

# Uma nova fase no nosso curso: Fazendo análise de dados de alto nível com o tidyverse

Atá agora aprendemos muito sobre o \$R# sem recorrer a muitas bobliotecas, mas agora vamos usar passar para um novo nível usando o **tidyverse**.

O tidyverse é uma coleção de pacotes R de código aberto introduzidos por Hadley Wickham e sua equipe que "compartilham uma filosofia de design, gramática e estruturas de dados subjacentes" de dados organizados.

As funções contêm vários recursos incomuns em comparação com outros paradigmas de programação R, incluindo avaliação não padrão, que permite cadeias de caracteres não citadas na maioria das funções e usando dados consistentemente como o primeiro argumento, o que permite o uso liberal da função de pipe `magrittr::>`.

Os pacotes principais são **ggplot2**, **dplyr**, **tidyr**, **readr**, **purrr**, **tibble**, **stringr** e **forcats**, que fornecem funcionalidade para modelar, transformar e visualizar dados.

Em novembro de 2018, o pacote tidyverse e alguns de seus pacotes individuais compunham 5 dos 10 pacotes R mais baixados, e são o assunto de vários livros e artigos.

Foi criticado por Norman Matloff e Bob Muenchen por causa de sua ampla promoção pelo RStudio e sua alegação de fácil de aprender, apesar de ser mais complicado de aprender para iniciantes ou não-programadores.

# Os pacotes do tidyverse



# ggplot2

**ggplot2** é um sistema para a criação declarativa de gráficos, baseado na Gramática dos Gráficos. Você fornece os dados, diz ao ggplot2 como mapear as variáveis para a estética, quais primitivos gráficos usar e ele cuida dos detalhes. [Vá para a documentação ...](https://ggplot2.tidyverse.org/)  
(<https://ggplot2.tidyverse.org/>).

# dplyr

**dplyr** fornece uma gramática de manipulação de dados, fornecendo um conjunto consistente de verbos que resolvem os desafios mais comuns de manipulação de dados. [Vá para a documentação ...](https://dplyr.tidyverse.org/) (<https://dplyr.tidyverse.org/>).

# tidyr

**tidyr** fornece um conjunto de funções que o ajudam a organizar os dados. Dados organizados são dados com uma forma consistente: em resumo, cada variável entra em uma coluna e cada coluna é uma variável. [Vá para a documentação ...](https://tidyr.tidyverse.org/)  
(<https://tidyr.tidyverse.org/>).

# readr

**readr** fornece uma maneira rápida e amigável de ler dados retangulares (como csv, tsv e fwf). Ele é projetado para analisar de forma flexível muitos tipos de dados encontrados em ambiente selvagem, enquanto ainda falha de forma limpa quando os dados são alterados inesperadamente. [Vá para a documentação ... \(https://readr.tidyverse.org/\)](https://readr.tidyverse.org/).

## stringr

**stringr** fornece um conjunto coeso de funções projetadas para tornar o trabalho com strings o mais fácil possível. Ele é construído sobre stringi, que usa a biblioteca ICU C para fornecer implementações corretas e rápidas de manipulações comuns de string. [Vá para a documentação ... \(https://stringr.tidyverse.org/\)](https://stringr.tidyverse.org/).

## forcats

**forcats** fornece um conjunto de ferramentas úteis que resolvem problemas comuns com fatores. R usa fatores para lidar com variáveis categóricas, variáveis que têm um conjunto fixo e conhecido de valores possíveis. [Vá para a documentação ... \(https://forcats.tidyverse.org/\)](https://forcats.tidyverse.org/).

## tibble

**tibble** é uma reformulação moderna dos dataframes, com o tempo que se mostrou eficaz. Tibbles são data.frames preguiçosos e rudes: eles fazem menos e reclamam mais, forçando você a enfrentar os problemas mais cedo, normalmente levando a um código mais limpo e expressivo. [Vá para a documentação ... \(https://tibble.tidyverse.org/\)](https://tibble.tidyverse.org/).

# purrr

**purrr** aprimora o kit de ferramentas de programação funcional (FP) do R, fornecendo um conjunto completo e consistente de ferramentas para trabalhar com funções e vetores. Depois de dominar os conceitos básicos, o purrr permite substituir muitos loops for por um código mais fácil de escrever e mais expressivo. [Vá para a documentação ...](https://purrr.tidyverse.org/)  
(<https://purrr.tidyverse.org/>).

## **Visualização de Dados Parte 2 - GGPlot2**



# 1. Introdução

*“O gráfico simples trouxe mais informações à mente do analista de dados do que qualquer outro dispositivo.” - John Tukey*

Esta aula irá ensiná-lo a visualizar seus dados usando ggplot2.

R possui vários sistemas para fazer gráficos, mas **ggplot2** é um dos mais elegantes e versáteis.

O **ggplot2** implementa a gramática dos gráficos, um sistema coerente para descrever e construir gráficos.

Com o **ggplot2**, você pode fazer mais rápido aprendendo um sistema e aplicando-o em vários lugares.

Se quiser saber mais sobre os fundamentos teóricos do **ggplot2** antes de começar, recomenda-se a leitura de “The Layered Grammar of Graphics”, <http://vita.had.co.nz/papers/layered-grammar.pdf> (<http://vita.had.co.nz/papers/layered-grammar.pdf>).

## 1.1 Pré-requisitos

Este capítulo concentra-se no **ggplot2**, um dos principais membros do **tidyverse**.

Para acessar os conjuntos de dados, páginas de ajuda e funções que usaremos neste capítulo, instale e carregue o tidyverse executando estes código, a instalação pode demorar alguns minutos:

```
In [1]: # Para instalar  
install.packages("tidyverse")
```

Installing package into 'C:/Users/domin/OneDrive/Documentos/R/win-library/4.0'  
(as 'lib' is unspecified)

package 'tidyverse' successfully unpacked and MD5 sums checked

The downloaded binary packages are in  
C:\Users\domin\AppData\Local\Temp\RtmpktbBM3\downloaded\_packages

```
In [121]: library(tidyverse)
```

Você só precisa instalar um pacote uma vez, mas precisa recarregá-lo sempre que iniciar uma nova sessão.

Se precisarmos ser explícitos sobre de onde uma função (ou conjunto de dados) vem, usaremos o formulário especial

**package :: function()**

Por exemplo, **ggplot2 :: ggplot ()** diz explicitamente que estamos usando a função **ggplot ()** do pacote **ggplot2**.

## 2 Primeiros passos

Vamos usar nosso primeiro gráfico para responder a uma pergunta: carros com motores grandes usam mais combustível do que carros com motores pequenos?

Você provavelmente já tem uma resposta, mas tente dar uma resposta precisa. Como é a relação entre tamanho do motor e eficiência de combustível? É positivo? Negativo? Linear? Não linear?

## 2.1 O *dataframe* mpg

Você pode testar sua resposta com o dataframe mpg encontrado em ggplot2 (também conhecido como `ggplot2 :: mpg`).

Um dataframe é uma coleção retangular de variáveis (nas colunas) e observações (nas linhas). mpg contém observações coletadas pela Agência de Proteção Ambiental dos Estados Unidos em 38 modelos de automóveis.

