Mais observações sobre gráficos com ggplot2

curso-r

Filosofia

O pacote `ggplot2` segue duas filosofias que nos ajudam a entender o processo de construção dos gráficos:

1. Um gráfico estatístico é uma representação visual dos dados por meio de atributos estéticos (posição, cor, forma, tamanho, ...) de formas geométricas (pontos, linhas, barras, ...). The Grammar of Graphics.

2. Um gráfico pode ser construído em camadas (um gráfico é a sobreposição de elementos visuais). A layered grammar of graphics.

Nos exemplos a seguir, vamos utilizar as bases `clima` e `pinguins`, do pacote `dados`. Para a base `clima` vamos usar uma tabela de apoio chamada `temperatura_por_mes`, que vai registrar as medições de temperatura ao longo dos dias.

```
library(tidyverse)
library(dados)
temperatura_por_mes <- clima %>%
  group_by(origem, mes = lubridate::floor_date(data_hora, "month")
  summarise(
    temperatura_media = (mean(temperatura, na.rm = TRUE)-30)/2
  ) %>%
  ungroup()
```

```
## `summarise()` has grouped output by 'origem'. You can override using th
```

Curiosidade: o `gg` em `ggplot` vem de *Grammar of Graphics*.

Canvas, a primeira camada de um gráfico

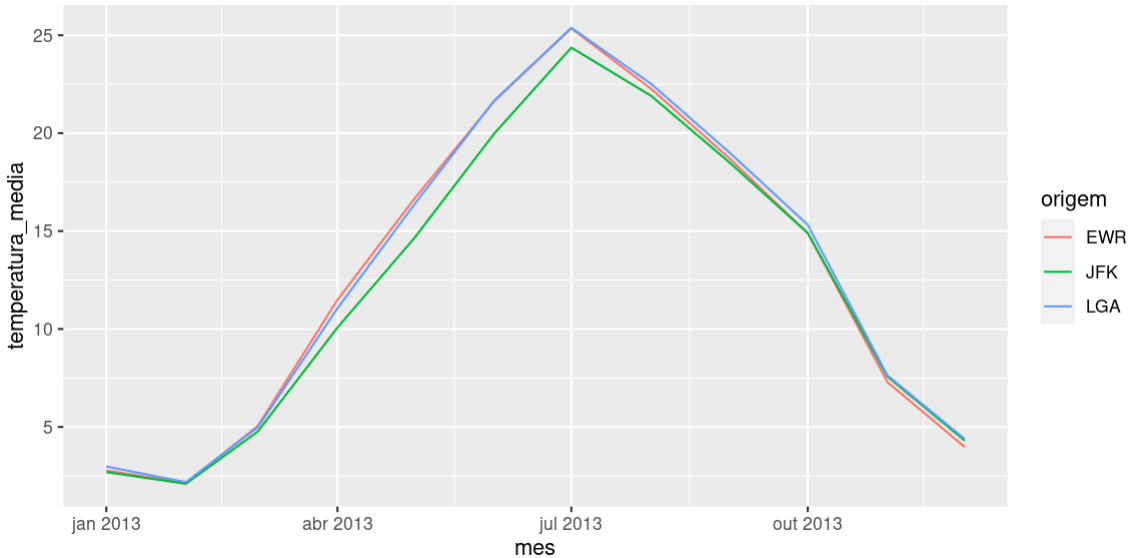
Para construir um gráfico usando o pacote `ggplot2`, começamos sempre com a função `ggplot()` (sim, sem o 2). Essa função recebe como argumento a nossa base de dados. Rodando apenas isso, percebemos que o R cria a primeira camada do nosso gráfico: uma tela em branco (cinza).

```
temperatura_por_mes %>%
  ggplot()
```

Um gráfico de linhas (série de tempo)

Podemos fazer um gráfico de linhas da temperatura média ao longo do tempo acrescentando a função `geom_line()` ao código anterior.

```
temperatura_por_mes %>%
  ggplot() +
  geom_line(aes(x = mes, y = temperatura_media, color = origem))
```



Muitos pontos para discutirmos:

- Esse gráfico tem duas camadas: o canvas, gerado pela função `ggplot()`, e as linhas, geradas pela função `geom_line()`.

- Unimos as camadas de um ggplot usando um `+`. Sim, precisamos controlar a nossa vontade de colocar um `%>%` em vez de `+`, e essa é uma fonte de erro bem comum. O motivo para precisarmos usar `+` em vez do `%>%` é o pacote ggplot ter nascido primeiro que o pipe.

- A função `geom_line()` define que a forma geométrica (daí o prefixo `geom`) utilizada para representar os dados será uma linha. Existe uma família de funções `geom`, sendo que cada uma vai representar uma forma geométrica diferente.

- O primeiro argumento de qualquer função `geom` é o `mapping`. Esse argumento serve para mapear os dados nos atributos estéticos da forma geométrica escolhida. Ele sempre receberá a função `aes()`. No código, nós omitimos o nome do argumento, mas poderíamos ter escrito `geom_line(mapping = aes(x = mes, y = temperatura, color = origem))`.

- A função `aes()` serve para *mapearmos os dados aos elementos estéticos do gráfico*. Os argumentos dela vão sempre depender da forma geométrica que estamos utilizando. No caso de um gráfico de linhas, precisamos definir por quais posições do eixo x e y as linhas vão passar. No exemplo, a posição dos pontos pelos quais as linhas vão passar no eixo y será dada pela coluna `temperatura_media` e a posição dos pontos pelos quais as linhas vão passar no eixo x será dada pela coluna `mes`.

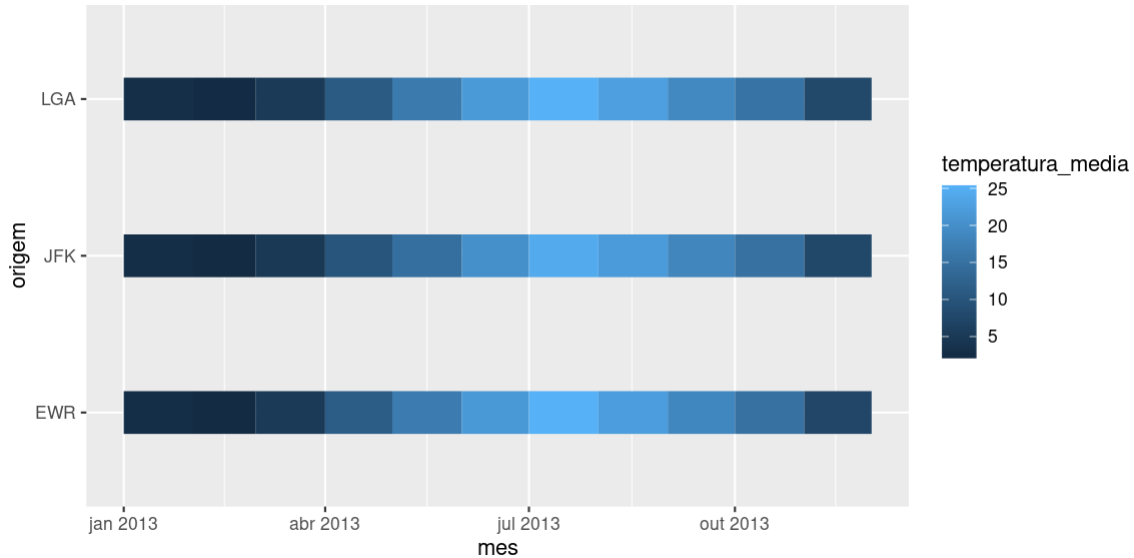
- Tem um terceiro parâmetro estático dentro do `aes()`, que é o `color`. No `geom_line` esse parâmetro é muito especial: além de colorir as linhas de acordo com os valores de uma determinada variável, o parâmetro `color` informa o `ggplot2` que devem ser desenhadas várias linhas, uma para cada valor da variável `color`.

