Metodologias Informacionais com R

Módulo II: Visualização de dados

Telmo dos Santos Klipp (telmo.klipp@inpe.br)

Informações Gerais sobre o Curso

- Materiais disponibilizados via Classroom;
- O aprendizado requer a prática que será constante nas aulas;

Bibliografia Básica:

- Kennedy, R., & Waggoner, P. D. (2021). Introduction to r for social scientists: a tidy programming approach. CRC Press.

Bibliografia Complementar:

- Wickham, H., Çetinkaya-Rundel, M., & Grolemund, G. (2023). R for data science (2e): import, tidy, transform, visualize, and model data. "O'Reilly Media, Inc.". Disponível em: https://r4ds.hadley.nz/. Acesso em: 14 de junho, 2023. (Online)
- Damiani, A. et. al., (2022). Ciência de Dados em R. Curso-R. Disponível em: https://livro.curso-r.com. Acesso em: 12 de maio, 2023. (Online)
- de Aquino, J. A. (2014). R para cientistas sociais. Editora da UESC (editus). Disponível em: http://www.uesc.br/editora/. Acesso em: 12 de maio, 2023.
- de Oliveira, P. F., Guerra, S., McDonnell, R. (2018). Ciência de Dados com R: Introdução. Editora IBPAD. Disponível em: https://cdr.ibpad.com.br/index.html. Acesso em: 12 de maio, 2023. (Online)

Nas últimas aulas vimos:

- Manipulação de dados usando o pacote **dplyr**.
- O uso do operador pipe.
- Operações mais comuns sobre dados e respectivas funções do **dplyr**.
- O uso de algumas funções auxiliares do **dplyr**.

Tidyverse

Coleção de pacotes que possuem filosofia de design, gramática e estrutura de dados em comum e, permitem trabalho conjunto, clareza de código, reprodutibilidade, dentre outros benefícios.

Visualização de dados

A visualização de gráficos pode ser crucial para a fase de análise exploratória de dados - obter informações, entender os dados, relações entre variáveis, desenvolver ou validar hipóteses. Não obstante, também é importante em etapas posteriores de comunicação e divulgação - em relatórios, documentos científicos, páginas na internet, entre outros.

O R possui como pacote gráfico básico o **graphics** (vem pré-instalado). Outros pacotes gráficos são:

- Gráficos estáticos: **ggplot2** (gráficos de tipos diversos), **lattice** (gráficos essencialmente do tipo *trellis*);
- Gráficos interativos: **plotly**, **dygraphs**, **highcharter**, **ggvis**, **gganimate** (gera animações);
- Mapas estáticos: **graphics**, **ggplot2**, **sp**, **sf**, **tmap**, **ggmap**;
- Mapas interativos: **gganimate**, **leaflet**, **mapview**, **tmap**, **plotly**;
- Gráficos 3D: **graphics**, **plotly**, **plot3d**, **scatterplot3d**, **lattice**, **RGL**, **ggrgl**, **rayshader**, **rayrender** (cenas 3D).

Visualização de dados - ggplot2

Qual o melhor lugar para começar? Certamente o **ggplot2** popularizou-se como o pacote gráfico mais usado por usuários do R. Não obstante, esse pacote faz parte do **tidyverse**, sendo baseado na fisosofia *"Grammar of Graphics"*, conquistou muitos adeptos, bem como, influenciou outros pacotes. O **ggplot2** possui um site próprio, onde podem ser encontradas informações, *vignettes* e *cheat sheets*. Também existe uma extensa lista de pacotes que estendem as capacidades do **ggplot2**

Mas o que é *"Grammar of Graphics"*? Um conceito filosófico para criação de gráficos com descrição concisa dos seus componentes que permite a produção de um gráfico em camadas complementares.

A descrição sólida dos elementos do **ggplot2** através de *"Layored Grammar"*, permitiu uma padronização que favoreceu o desenvolvimento de extensões. Não obstante, essa padronização, favorece a intuição humana, o aprendizado, e a gestão dos pacotes gráficos ao longo do tempo.

Visualização de dados - ggplot2

Os cinco componentes de um gráfico segundo a interpretação de "Layered Grammar" no ggplot2 são:

1. Camada(s) - representação visual das propriedades físicas dos dados:
    - Dados (***data***);
    - Mapeamento de objetos (***aesthetic***) - define as variáveis que compõem o gráfico; divisões categóricas dos dados que são traduzidas em cores, formatos e tamanhos; elementos da legenda e agrupamentos;
    - Geometria dos objetos (***geom***) - basicamente o tipo do gráfico;
    - Transformações estatísticas (***stat***) - basicamente sumarização de dados;
    - Posicionamento (***position***) - permite o deslocamento dos elementos gráficos, se houver sobreposição.
2. Escalas (***scales***) - permite o controle sobre atributos estéticos como cores, formas e tamanhos dos elementos geométricos, espaçamento e disposição dos *labels* nos eixos e legenda, entre outros.
3. Sistema de coordenadas (***coord***) - permite definir posição/aparência dos objetos mapeados no plano.
4. Facetamento/tabulamento (***faceting***) - gera painéis contendo subdivisões dos dados baseadas em informações categóricas. Os subgráficos compartilham os mesmos atributos estéticos.
5. Tema (***theme***) - Define elementos gerais como cor do plano de fundo, tamanho de texto e tipo de fonte.

Visualização de dados - ggplot2

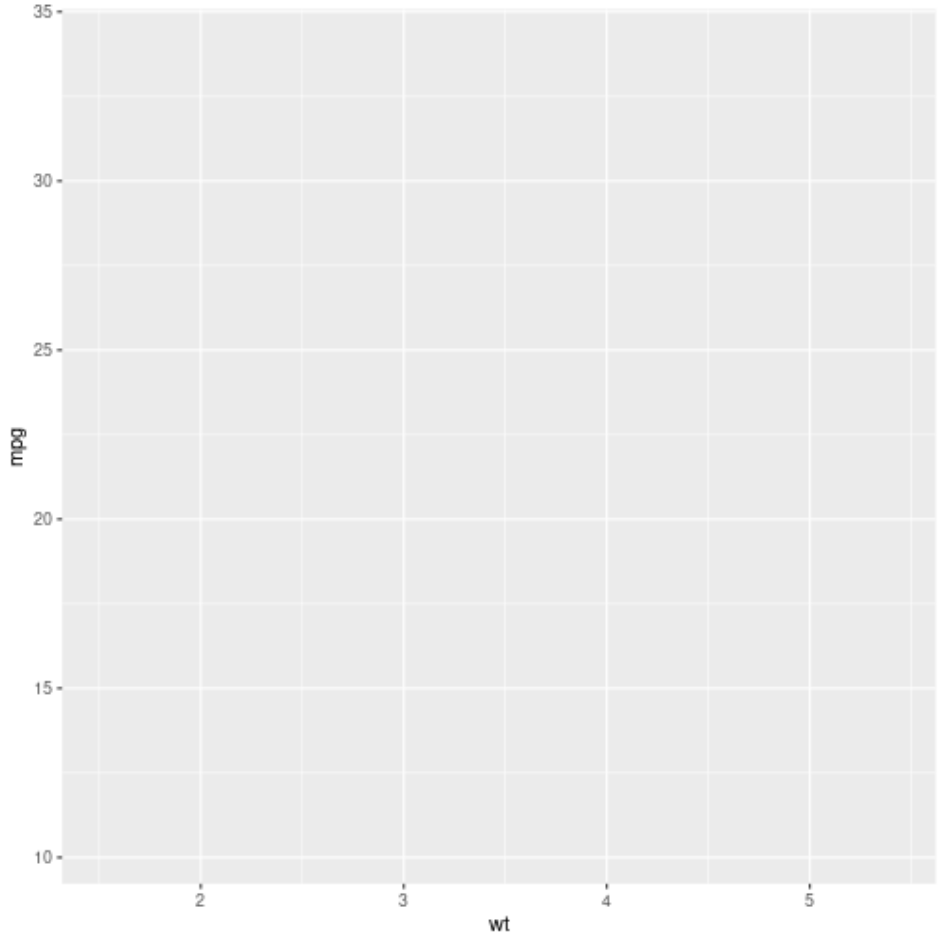
Um gráfico do **ggplot2** pode ter um ou mais conjunto de dados e multiplas camadas, assim:

- Podem ser geradas combinações de camadas, e
- Elementos gráficos podem ser alterados individualmente;

Essa flexibilidade permite personalizações e até criação de novos conceitos gráficos. Vejamos um exemplo básico de gráfico:

```
library(ggplot2) # Importar as bibliotecas
library(dplyr)   # que serão usadas
library(MASS)
```

```
ggplot(
  data = mtcars,
  mapping = aes(x = wt, y = mpg)
)
```



Visualização de dados - ggplot2

Vamos adicionar uma camada para criar um gráfico de dispersão (*scatter plot*). Este representa com pontos a relação entre duas variáveis numéricas, conforme o exemplo:

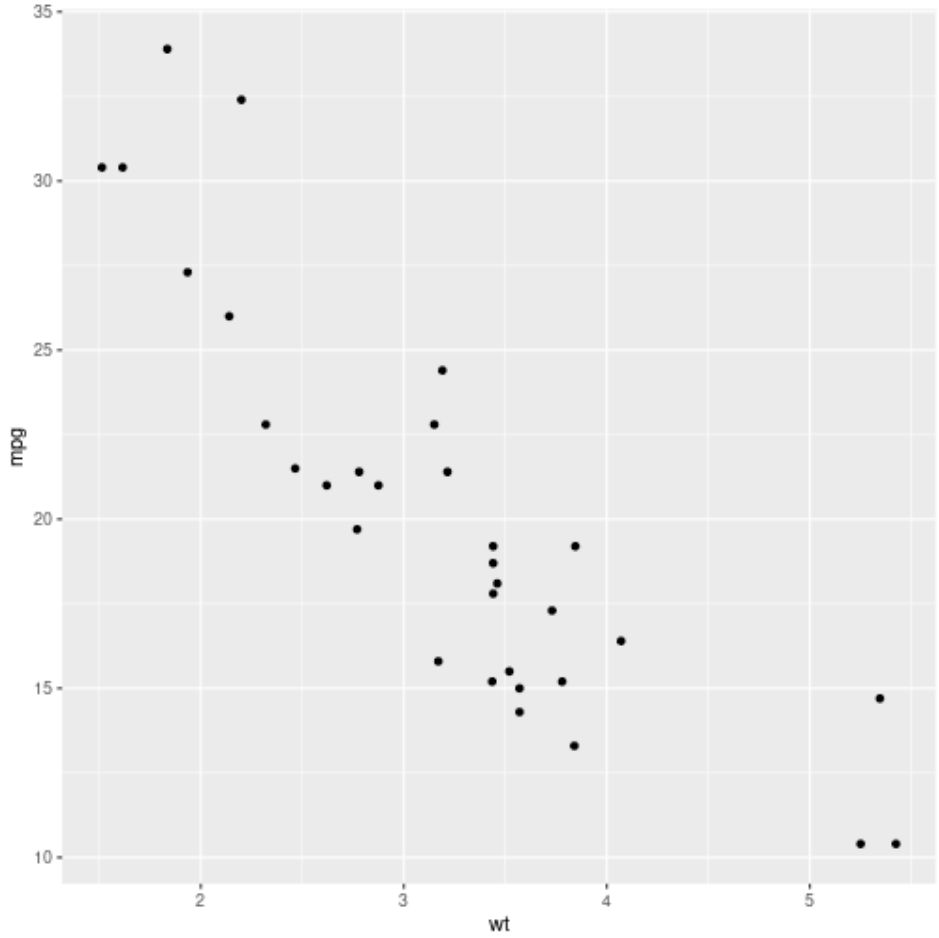
```
ggplot(mtcars, aes(x = wt, y = mpg)) +
  geom_point()
```

Ou:

```
ggplot() +
  geom_point(data = mtcars,
             aes(x = wt, y = mpg))
```

O correspondente no pacote **base** é:

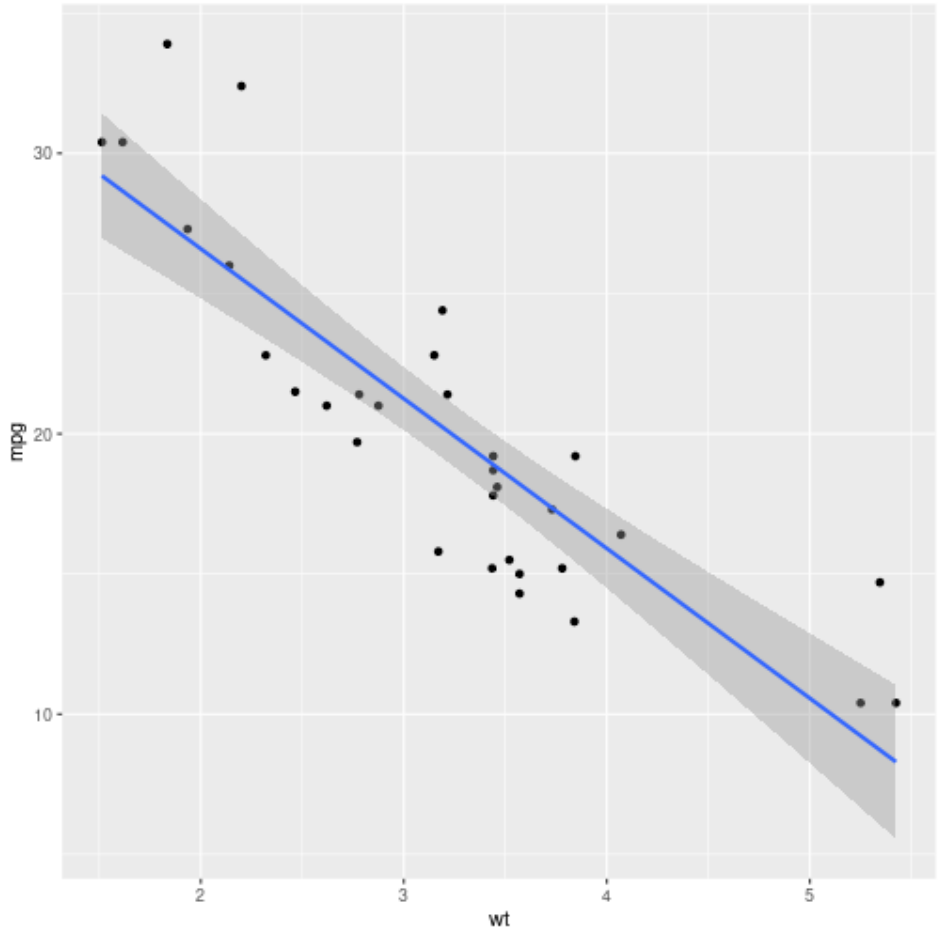
```
plot(mtcars$wt, mtcars$mpg)
```



Visualização de dados - ggplot2

Vamos adicionar camadas para criar um gráfico de dispersão (*scatter plot*), sobreposto por um modelo de regressão linear. Com isso podemos observar tendências sobre os dados, conforme:

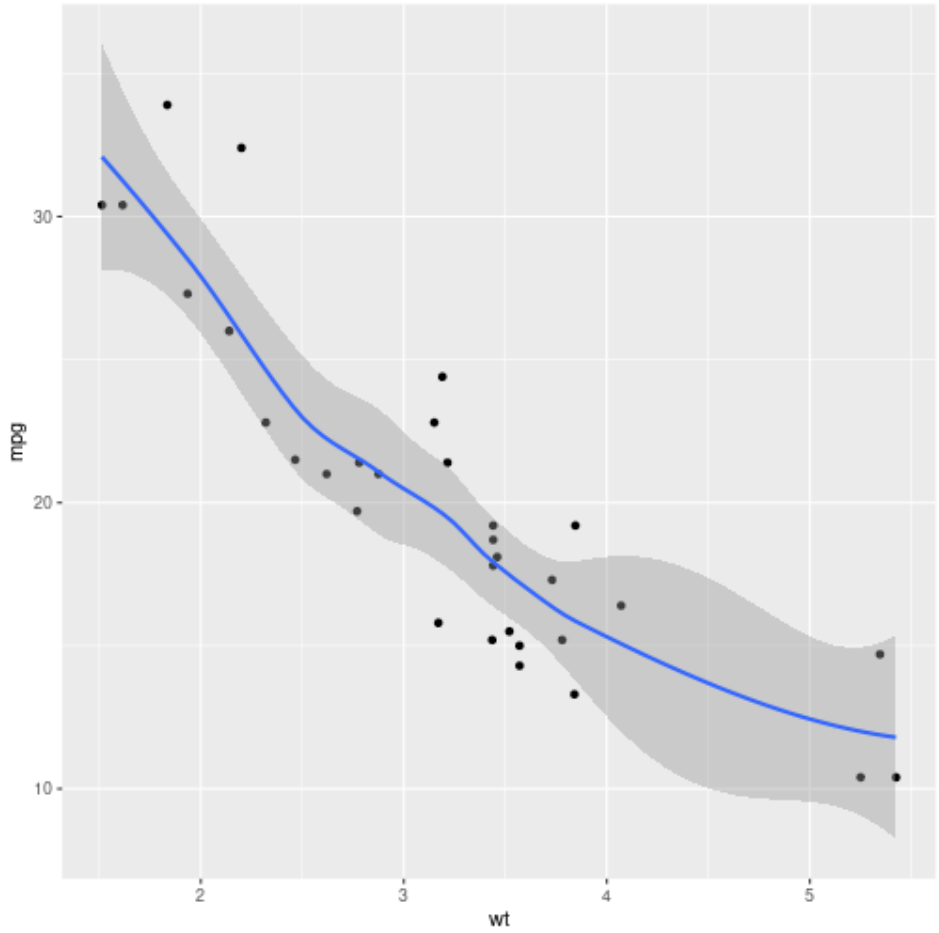
```
ggplot(mtcars, aes(x = wt, y = mpg)) +
  geom_point() +
  geom_smooth(method = lm)
```



