Revisão sobre gráficos com ggplot2

curso-r

Gráficos com ggplot2

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, continuaremos a utilizar a base do IMDB. Criamos de antemão a coluna `lucro` pois a utilizaremos bastante.

```
library(tidyverse)
imdb <- read_rds("dados/imdb.rds")
imdb <- imdb %>% mutate(lucro = receita - orcamento)
```

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).

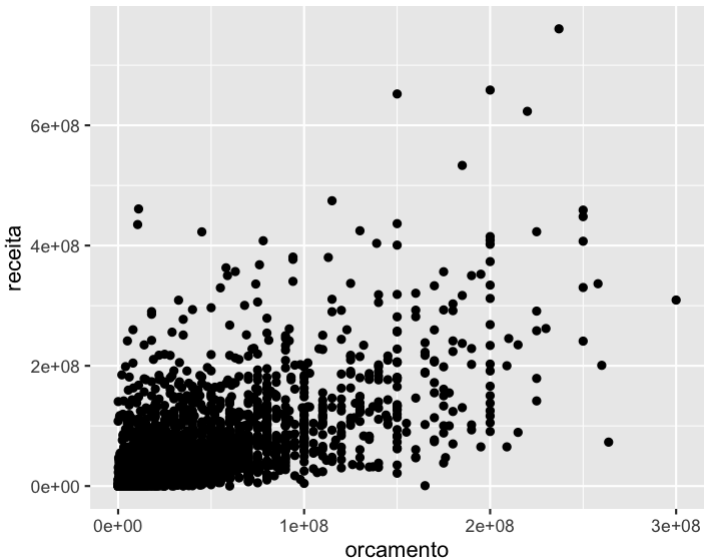
```
imdb %>% ggplot()
```

Um gráfico de pontos (dispersão)

Podemos fazer um gráfico de dispersão da receita contra o orçamento dos filmes acrescentando a função `geom_point()` ao código anterior.

```
imdb %>%
  ggplot() +
  geom_point(aes(x = orcamento, y = receita))
```

```
## Warning: Removed 720 rows containing missing values (geom_point).
```



Muitos pontos para discutirmos:

- Esse gráfico tem duas camadas: o canvas, gerado pela função `ggplot()`, e os pontos, gerado pela função `geom_point()`.
- 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_point()` define que a forma geométrica (daí o prefixo `geom`) utilizada para representar os dados será pontos. 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_point(mapping = aes(x = orcamento, y = receita))`.

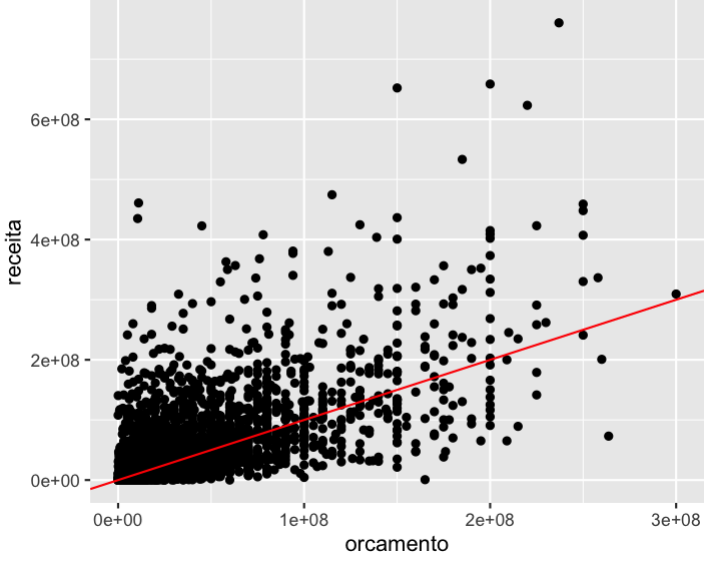
- A função `aes()` serve para *mapearmos os dados aos elementos estéticos do gráfico*. Os argumentos dela vão sempre depender da forma geométrica que estamos utilizando. No caso de um gráfico de pontos, precisamos definir como as posições do eixo x e y serão construídas. No exemplo, a posição do ponto no eixo x será dada pela coluna `orcamento` e a posição do ponto no eixo y será dada pela coluna `receita`.
- O *warning* indica quantas observações (linhas) precisaram ser removidas, por não possuir informação de orçamento ou receita.
- Veremos nos próximos exemplos que será muito comum manipularmos a base (aplicarmos diversas funções do `dplyr`, por exemplo) antes de chamarmos a função `ggplot`.

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

Vamos agora inserir um novo elemento visual ao gráfico: a reta x = y.

```
imdb %>%
  ggplot() +
  geom_point(aes(x = orcamento, y = receita)) +
  geom_abline(intercept = 0, slope = 1, color = "red")
```

```
## Warning: Removed 720 rows containing missing values (geom_point).
```

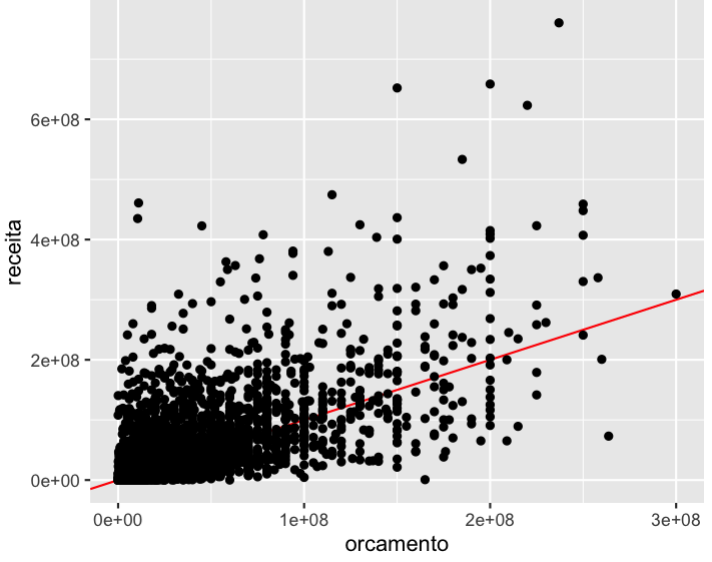


- A reta x = y é acrescentada ao gráfico pela função `geom_abline()`. Esse `geom` pode ser utilizado para desenhar qualquer reta do tipo `y = a + b * x`, sendo `a` o intercepto (*intercept*) da reta e `b` o seu coeficiente angular (*slope*).
- Essa reta nos permite observar o número de filmes que obtiveram lucro (pontos acima da reta) e aqueles que obtiveram prejuízo (pontos abaixo da reta).
- Como não estamos mapeando colunas a essa reta, não precisamos colocar os argumentos da função `geom_abline()` do `aes()`.

