Visualização de Dados

Introdução à ciência de dados

Daniel Brito dos Santos

2.1 Introdução

- Na aula de hoje vamos apresentar o pacote **ggplot2**.
  - Considerado o mais elegante e versátil sistema de gráficos disponível em R.
  - Implementa "**gramática dos gráficos**", um sistema coerente para descrever e construir gráficos. Assim vamos poder fazer mais coisas, mais rápido aprendendo um sistema que pode ser utilizado em muitos lugares.

- Vamos começar fazendo um “**scatterplot**” (gráfico de dispersão) para introduzir as peças fundamentais do ggplot2:
  - aesthetic mappings e geometric objects.
- Em seguida vamos construir visualizações de **distribuições** tanto de variáveis sozinhas quanto de **relações** entre variáveis.
- Vamos finalizar salvando os gráficos e apresentando dicas de para solução de possíveis problemas.

2.1.1 Pré-Requisitos

- ggplot2 faz parte do tidyverse, portanto vamos carrega-lo em memória

```
library("tidyverse")
#> ── Attaching packages ──────────────────────────────────────
#> ✔ ggplot2 3.4.0           ✔ purrr   1.0.0.9000
#> ✔ tibble  3.1.8           ✔ dplyr   1.0.99.9000
#> ✔ tidyr   1.2.1.9001      ✔ stringr 1.5.0
#> ✔ readr   2.1.3           ✔ forcats 0.5.2
#> ── Conflicts ─────────────────────────────────────────── ti
#> ✖ dplyr::filter() masks stats::filter()
#> ✖ dplyr::lag()    masks stats::lag()
```

- Esse comando carrega os principais pacores do tidyverse. Vamos utilizá-lo em quase todas as análises.
- Ele também informa quais funções conflitam, ou seja, que existem funções de mesmo nome em outros pacotes carregados ou no “base R”.

Palmerpenguins

- **palmerpenguins** que contém o dataset “penguins” com medidas corporais dos pinguins do arquipélago de Palmer.

```
1 library(palmerpenguins)
```

Primeiros passos

- Vamos criar nosso primeiro gráfico para responder a pergunta:
  - Pinguins com nadadeiras **mais longas** são **mais pesados** ou mais leves que pinguins com nadadeiras mais curtas?
- Você já deve suspeitar da resposta, mas vamos deixar-la bem precisa:

Pergunta precisa

- **Como** deve ser a **relação** entre comprimento de nadadeira e massa corporal? Positiva? Negativa? Linear? Não-linear?
  - Por acaso essa relação **varia** de acordo com a **espécie** do pinguim?
  - E com a **ilha** em que eles vivem?

2.2.1 o data frame *penguins*

- Data frame é uma coleção retangular de variáveis e observações em colunas e linhas
- *penguins* contem 344 observações coletadas pela Dra. Kristen Gorman e a Estação Palmer na antartica.

```
1 penguins
```

```
# A tibble: 344 × 8
   species island    bill_length_mm bill_depth_mm
flipper_…¹ body_…² sex    year
   <fct>   <fct>              <dbl>         <dbl>
<int>   <int> <fct> <int>
 1 Adelie  Torgersen           39.1          18.7
181    3750 male   2007
 2 Adelie  Torgersen           39.5          17.4
186    3800 fema…  2007
 3 Adelie  Torgersen           40.3          18
195    3250 fema…  2007
```

```
 4 Adelie  Torgersen           NA             NA
NA       NA <NA>   2007
 5 Adelie  Torgersen           36.7           19.3
193    3450 fema…  2007
```

2.2.1 o data frame *penguins*

- Podemos usar a função *glimpse()* para apresentá-lo de modo a vermos poucas observações de cada variável. Ou a função *View(penguins)* para abrir uma janela de visualização interativa.

```
1 glimpse(penguins)
```

```
Rows: 344
Columns: 8
$ species           <fct> Adelie, Adelie, Adelie, Adelie,
Adelie, Adelie, Adel…
$ island            <fct> Torgersen, Torgersen, Torgersen,
Torgersen, Torgerse…
$ bill_length_mm    <dbl> 39.1, 39.5, 40.3, NA, 36.7, 39.3,
38.9, 39.2, 34.1, …
$ bill_depth_mm     <dbl> 18.7, 17.4, 18.0, NA, 19.3, 20.6,
17.8, 19.6, 18.1, …
$ flipper_length_mm <int> 181, 186, 195, NA, 193, 190, 181,
195, 193, 190, 186…
$ body_mass_g       <int> 3750, 3800, 3250, NA, 3450, 3650,
```

```
3625, 4675, 3475, …
$ sex               <fct> male, female, female, NA, female,
male, female, male
```

