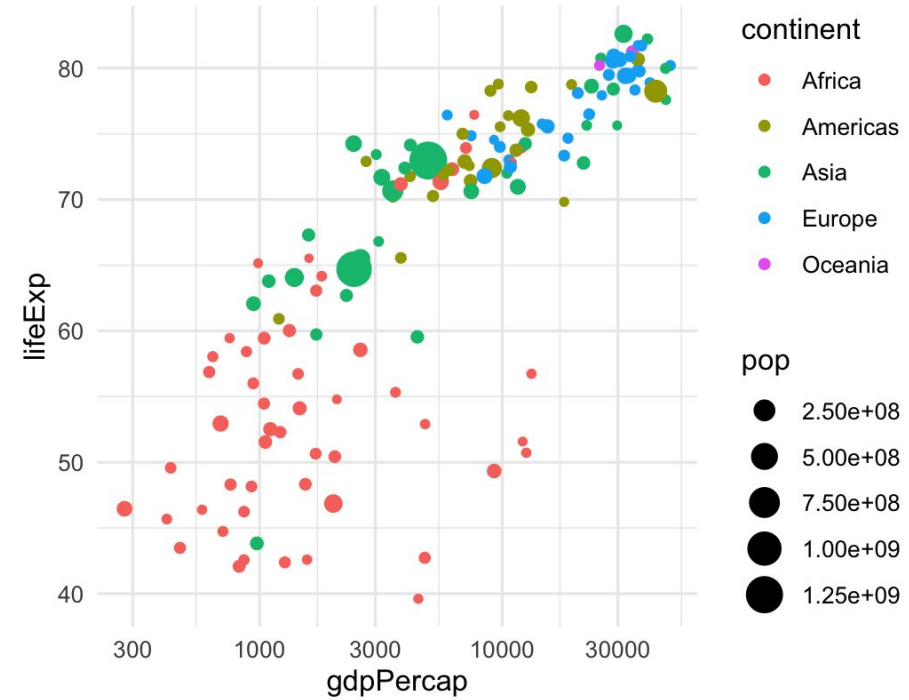
Muda a aparência de vários itens do gráfico.



`theme_dark()`

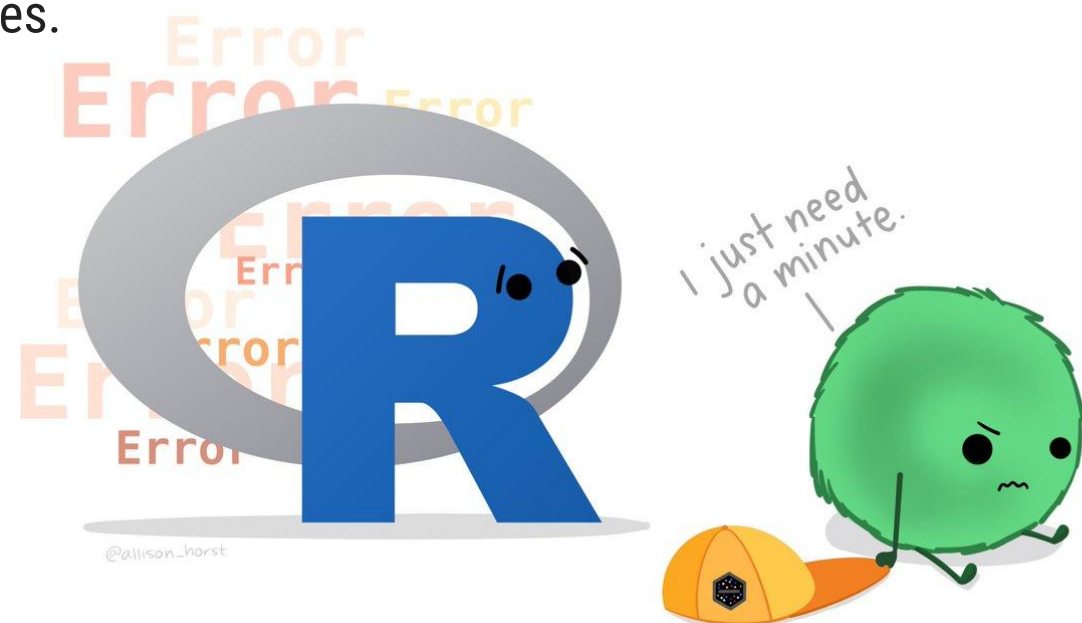


`theme_minimal()`



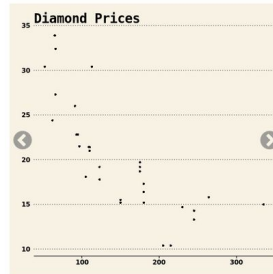
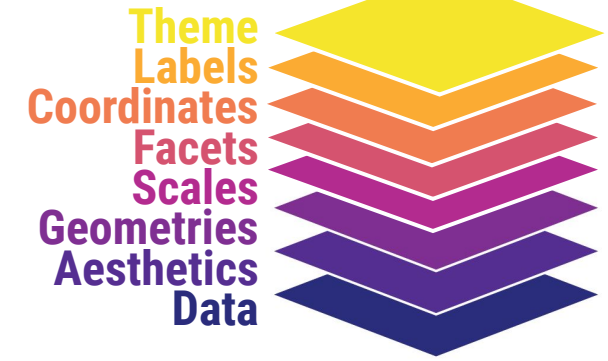
# Agora é com vocês!

- Acessem o quizz 1 contido na aba da semana 1, e respondam as questões de múltipla escolha.
- Uma vez enviado, não é possível enviar novamente.
- Vocês podem consultar a internet para responder as questões.



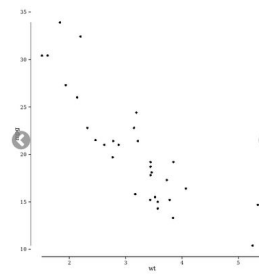
# Temas

Há coleções de temas pré-construídos disponíveis em pacotes que funcionam como extensões do Ggplot2, como o Ggthemes.



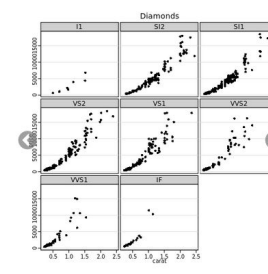
**theme\_wsj**

Wall Street Journal theme



**theme\_tufte**

Tufte Maximal Data, Minimal Ink Theme



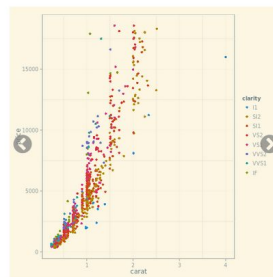
**theme\_stata**

Themes based on Stata graph schemes



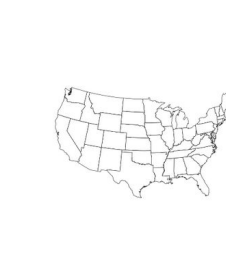
**theme\_solid**

Theme with nothing other than a background color



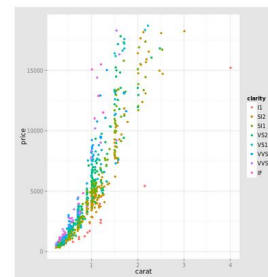
**theme\_solarized**

ggplot color themes based on the Solarized palette



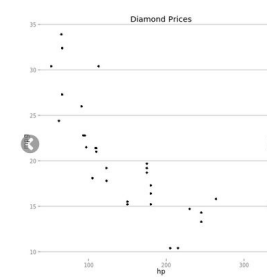
**theme\_map**

Clean theme for maps



**theme\_igray**

Inverse gray theme



**theme\_hc**

Highcharts JS theme

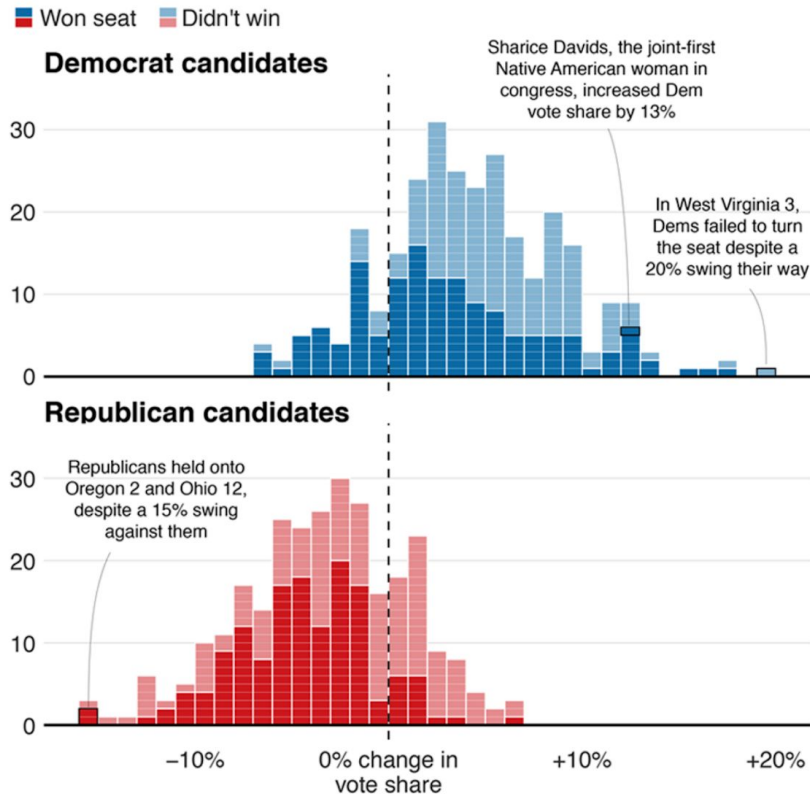


# Temas

Há organizações que fazem os seus próprios temas, como a BBC.