Visualização de dados - ggplot2

Vamos adicionar camadas para criar um gráfico de dispersão (*scatter plot*), sobreposto por um modelo de regressão polinomial. Com isso podemos observar tendências sobre os dados, conforme:

```
ggplot(mtcars, aes(x = wt, y = mpg)) +
  geom_point() +
  geom_smooth(method = loess)
```



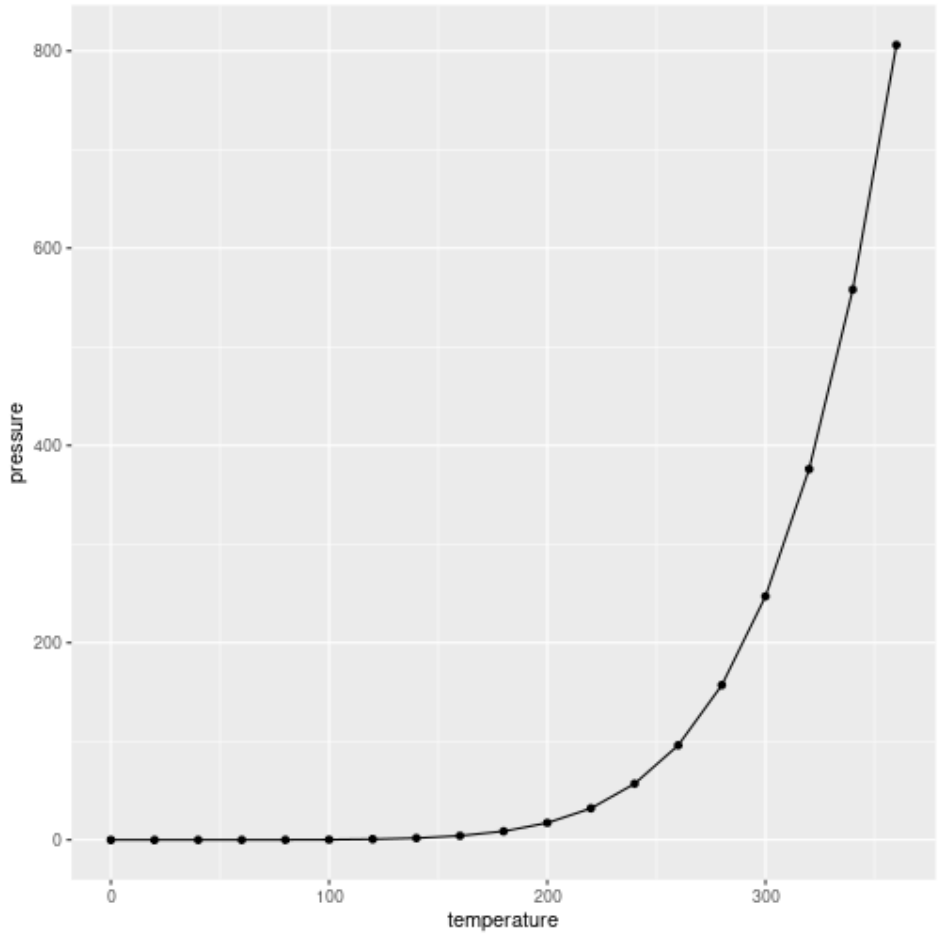
Visualização de dados - ggplot2

Vamos adicionar duas camadas para criar um gráfico de linhas. Estes tipicamente mostram as mudanças de uma variável em relação a outra.

```
ggplot(pressure, aes(x = temperature,
                     y = pressure)) +
  geom_line() +
  geom_point()
```

O correspondente no pacote **base** é:

```
plot(pressure$temperature,
     pressure$pressure, type = "l")
points(pressure$temperature,
       pressure$pressure)
```



Você consegue distinguir as camadas do gráfico?

Visualização de dados - ggplot2

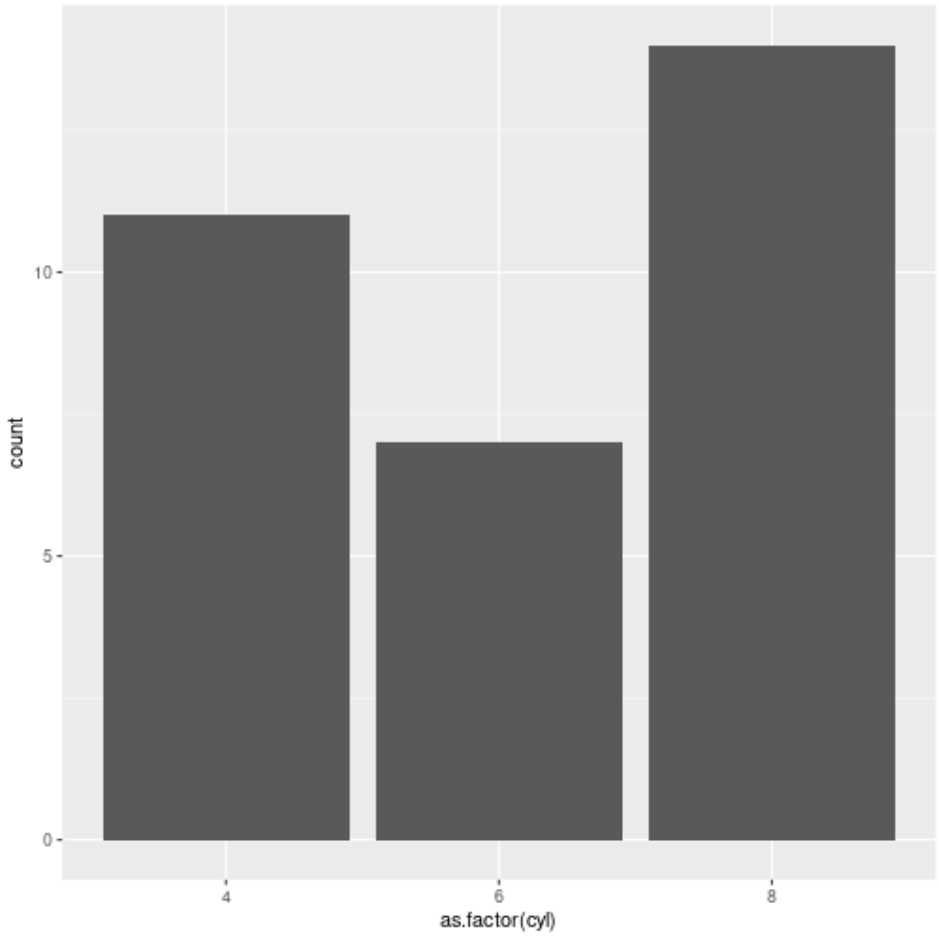
Vamos produzir um gráfico de barras. Estes representam a contagem dos elementos de uma variável, com o eixo x na forma discreta. Obs: o eixo x como contínuo representa tipicamente um histograma.

```
# x como variável contínua
ggplot(mtcars, aes(x = cyl)) +
  geom_bar()
```

```
# x como variável discreta (aqui categórica)
ggplot(mtcars, aes(x = as.factor(cyl))) +
  geom_bar()
```

O correspondente no pacote **base** é:

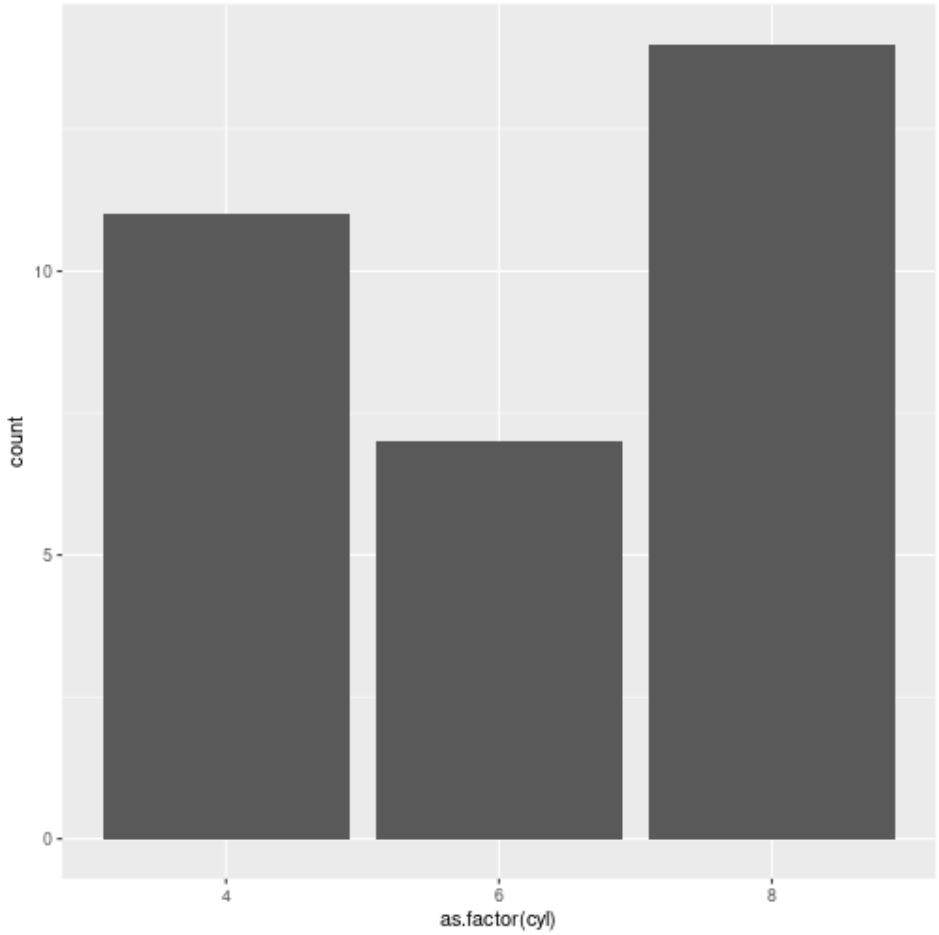
```
df <- as.data.frame(table(mtcars$cyl))
barplot(df$count, names.arg = df$cyl)
```



Visualização de dados - ggplot2

Podemos produzir o mesmo gráfico de barras usando geom_col(). Porém, este, requer as variáveis correspondentes nos eixos x e y. Resolveremos isso com summarise() e n().

```
mtcars |>
  summarise(count = n(), .by = cyl) |>
  ggplot(aes(x = as.factor(cyl),
             y = count)) +
    geom_col()
```



Visualização de dados - ggplot2

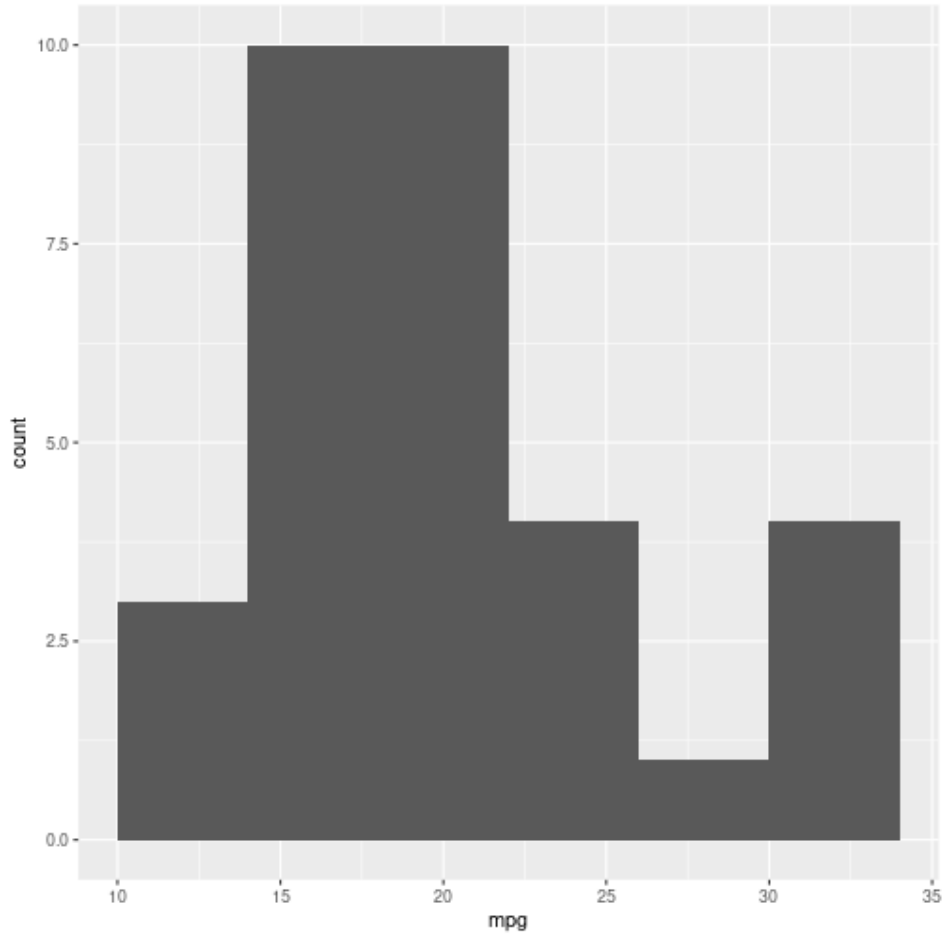
Vamos produzir um histograma (distribuição de frequências). Estes representam a contagem dos elementos de uma variável e sua distribuição.

```
ggplot(mtcars, aes(x = mpg)) +
  geom_histogram(binwidth = 4)
```

O correspondente no pacote **base** é:

```
hist(mtcars$mpg, breaks = 10)
```

Obs: **binwidth** e **breaks** são parâmetros que usam métricas diferentes para o particionamento das barras. As funções geom_histogram() e hist() são correspondentes apenas para o tipo de gráfico.



Visualização de dados - ggplot2

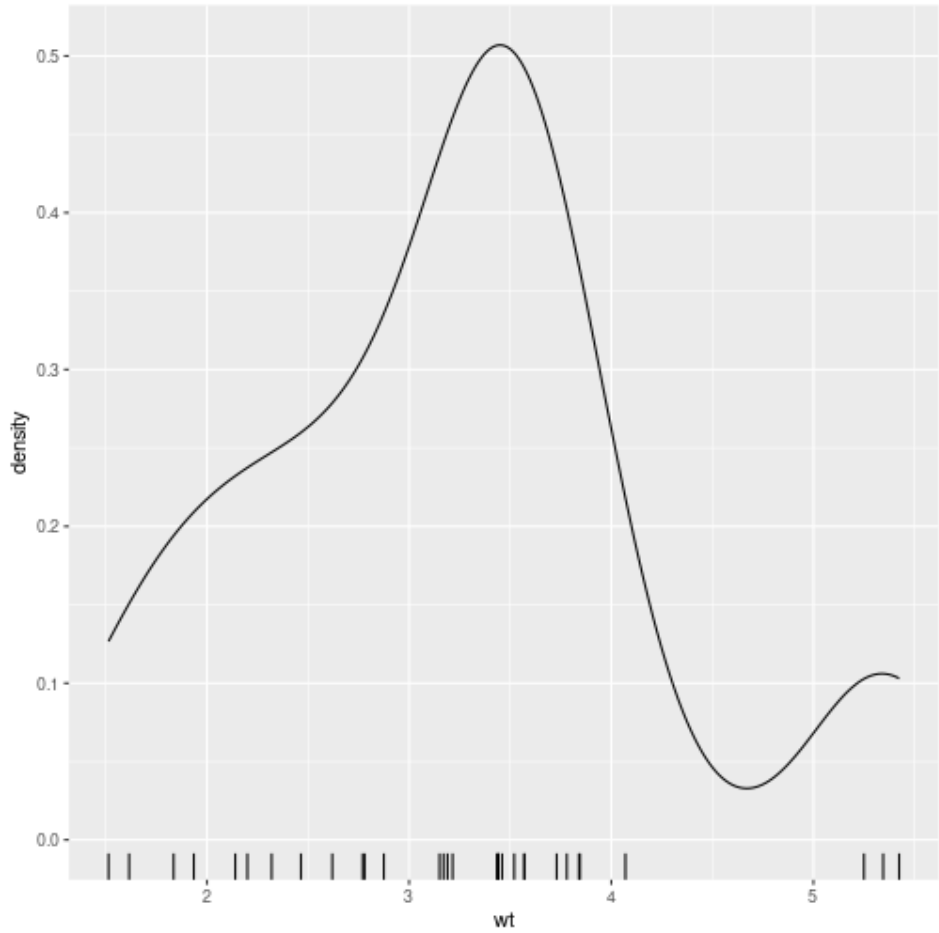
Vamos produzir um gráfico de densidade. Estes mostram a distribuição de uma variável númerica.

```
ggplot(mtcars, aes(x = wt)) +
  geom_density() + geom_rug()
```

O correspondente no pacote **base** é:

```
plot(density(mtcars$wt))
```

Obs: Também existem diferenças entre os gráficos de densidade gerados por geom_density() e density().