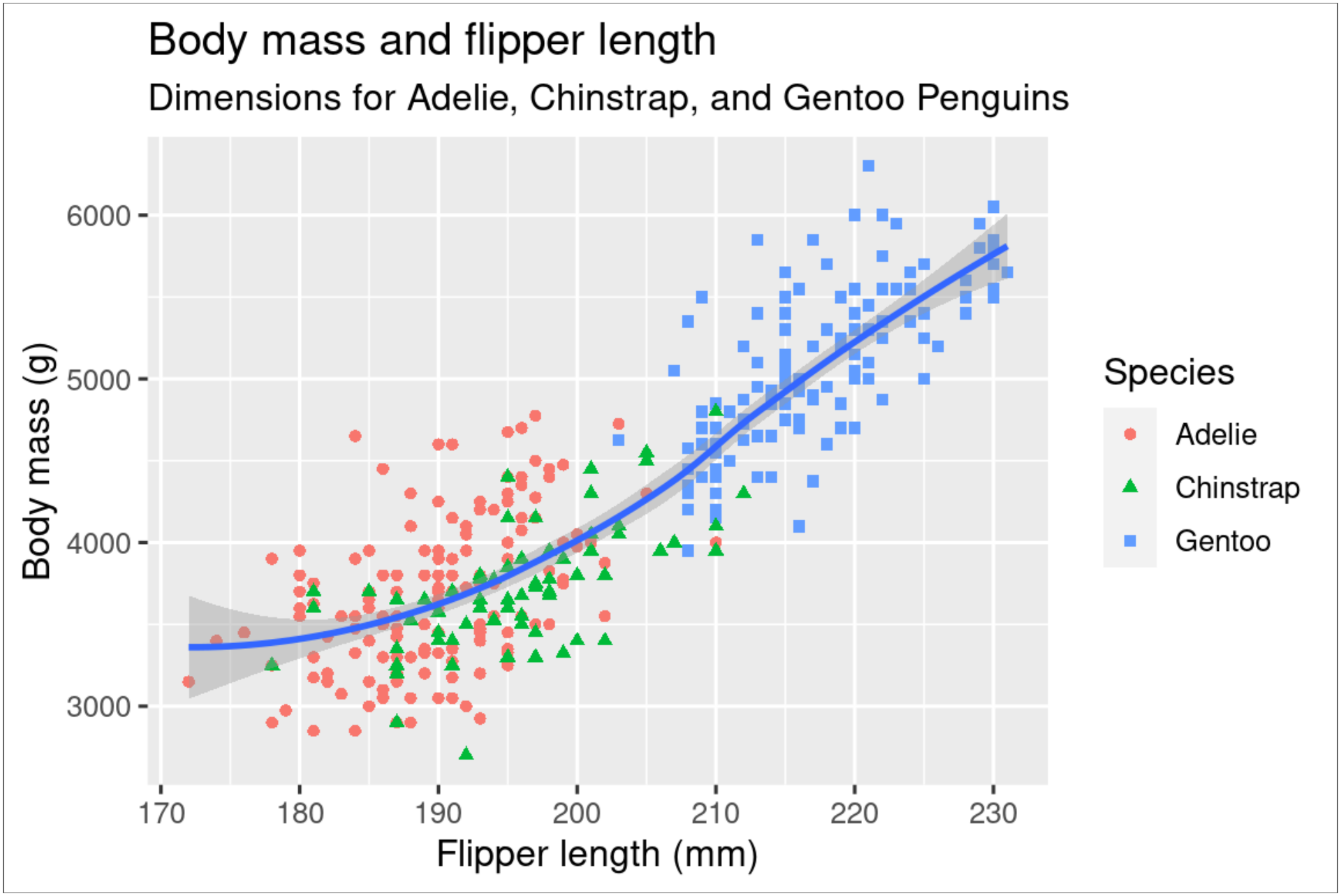
2.2.1 o data frame *penguins*

- Dentre as variáveis em *penguins* temos:
    1. *species*: a espécie do pinguim (Adelie, Chinstrap, ou Gentoo)
    2. *flipper_leght_mm*: comprimento da nadadeira do pinguim em milímitros
    3. *body_mass_g*: massa corporal do pinguim em gramas
- Para mais informações podemos utilizar o seguinte comando:

```
1 ?penguins
```

2.2.2 Objetivo final

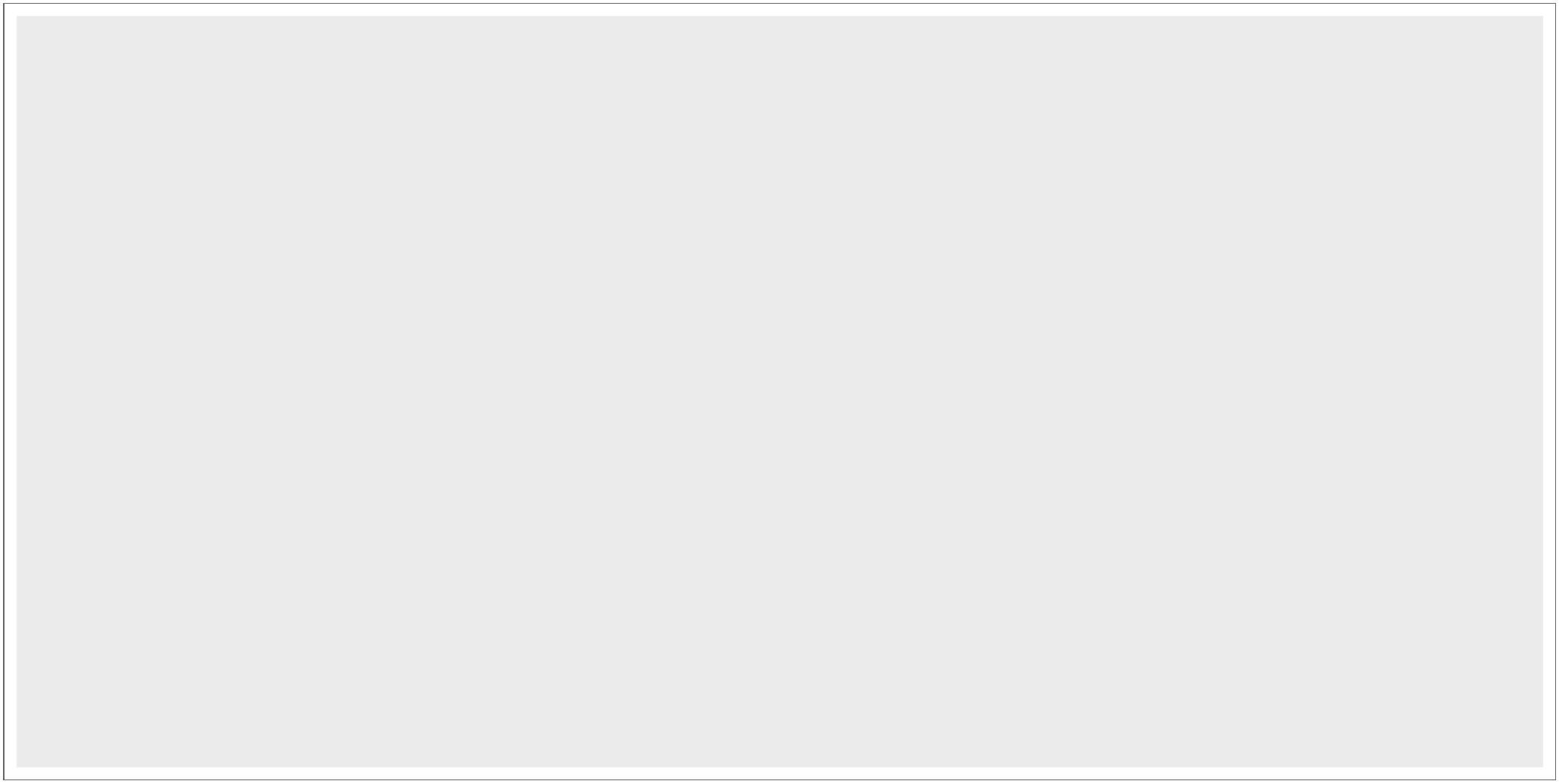
- Nosso objetivo final é recriar a seguinte visualização
  - Ela representa a relação entre o comprimento das nadadeiras e a massa corporal dos pinguins, considerando a espécie de cada um deles