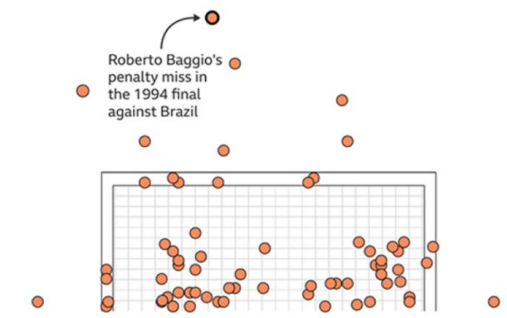
## Blue wave



Source: AP, 19:01 ET

## Where penalties are saved

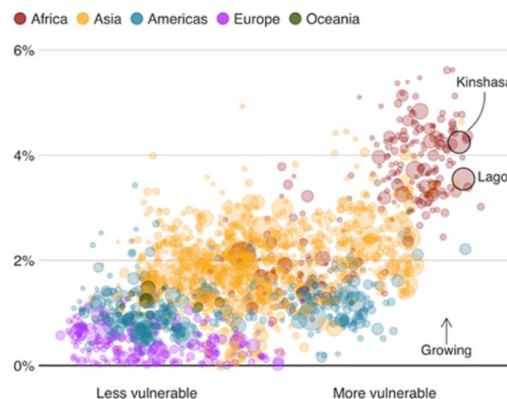
World Cup shootout misses and saves, 1982-2014



Source: Opta

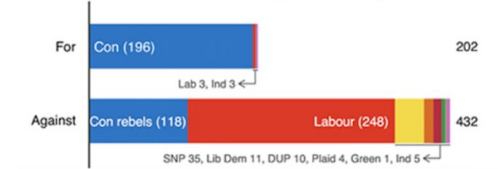
## Fast-growing cities face worse climate risks

Population growth 2018-2035 over climate change vulnerability



Source: Verisk Maplecroft. Circle size represents current population.

## MPs rejected Theresa May's deal by 230 votes



Source: Commons Votes Services. Excludes 'tellers', the Speaker and deputies

## Earnings vary across units even within subjects

Impact on men's earnings relative to the average degree



Source: Institute for Fiscal Studies



# Temas

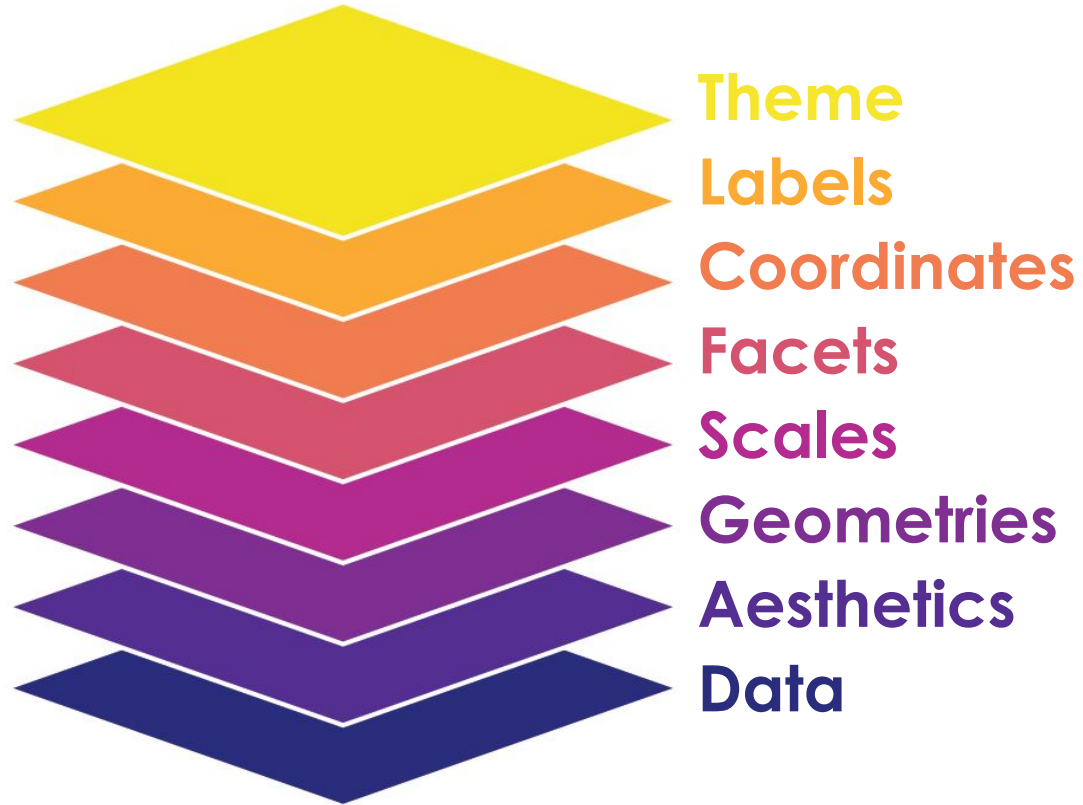
Também podemos fazer ajustes a partir de algum tema, para isso usamos a função `theme()`



```
theme_bw() +  
theme(legend.position = "bottom",  
      plot.title = element_text(face =  
"bold"),  
      panel.grid = element_blank(),  
      axis.title.y = element_text(face =  
"italic"))
```

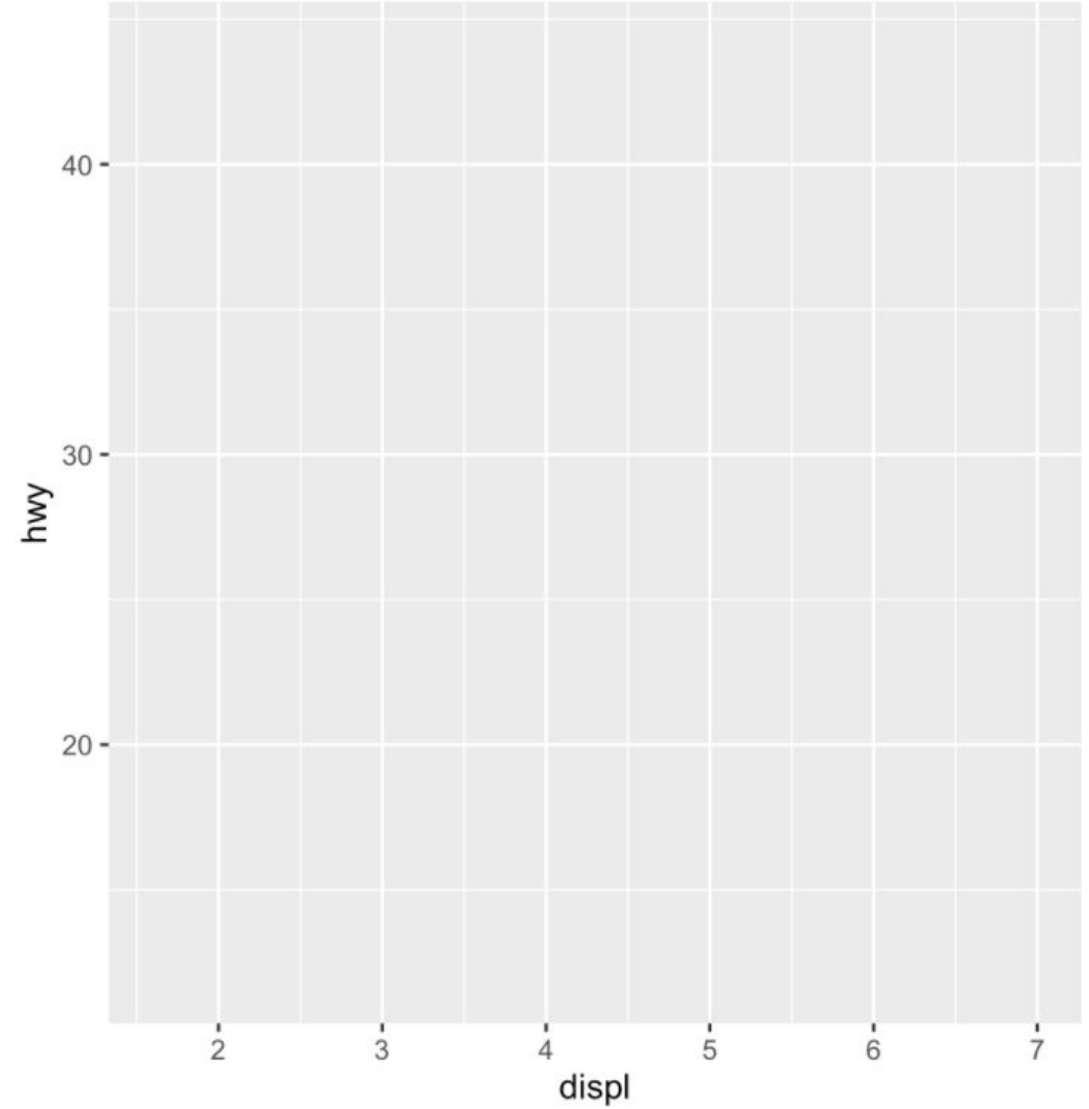
# Juntando tudo...

