Nesta aula, veremos de forma mais detalhada os elementos *data*, *aesthetics*, *stats* e *facetting* e o que significa cada um desses elementos na estrutura da gramática de gráficos. Vamos utilizar dois *datasets*: o de produção madeireira disponibilizado pelo IBAMA e um novo, contendo uma série histórica modificada do censo escolar agregado do Instituto de Pesquisas Educacionais Anísio Teixeira (INEP). Os dados contém o número de estudantes por série de ensino, escola, munícipio e outras variáveis para o período compreendido entre 2007 e 2019, apenas para os estados do Espírito Santo, Mato Grosso do Sul e Roraima.

Como o arquivo do Censo Escolar está no formato .csv pt-br (separado por ;), podemos utilizar a função read\_csv2() do pacote readr.

library(readr) *# para ler os dados*

library(magrittr) *# vamos usar pipe %>%*

library(ggplot2) *# para fazer os gráficos*

*# leitura*

tb\_censo\_escolar <- read\_csv2(file="https://raw.githubusercontent.com/allanvc/book\_IADR-T/master/datasets/tab\_censo\_sample\_ES\_MS\_RR.csv")

arquivo do ibama

library(dplyr)

tb\_ibama <- read\_csv(file="https://raw.githubusercontent.com/allanvc/book\_IADR-T/master/datasets/PA%20GF%202017%20jan-jun\_editada.csv",

col\_types = cols(

X1 = col\_double(),

TIPO\_GF = col\_character(),

STATUS\_GF = col\_character(),

UF\_REMETENTE = col\_character(),

MUNICÍPIO\_REMETENTE = col\_character(),

TIPO\_DESTINO = col\_character(),

CEPROF\_DESTINATÁRIO = col\_character(),

UF\_DESTINATÁRIO = col\_character(),

MUNICÍPIO\_DESTINATÁRIO = col\_character(),

N\_AUTORIZAÇÃO = col\_character(),

PROCESSO = col\_character(),

EMISSAO = col\_integer(),

NOME\_CIENTÍFICO = col\_character(),

PRODUTO = col\_character(),

VOLUME = col\_double(),

UNID = col\_character(),

PRECO\_TOTAL = col\_double()

)

)

tb\_ibama$STATUS\_GF[1:50000] <- rep("NÃO VERIFICADO", 50000)

tb\_ibama2 <- mutate(tb\_ibama,

preco\_unidade = PRECO\_TOTAL / VOLUME,

preco\_unidade\_vezes\_1000 = preco\_unidade \* 1000

)

**Data (Dados)**

Os dados utilizados com as funções de ggplot2 **DEVEM** ser armazenados como um tibble ou dataframe. Há possibilidade, inclusive, de se usar mais de um dataframe em um gráfico. A recomendação é que se defina e se certifique acerca dos tipos das variáveis (*numeric*, *factor*, etc) antes de começar a construir um gráfico

**A função ggplot()**

Os dados que serão usados para plotagem serão especificados dentro da função ggplot(). Note que é **ggplot()** e não **ggplot2()**. O nosso data.frame ou tibble será sempre o primeiro argumento desta função.

Um outro argumento a ser passado para a função ggplot() são os *aesthetics*, utilizando a função auxiliar aes() que mapeia as variáveis dos dados para os elementos gráficos perceptíveis (estéticos), como por exemplo a posição nos eixos dos gráficos, a definição de cores por categorias, etc.

Qualquer função subsequente vai herdar os dados e os *aesthetics* de ggplot(), a não ser que estes parâmetros sejam sobrescritos em outra camada. Para encadear outras funções à função ggplot(), usamos sempre +.

**Aesthetics (elementos estéticos dos gráficos)**

Variáveis são mapeadas para os **aesthetics**, o que significa que são traduzidas para os elementos estéticos do gráfico. A função aes(), como já vimos, é quem faz esse mapeamento. Ela também pode ser especificada dentro de outras camadas que compõem o gráfico, como *geoms* ou *stats*, por exemplo . Os **aesthetics** mais utilizados são:

* x - posição no eixo x;
* y - posição no eixo y;
* color ou colour - “cor de fora” ou do contorno do objeto;
* fill - “cor de dentro” do objeto;
* alpha - nível transparência do gráfico;
* shape - forma dos marcadores (pontos, triângilos, cruz, etc);
* size - tamanho (do raio) dos objetos.

**Exemplos para *aesthetics***

Para fazermos um gráfico, primeiro precisamos indicar os dados e fazer o mapeamento das variáveis aos **aesthetics**. Isso reserva uma área de gráfico a ser utilizada.

options(scipen= 999) *# para evitar a exibição em notação exponencial*

*# ggplot(data=tb\_ibama2, mapping = aes(x=VOLUME, y=PRECO\_TOTAL))*

*# OU*

tb\_ibama2 %>%

ggplot(data=tb\_ibama2, mapping = aes(x=VOLUME, y=PRECO\_TOTAL))

Gráfico, Histograma

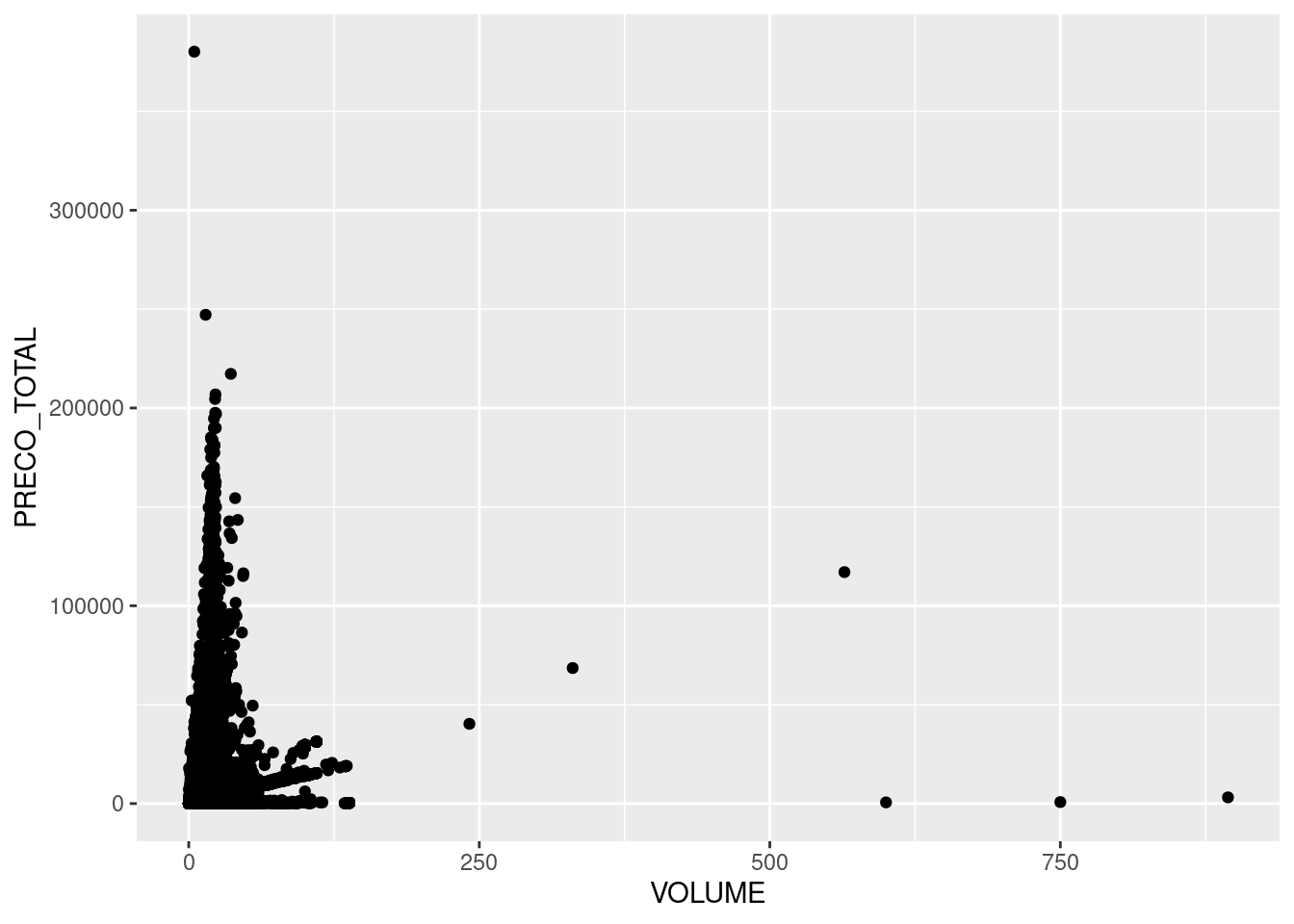
Descrição gerada automaticamente

Vejamos um primeiro exemplo para o cruzamento entre *VOLUME* vs *PRECO\_TOTAL*.

tb\_ibama2 %>%

ggplot(mapping = aes(x=VOLUME, y=PRECO\_TOTAL))+

geom\_point()