Visualização de dados - ggplot2

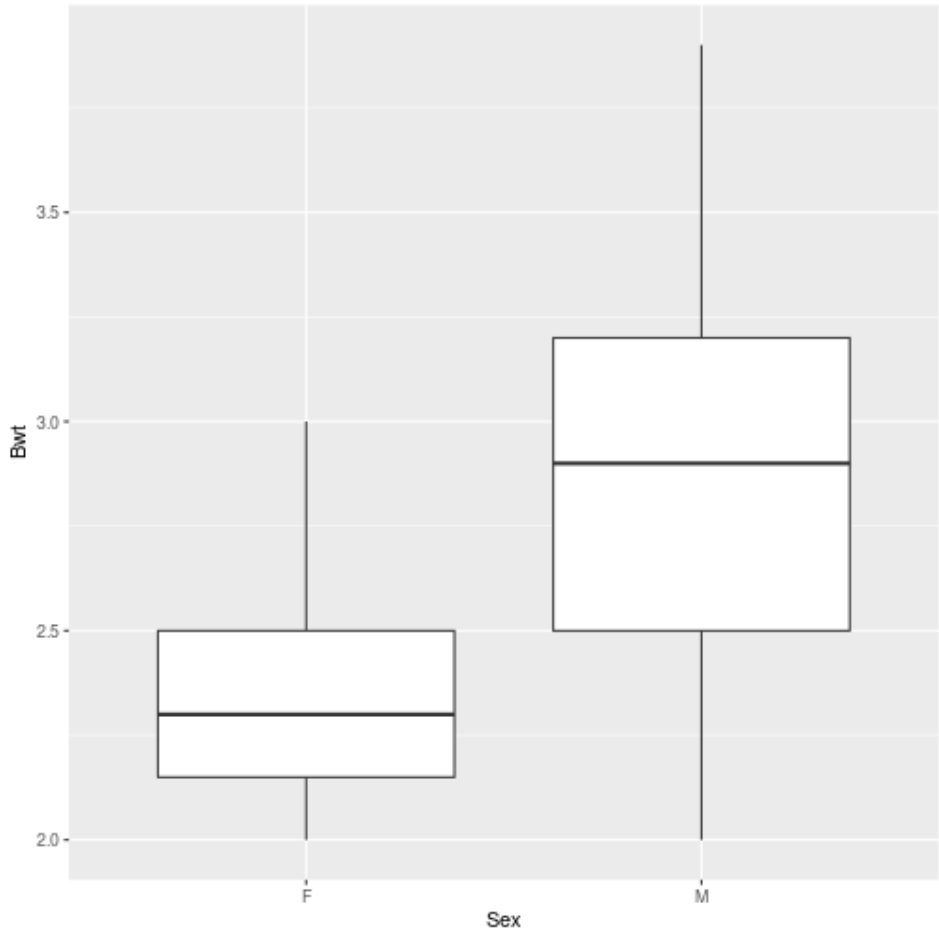
Vamos produzir um diagrama de caixas (*Box Plot*). *Estes representam conjuntos de dados por meio de diferentes medidas descritivas.*

```
ggplot(cats, aes(x = Sex, y = Bwt)) +
  geom_boxplot()
```

O correspondente no pacote ***base*** *é:*

```
boxplot(cats$Sex, cats$Bwt)
```

Como interpretar um diagrama de caixas? *O seguinte video pode ajudar.*



Visualização de dados - ggplot2

Quantos tipos de plots existem?

- Descubra algumas opções digitando geom_ ou stat_ e presionando **TAB**.
- Consulte os exemplos na documentação de cada função do tipo geom_ ou stat_.
- O livro R Graphics Cookbook apresenta exemplos de construção de gráficos no R - basicamente usando o **ggplot2** - de forma simples e prática para consultas rápidas.
- Veja exemplos em R Graph Gallery.
- O livro ggplot2: Elegant Graphics for Data Analysis apresenta uma abordagem mais aprofundada do pacote **ggplot2** e seu uso.

Exercícios

1. Construa um gráfico de dispersão relacionando quaisquer variáveis do dataframe **mtcars**.
2. Construa um gráfico de linhas relacionando as colunas **speed** e **dist** do dataframe **cars**.

Visualização de dados - ggplot2 - aesthetics

Aesthetics. Representa o mapeamento das propriedades físicas dos dados (tipicamente valores numéricos ou categóricos) em aspectos visuais - estéticos - do gráfico (ex: forma, cor e tamanho de elementos geométricos como pontos, linha e barras). Seguiremos por exemplos de **aesthetics** usando o dataframe **mpg** que contém dados de consumo de combustível de diferentes modelos/classes de carro.

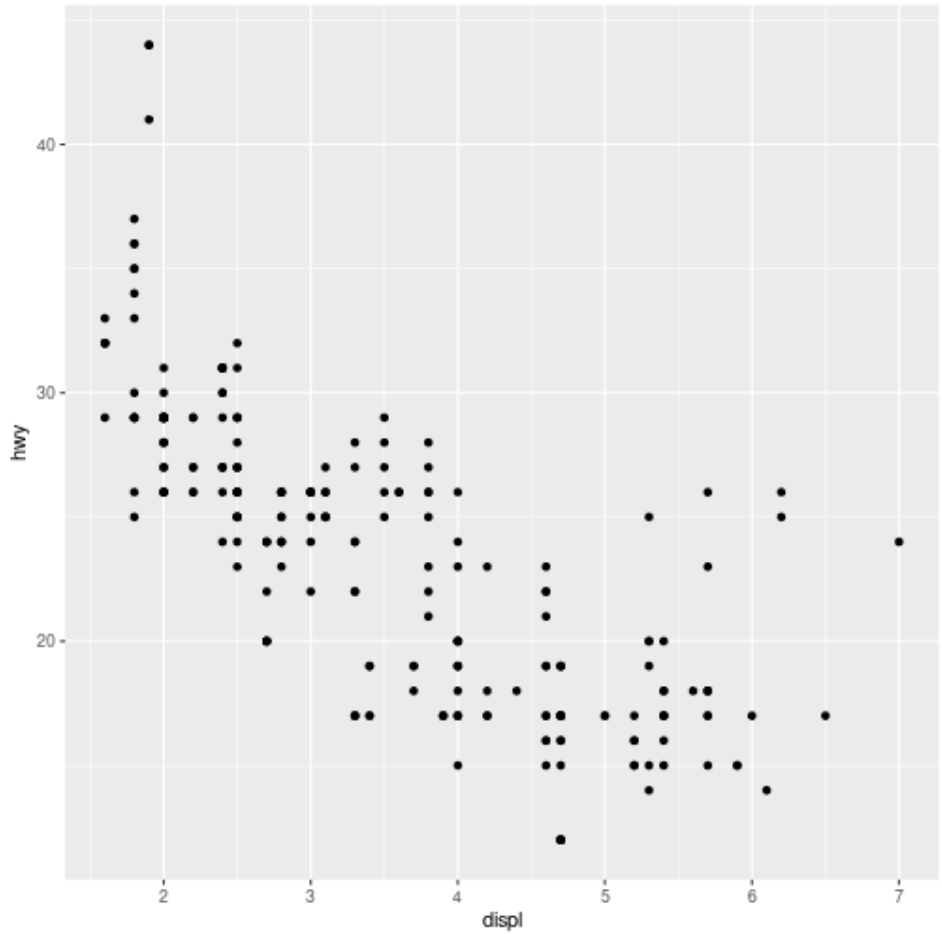
```
glimpse(mpg)
```

```
## Rows: 234
## Columns: 11
## $ manufacturer <chr> "audi", "audi", "audi", "audi", "audi", "audi", "audi", "…
## $ model        <chr> "a4", "a4", "a4", "a4", "a4", "a4", "a4", "a4 quattro", "…
## $ displ        <dbl> 1.8, 1.8, 2.0, 2.0, 2.8, 2.8, 3.1, 1.8, 1.8, 2.0, 2.0, 2.…
## $ year         <int> 1999, 1999, 2008, 2008, 1999, 1999, 2008, 1999, 1999, 200…
## $ cyl          <int> 4, 4, 4, 4, 6, 6, 6, 4, 4, 4, 4, 6, 6, 6, 6, 6, 6, 8, 8, …
## $ trans        <chr> "auto(l5)", "manual(m5)", "manual(m6)", "auto(av)", "auto…
## $ drv          <chr> "f", "f", "f", "f", "f", "f", "f", "4", "4", "4", "4", "4…
## $ cty          <int> 18, 21, 20, 21, 16, 18, 18, 18, 16, 20, 19, 15, 17, 17, 1…
## $ hwy          <int> 29, 29, 31, 30, 26, 26, 27, 26, 25, 28, 27, 25, 25, 25, 2…
## $ fl           <chr> "p", "p", "p", "p", "p", "p", "p", "p", "p", "p", "p", "p…
## $ class        <chr> "compact", "compact", "compact", "compact", "compact", "c…
```

Visualização de dados - ggplot2 - aesthetics

Os parâmetros estéticos x e y mapeiam os valores de uma variável para localizações/aparências dos elementos geométricos no gráfico. Portanto, podemos criar um gráfico de dispersão para as colunas **displ** e **hwy** que indicam respectivamente as cilindradas do motor e o consumo de combustível dos carros em rodovias, conforme:

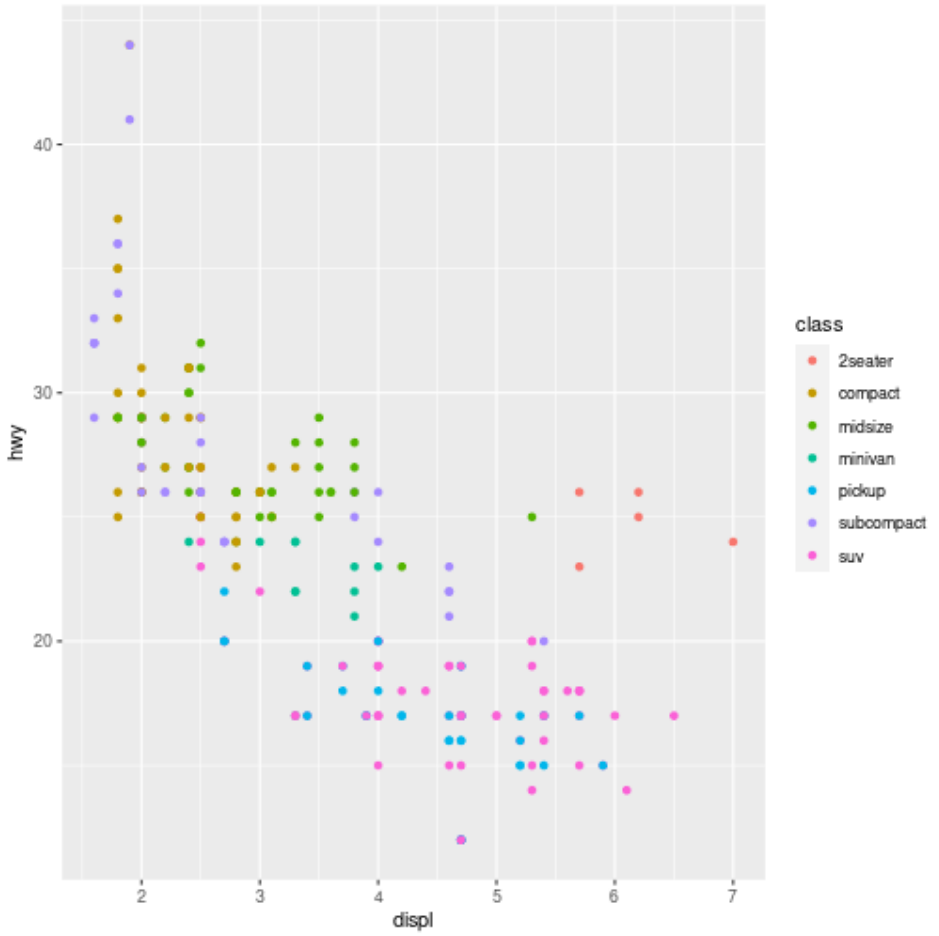
```
ggplot() +
  geom_point(
    data = mpg,
    mapping = aes(x = displ, y = hwy))
```



Visualização de dados - ggplot2 - aesthetics

Também podemos diferenciar as classes dos carros (coluna **class**) através de cores (parâmetro color ou colour), conforme:

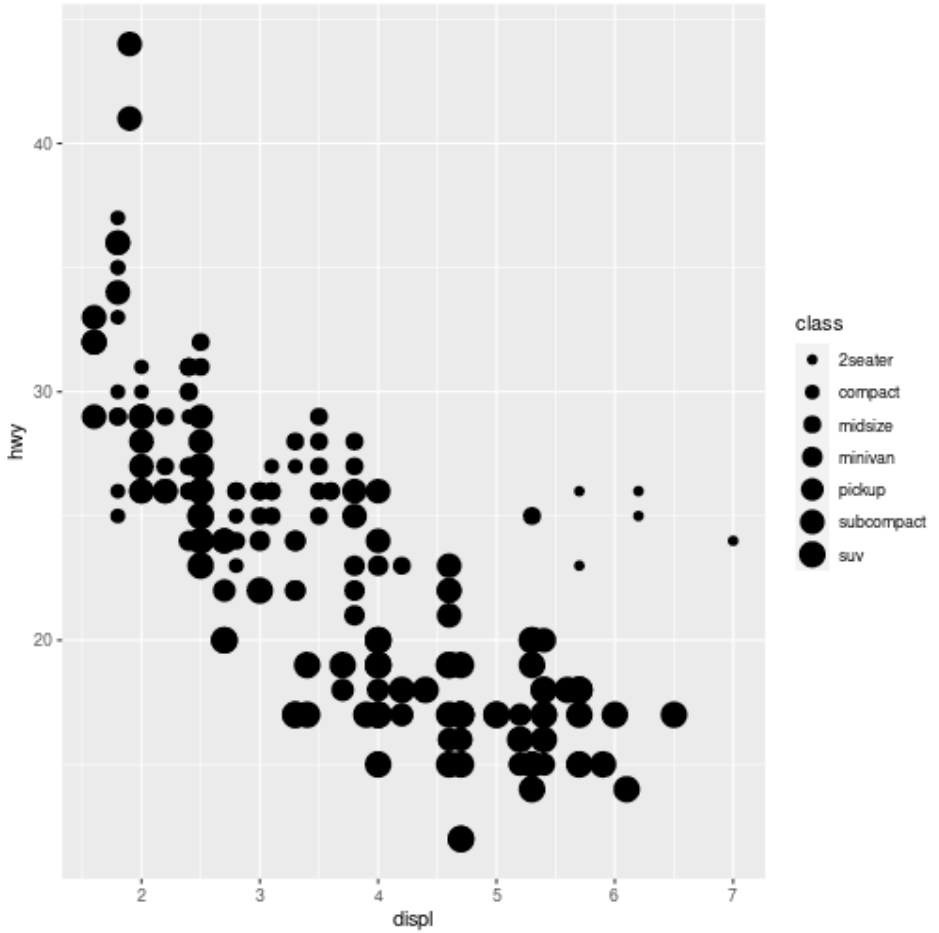
```
ggplot() +
  geom_point(
    data = mpg,
    mapping = aes(x = displ, y = hwy,
                  color = class)
  )
```



Visualização de dados - ggplot2 - aesthetics

Podemos diferenciar as classes dos carros por meio do tamanho (parâmetro size) dos pontos:

```
ggplot() +
  geom_point(
    data = mpg,
    mapping = aes(x = displ, y = hwy,
                  size = class)
  )
```

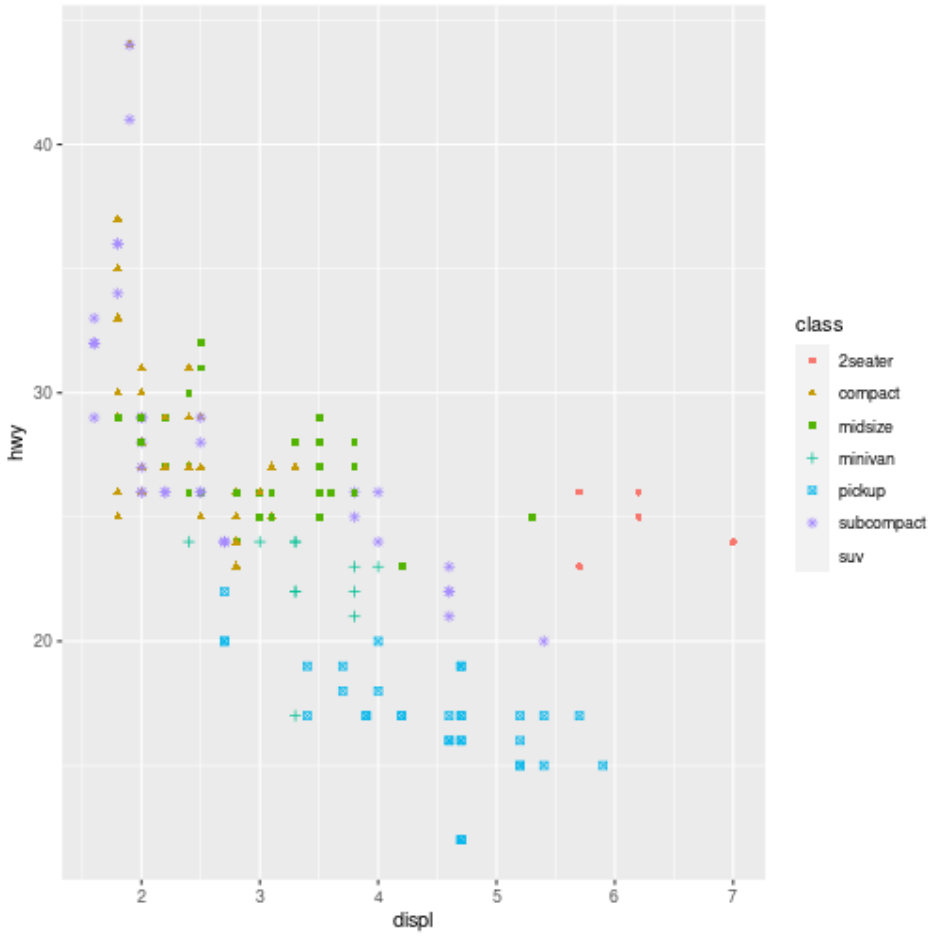


Perceba que usar size para representar variáveis discretas não é uma escolha sensata.