In [122]: `head(mpg)`

A tibble: 6 × 11

manufacturer	model	displ	year	cyl	trans	drv	cty	hwy	fl	class
<chr>	<chr>	<dbl>	<int>	<int>	<chr>	<chr>	<int>	<int>	<chr>	<chr>
audi	a4	1.8	1999	4	auto(l5)	f	18	29	p	compact
audi	a4	1.8	1999	4	manual(m5)	f	21	29	p	compact
audi	a4	2.0	2008	4	manual(m6)	f	20	31	p	compact
audi	a4	2.0	2008	4	auto(av)	f	21	30	p	compact
audi	a4	2.8	1999	6	auto(l5)	f	16	26	p	compact
audi	a4	2.8	1999	6	manual(m5)	f	18	26	p	compact



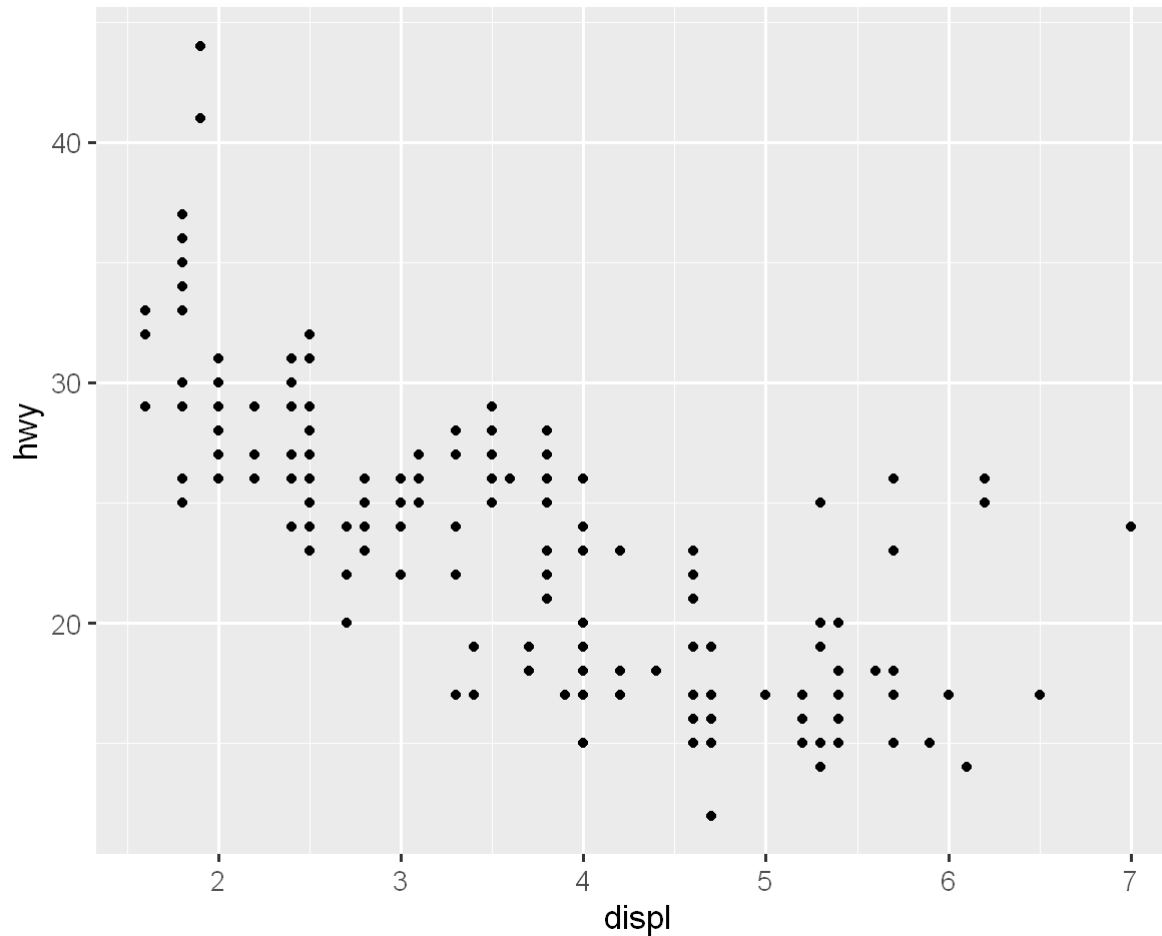
## As colunas de mpg são:

- manufacturer - nome do fabricante
- model - nome do modelo
- displ - cilindrada do motor, em litros
- year - ano de fabricação
- cyl - número de cilindros
- trans - tipo de câmbio
- drv - o tipo de transmissão, onde f = tração dianteira, r = tração traseira, 4 = 4wd
- cty - milhas urbanas por galão
- hwy - milhas rodoviárias por galão
- fl - tipo de combustível
- classe - "tipo" de carro

## 2.2 Criando um ggplot

Para plotar mpg, execute este código para colocar displ no eixo x e hwy no eixo y:

```
In [123]: # a bibliotec repr permite ajustes no tamanho do gráfico
library(repr)
# usando options você pode ajustar a altura e largura do gráfico
options(repr.plot.width=10, repr.plot.height=8)
theme_set(theme_gray(base_size = 20))
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy), size=2.5)
```



O gráfico mostra uma relação negativa entre o tamanho do motor (displ) e a eficiência do combustível (hwy).

Em outras palavras, carros com motores grandes consomem mais combustível. Isso confirma ou refuta sua hipótese sobre a eficiência do combustível e o tamanho do motor?

Com **ggplot2**, você começa um gráfico com a função `ggplot ()`. `ggplot ()` cria um sistema de coordenadas ao qual você pode adicionar camadas.

O primeiro argumento de `ggplot ()` é o conjunto de dados a ser usado no gráfico.

Então `ggplot (data = mpg)` cria um gráfico vazio, mas não é muito interessante, então não vou mostrá-lo aqui.

Você completa seu gráfico adicionando uma ou mais camadas ao `ggplot()`.

A função `geom_point()` adiciona uma camada de pontos ao seu gráfico, que cria um gráfico de dispersão.

O **ggplot2** vem com muitas funções geom, cada uma delas adicionando um tipo diferente de camada a um gráfico.

Você aprenderá um monte deles ao longo desta aula.

Cada função geom em **ggplot2** recebe um argumento de mapeamento.

Isso define como as variáveis em seu conjunto de dados são mapeadas para propriedades visuais.

O argumento de mapeamento é sempre emparelhado com `aes()`, e os argumentos `x` e `y` de `aes()` especificam quais variáveis mapear para os eixos `x` e `y`.

O **ggplot2** procura as variáveis mapeadas no argumento de dados, neste caso, `mpg`.

## 2.3 Um modelo de gráfico

Vamos transformar este código em um modelo reutilizável para fazer gráficos com **ggplot2**.

Para fazer um gráfico, substitua as seções entre colchetes no código abaixo por um conjunto de dados, uma função geom ou uma coleção de mapeamentos.