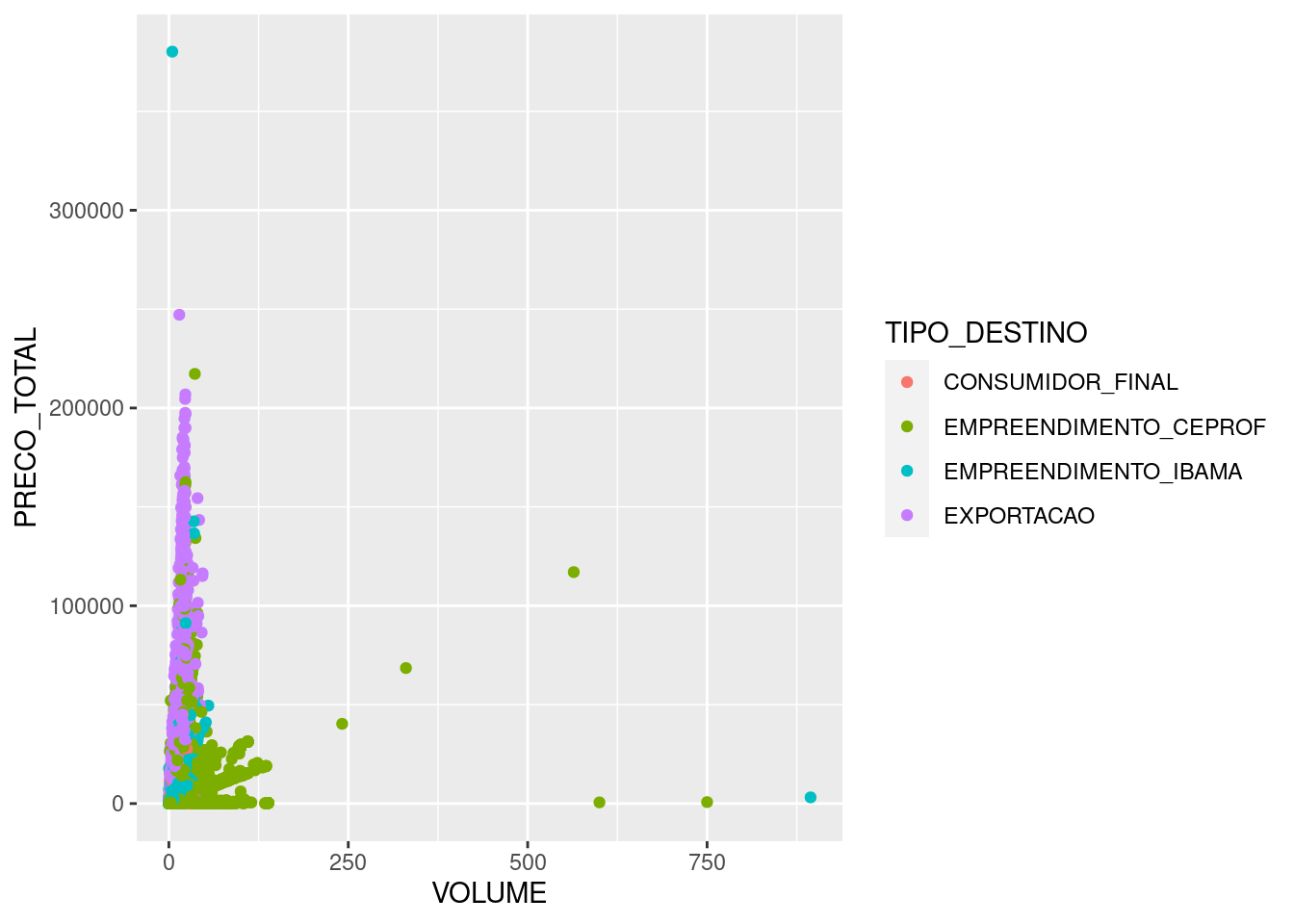
Note que não houve a necessidade de re-especificar os argumentos data e aes() para a função geom\_point().

Poderíamos melhorar nosso gráfico adicionando cores dentro de aes(). Isso acrescentaria mais informação. Poderíamos atribuir cores diferentes para cada tipo de destino das toras de madeira:

tb\_ibama2 %>%

ggplot(mapping = aes(x=VOLUME, y=PRECO\_TOTAL))+

geom\_point(aes(color=TIPO\_DESTINO))



*# ou*

*# tb\_ibama2 %>%*

*# ggplot(mapping = aes(x=VOLUME, y=PRECO\_TOTAL, color=TIPO\_DESTINO))+*

*# geom\_point()*

Podemos atribuir diferentes intensidades aos pontos, dependendo do valor de uma terceira variável (de preferência contínua entre 0 e 1), utilizando o argumento alpha. Como no nosso dataset do IBAMA, não temos tal variável, vamos criar uma variável contínua por meio da simulação de valores dentro de uma distribuição de probabilidade Normal (N(0,1)𝑁(0,1)):

*# simulando valores*

tb\_ibama2 <- tb\_ibama2 %>%

mutate(quant\_sim = rnorm(n()),

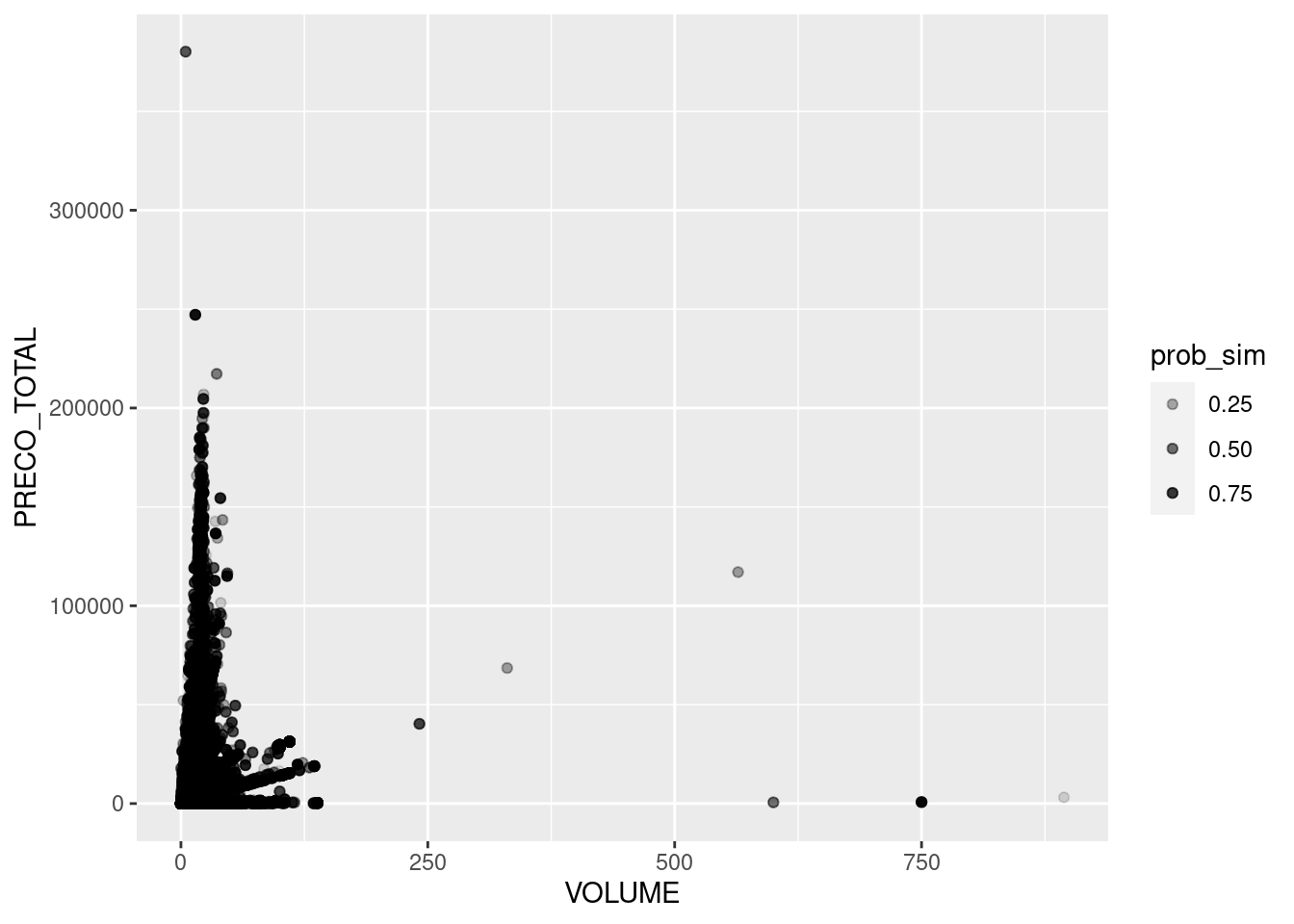
prob\_sim = pnorm(quant\_sim)

)

tb\_ibama2 %>%

ggplot(mapping = aes(x=VOLUME, y=PRECO\_TOTAL, alpha=prob\_sim))+

geom\_point()



