ggplot2

Nicholas A. C. Marino

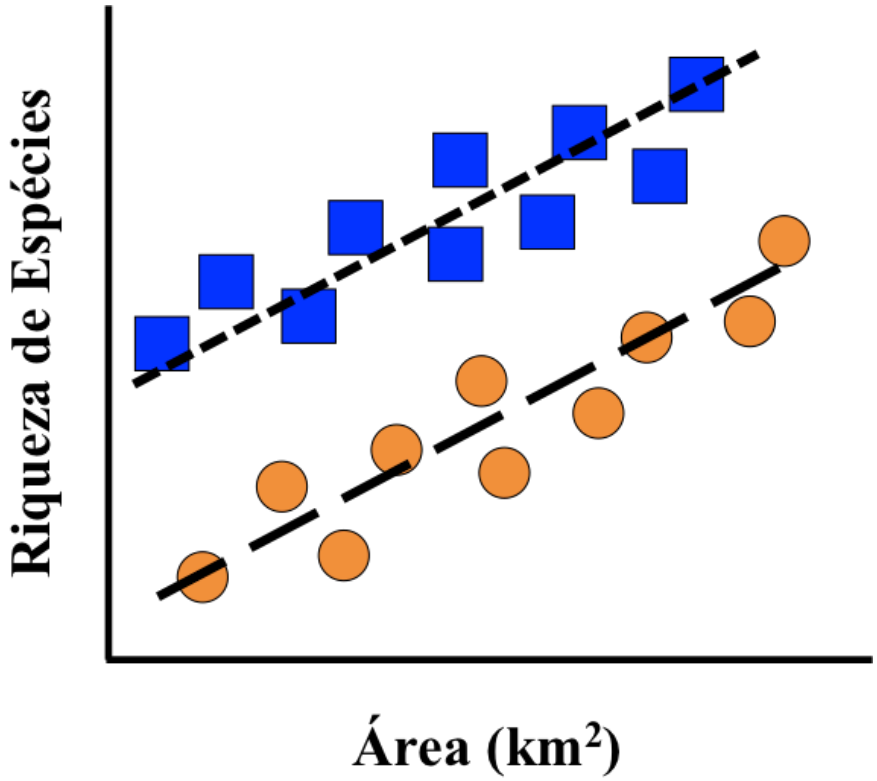
nac.marino@gmail.com

github.com/nacmarino/compartilhaR

Estrutura da Aula

1. Gramática dos Gráficos
2. `ggplot2`
3. Exportando figuras do `ggplot2`
4. Integrando o `ggplot2` ao `tidyverse`

Como você descreveria a figura abaixo?



Que figura pode ser feita com estes dados?

```
##   var1    var2   x   y
## 1    a fechado 2.0 1.0
## 2    b fechado 3.0 1.5
## 3    a fechado 2.5 1.4
## 4    b  aberto 5.0 4.2
## 5    a  aberto 6.0 4.0
## 6    b  aberto 5.5 4.6
```

Gramática dos Gráficos

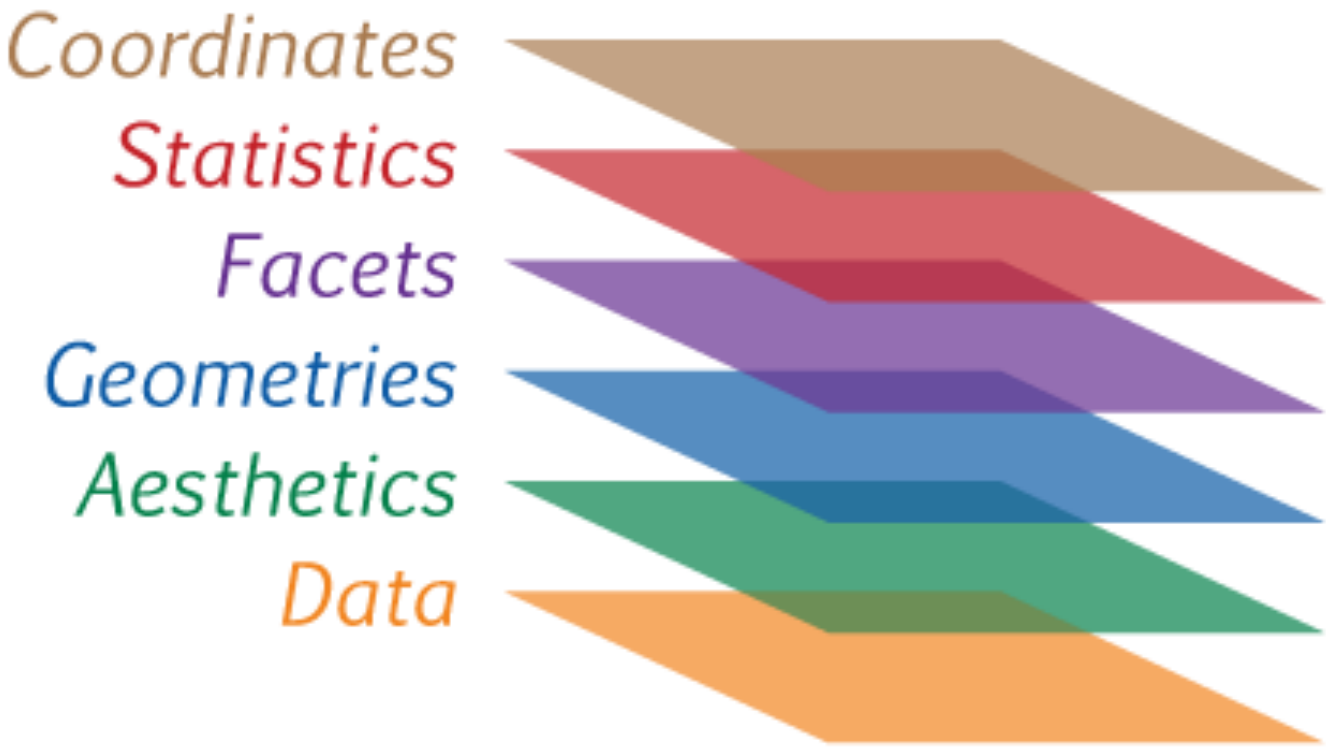
- Leland Wilkinson desenvolveu a ideia de que todo gráfico pode ser descrito por um conjunto de componentes semanticos:
  - um sujeito, que são as variáveis que queremos colocar no gráfico (*data*, *aesthetics*);
  - um ou mais verbos, que representam de que forma o sujeito será apresentado (*geometries*, *facets*);
  - um ou mais adjetivos, que descrevem características do sujeito (*color*, *fill*, *shape*, *alpha*).
- Ele organizou isto na forma da Gramática dos Gráficos (*The Grammar of Graphics*) - um esquema para facilitar e descrever a visualização dos dados.

Gramática dos Gráficos

“Resumidamente, a gramática nos diz que um gráfico deve ser mapeado (**mapping**) a partir dos dados (**data**), de acordo com a aparência (**aesthetic**; color, formato, tamanho) de objetos geométricos (**geometric**; pontos, linhas, barras). A figura também pode conter transformações estatísticas dos dados, sendo desenhada a partir de um sistema específico de coordenadas.” (Do livro "*ggplot2*")

Gramática dos Gráficos

- Uma figura é desenhada através da sobreposição de diferentes camadas de informações sobre os dados.



Gramática dos Gráficos

- A ideia principal por traz da Gramática dos Gráficos eé reduzir a distância entre a mente e a página, a ideia e a criação: você desenha, adicionando e modifica as camadas durante o processo de criação da figura.
- Isto traz algumas vantagens:
  - Não é preciso desenhar toda a figura de uma vez só (ao contrário das funcionalidades do pacote `lattice`);
  - Toda camada adicionada à figura pode ser alterada posteriormente (ao contrário das funcionalidades do pacote `base`);
  - Criação de figuras complexas com comandos simples e intuitivos (ao contrário dos pacotes `lattice` e `base`).

ggplot2

- É a implementação da Gramática dos Gráficos na linguagem de programação R.
- É um dos pacotes gráficos mais utilizados, por facilitar muito mais a visualização dos dados quando em comparação com os pacotes `base` e `lattice`.
- Escrito por Hadley Wickham (cientista chefe do RStudio), quando ainda era um estudante de pós-graduação.
- Podemos carregar este pacote usando o próprio nome `ggplot2` ou através do `tidyverse`.

```
library(tidyverse)
```

```
# outra opção - usar uma ou outra!
library(ggplot2)
```

Exercício 1

1. Importe para o R a tabela de dados das `ilhas.xlsx`, e atribua esta tabela ao objeto **ilhas**.
2. Carregue o conjunto de dados **gapminder** disponível no pacote `gapminder`.

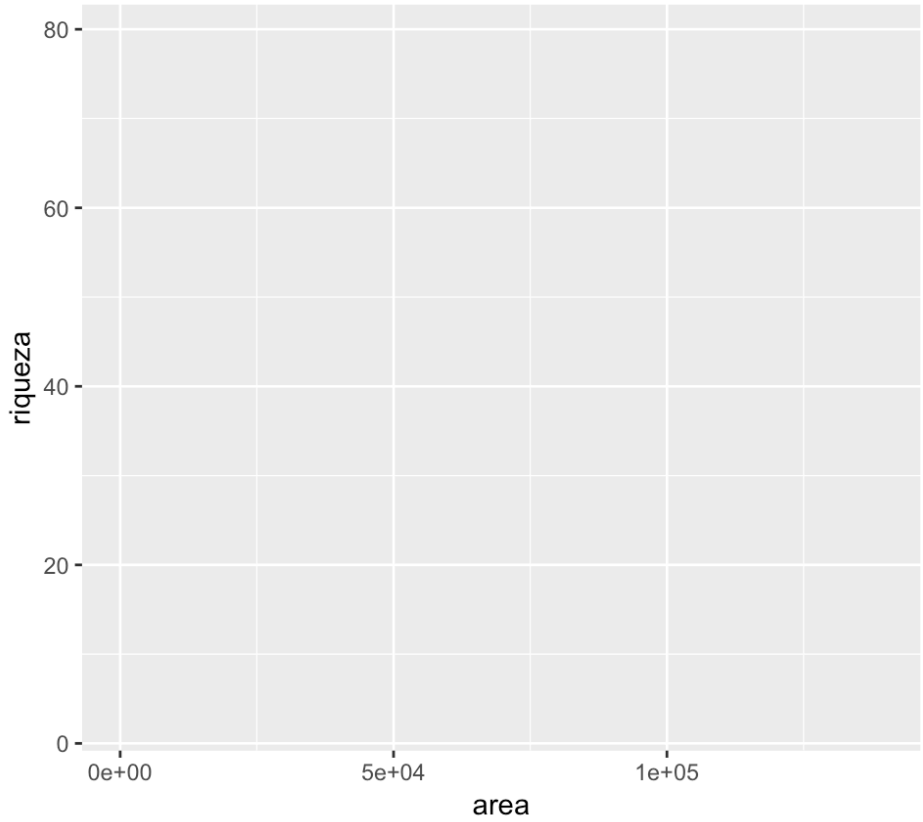
Começando uma figura

- Toda figura criada através do `ggplot2` começa da mesma forma: através da função `ggplot()`.
- Ainda assim, existem duas formas de iniciar uma figura através dessa função, sendo a primeira delas dizendo o que iremos desenhar: **mapearemos** uma unidade **estética** que virá de um conjunto de dados.

```
ggplot(data = ilhas, mapping = aes(x = area, y = riqueza))
```

Começando uma figura

```
ggplot(data = ilhas, mapping = aes(x = area, y = riqueza))
```



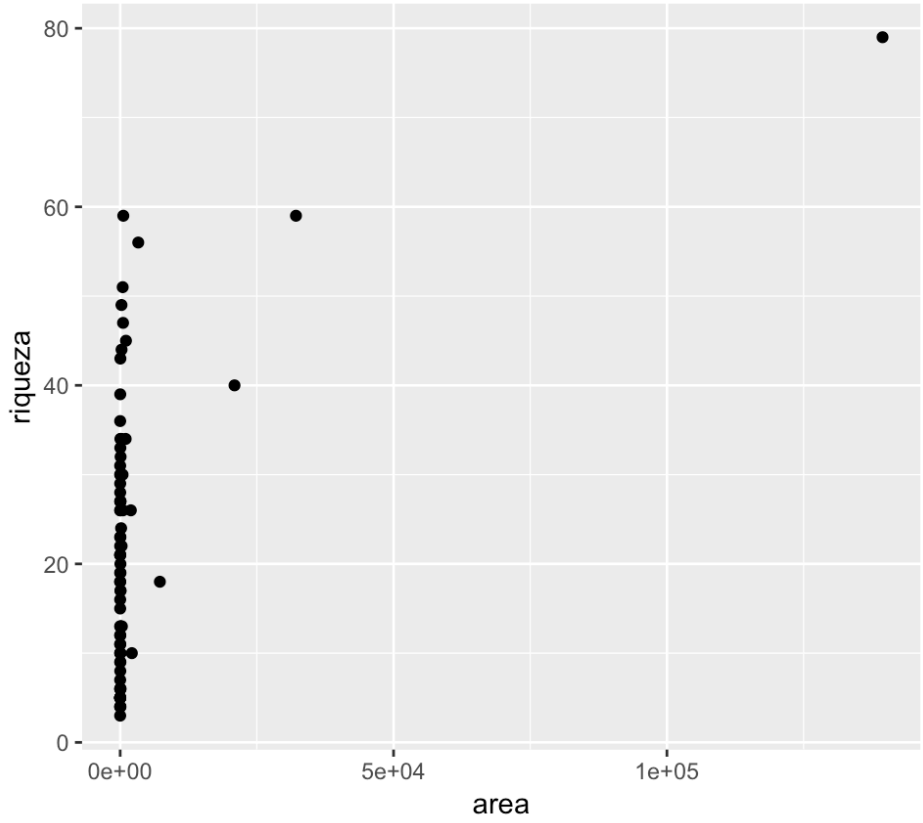
Definindo como `x` e `y` são representados

- Uma vez que já definimos que unidade **estética** será **mapeada**, precisamos definir de que forma esta relação será representada.
- A representação destas relações é feita através de representações **geométricas**, que indicam o tipo de figura que gostaríamos de produzir.
- Como queremos adicionar uma representação **geométrica** à unidade **estética** que foi **mapeada** precisamos, literalmente, adicionar uma camada à outra, utilizando o sinal + ao final da linha.

```
ggplot(data = ilhas, mapping = aes(x = area, y = riqueza)) +
  geom_point()
```

Definindo como `x` e `y` são representados

```
ggplot(data = ilhas, mapping = aes(x = area, y = riqueza)) +
  geom_point()
```



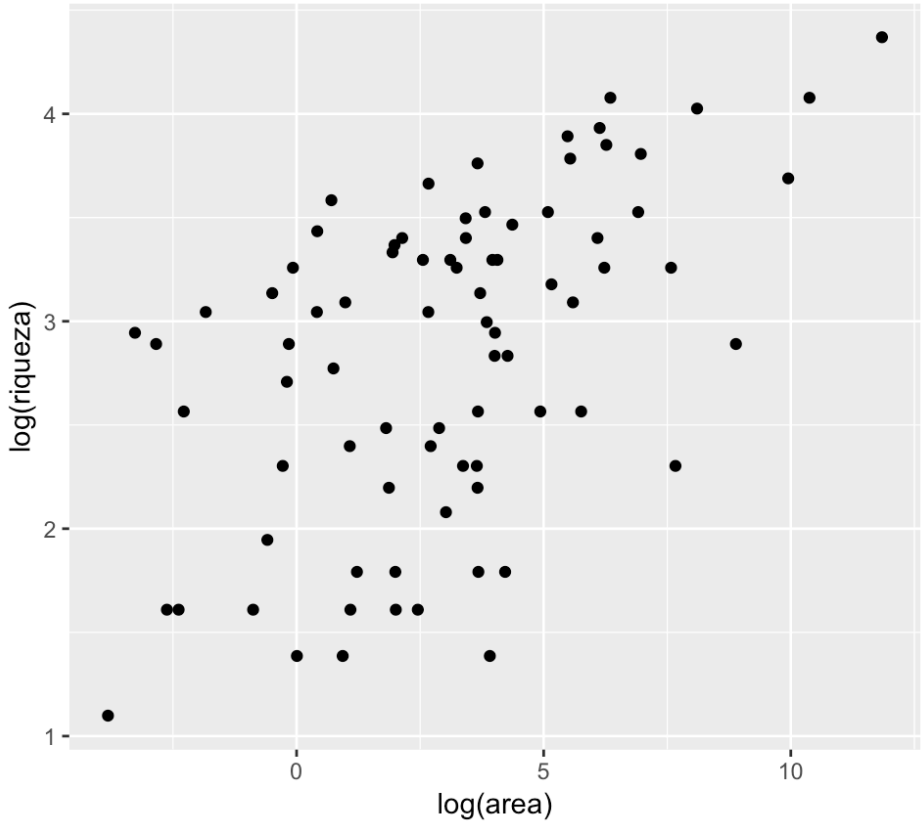
Definindo como `x` e `y` são representados