Veja como o ggplot realmente é construído por camadas. Agora, colocamos a camada da reta antes da camada dos pontos. Os pontos ficam em cima da reta.

```
imdb %>%
  ggplot() +
  geom_abline(intercept = 0, slope = 1, color = "red") +
  geom_point(aes(x = orcamento, y = receita))
```

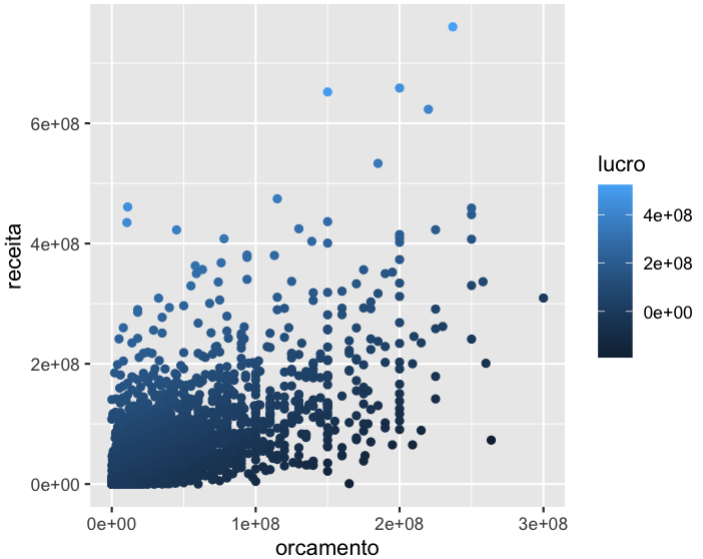
```
## Warning: Removed 720 rows containing missing values (geom_point).
```



Os atributos `x` e `y` são necessários para construirmos um gráfico de pontos. Outros atributos também podem ser mapeados em pontos, como a cor. Como a coluna `lucro` é numérica, um degradê de cores é criado para os pontos, a depender do lucro.

```
imdb %>%
  ggplot() +
  geom_point(aes(x = orcamento, y = receita, color = lucro))
```

```
## Warning: Removed 720 rows containing missing values (geom_point).
```



Poderíamos também classificar os filmes entre aqueles que lucraram ou não. Uma cor é atribuída a cada categoria.

```
imdb %>%
  mutate(
    lucrou = ifelse(lucro <= 0, "Não", "Sim")
  ) %>%
  ggplot() +
  geom_point(aes(x = orcamento, y = receita, color = lucrou))
```

```
## Warning: Removed 720 rows containing missing values (geom_point).
```

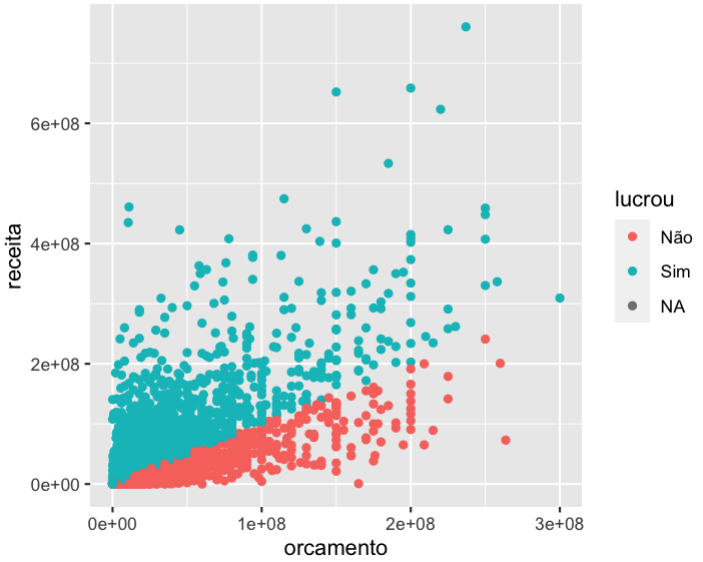


Gráfico de linhas

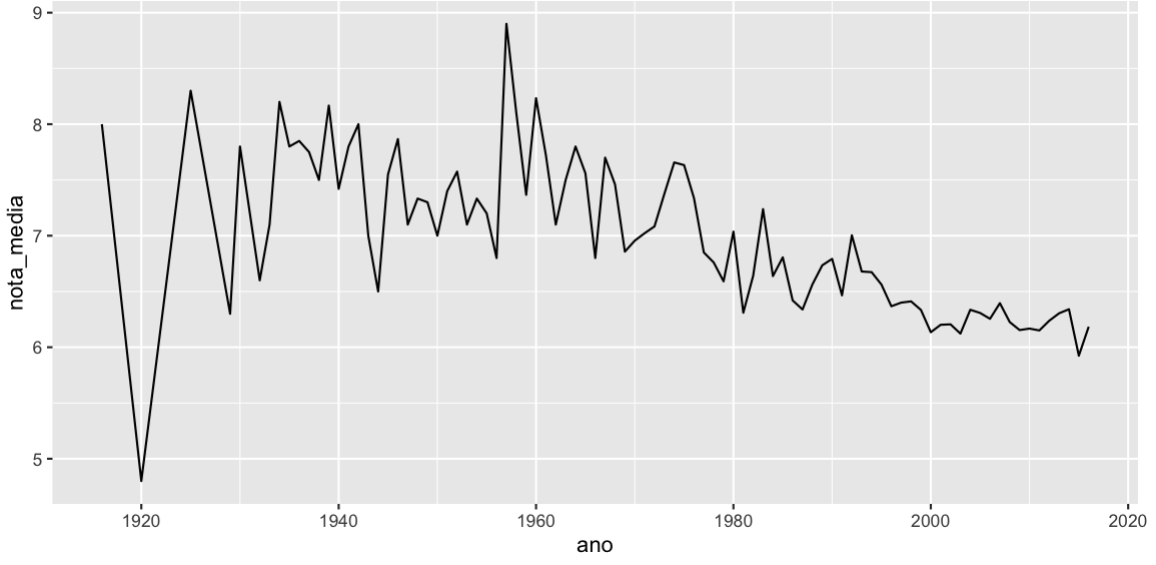
Utilizamos o `geom_line` para fazer gráficos de linhas. Gráficos de linha são muito utilizados para representar *séries temporais*, isto é, observações medidas repetidamente em intervalos (em geral) equidistantes de tempo.

Assim como nos gráficos de pontos, precisamos definir as posições `x` e `y` para construirmos gráficos de linhas.

A seguir, construímos o gráfico das notas médias dos filmes produzidos em cada ano,

```
imdb %>%
  group_by(ano) %>%
  summarise(nota_media = mean(nota_imdb, na.rm = TRUE)) %>%
  ggplot() +
  geom_line(aes(x = ano, y = nota_media))
```

```
## Warning: Removed 1 row(s) containing missing values (geom_path).
```

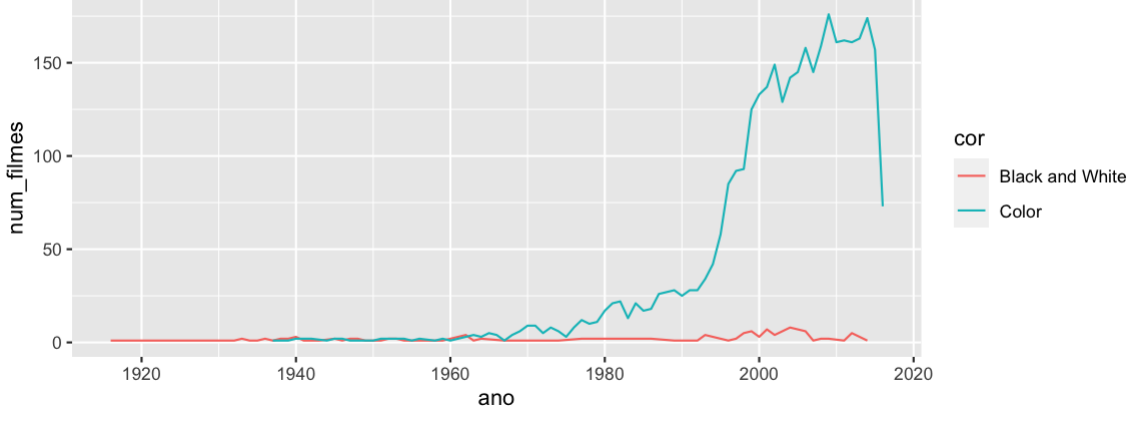


Se mapearmos uma coluna à cor das retas, serão construídas uma reta correspondente a cada categoria distinta dessa coluna.

```
imdb %>%
  filter(!is.na(cor)) %>%
  group_by(ano, cor) %>%
  summarise(num_filmes = n()) %>%
  ggplot() +
  geom_line(aes(x = ano, y = num_filmes, color = cor))
```