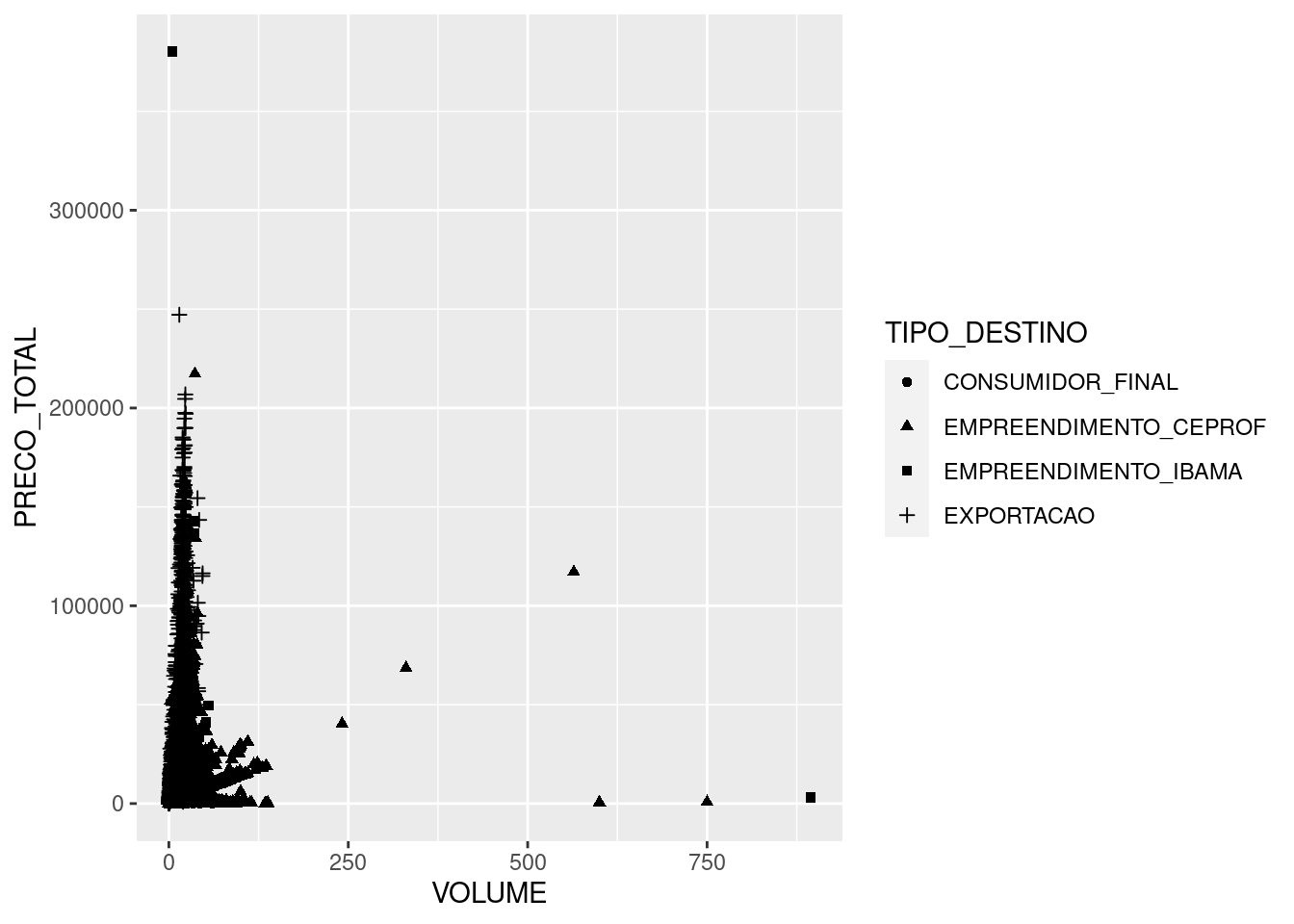
Neste nosso caso, esse gráfico não é muito útil porque temos vários pontos sobrepostos e isto dificulta a percepção de como as cores estão variando. No entanto, serve para ilustrar essa possibilidade com ggplot2.

Uma melhor forma de demonstrar os diferentes tipos de destino seria utilizar diferentes formas (argumento shape):

tb\_ibama2 %>%

ggplot(mapping = aes(x=VOLUME, y=PRECO\_TOTAL, shape=TIPO\_DESTINO))+

geom\_point()



**Mapping vs Setting aesthetics**

Nos exemplos anteriores, poderíamos optar por usar apenas uma única cor ou um único shape que não fosse o padrão de “bolinha” na cor preta. O que deveríamos fazer então, ao invés de **mapear** variáveis para **aesthetic**, é **setar** a **aesthetic** constante.

Quando vamos mapear um **aesthetic**, as variáveis são passadas **dentro** de aes(); ao passo que quando vamos setar um **aesthetic** como constante, o valor é passado fora de aes().

Vejamos os exemplos anteriores com um *aesthetic* constante para cor e shape:

tb\_ibama2 %>%

ggplot(mapping = aes(x=VOLUME, y=PRECO\_TOTAL))+

geom\_point(color = "red")

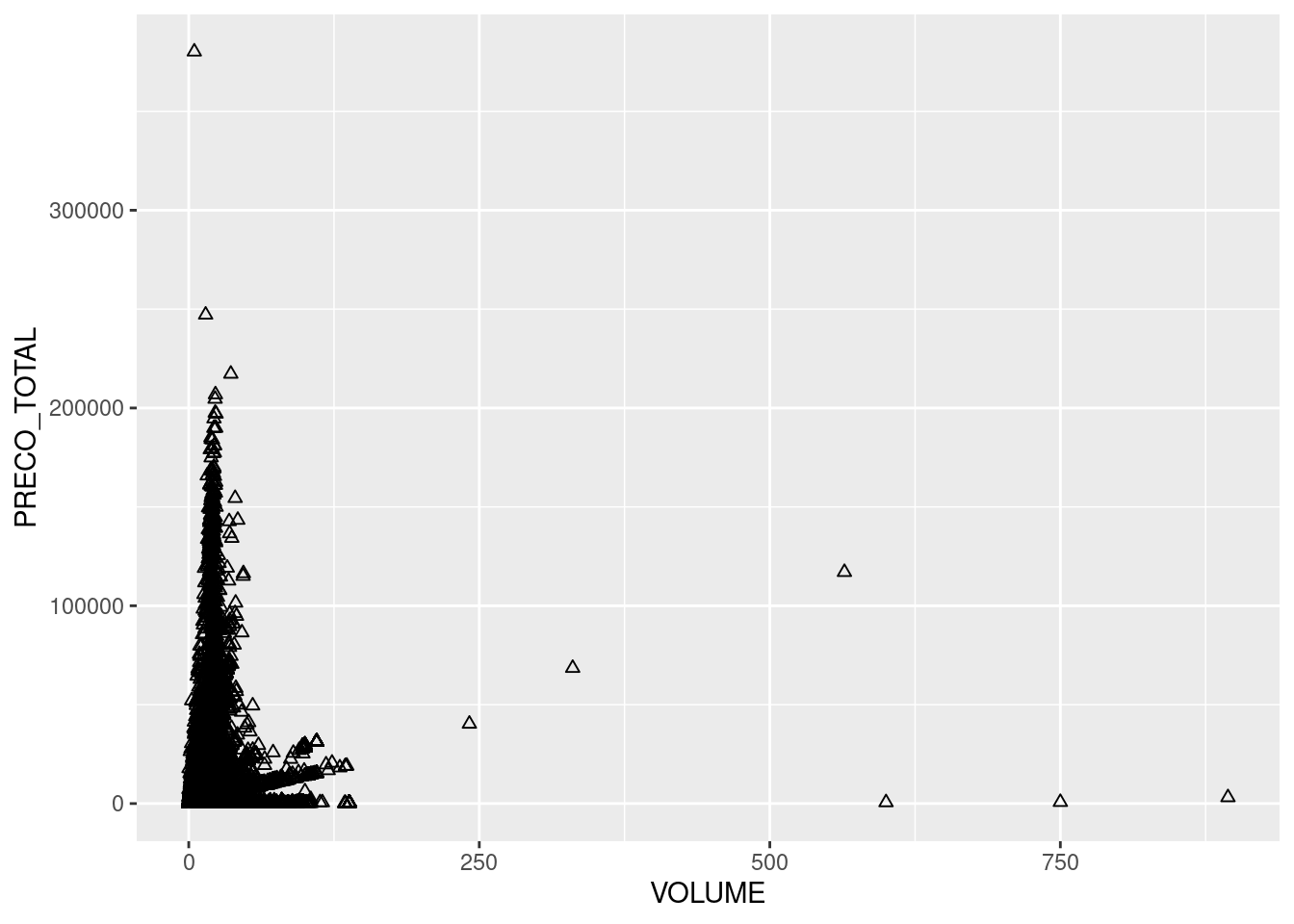
Gráfico, Gráfico de dispersão

Descrição gerada automaticamente

tb\_ibama2 %>%

ggplot(mapping = aes(x=VOLUME, y=PRECO\_TOTAL))+

geom\_point(shape = 2)



Note que devemos passar as constantes para geom\_point() (fora de aes()) - o a função de objetos geométricos - e não para ggplot().