- É muito comum manipularmos a base (aplicarmos diversas funções do `dplyr`, por exemplo) antes de chamarmos a função `ggplot`, mas nesse exemplo nós já fizemos isso antes do gráfico e salvamos o resultado no objeto `temperatura_por_mes`.

O mapeamento das COLUNAS nas FORMAS GEOMÉTRICAS deve ser SEMPRE feito dentro da função `aes()`.

Para ilustar como a gramática dos gráficos é bacana, vamos agora construir um gráfico completamente diferente, mantendo fixo o `geom_line`, mas alterando a atribuição das variáveis dentro do `aes`

```
temperatura_por_mes %>%
  ggplot() +
  geom_line(aes(x = mes, y = origem, color = temperatura_media), size = 10)
```



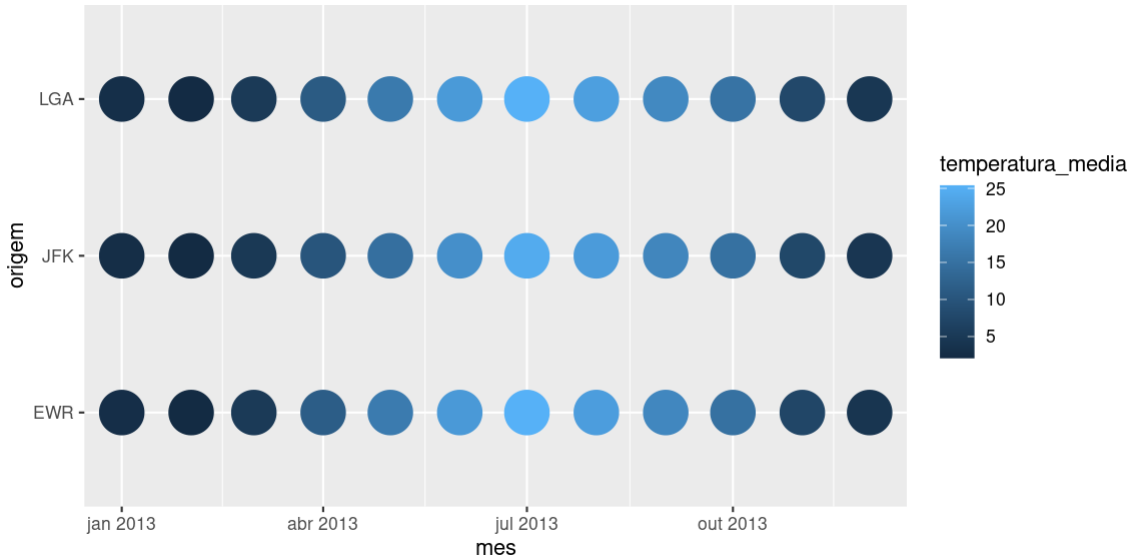
- Aqui nós invertemos o que vai aparecer no eixo `x` e o que vai aparecer no eixo `y`. Agora, no eixo `y`, que não é numérico, vamos colocar os aeroportos aonde as medições de temperatura foram feitas. Como o `y` é uma categoria, o `ggplot2` deixa elas com o mesmo espaçamento.

- `temperatura_media`, que é um valor numérico, agora foi mapeado para as cores, pois utilizamos o parâmetro `color`. Ou seja, os pedaços da reta que correspondem a cada mês vão ser coloridos de acordo com um gradiente de tons de azul que se baseia na variável `temperatura_media`

- A visualização é prejudicada pela finura da linha, mas isso pode ser resolvido usando um outro parâmetro do `geom_line`, que se chama `size`. Colocando `size=10` nós informamos ao `ggplot2` que a linha precisa ser bem grossa.

Claro, nós também poderiamos fazer esse mesmo gráfico com pontinhos ao invés da reta grossa:

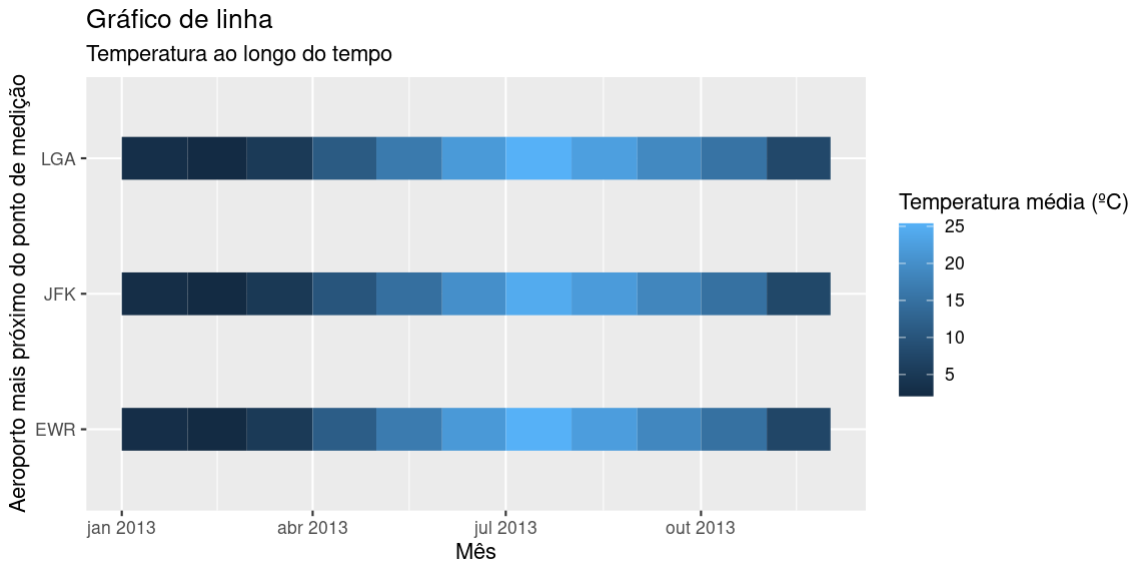
```
temperatura_por_mes %>%
  ggplot() +
  geom_point(aes(x = mes, y = origem, color = temperatura_media), size = 10)
```



Outros elementos estéticos dos gráficos

Além de simplesmente construir o gráfico, nós também podemos modificar outros aspectos da imagem, tais como a cor do fundo do canvas, a cor das bordas do gráfico, os títulos dos eixos etc. Para mexermos nos textos que aparecem ao redor do gráfico (fora do canvas), usamos a função `labs` e seus parâmetros.

```
temperatura_por_mes %>%
  ggplot() +
  geom_line(aes(x = mes, y = origem, color = temperatura_media), size = 10) +
  labs(title = "Gráfico de linha",
       subtitle = "Temperatura ao longo do tempo",
       x = "Mês",
       y = "Aeroporto mais próximo do ponto de medição",
       color = "Temperatura média (°C)" )
```



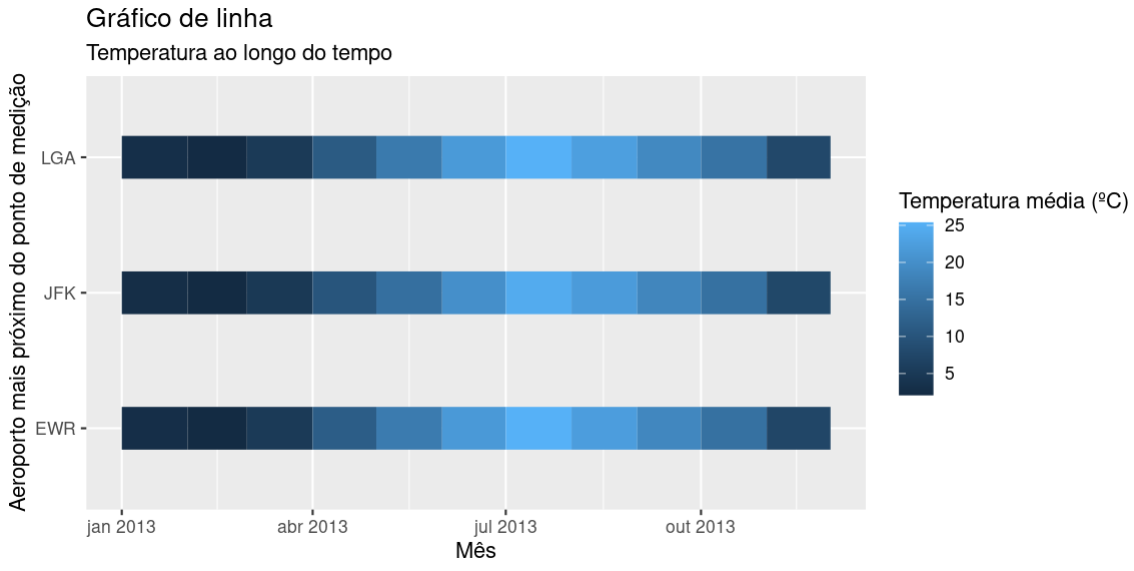


Gráfico de linha
Temperatura ao longo do tempo
Aeroporto mais próximo do ponto de medição
LGA
JFK
EWR
jan 2013
abr 2013
jul 2013
out 2013
Mês
Temperatura média (ºC)
25
20
15
10
5