Visualização de dados - ggplot2 - aesthetics

Podemos distinguir as classes dos carros por meio de cores e formas dos pontos (parâmetro shape):

```
ggplot() +
  geom_point(
    data = mpg,
    mapping = aes(x = displ, y = hwy,
                  color = class,
                  shape = class)
  )
```

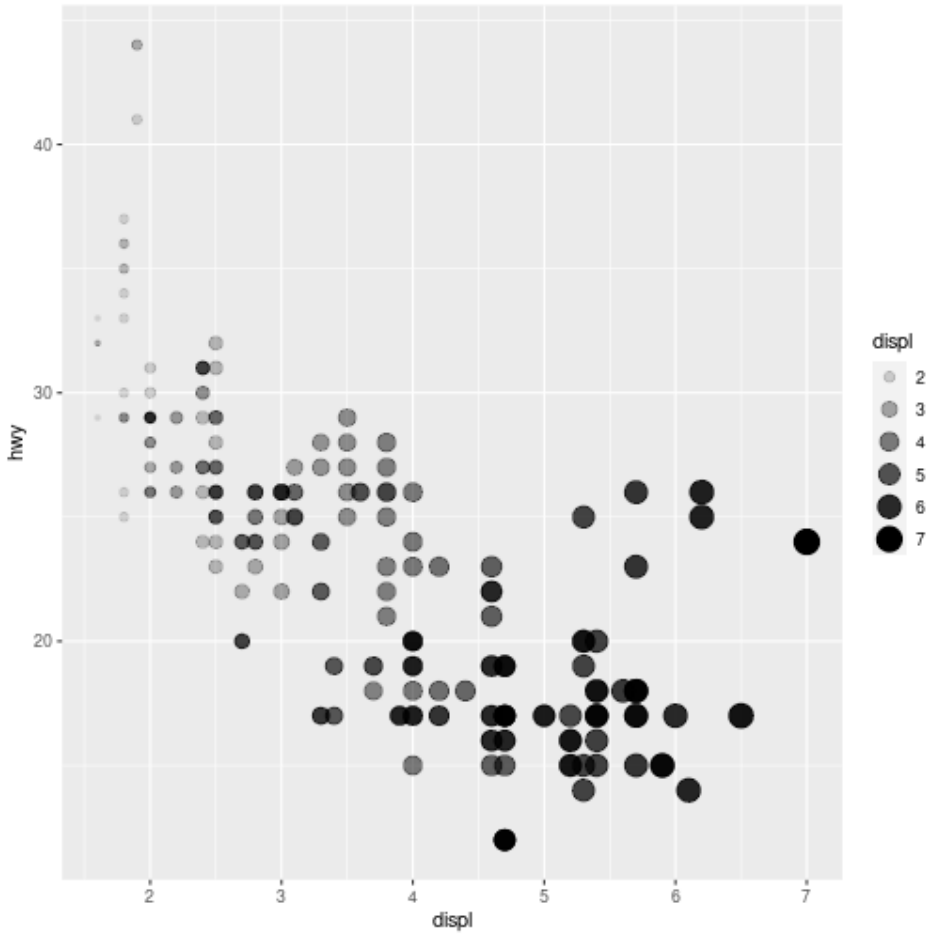


Perceba que a classe **suv** ficou sem ponto. Há um limite de 6 tipos de pontos para representação automatica com *shape*. Quantidades maiores ainda podem ser indicadas, desde que manualmente.

Visualização de dados - ggplot2 - aesthetics

Podemos distinguir as cilindradas dos carros por meio da transparência (parâmetro alpha) e tamanho dos pontos:

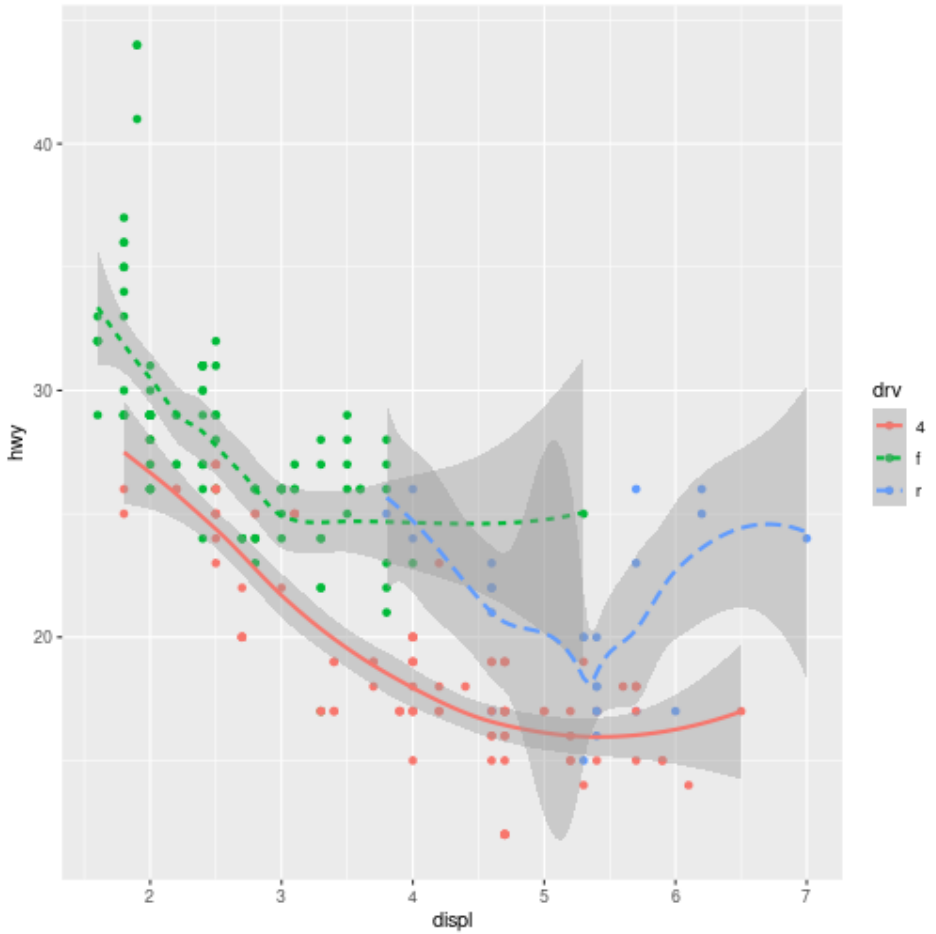
```
ggplot() +
  geom_point(
    data = mpg,
    mapping = aes(x = displ, y = hwy,
                  size = displ,
                  alpha = displ)
  )
```



Visualização de dados - ggplot2 - aesthetics

Podemos sobrepor um modelo de regressão polinomial para cada divisão categórica de uma variável através do parâmetro estético group, conforme:

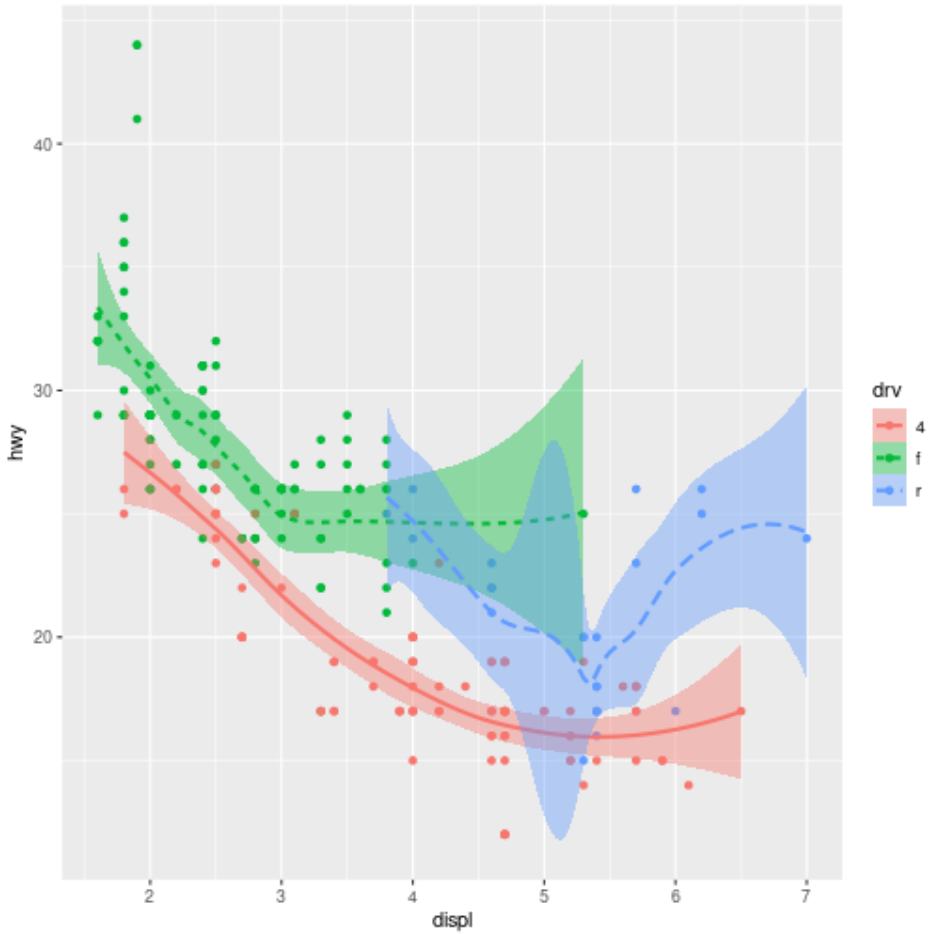
```
ggplot(
  data = mpg,
  mapping = aes(x = displ, y = hwy,
                color = drv)) +
  geom_point() +
  geom_smooth(
    mapping = aes(linetype = drv,
                  group = drv))
```



Visualização de dados - ggplot2 - aesthetics

Algumas funções como geom_smooth() possuem aspectos visuais de preenchimento nos elementos geométricos que geram. Portanto, podemos distinguir os modelos de regressão usando o parâmetro estético fill que define o preenchimento do intervalo de confiança, conforme:

```
ggplot(
  data = mpg,
  mapping = aes(x = displ, y = hwy,
                color = drv)) +
  geom_point() +
  geom_smooth(
    mapping = aes(linetype = drv,
                  group = drv,
                  fill = drv))
```



Visualização de dados - ggplot2 - aesthetics

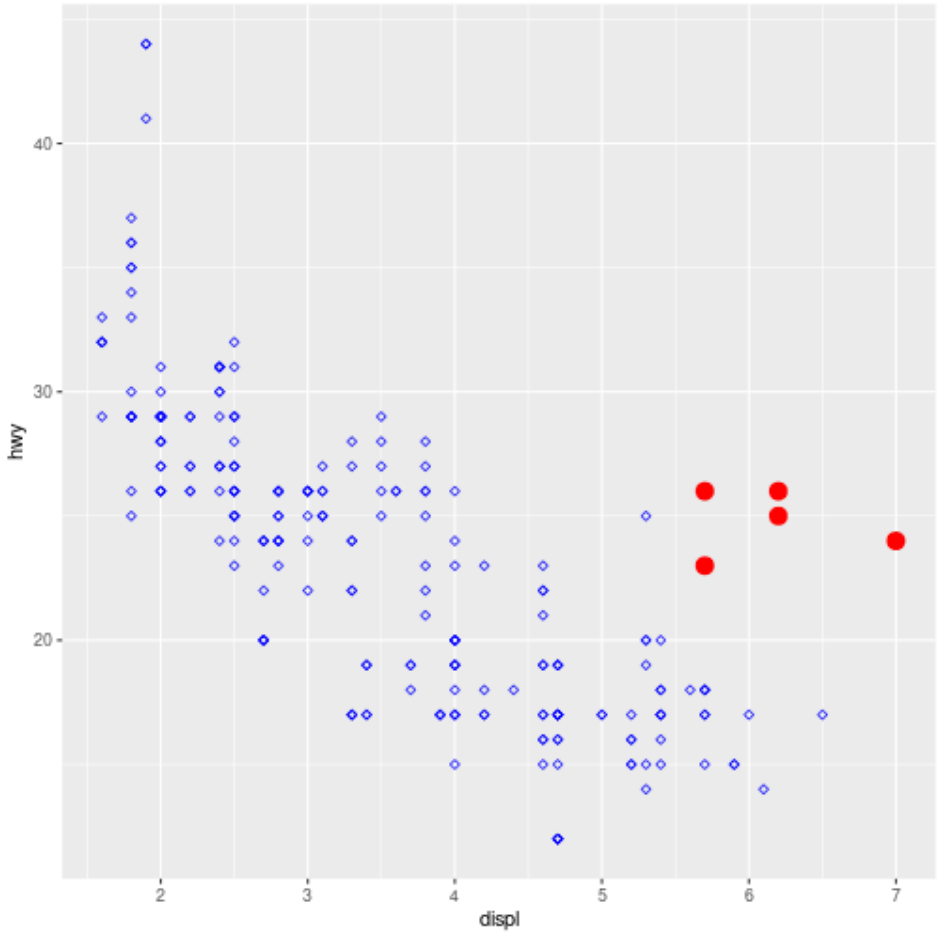
Vimos alguns parâmetros estéticos: x, y, color, size, shape, alpha, group e fill. Uma vez definidos os parâmetros estéticos e gerado o gráfico, o ggplot2 automaticamente:

- Seleciona a transformação estatística aplicada aos dados - por padrão é definida como identity (1:1) pelo componente gráfico **stat**;
- Seleciona o posicionamento dos elementos geométricos - por padrão é definida como identity pelo componente gráfico **position**;
- Gera a legenda com labels caso tenham sido mapeados valores de variáveis em níveis categóricos;
- Seleciona a escala apropriada para os eixos x e y - ex: contínua, discreta;
- Gera labels e marcações nos eixos x e y em conformidade com a escala.

Visualização de dados - ggplot2 - aesthetics

Podemos informar manualmente (fixar) alguns aspectos visuais dos elementos geométricos. Basta dispor os parâmetros estéticos fora da função aes(), conforme:

```
ggplot() +
  geom_point(
    data = mpg,
    mapping = aes(x = displ, y = hwy),
    color = "blue",
    shape = 23) +
  geom_point(
    data = filter(mpg, class == "2seater"),
    mapping = aes(x = displ, y = hwy),
    color = "red",
    size = 4 )
```

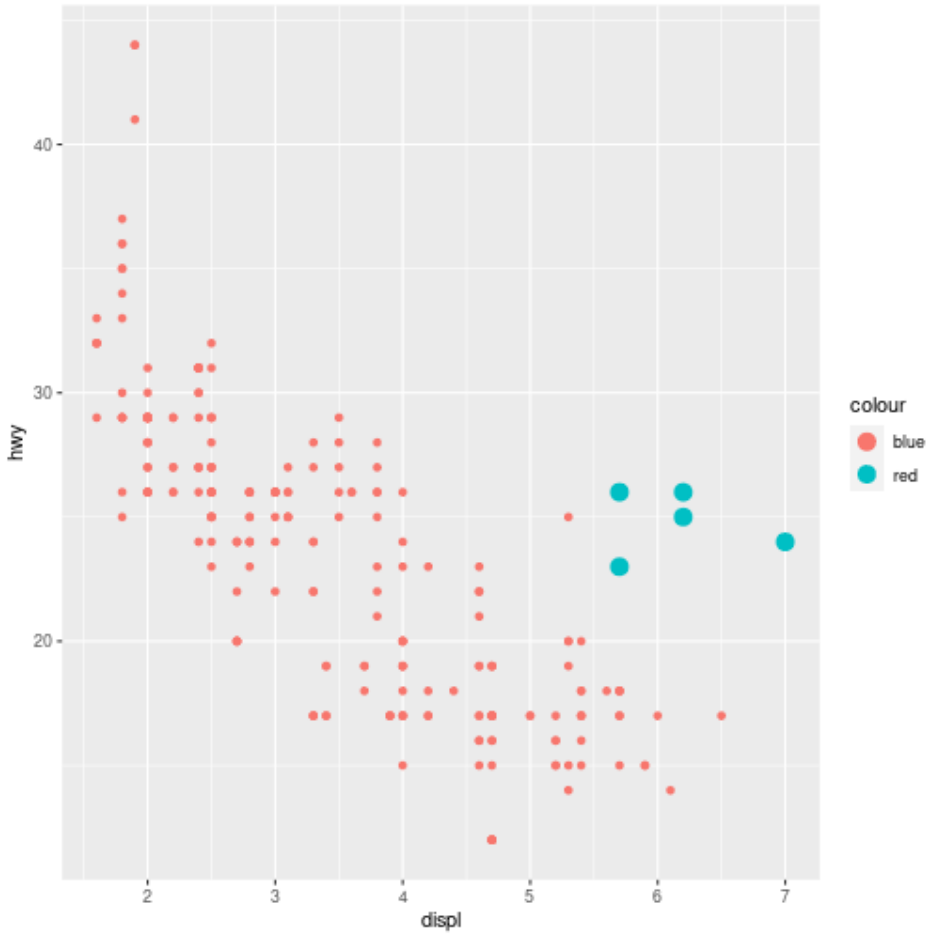


Visualização de dados - ggplot2 - aesthetics

Atenção! Conforme vimos, parâmetros estéticos mapeiam os valores das variáveis em aspectos visuais. Portanto, na seguinte situação:

```
ggplot() +
  geom_point(
    data = mpg,
    mapping = aes(x = displ, y = hwy,
                  color = "blue")) +
  geom_point(
    data = filter(mpg, class == "2seater"),
    mapping = aes(x = displ, y = hwy,
                  color = "red"),
    size = 4)
```

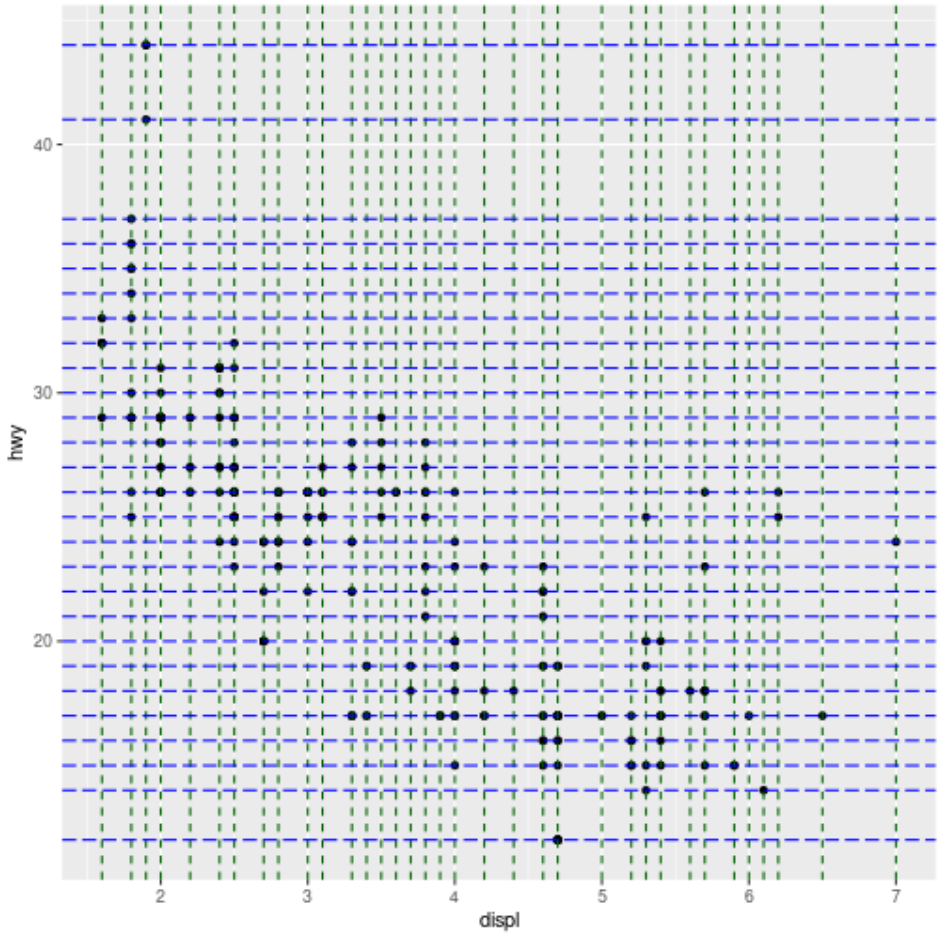
As cores atribuídas no parâmetro color são assumidas como categorias (labels) para os pontos e não como aspecto visual.