Passo-a-passo de construção de um gráfico



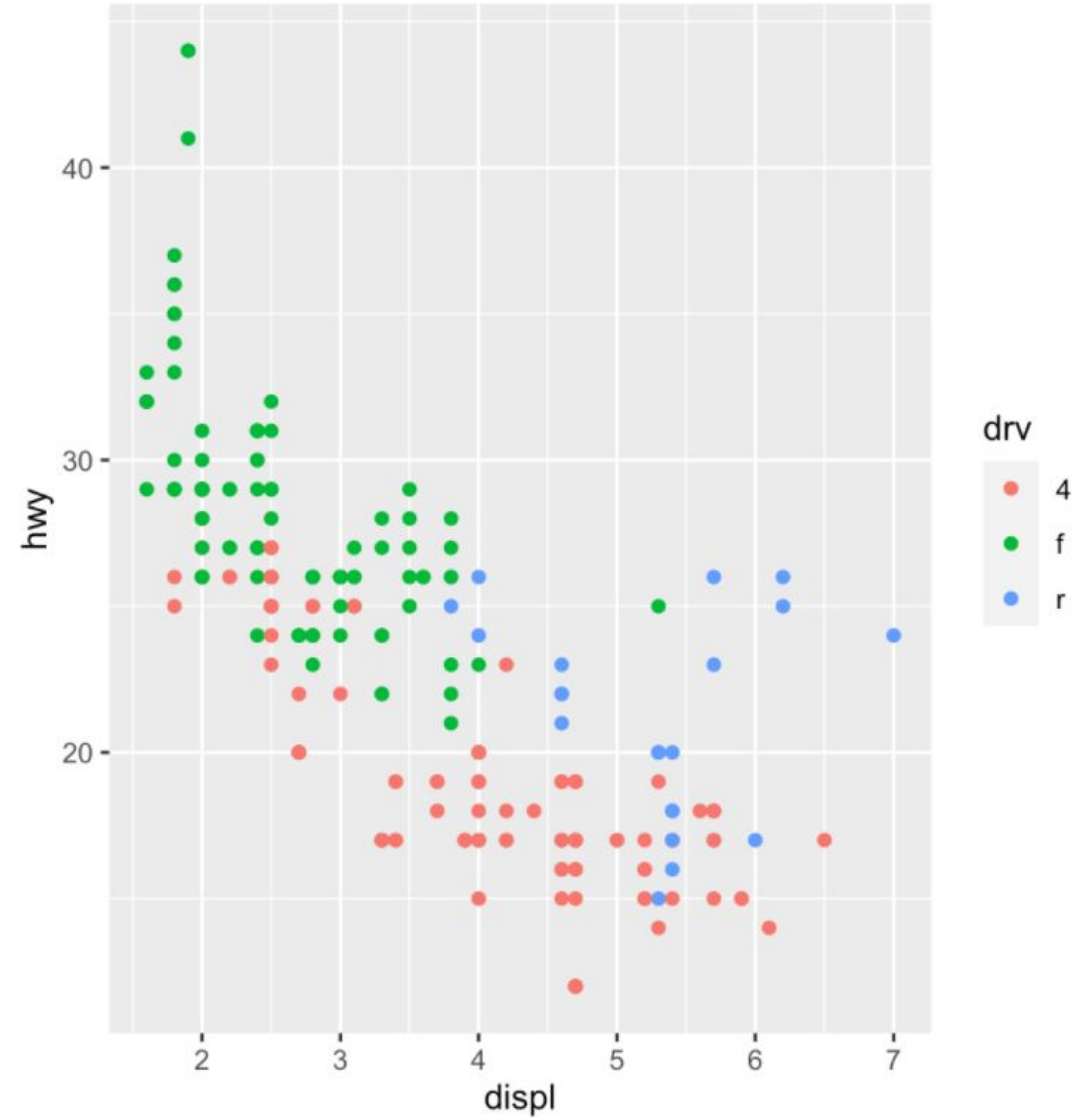
# Juntando tudo...

```
ggplot(data = mpg,  
       mapping = aes(x = displ,  
                     y = hwy,  
                     color = drv))
```



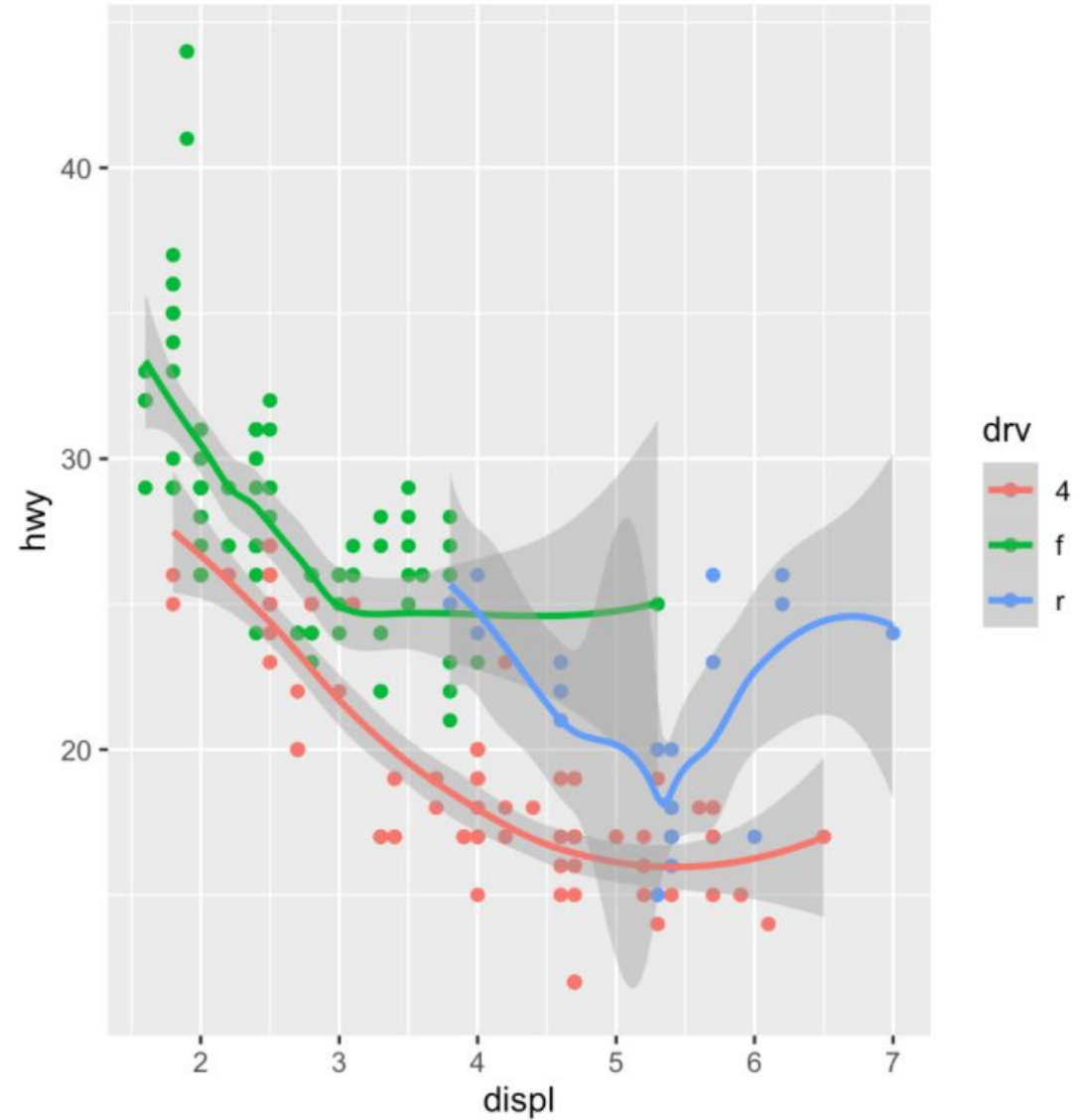
# Juntando tudo...

```
ggplot(data = mpg,  
       mapping = aes(x = displ,  
                     y = hwy,  
                     color = drv)) +  
geom_point()
```