- O parâmetro `title` da função `labs` dá o título do gráfico, que aparece acima do gráfico.
- O parâmetro `subtitle` dá o subtitulo do gráfico, um texto que aparecerá menor abaixo do gráfico, ideal para explicações mais detalhadas e textos curtos que expliquem o que está sendo mostrado.
- Os parâmetros `x` e `y` representam os textos dos eixos x e y, respectivamente.
- O parâmetro `color` é o texto que aparecerá acima da legenda de cor.

Para mexermos em parâmetros estéticos diferentes além dos textos que aparecem ao redor do gráfico, temos algumas opções. Um jeito muito prático de fazer isso é usando os temas prontos do `ggplot2` ou de outros pacotes auxiliares. Quando somamos uma camada `theme_minimal()`, por exemplo, o nosso gráfico adota um estilo minimalista e *clean*. Esse visual já dá um ar muito mais fresco e objetivo para os nossos gráficos, deixando na tela só o essencial.

```
temperatura_por_mes %>%
  ggplot() +
  geom_line(aes(x = mes, y = origem, color = temperatura_media), size = 10) +
    labs(
    title = "Gráfico de linha",
    subtitle = "Temperatura ao longo do tempo",
    x = "Mês",
    y = "Aeroporto mais próximo do ponto de medição",
    color = "Temperatura média (°C)"
  ) +
  theme_minimal()
```

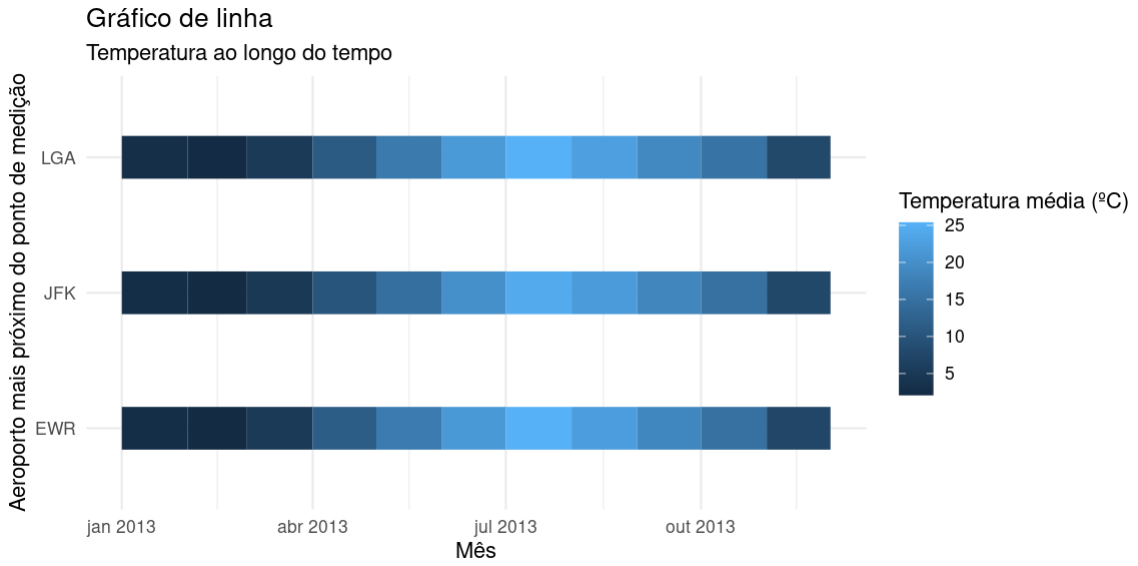


Gráfico de linha
Temperatura ao longo do tempo
Aeroporto mais próximo do ponto de medição
LGA
JFK
EWR
jan 2013
abr 2013
jul 2013
out 2013
Mês
Temperatura média (ºC)
25
20
15
10
5

Um outro jeito de customizar um gráfico é usando a função `theme`. A maior vantagem dessa abordagem é que nós temos controle total sobre tudo que aparece na tela. Como os nossos códigos podem começar a ficar grandes, vamos guardar o nosso `ggplot2` de base dentro de um objeto `grafico_base`

```
grafico_base <- temperatura_por_mes %>%
  ggplot() +
  geom_line(aes(x = mes, y = origem, color = temperatura_media), size = 10) +
  labs(
    title = "Gráfico de linha",
    subtitle = "Temperatura ao longo do tempo",
    x = "Mês",
    y = "Aeroporto mais próximo do ponto de medição",
    color = "Temperatura média (°C)"
  )
```

A função `theme` aceita uma lista enorme de parâmetros estéticos, que se dividem em três categorias:

1. *Textos*, como por exemplo `axis.text.x` e `axis.text.y`, que representam o formato (cor, tamanho, fonte etc) dos textos dos eixos x e y.
2. *Retângulos*, como por exemplo `panel.background`, que representa a primeira camada do nosso gráfico, anterior a todos os `geoms`
3. *Linhas*, como `axis.line.x` e `axis.line.y`, que representam as linhas que marcam os valores dos eixos x e y, respectivamente.

Os parâmetros de cada um desses tipos devem ser definidos seguindo a um padrão específico. Vamos a alguns exemplos

Mudando a cor do eixo dos textos

Quando queremos mudar a cor do textos dos eixos devemos usar a função `theme` com os parâmetros `axis.text.x` e `axis.text.y`.

Agora vem um novo elemento: para definir as características de *textos* usamos a função `element_text`, cujos parâmetros indicam características comuns de textos, tais como cor e tamanho.

```
grafico_base +
  theme(
    axis.text.x = element_text(color = 'red', size = 20),
    axis.text.y = element_text(color = 'red', size = 20)
  )
```