Body mass and flipper length
Dimensions for Adelie, Chinstrap, and Gentoo Penguins
Body mass (g)
6000
5000
4000
3000
170
180
190
200
210
220
230
Flipper length (mm)
Species
Adelie
Chinstrap
Gentoo

Criando um gráfico com ggplot

- Vamos recriar o gráfico camada por camada
- Iniciamos com a função *ggplot()*que **define** um objeto de plotagem no qual podemos adicionar camadas.
    - O primeiro **argumento** dessa função é o *dataset* que será usado no gráfico:

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



- então *ggplot(data = penguins)* cria um gráfico vazio.
- Não é muito animador, mas é nessa tela que colocaremos as camadas do nosso gráfico.

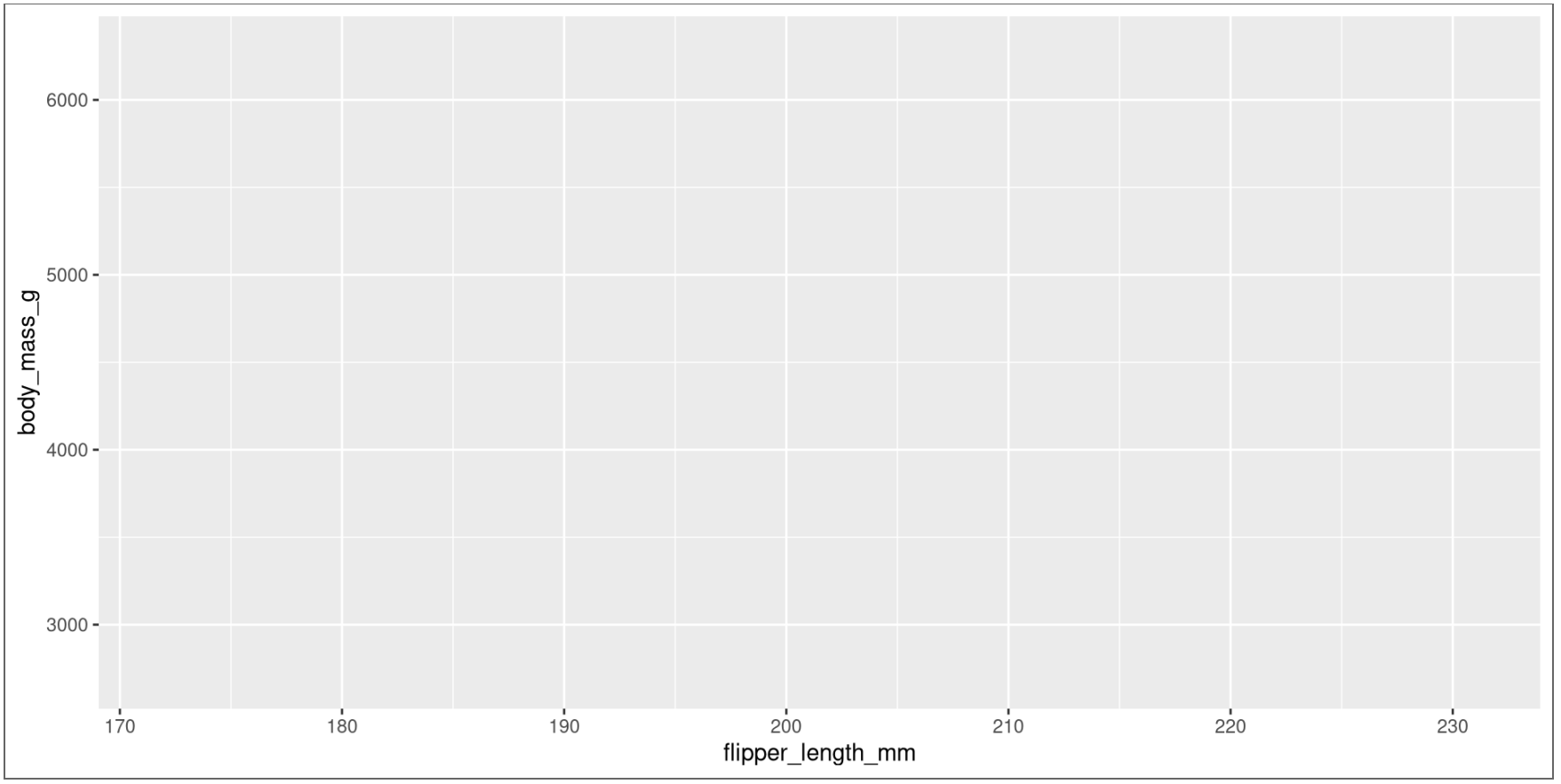
Criando um gráfico com ggplot

- Agora vamos informar a *ggplot()* quais as variáveis do data frame que queremos mapear para propriedades visuais (aesthetics) do plot.
- O argumento *mapping* da função *ggplot()* define como as variáveis do data set vão ser mapeadas para propriedades visuais do gráfico.