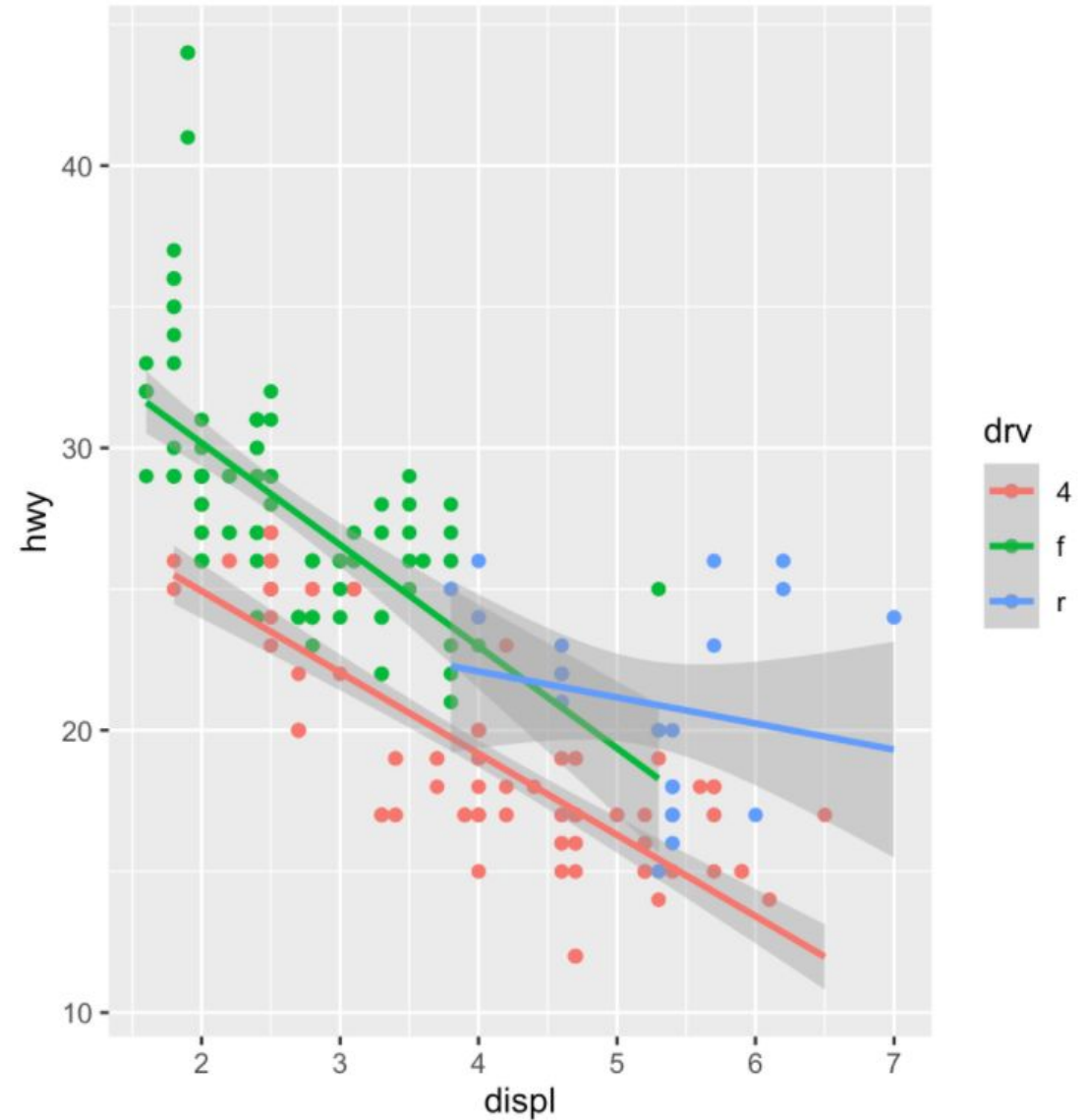
# Juntando tudo...

```
ggplot(data = mpg,  
       mapping = aes(x = displ,  
                     y = hwy,  
                     color = drv)) +  
  geom_point() +  
  geom_smooth()
```



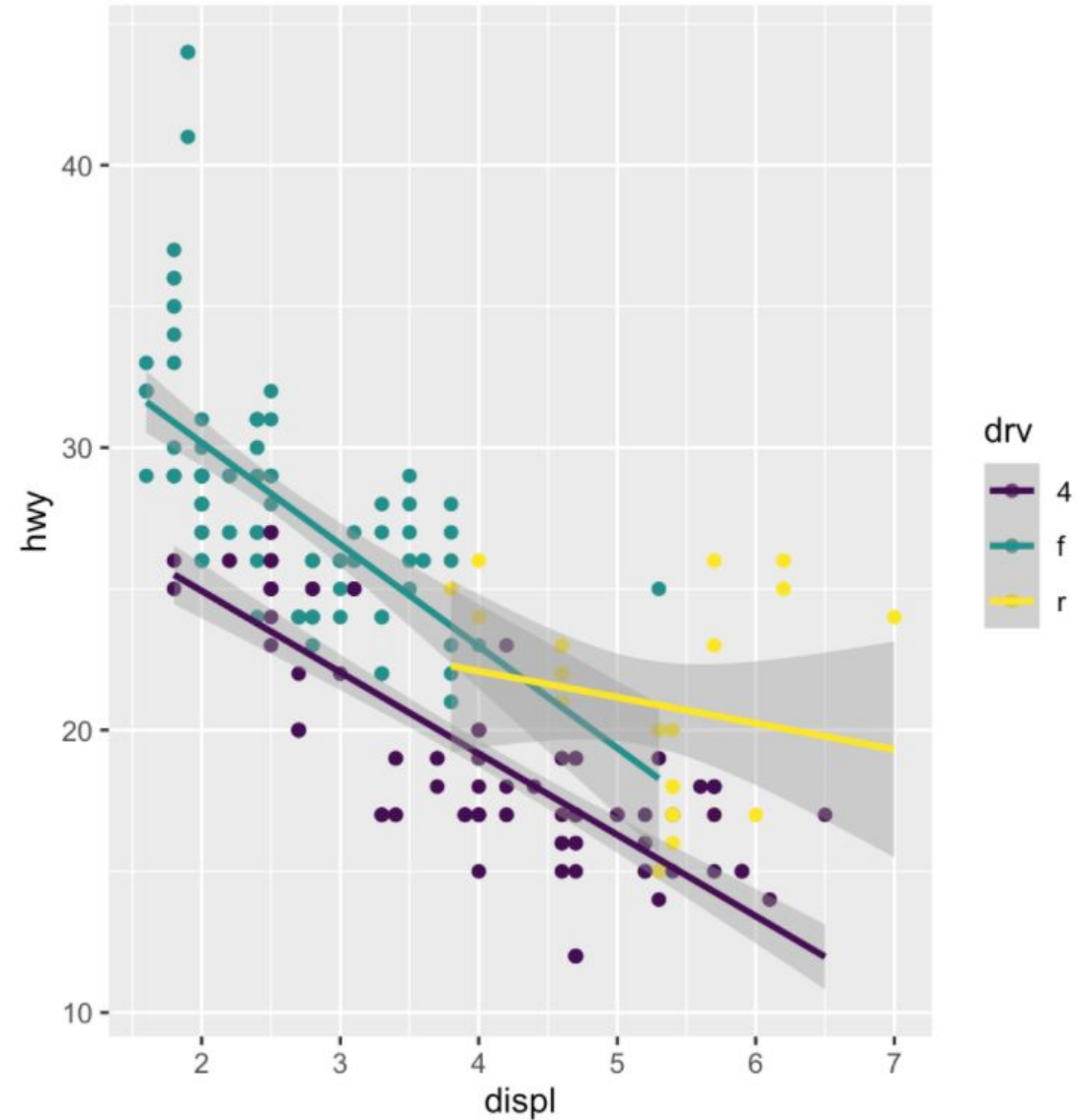
# Juntando tudo...

```
ggplot(data = mpg,  
       mapping = aes(x = displ,  
                     y = hwy,  
                     color = drv)) +  
  geom_point() +  
  geom_smooth(method = "lm")
```