Veja quantos *shapes* podemos *usar* no **R**:

d=data.frame(p=c(0:25,32:127))

ggplot() +

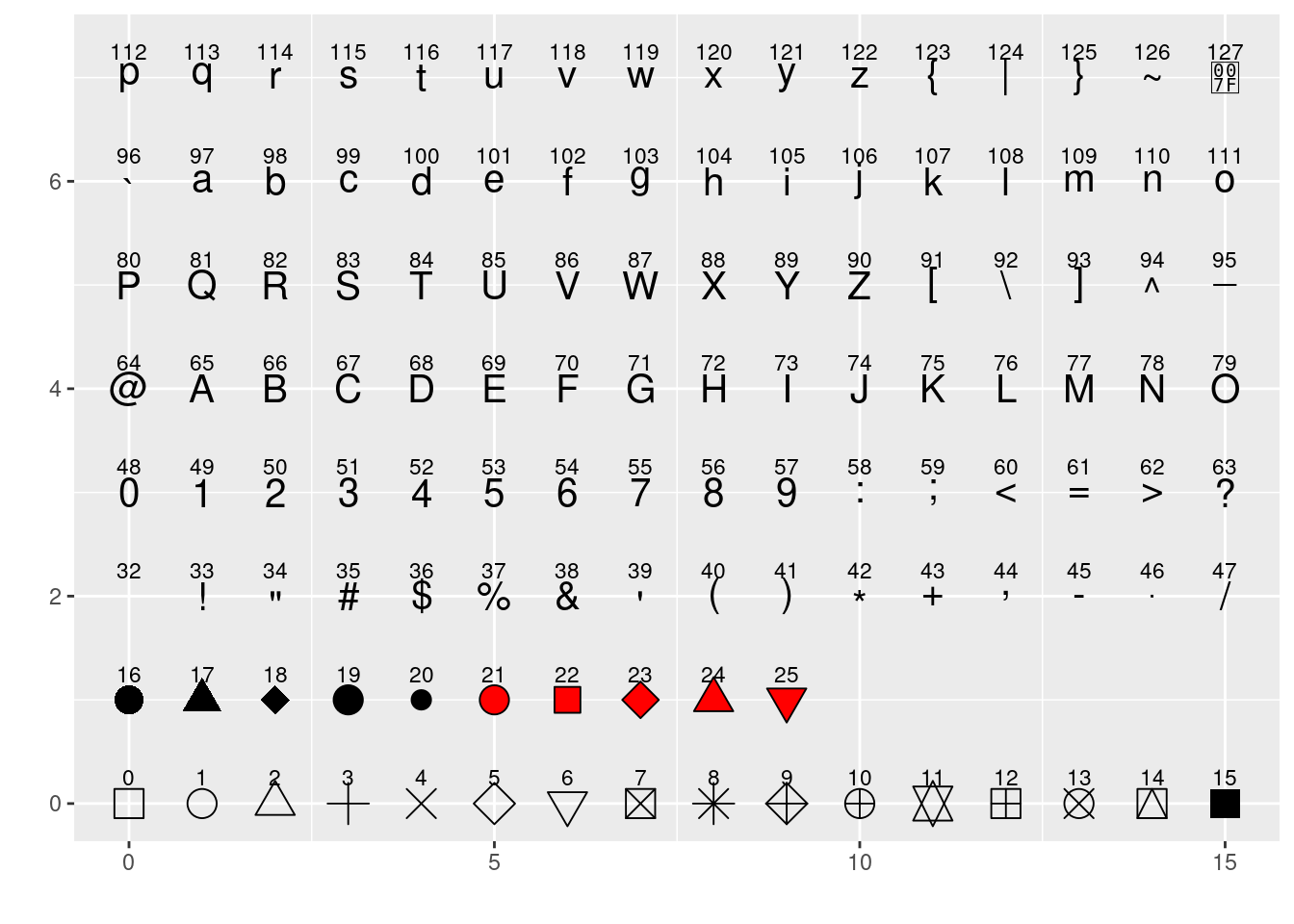
scale\_y\_continuous(name="") +

scale\_x\_continuous(name="") +

scale\_shape\_identity() +

geom\_point(data=d, mapping=aes(x=p%%16, y=p%/%16, shape=p), size=5, fill="red") +

geom\_text(data=d, mapping=aes(x=p%%16, y=p%/%16+0.25, label=p), size=3)



*# http://sape.inf.usi.ch/quick-reference/ggplot2/shape*

Cuidado ao setar uma *aesthetic* como constante dentro de aes(), pois isso pode resultar em um comportamento inesperado.

tb\_ibama2 %>%

ggplot(mapping = aes(x=VOLUME, y=PRECO\_TOTAL)) +

geom\_point(aes(color = "green"))

Gráfico, Gráfico de dispersão

Descrição gerada automaticamente

Embora tenhámos setado a cor como "green", o fato de termos feito dentro de aes() faz com que o ggplot

**Geoms**

**Geoms** são as formas geométricas a serem plotadas nos gráficos, exemplo geom\_line(), geom\_col(), geom\_point(), etc. Eles diferem de um para o outro nos tipos de *aesthetics* que eles *requerem* ou *entendem*. Por exemplo: geom\_point() requer aes(x, y), ao passo que geom\_bar() requer aes(x) apenas.

Para obter ajuda digite ?geom\_x.

**Exemplos para *Geoms***

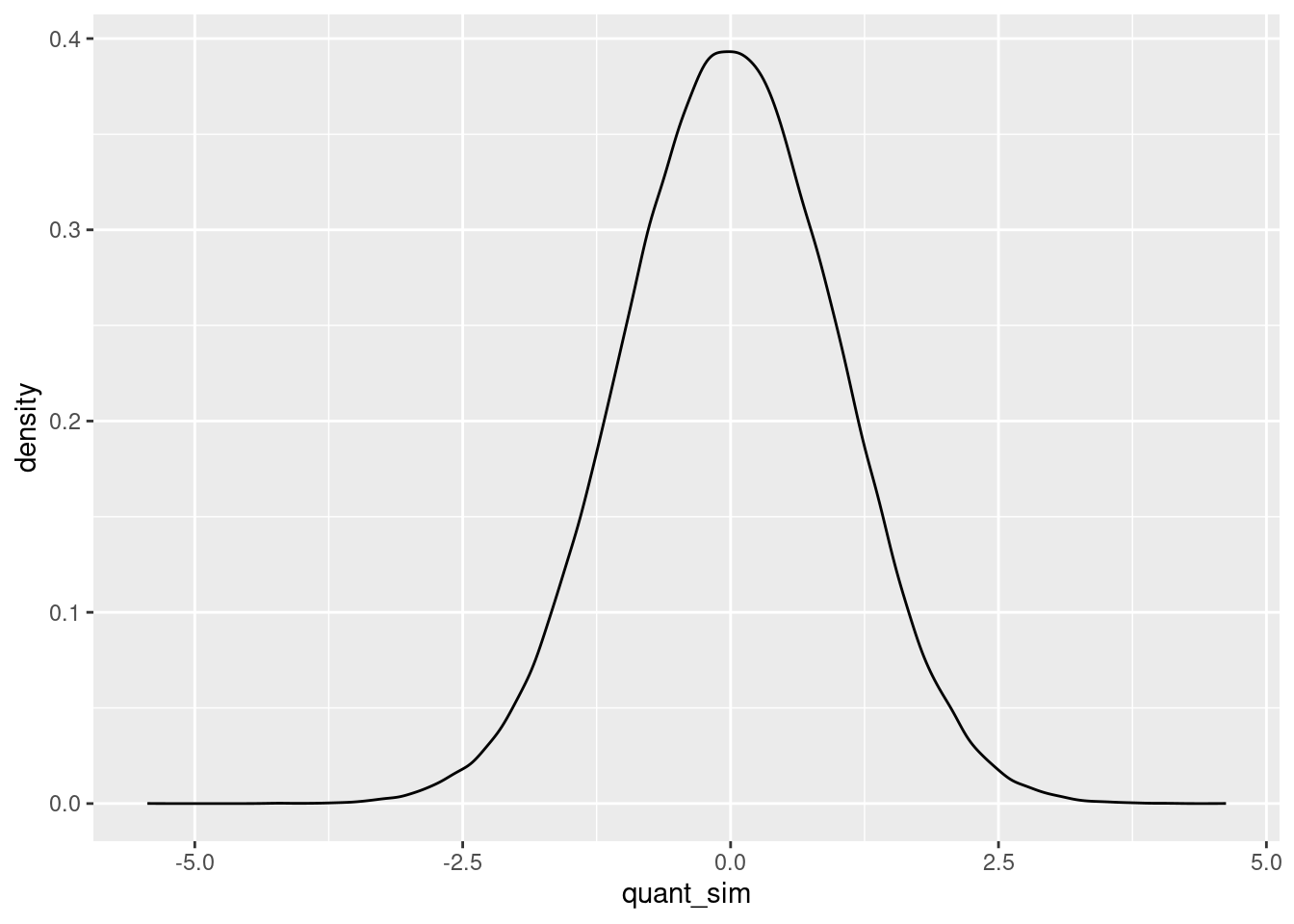
***geoms* para uma variável numérica (contínua de preferência)**

c <- tb\_ibama2 %>%

ggplot(mapping = aes(x=quant\_sim))