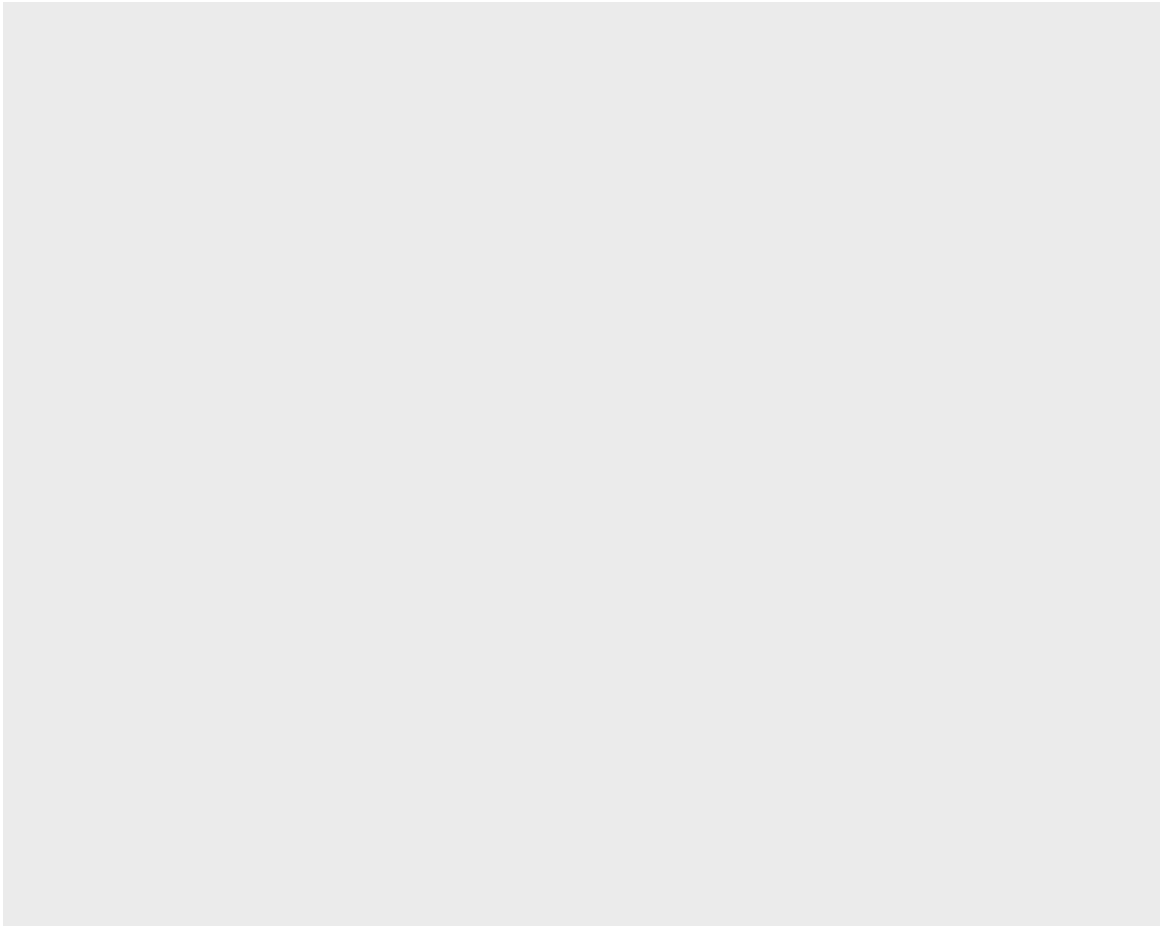
```
ggplot(data = <DATA>) +  
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

# Exercícios



Execute `ggplot(data = mpg)`. O que você vê?

In [124]: `ggplot(data = mpg)`



Quantas linhas estão em mpg? Quantas colunas?

In [35]: mpg

A tibble: 234 × 11

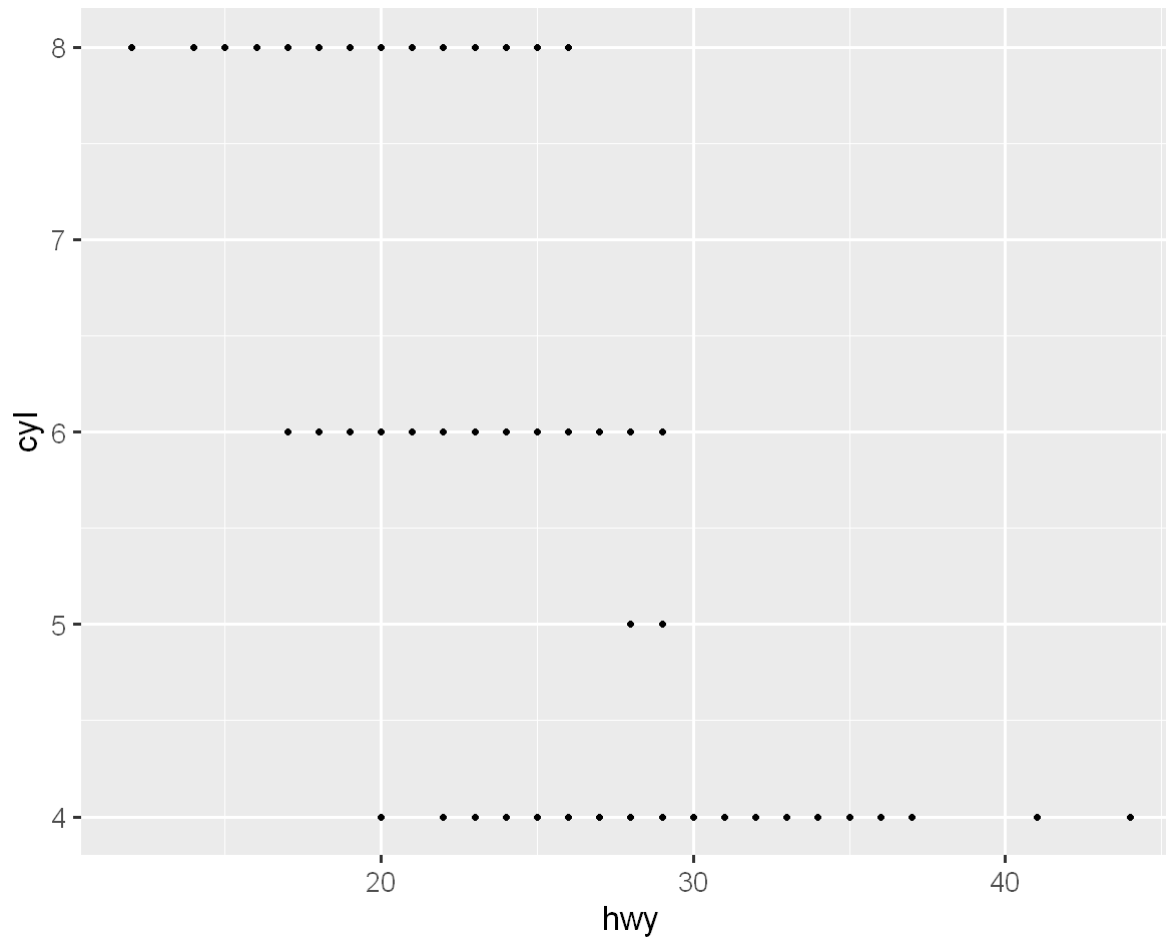
manufacturer	model	displ	year	cyl	trans	drv	cty	hwy	fl	class
<chr>	<chr>	<dbl>	<int>	<int>	<chr>	<chr>	<int>	<int>	<chr>	<chr>
audi	a4	1.8	1999	4	auto(l5)	f	18	29	p	compact
audi	a4	1.8	1999	4	manual(m5)	f	21	29	p	compact
audi	a4	2.0	2008	4	manual(m6)	f	20	31	p	compact
audi	a4	2.0	2008	4	auto(av)	f	21	30	p	compact
audi	a4	2.8	1999	6	auto(l5)	f	16	26	p	compact
audi	a4	2.8	1999	6	manual(m5)	f	18	26	p	compact
audi	a4	3.1	2008	6	auto(av)	f	18	27	p	compact
audi	a4 quattro	1.8	1999	4	manual(m5)	4	18	26	p	compact
audi	a4 quattro	1.8	1999	4	auto(l5)	4	16	25	p	compact
audi	a4 quattro	2.0	2008	4	manual(m6)	4	20	28	p	compact
audi	a4 quattro	2.0	2008	4	auto(s6)	4	19	27	p	compact
audi	a4 quattro	2.8	1999	6	auto(l5)	4	15	25	p	compact
audi	a4 quattro	2.8	1999	6	manual(m5)	4	17	25	p	compact
audi	a4 quattro	3.1	2008	6	auto(s6)	4	17	25	p	compact
audi	a4 quattro	3.1	2008	6	manual(m6)	4	15	25	p	compact
audi	a6 quattro	2.8	1999	6	auto(l5)	4	15	24	p	midsize
audi	a6 quattro	3.1	2008	6	auto(s6)	4	17	25	p	midsize
audi	a6 quattro	4.2	2008	8	auto(s6)	4	16	23	p	midsize
chevrolet	c1500 suburban 2wd	5.3	2008	8	auto(l4)	r	14	20	r	suv
chevrolet	c1500 suburban 2wd	5.3	2008	8	auto(l4)	r	11	15	e	suv
chevrolet	c1500 suburban 2wd	5.3	2008	8	auto(l4)	r	14	20	r	suv
chevrolet	c1500 suburban 2wd	5.7	1999	8	auto(l4)	r	13	17	r	suv
chevrolet	c1500 suburban 2wd	6.0	2008	8	auto(l4)	r	12	17	r	suv
chevrolet	corvette	5.7	1999	8	manual(m6)	r	16	26	p	2seater
chevrolet	corvette	5.7	1999	8	auto(l4)	r	15	23	p	2seater
chevrolet	corvette	6.2	2008	8	manual(m6)	r	16	26	p	2seater
chevrolet	corvette	6.2	2008	8	auto(s6)	r	15	25	p	2seater
chevrolet	corvette	7.0	2008	8	manual(m6)	r	15	24	p	2seater
chevrolet	k1500 tahoe 4wd	5.3	2008	8	auto(l4)	4	14	19	r	suv
chevrolet	k1500 tahoe 4wd	5.3	2008	8	auto(l4)	4	11	14	e	suv

O que a variável `drv` descreve? Leia a ajuda de `Mpg` para descobrir.

In [125]: `help(mpg)`

Faça um gráfico de dispersão de hwy vs cyl.

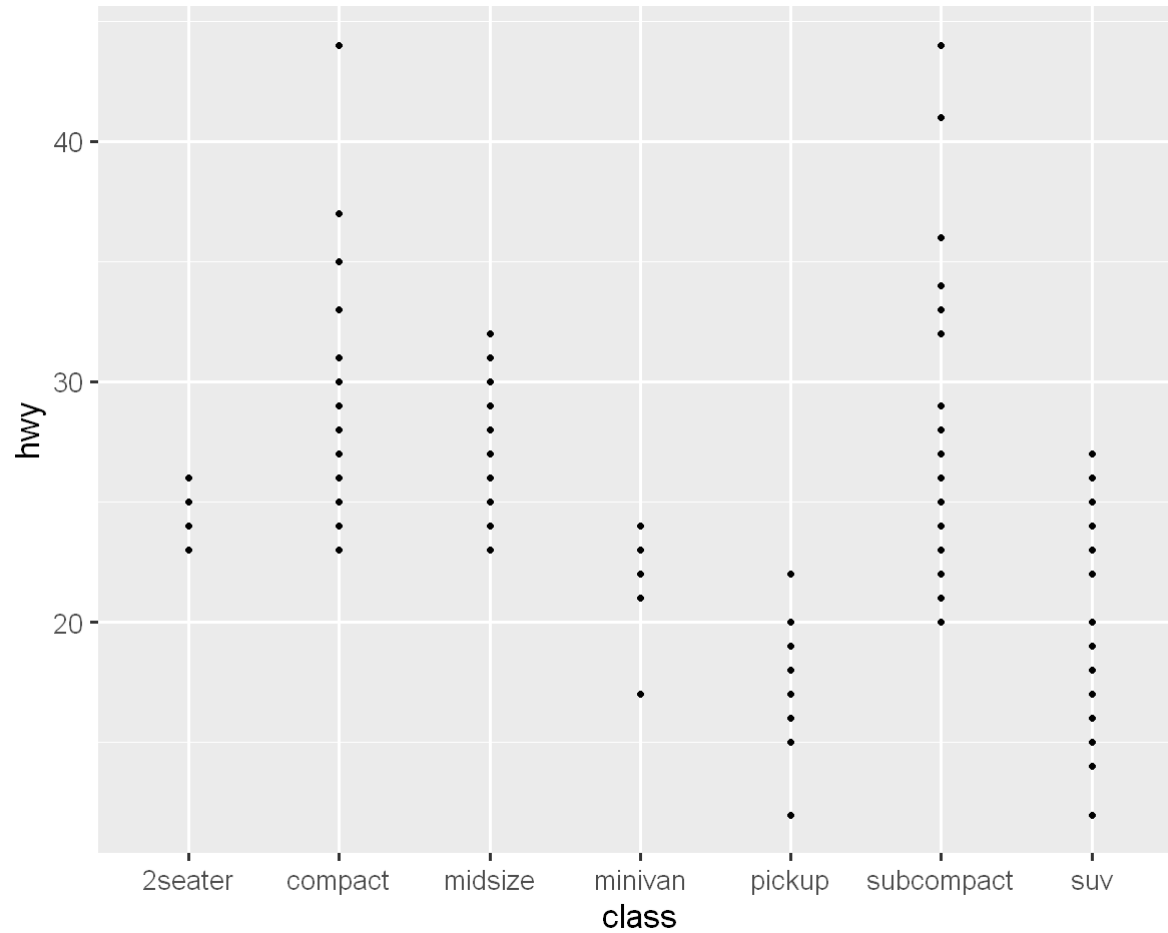
```
In [126]: ggplot(data = mpg) +  
  geom_point(mapping = aes(x = hwy, y = cyl))
```



O que acontece se você fizer um gráfico de dispersão de classe vs drv? Por que o gráfico não é útil?