# Juntando tudo...

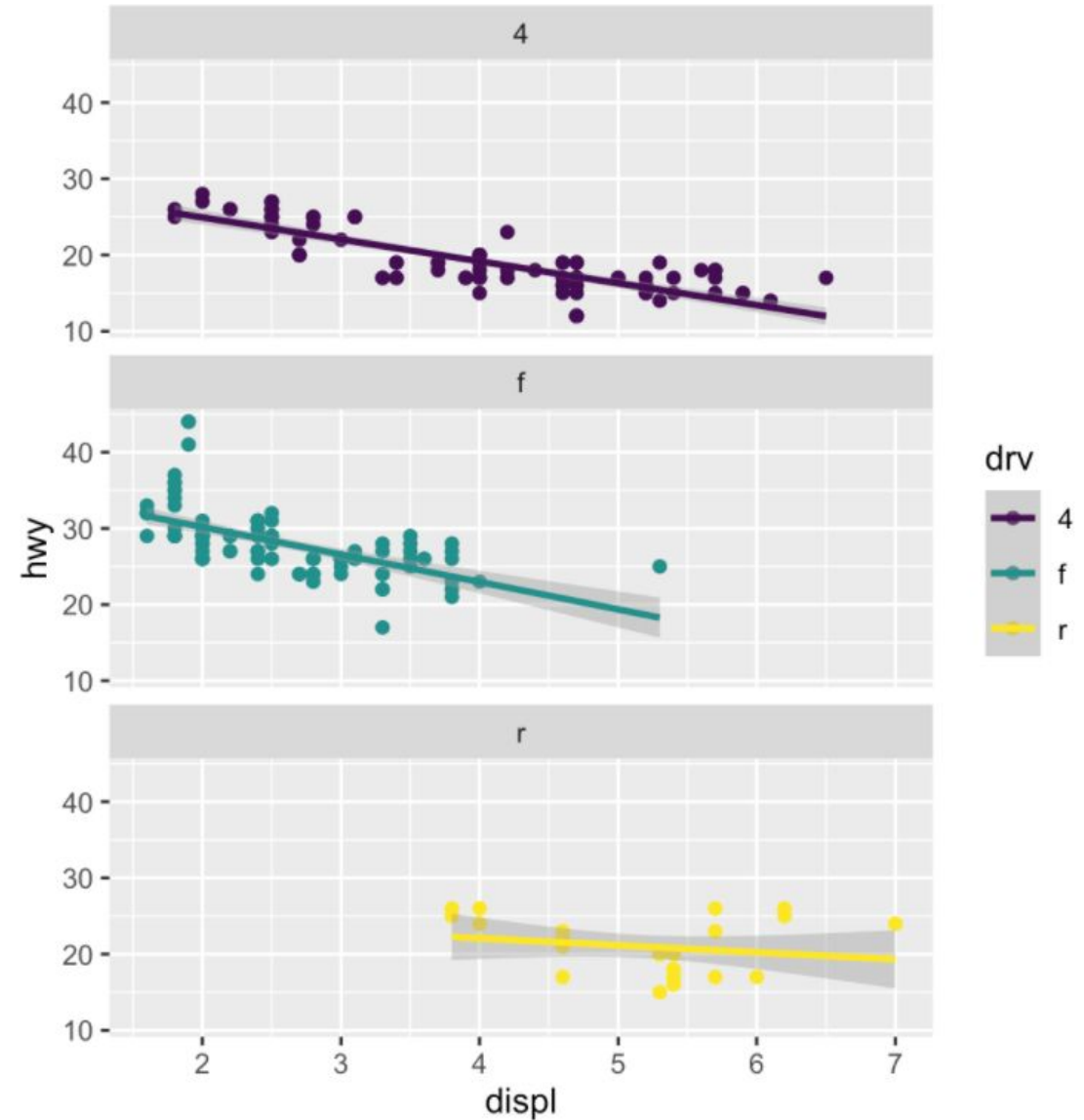
```
ggplot(data = mpg,  
       mapping = aes(x = displ,  
                     y = hwy,  
                     color = drv)) +  
  geom_point() +  
  geom_smooth(method = "lm") +  
  scale_color_viridis_d()
```





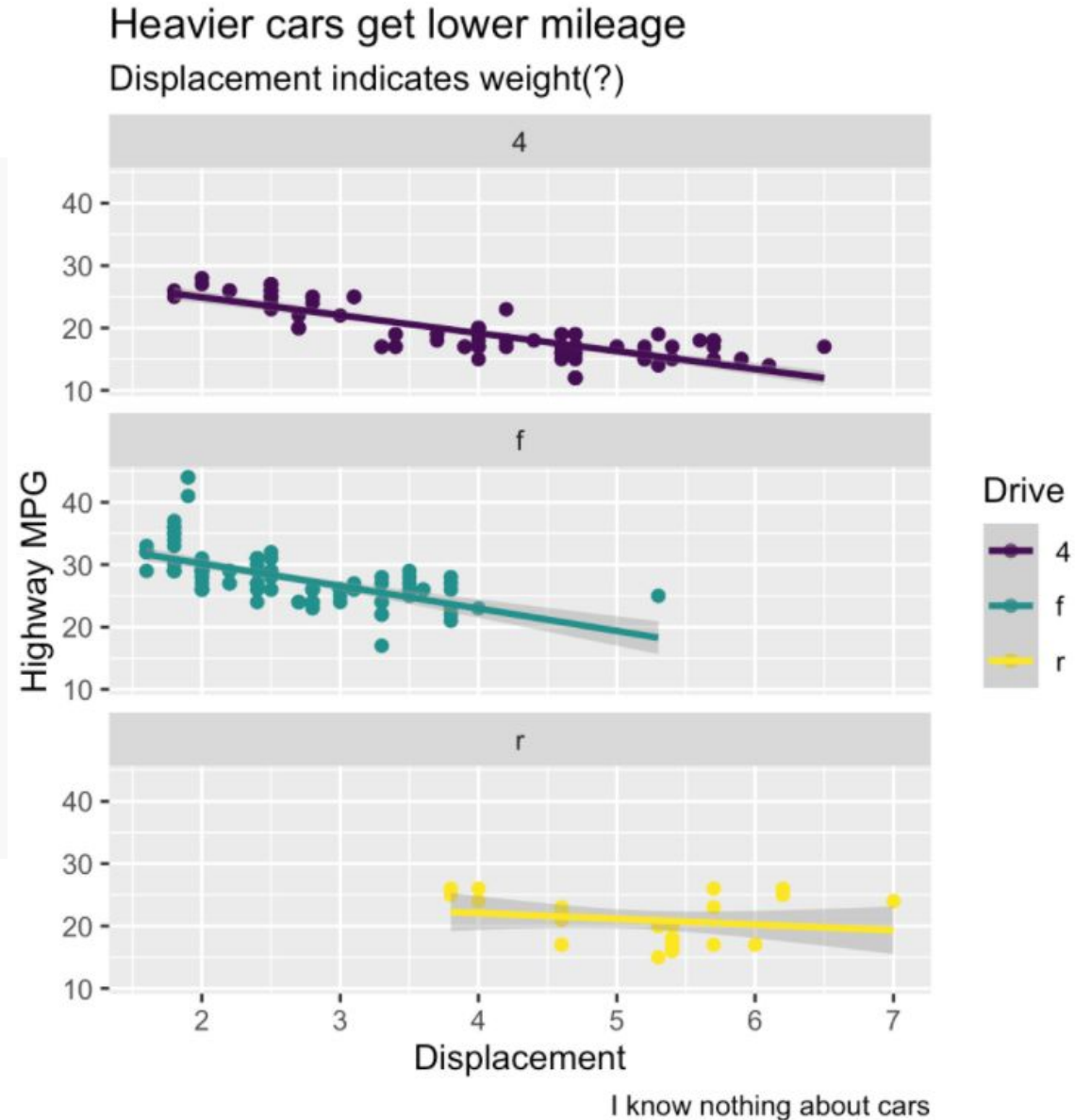
# Juntando tudo...

```
ggplot(data = mpg,  
       mapping = aes(x = displ,  
                     y = hwy,  
                     color = drv)) +  
  geom_point() +  
  geom_smooth(method = "lm") +  
  scale_color_viridis_d() +  
  facet_wrap(vars(drv), ncol = 1)
```



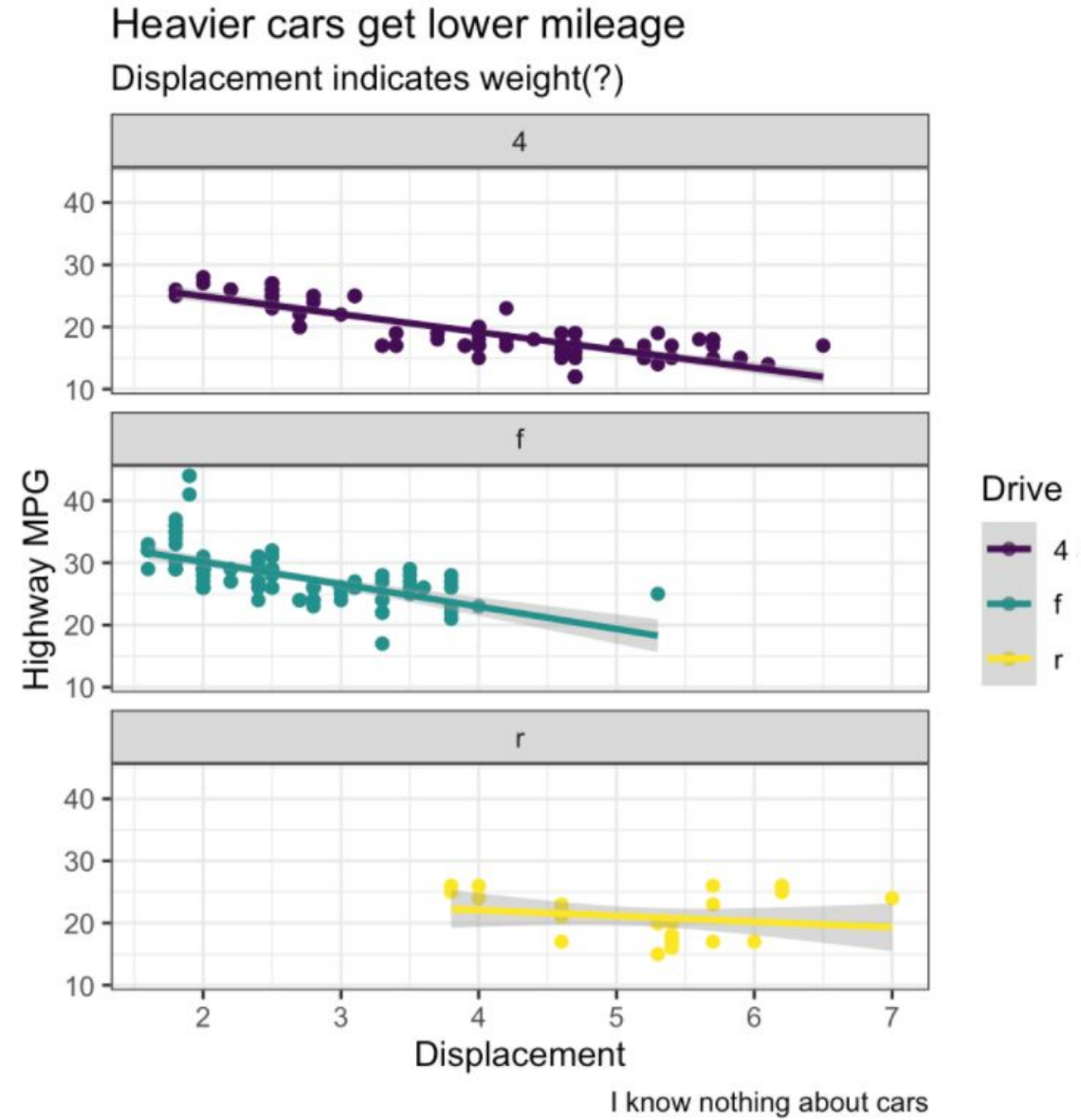
# Juntando tudo...

```
ggplot(data = mpg,  
mapping = aes(x = displ,  
y = hwy,  
color = drv)) +  
geom_point() +  
geom_smooth(method = "lm") +  
scale_color_viridis_d() +  
facet_wrap(vars(drv), ncol = 1) +  
labs(x = "Displacement", y = "Highway MPG",  
color = "Drive",  
title = "Heavier cars get lower mileage",  
subtitle = "Displacement indicates weight?",  
caption = "I know nothing about cars")
```