Visualização de dados - ggplot2 - aesthetics

Alguns elementos geométricos possuem parâmetros estéticos próprios, conforme:

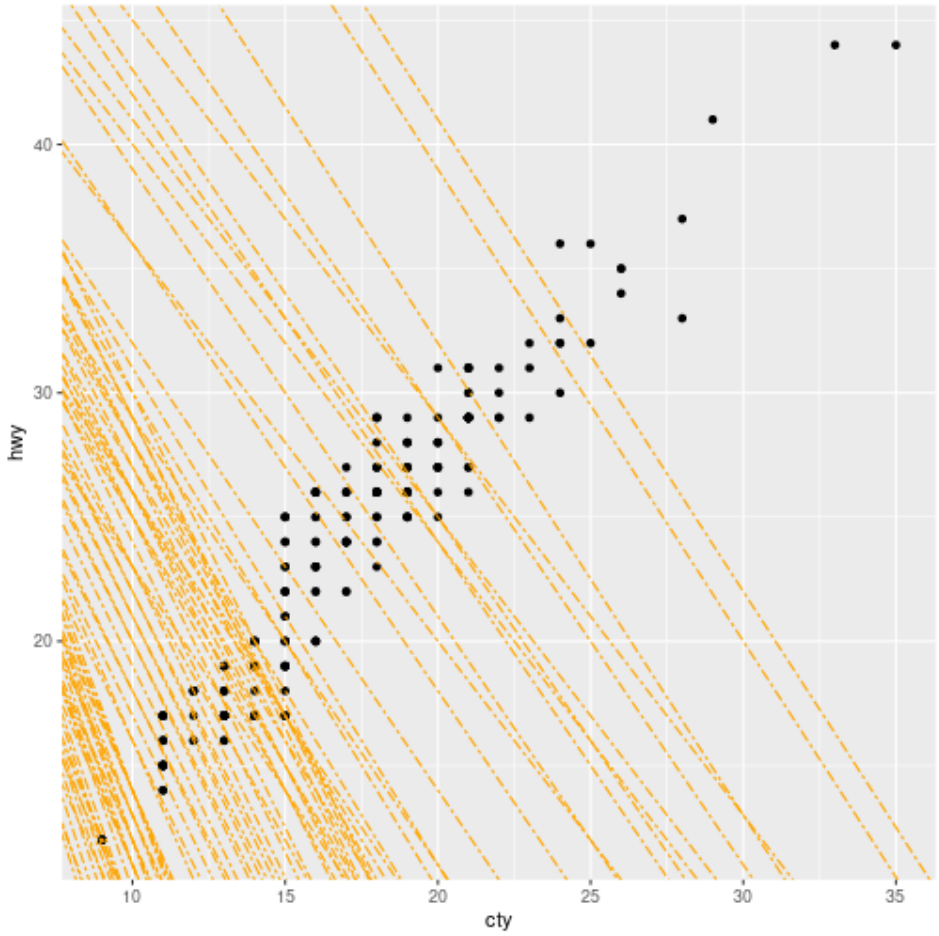
```
ggplot(data = mpg) +
  geom_point(
    mapping = aes(x = displ, y = hwy)
  ) +
  geom_vline(
    mapping = aes(xintercept = displ),
    color = "darkgreen",
    linetype = "dashed"
  ) +
  geom_hline(
    mapping = aes(yintercept = hwy),
    color = "blue",
    linetype = "longdash"
  )
```



Visualização de dados - ggplot2 - aesthetics

Alguns elementos geométricos possuem parâmetros estéticos próprios, conforme:

```
ggplot(data = mpg) +
  geom_point(
    mapping = aes(x = cty, y = hwy)) +
  geom_abline(
    mapping = aes(slope = -displ,
                  intercept = hwy + cty),
    color = "orange",
    linetype = "twodash"
  )
```

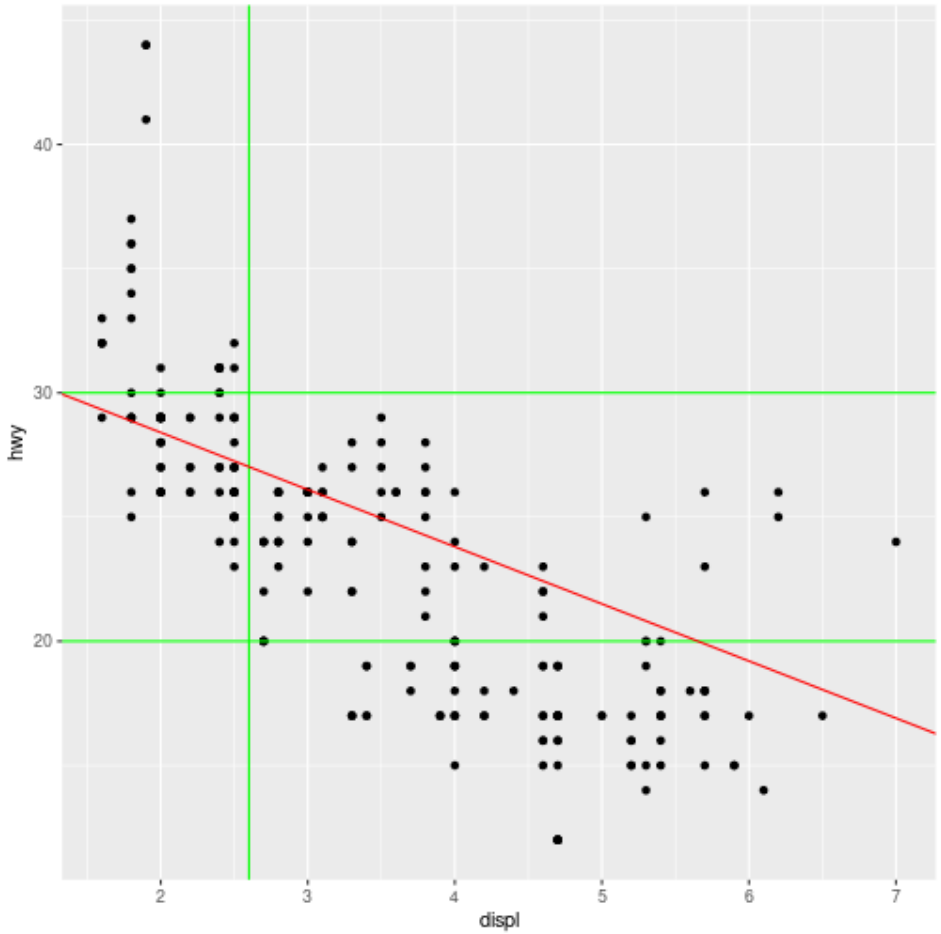


Visualização de dados - ggplot2 - aesthetics

Perceba que os parâmetros estéticos usados anteriormente tem comportamento diferente quando fixados manualmente fora de aes(), pois são tratados como parâmetros das próprias funções. Observe:

```
ggplot() +
  geom_point(
    data = mpg,
    mapping = aes(x = displ, y = hwy)) +
  geom_vline(xintercept = 2.6,
             color = "green") +
  geom_hline(yintercept = c(20, 30),
             color = "green") +
  geom_abline(slope = -2.3, intercept = 33,
              color =  "red")
```

Porém, xintercept, yintercept, slope e intercept também são parâmetros das funções acima.



Visualização de dados - ggplot2 - aesthetics

Resumo:

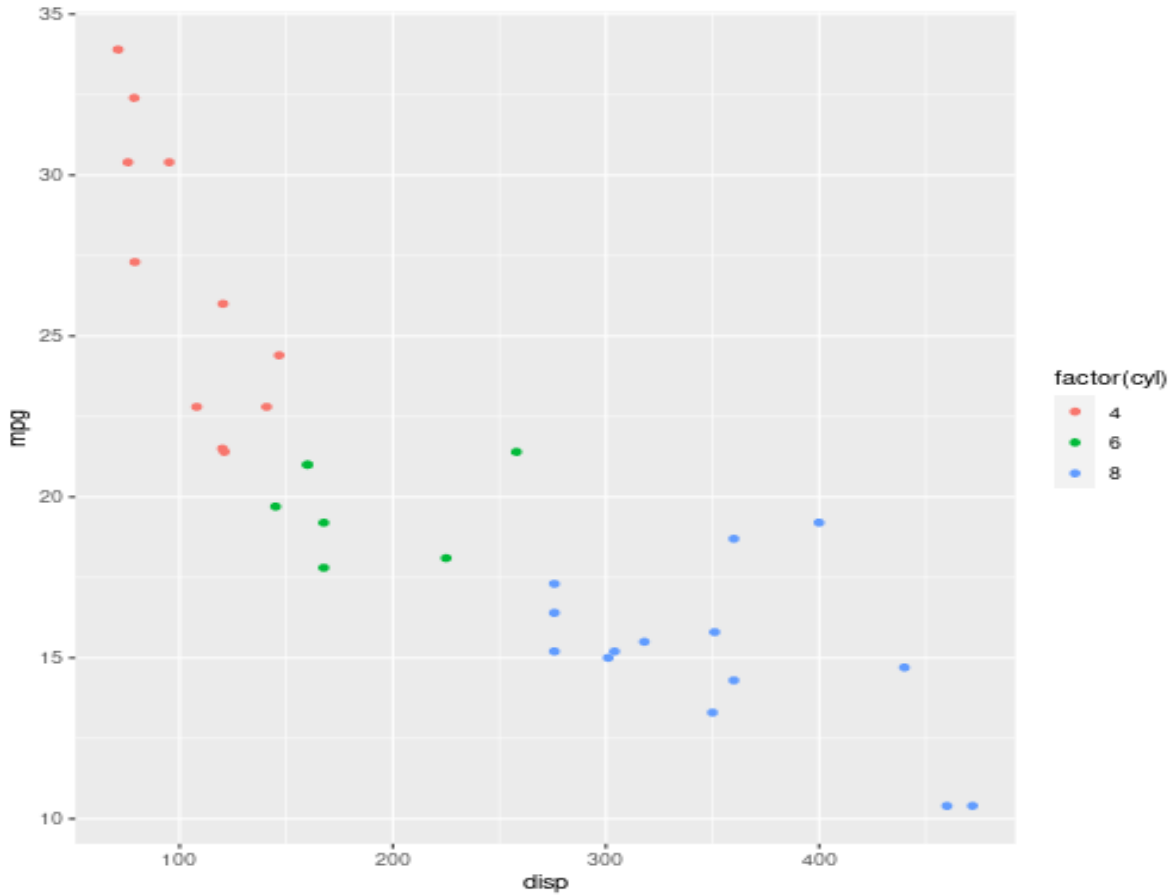
- Parâmetros estéticos permitem mapear atributos físicos dos dados em aspectos visuais.
- Quando os parâmetros estéticos são usados fora de aes() são tratados como parâmetros da própria função geom_ acionada. Assim, os aspectos visuais assumem os valores fixos informados.

Quais são os parâmetros estéticos existentes para cada elemento geométrico?

- Consulte a lista de elementos estéticos na documentação de cada função (ex: ?geom_line).
- Consulte materiais online como esse Vignette.

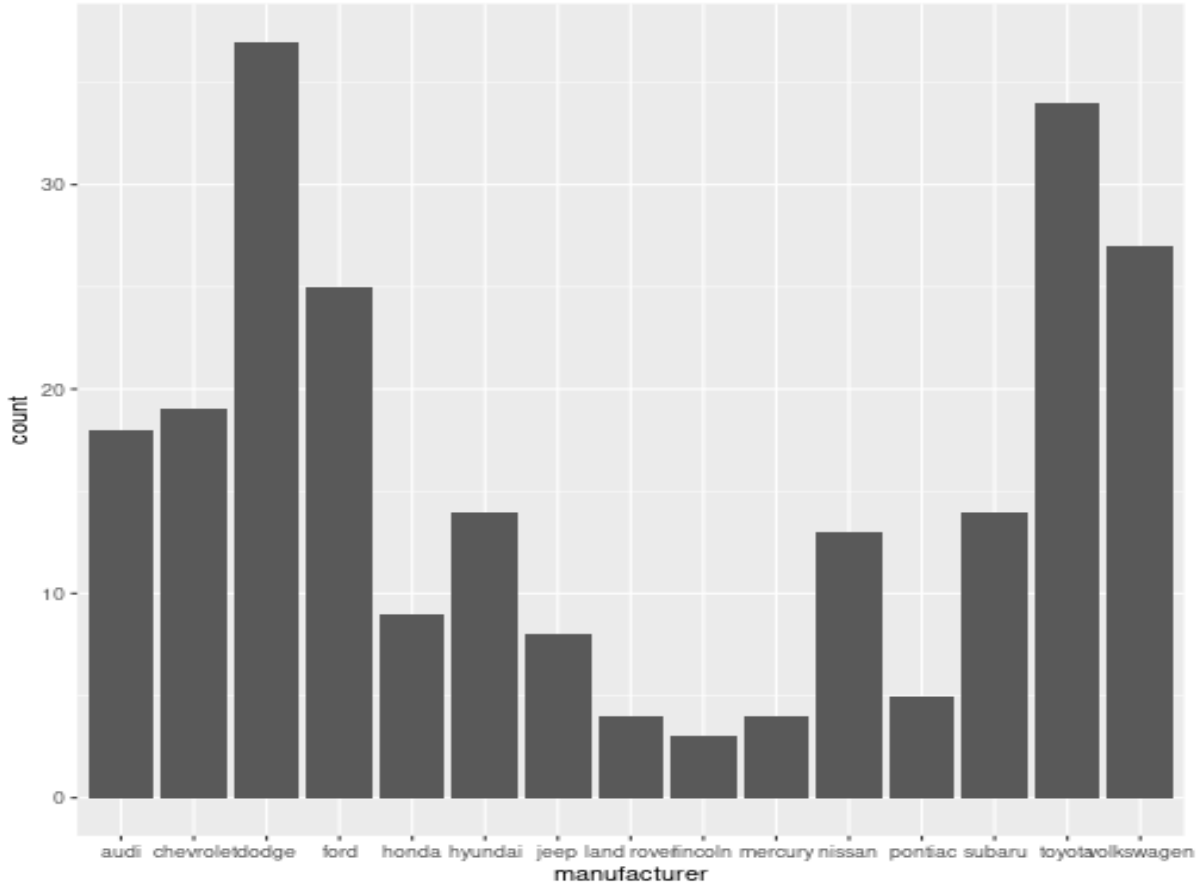
Exercícios

(1) Usando o dataframe **mtcars**, construa o seguinte gráfico:



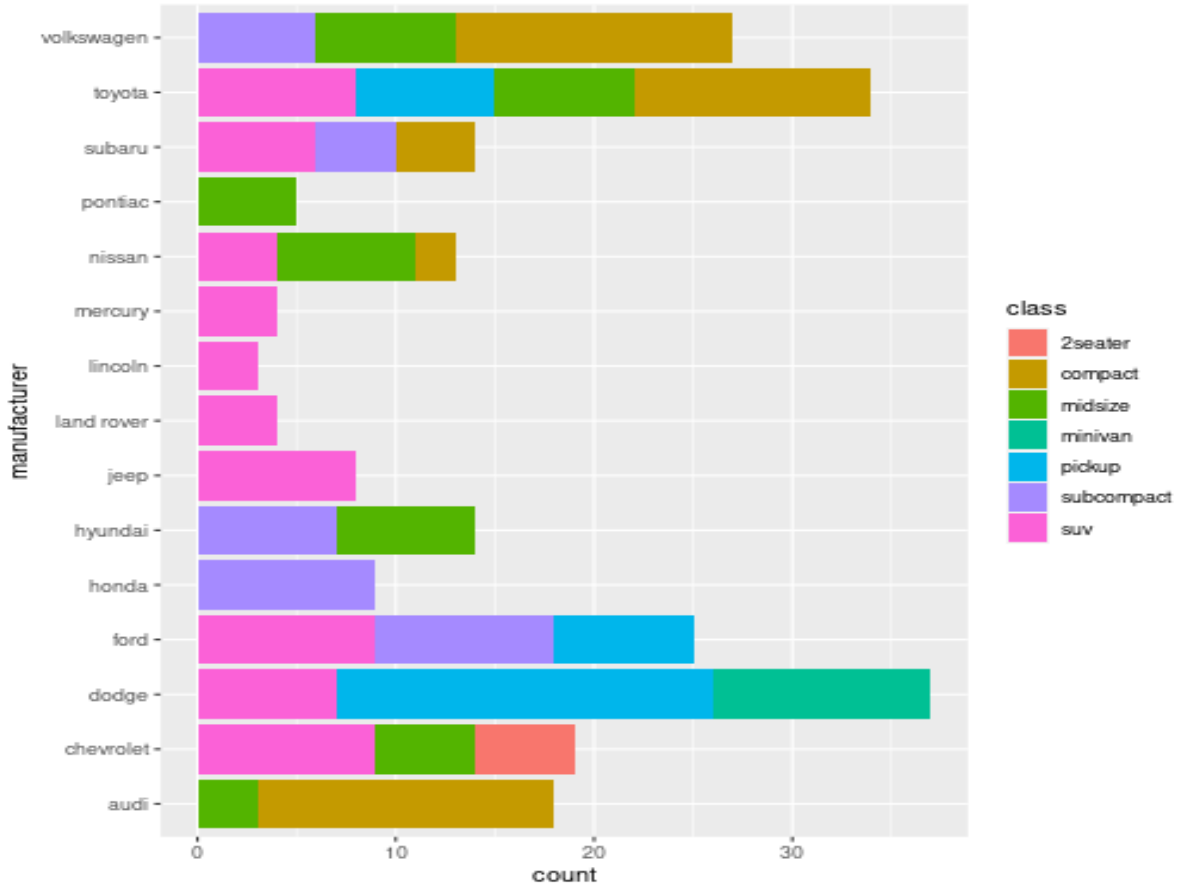
Exercícios

(2) Usando o dataframe **mpg**, gere um gráfico para contar os modelos de carro por fabricante, conforme:



Exercícios

(3) Considerando o gráfico do exercício anterior, diferencie os modelos e produza o seguinte gráfico:



Visualização de dados - ggplot2 - geom

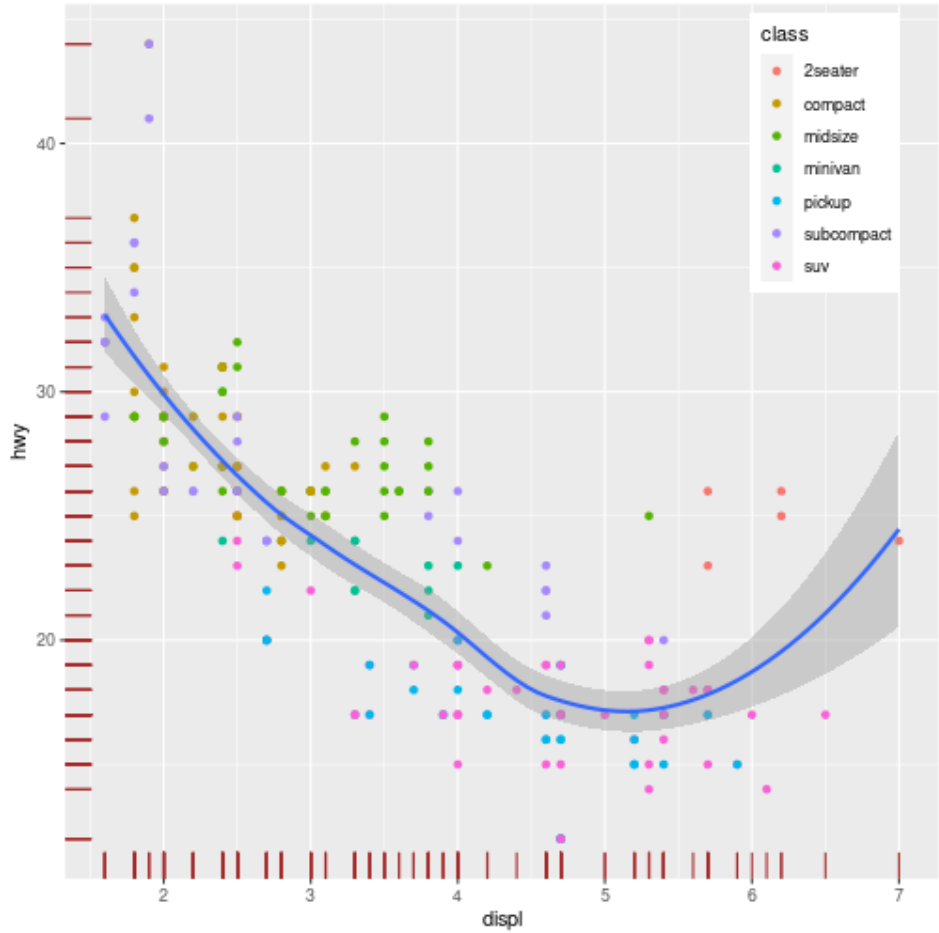
Geom. Define uma camada de representação visual dos dados ou simplificadamente um tipo de gráfico. Cada **geom** geralmente engloba: um conjunto de dados (**data**), atributos estéticos (**aes**), tranformações estatísticas dos dados (**stat**) e posicionamento (**position**) dos objetos no plano. Conforme vimos, várias dessas camadas (**geoms**) podem ser combinadas para formar um gráfico, assim como, existe uma família de funções do tipo geom_tipo-de-elemento-geometrico.

Descubra algumas opções digitando geom_ e presionando **TAB**.

Visualização de dados - ggplot2 - geom

Combinações variadas podem ser usadas:

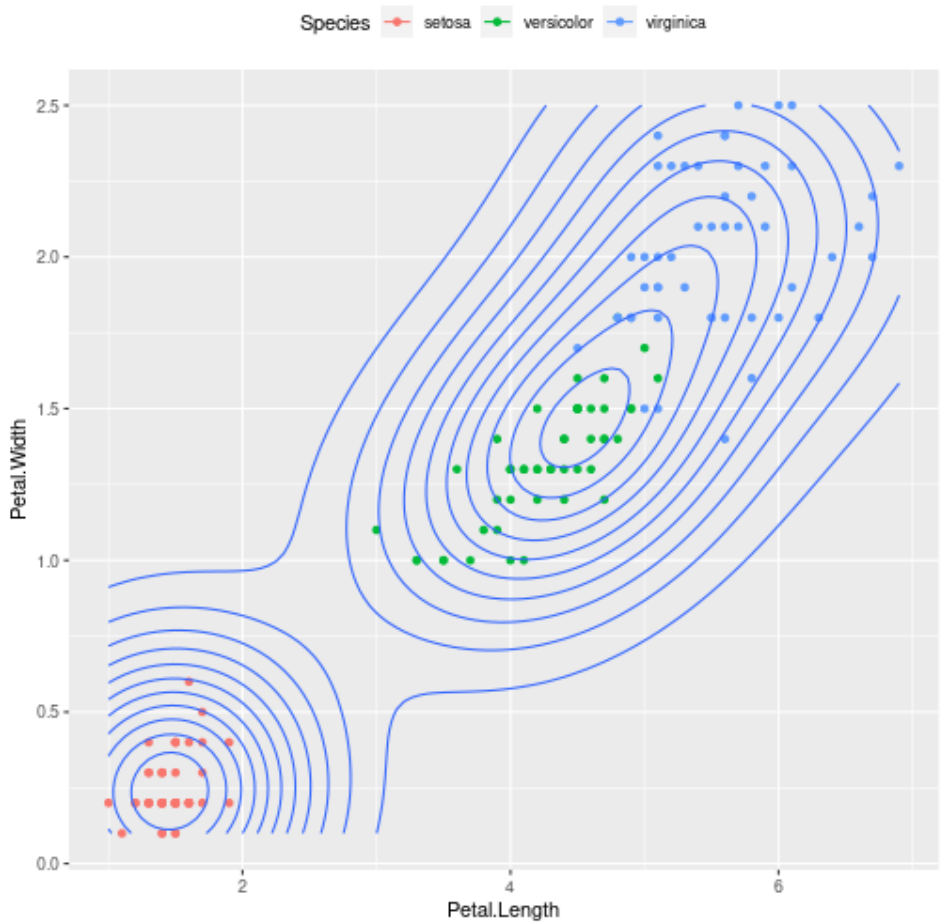
```
ggplot(
  data = mpg,
  mapping = aes(x = displ, y = hwy)) +
  geom_point(aes(color = class)) +
  geom_smooth() +
  geom_rug(color = "brown") +
  theme(legend.position = c(.87, .83))
```



Visualização de dados - ggplot2 - geom

Combinações variadas podem ser usadas:

```
ggplot(
  data = iris,
  mapping = aes(x = Petal.Length,
                y = Petal.Width)) +
  geom_point(
    mapping = aes(color = Species)) +
  geom_density2d() +
  theme(legend.position = "top")
```



Visualização de dados - ggplot2 - geom

Resumo:

- Geoms permitem vários elementos geométricos e tipos de gráficos.
- Também permitem certa liberdade criativa devido à possibilidade de combinar camadas.

Quais formas geométricas e tipos de gráficos existem? Há vasto material na internet. Vamos relembrar algumas opções:

- Consulte os exemplos na documentação de cada função do tipo geom_.
- O livro R Graphics Cookbook apresenta exemplos de construção de gráficos no R - basicamente usando o **ggplot2** - de forma simples e prática para consultas rápidas.
- Veja exemplos em R Graph Gallery.
- O livro ggplot2: Elegant Graphics for Data Analysis apresenta uma abordagem mais aprofundada do pacote **ggplot2** e seu uso.

Visualização de dados - ggplot2 - scale

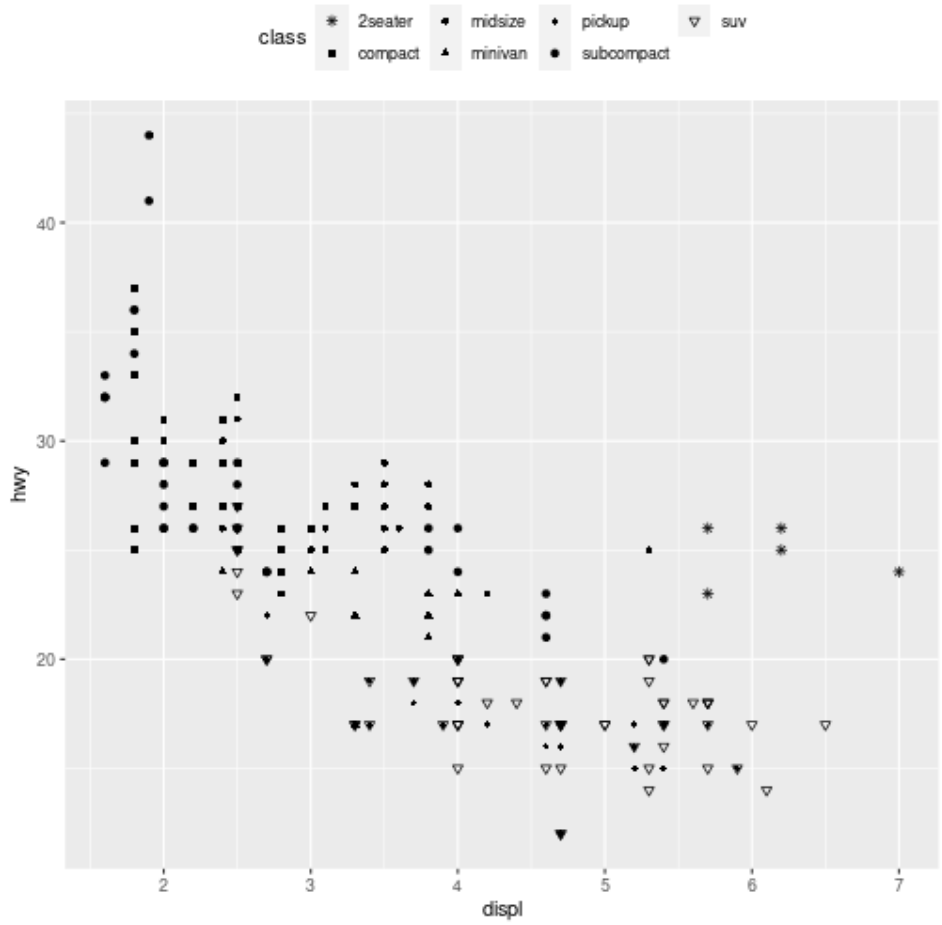
Scale. Esse componente gráfico permite o controle sobre aspectos visuais. Diversos atributos estéticos tem uma ou mais escalas correspondentes que formam uma família de funções do tipo scale_atributo-estetico. Cada uma dessas funções permite alterar aspectos visuais conforme seu atributo estético.

Descubra algumas opções digitando scale_ e presionando **TAB**.

Visualização de dados - ggplot2 - scale

Podemos usar o componente gráfico escala para alterar tipos de pontos:

```
ggplot() +
  geom_point(
    data = mpg,
    mapping = aes(x = displ, y = hwy,
                  shape = class)) +
  scale_shape_manual(values = c(8,15,16,17,
                                18,19,25)) +
  theme(legend.position = "top")
```

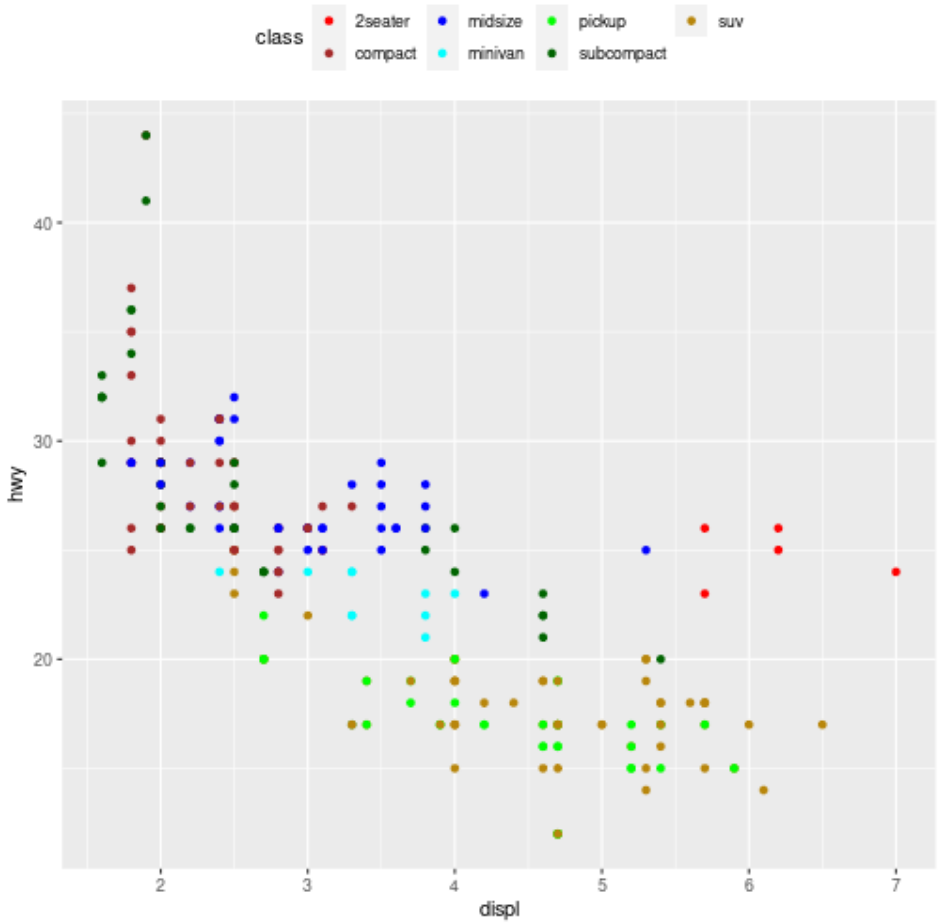


Quantos tipos de pontos ou linhas existem?
Consulte no material online.