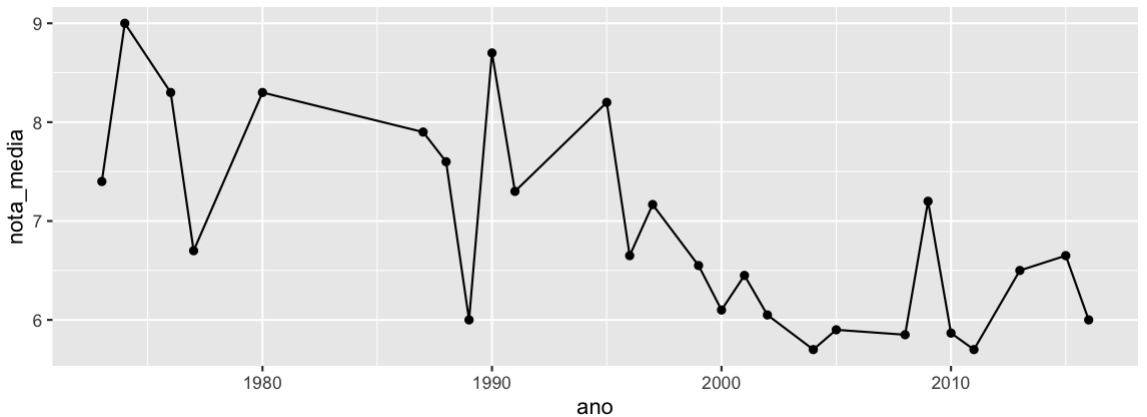
```
## `summarise()` has grouped output by 'ano'. You can override using the `.groups` argument.
```

```
## Warning: Removed 2 row(s) containing missing values (geom_path).
```



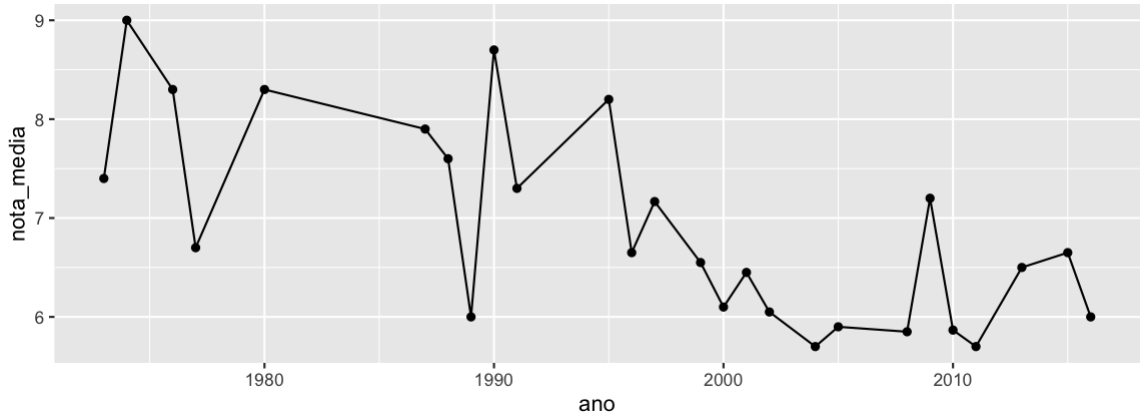
Podemos colocar pontos e retas no mesmo gráfico. Basta acrescentar os dois `geoms`. O gráfico abaixo mostra nota média anual dos filmes do Robert De Niro.

```
imdb %>%
  filter(ator_1 == "Robert De Niro") %>%
  group_by(ano) %>%
  summarise(nota_media = mean(nota_imdb, na.rm = TRUE)) %>%
  ggplot() +
  geom_line(aes(x = ano, y = nota_media)) +
  geom_point(aes(x = ano, y = nota_media))
```



Quando precisamos usar o mesmo `aes()` em vários `geoms`, podemos defini-lo dentro da função `ggplot()`. Esse `aes()` será então distribuído para todo `geom` do gráfico. O código anterior pode ser reescrito da seguinte forma.

```
imdb %>%
  filter(ator_1 == "Robert De Niro") %>%
  group_by(ano) %>%
  summarise(nota_media = mean(nota_imdb, na.rm = TRUE)) %>%
  ggplot(aes(x = ano, y = nota_media)) +
  geom_line() +
  geom_point()
```



Se algum `geom` necessitar de um atributo que os outros não precisam, esse atributo pode ser especificado normalmente dentro dele. Abaixo, utilizamos o `geom_label` para colocar as notas médias no gráfico. Além do `x` e `y`, o `geom_label` também precisa do texto que será escrito no gráfico.

```
imdb %>%
  filter(ator_1 == "Robert De Niro") %>%
  group_by(ano) %>%
  summarise(nota_media = mean(nota_imdb, na.rm = TRUE)) %>%
  mutate(nota_media = round(nota_media, 1)) %>%
  ggplot(aes(x = ano, y = nota_media)) +
  geom_line() +
  geom_label(aes(label = nota_media))
```

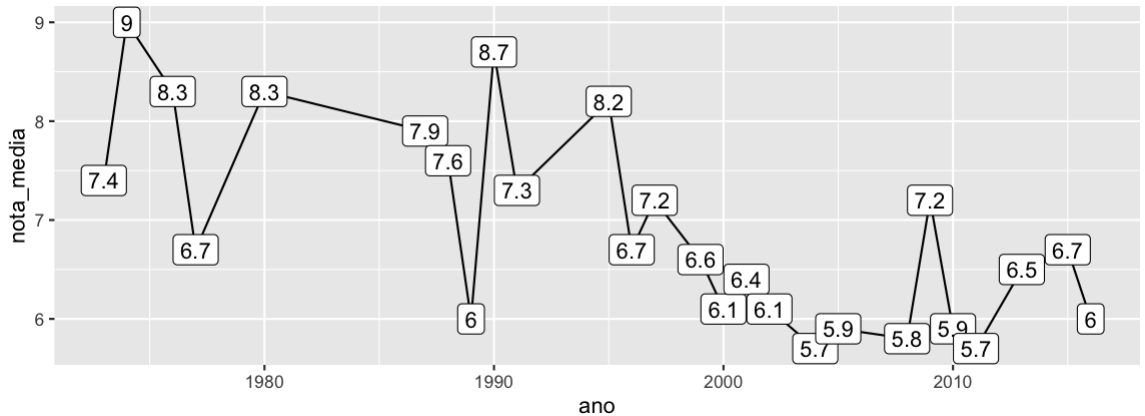
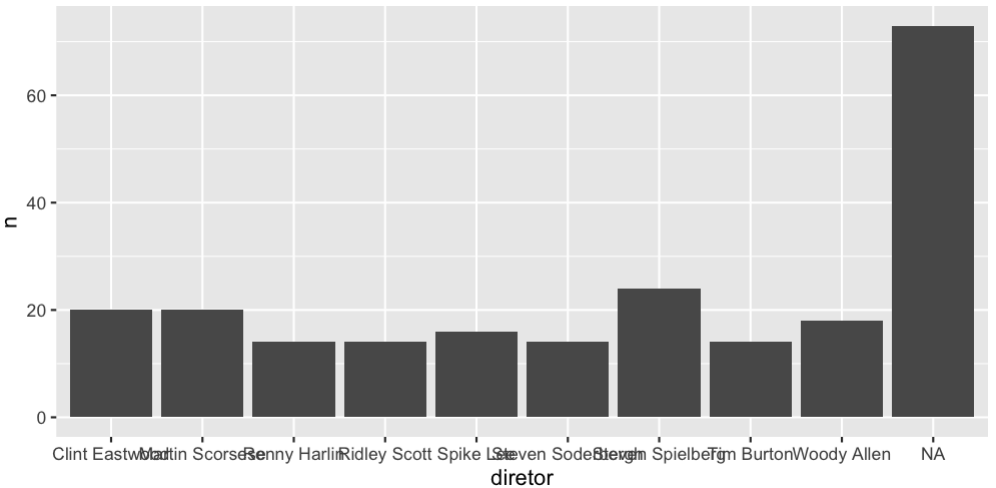