- O *mapping* sempre recebe o resultado da função **aes()**, que por sua vez tem os argumento **x** e **y** para especificar quais variáveis mapear com os eixos x e y

Criando um gráfico com ggplot

- Assim, vamos mapear o comprimento da nadadeira para o eixo x da **aesthetic** e a massa corporal para o eixo y da aesthetic.
- O ggplot vai automaticamente procurar essas variáveis no argumento *data*
- Vamos plotar o resultado de adicionar esses mapeamentos:

```
ggplot(
    data = penguins,
    mapping = aes(x = flipper_length_mm, y = body_mass_g)
)
```



- Vemos que o gráfico está preprado para apresentar as variáveis nos seus eixos adequados, mas ainda está em branco
- isto ocorre porque ainda não **articulamos** no código **como representar** as observações do nosso data frame no nosso gráfico

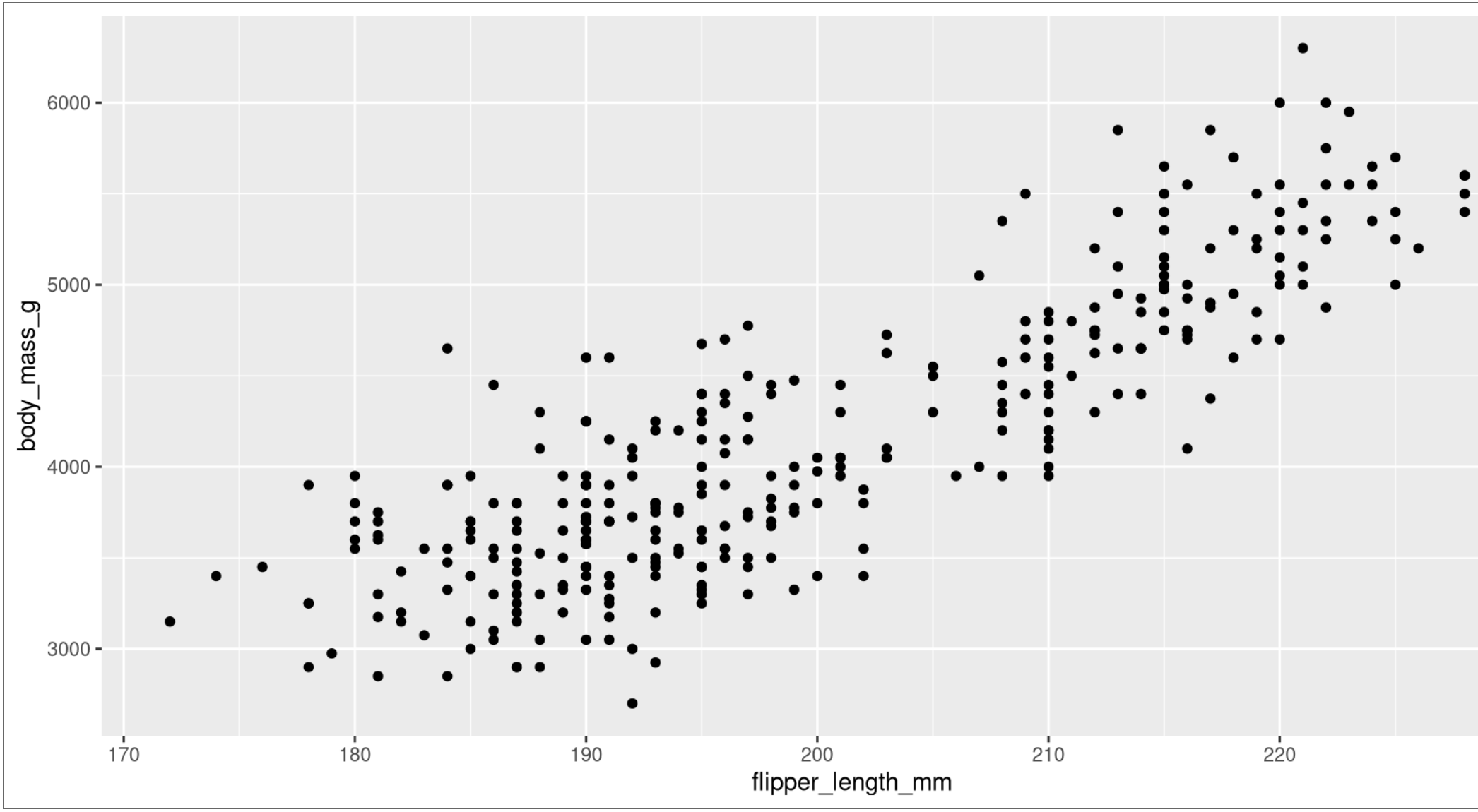
Criando um gráfico com ggplot

- Para articular essa representação precisamos definir uma *geom*. O objeto geométrico que o gráfico usa para representar dados
    - Esses objetos estão disponíveis no ggplot2 por meio de funções que iniciam com *geom_*.