```
In [127]: ggplot(data = mpg) +  
  geom_point(mapping = aes(x = class, y = hwy))
```



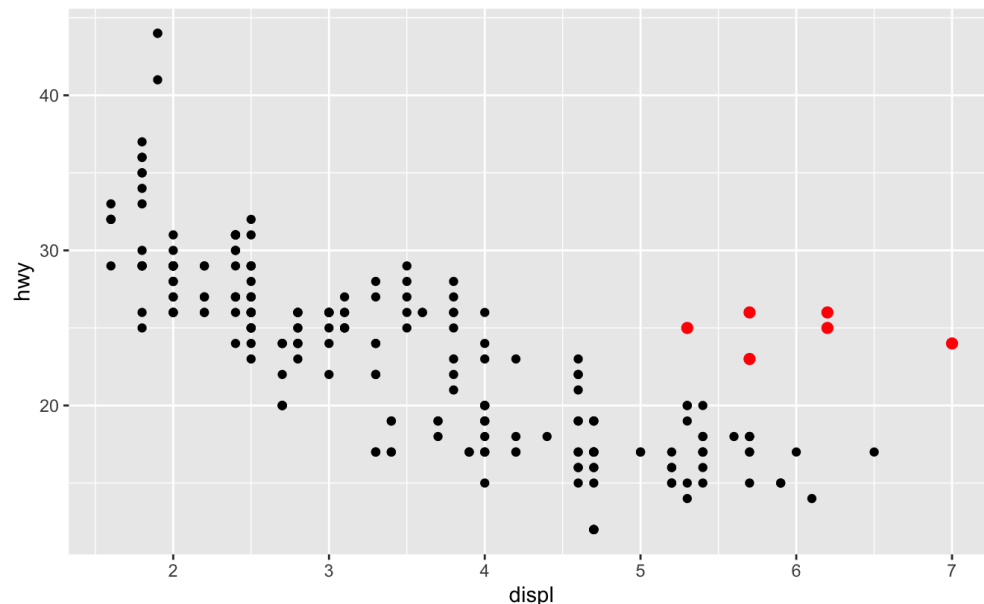
# 3 Aesthetic mappings (mapeamentos estéticos)

*“O maior valor de uma imagem é quando nos obriga a perceber o que nunca esperamos ver.” - John Tukey*

No gráfico abaixo, um grupo de pontos (destacado em vermelho) parece estar fora da tendência linear.

Esses carros têm uma quilometragem maior do que você imagina.

Como você pode explicar esses carros?



Vamos supor que os carros são híbridos.

Uma maneira de testar essa hipótese é examinar o valor da classe para cada carro. A variável de classe do conjunto de dados mpg classifica os carros em grupos, como compacto, médio e SUV.

Se os pontos periféricos forem híbridos, eles devem ser classificados como carros compactos ou, talvez, carros subcompactos (lembre-se de que esses dados foram coletados antes das Pick Ups híbridos e SUVs se tornarem populares).

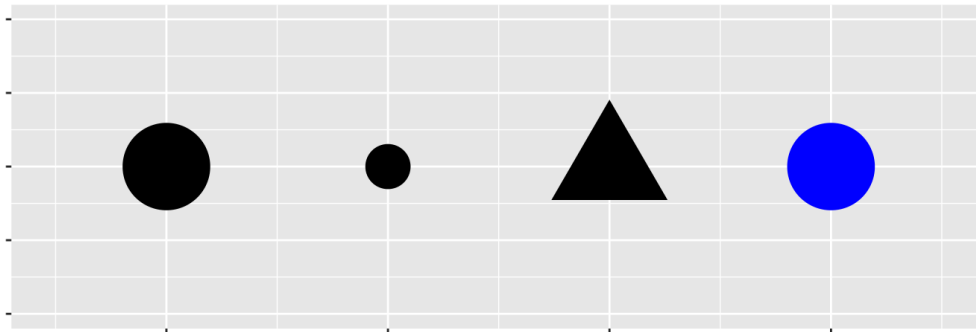
Você pode adicionar uma terceira variável, como classe, a um gráfico de dispersão bidimensional mapeando-o para uma estética .

Uma estética é uma propriedade visual dos objetos em seu enredo. A estética inclui coisas como o tamanho, a forma ou a cor dos seus pontos.

Você pode exibir um ponto (como o mostrado abaixo) de diferentes maneiras, alterando os valores de suas propriedades estéticas.

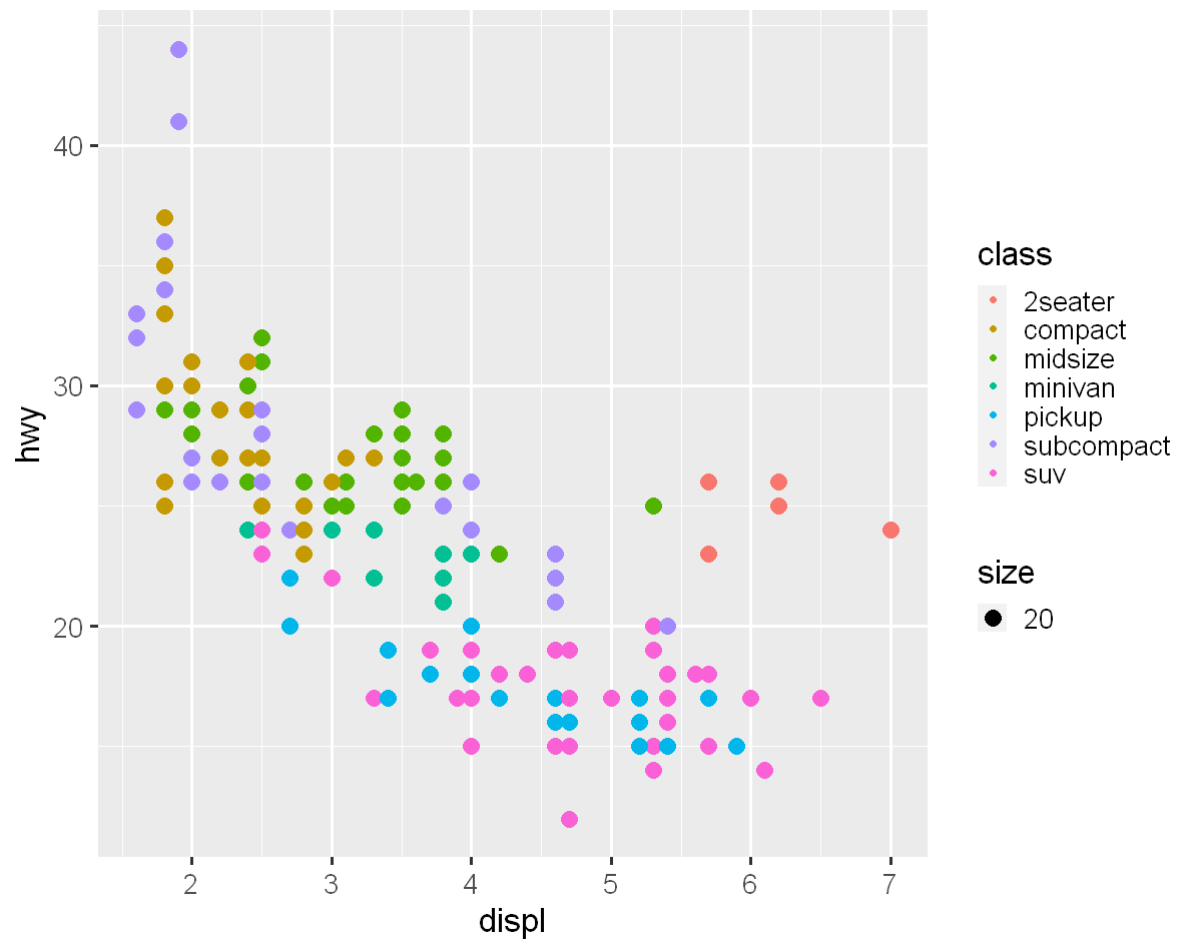
Uma vez que já usamos a palavra "valor" para descrever dados, vamos usar a palavra "nível" para descrever propriedades estéticas.

Aqui, alteramos os níveis de tamanho, forma e cor de um ponto para torná-lo pequeno, triangular ou azul:



Você pode transmitir informações sobre seus dados mapeando a estética em seu gráfico para as variáveis em seu conjunto de dados. Por exemplo, você pode mapear as cores de seus pontos para a variável de classe para revelar a classe de cada carro.