Gráfico de barras

Para construir gráficos de barras, utilizamos o `geom_col`. A seguir, construímos um gráfico de barras do número de filmes dos 10 diretores que mais aparecem na nossa base do IMDB.

```
imdb %>%
  count(diretor) %>%
  top_n(10, n) %>%
  ggplot() +
  geom_col(aes(x = diretor, y = n))
```

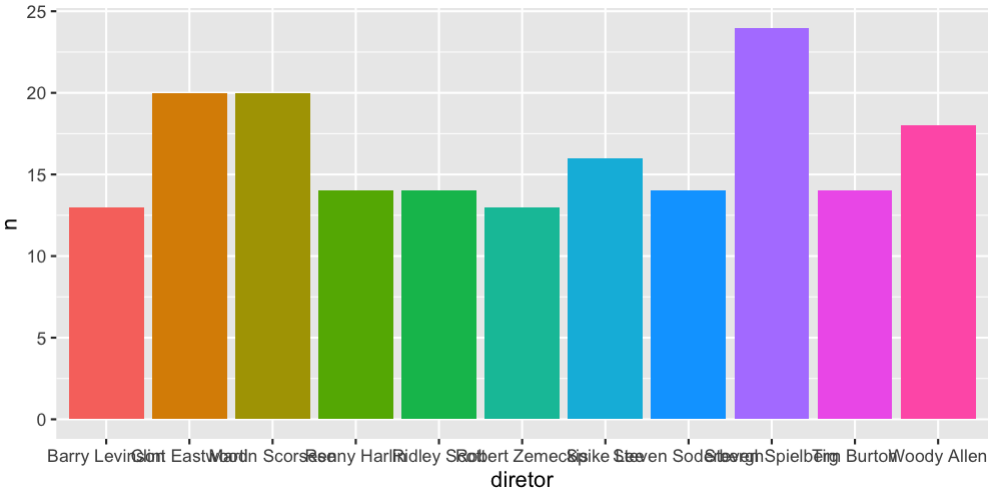


Gráficos de barras também precisam dos atributos `x` e `y`, sendo que o atributo `y` representará a altura de cada barra.

No gráfico anterior, vemos que o `NA` é considerado uma "opção" de diretor e entra no gráfico. Podemos retirar os `NAs` dessa coluna previamente utilizando a função `filter()`.

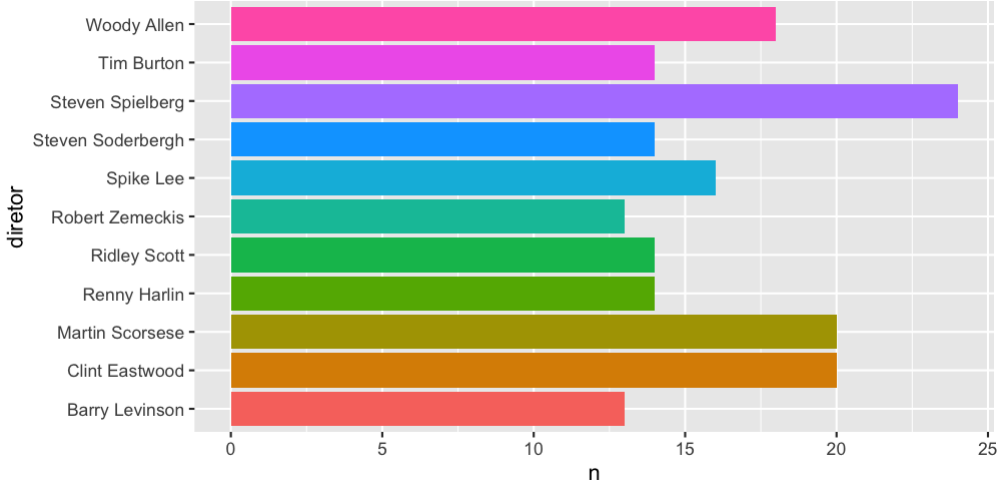
A seguir, além de retirar os `NAs`, atribuímos a coluna `diretor` à cor das colunas. Repare que, nesse caso, não utilizamos o atributo `color` e sim `fill`. A regra é a seguinte: o atributo `color` colore objetos sem área (pontos, linhas, contornos), o atributo `fill` preenche objetos com cor (barras, áreas, polígonos em geral).

```
imdb %>%
  count(diretor) %>%
  filter(!is.na(diretor)) %>%
  top_n(10, n) %>%
  ggplot() +
  geom_col(
    aes(x = diretor, y = n, fill = diretor),
    show.legend = FALSE
  )
```



Para consertar as labels do eixo `x`, a melhor prática é invertermos os eixos do gráfico, construindo barras horizontais.

```
imdb %>%
  count(diretor) %>%
  filter(!is.na(diretor)) %>%
  top_n(10, n) %>%
  ggplot() +
  geom_col(
    aes(x = diretor, y = n, fill = diretor),
    show.legend = FALSE
  ) +
  coord_flip()
```

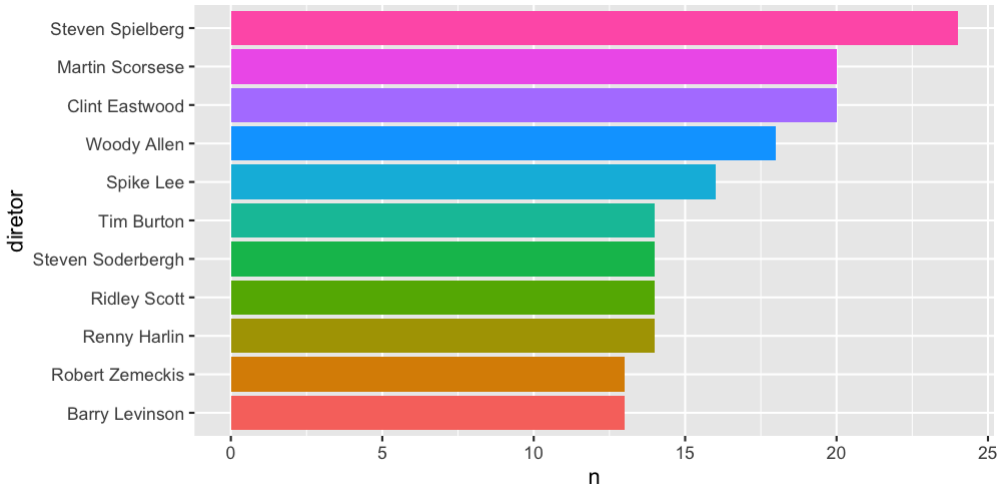


Para ordenar as colunas, precisamos mudar a ordem dos níveis do *fator* `diretor`. Para isso, utilizamos a função `fct_reorder()` do pacote `forcats`. A nova ordem será estabelecida pela coluna `n` (quantidade de filmes).

Fatores dentro do R são números inteiros (1, 2, 3, ...) que possuem uma representação textual. Variáveis categóricas são transformadas em fatores pelo ggplot pois todo eixo cartesiano é numérico. Assim, os textos de uma variável categórica são, internamente, números inteiros.