- Note, no entanto, que o gráfico sugere uma forte relação não-linear - o que é esperado ao plotarmos a área contra a riqueza de espécies em sua escala natural.
- Assim como outros pacotes, o `ggplot2` permite que adicionemos transformações às variáveis que estamos mapeando.

```
ggplot(data = ilhas, mapping = aes(x = log(area), y = log(riqueza))) +
  geom_point()
```

Definindo como `x` e `y` são representados

```
ggplot(data = ilhas, mapping = aes(x = log(area), y = log(riqueza))) +
  geom_point()
```



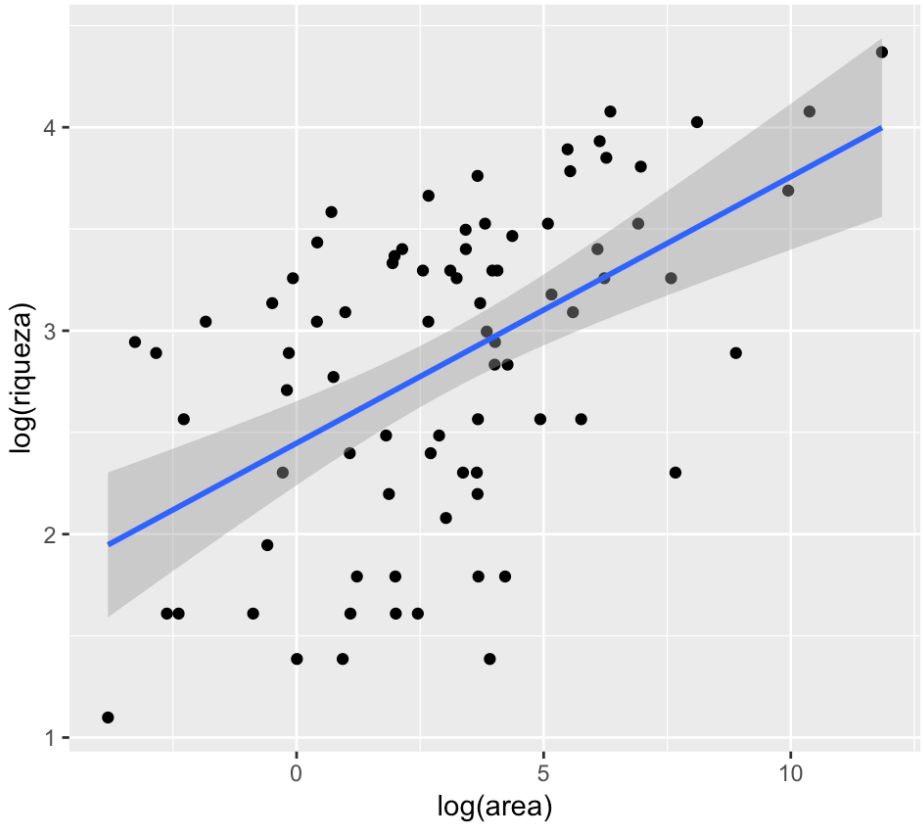
Representando a tendência da relação entre `x` e `y`

- Podemos adicionar tantas representações **geométricas** quanto quisermos à uma figura, desde que faça sentido adicionar uma camada à outra já existente.
- Por exemplo, podemos adicionar uma representação **geométrica** indicando a tendência da relação entre a área e a riqueza de espécies.

```
ggplot(data = ilhas, mapping = aes(x = log(area), y = log(riqueza))) +
  geom_point() +
  geom_smooth(method = "lm")
```

Representando a tendência da relação entre `x` e `y`

```
ggplot(data = ilhas, mapping = aes(x = log(area), y = log(riqueza))) +
  geom_point() +
  geom_smooth(method = "lm")
```



Outra forma de desenhar uma figura através do `ggplot2`

- Uma maneira alternativa a criar uma figura no `ggplot2` é começando através de um gráfico em branco, e ir adicionando as informações que queremos mapear camada a camada.
- Todavia, ao utilizar este método será necessário dizer à função quais são os dados e quais são os elementos estéticos que serão mapeados em cada camada.

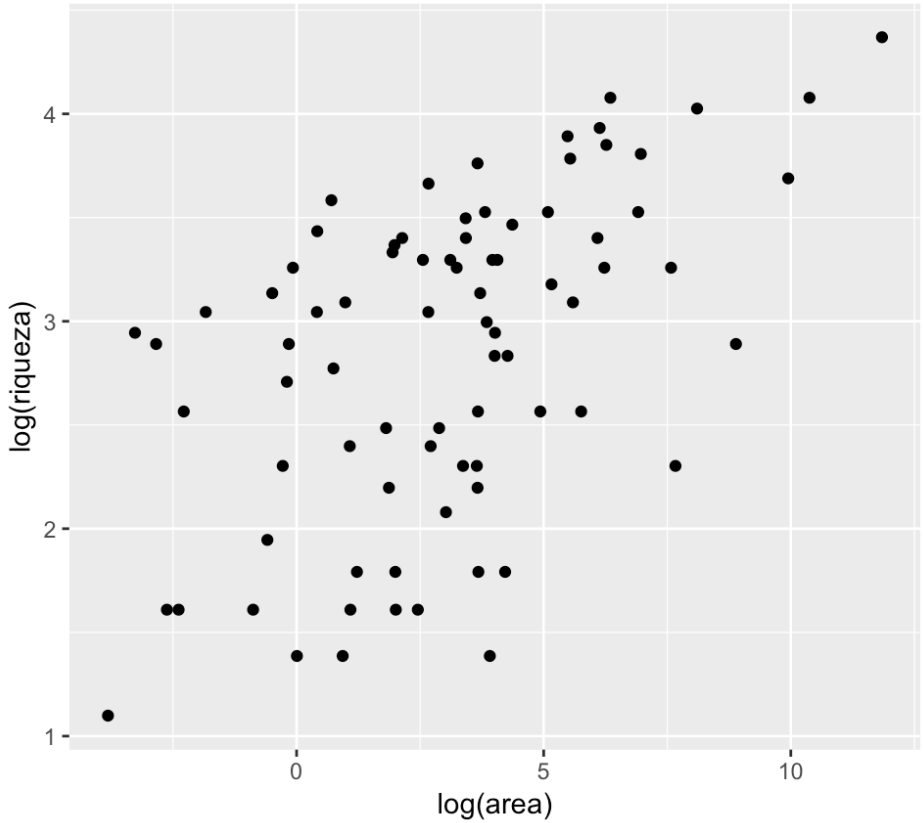
```
ggplot()
```

Outra forma de desenhar uma figura através do `ggplot2`

```
ggplot()
```

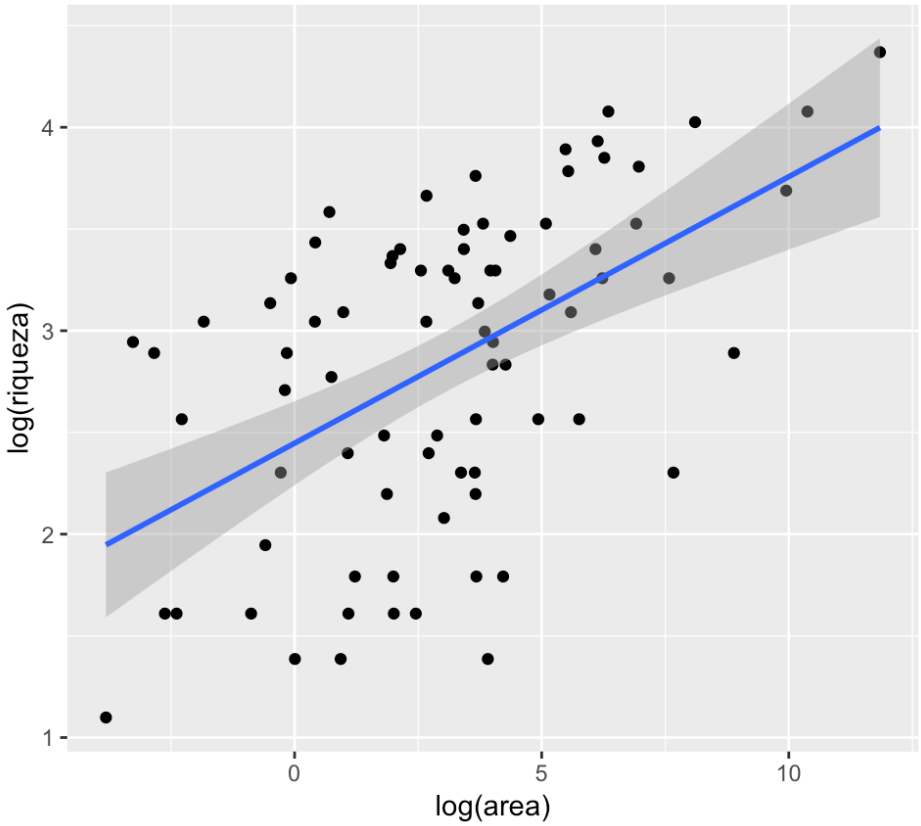
Outra forma de desenhar uma figura através do `ggplot2`

```
ggplot() +
  geom_point(data = ilhas, mapping = aes(x = log(area), y = log(riqueza)))
```



Outra forma de desenhar uma figura através do `ggplot2`

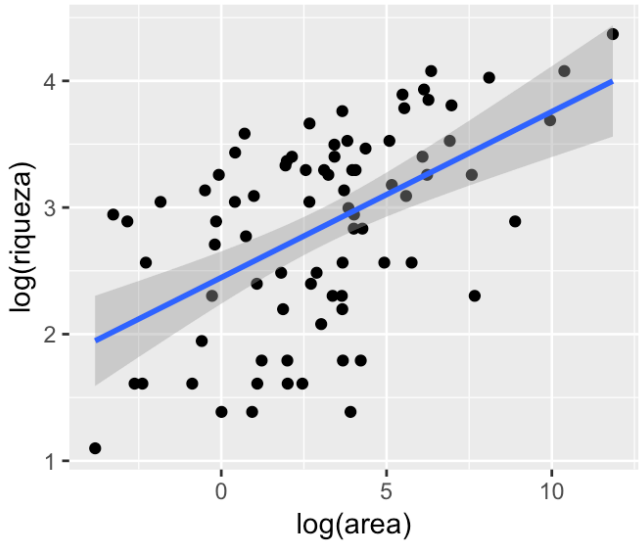
```
ggplot() +
  geom_point(data = ilhas, mapping = aes(x = log(area), y = log(riqueza))) +
  geom_smooth(data = ilhas, mapping = aes(x = log(area), y = log(riqueza)), method = "lm")
```



Outra forma de desenhar uma figura através do `ggplot2`

- Finalmente, uma terceira forma de iniciar uma figura é determinando apenas o conjunto de dados de onde sairão todas as informações, e determinar o que será mapeado em cada camada

```
ggplot(data = ilhas) +
  geom_point(mapping = aes(x = log(area), y = log(riqueza))) +
  geom_smooth(mapping = aes(x = log(area), y = log(riqueza)), method = "lm")
```



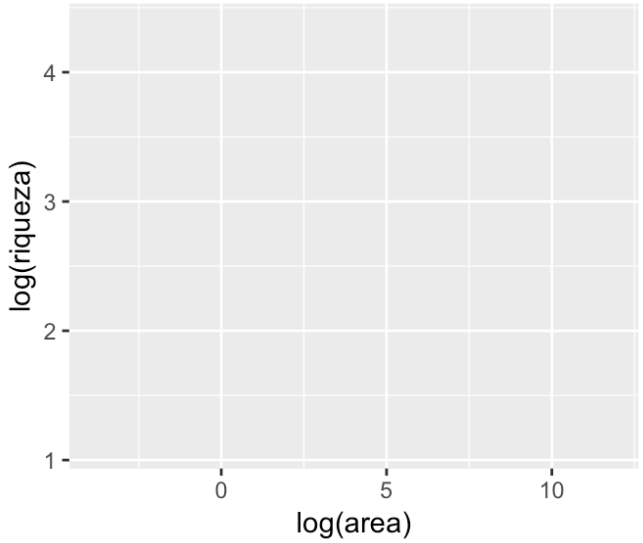
Mapear desde o início ou não?

- Existe uma diferença fundamental na forma como começamos a desenhar um gráfico no `ggplot2`, que podemos usar ao nosso favor dependendo do objetivo e dados que temos:
  - **Especificar apenas o conjunto de dados na função `ggplot`:** útil quando todas as informações que irão para cada `geom` estão dentro de uma mesma tabela.
  - **Especificar o conjunto de dados e a unidade estética dentro da função `geom`:** útil quando cada `geom` representa informações presentes em conjunto de dados diferentes.
  - **Especficiar o conjunto de dados e a unidade estética dentro da função `ggplot`:** útil quando toda a figura é feita com referência àquela unidade estética vinda daquele conjunto de dados.
- Dependendo da forma como você escolher desenhar a figura, você pode ter problemas ao definir alguns outros elementos gráficos - falaremos mais sobre isso depois.

Criando uma figura do `ggplot2` objeto-a-objeto

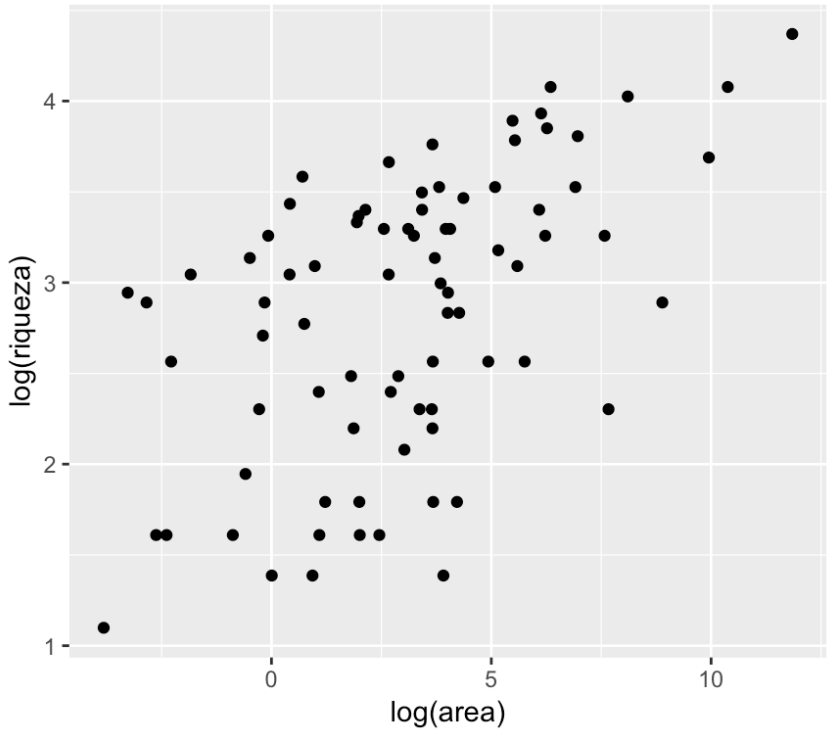
- Não precisamos criar todas as camadas de uma vez ao desenhar uma figura no `ggplot2`: podemos criar uma camada, atribuir ela a um objeto, adicionar mais uma camada a esse objeto e sobre-escrevelo, e assim sucessivamente.

```
figura <- ggplot(data = ilhas, mapping = aes(x = log(area), y = log(riqueza)))
figura
```



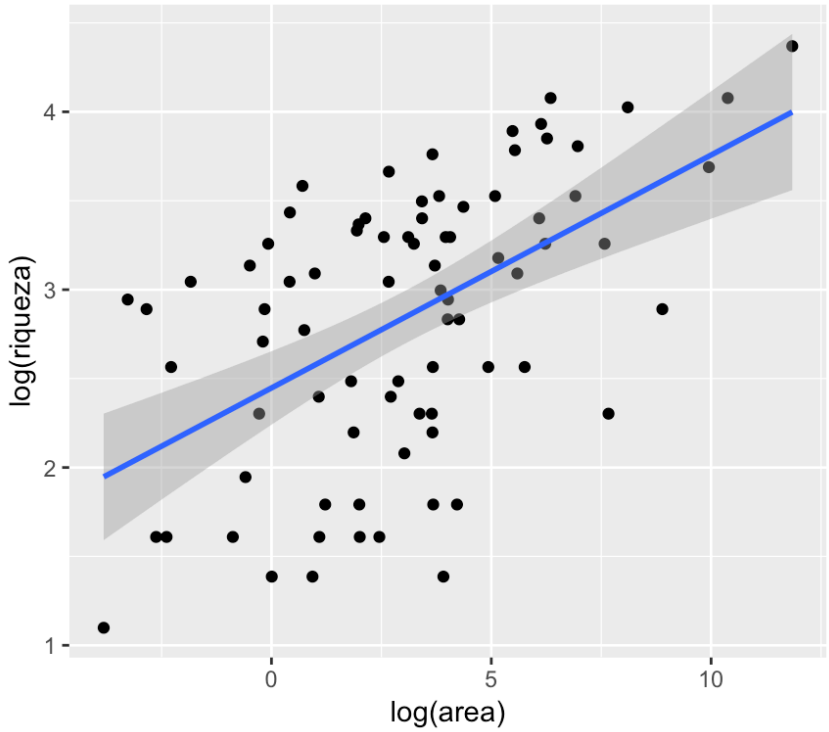
Criando uma figura do `ggplot2` objeto-a-objeto

```
figura <- figura +
    geom_point()
figura
```



Criando uma figura do `ggplot2` objeto-a-objeto

```
figura <- figura +
    geom_smooth(method = "lm")
figura
```



Exercício 2

1. Crie uma figura demonstrando a relação entre a riqueza de espécies de mamíferos e a produtividade primária das ilhas.
2. Crie um `boxplot` demonstrando de que maneira a riqueza de espécies varia em função do tipo de ilha.