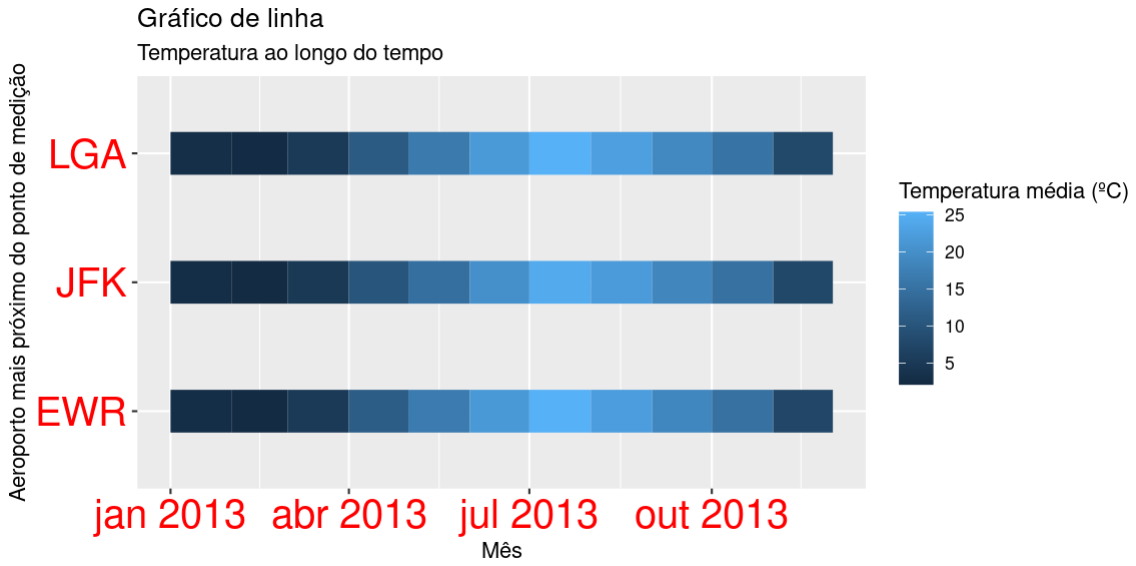
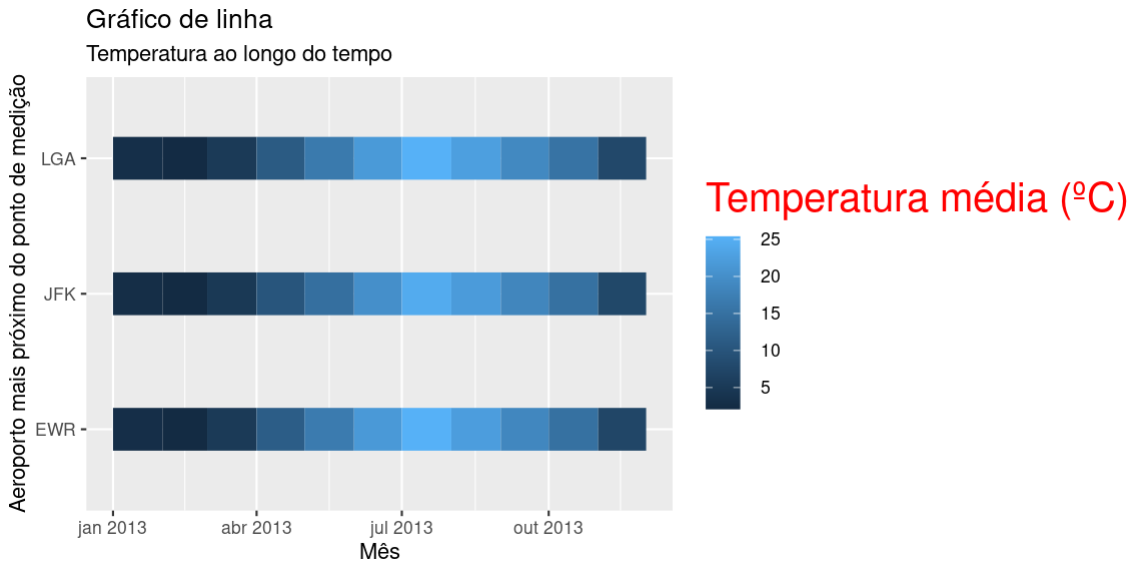


Gráfico de linha
Temperatura ao longo do tempo
Aeroporto mais próximo do ponto de medição
LGA
JFK
EWR
jan 2013
abr 2013
jul 2013
out 2013
Mês
Temperatura média (ºC)
25
20
15
10
5

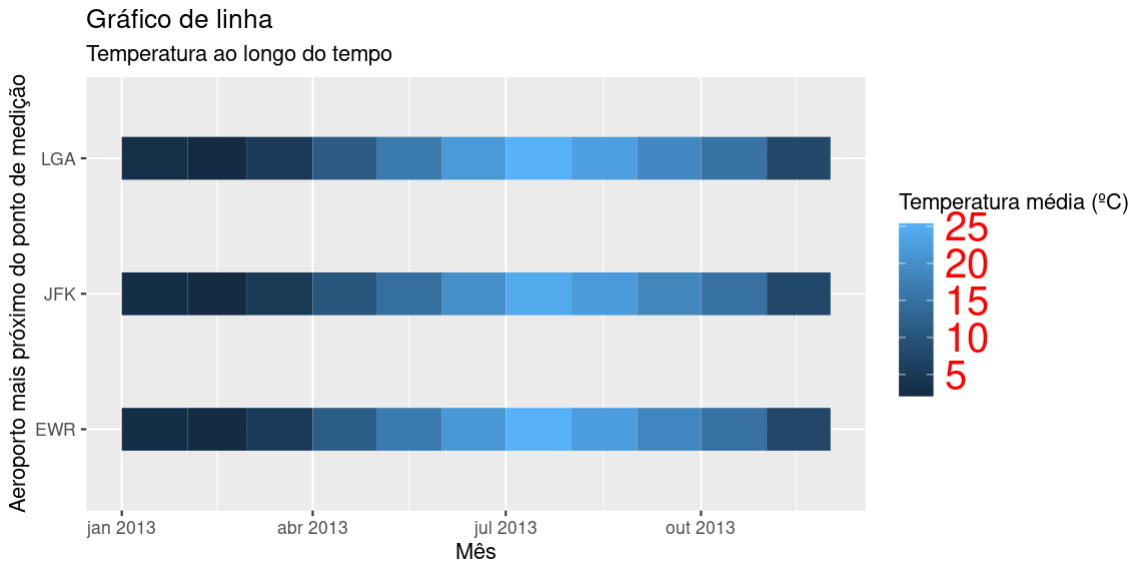
Mudando o tamanho e a cor do texto do título da legenda

```
grafico_base +
  theme(
    legend.title = element_text(color = 'red', size = 20)
  )
```



Mudando o tamanho e a cor do texto da legenda

```
grafico_base +
  theme(
    legend.text = element_text(color = 'red', size = 20)
  )
```



Mudando a cor do fundo do gráfico

Quando queremos mudar a cor do fundo do gráfico devemos usar a função `theme` com os parâmetros `panel.background`.

Agora vem um novo elemento: para definir as características de *retângulos* usamos a função `element_rec`, cujos parâmetros indicam características comuns de retângulos, como cor das bordas `color`, grossura das bordas `size` e a cor do interior do retângulo `fill`.

```
grafico_base +
  theme(
    panel.background = element_rect(fill = 'white')
  )
```