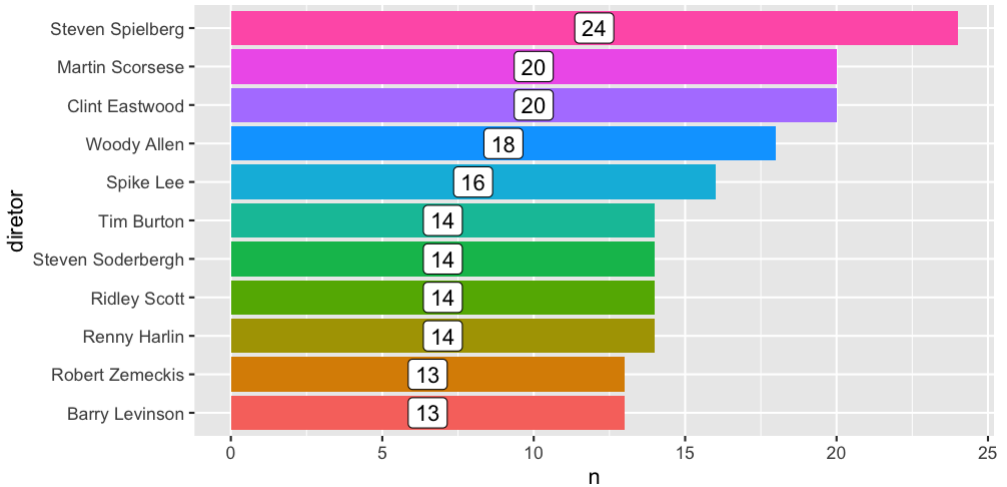
Por padrão, os inteiros são atribuídos a cada categoria de uma variável pela ordem alfabética (repare na ordem dos diretores nos gráficos anteriores). Assim, se transformássemos o vetor `c("banana", "uva", "melancia")` em um fator, a atribuição de inteiros seria: "banana" vira 1, "melancia" vira 2 e "uva" vira 3. Embora sejam inteiros internamente, sempre que chamássemos esse novo vetor, ainda sim veríamos os textos "banana", "uva" e "melancia".

```r
imdb %>%
  count(diretor) %>%
  filter(!is.na(diretor)) %>%
  top_n(10, n) %>%
  mutate(diretor = forcats::fct_reorder(diretor, n)) %>%
  ggplot() +
  geom_col(
    aes(x = diretor, y = n, fill = diretor),
    show.legend = FALSE
  ) +
  coord_flip()
```



Por fim, podemos colocar uma label com o número de filmes de cada diretor dentro de cada barra.

```
imdb %>%
  count(diretor) %>%
  filter(!is.na(diretor)) %>%
  top_n(10, n) %>%
  mutate(diretor = forcats::fct_reorder(diretor, n)) %>%
  ggplot() +
  geom_col(aes(x = diretor, y = n, fill = diretor), show.legend = FALSE) +
  geom_label(aes(x = diretor, y = n/2, label = n)) +
  coord_flip()
```



Histograma

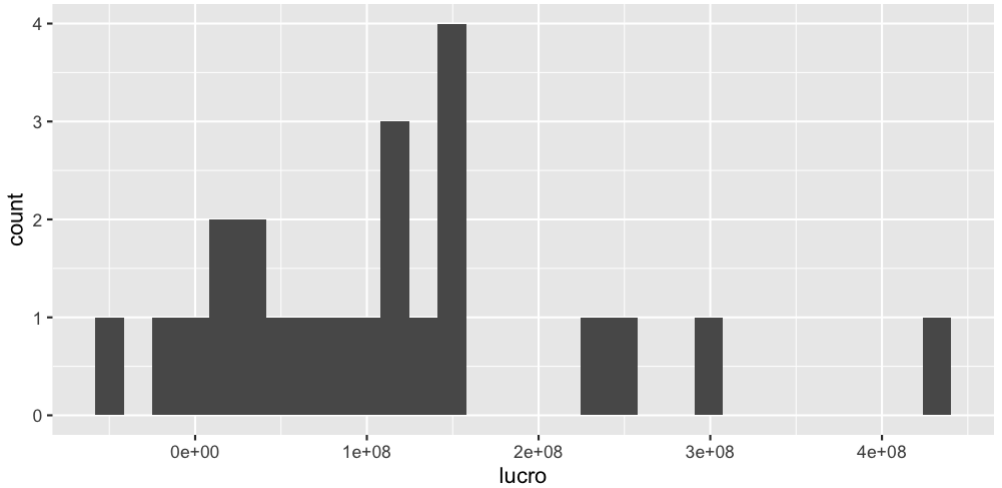
Para construir histogramas, usamos o `geom_histogram`. Esse `geom` só precisa do atributo `x` (o eixo y é construído automaticamente). Histogramas são úteis para avaliarmos a distribuição de uma variável.

A seguir, construímos o histograma do lucro dos filmes do diretor Steven Spielberg. O primeiro *warning* nos diz que o eixo x foi dividido em 30 intervalos para a construção do histograma.

```r
imdb %>%
  filter(diretor == "Steven Spielberg") %>%
  ggplot() +
  geom_histogram(aes(x = lucro))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

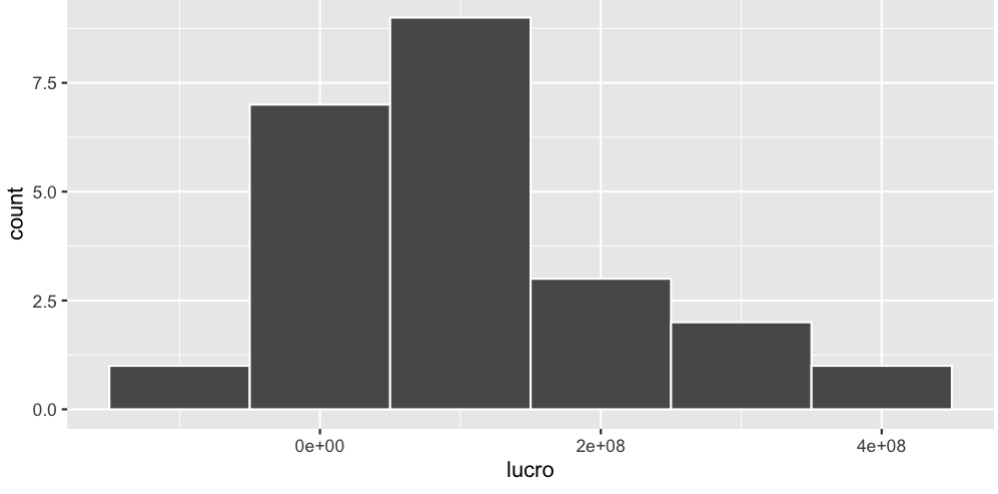
```
## Warning: Removed 1 rows containing non-finite values (stat_bin).
```



Para definir o tamanho de cada intervalo, podemos utilizar o argumento `bindwidth`.

```
imdb %>%
  filter(diretor == "Steven Spielberg") %>%
  ggplot() +
  geom_histogram(
    aes(x = lucro),
    binwidth = 100000000,
    color = "white"
  )
```

```
## Warning: Removed 1 rows containing non-finite values (stat_bin).
```



Boxplot

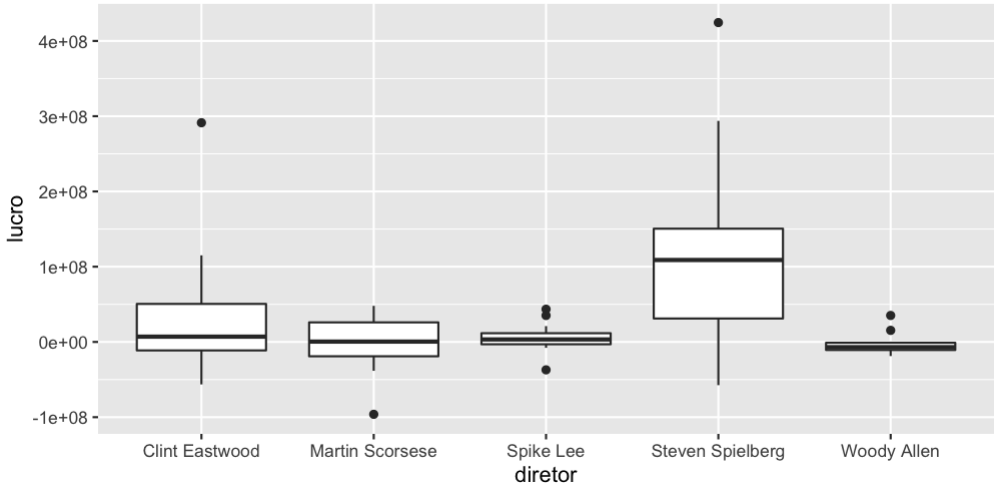
Boxplots também são úteis para estudarmos a distribuição de uma variável, principalmente quando queremos comparar várias distribuições.

Para construir um boxplot no ggplot, utilizamos a função `geom_boxplot`. Ele precisa dos atributos `x` e `y`, sendo que ao atributo `x` devemos mapear uma variável categórica.

A seguir, construímos boxplots do lucro dos filmes dos diretores que fizeram mais de 15 filmes.