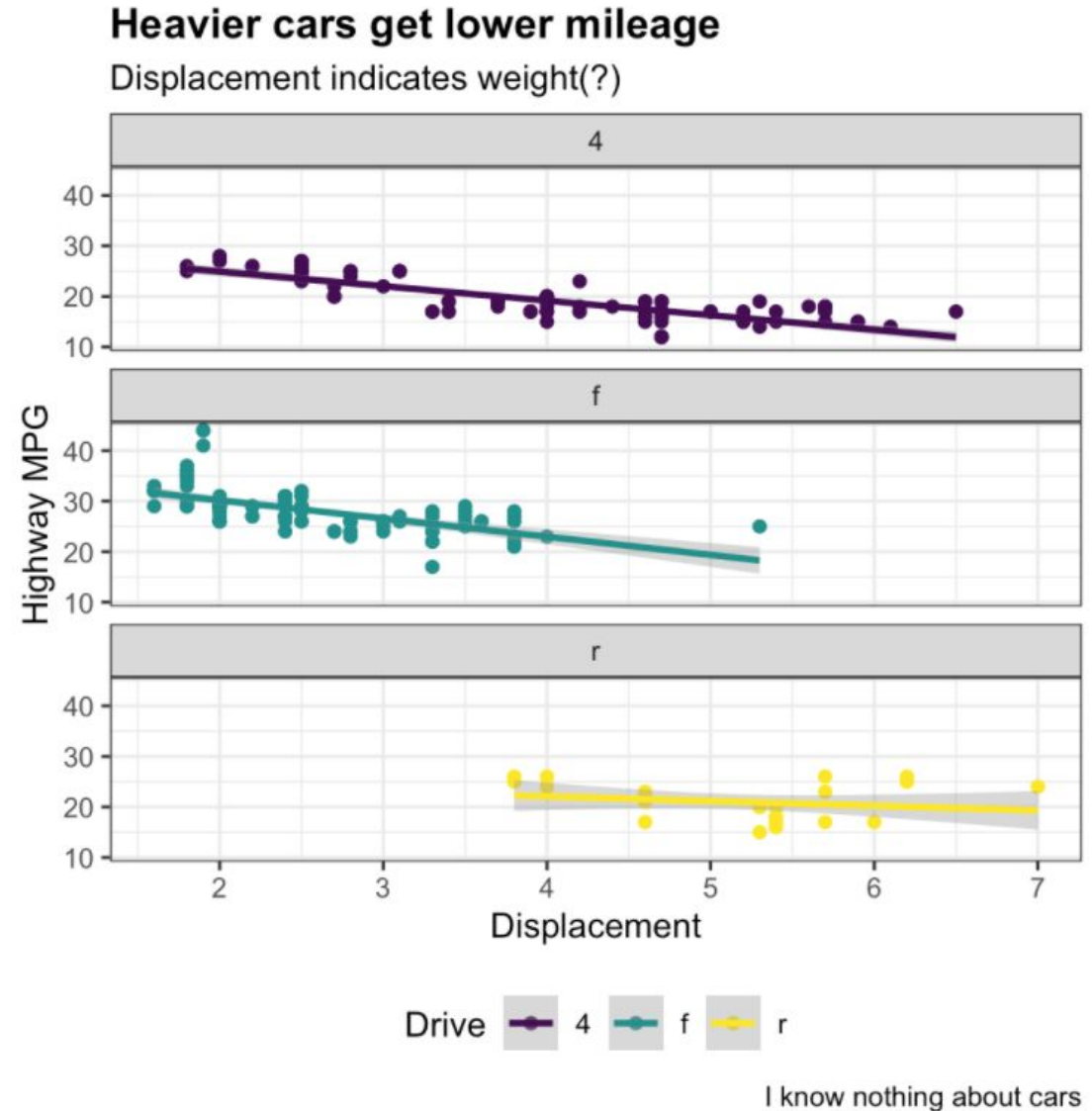
# Juntando tudo...

```
ggplot(data = mpg,  
mapping = aes(x = displ,  
y = hwy,  
color = drv)) +  
geom_point() +  
geom_smooth(method = "lm") +  
scale_color_viridis_d() +  
facet_wrap(vars(drv), ncol = 1) +  
labs(x = "Displacement", y = "Highway MPG",  
color = "Drive",  
title = "Heavier cars get lower mileage",  
subtitle = "Displacement indicates weight(?)",  
caption = "I know nothing about cars") +  
theme_bw()
```



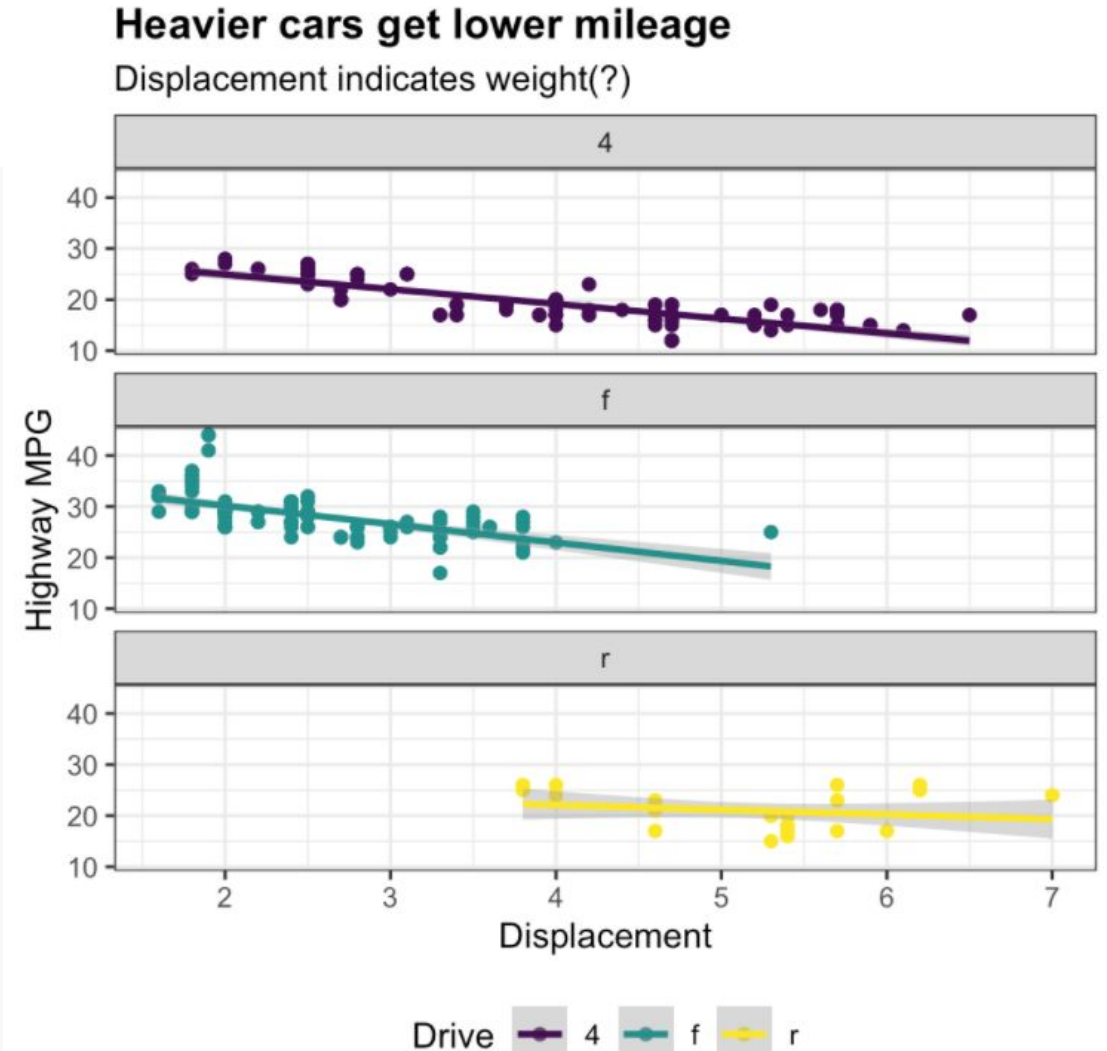
# Juntando tudo...

```
ggplot(data = mpg,  
mapping = aes(x = displ,  
y = hwy,  
color = drv)) +  
geom_point() +  
geom_smooth(method = "lm") +  
scale_color_viridis_d() +  
facet_wrap(vars(drv), ncol = 1) +  
labs(x = "Displacement", y = "Highway MPG",  
color = "Drive",  
title = "Heavier cars get lower mileage",  
subtitle = "Displacement indicates weight(?)",  
caption = "I know nothing about cars") +  
theme_bw() +  
theme(legend.position = "bottom",  
plot.title = element_text(face = "bold"))
```