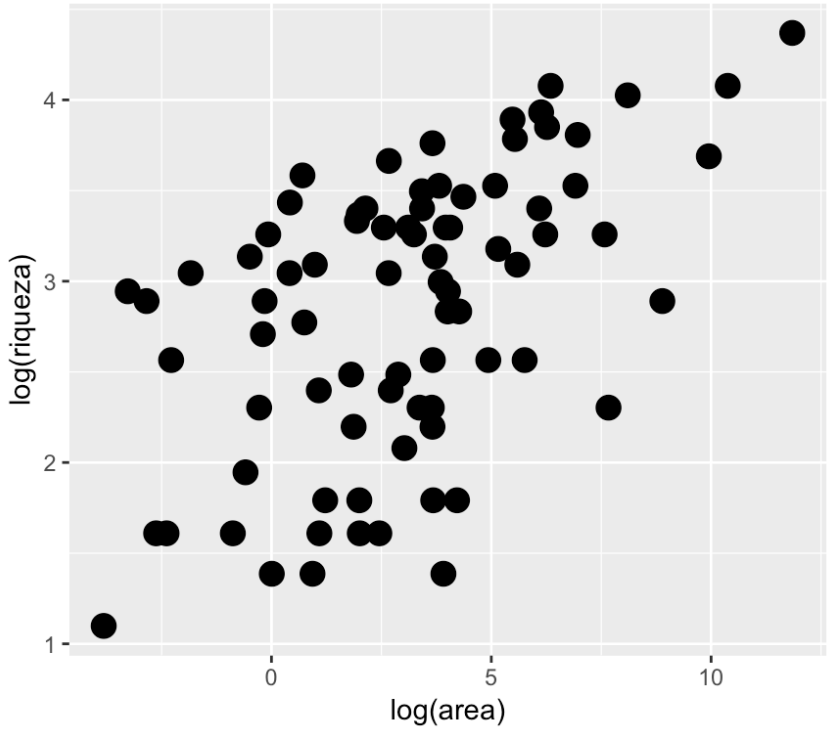
Customizando um `geom`

- Assim como nas figuras criadas na `base` e no `lattice`, também podemos modificar o tamanho (`size`), formato (`shape`), cor da borda (`color`), preenchimento (`fill`), grossura da linha dos símbolos (`stroke`) e das linhas (`size`), tipo de linhas (`linetype`) de cada `geom`.

```
ggplot(data = ilhas, mapping = aes(x = log(area), y = log(riqueza))) +
  geom_point(size = 4)
```

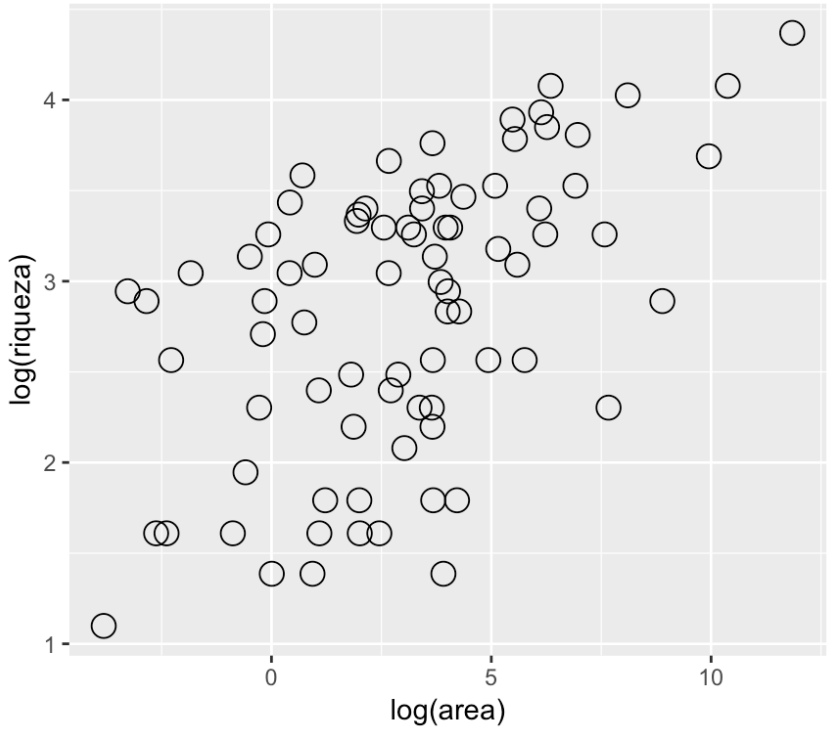
Customizando um geom

```
ggplot(data = ilhas, mapping = aes(x = log(area), y = log(riqueza))) +
  geom_point(size = 4)
```



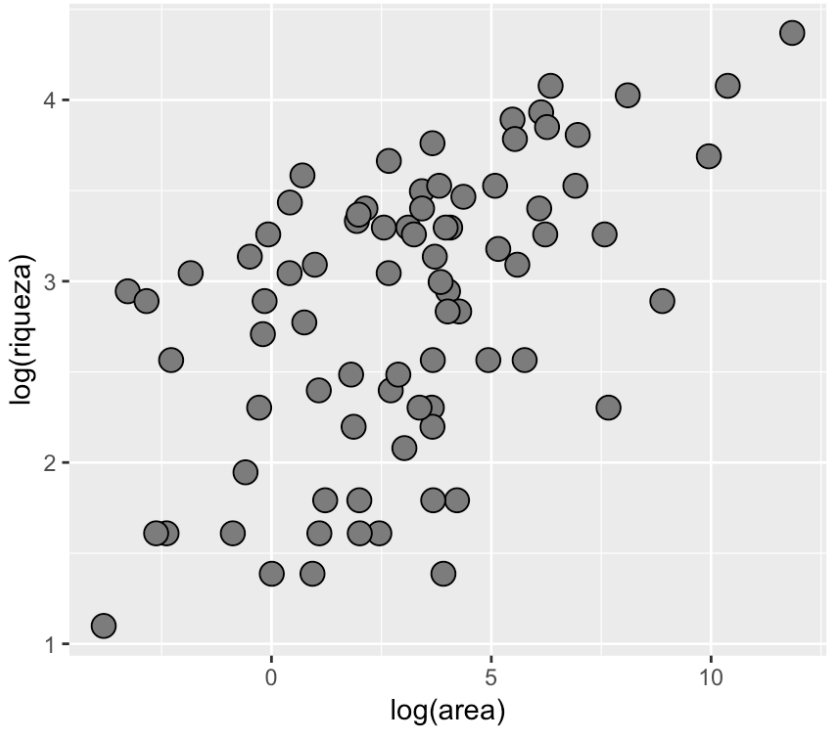
Customizando um geom

```
ggplot(data = ilhas, mapping = aes(x = log(area), y = log(riqueza))) +
  geom_point(size = 4, shape = 21)
```



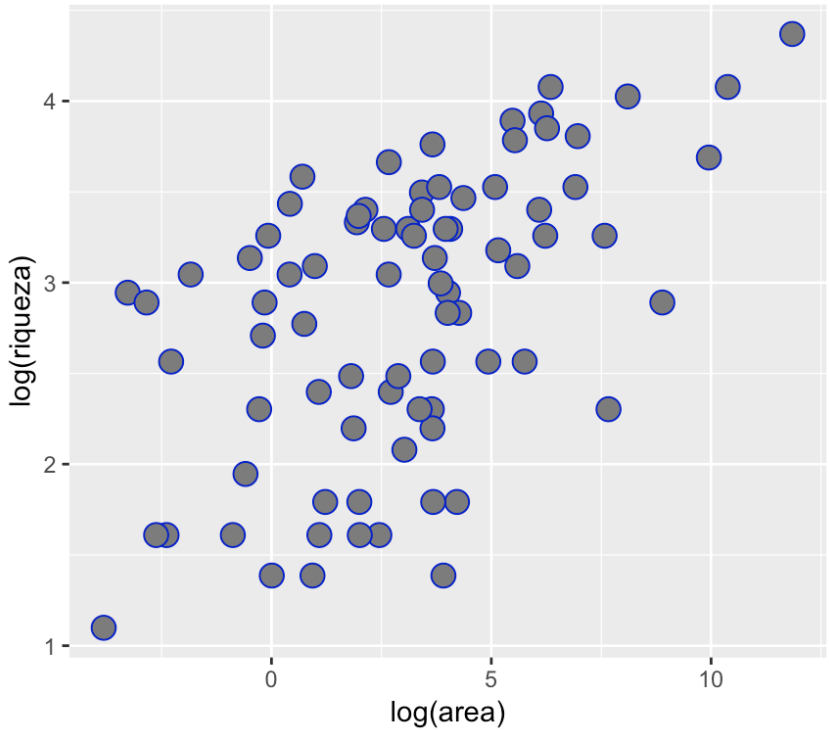
Customizando um `geom`

```
ggplot(data = ilhas, mapping = aes(x = log(area), y = log(riqueza))) +
  geom_point(size = 4, shape = 21, fill = "grey50")
```



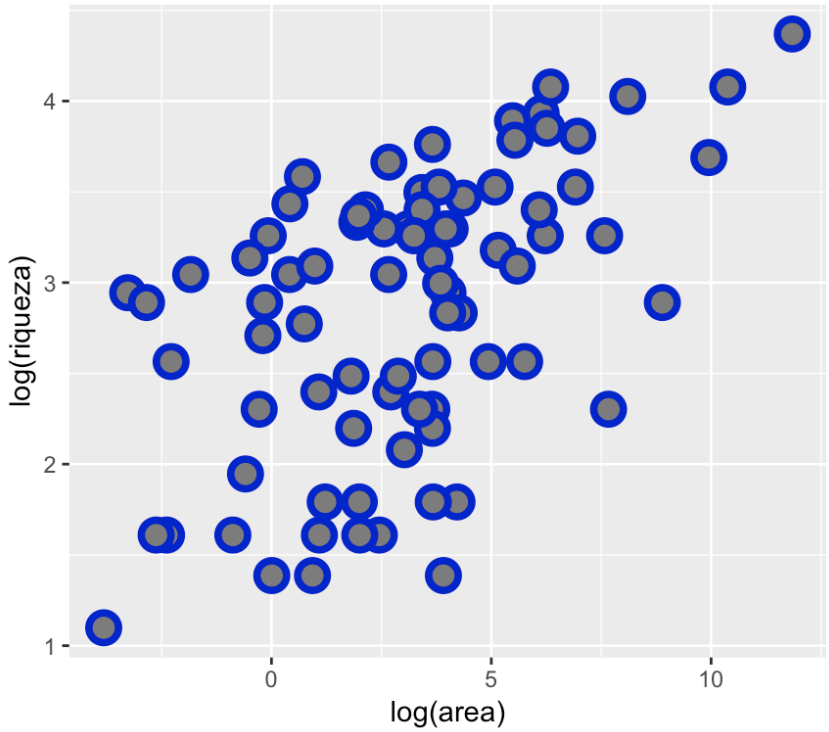
Customizando um geom

```
ggplot(data = ilhas, mapping = aes(x = log(area), y = log(riqueza))) +
  geom_point(size = 4, shape = 21, fill = "grey50",
             colour = "blue3")
```



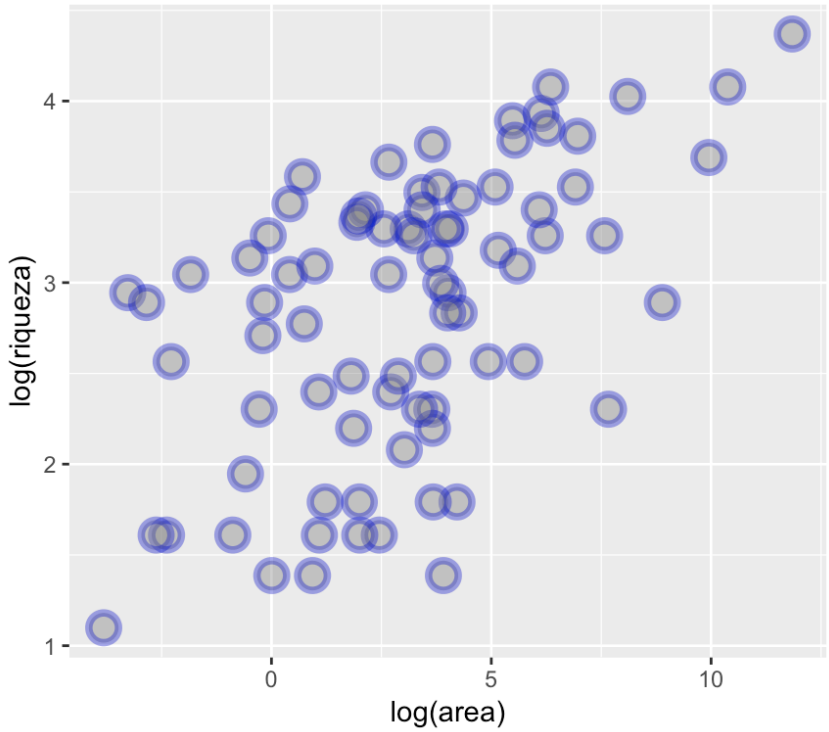
Customizando um `geom`

```
ggplot(data = ilhas, mapping = aes(x = log(area), y = log(riqueza))) +
  geom_point(size = 4, shape = 21, fill = "grey50",
             colour = "blue3", stroke = 2)
```



Customizando um `geom`

```
ggplot(data = ilhas, mapping = aes(x = log(area), y = log(riqueza))) +
  geom_point(size = 4, shape = 21, fill = "grey50",
             colour = "blue3", stroke = 2, alpha = 0.4)
```



Exercício 3

1. Customize o `boxplot` que você criou no exercício anterior. Para saber que tipo de customização pode ser feita a esse `geom`, você pode consular o arquivo de ajuda dele `?geom_boxplot`.

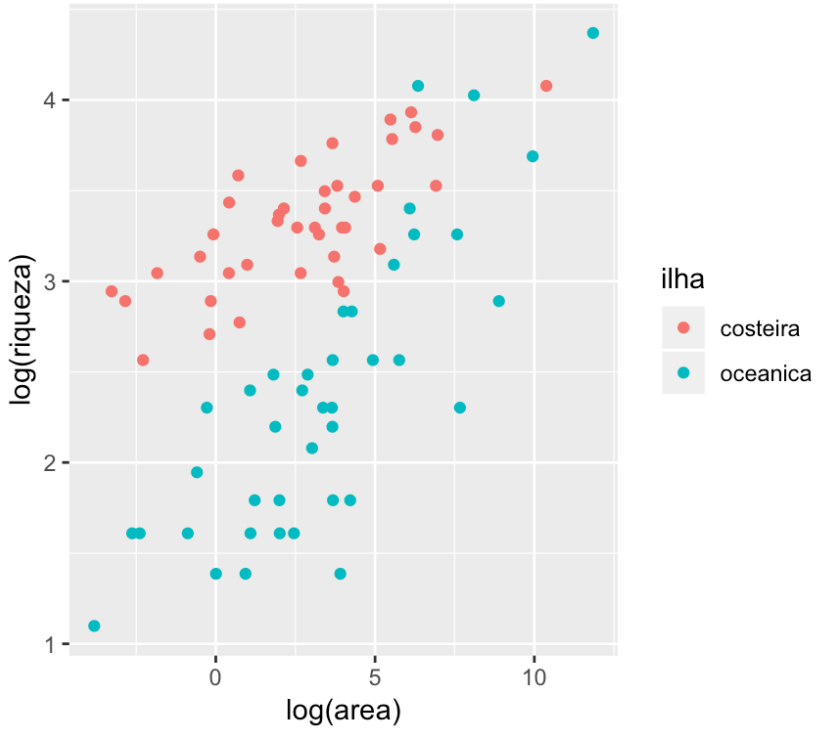
Customizando um `geom` através da unidade estética

- Quando fazemos uma análise exploratória dos dados ou até mesmo quando estamos criando uma figura para um trabalho, podemos ter como objetivo demonstrar de que forma múltiplas variáveis afetam a relação entre `x` e `y`.
- Graficamente, quando fazemos isso na realidade estamos mapeando estas outras variáveis a um dado `geom` - isto é, fazemos as características deste `geom` variar em função dos valores destas variáveis.
- Para tal, devemos dizer para o `ggplot` que todas aquelas características que acabamos de ver para a customização dependem dos valores estéticos que queremos mapear.

```
ggplot(data = ilhas, mapping = aes(x = log(area), y = log(riqueza), colour = ilha)) +
  geom_point()
```