```
In [128]: ggplot(data = mpg) +  
  geom_point(mapping = aes(x = displ, y = hwy, color = class, size=20))
```



Para mapear uma estética a uma variável, associe o nome da estética ao nome da variável dentro de `aes()`. O `ggplot2` atribuirá automaticamente um nível único de estética (aqui uma cor única) para cada valor único da variável, um processo conhecido como escala. O `ggplot2` também adicionará uma legenda que explica quais níveis correspondem a quais valores.

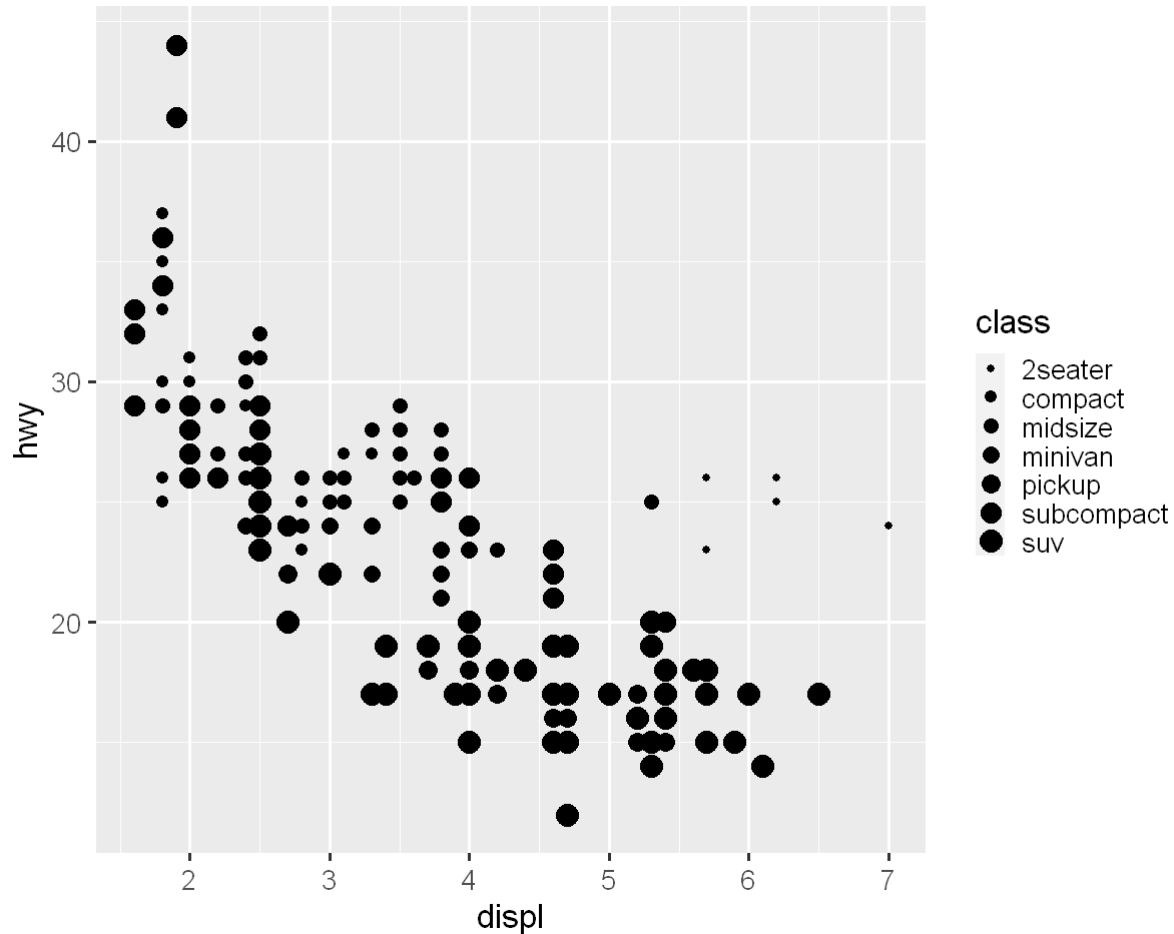
As cores revelam que muitos dos pontos incomuns são carros de dois lugares. Esses carros não parecem híbridos e são, na verdade, carros esportivos! Os carros esportivos têm motores grandes, como SUVs e picapes, mas carrocerias pequenas, como carros de tamanho médio e compactos, o que aumenta o consumo de combustível. Em retrospectiva, era improvável que esses carros fossem híbridos, pois têm motores grandes.

No exemplo acima, mapeamos a classe para a estética da cor, mas poderíamos ter mapeado a classe para a estética do tamanho da mesma maneira. Nesse caso, o tamanho exato de cada ponto revelaria sua filiação de classe. Recebemos um aviso aqui, porque mapear uma variável não ordenada (classe) para uma estética ordenada (tamanho) não é uma boa ideia.