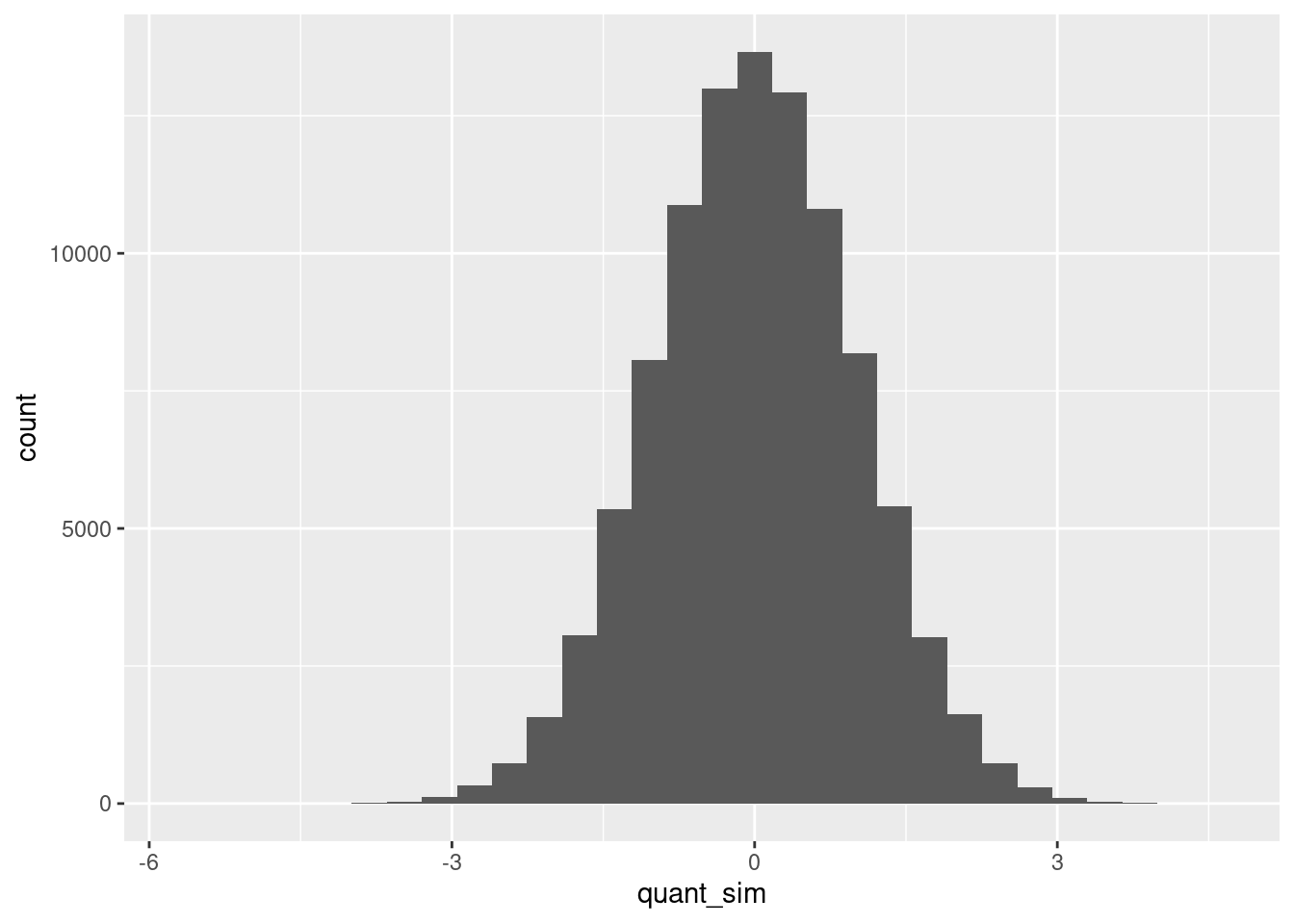
**DICA:** Note que podemos atribuir o resultado do gráfico a um objeto no R e acrescentar outras *layers* posteriormente.

c + geom\_density()



c + geom\_histogram()

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



***geoms* para uma variável numérica discreta**

*# contagem*

tb\_ibama2 %>%

ggplot(mapping = aes(x=TIPO\_GF))+

geom\_bar()

Gráfico, Gráfico de barras

Descrição gerada automaticamente

***geoms* para duas variáveis (contínuas)**

geom\_point() tambpem pode ser usado neste caso:

e <- tb\_ibama2 %>%

ggplot(mapping=aes(x=VOLUME, y=PRECO\_TOTAL))

e + geom\_point()

Gráfico, Gráfico de dispersão

Descrição gerada automaticamente

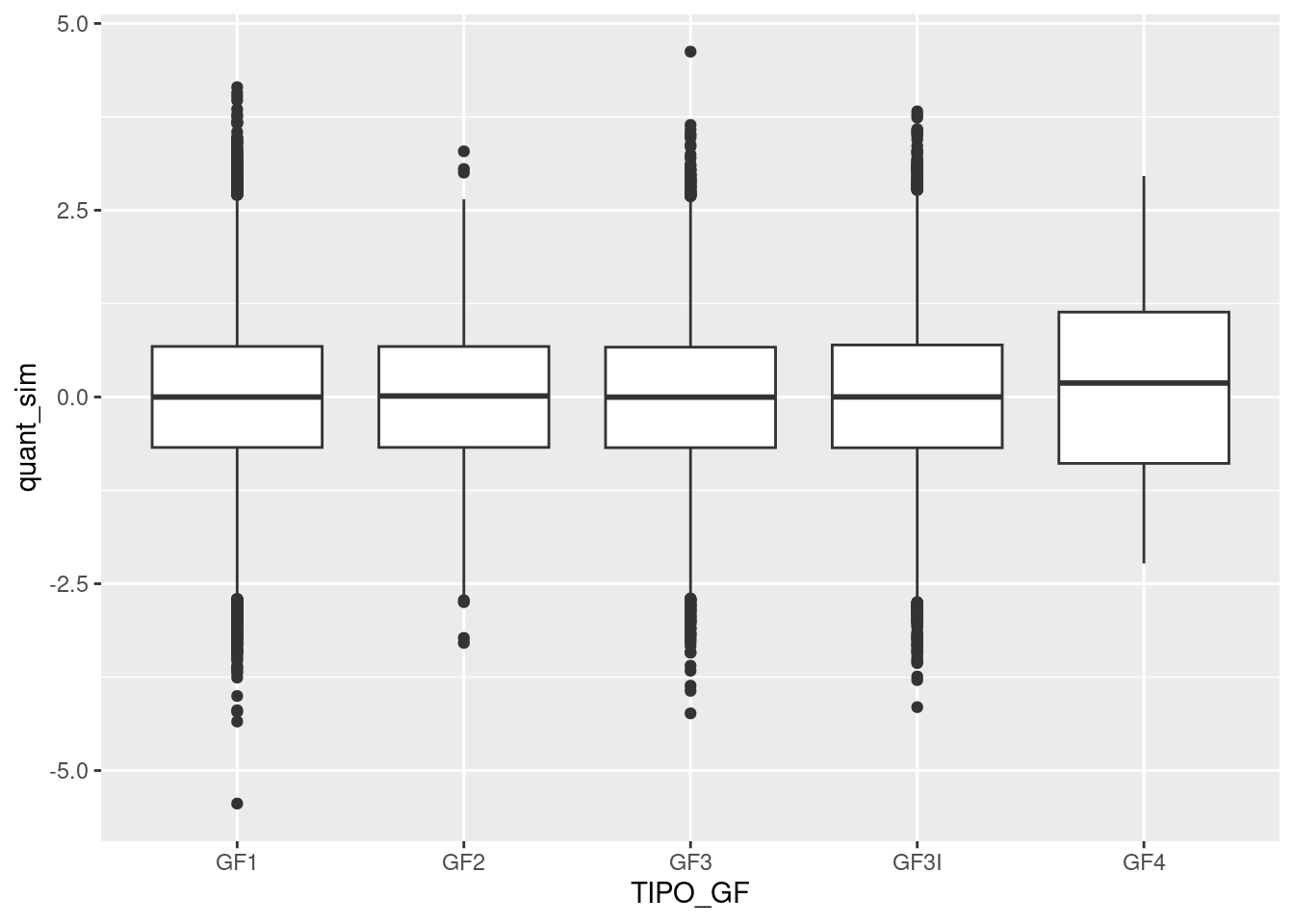
***geoms* para duas variáveis (uma qualitativa e uma numérica)**

Os gráficos abaixo são ideais para verificar como os valores se distribuem, separando-os em difernetes categorias.

f <- tb\_ibama2 %>%

ggplot(mapping=aes(TIPO\_GF, quant\_sim))

f + geom\_boxplot()



f + geom\_violin()

Diagrama

Descrição gerada automaticamente

***geoms* para gráficos de linha**

Neste ponto, vamos trocar de *dataset*. Passaremos a utilizar agora o dataset tb\_censo\_escolar. Faremos algumas transformações com dplyr e seguiremos direto para o plot usando pipe %>%.

tb\_censo\_escolar %>%

group\_by(SG\_UF, NU\_ANO\_CENSO) %>%

summarise(QTD\_TOTAL = sum(QTD\_ALUNOS, na.rm = TRUE)) %>%

ggplot(mapping=aes(NU\_ANO\_CENSO, QTD\_TOTAL, colour=SG\_UF))+

geom\_line()