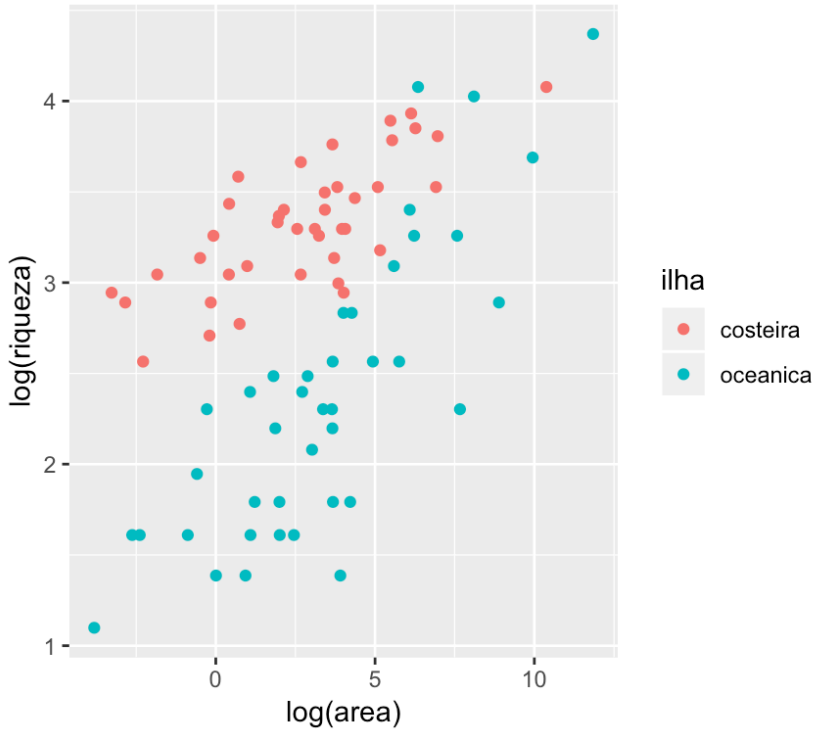
Customizando um geom através da unidade estética

```
ggplot(data = ilhas, mapping = aes(x = log(area), y = log(riqueza), colour = ilha)) +
  geom_point()
```



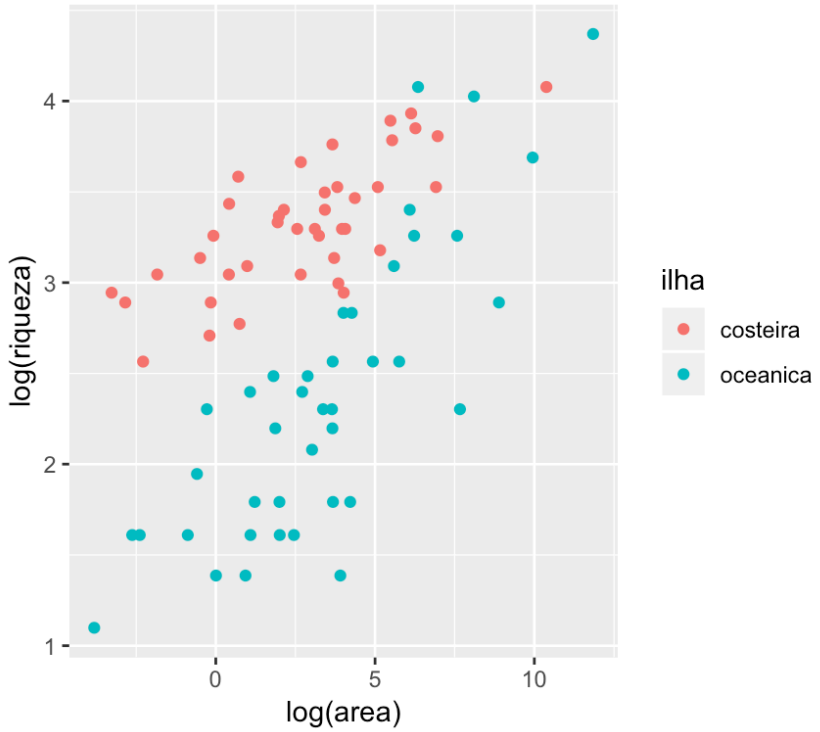
Customizando um geom através da unidade estética

```
# outra forma de fazer isso
ggplot(data = ilhas, mapping = aes(x = log(area), y = log(riqueza))) +
  geom_point(mapping = aes(colour = ilha))
```



Customizando um `geom` através da unidade estética

```
# e mais uma
ggplot() +
  geom_point(data = ilhas, mapping = aes(x = log(area), y = log(riqueza), colour = ilha))
```

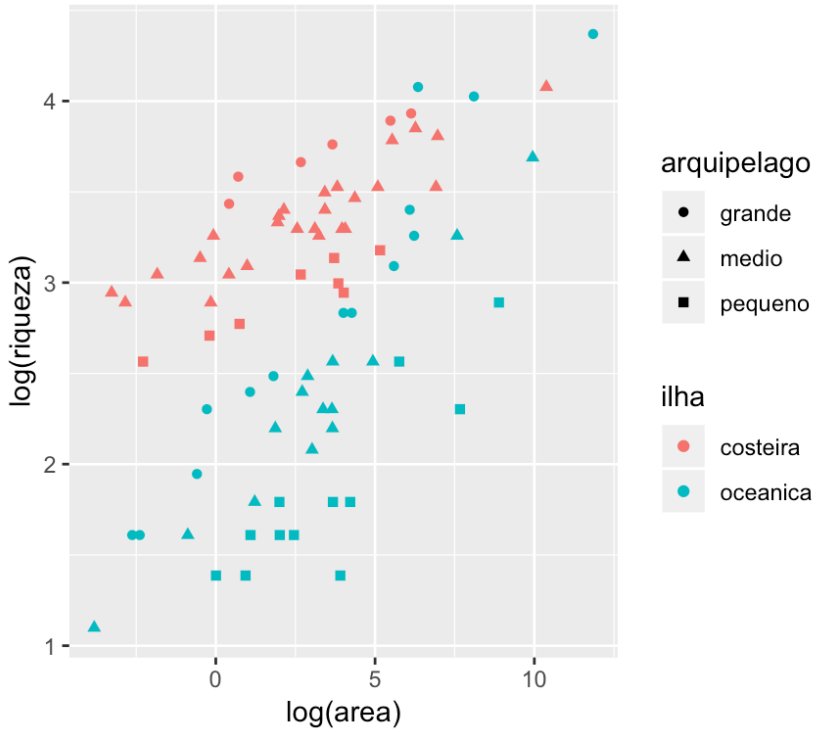


Customizando um `geom` através da unidade estética

- É importante notarmos algumas coisas ao fazermos isso:
  - A característica que será mapeada à variável deve estar dentro do `mapping`!
  - O nome da variável que será mapeada a alguma dessas características **nunca** deve estar entre aspas!
  - Ao definir que variável será mapeada a que característica logo na função `ggplot`, esta característica será mapeada da mesma forma em todas as camadas que você desenhar!
  - Podemos mapear múltiplas variáveis ao mesmo tempo.

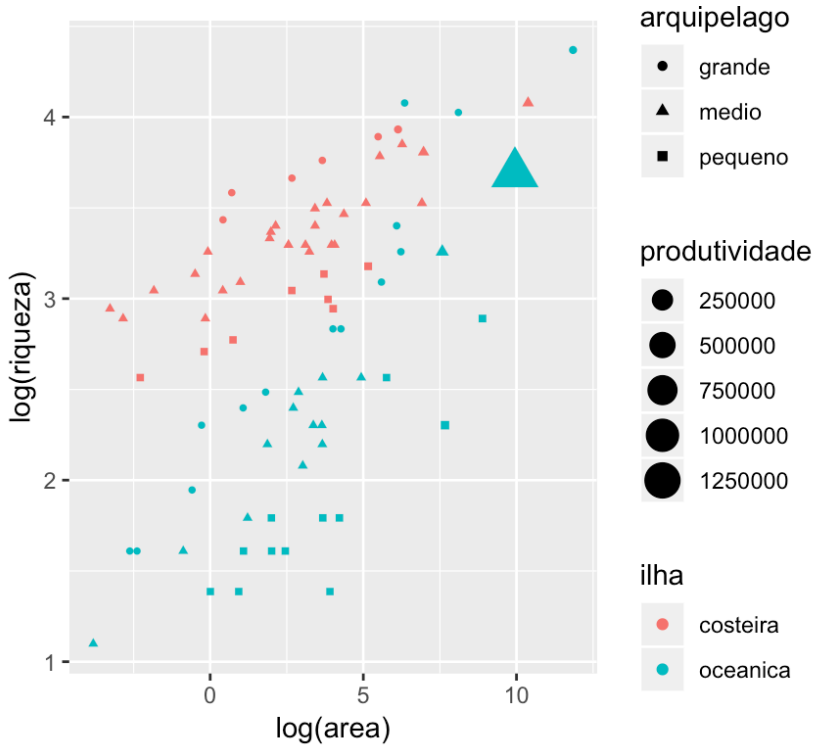
Customizando um `geom` através da unidade estética

```
ggplot(data = ilhas, mapping = aes(x = log(area), y = log(riqueza), colour = ilha,
                                   shape = arquipelago)) +
  geom_point()
```



Customizando um `geom` através da unidade estética

```
ggplot(data = ilhas, mapping = aes(x = log(area), y = log(riqueza), colour = ilha,
                                   shape = arquipelago, size = produtividade)) +
  geom_point()
```



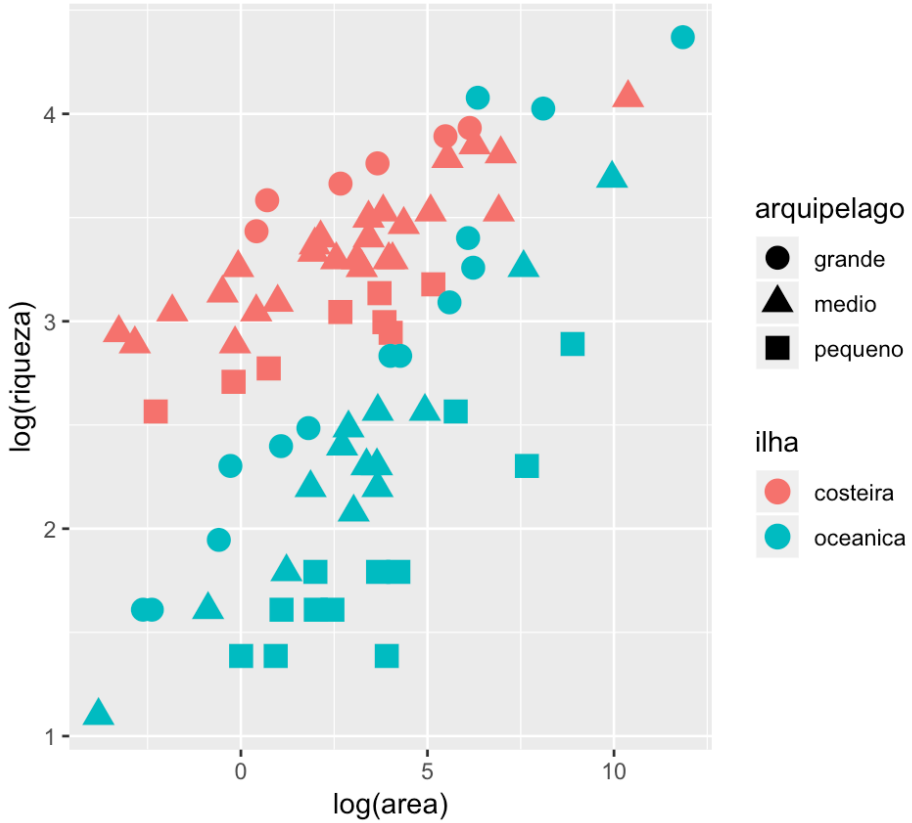
Customizando um `geom` através da unidade estética

- Também podemos combinar características mapeadas à variáveis com características mapeadas de forma fixa.
- Mas não se esqueça: toda característica mapeada a uma variável vai dentro do parenteses, toda característica fixa vai fora dele.

```
ggplot(data = ilhas, mapping = aes(x = log(area), y = log(riqueza))) +
  geom_point(mapping = aes(colour = ilha, shape = arquipelago), size = 4)
```

Customizando um `geom` através da unidade estética

```
ggplot(data = ilhas, mapping = aes(x = log(area), y = log(riqueza))) +
  geom_point(mapping = aes(colour = ilha, shape = arquipelago), size = 4)
```



Exercício 4

1. Adicione uma linha de tendência ao gráfico abaixo. O que aconteceu e o que isso representa?
2. Se você quisesse que houvesse apenas uma linha de tendência, como deveríamos desenhar a figura?
3. Modifique as características da linha de tendência que você adicionou, mapeando ela à qualquer variável que você deseje.
4. O que acontece quando invertemos a sequência como os `geom` são adicionados?

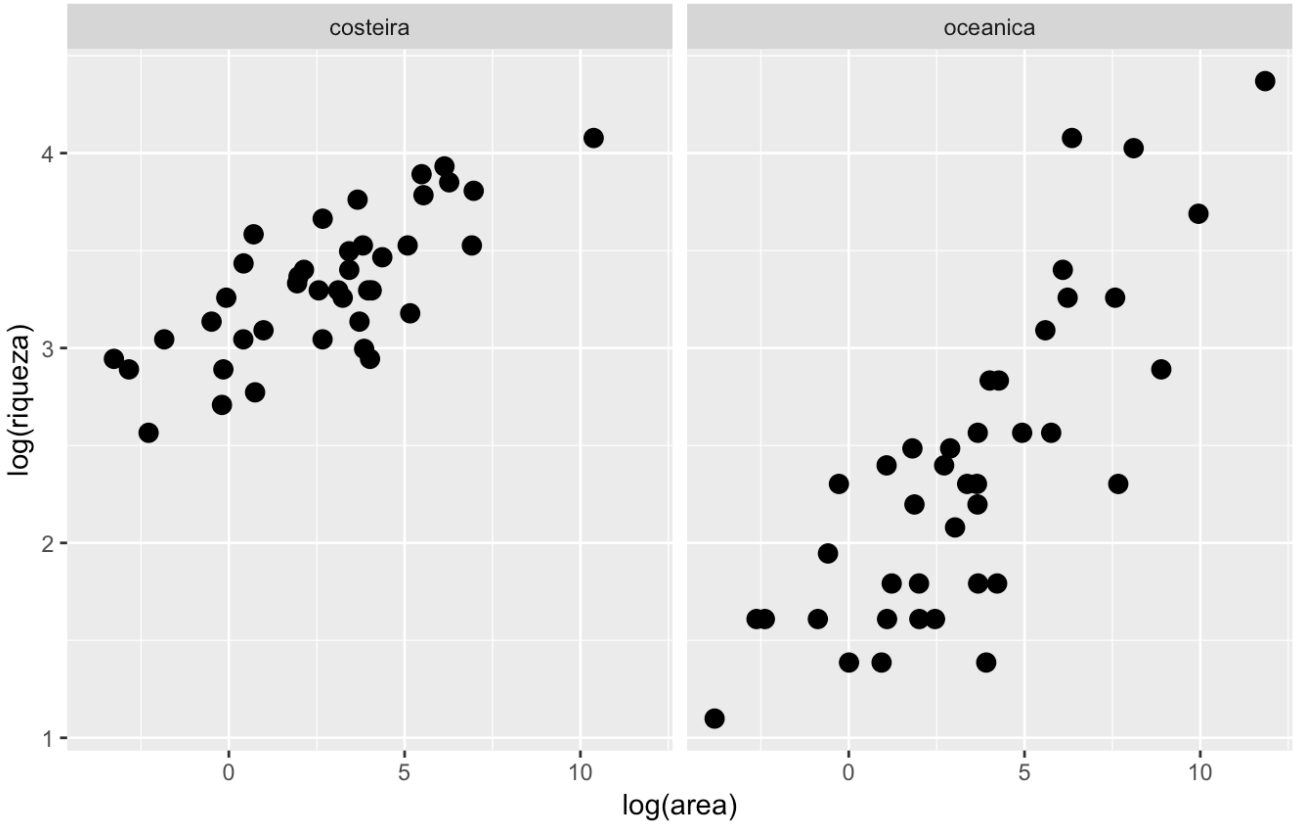
```
ggplot(data = ilhas, mapping = aes(x = log(area), y = log(riqueza), colour = ilha)) +
  geom_point(size = 3)
```

Criando múltiplos painéis a partir dos mesmo dados

- Muitas vezes queremos desenhar uma figura contendo múltiplos painéis, um para cada nível ou combinação de níveis de uma dada variável.
- O `ggplot2` oferece duas funções que permitem mapear estes painéis em múltiplos `facet`s. No entanto, note que aqui o que estamos fazendo é dividir uma mesma informação entre múltiplos painéis, e não desenhar informações diferentes em cada painel!
    - `facet_wrap`: permite que se determine de que forma os painéis serão distribuídos entre linhas e colunas. É uma função bastante flexível.
    - `facet_grid`: não permite que os painéis sejam distribuídos entre linhas e colunas, que são determinados de acordo com a quantidade de níveis de cada variável que se está mapeando aos painéis.