Visualização de dados - ggplot2 - scale

Podemos usar o componente gráfico escala para alterar as cores:

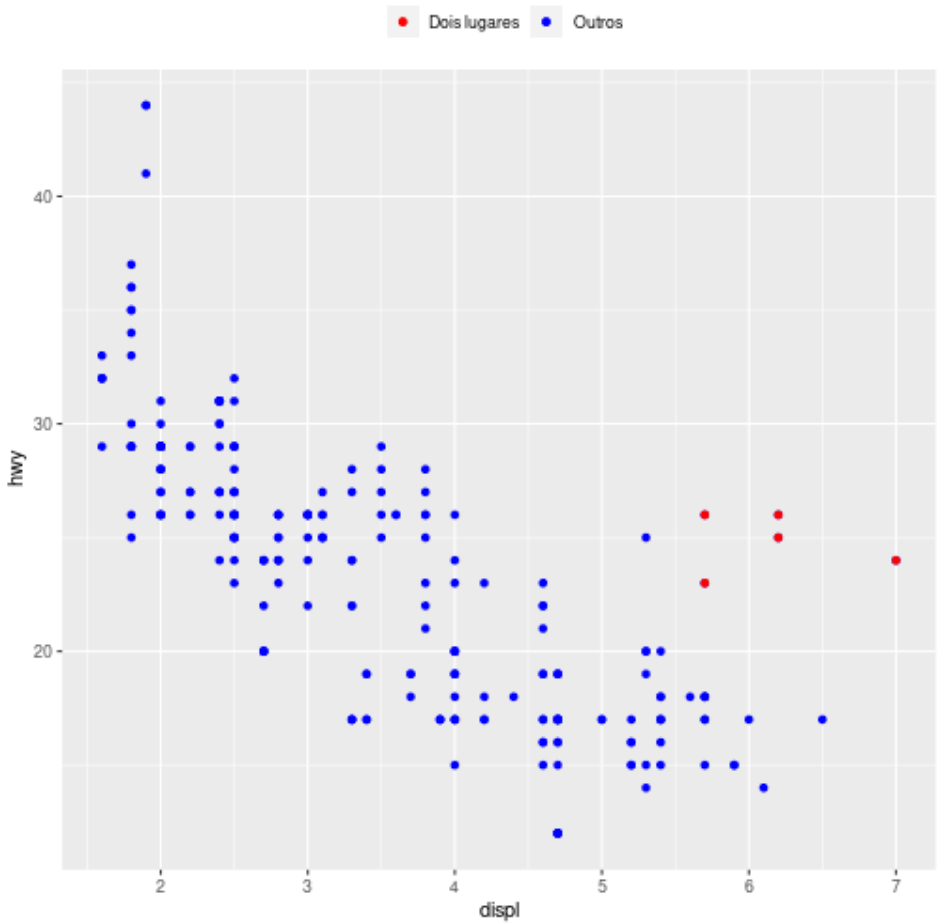
```
ggplot() +
  geom_point(
    data = mpg,
    mapping = aes(x = displ, y = hwy,
                  color = class)) +
  scale_color_manual(
    values = c("red","brown","blue","cyan",
               "green","darkgreen",
               "darkgoldenrod")) +
  theme(legend.position = "top")
```



Visualização de dados - ggplot2 - scale

Podemos especificar (fixar) um elemento categórico e atribuir cores específicas:

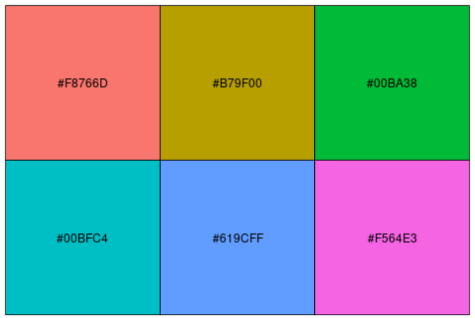
```
type <- c("Outros" = "blue",
          "Dois lugares" = "red")
two_seater <- filter(mpg,
                     class == "2seater")
ggplot() +
  geom_point(
    data = mpg,
    mapping = aes(x = displ, y = hwy,
                  color = "Outros")) +
  geom_point(data = two_seater,
    mapping = aes(x = displ, y = hwy,
                  color = "Dois lugares")) +
  scale_color_manual(values = type) +
  theme(legend.position = "top",
        legend.title = element_blank())
```



Visualização de dados - ggplot2 - scale

Quais as cores padrão do ggplot2?

```
library(scales)
show_col(hue_pal()(6))
```



Existem muitos pacotes que fornecem paletas de cores como **ggsci**. Veja as possibilidades desse pacote no vignette.

Visualização de dados - ggplot2 - scale

Resumo:

- Escalas permitem especificar/alterar os aspectos visuais do gráfico referentes aos dados mapeados.
- Geralmente, para cada atributo estético, existe uma escala correspondente.

Quais são as escalas existentes e o que elas modificam?

- Consulte as funções (ex: scale_ + **TAB**) e veja exemplos na documentação de cada função (ex: ? scale_shape_manual).
- Consulte materiais online como ggplot2 colors.
- Faça pequisas em buscadores na internet ou fóruns especializados.

Visualização de dados - ggplot2 - coord

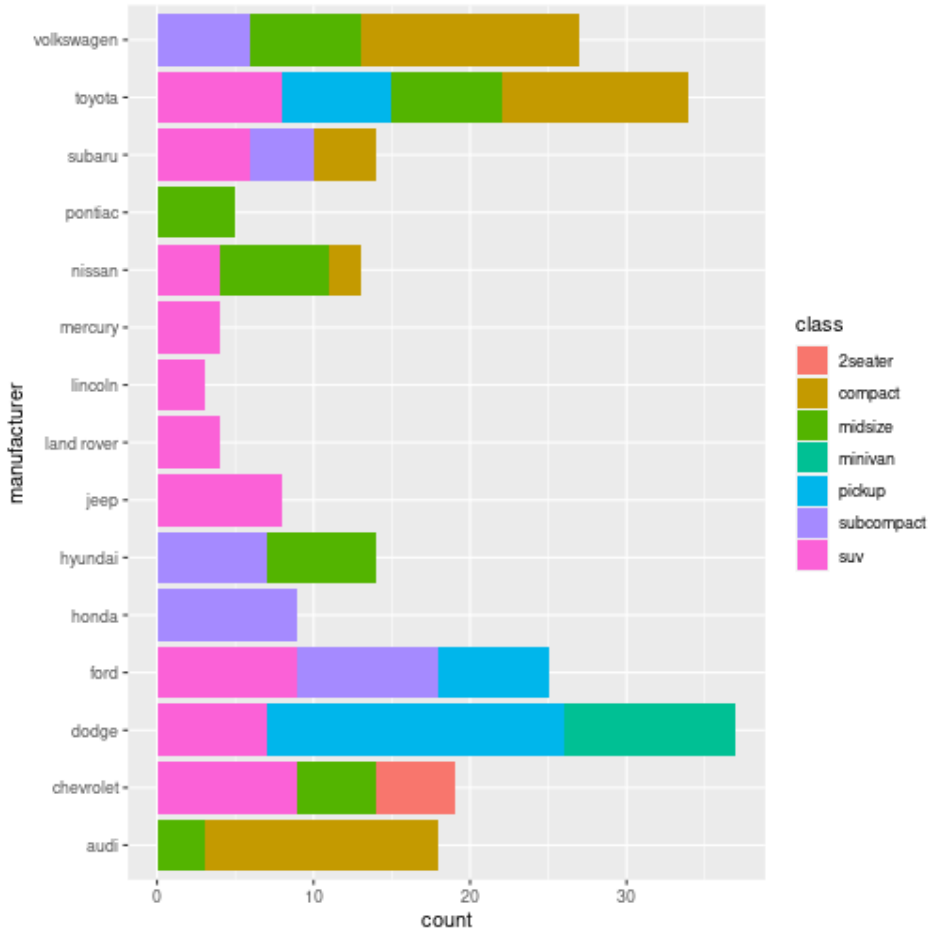
Coord. Permite definir posição e aparência dos objetos mapeados no plano gráfico, dentre outras disposições, orientações e perpectivas dos elementos gráficos. Existe uma família de funções do tipo coord_tipo-de-operacao ou coord_sistema-de-coordenadas.

Descubra algumas opções digitando coord_ e presionando TAB.

Visualização de dados - ggplot2 - coord

Podemos especificar a orientação dos eixos no gráfico, conforme:

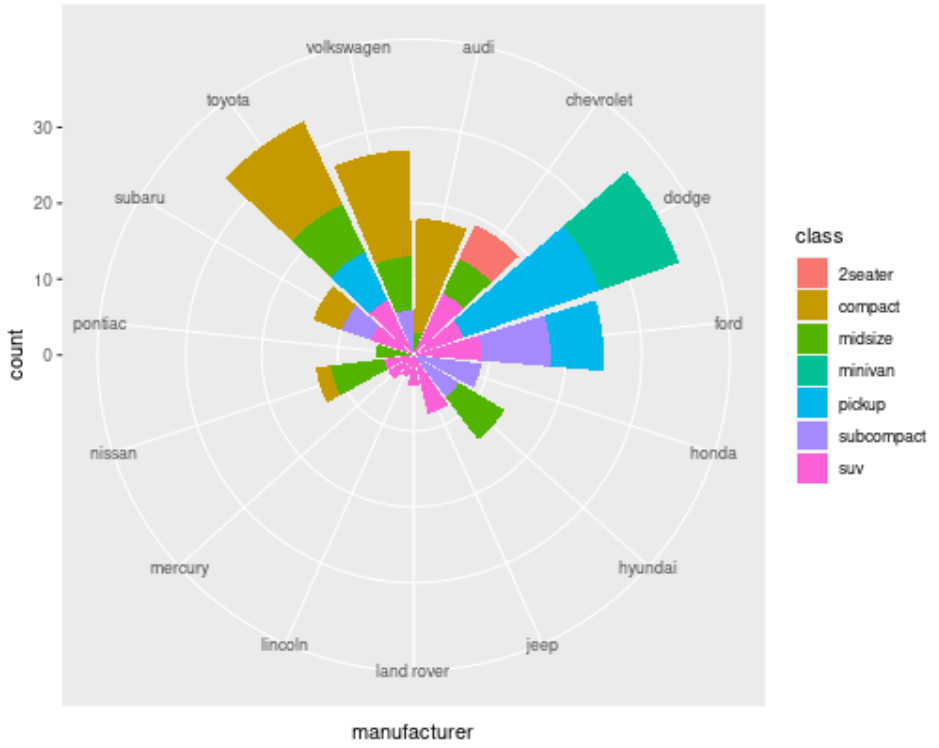
```
ggplot() +
  geom_bar(
    data = mpg,
    mapping = aes(x = manufacturer,
                  fill = class)) +
  coord_flip()
```



Visualização de dados - ggplot2 - coord

Podemos especificar o sistema de coordenadas do gráfico, conforme:

```
ggplot() +
  geom_bar(
    data = mpg,
    mapping = aes(x = manufacturer,
                  fill = class)) +
  coord_polar()
```



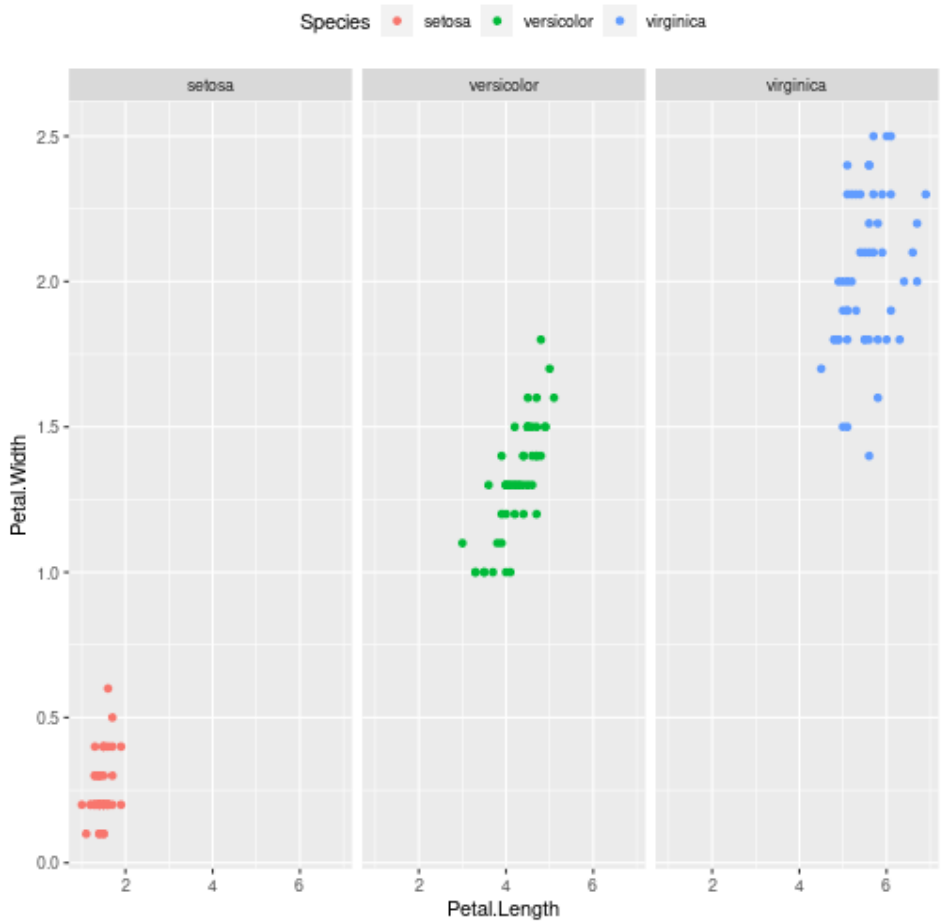
Visualização de dados - ggplot2 - faceting

Faceting. Gera painéis contendo subdivisões dos dados baseadas em informações categóricas. Os subgráficos compartilham os mesmos atributos estéticos. O **ggplot2** possui duas funções para gerar facetamento: facet_wrap() e facet_grid().

Visualização de dados - ggplot2 - faceting

Podemos aplicar o facetamento, conforme:

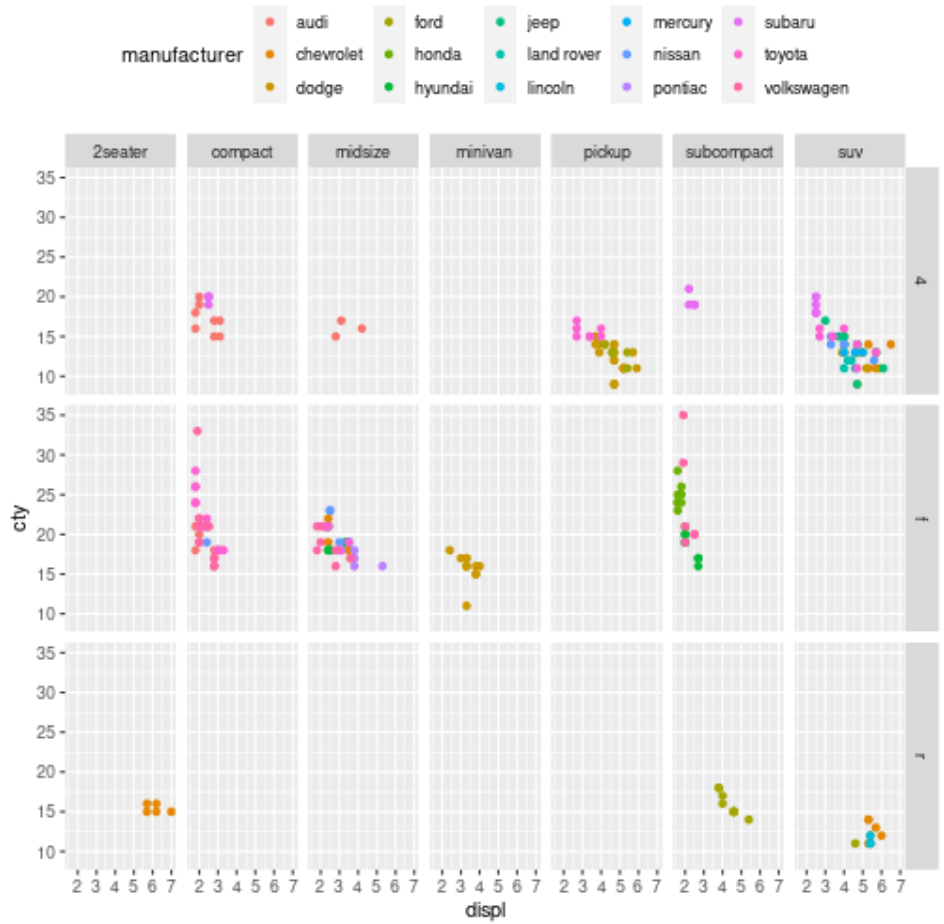
```
ggplot(data = iris,
       mapping = aes(x = Petal.Length,
                     y = Petal.Width)) +
  geom_point(aes(color = Species)) +
  theme(legend.position = "top") +
  facet_wrap(~Species)
```



Visualização de dados - ggplot2 - faceting

Podemos aplicar o facetamento, conforme:

```
ggplot(data = mpg,
       mapping = aes(x = displ,
                     y = cty)) +
  geom_point(aes(color = manufacturer)) +
  theme(legend.position = "top") +
  facet_grid(drv~class)
```



Visualização de dados - ggplot2 - theme

Theme. Define diversos elementos gerais de um gráfico como: cor do plano de fundo; tamanho e cores das bordas externas; tamanho do texto e tipo de fonte dos eixos, legenda, títulos e demais anotações; posicionamento e orientação de legendas; posicionamento e orientação de labels nos eixos; entre outros.

- Permite personalizações e refinamento do gráfico.
- Uma vez descritas, as personalizações podem ser aplicadas em gráficos diversos.

Por onde começar?

- Existem temas gráficos prontos. Consulte com theme_ + **TAB**.

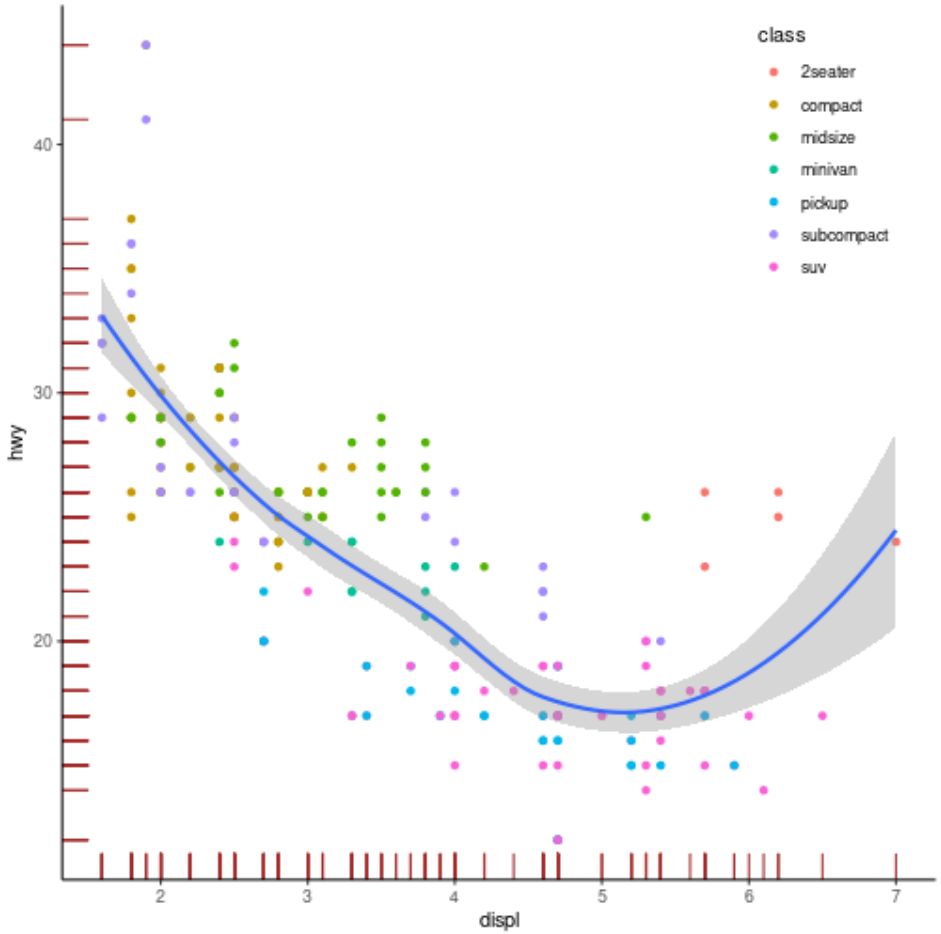
Quantos elementos um tema gráfico possui?