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

Gráfico, Gráfico de linhas

Descrição gerada automaticamente

No caso de gráficos de linhas, ao invés de usar cores diferentes, poderíamos ainda alterar o tipo de linha conforme as categorias de uma 3ª variável, por meio do argumento linetype:

tb\_censo\_escolar %>%

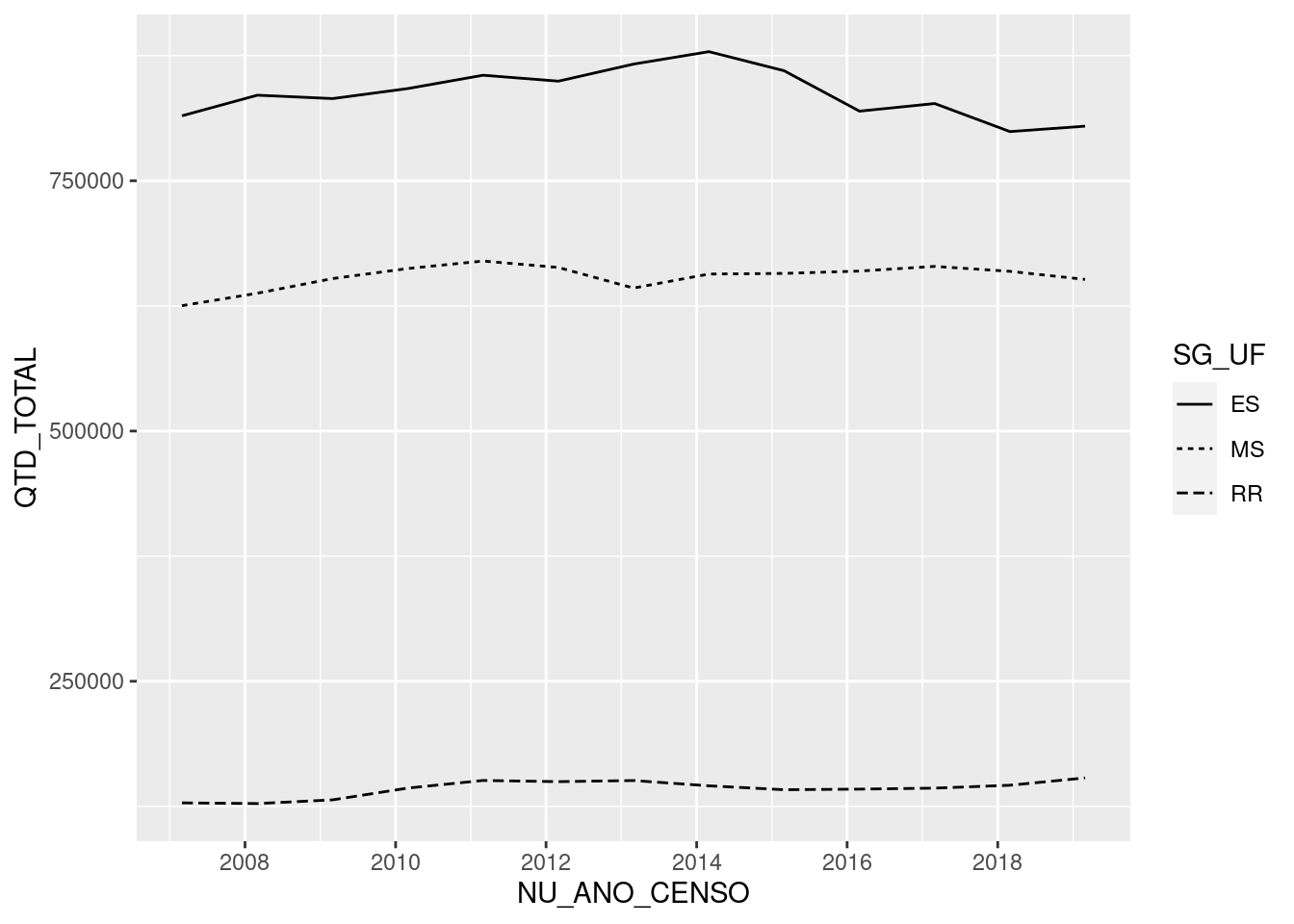
group\_by(SG\_UF, NU\_ANO\_CENSO) %>%

summarise(QTD\_TOTAL = sum(QTD\_ALUNOS, na.rm = TRUE)) %>%

ggplot(mapping=aes(NU\_ANO\_CENSO, QTD\_TOTAL, linetype=SG\_UF))+

geom\_line()

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



**DICA:** Há também outros *geoms* que você encontrará na página de ajuda.

**Positions**

Um aspecto dos gráficos de ggplot2 que não chega a ser uma *layer*, mas que será importante de observarmos diz respeito a ajustes de posição. Elas serão muito importantes para gráficos de de barras com duas variáveis.

Os ajustes de posição definirão como os *geoms* se localizam no gráfico, de modo que não ocupem o mesmo espaço. No exemplo do gráfico de barras, podemos fazer gráficos de barras justapostas ou empilhadas.

-position="stacked é o ajuste padrão de geom\_point(). Ela coloca cada objeto na posição exata do contexto do gráfico. No caso de gráficos de barras, teríamos barras acumuladas (*stacked*):

*# tb\_censo\_escolar %>%*

*# ggplot(mapping=aes(x=SG\_UF, fill=factor(CO\_ETAPA\_ENSINO)))+*

*# geom\_bar(position="stack")*

*# precisa passar CO\_ETAPA\_ENSINO como factor, senao ele lê como numerica*

tb\_censo\_escolar %>%

ggplot(mapping=aes(x=SG\_UF, fill=factor(NU\_ANO\_CENSO)))+

geom\_bar(position="stack") *# ou simplemente geom\_bar()*

Gráfico, Gráfico de barras

Descrição gerada automaticamente

**DICA:** Note que *forçamos* a variável *NU\_ANO\_CENSO* como um *factor*, para que o R a trate como uma variável categórica e não como uma data para realizar a plotagem. Um *factor* é uma abstração que o R faz para armazenas variáveis categóricas de mod mais eficiente na memória do computador. Para cada valor ele cria um *level* numérico. É muito mais eficiente guardar um número do que um nome ou uma data, por exemplo.

* **position = “dodge”** coloca objetos sobrepostos um ao lado de outro. Nesse caso, teremos barras justapostas:

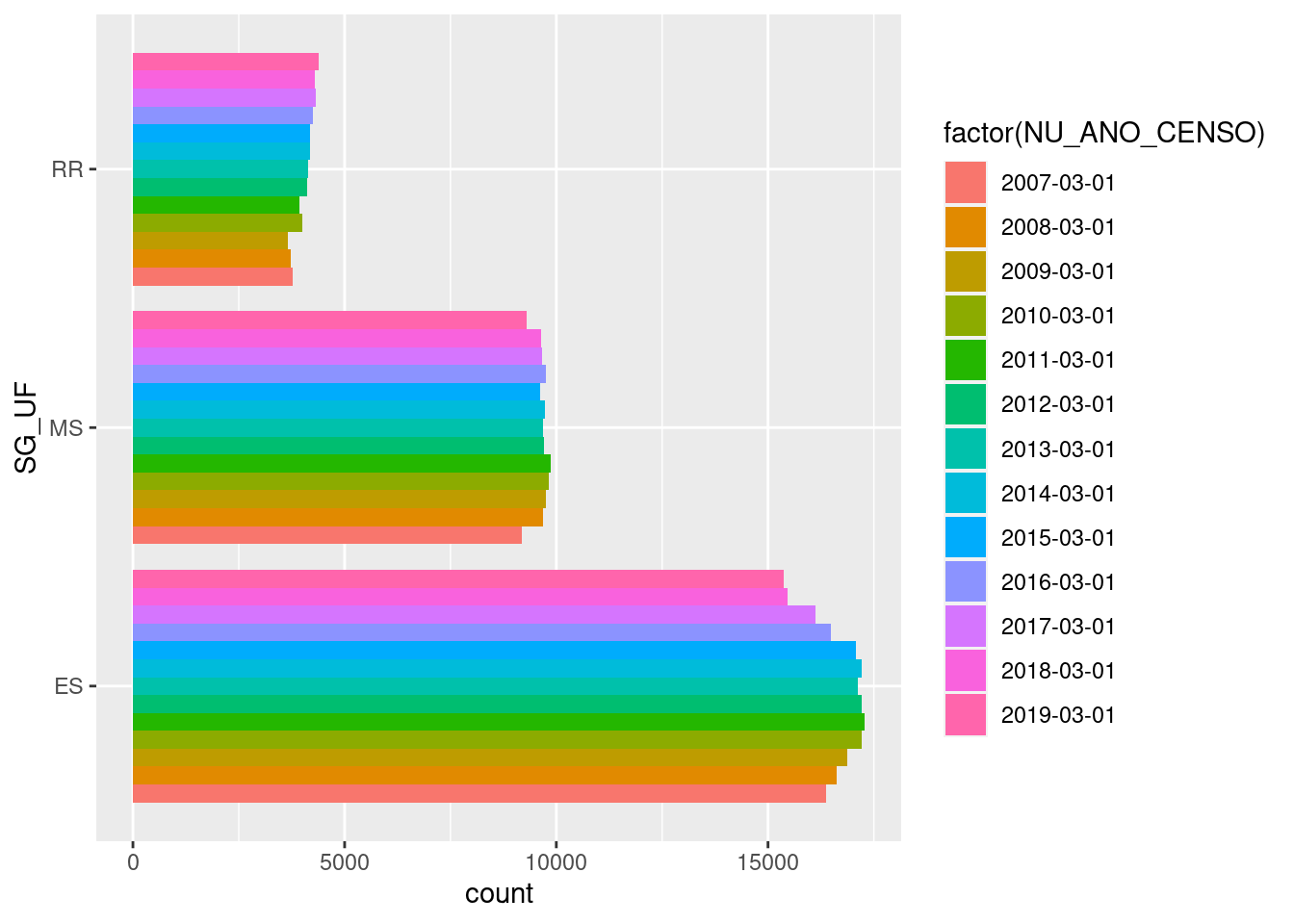
gg\_barras <- tb\_censo\_escolar %>%

ggplot(mapping=aes(x=SG\_UF, fill=factor(NU\_ANO\_CENSO)))+

geom\_bar(position="dodge")