```
In [129]: ggplot(data = mpg) +  
  geom_point(mapping = aes(x = displ, y = hwy, size = class))  
#> Warning: Using size for a discrete variable is not advised.  
#> Aviso: Usar o tamanho (size) para uma variável discreta não é recomendado.
```

Warning message:

"Using size for a discrete variable is not advised."



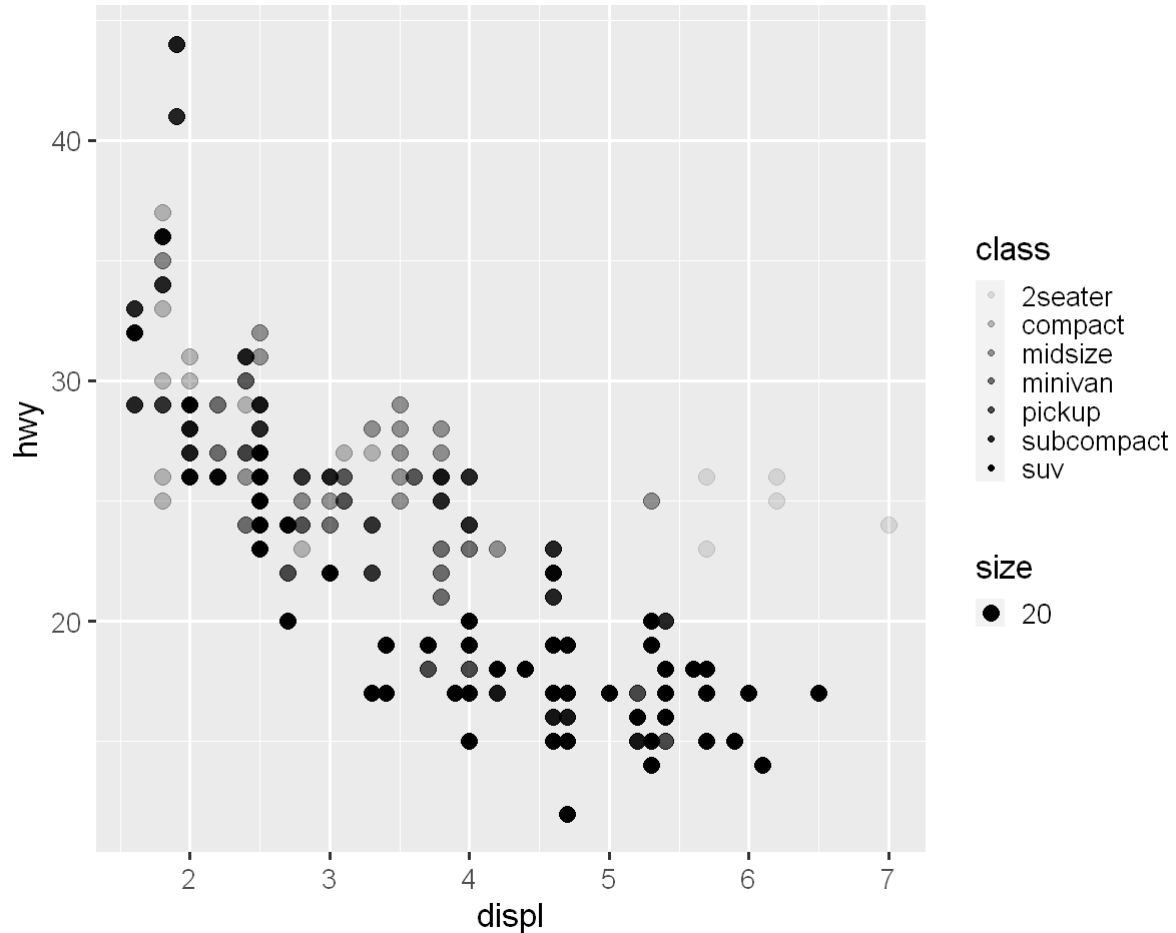


Ou poderíamos ter mapeado a classe para a estética alfa, que controla a transparência dos pontos, ou para a estética da forma, que controla a forma dos pontos.