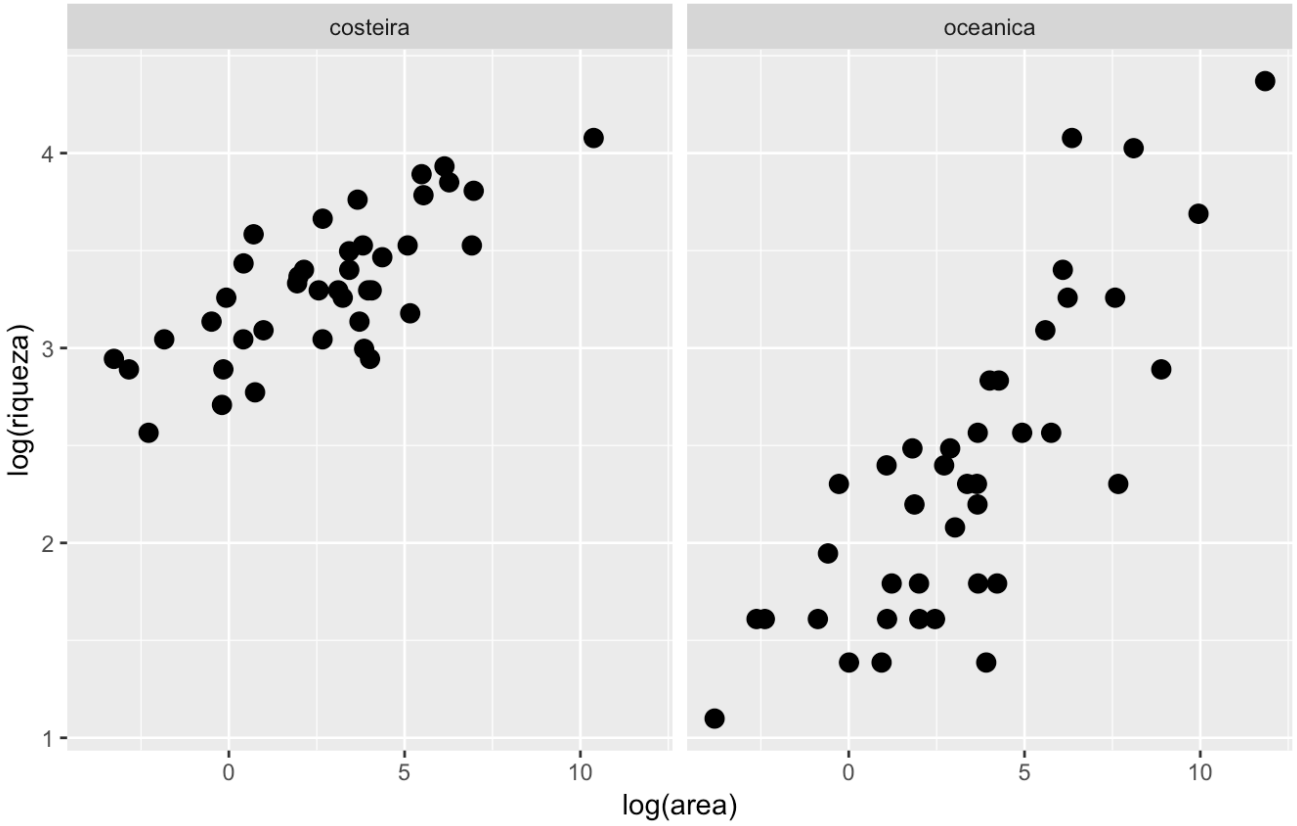
Criando múltiplos painéis a partir dos mesmo dados

```
ggplot(data = ilhas, mapping = aes(x = log(area), y = log(riqueza))) +
  facet_wrap(~ ilha) +
  geom_point(size = 3)
```



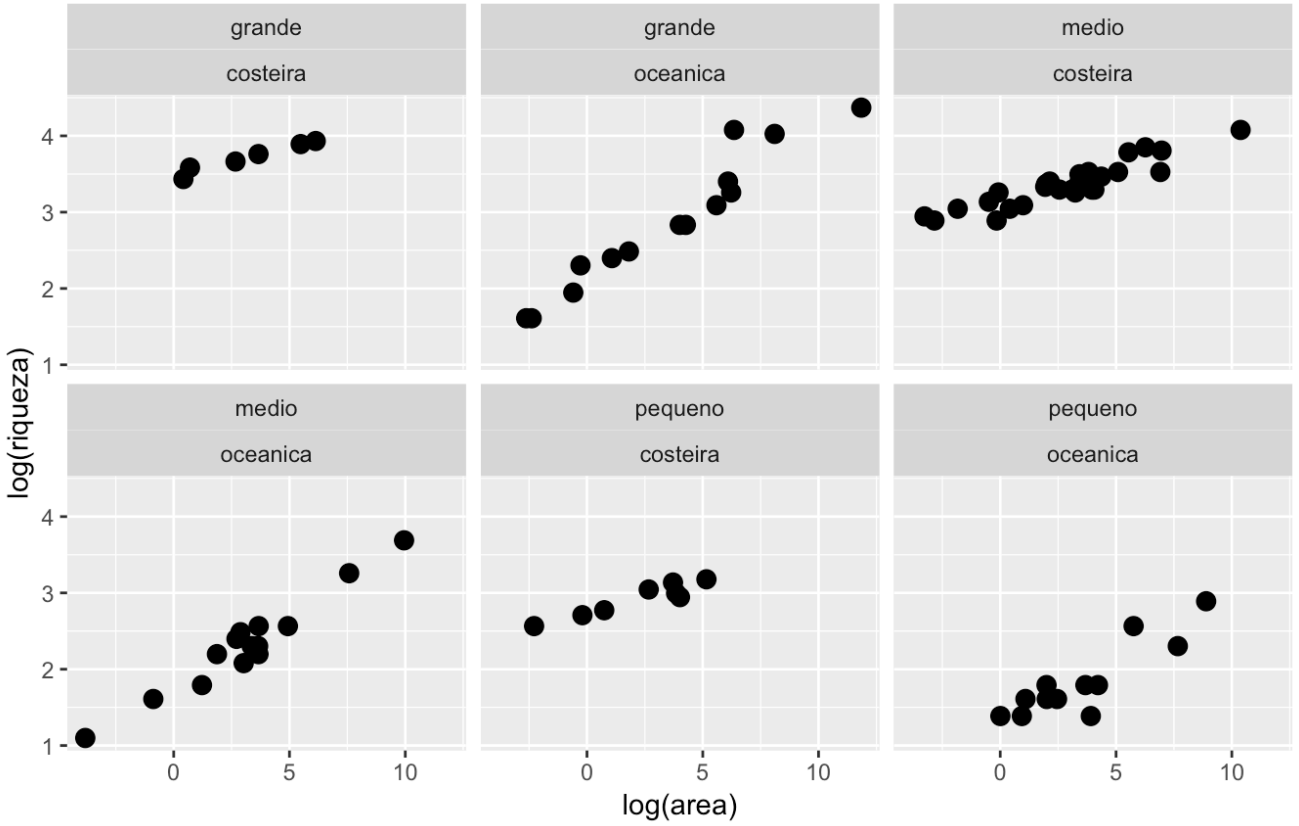
Criando múltiplos painéis a partir dos mesmo dados

```
ggplot(data = ilhas, mapping = aes(x = log(area), y = log(riqueza))) +
  facet_grid(~ ilha) +
  geom_point(size = 3)
```



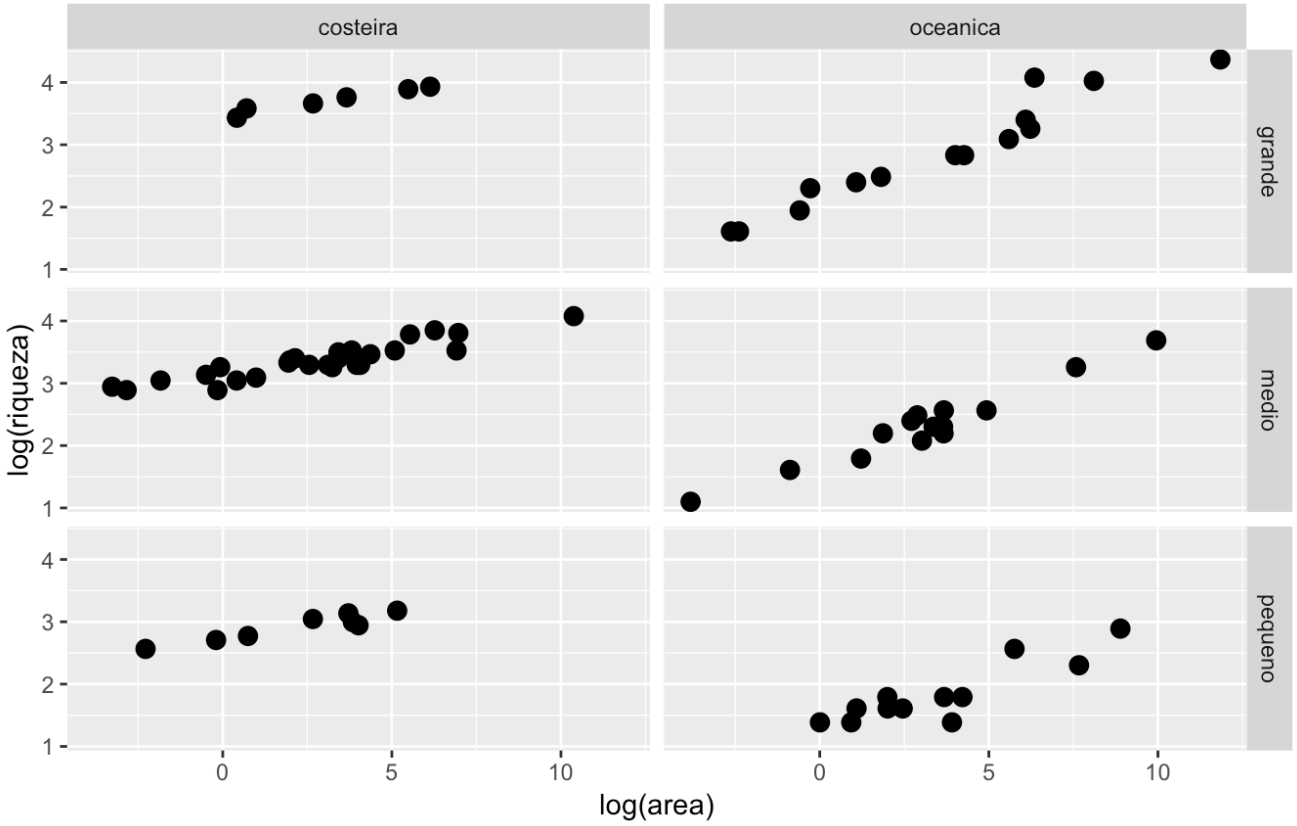
Criando múltiplos painéis a partir dos mesmo dados

```
ggplot(data = ilhas, mapping = aes(x = log(area), y = log(riqueza))) +
  facet_wrap(arquipelago ~ ilha) +
  geom_point(size = 3)
```



Criando múltiplos painéis a partir dos mesmo dados

```
ggplot(data = ilhas, mapping = aes(x = log(area), y = log(riqueza))) +
  facet_grid(arquipelago ~ ilha) +
  geom_point(size = 3)
```



Personalizando o aspecto visual do `geom`

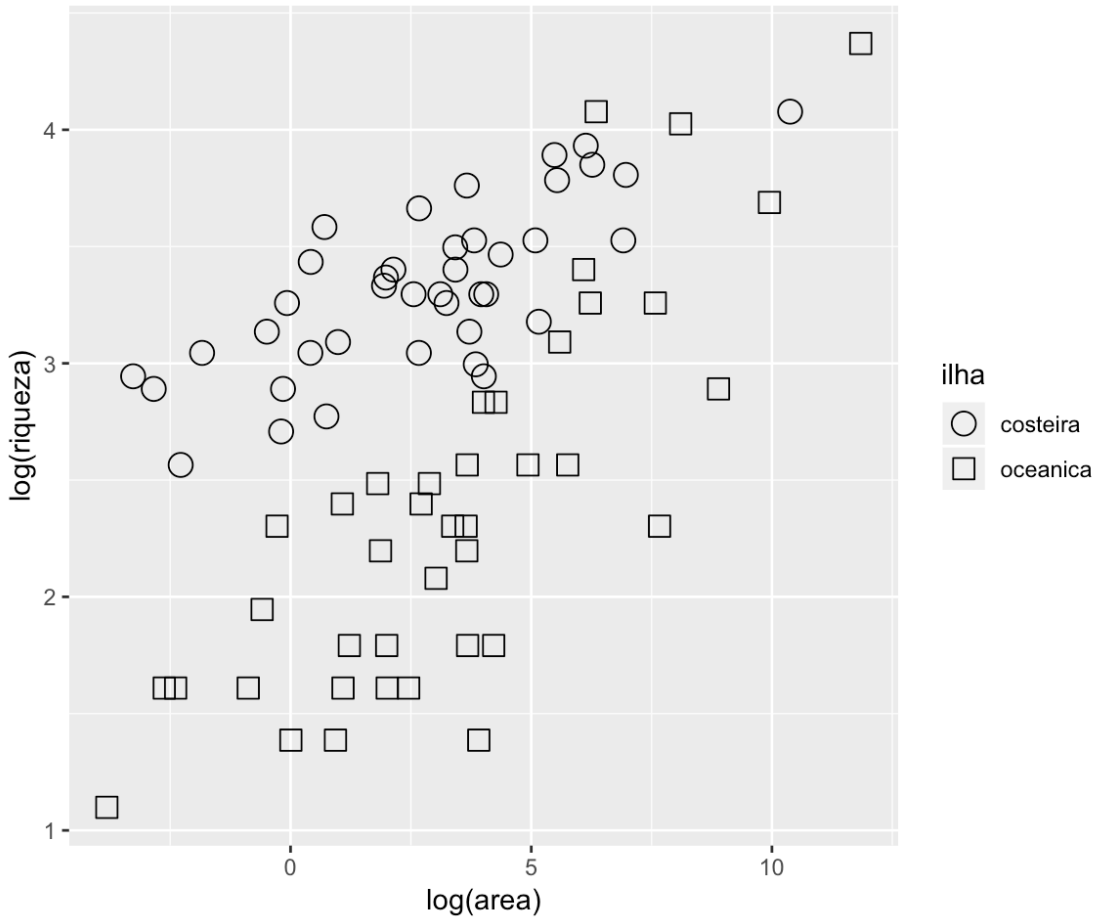
- Muito do que podemos fazer manipulando a estrutura estética do que queremos desenhar só diz o que será mapeado, mas não nos permite determinar exatamente de que forma queremos que cada característica mapeada seja representa.
- Toda a personalização do que é mapeado é feita através de camadas que descrevem a **escala** da figura:
  - `scale_fill_?`: modifica o preenchimento do objeto (`fill`) de forma determinada pelo usuário;
  - `scale_colour_?`: modifica a cor dos pontos ou das bordas do objeto geométrico (`colour`) de forma determinada pelo usuário;
  - `scale_shape_?`: modifica o formato dos pontos (`shape`) de forma determinada pelo usuário;
  - `scale_size_?`: modifica o tamanho dos pontos (`size`) de forma determinada pelo usuário;
  - `scale_alpha_?`: modifica a transparência (`alpha`) de forma determinada

Personalizando o aspecto visual do geom

- Para personalizar grande parte destas escalas é necessário que aquilo que ela modifica tenha sido mapeado à alguma unidade estética.

```
ggplot(data = ilhas, mapping = aes(x = log(area), y = log(riqueza), shape = ilha)) +
  geom_point(size = 4) +
  geom_smooth(method = "lm") +
  scale_shape_manual(values = c(21, 22))
```

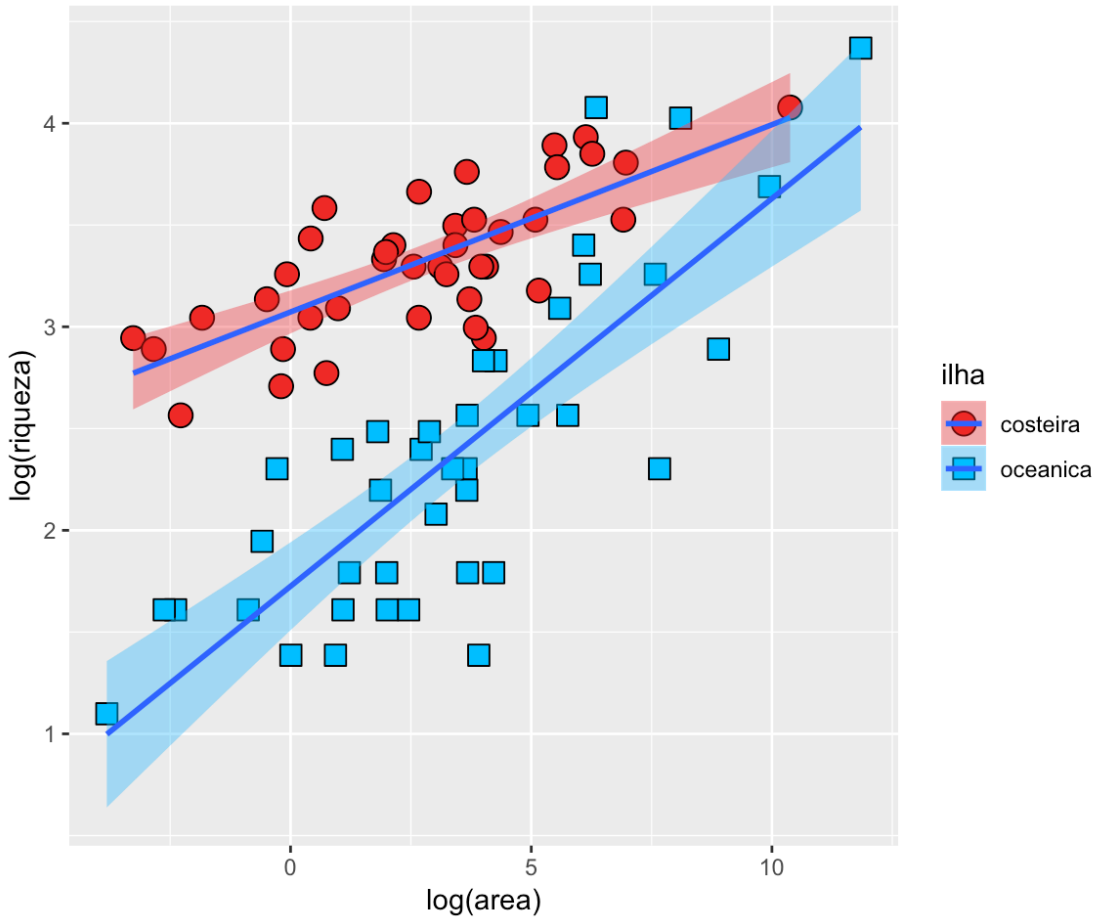
Personalizando o aspecto visual do `geom`



Personalizando o aspecto visual do geom

```
ggplot(data = ilhas, mapping = aes(x = log(area), y = log(riqueza),
                                   shape = ilha, fill = ilha)) +
  geom_point(size = 4) +
  geom_smooth(method = "lm") +
  scale_shape_manual(values = c(21, 22)) +
  scale_fill_manual(values = c("firebrick2", "deepskyblue"))
```

Personalizando o aspecto visual do `geom`



Exercício 5

1. Customize o `boxplot` que apresenta de que forma a riqueza de espécies variam em função do tipo de ilha, mas fazendo um painel para cada tamanho de arquipélago e utilizando as funções `scale`.
    - Dica: o intuito é que esta customizando leve o gráfico a ficar **apresentável** e adequado para uma publicação, tese, documento ou qualquer coisa similar. Para tanto, é essencial que ela seja visualmente agradável e auto-explicativa!