**DICA:** Se voce quiser inverter as coordenadas do gráfico, pode fazer isso trocando os parâmetros x e y e em aes(), ou utilizando a função coord\_flip():

gg\_barras + coord\_flip()



* **position=“fill”** também empilhará os elementos, mas irá normalizar a altura de todas as barras para que fiquem na mesma escala. Isso torna mais fácil comparar as proporções entre os grupos:

tb\_censo\_escolar %>%

ggplot(mapping=aes(x=SG\_UF, fill=factor(NU\_ANO\_CENSO)))+

geom\_bar(position="fill")

Gráfico, Gráfico de barras

Descrição gerada automaticamente

* **position=“jitter”** é útil para gráficos de dispersão, pois hpa um problema muito comum neste tipo de gráfico que é a sobreposição de pontos (*overplotting*). Com *jitter*, *ggplot2* adiciona um ruído aleatório nas posições de X e Y para que eles não se sobreponham:

*# mpg %>%*

*# ggplot(mapping=aes(x=displ, y=hwy))+*

*# geom\_point(position="jitter")*

Há outros ajustes de posição bem úteis, como:

* **position=“nudge”** afasta os rótulos dos pontos;
* **position=“identity”** sobrepõe os elementos uns sobre os outros.

**Facetting**

As **facetts** em ggplot2 nos permitem criar vários gráficos com base em subconjuntos de dados e plotá-los numma mesma área. É uma técnica muito útil (quando temos variáveis categóricas) e desejamos visualizar o comportamento de diferentes categorias num mesmo gráfico.

Há duas funções que permitem criar facetas: facet\_wrap() e facet\_grid().

Tanto **facet\_wrap** quanto **facet\_grid** permitem a divisão do plot da(s) variável(is) passadas em aes() de acordo com as categorias de uma 3ª ou 3ª e 4ª variáveis.

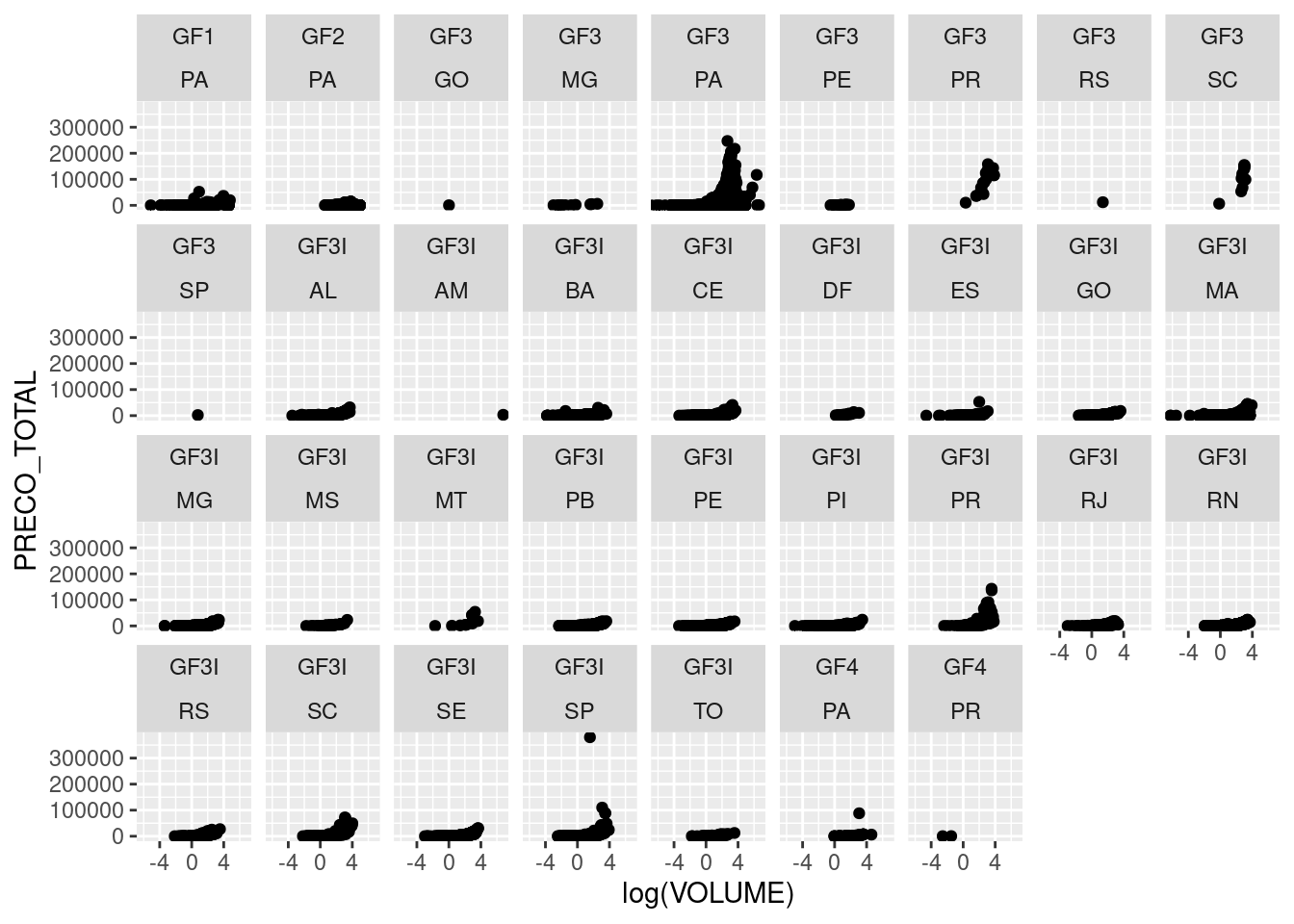
Voltaremos aos exemplos com a base modificada do IBAMA tb\_ibama2. Usaremos o log do volume apenas para que os valores dos eixos não se sobreponham. Embora seja possível ajustar isso com as funções de ggplot2, por uma questão de priorização e espaço optamos por não abordar essa funcionalidade nesta apostila.

tb\_ibama2 %>%

ggplot()+

geom\_point(mapping=aes(x=log(VOLUME), y=PRECO\_TOTAL))+

facet\_wrap(TIPO\_GF ~ UF\_DESTINATÁRIO, nrow = 4)



Note acima que podemos definir o número de linhas desejado para plotagem com o argumento nrow ou o número de colunas com ncol e automaticamente o outro argumento será determinado. O símbolo ~ é muito utilizado no **R**. Nesse casos, isso significa que as variáveis de aes() serão plotadas em função das variáveis STATUS\_GF e TIPO\_DESTINO, ou seja, para cada combinação das categorias destas variáveis, será feito um plot com as variáveis passadas em aes(). Note que o arranjo das combinações das categorias STATUS\_GF e TIPO\_DESTINO são passadas nas colunas de gráficos, como se fossem os títulos de cada plot específico.

No caso de facet\_grid(), a organização do *output* será um pouco diferente. Será feito, como o próprio nome da função diz, um *grid*, onde as linhas dirão respeito a uma das variáveis e a coluna à outra variável. É como se fosse uma uma matriz de pequenos gráficos. Como resultado, você verá que há cruzamentos entre as variáveis *TIPO\_DESTINO* e *STATUS\_GF* que resultam em um gráfico vazio por ausência de dados que estejam naquelas respectivas categorias das duas variáveis. No caso anterior, com facte\_wrap() isso não ocorre.

tb\_ibama2 %>%

ggplot()+

geom\_point(mapping=aes(x=log(VOLUME), y=PRECO\_TOTAL))+

facet\_grid(TIPO\_GF ~ UF\_DESTINATÁRIO)

Gráfico, Gráfico de caixa estreita

Descrição gerada automaticamente

**Títulos, rótulos, temas e legendas**

Há duas formas de alterar títulos e rótulos dos eixos em ggplot2. Você pode optar por ggtitle("título do gráfico"), xlab("rótulo eixo x"), ylab("rótulo eixo y"). Ou então, pode usar labs(). No caso, de usar labs, você pode alterar todos esses campos e outros, como o de legenda dentro da mesma função.