Criando um gráfico com ggplot

- Gráficos geralmente são definidos pelo tipo de geometria que utilizam. Por exemplo, gráficos de barras usam *geom_bar()*, gráficos de linha usam *geom_line()*, boxplots usam *geom_boxplot()* e assim sucessivamente.
- O scatterplot quebra essa sequência, ele utiliza *geom_point()*.. Essa função adiciona uma camada de pontos ao gráfico, criando uma dispersão de pontos.

```
ggplot(
    data = penguins,
    mapping = aes(x = flipper_length_mm, y = body_mass_g)
) +
    geom_point()
#> Warning: Removed 2 rows containing missing values (`geo
```



- Agora sim temos um scatterplot!

- Ainda não responde nosso objetivo principal, mas já podemos ter uma noção da relação entre nadadeira e massa corporal dos pinguins.
    - Vemos que em geral é uma relação positiva, um tanto linear, e moderadamente forte.
    - Pinguins com nadadeiras maiores, em geral são mais pesados.

- Também podemos observar uma mensagem de aviso: “Removed 2 rows containing missing values (`geom_point()`).”
    - Essa mensagem está nos avisando que duas observações estavam com dados faltantes nas variáveis que tentamos exibir, portanto não tinham como esses pontos serem exibidos.
    - Não vamos nos preocupar muito com isso por enquanto, mas dados faltantes é uma das situações mais comuns em problemas do mundo real.

2.2.4 Adicionando estética e camadas

- **Scatterplots** são úteis para apresentar a **relação entre duas variáveis**, mas é sempre bom manter o ceticismo quanto a relação aparente.
- Por exemplo perguntar se **outras variáveis podem explicar** ou modificar a natureza da relação aparente.
- Vamos adicionar a espécie dos pinguins para ver se ela nos dá novos insights sobre a relação do comprimento da nadadeira e a massa corporal. Vamos **representar a espécie** como **cor** dos pontos no gráfico.

- Onde vocês acham que devemos colocar a espécie nesse código?

```
ggplot(
    data = penguins,
    mapping = aes(x = flipper_length_mm, y = body_mass_g)
) +
    geom_point()
```

- Assim, no mapeamento estético, dentro da função *aes()*

```
ggplot(
    data = penguins,
    mapping = aes(x = flipper_length_mm, y = body_mass_g,
) +
    geom_point()
```

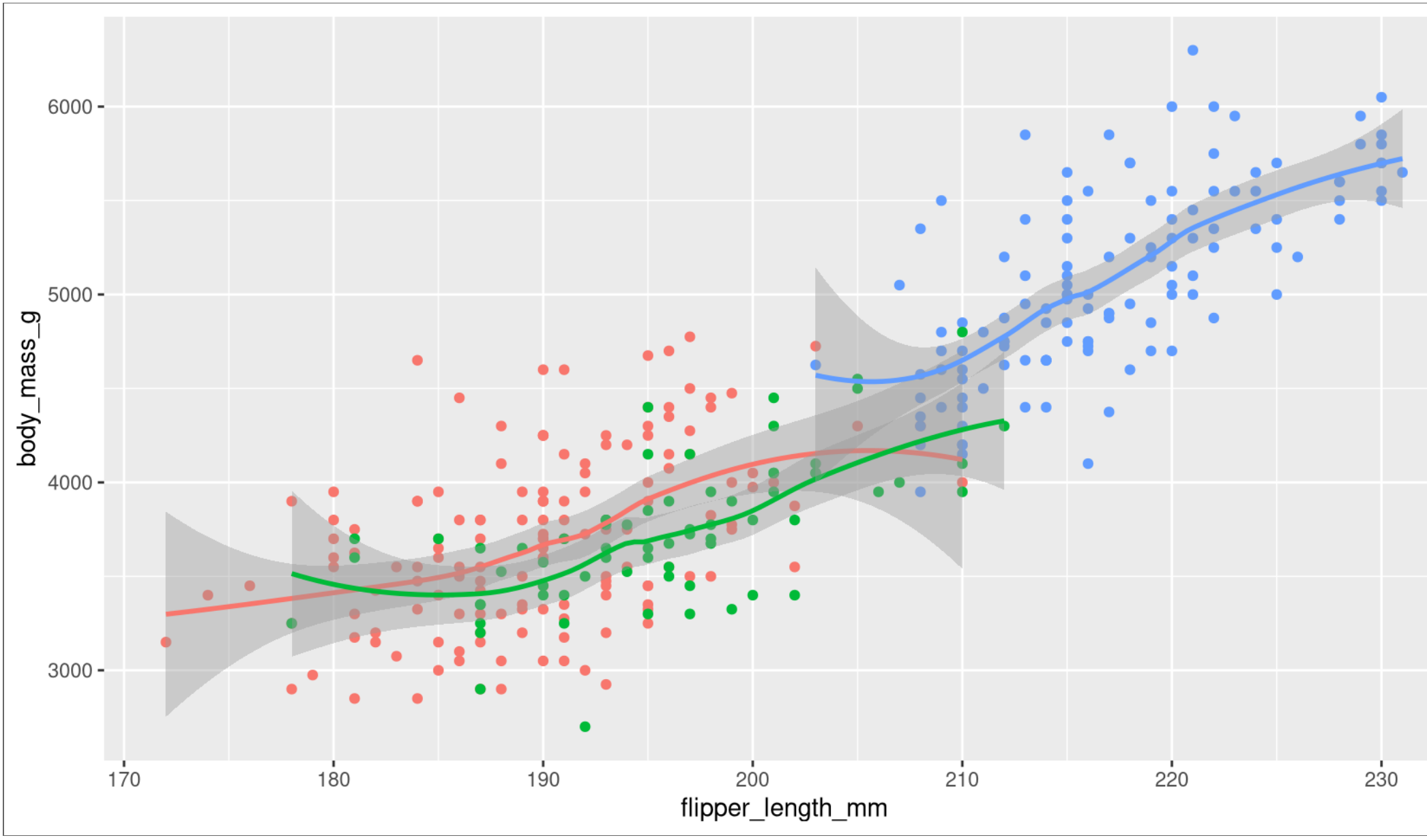
- Quando uma variável é mapeada a uma estética ggplot2 automaticamente atribui um valor estético único (nesse caso uma cor) para cada valor da variável (nesse cada cada uma das três espécies).*ggplot2* também adiciona a legenda explicando qual valor estético corresponde a qual valor da variável.