```
In [131]: # Superior  
ggplot(data = mpg) +  
  geom_point(mapping = aes(x = displ, y = hwy, alpha = class, size=20))
```

Warning message:

"Using alpha for a discrete variable is not advised."



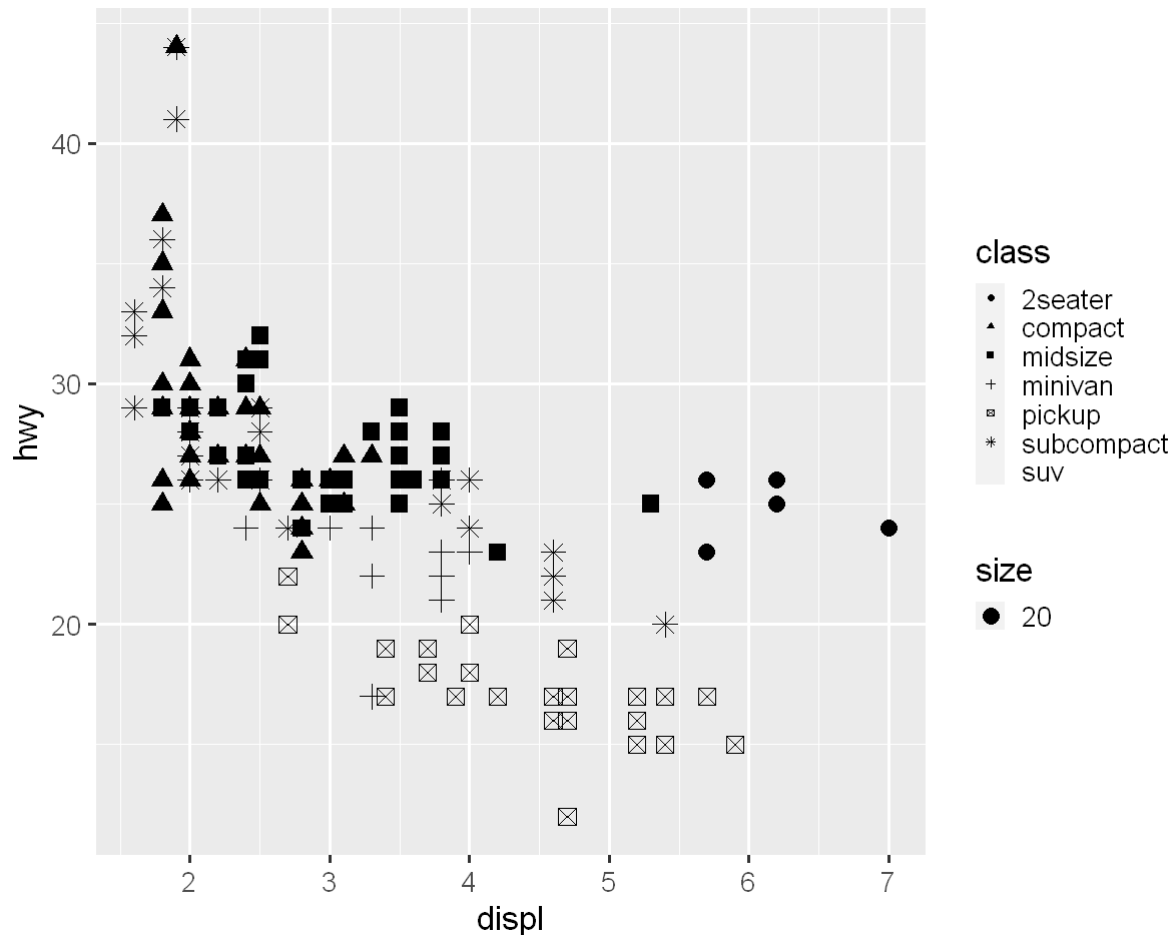
```
In [132]: # Inferior
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, shape = class, size=20))
```

Warning message:

"The shape palette can deal with a maximum of 6 discrete values because more than 6 becomes difficult to discriminate; you have 7. Consider specifying shapes manually if you must have them."

Warning message:

"Removed 62 rows containing missing values (geom\_point)."





O que aconteceu com os SUVs? O **ggplot2** usará apenas seis formas por vez. Por padrão, grupos adicionais não serão plotados quando você usar a estética de forma.

Para cada estética, você usa `aes()` para associar o nome da estética a uma variável a ser exibida. A função `aes()` reúne cada um dos mapeamentos estéticos usados por uma camada e os passa para o argumento de mapeamento da camada. A sintaxe destaca um insight útil sobre `x` e `y`:

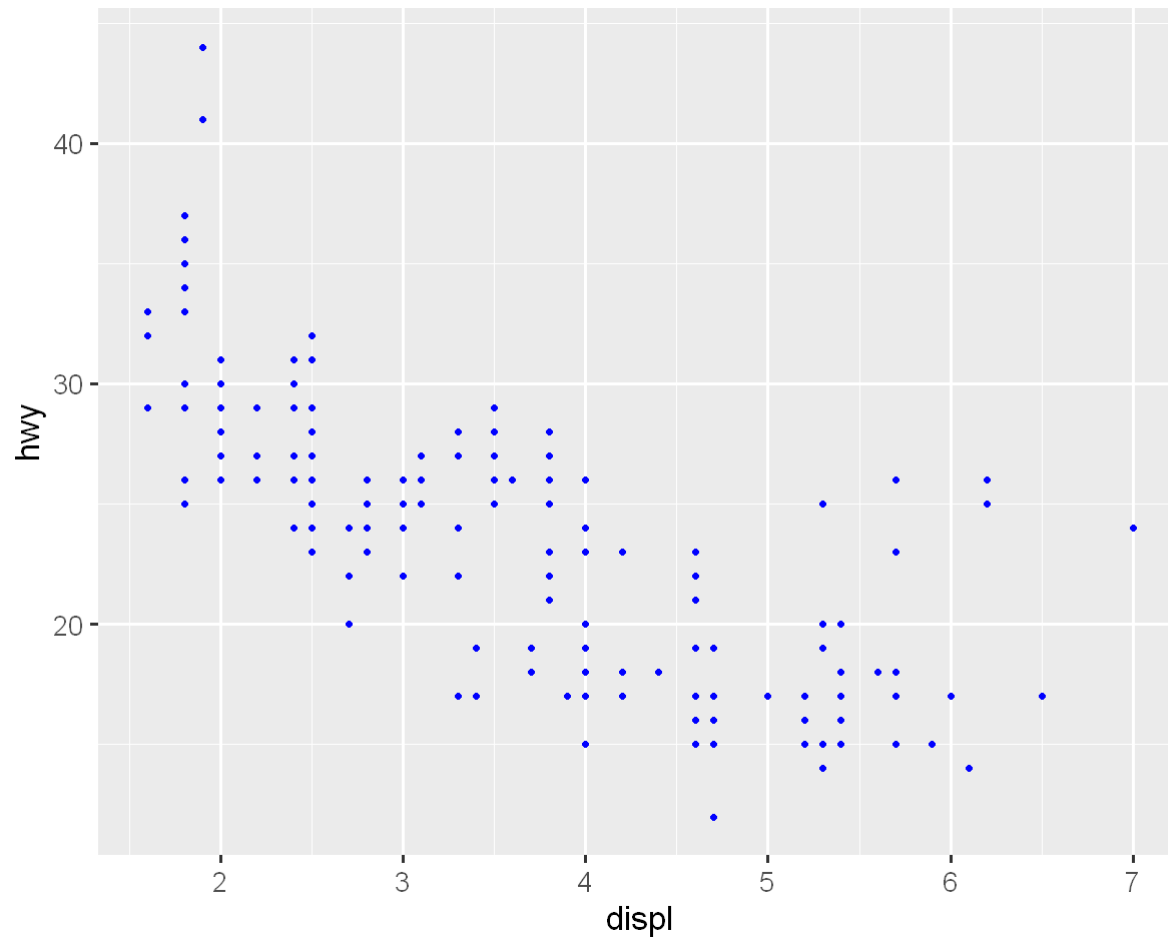
*as localizações `x` e `y` de um ponto são estéticas, propriedades visuais que você pode mapear para variáveis para exibir informações sobre os dados.*

Depois de mapear uma estética, o **ggplot2** cuida do resto. Ele seleciona uma escala razoável para usar com a estética e constrói uma legenda que explica o mapeamento entre níveis e valores.

Para a estética `x` e `y`, **ggplot2** não cria uma legenda, mas cria uma linha de eixo com marcas de escala e um rótulo. A linha do eixo atua como uma legenda; explica o mapeamento entre locais e valores.

Você também pode definir as propriedades estéticas de seu **geom** manualmente. Por exemplo, podemos tornar todos os pontos em nosso gráfico azuis:


