Note, no entanto, que o parâmetro que controla o título da legenda muda dependendo do parâmetro dentro de mapping = aes() a partir do qual a legenda foi gerada. Normalmente, a legenda é gerada ou pelos parâmetros colour, fill, size, shape, linetype e alpha. Esses mesmos parâmetros, dependendo do caso, deverão ser passados dentro da função labs() para controlar o título da legenda, como por exemplo: labs(..., shape = "Título da legenda").

tb\_ibama2 %>%

ggplot()+

geom\_point(mapping=aes(x=VOLUME, y=PRECO\_TOTAL, colour=TIPO\_GF))+

labs(title="Título do gráfico",

x = "Rótulo eixo x",

y = "Rótulo eixo y",

colour = "Título da Legenda")

Gráfico, Gráfico de dispersão

Descrição gerada automaticamente

*# OU*

*# tb\_ibama2 %>%*

*# ggplot()+*

*# geom\_point(mapping=aes(x=VOLUME, y=PRECO\_TOTAL, colour=TIPO\_GF))+*

*# ggtitle("Título do gráfico")+*

*# xlab("Rótulo eixo x")+*

*# ylab("Rótulo eixo y")+*

*# labs(colour="Título da legenda")*

Há ainda funções específicas que também controlam o nome da legenda dependendo do argumento utilizado para gerar a legenda dentro de aes() e também a partir de qual tipo de dados, se contínuos ou discretos. Temos os seguintes casos:

* scale\_GERADOR\_discrete(name="título da legenda")
* scale\_GERADOR\_continuous(name="título da legenda")

Dessa forma, temos: scale\_fill\_discrete(name="título da legenda"), scale\_fill\_continuous(name="título da legenda"), scale\_colour\_discrete(name="título da legenda"), scale\_colour\_continuous(name="título da legenda"), e assim por diante para os demais argumentos geradores da legenda, como size, linetype, alpha e shape.

tb\_censo\_escolar %>%

group\_by(SG\_UF, NU\_ANO\_CENSO) %>%

summarise(QTD\_TOTAL = sum(QTD\_ALUNOS, na.rm = TRUE)) %>%

ggplot(mapping=aes(NU\_ANO\_CENSO, QTD\_TOTAL, linetype=SG\_UF))+

geom\_line()+

scale\_linetype\_discrete(name="UF")

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

Gráfico, Gráfico de linhas

Descrição gerada automaticamente

Podemos inclusive ter mais de um argumento de *mapping* dentro de aes() e podemos controlar as legendas de cada um, adicionando mais uma camada ao gráfico:

tb\_ibama2 %>%

ggplot()+

geom\_point(mapping=aes(x=VOLUME, y=PRECO\_TOTAL,

size=preco\_unidade,

colour=TIPO\_GF))+

ggtitle("Produção madeireira - Dados IBAMA")+

xlab("Volume toras")+

ylab("Preço Total toras")+

scale\_size\_continuous(name="Preço por unidade de volume")+

scale\_colour\_discrete(name="Grupo")

Gráfico, Gráfico de dispersão

Descrição gerada automaticamente

**Alterar cores e paleta de cores**

Você deve ter notado, que na maioria dos gráficos, o ggplot2 utiliza cores padrão da biblioteca para as plotagens. Se você não estiver contente com a paleta de cores padrão utilizada pelo pacote no *mapping* de suas variáveis, você pode definir manualmente esses cores utilizando uma outra paleta ou outras cores de sua preferência.

O modo de alterar a paleta de cores utilizadas em um argumento de *mapping* dentro de aes() é por meio de funções do tipo scale\_GERADOR\_manual(), alterando-se o parâmetro values.

tb\_ibama2 %>%

ggplot()+

geom\_point(mapping=aes(x=VOLUME, y=PRECO\_TOTAL,

size=preco\_unidade,

colour=TIPO\_GF))+

ggtitle("Produção madeireira - Dados IBAMA")+

xlab("Volume toras")+

ylab("Preço Total toras")+

scale\_size\_continuous(name="Preço por unidade de volume")+

scale\_colour\_manual(values=c("red", "blue", "green", "orange", "yellow"))

Gráfico, Gráfico de dispersão

Descrição gerada automaticamente

Por meio dessas mesmas funções do tipo de funções do tipo scale\_GERADOR\_manual(), podemos alterar também os rótulos das categorias exibidas na legenda utilizando o argumento labels.

tb\_ibama2 %>%

ggplot()+

geom\_point(mapping=aes(x=VOLUME, y=PRECO\_TOTAL,

size=preco\_unidade,

colour=TIPO\_GF))+

ggtitle("Produção madeireira - Dados IBAMA")+

xlab("Volume toras")+

ylab("Preço Total toras")+

scale\_size\_continuous(name="Preço por unidade de volume")+

scale\_colour\_manual(values=c("red", "blue", "green", "orange", "yellow"),

labels=c("tipo1", "tipo2", "tipo3", "tipo4", "tipo5"))

Gráfico, Gráfico de dispersão

Descrição gerada automaticamente

Como menicionamos várias vezes atpe este ponto do curso, no **R** existem várias formas de fazer uma mesma coisa. As funções do tipo scale\_GERADOR\_manual() também permite alterar o título da legenda com o argumento name:

tb\_ibama2 %>%

ggplot()+

geom\_point(mapping=aes(x=VOLUME, y=PRECO\_TOTAL,

size=preco\_unidade,

colour=TIPO\_GF))+

ggtitle("Produção madeireira - Dados IBAMA")+

xlab("Volume toras")+

ylab("Preço Total toras")+

scale\_size\_continuous(name="Preço por unidade de volume")+

scale\_colour\_manual(values=c("red", "blue", "green", "orange", "yellow"),

labels=c("tipo1", "tipo2", "tipo3", "tipo4", "tipo5"),

name="Grupo")

Gráfico, Gráfico de dispersão

Descrição gerada automaticamente

**Área de Plotagem e Tema**

Como já afirmamos antes, com ggplot2 é possível modificar praticamente todos os parâmetros do seu gráfico. E normalmente há mais de uma forma para se fazer cada uma das alterações. Com a função theme(), é possível controlar desde o tamanho dos elementos do de plotagem do gráfico, passando pelo tamnho do texto dos rótulos e indo até o tamanho dos quadrados ou “bolinhas” da legenda. No entanto, esta será uma pesquisa a ser feita por você.

Nesta seção traremos apenas algumas funções do ggplot2 que trazem alguns temas pré-formatados:

* theme\_gray(): plano de fundo cinza e linhas de grid brancas. É como se trouxesse os dados para frente para facilitar comparações;
* theme\_bw(): plano de fundo branco e linhas de grid cinzas. Recomendado para apresentações utilizando projetor;
* theme\_linedraw(): plano de fundo branco e linhas de grid com largura diferenciada.
* theme\_light(): linhas de grid cor cinza claro e com presença de eixos. Parecido como theme\_linedraw();
* theme\_dark(): igual ao theme\_light() mas com plano de fundo escuro. Faz linhas coloridas finas se destacarem;
* theme\_minimal(): tema minimalista sem plano de fundo;
* theme\_classic(): tema clássico sem linhas de grid;
* theme\_void(): tema vazio, sem cores de fundo e grid. Recomendado para plots de coordenadas não-usuais ou para desenhos.

tb\_ibama2 %>%

ggplot()+

geom\_point(mapping=aes(x=VOLUME, y=PRECO\_TOTAL, colour=TIPO\_GF))+

labs(title="Título do gráfico", x = "Rótulo eixo x", y = "Rótulo eixo y", colour = "Título da Legenda")+

theme\_bw()

Gráfico, Gráfico de dispersão

Descrição gerada automaticamente

**DICA:** Estranhamente, ggplot2 alinha o título do gráfico sempre à esquerda. Você pode centralizar o título usando justamente a função theme(), que comentamos mais acima, escrevendo theme(plot.title = element\_text(hjust=0.5). 0.5 indica que o título deve ficar no meio do gráfico, conforme alinhamento horizontal especificado (parâmetro hjust). element\_text() é uma *helper function* que serve para controlar aspectos textuais como tamanho (size) e outros parâmetros como o próprio alinhamento horizontal.

**Referências da seção**

* *Ggplot themes gallery*. (2020). DATANOVIA website. URL <https://www.datanovia.com/en/blog/ggplot-themes-gallery/>
* Lin, A. *Introduction to ggplot2*. Slides. IDRE Statistical Consulting Group. URL <https://stats.idre.ucla.edu/stat/data/intro_ggplot2_int/ggplot2_intro_interactive.html#(1)>
* Lopes, J. G. (2019). *O guia do ggplot2: como fazer qualquer tipo de gráfico no R*. Blog: Explorando o Universo da Ciência de Dados. 05, 2017. URL <http://joseguilhermelopes.com.br/o-guia-do-ggplot2-como-fazer-qualquer-tipo-de-grafico-no-r/>
* Wickham, H.; Grolemund, G. (2016). *R for Data Science: Import, Tidy, Transform, Visualize, and Model Data*. O’Reilly Media. december 2016. 522 pages. Disponível em: [https://www.r4ds.co.nz](https://www.r4ds.co.nz/).
* Wickham, H. (2016). *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. 2016. URL [https://ggplot2.tidyverse.org](https://ggplot2.tidyverse.org/)

**Exercícios**

1. Escolha uma base que você tenha utilizado recentemente no trabalho e a partir da qual você tenha gerado algum gráfico no excel ou outro software. Tente reproduzir dois ou três gráficos utilizando o pacote ggplot2. Não se esqueça de procurar ajuda na internet, principalmente no *Stack Overflow* e na documentação do pacote, além da apostila.
2. Nos gráficos do exercício 1, há algum aspecto que você gostaria de melhorar? Pesquise sobre como fazê-lo utilizando ggplot2.
3. Sofistique seu gráfico do ggplot2 pro meio da função ggplotly(objeto\_criado\_com\_ggplot2) do pacote plotly. Você pode tornar o seu gráfico interativo e também construir outros gráficos diferentes com plotly.