```r
imdb %>%
  filter(!is.na(diretor)) %>%
  group_by(diretor) %>%
  filter(n() >= 15) %>%
  ggplot() +
  geom_boxplot(aes(x = diretor, y = lucro))
```

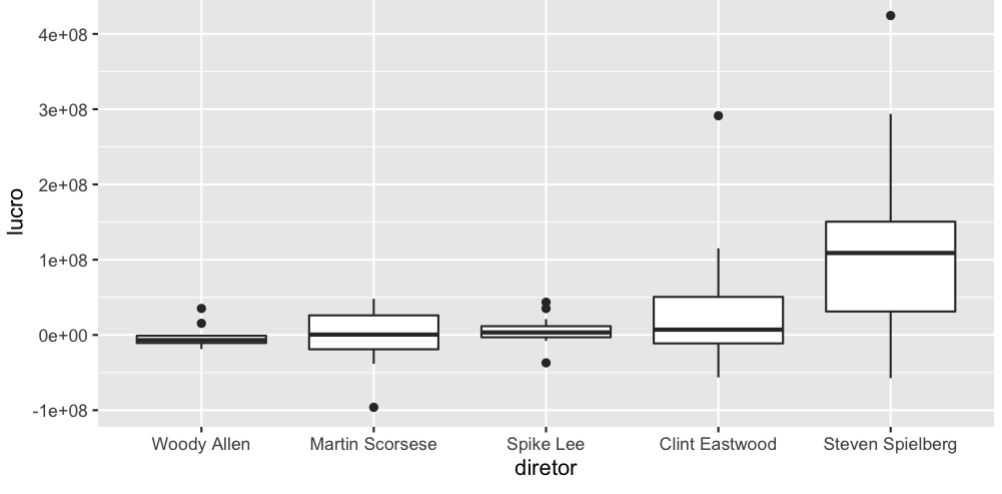
```
## Warning: Removed 10 rows containing non-finite values (stat_boxplot).
```



Também podemos reordenar a ordem dos boxplots utilizando a função `forcats::fct_reorder`.

```
imdb %>%
  filter(!is.na(diretor)) %>%
  group_by(diretor) %>%
  filter(n() >= 15) %>%
  ungroup() %>%
  mutate(diretor = forcats::fct_reorder(diretor, lucro, na.rm = TRUE)) %>%
  ggplot() +
  geom_boxplot(aes(x = diretor, y = lucro))
```

```
## Warning: Removed 10 rows containing non-finite values (stat_boxplot).
```



Títulos, labels e escalas

Para colocar títulos no gráfico ou trocar as labels dos atributos, utilizamos a função `labs()`.

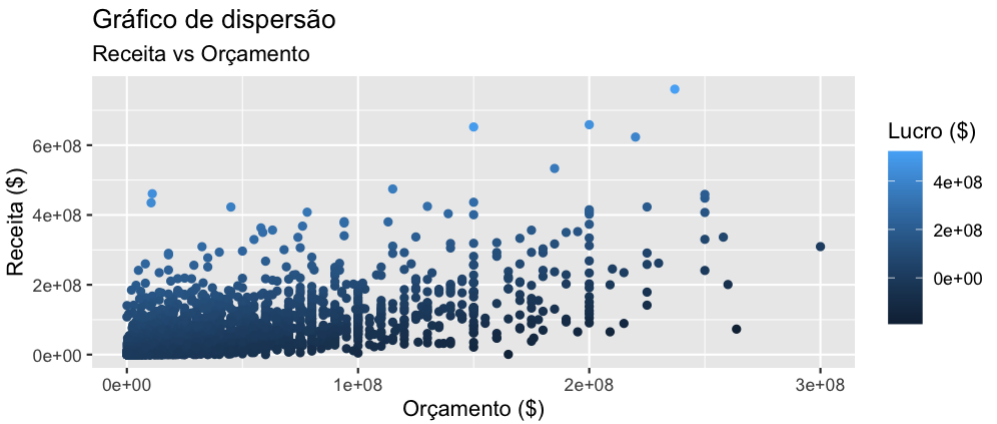
Para mudar as escalas (textos e quebras), utilizamos as funções da família `scale_`.

Podemos usar a função `coord_cartesian()` para definir qual porção do gráfico deve ser mostrada.

Colocando título, subtítulo e mudando as labels.

```
imdb %>%
  ggplot() +
  geom_point(mapping = aes(x = orcamento, y = receita, color = lucro)) +
  labs(
    x = "Orçamento ($)",
    y = "Receita ($)",
    color = "Lucro ($)",
    title = "Gráfico de dispersão",
    subtitle = "Receita vs Orçamento"
  )
```

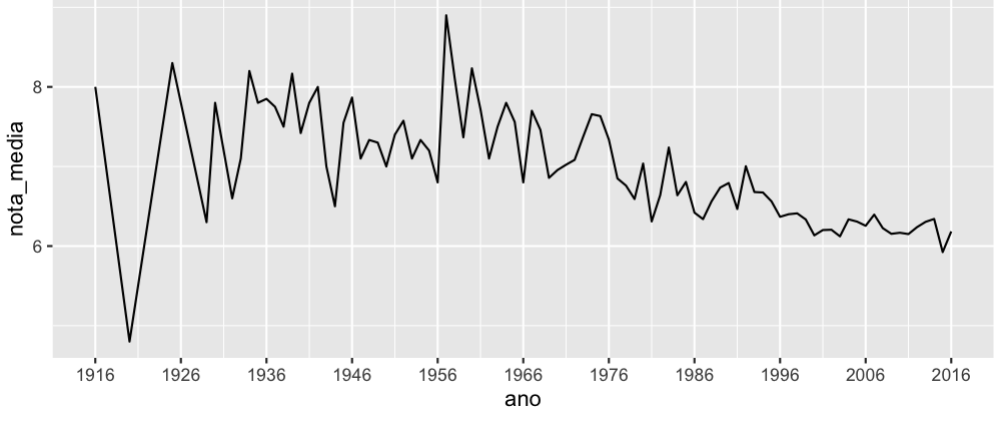
```
## Warning: Removed 720 rows containing missing values (geom_point).
```



Mudando as quebras dos eixos `x` e `y`.

```
imdb %>%
  group_by(ano) %>%
  summarise(nota_media = mean(nota_imdb, na.rm = TRUE)) %>%
  ggplot() +
  geom_line(aes(x = ano, y = nota_media)) +
  scale_x_continuous(breaks = seq(1916, 2016, 10)) +
  scale_y_continuous(breaks = seq(0, 10, 2))
```

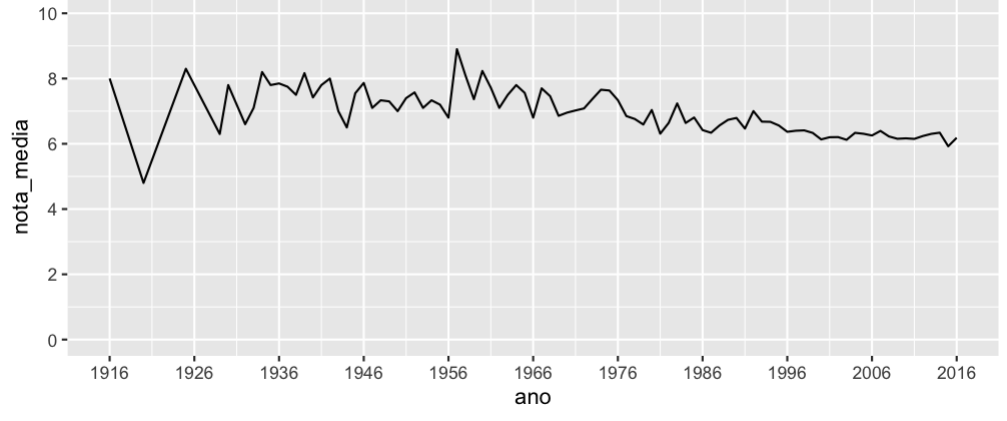
```
## Warning: Removed 1 row(s) containing missing values (geom_path).
```



Mudando a escala do gráfico.