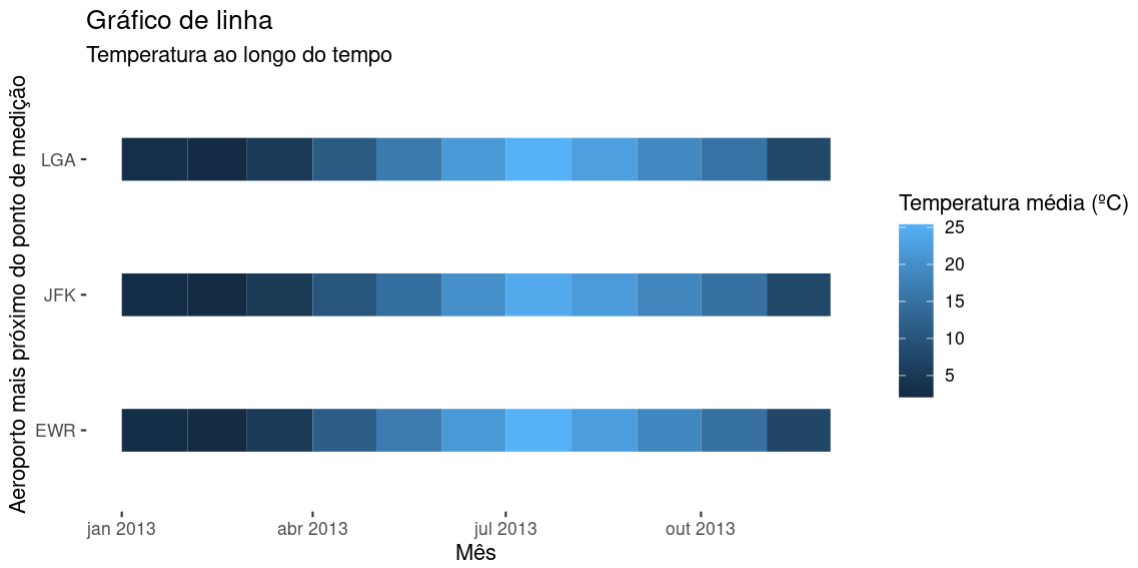
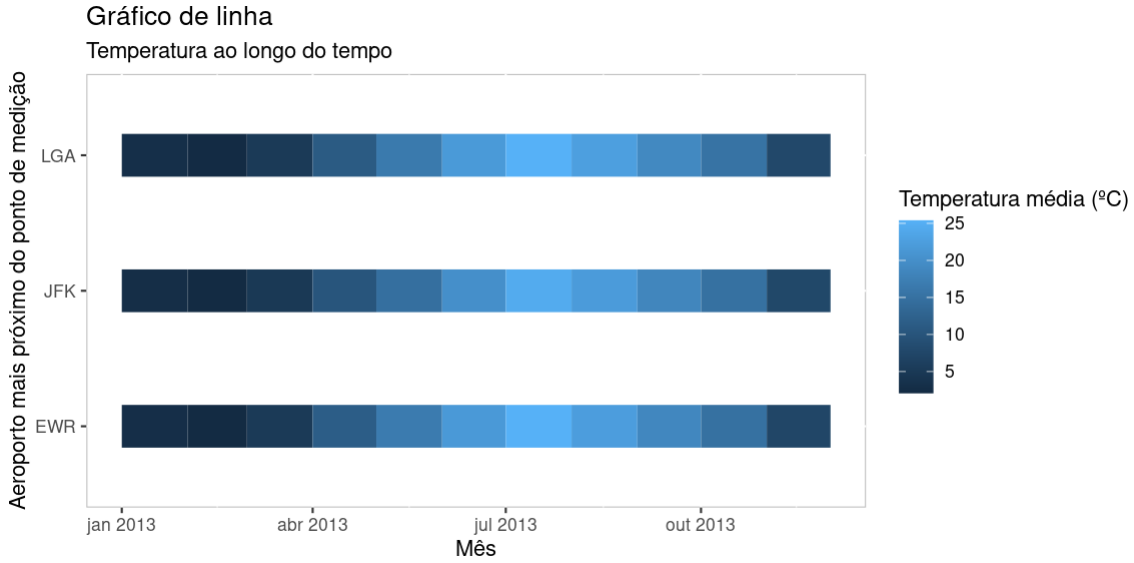


Gráfico de linha
Temperatura ao longo do tempo
Aeroporto mais próximo do ponto de medição
LGA
JFK
EWR
jan 2013
abr 2013
jul 2013
out 2013
Mês
Temperatura média (ºC)
25
20
15
10
5

curso-r

Mudando a cor do fundo e as bordas

```
grafico_base +
  theme(
    panel.background = element_rect(fill = 'white',
                                    color = 'gray')
  )
```



Mudando a cor do fundo, das bordas e aumentando o tamanho das bordas

```
grafico_base +
  theme(panel.background = element_rect(
    fill = 'white',
    color = 'gray',
    size = 2
  ))
```

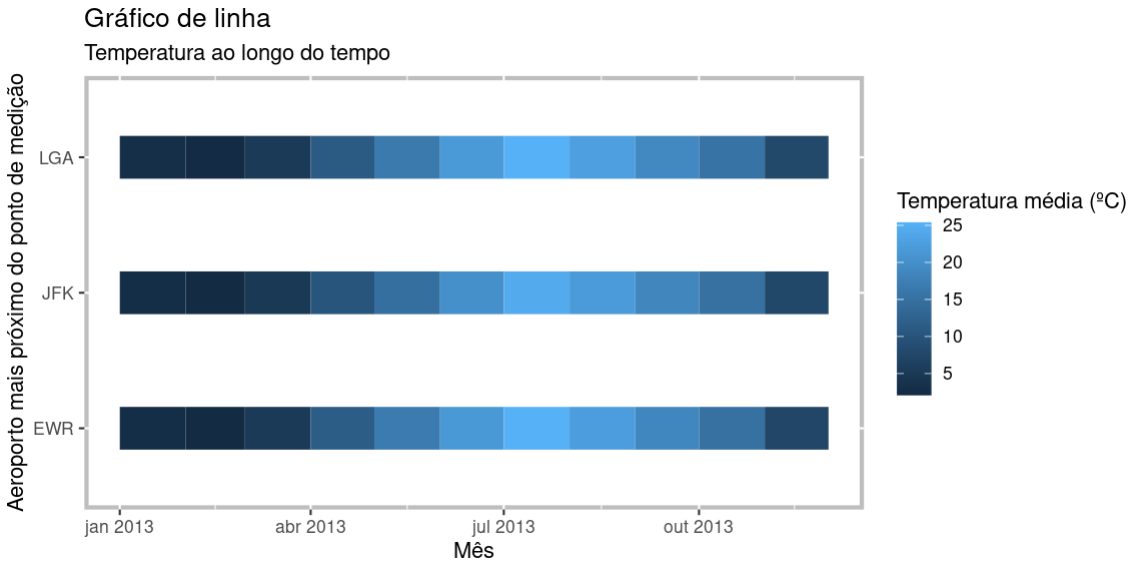


Gráfico de linha
Temperatura ao longo do tempo
Aeroporto mais próximo do ponto de medição
LGA
JFK
EWR
jan 2013
abr 2013
jul 2013
out 2013
Mês
Temperatura média (ºC)
25
20
15
10
5

Mudando a cor das linhas dos eixos

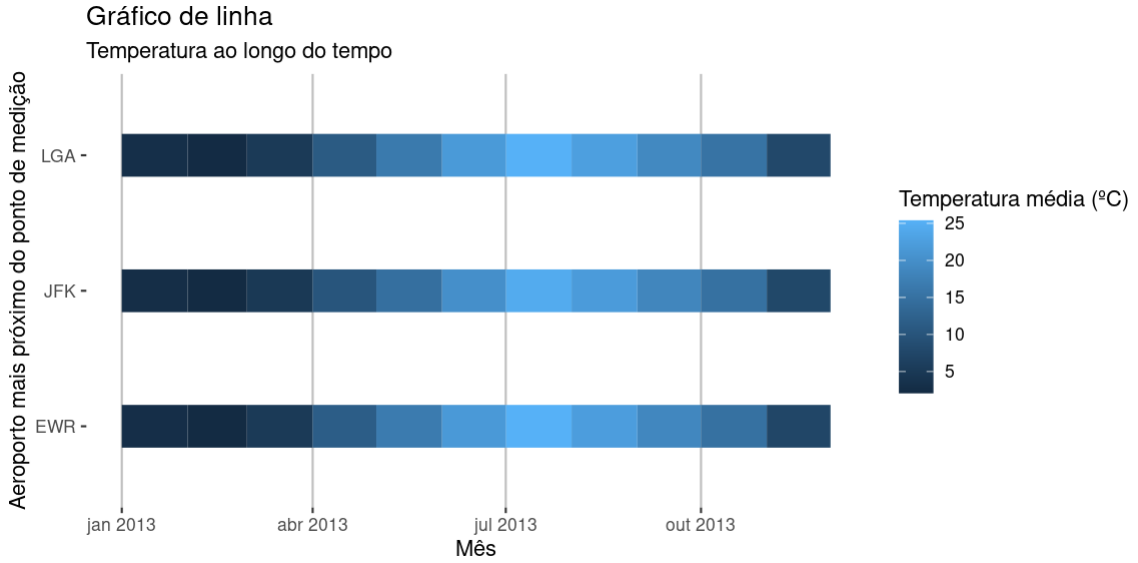
Você notou que no último gráfico as linhas dos eixos sumiram? Isso aconteceu porque a cor padrão das linhas dos eixos é branco. Quando alteramos a cor do fundo para branco, a cor do fundo e das linhas ficou igual e por isso não foi mais possível discernir uma da outra.