2.2.4 Adicionando estética e camadas

- Vamos agora adicionar uma nova camada: uma curva suave representando a relação entre a massa corporal e o comprimento de nadadeira dos pinguins.
- Onde vocês acham que precisamos mexer no código pra isso?

```
ggplot(
    data = penguins,
    mapping = aes(x = flipper_length_mm, y = body_mass_g,
    ) +
    geom_point() +
```

```
ggplot(
    data = penguins,
    mapping = aes(x = flipper_length_mm, y = body_mass_g,
    ) +
    geom_point() +
    geom_smooth()
```


6000
5000
4000
3000
body_mass_g
170
180
190
200
210
220
230
flipper_length_mm

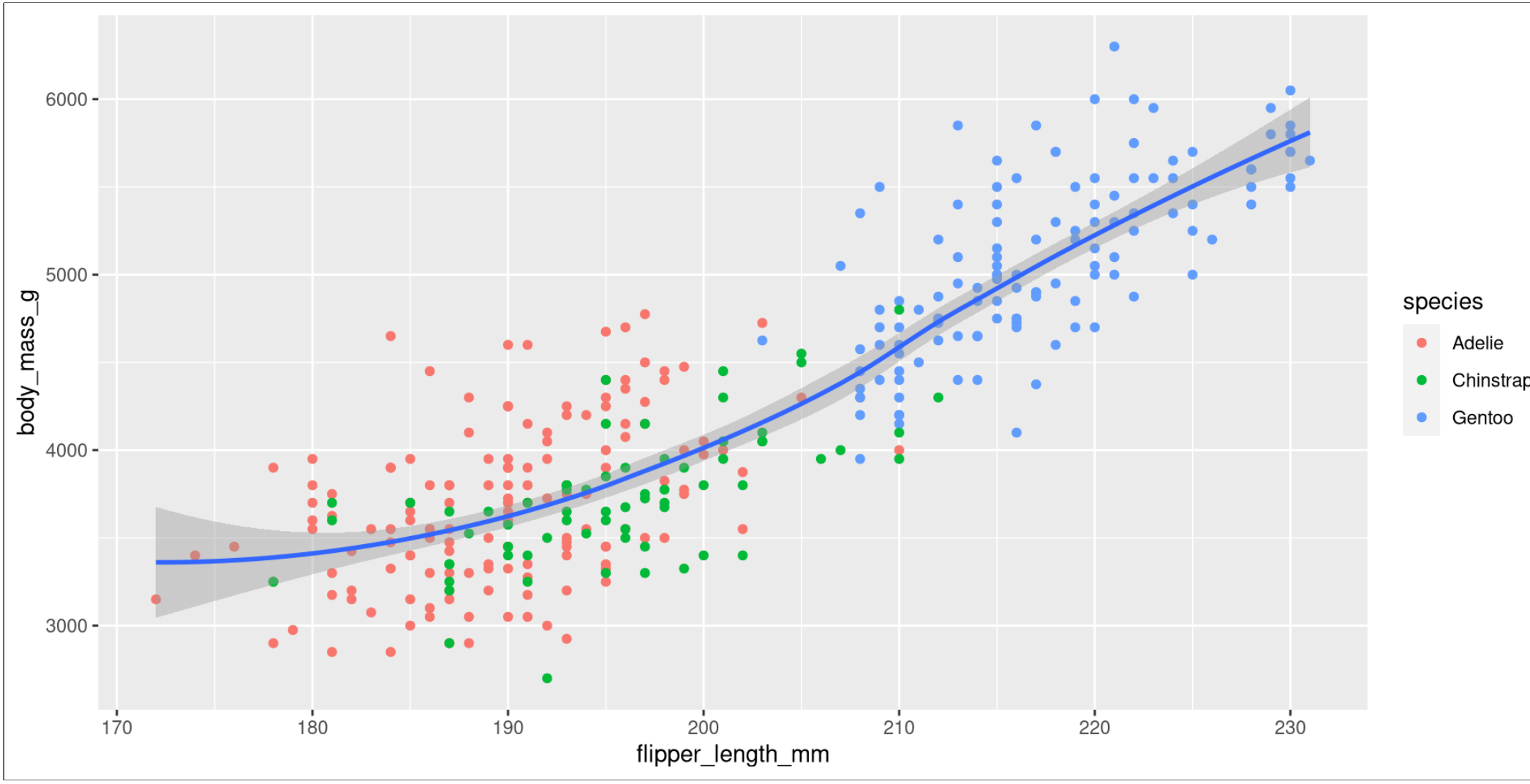
- Adicionamos com sucesso as curvas. Mas o gráfico não está igual nosso objetivo, que tem apenas uma curva, ao invés de uma curva para cada espécie
- Quando os mapeamentos estéticos são definidos na *ggplot()* eles são são **herdados** pelas camadas geométricas subsequentes.
    - Entretanto, cada função *geom_* **também** pode receber um argumento de mapping que permite mapeamento estético **local** naquela geometria