```
imdb %>%
  group_by(ano) %>%
  summarise(nota_media = mean(nota_imdb, na.rm = TRUE)) %>%
  ggplot() +
  geom_line(aes(x = ano, y = nota_media)) +
  scale_x_continuous(breaks = seq(1916, 2016, 10)) +
  scale_y_continuous(breaks = seq(0, 10, 2)) +
  coord_cartesian(ylim = c(0, 10))
```

```
## Warning: Removed 1 row(s) containing missing values (geom_path).
```



Cores

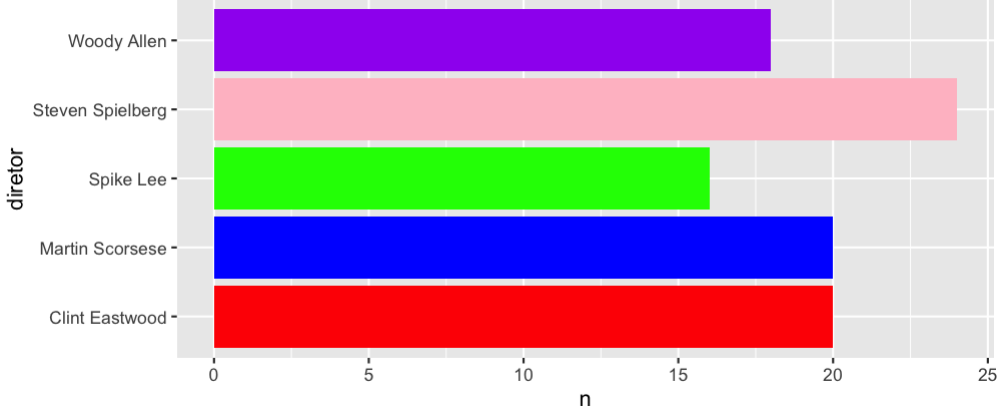
Também existe funções `scale` para os atributos de cor: `scale_color_` e `scale_fill_`.

Para escolher manualmente as cores de um gráfico, utilize as funções `scale_color_manual` e `scale_fill_manual()`.

Para trocar o nome nas legendas geradas por atributos de cor, utilize as funções `scale_color_discrete` e `scale_fill_discrete`.

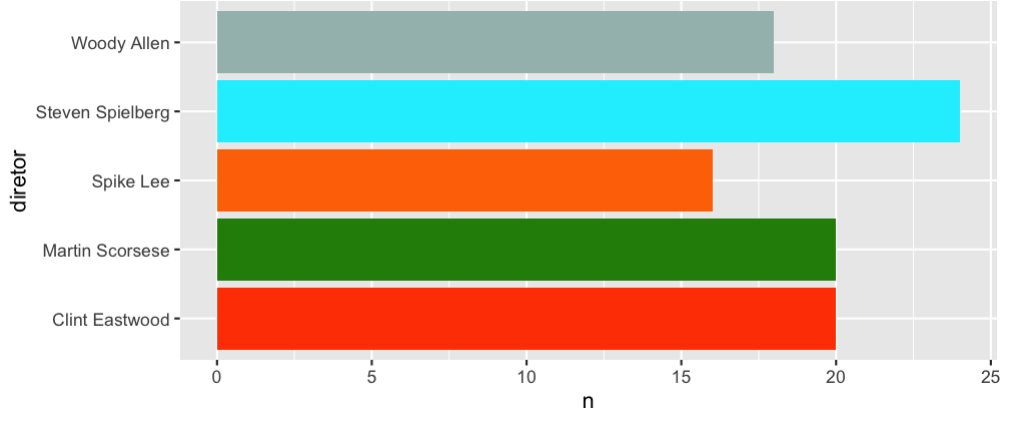
Substituindo as cores padrão do gráfico por um outro conjunto de cores.

```
imdb %>%
  count(diretor) %>%
  filter(!is.na(diretor)) %>%
  top_n(5, n) %>%
  ggplot() +
  geom_col(
    aes(x = diretor, y = n, fill = diretor),
    show.legend = FALSE
  ) +
  coord_flip() +
  scale_fill_manual(values = c("red", "blue", "green", "pink", "purple"))
```



Também podemos usar códigos hexadecimais.

```
imdb %>%
  count(diretor) %>%
  filter(!is.na(diretor)) %>%
  top_n(5, n) %>%
  ggplot() +
  geom_col(
    aes(x = diretor, y = n, fill = diretor),
    show.legend = FALSE
  ) +
  coord_flip() +
  scale_fill_manual(
    values = c("#ff4500", "#268b07", "#ff7400", "#0befff", "#a4bdba")
  )
```

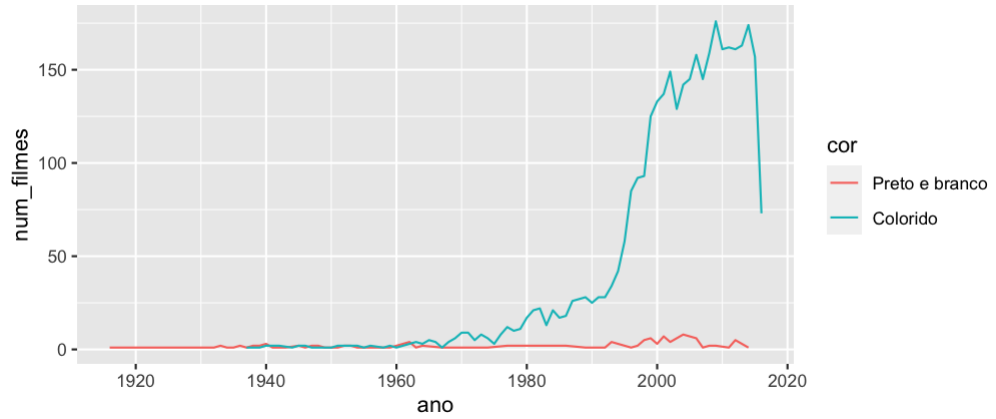


Trocando os textos da legenda.

```
imdb %>%
  filter(!is.na(cor)) %>%
  group_by(ano, cor) %>%
  summarise(num_filmes = n()) %>%
  ggplot() +
  geom_line(aes(x = ano, y = num_filmes, color = cor)) +
  scale_color_discrete(labels = c("Preto e branco", "Colorido"))
```

```
## `summarise()` has grouped output by 'ano'. You can override using the `.groups` argument.
```