Os parâmetros dentro do theme que representam as linhas dos eixos são `panel.grid.x` e `panel.grid.y`. Como esses elementos do gráfico são **linhas**, nós usaremos a função `element_line`, cujos parâmetros são características comuns de linhas como cor `color`, tamanho `size` e tipo de linha (pontilhado, tracejado, contínuo etc) `linetype`.

```
grafico_base +
  theme(
    panel.background = element_rect(fill = 'white'),
    panel.grid  = element_line(color = 'gray')
  )
```

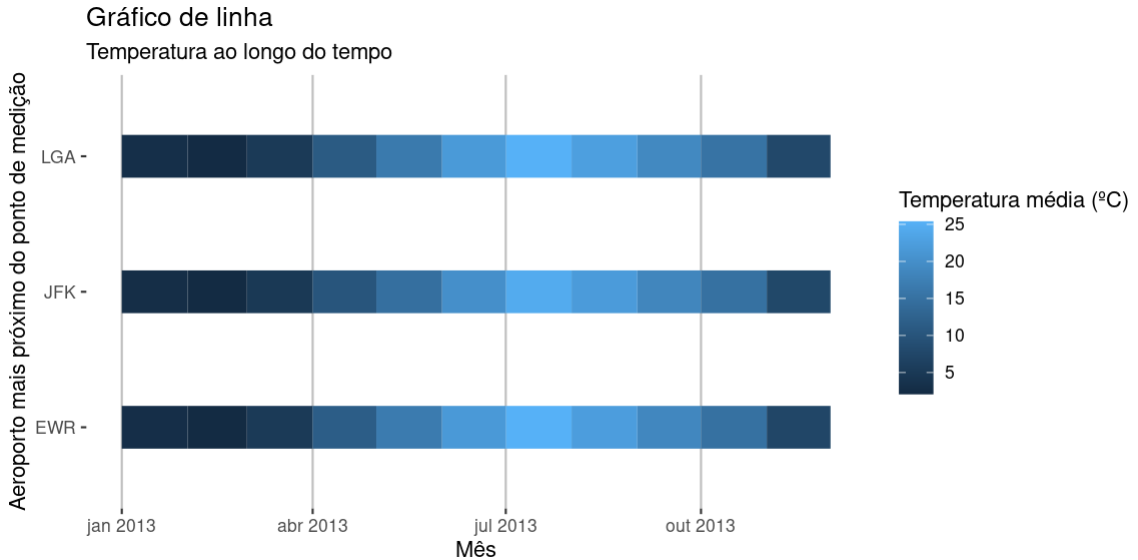
Mudando a cor da linhas principais do eixo x

```
grafico_base +
  theme(
    panel.background = element_rect(fill = 'white'),
    panel.grid.major.x  = element_line(color = 'gray')
  )
```



Mudando a cor das linhas secundárias do eixo x

```
grafico_base +
  theme(
    panel.background = element_rect(fill = 'white'),
    panel.grid.major.x  = element_line(color = 'gray')
  )
```



Acabando com a diferença entre majors e minors

```
grafico_base +
  theme(
    panel.background = element_rect(
      fill = 'white', color = 'gray', size = 2
    ),
    panel.grid.major.x  = element_line(color = 'lightgray'),
    panel.grid.minor.x  = element_line(color = 'lightgray')
  )
```

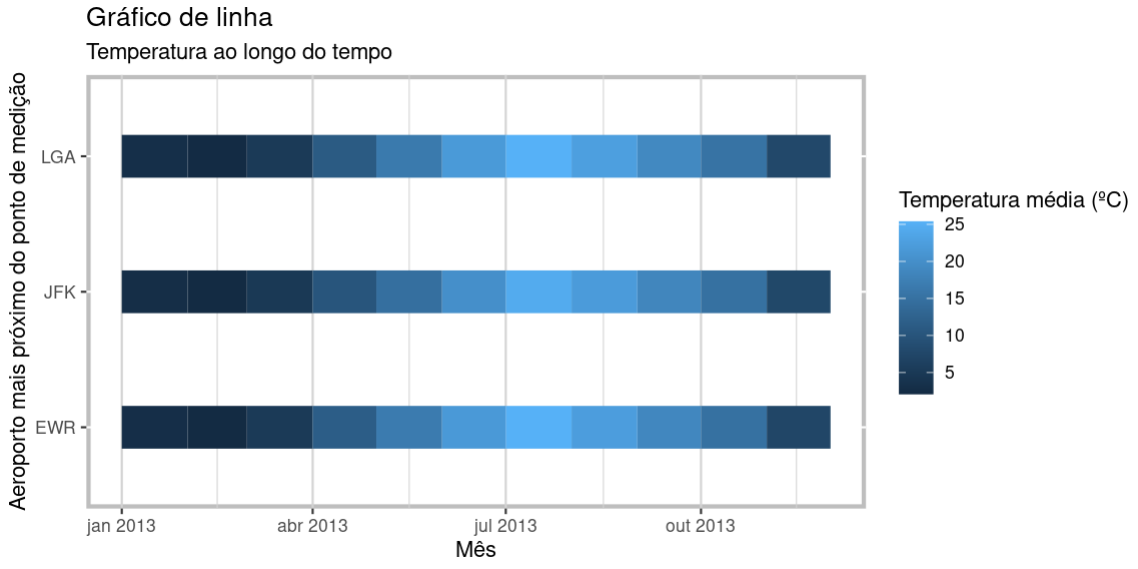


Gráfico de linha
Temperatura ao longo do tempo
Aeroporto mais próximo do ponto de medição
LGA
JFK
EWR
jan 2013
abr 2013
jul 2013
out 2013
Mês
Temperatura média (ºC)
25
20
15
10
5

Mais temas

Mexendo na função `themes` nós podemos criar o theme que nós quisermos, mas você não necessariamente precisa partir do 0 nos seus gráficos. O pacote `ggthemes` tem vários temas prontos que basta você somar uma camada de temas, similar ao que já estávamos fazendo, e seu gráfico se transforma completamente.

A função `ggthemes::theme_fivethirtyeight()`, por exemplo, aplica o tema do famoso site (FiveThirtyEight)[https://fivethirtyeight.com/].

O pacote `ggthemes` tem uma série de temas interessantes que você pode acessar (aqui) [https://yutannihilation.github.io/allYourFigureAreBelongToUs/ggthemes/]

Aplicando o tema do FiveThirtyEight

```
temperatura_por_mes %>%
  ggplot() +
  geom_line(aes(x = mes, y = origem, color = temperatura_media), s
  labs(
    title = "Gráfico de dispersão",
    subtitle = "Receita vs Orçamento",
    x = "Mês",
    y = "Aeroporto mais próximo do ponto de medição",
    color = "Temperatura média (°C)"
  ) +
  ggthemes::theme_fivethirtyeight()
```