# Juntando tudo...

```
ggplot(data = mpg,
mapping = aes(x = displ,
y = hwy,
color = drv)) +
geom_point() +
geom_smooth(method = "lm") +
scale_color_viridis_d() +
facet_wrap(vars(drv), ncol = 1) +
labs(x = "Displacement", y = "Highway MPG",
color = "Drive",
title = "Heavier cars get lower mileage",
subtitle = "Displacement indicates weight(?)",
caption = "I know nothing about cars") +
theme_bw() +
theme(legend.position = "bottom",
plot.title = element_text(face = "bold"))
```

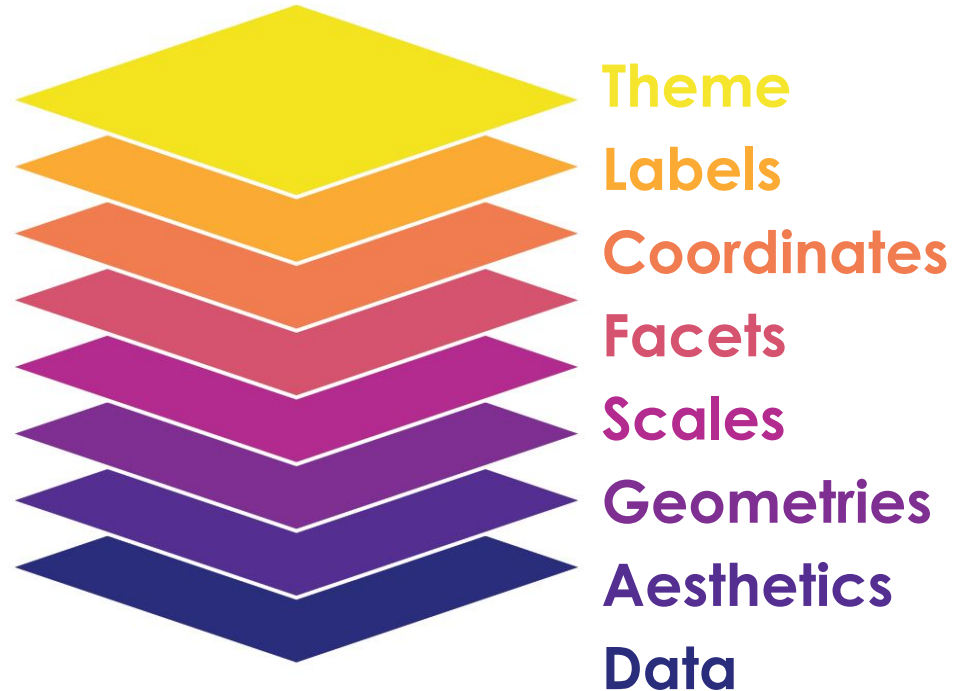


I know nothing about cars

# Juntando tudo...

Uma verdadeira gramática dos gráficos

Com o conceito de *grammar of graphics* do Ggplot2 nós não falamos de um tipo de gráfico em específico, mas sim, sobre elementos específicos de um gráfico.

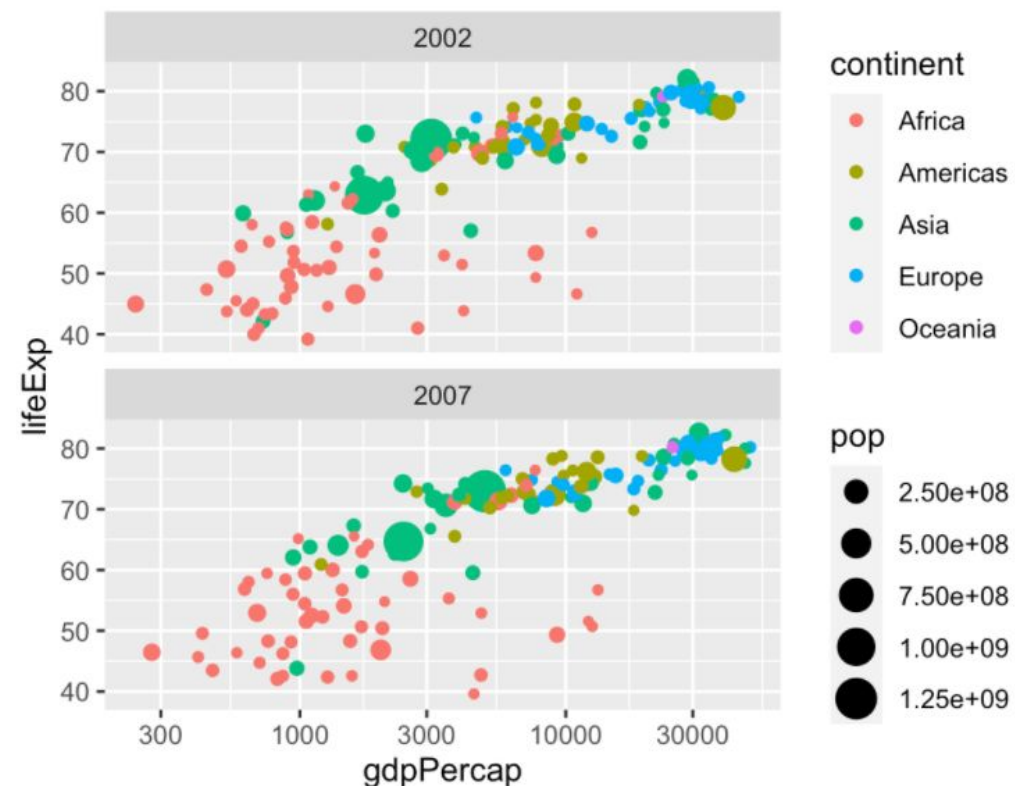




# Descrivendo gráficos com a gramática

Com o conjunto de dados do GapMinder, mapeie a variável riqueza no eixo x, a variável saúde no eixo y e os represente por pontos. Pinte esses pontos de acordo com o continente e dimensione-os de acordo com o tamanho da população. Aplique um ajuste logarítmico na escala do eixo x e faça o gráfico de acordo com a variável ano.

```
ggplot(data = filter(gapminder,  
  year %in% c(2002, 2007)),  
  mapping = aes(x = gdpPercap,  
    y = lifeExp,  
    color = continent,  
    size = pop)) +  
  geom_point() +  
  scale_x_log10() +  
  facet_wrap(vars(year), ncol = 1)
```

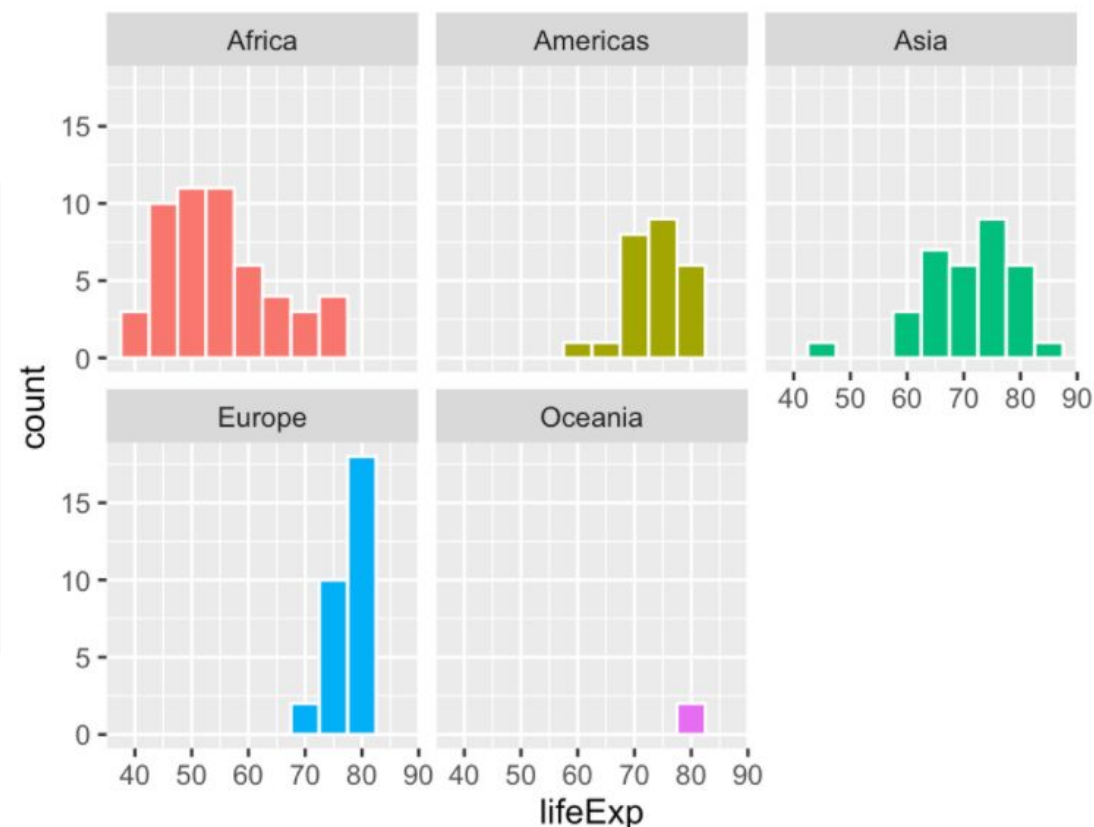




# Descrivendo gráficos com a gramática

Com o conjunto de dados do GapMinder, mapeie a saúde no eixo x e adicione um histograma com barras representando intervalos de 5 anos. Pinte as barras de acordo com o continente e facie o gráfico também de acordo com o continente.