- Alguém se arrisca com o código?

```
ggplot(
    data = penguins,
    mapping = aes(x = flipper_length_mm, y = body_mass_g,
    ) +
    geom_point() +
    geom_smooth()
```

- Como queremos os pontos coloridos mas não queremos a curva separada por cor, devemos especificar *color = species* apenas para *gem_point().*

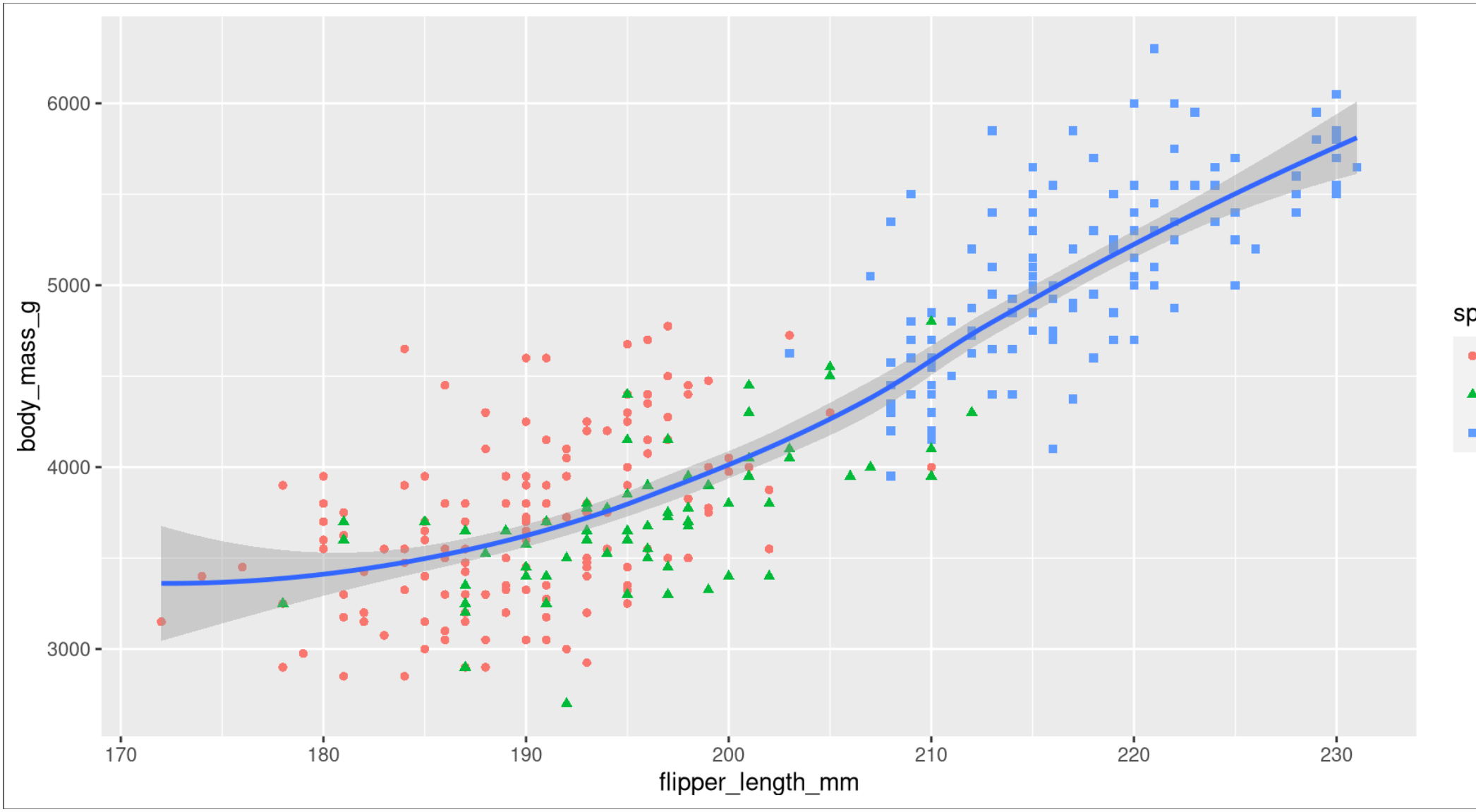
```
ggplot(
    data = penguins,
    mapping = aes(x = flipper_length_mm, y = body_mass_g)
) +
geom_point(mapping = aes(color = species)) +
geom_smooth()
```



- Vóila! Estamos quase! Agora só falta usar formas diferentes para representar as espécies além das cores.