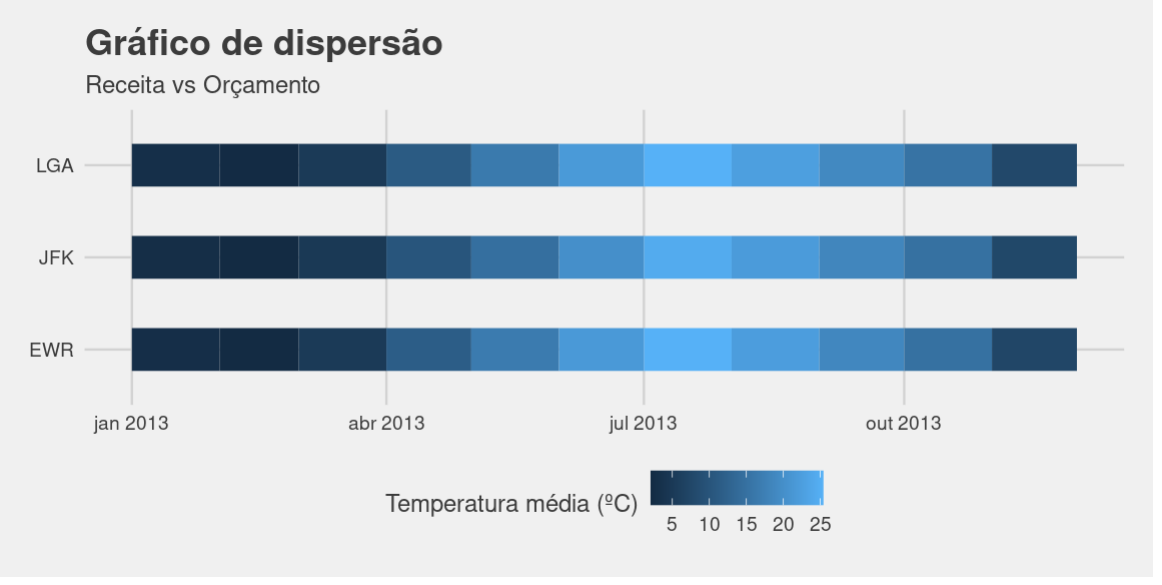


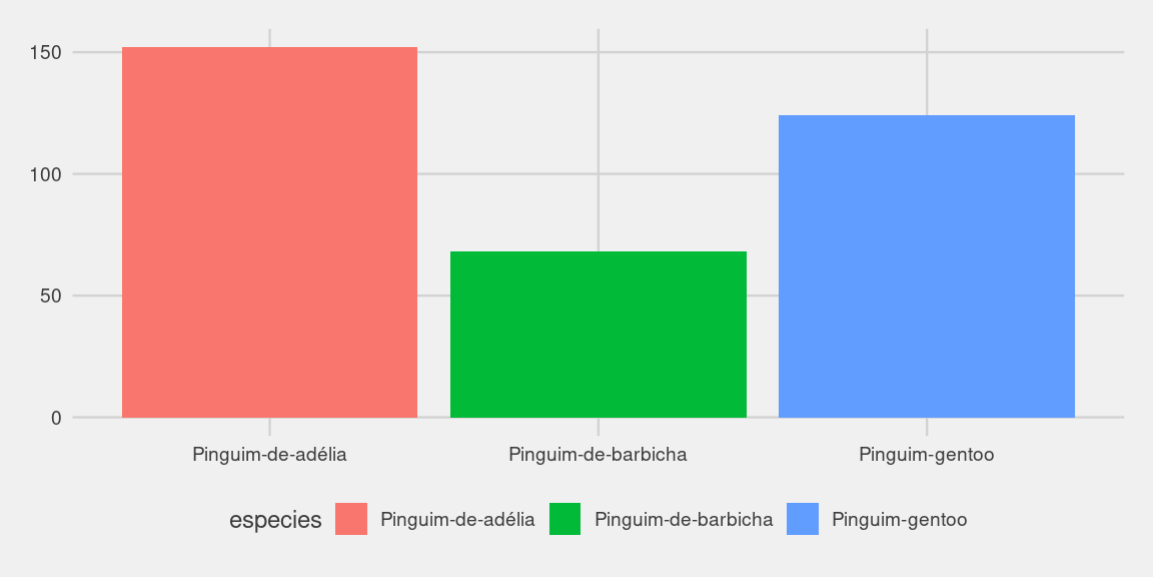
Gráfico de dispersão
Receita vs Orçamento
LGA
JFK
EWR
jan 2013
abr 2013
jul 2013
out 2013
Temperatura média (ºC)
5
10
15
20
25

Gráfico de barras

Até o momento exploramos bastante o gráfico de linhas, mas um tipo de gráfico muito usado para destacar a comparação entre valores para diferentes categorias, por exemplo, é o gráfico de barras:

```
pinguins %>%
  count(especies) %>%
  ggplot(aes(x = especies, y = n,fill = especies)) +
  geom_col() +
  ggthemes::theme_fivethirtyeight() +
  labs(x = "Espécie do pinguin", y = "Número de pinguins estudados
```

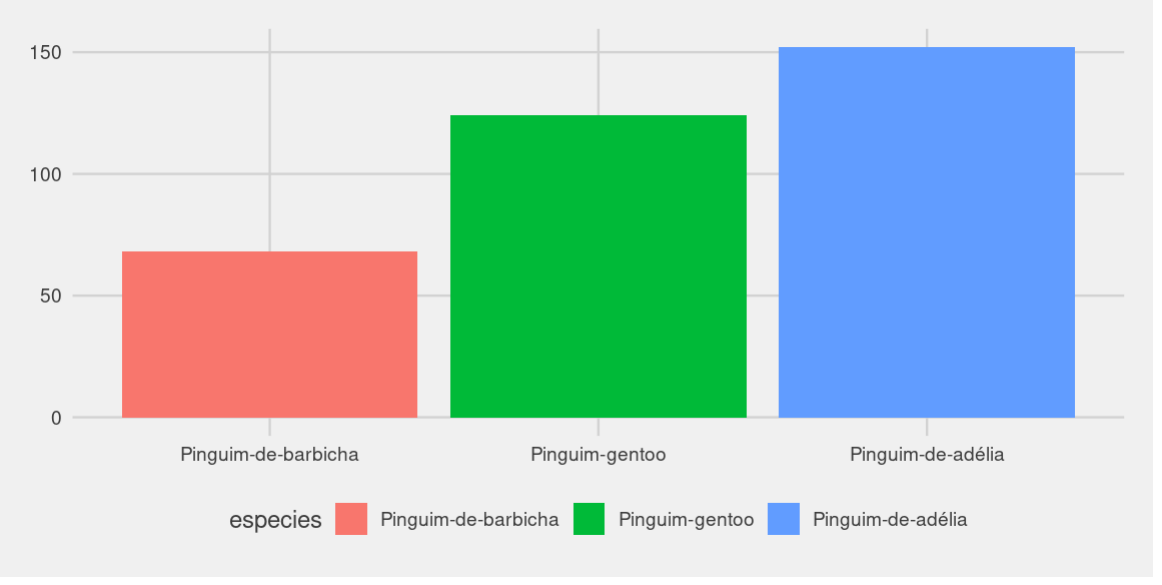
Este gráfico destaca o fato dos Pinguins-de-barbicha serem muito menos frequentes do qeu os demais.



Ordenando as barras

É muito comum que a gente ordene as barras da menor pra maior ou da menor pra maior, justamente para destacar ainda mais as discrepâncias nos dados. Para isso, usamos a função `fct_reorder` do pacote `forcats`, que nos ajuda a mexer com categorias.

```
pinguins %>%
  count(especies) %>%
  mutate(especies = fct_reorder(especies, n)) %>%
  ggplot(aes(x = especies, y = n,fill = especies)) +
  geom_col() +
  ggthemes::theme_fivethirtyeight()
```


150
100
50
0
Pinguim-de-barbicha
Pinguim-gentoo
Pinguim-de-adélia
especies
Pinguim-de-barbicha
Pinguim-gentoo
Pinguim-de-adélia