- Observe esse cheat sheet.
- Observe esse material online.

Visualização de dados - ggplot2 - theme

Podemos aplicar um tema, conforme:

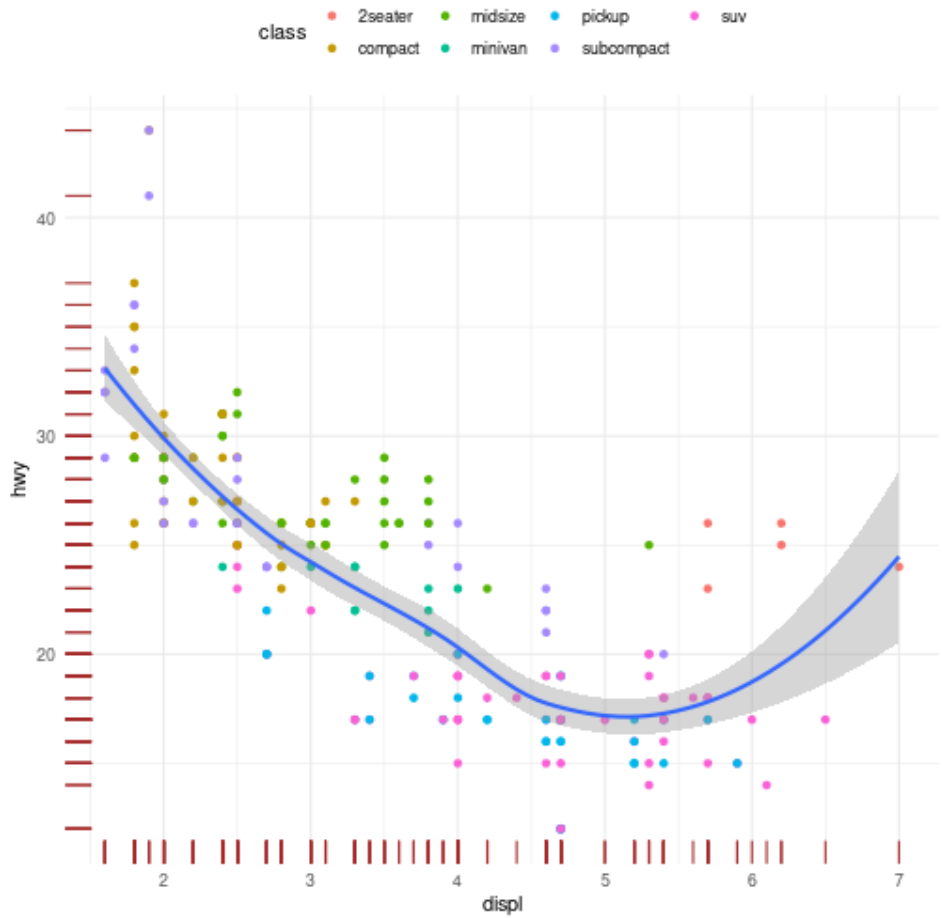
```
ggplot(data = mpg,
       mapping = aes(x = displ, y = hwy)) +
    geom_point(aes(color = class)) +
    geom_smooth() +
    geom_rug(color = "brown") +
    theme_classic() +
    theme(legend.position = c(.87, .83))
```



Visualização de dados - ggplot2 - theme

Podemos aplicar um tema, conforme:

```
ggplot(data = mpg,
       mapping = aes(x = displ, y = hwy)) +
    geom_point(aes(color = class)) +
    geom_smooth() +
    geom_rug(color = "brown") +
    theme_minimal() +
    theme(legend.position = "top",
          legend.direction = "horizontal")
```



Visualização de dados - ggplot2 - annotations

Annotations. Permitem definir elementos textuais como títulos do gráfico, eixos e legendas, assim como, inserções textuais no plano gráfico ou em elementos geométricos específicos (pontos, linhas e barras). O **ggplot2** possui algumas funções para isso como labs(), ylab(), xlab(), geom_text(), geom_label() e annotate().

Por onde começar?

- Testando funções mais simples como labs(), xlab() e ylab().

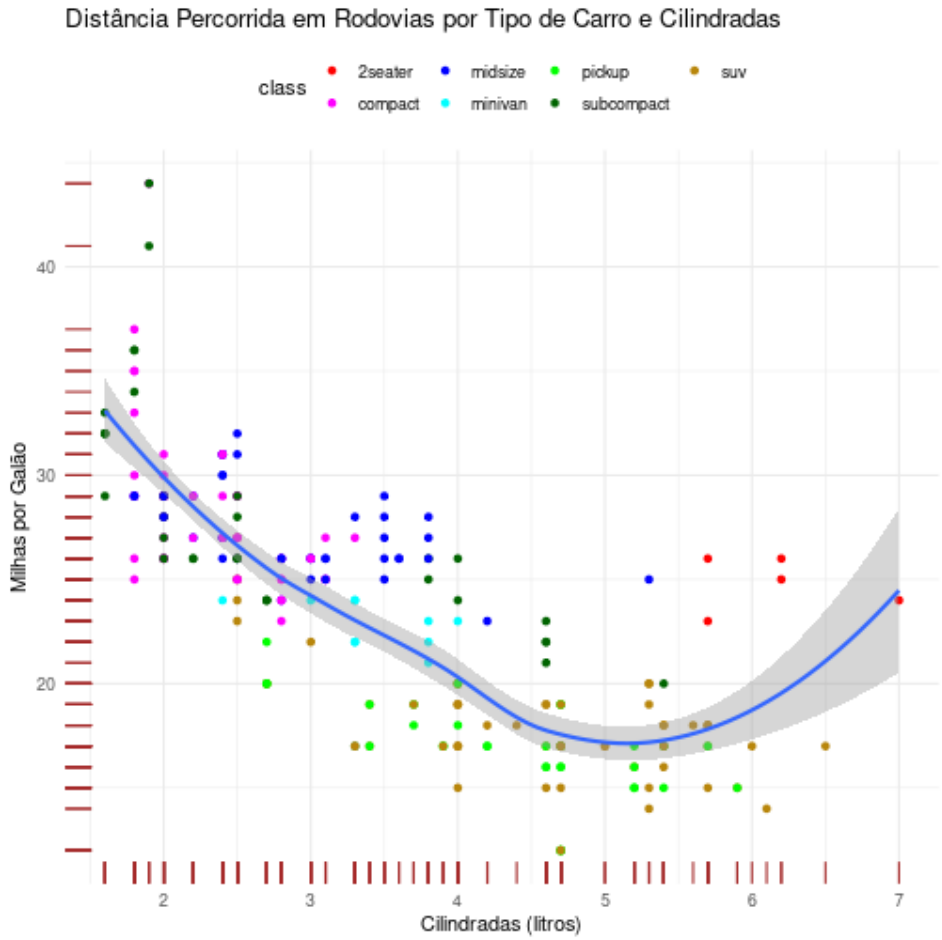
Conteúdo mais aprofundado?

- Materiais online como o capítulo Annotations do livro ggplot2: Elegant Graphics for Data Analysis.

Visualização de dados - ggplot2 - annotations

Vejamos exemplos de anotações:

```
my_title <- paste0("Distância Percorrida",
" em Rodovias por Tipo", " de Carro e ",
"Cilindradas")
my_colors <- c("red","magenta","blue",
"cyan","green","darkgreen","darkgoldenrod")
ggplot(data = mpg,
       mapping = aes(x = displ, y = hwy)) +
  geom_point(aes(color = class)) +
  geom_smooth() +
  geom_rug(color = "brown") +
  scale_color_manual(values = my_colors) +
  theme_minimal() +
  theme(legend.position = "top",
        legend.direction = "horizontal") +
  labs(title = my_title,
       x = "Cilindradas (litros)",
       y = "Milhas por Galão")
```



Visualização de dados - plotly

O pacote **plotly** permite a criação de gráficos interativos diversos. Essa bibliteca é fornecida pela empresa Plotly (sede em Quebec, Canadá) que produz serviços para análise e visualização de dados em páginas *web*. Também são fornecidas bibliotecas para outras linguagens de programação como MATLAB, python, entre outras. O pacote **plotly** tem como principais características:

- Tem licença aberta, ou seja, pode ser usada livremente.
- Renderiza os gráficos através da biblioteca *JavaScript* plotly.js.
- Baseado na fisosofia *"Grammar of Graphics"*.
- Permite converter gráficos estáticos gerados pelo **ggplot2** em interativos.

Visualização de dados - plotly - ggplotly

Como criar um gráfico interativo?

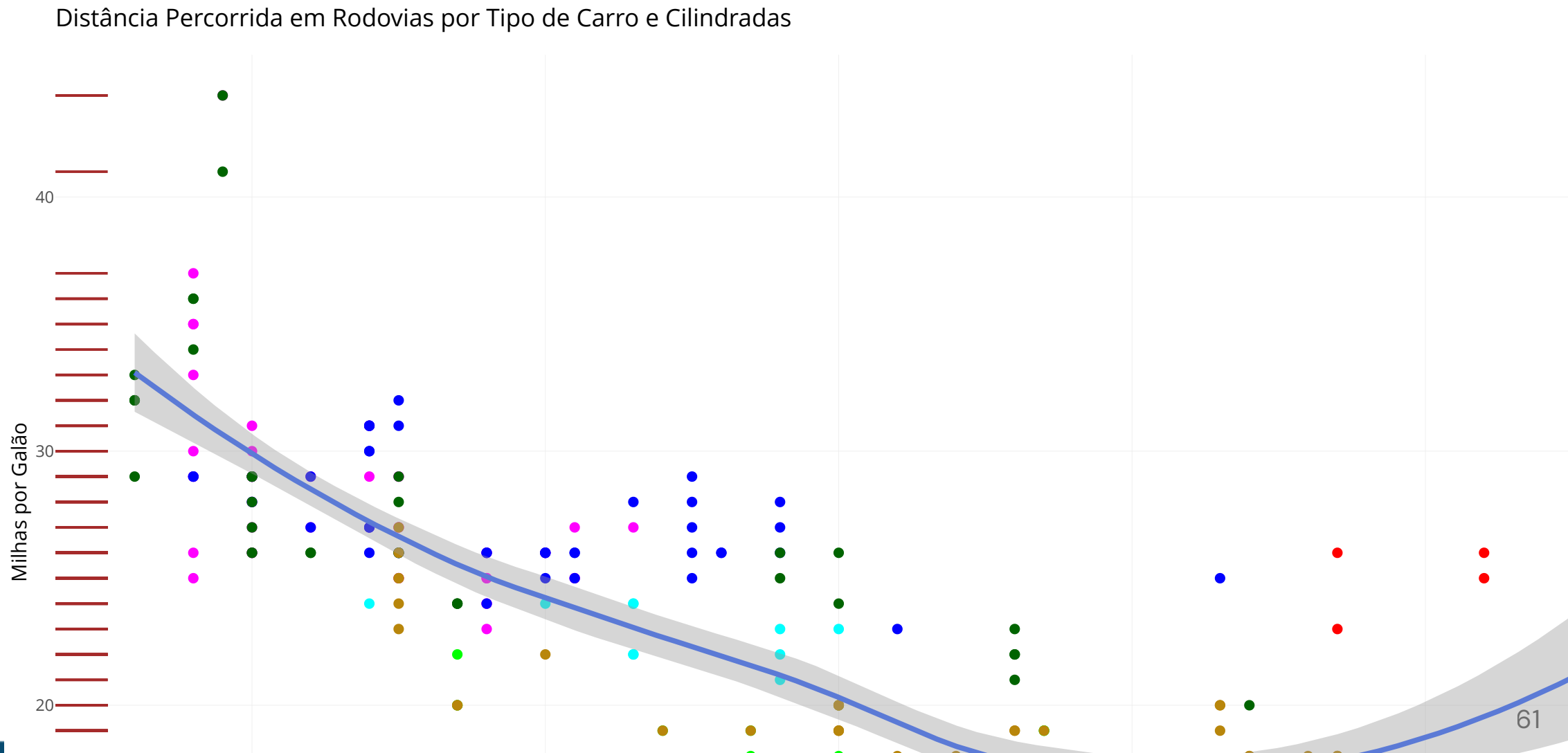
Um modo prático para obter um gráfico interativo é converter um gráfico gerado a partir do pacote **ggplot2** usando a função de conversão ggplotly() do pacote **plotly**. Primeiro, precisamos desse pacote instalado e carregado:

```
# install.packages("plotly")
library(plotly)
```

Também criaremos um gráfico com ggplot() e faremos a conversão com ggplotly():

```
my_plot <- ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
           geom_point(aes(color = class)) + geom_smooth() + geom_rug(color = "brown") +
           scale_color_manual(values = my_colors) + theme_minimal() +
           theme(legend.position = "top", legend.direction = "horizontal") +
           ylab("Milhas por Galão") + xlab("Cilindradas (litros)") +
           labs(title = my_title)
ggplotly(my_plot)
```

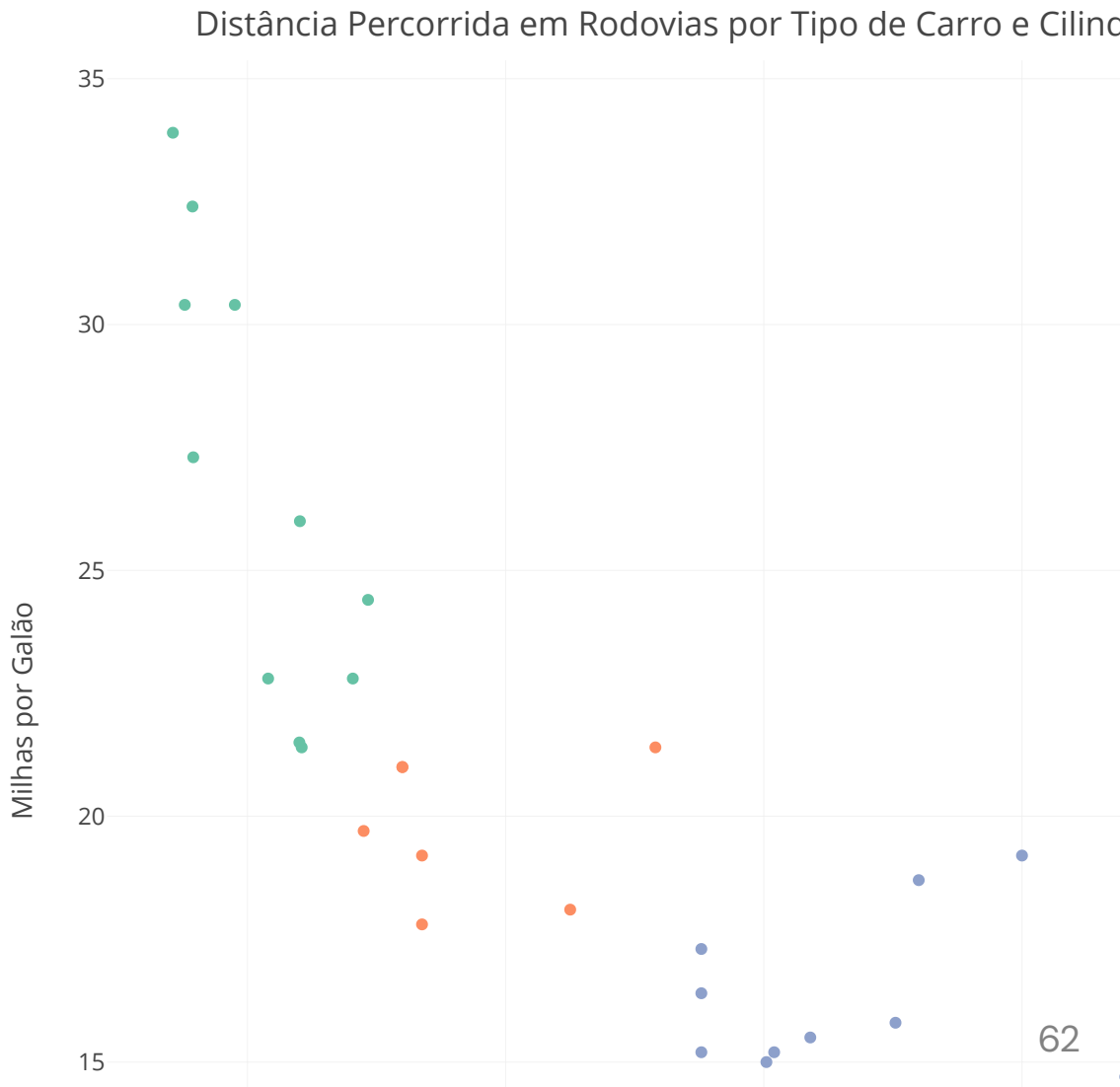
Visualização de dados - plotly - ggplotly



Visualização de dados - plotly

Também podemos criar o gráfico diretamente, conforme:

```
mtcars <- as_tibble(mtcars,
                    rownames = 'model')
plot_ly(
  data = mtcars,
  x = ~disp, y = ~mpg,
  color = ~factor(cyl), text = ~model,
  type = "scatter", mode = "markers"
) |>
layout(
  title = my_title,
  xaxis = list(
    title = "Cilindradas (litros)"),
  yaxis = list(
    title = "Milhas por Galão")
)
```



Visualização de dados - plotly

Mais informações:

- No próprio site do pacote.
- Materiais online.

Metotologias Informacionais com 

Muito Obrigado pela Atenção!