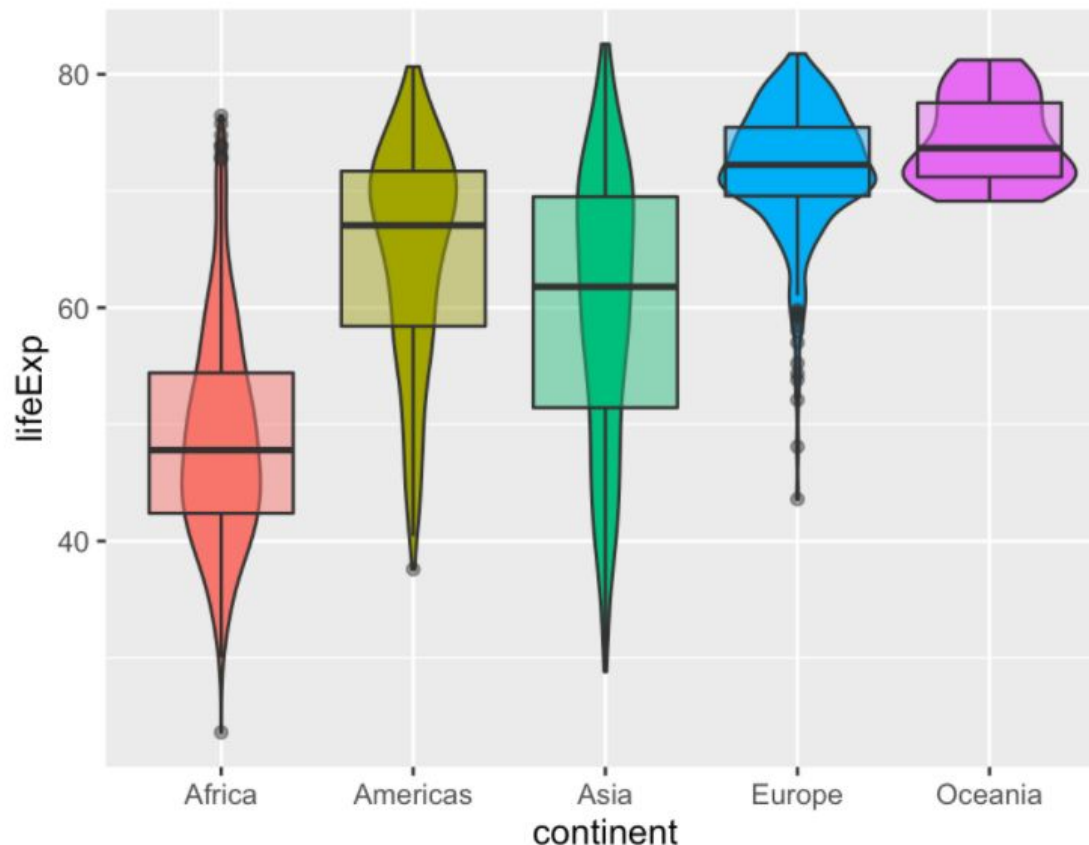
```
ggplot(data = gapminder_2007,  
mapping = aes(x = lifeExp,  
fill = continent)) +  
geom_histogram(binwidth = 5,  
color = "white") +  
guides(fill = FALSE) + # Turn off legend  
facet_wrap(vars(continent))
```



# Descrivendo gráficos com a gramática

Com o conjunto de dados do GapMinder, mapeie o continente no eixo x, a saúde no eixo y, adicione violin plots e box-plots semi-transparentes e os preencha de acordo com o continente.

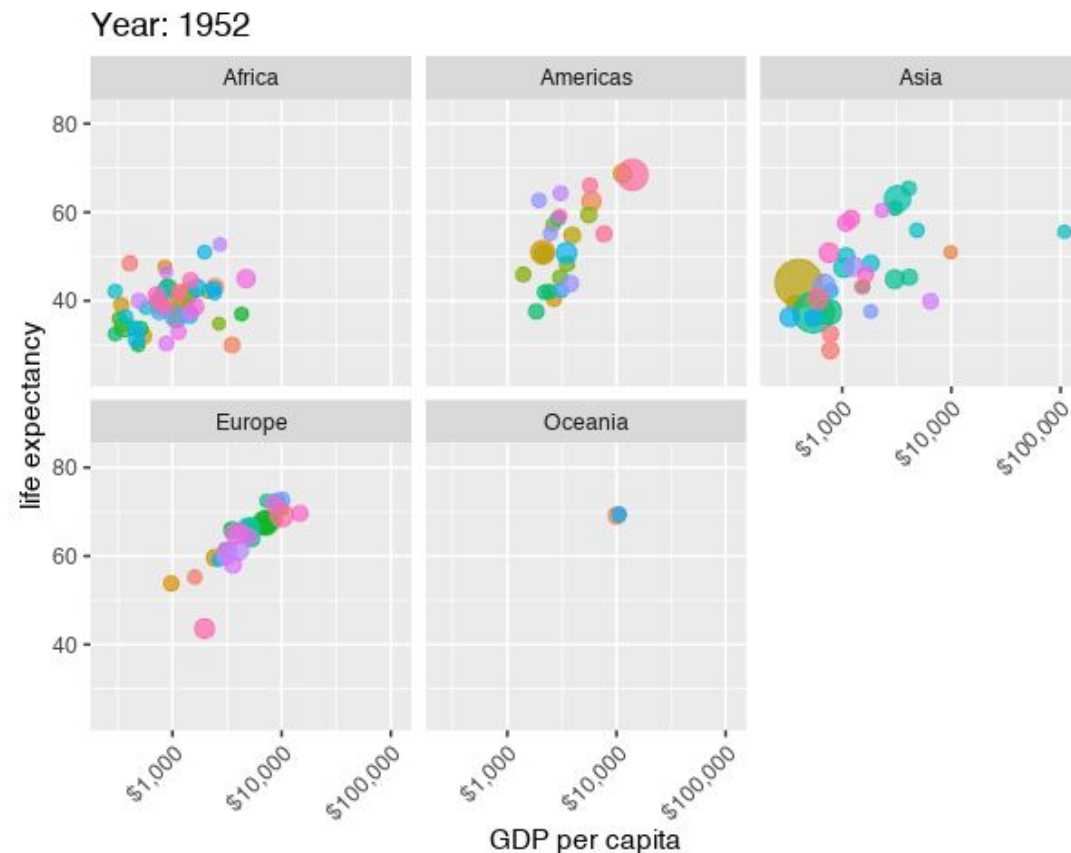
```
ggplot(data = gapminder,  
mapping = aes(x = continent,  
y = lifeExp,  
fill = continent)) +  
geom_violin() +  
geom_boxplot(alpha = 0.5) +  
guides(fill = FALSE) # Turn off legend
```



# Descrivendo gráficos com a gramática

Como veremos nas próximas aulas, podemos variar o aspecto estético de tempo, para isso usamos a extensão **Gganimate**.

```
ggplot(gapminder, aes(x = gdpPercap, y =  
  lifeExp,  
  size = pop, color = country)) +  
  geom_point(alpha = 0.7) +  
  scale_size(range = c(2, 12)) +  
  scale_x_log10(labels = scales::dollar) +  
  guides(size = FALSE, color = FALSE) +  
  facet_wrap(~continent) +  
  # Special gganimate stuff  
  labs(title = 'Year: {frame_time}', x =  
  'GDP per capita', y = 'life expectancy') +  
  transition_time(year) +  
  ease_aes('linear')
```



# Referências

Livros, pacotes, sites, animações, tudo está aqui

- Desenhos da Allison Horst - <https://github.com/allisonhorst>
- Revisão R e Rstudio e base de dados do Titanic - [https://github.com/rladies/meetup-presentations\\_sao-paulo](https://github.com/rladies/meetup-presentations_sao-paulo)
- Conteúdo da disciplina PMAP 8921: Data Visualization ministrado pelo professor Dr. Andrew Heiss na Georgia State University - <https://datavizm20.classes.andrewheiss.com/content/>
- Bíblias deste curso :
- R for Data Science - <https://r4ds.had.co.nz/>
- Ggplot2: elegant graphics for data analysis - <https://ggplot2-book.org/index.html>

# Até a próxima aula!