```
## Warning: Removed 2 row(s) containing missing values (geom_path).
```



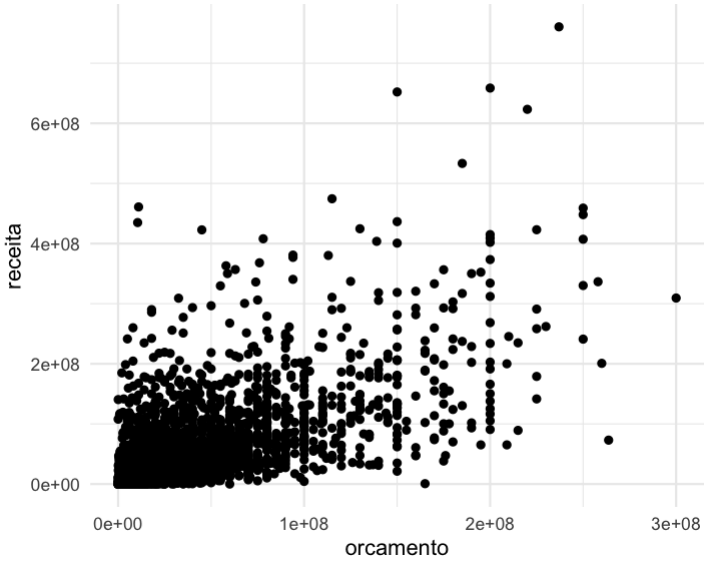
Temas

Os gráficos que vimos até agora usam o tema padrão do ggplot2. Existem outros temas prontos para utilizarmos presentes na família de funções `theme_`.

Você também pode criar o seu próprio tema utilizando a função `theme()`. Nesse caso, para trocar os elementos estéticos do gráfico precisamos usar as funções `element_text()` para textos, `element_line()` para linhas, `element_rect()` para áreas e `element_blank()` para remover elementos.

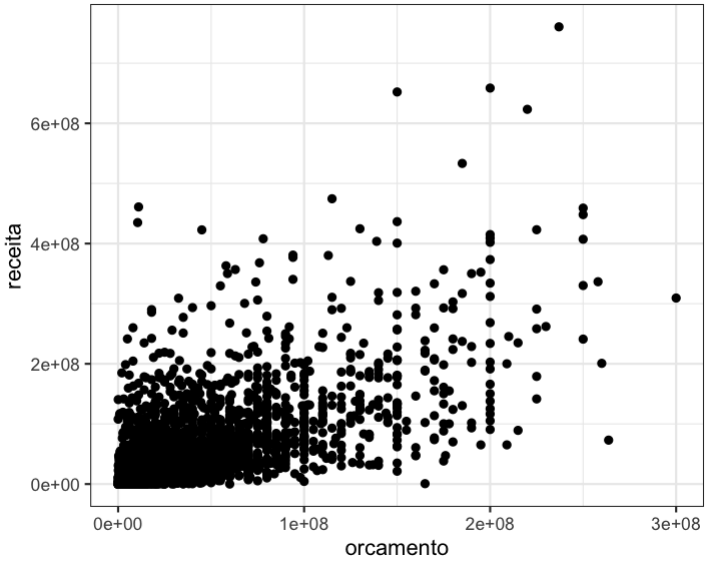
```
imdb %>%
  ggplot() +
  geom_point(mapping = aes(x = orcamento, y = receita)) +
  theme_minimal()
```

```
## Warning: Removed 720 rows containing missing values (geom_point).
```



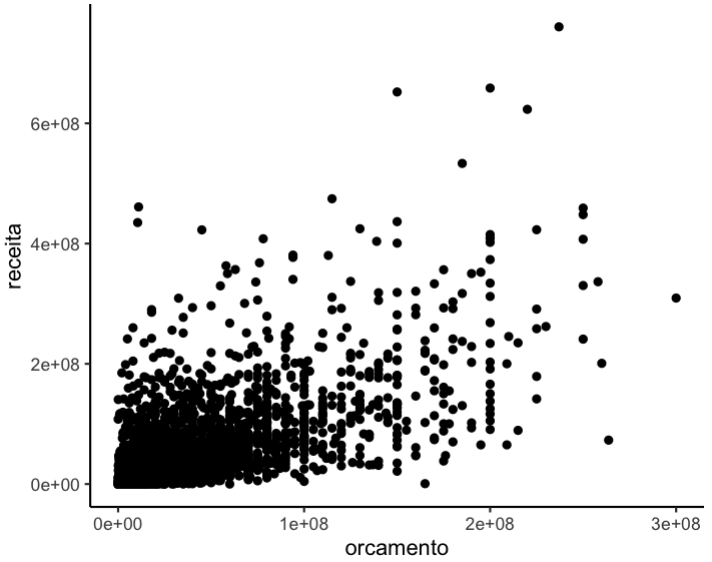
```
imdb %>%
  ggplot() +
  geom_point(mapping = aes(x = orcamento, y = receita)) +
  theme_bw()
```

```
## Warning: Removed 720 rows containing missing values (geom_point).
```



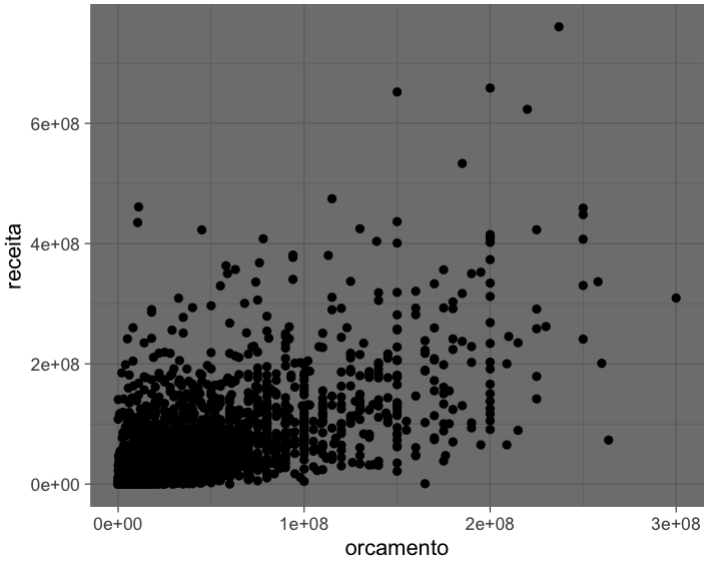
```
imdb %>%
  ggplot() +
  geom_point(mapping = aes(x = orcamento, y = receita)) +
  theme_classic()
```

```
## Warning: Removed 720 rows containing missing values (geom_point).
```



```
imdb %>%
  ggplot() +
  geom_point(mapping = aes(x = orcamento, y = receita)) +
  theme_dark()
```

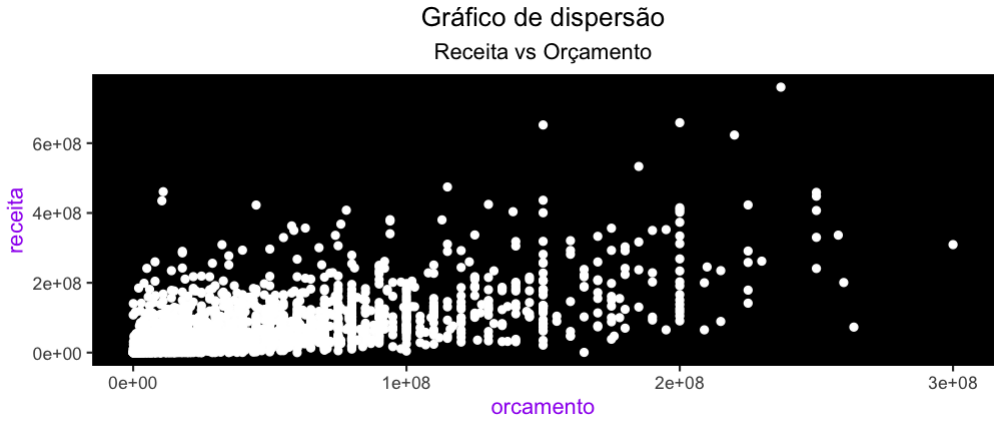
```
## Warning: Removed 720 rows containing missing values (geom_point).
```



Criando um próprio tema.

```
imdb %>%
  ggplot() +
  geom_point(mapping = aes(x = orcamento, y = receita), color = "white") +
  labs(title = "Gráfico de dispersão", subtitle = "Receita vs Orçamento") +
  theme(
    plot.title = element_text(hjust = 0.5),
    plot.subtitle = element_text(hjust = 0.5),
    axis.title = element_text(color = "purple"),
    panel.background = element_rect(fill = "black"),
    panel.grid = element_blank()
  )
```

```
## Warning: Removed 720 rows containing missing values (geom_point).
```



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/