- Em geral não é uma boa ideia utilizar apenas cores para representar informação em um gráfico porque as pessoas percebem cores de forma diferente, inclusive por daltonismo e outras diferenças de percepção cromática.
- Assim, além das cores, vamos mapear *species* a estética *shape*:

```
ggplot(
    data = penguins,
    mapping = aes(x = flipper_length_mm, y = body_mass_g)
) +
geom_point(mapping = aes(color = species, shape = species)
geom_smooth()
```

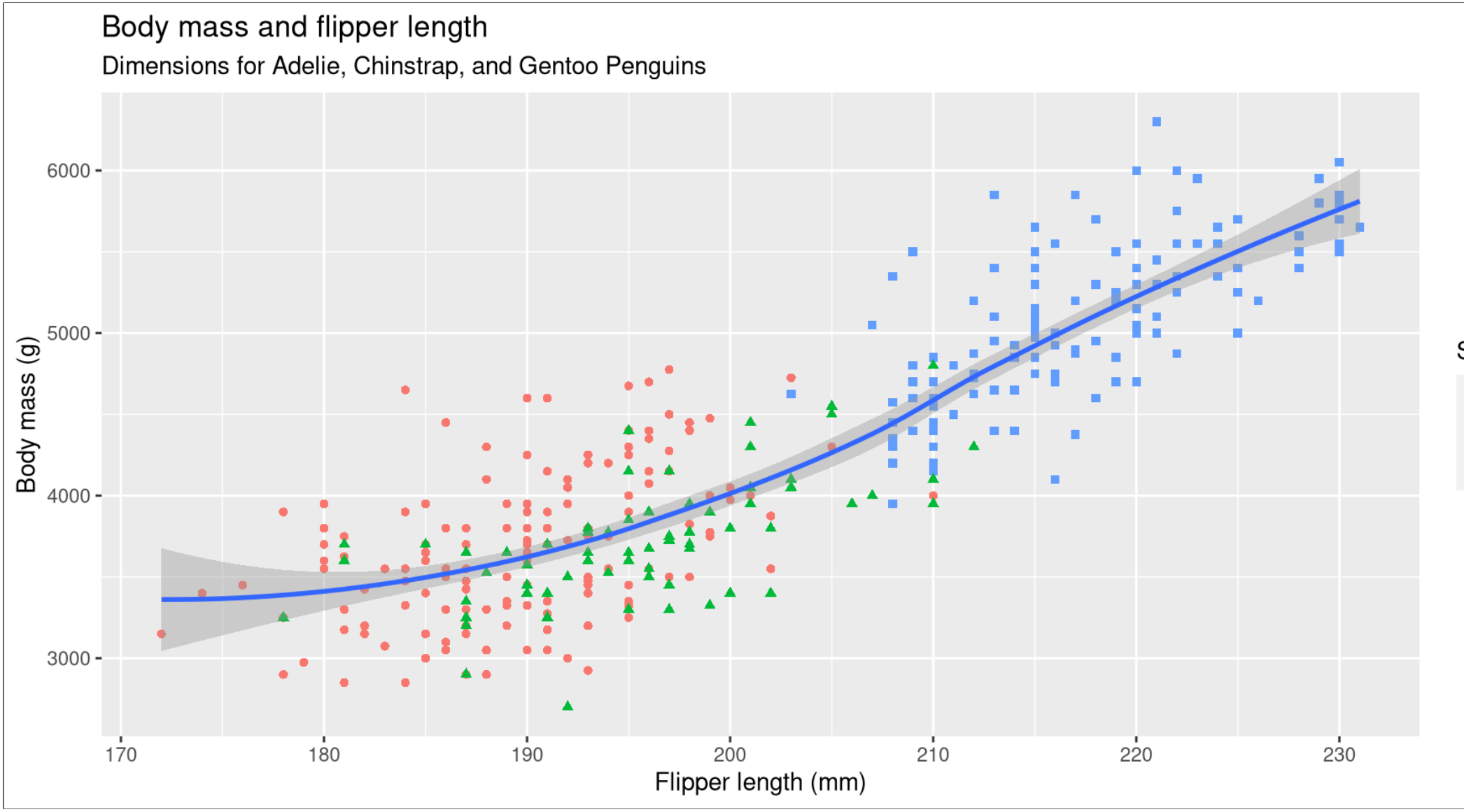


- Vemos que legenda também atualizou para refletir as formar diferentes.

- Finalmente, resta apenas ajustar as legendas do gráfico
  - Vamos usar a função *labs* em uma nova camada, seus argumentos são autoexplicativos:

```
ggplot(
    data = penguins,
    mapping =aes(x = flipper_length_mm, y = body_mass_g)
) +
    geom_point(aes(color = species, shape = species)
    ) +
    geom_smooth() +
    labs(    title = "Body mass and flipper length",
    subtitle = "Dimensions for Adelie, Chinstrap, and Gent
    x = "Flipper length (mm)",
    y = "Body mass (g)",
    color = "Species",
    shape = "Species"
    )
```

- Está pronto o sorvetinho:



2.3 Chamadas ggplot2

- Agora que já sabemos usar a função *ggplot()* vamos usar os argumentos posicionais ao invés das palavras chave "*data*" e "*mapping*". Passaremos a escrever assim:

```
1 ggplot(penguins,aes(x = flipper_length_mm, y = body_mass_g
2 geom_point()
```

- Futuramente vamos usar a notação mais comum que você vai encontrar por aí, utilizando pipes:

```
1 penguins |>
2 ggplot(aes(x = flipper_length_mm, y = body_mass_g)) +
3 geom_point()
```

Error