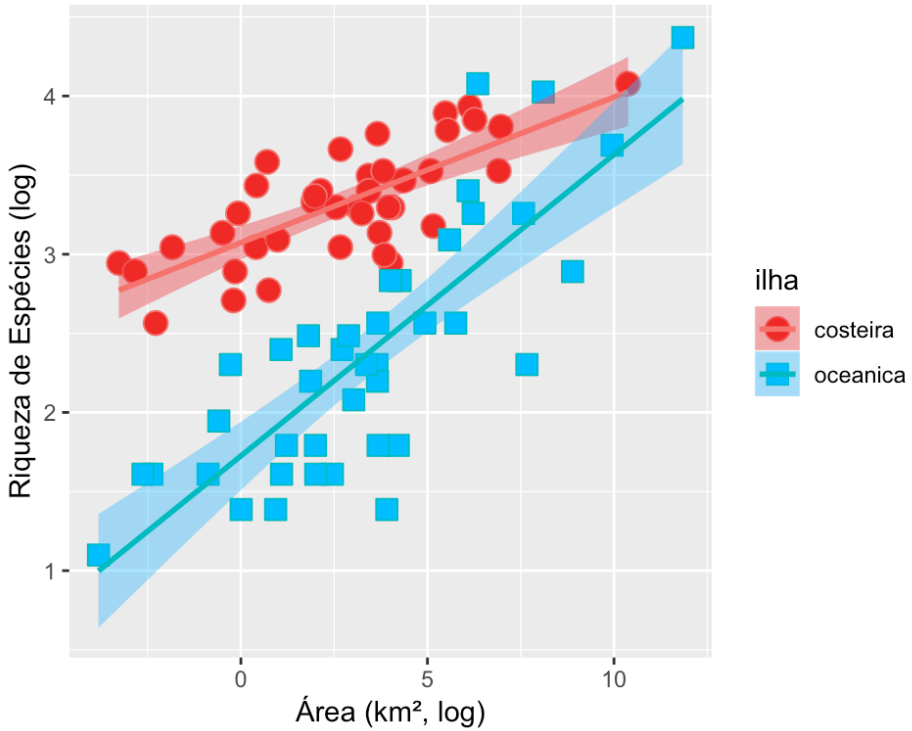
Personalizando o aspecto visual do `geom`

- Podemos utilizar as funções `scale_x_?` e `scale_y_?` para realizar diversos tipos de manipulação aos eixos `x` e `y`.

```
figura <- ggplot(data = ilhas, mapping = aes(x = log(area), y = log(riqueza),
                                             shape = ilha, fill = ilha, colour = ilha)) +
  geom_point(size = 4) +
  geom_smooth(method = "lm") +
  scale_shape_manual(values = c(21, 22)) +
  scale_fill_manual(values = c("firebrick2", "deepskyblue"))
figura
```

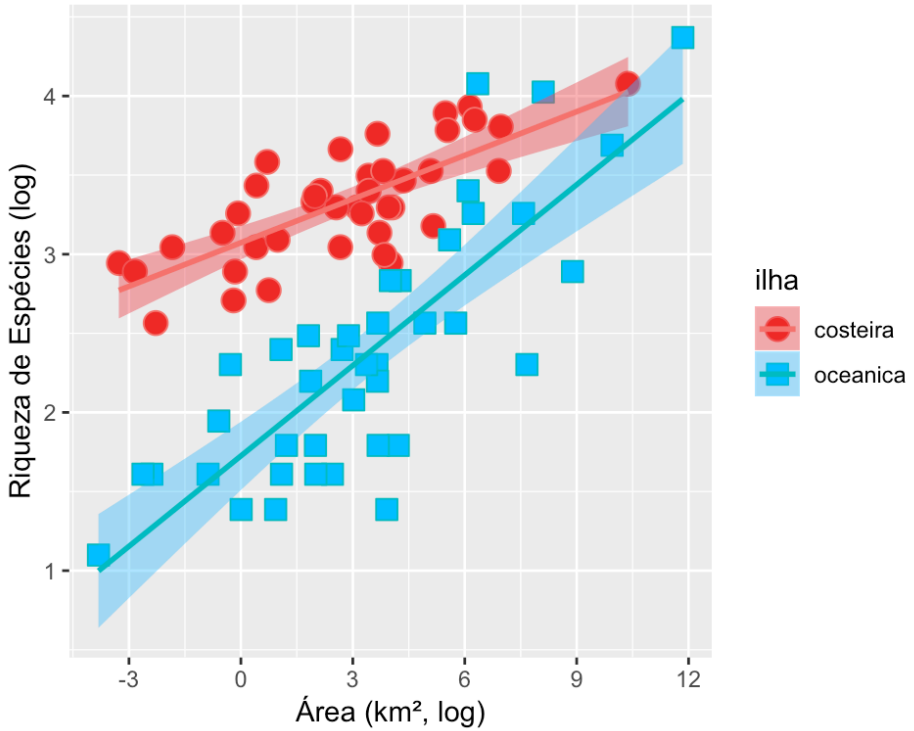
Personalizando o aspecto visual do geom

```
## adicionando um título adequado a cada eixo
figura +
  scale_x_continuous(name = "Área (km², log)") +
  scale_y_continuous(name = "Riqueza de Espécies (log)")
```



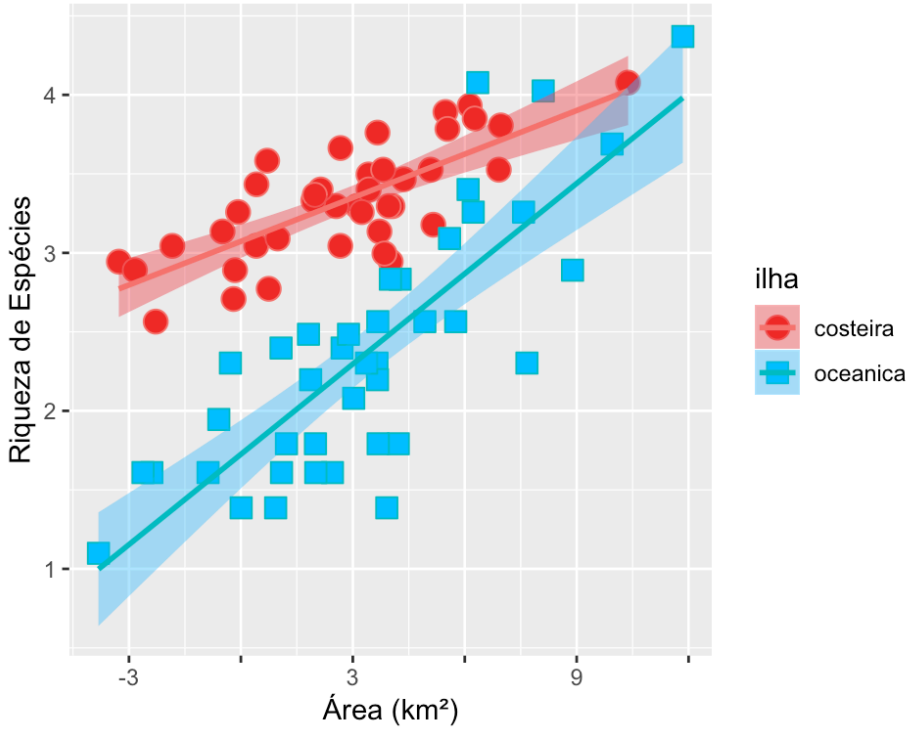
Personalizando o aspecto visual do `geom`

```
## modificar a sequência de números de um dos eixos
figura +
  scale_x_continuous(name = "Área (km², log)", breaks = seq(from = -3, to = 12, by = 3)) +
  scale_y_continuous(name = "Riqueza de Espécies (log)")
```



Personalizando o aspecto visual do geom

```
## modificar o texto dos eixos
figura +
  scale_x_continuous(name = "Área (km²)", breaks = seq(from = -3, to = 12, by = 3),
                     labels = c(-3, "", "3", "", 9, "")) +
  scale_y_continuous(name = "Riqueza de Espécies")
```



Exercício 6

1. Customize os eixos do `boxplot` que você criou no último exercício.

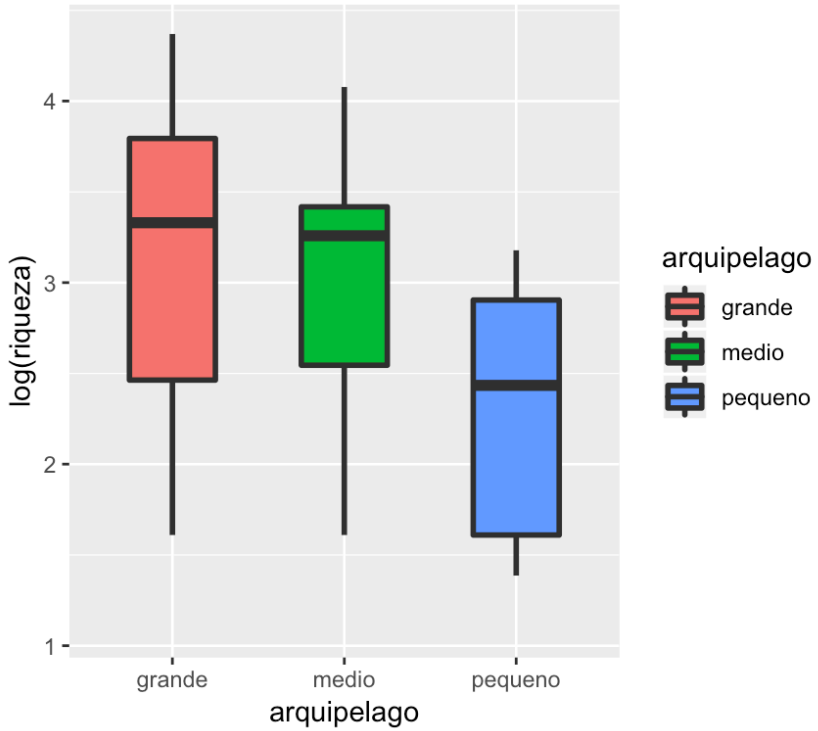
Personalizando o aspecto visual do `geom`

- Quando criamos figuras onde os valores de um dos eixos é discreto/categórico, o R vai utilizar como sequência a ordem alfabética (ou numérica) desta variáveil. No entanto, muitas vezes, a ordem alfabética não condiz com a magnitude ou efeito daquilo que queremos demonstrar e, portanto, precisamos modificar este padrão imposto pelo R.
- Existem duas formas principais de fazer isso.

```
ggplot(data = ilhas, mapping = aes(x = arquipelago,
                                   y = log(riqueza), fill = arquipelago)) +
  geom_boxplot(outlier.colour = NA, width = 0.5, size = 1)
```

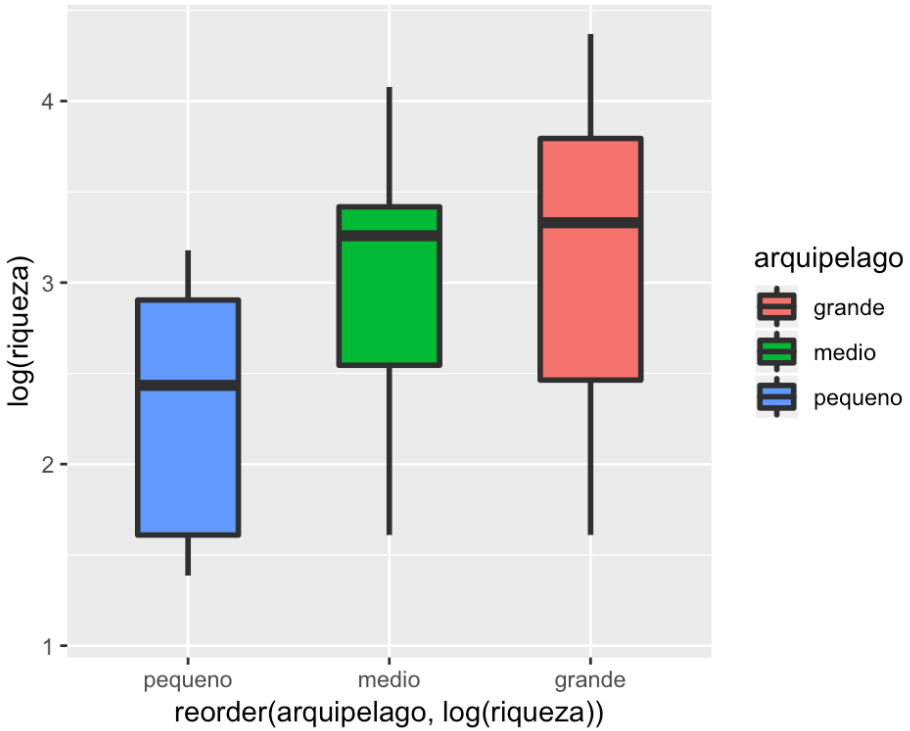
Personalizando o aspecto visual do geom

```
ggplot(data = ilhas, mapping = aes(x = arquipelago,
                                   y = log(riqueza), fill = arquipelago)) +
  geom_boxplot(outlier.colour = NA, width = 0.5, size = 1)
```



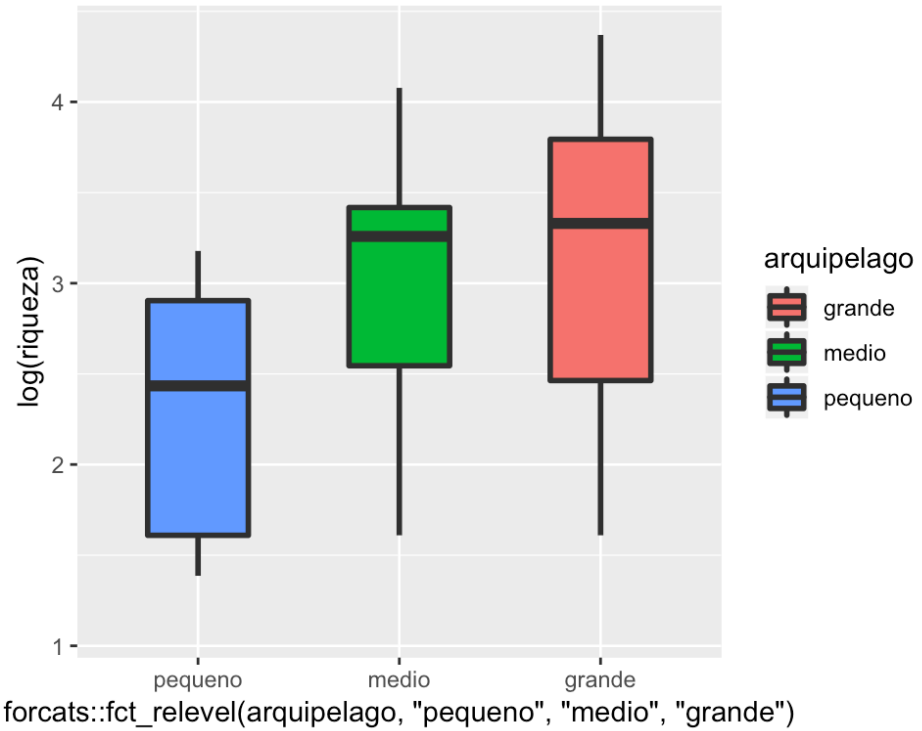
Personalizando o aspecto visual do geom

```
# através da função reorder
ggplot(data = ilhas, mapping = aes(x = reorder(arquipelago, log(riqueza)),
                                   y = log(riqueza), fill = arquipelago)) +
  geom_boxplot(outlier.colour = NA, width = 0.5, size = 1)
```