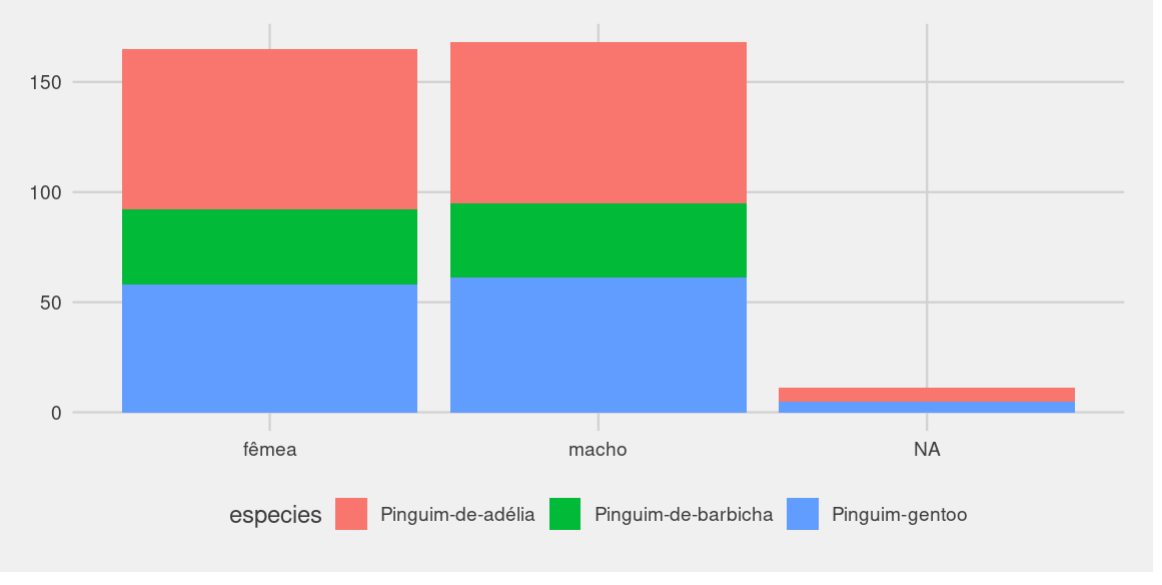
Comparando grupos e subgrupos

Às vezes queremos comparar, ao mesmo tempo, grupos e subgrupos. Ou seja, no mesmo "x" vamos plotar várias barras. Aqui teremos três opções, que serão passadas para o parâmetro `position` do `geom_col`:

- empilhar, onde os as barras são plotadas uma sobre a outra. Essa opção é representada pelo parâmetro `position = "stack"` (padrão);
- colocar lado a lado, onde as parras são plotadas uma ao lado da outra, com um pequeno espaço entre elas. Essa opção é representada pelo parâmetro `position = "dodge"`.
- empilhar, mas normalizar os dados antes de fazer isso, de tal forma que todas as barras vão ter o mesmo tamanho. Essa opção é representada pelo parâmetro `position = "fill"`.

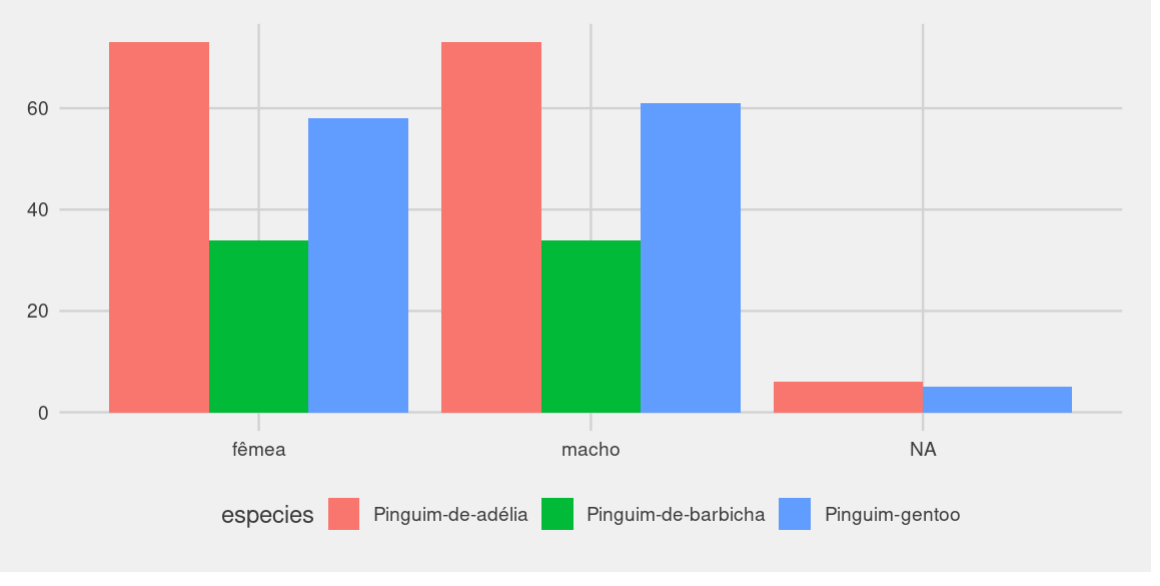
Barras empilhadas

```
pinguins %>%
  count(especies, sexo) %>%
  ggplot(aes(x = sexo, y = n,fill = especies)) +
  geom_col() +
  ggthemes::theme_fivethirtyeight()
```



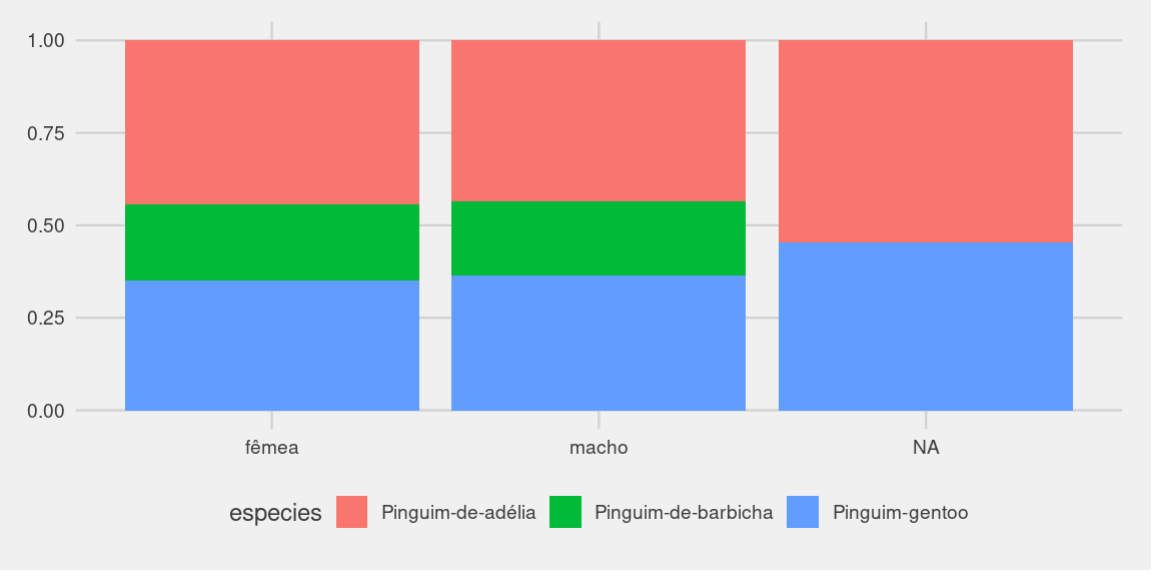
Barras lado-a-lado

```
pinguins %>%
  count(especies, sexo) %>%
  ggplot(aes(x = sexo, y = n,fill = especies)) +
  geom_col(position = 'dodge') +
  # "dodge" significa "desviar"
  ggthemes::theme_fivethirtyeight()
```



Barras empilhadas e normalizadas

```
pinguins %>%
  count(especies, sexo) %>%
  ggplot(aes(x = sexo, y = n,fill = especies)) +
  geom_col(position = 'fill') +
  # "fill" significa "preencher"
  ggthemes::theme_fivethirtyeight()
```



Visualizando distribuições

Ás vezes queremos visualizar distribuições, no sentido estatístico mesmo

Links úteis

- Extensões do ggplot2: https://exts.ggplot2.tidyverse.org/
- Seção de gráficos do R cookbook (ótima folha de cola): http://www.cookbook-r.com/Graphs/
- Temas do `ggthemes`: https://yutannihilation.github.io/allYourFigureAreBelongToUs/ggthemes/