Personalizando o aspecto visual do `geom` e da figura

```
# através da função fct_relevel, disponível no pacote forcats
ggplot(data = ilhas, mapping = aes(
  x = forcats::fct_relevel(arquipelago, "pequeno", "medio", "grande"),
  y = log(riqueza), fill = arquipelago)) +
  geom_boxplot(outlier.colour = NA, width = 0.5, size = 1)
```



Uma figura-padrão, antes de prosseguirmos

```
figura <- ggplot(data = ilhas,
                 mapping = aes(x = log(area), y = log(riqueza),
                               shape = ilha, fill = ilha, colour = ilha)) +
  geom_point(size = 4) +
  geom_smooth(method = "lm") +
  scale_shape_manual(values = c(21, 22)) +
  scale_fill_manual(values = c("firebrick2", "deepskyblue")) +
  scale_x_continuous(name = "Área (km², log)", breaks = seq(from = -3, to = 12, by = 3)) +
  scale_y_continuous(name = "Riqueza de Espécies (log)")
figura
```

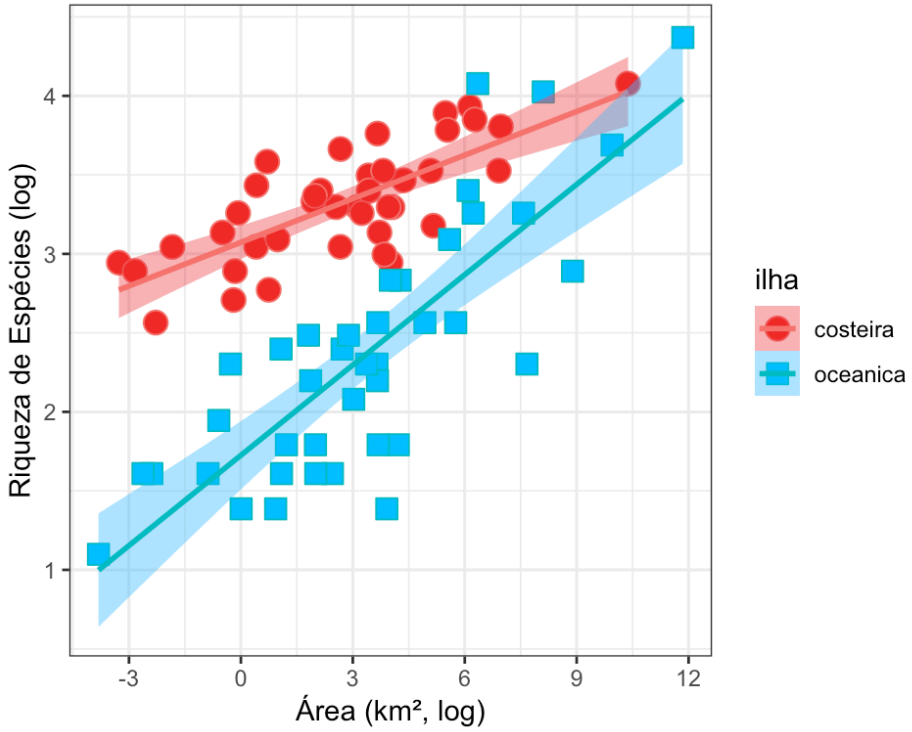
Personalizando o aspecto visual da figura

- Um importante aspecto negligenciado por grande parte das pessoas que acabam usando o `ggplot2` é esse fundo cinza e grids, que são legais para explorar visualmente os dados, mas são péssimos para uma publicação.
- Esses e outros aspectos visuais gerais de um gráfico do `ggplot2` são o **tema** dele, sendo controlados por uma função de mesmo nome - `theme`.
- **Sempre modifique e edite o tema que você usa ao criar uma figura do ggplot!**

Personalizando o aspecto visual da figura

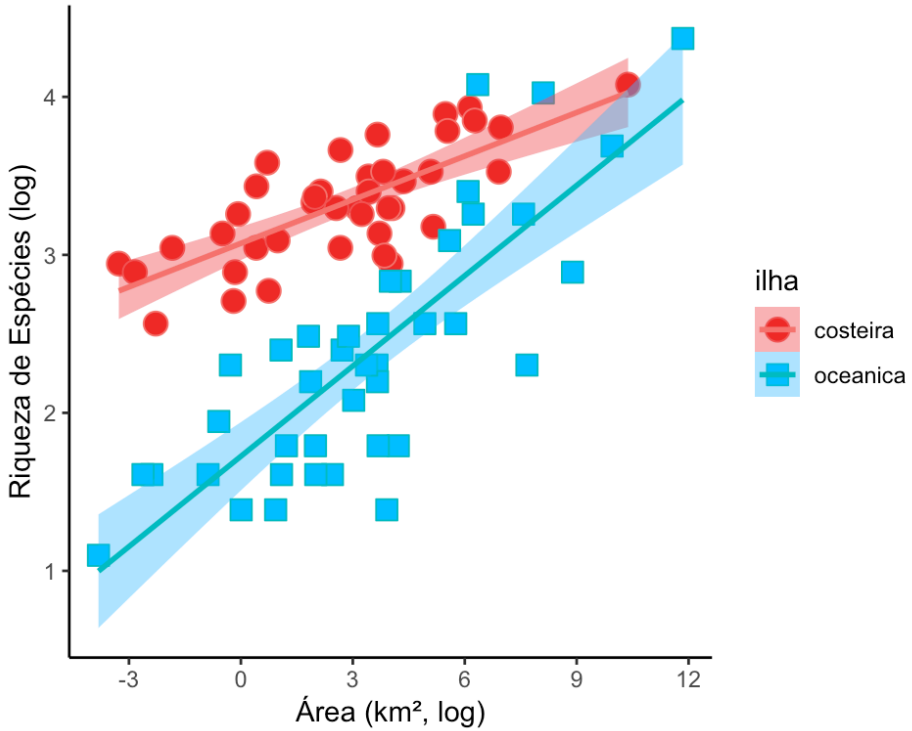
- Existem muitos temas pré-definidos disponíveis no `ggplot2` quando você carrega o pacote.

```
figura +
  theme_bw()
```



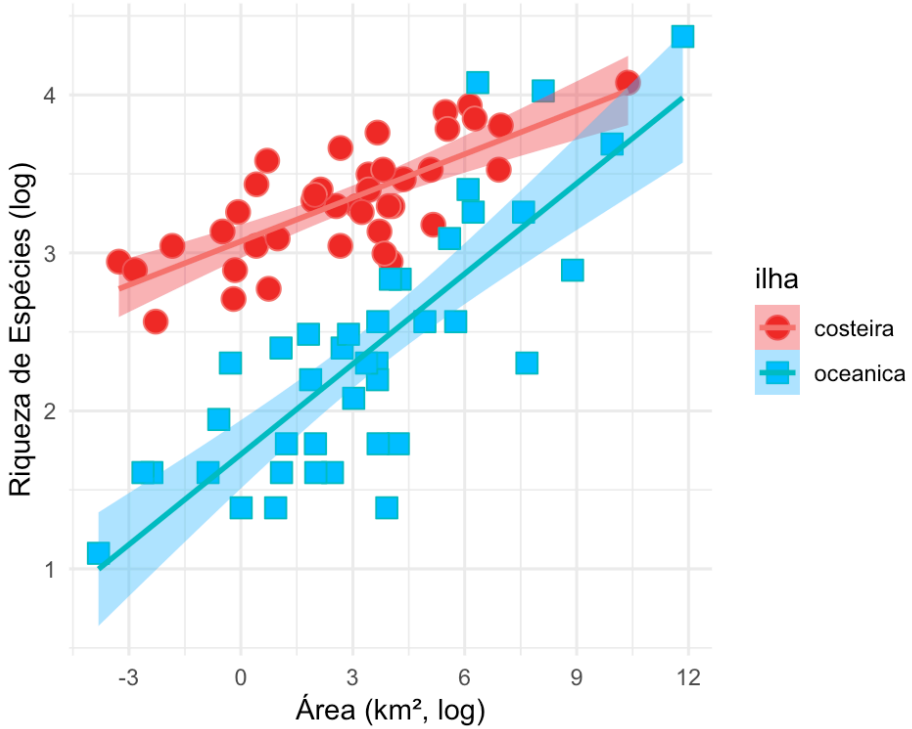
Personalizando o aspecto visual da figura

```
figura +
  theme_classic()
```



Personalizando o aspecto visual da figura

```
figura +
  theme_minimal()
```



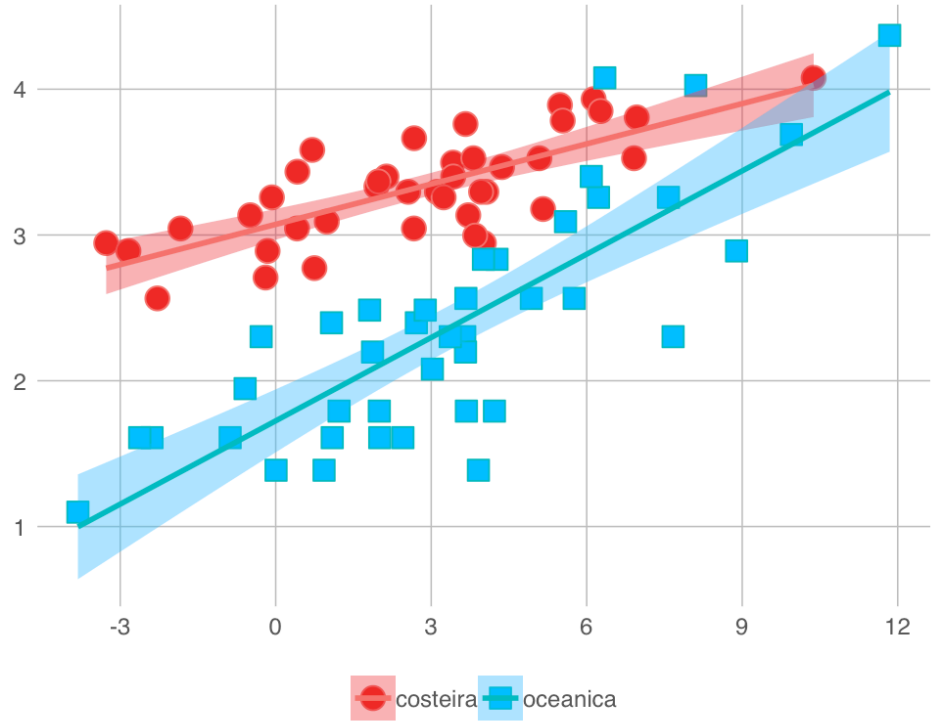
Personalizando o aspecto visual da figura

- E existem outros temas pré-definidos disponíveis em outros pacotes, como no `ggthemes`.

```
library(ggthemes)
figura +
  theme_base()
```

Personalizando o aspecto visual da figura

```
figura +
  theme_excel_new()
```



Personalizando o aspecto visual da figura

- Mas como tudo no `ggplot2`, o ideal é sempre personalizarmos aquilo que estamos desenhando. Para modificar o **tema** desta forma, precisamos modificar os padrões de valores que existe dentro da função `theme`, adicionando esta camada de personalização à nossa figura.

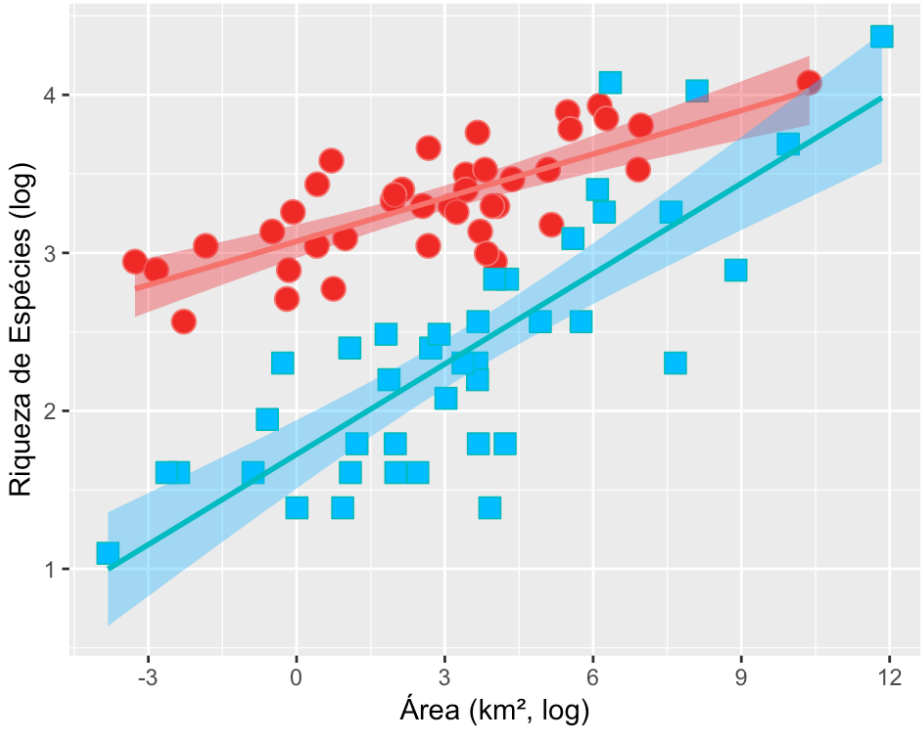
```
theme()
```

Personalizando o aspecto visual da figura

- Todos os elementos do **tema** são manipulados de acordo com a sua natureza:
  - `element_text`: elementos textuais;
  - `element_rect`: elementos que têm a forma de um retângulo;
  - `element_line`: elementos que têm a forma de linha;
  - `element_blank`: suprime uma opção.

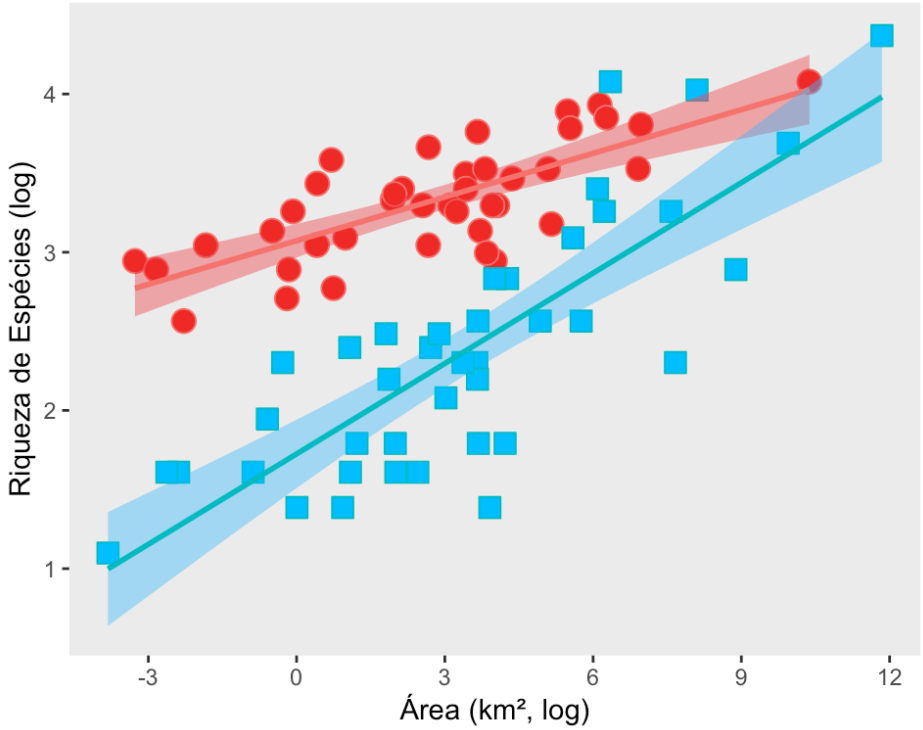
Personalizando o aspecto visual da figura

```
figura +
  theme(legend.position = "none")
```



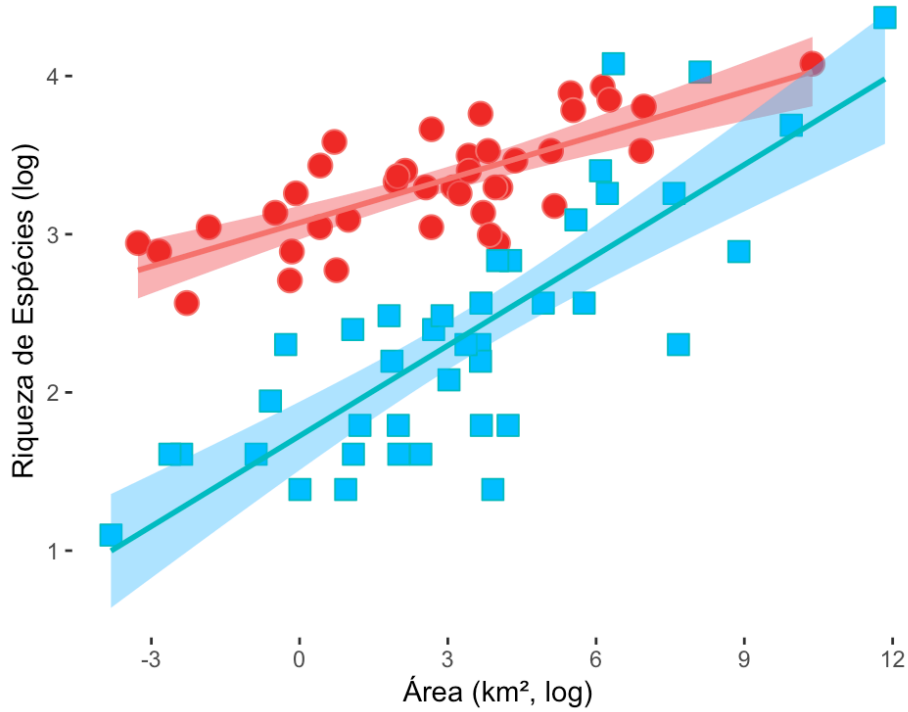
Personalizando o aspecto visual da figura

```
figura +
  theme(legend.position = "none",
        panel.grid = element_blank())
```



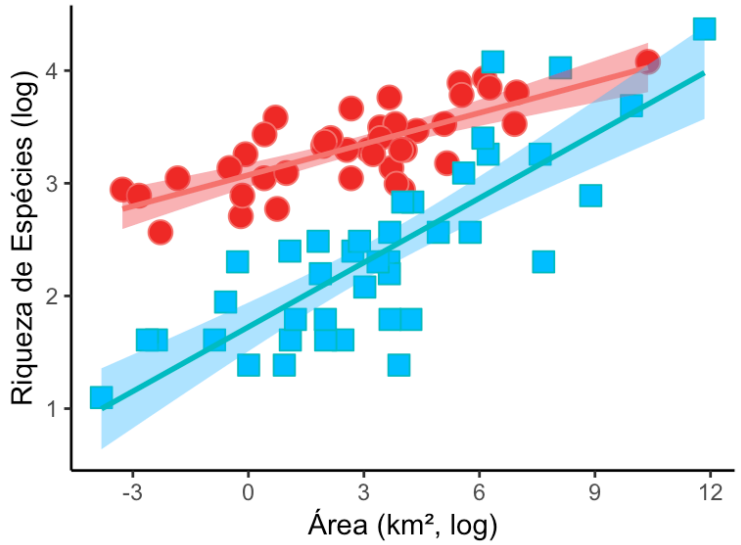
Personalizando o aspecto visual da figura

```
figura +
  theme(legend.position = "none",
        panel.grid = element_blank(),
        panel.background = element_blank())
```



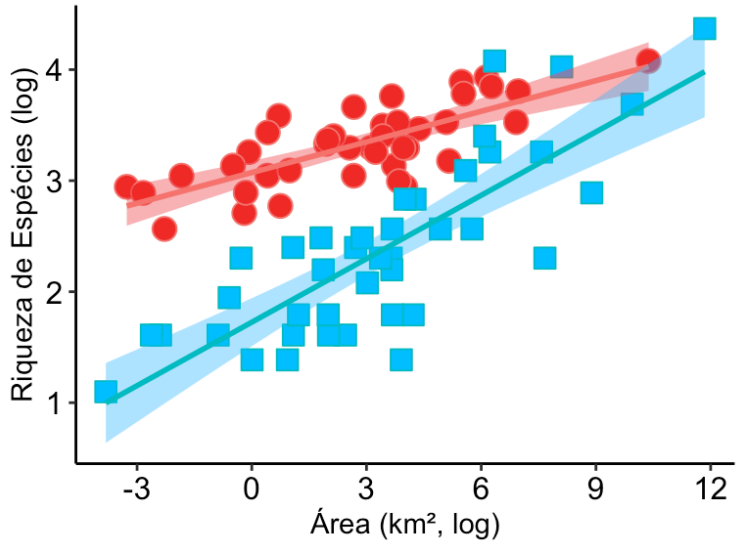
Personalizando o aspecto visual da figura

```
figura +
  theme(legend.position = "none",
        panel.grid = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black"))
```



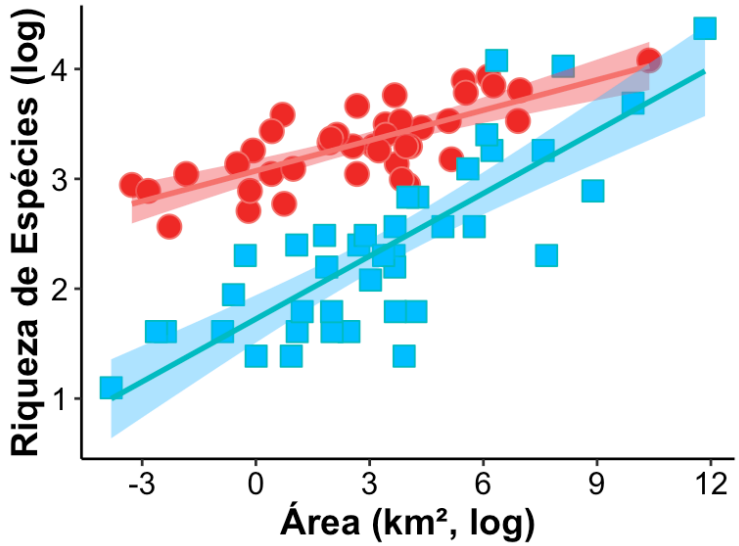
Personalizando o aspecto visual da figura

```
figura +
  theme(legend.position = "none",
        panel.grid = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black"),
        axis.text = element_text(colour = "black", size = 12))
```



Personalizando o aspecto visual da figura

```
figura +
  theme(legend.position = "none",
        panel.grid = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black"),
        axis.text = element_text(colour = "black", size = 12),
        axis.title = element_text(colour = "black", size = 14, face = "bold"))
```



Personalizando o aspecto visual da figura

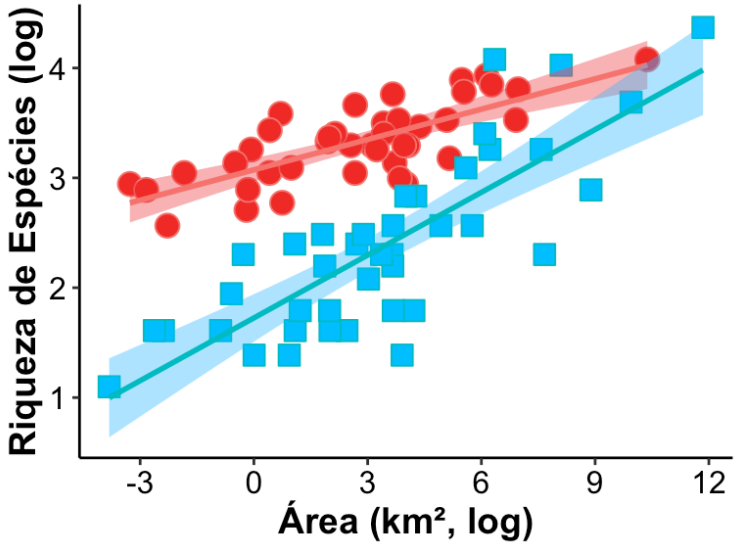
- O `ggplot2` também permite que coloquemos as especificações do tema que personalizamos em um objeto, o qual podemos adicionar diretamente à figura que criamos.

```
meu_tema <- theme(legend.position = "none",
                  panel.grid = element_blank(),
                  panel.background = element_blank(),
                  axis.line = element_line(colour = "black"),
                  axis.text = element_text(colour = "black", size = 12),
                  axis.title = element_text(colour = "black", size = 14, face = "bold"))
```

Personalizando o aspecto visual da figura

- A vantagem disso é que definimos uma única vez um tema e podemos empregá-lo em tantas figuras quanto quisermos.

```
figura <- figura +
  meu_tema
figura
```



Exportando figuras do `ggplot2`

- Podemos exportar uma figura que criamos através do `ggplot2` utilizando a função `ggsave`.

```
ggsave(filename = "especie_area.png", # nome/diretorio onde a figura será salva
       plot = figura, # objeto que contém a figura que você quer exportar
       width = 80, height = 80, units = "mm", # tamanho final da figura
       dpi = 150 # qualidade da imagem
       )
```

Exercício 7

1. Personalize o tema do `boxplot` que você tem trabalhado nos exercícios anteriores, e exporte ele para algum diretório no seu computador.

Integrando o `ggplot2` ao `tidyverse`

- Uma das figuras que normalmente queremos fazer é aquela onde apresentamos o valor da média e do erro padrão de acordo com diferentes níveis de uma variável categórica.
- Uma forma de fazer isso no `ggplot2` é fornecendo uma tabela que contenha estes valores para a função `ggplot`, e associando o `geom_bar` ao `geom_errorbar`.
- Este requerimento impõem então uma tarefa prévia para criarmos este tipo de figura a partir da tabela de dados que normalmente temos, pois precisamos calcular estes sumários estatísticos antes de criar a figura.
- Neste sentido, podemos utilizar as mesmas técnicas de manipulação, limpeza e processamento de dados que vimos nas aulas anteriores.

Exercício 8

1. Calcule o valor médio e o erro padrão da riqueza de espécies por tipo de ilha, e coloque os resultados deste processamento no objeto **sumario**.

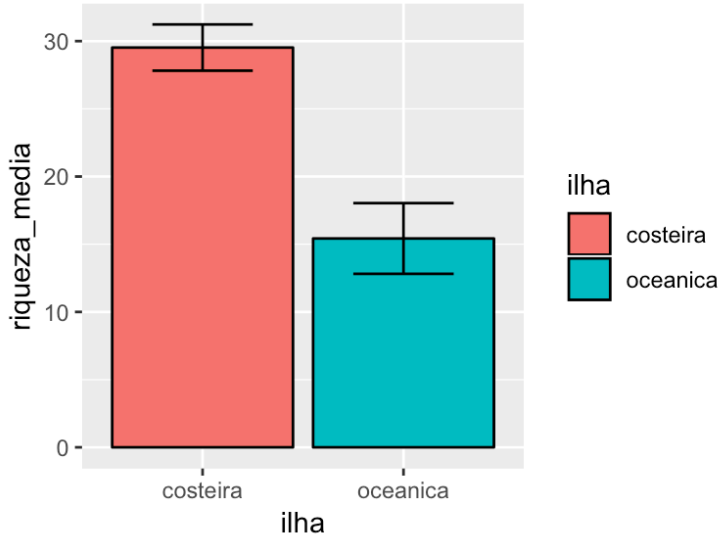
```
## Warning: package 'bindrcpp' was built under R version 3.4.4
```

```
## # A tibble: 2 x 3
##   ilha     riqueza_media  erro
##   <chr>            <dbl> <dbl>
## 1 costeira          29.5  1.71
## 2 oceanica          15.4  2.61
```

Integrando o `ggplot2` ao `tidyverse`

- Agora, com essa tabela de dados, podemos criar a figura que queremos.

```
ggplot(data = sumario, mapping = aes(x = ilha, y = riqueza_media, fill = ilha)) +
  geom_bar(stat = "identity", colour = "black") +
  geom_errorbar(mapping = aes(ymin = riqueza_media - erro, ymax = riqueza_media + erro),
                width = 0.5)
```



Exercício 9

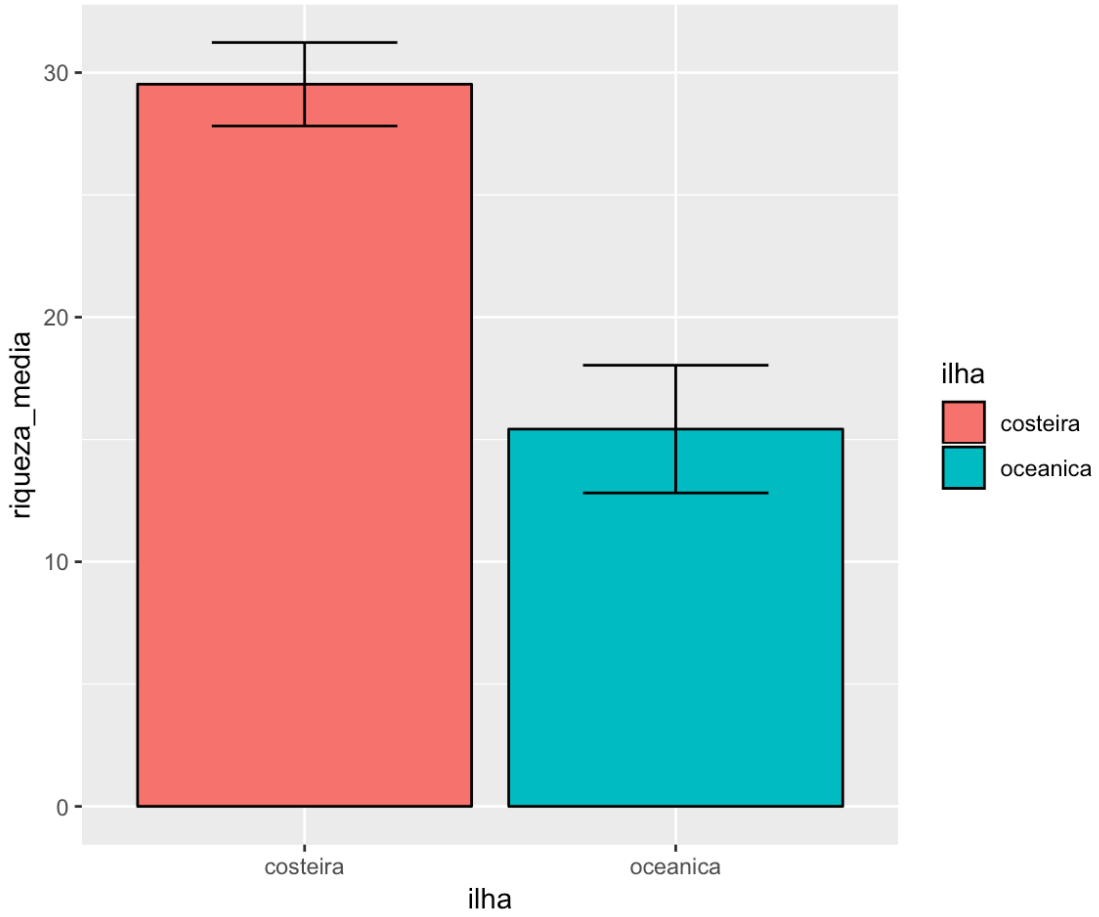
- Personalize a figura anterior:
  - Modificando a cor da barra que cada categoria recebe;
  - Editando os nomes e os valores dos eixos `x` e `y`, respectivamente;
  - Editando o título dos eixos `x` e `y`;
  - Removendo o espaço que aparece abaixo do 0 no eixo `y`.

Integrando o `ggplot2` ao `tidyverse`

- Note, no entanto, que o `ggplot2` é parte integrante do `tidyverse`, que foi todo pensado com a finalidade de encurtar a distância entre a ideia e o resultado.
- Portanto, podemos integrar todo o cálculo prévio da média e erro ao uso da função `ggplot`.

```
ilhas %>%
  group_by(ilha) %>%
  summarise(riqueza_media = mean(riqueza, na.rm = TRUE),
            erro = sd(riqueza, na.rm = TRUE)/sqrt(n())) %>%
  ggplot(data = sumario, mapping = aes(x = ilha, y = riqueza_media, fill = ilha)) +
  geom_bar(stat = "identity", colour = "black") +
  geom_errorbar(mapping = aes(ymin = riqueza_media - erro, ymax = riqueza_media + erro),
                width = 0.5)
```

Integrando o `ggplot2` ao `tidyverse`



Uma mensagem final

- Utilize o `ggplot2` como uma ferramenta para a análise exploratória e visualização dos dados, mas não utilize diretamente o resultado daquilo que você produz através dele nas suas publicações.
- O principal motivo para isso é que muito do que queremos mostrar visualmente em uma publicação são os resultados das análises estatísticas, que descrevem a relação entre as variáveis considerando toda a dependência numérica e outras peculiaridades que existem nos seus dados.
- Nesse sentido, produza os resultados numéricos das suas análises estatísticas e utilize eles para criar as figuras no `ggplot2`. Veremos como fazer isso na próxima aula.

Exercício 10

- Utilizando o conjunto de dados **gapminder**:
  - Calcule o GDP per capita médio por continente considerando apenas os dados do ano de 2007.
  - Crie uma figura demonstrando como variou o tamanho da população brasileira ao longo do tempo.
  - Crie uma figura demonstrando de que forma a expectativa de vida varia ao longo do tempo para cada país, mas dividindo os padrões em diferentes painéis de acordo com o continente.

Mais sobre o `ggplot2`

- No livro *ggplot2*: http://ggplot2.org/book/;
- No site: http://ggplot2.tidyverse.org/reference/;
- No livro *"R for Data Science"*: http://r4ds.had.co.nz/data-visualisation.html;
- O Google.