```
In [43]: ggplot(data = mpg) +  
  geom_point(mapping = aes(x = displ, y = hwy), color = "blue")
```



Aqui, a cor não transmite informações sobre uma variável, mas apenas altera a aparência do gráfico. Para definir uma estética manualmente, defina a estética pelo nome como um argumento de sua função geom; ou seja, sai de aes (). Você precisará escolher um nível que faça sentido para essa estética:

- O nome de uma cor como sequência de caracteres.
- O tamanho de um ponto em mm.

A forma de um ponto como um número, conforme mostrado na Figura abaixo:

 0	 4	 10	 15	 22
 1	 6	 11	 16	 21
 2	 7	 12	 17	 24
 5	 8	 13	 18	 23
 3	 9	 14	 19	 20

R tem 25 formas integradas que são identificadas por números. Existem algumas duplicatas aparentes: por exemplo, 0, 15 e 22 são todos quadrados.

A diferença vem da interação da estética da `colour` e do `fill`.

As formas ocas (0–14) têm uma borda determinada pela `colour`; as formas sólidas (15–20) são preenchidas com `colour`;

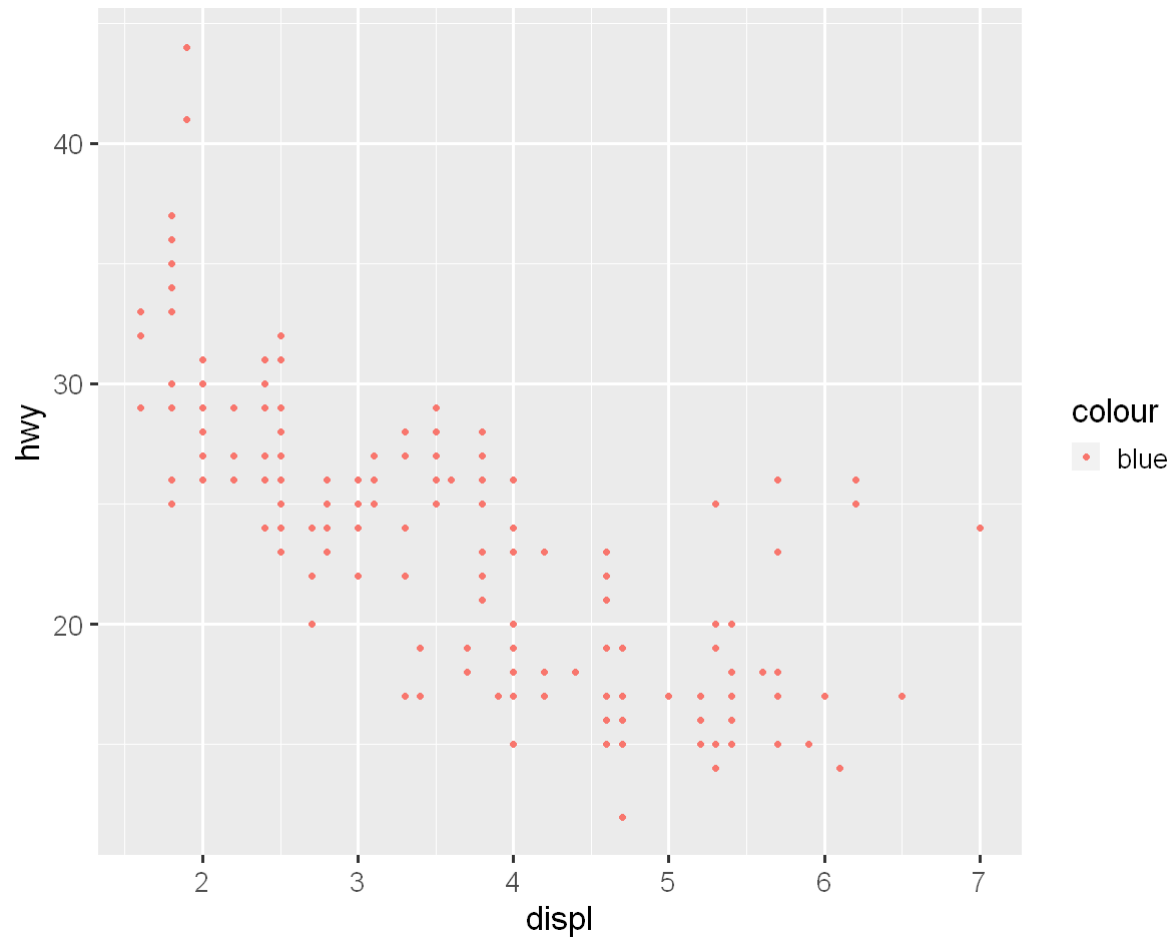
as formas preenchidas (21–24) têm a borda definida por `colour` e são preenchidas com `fill`.



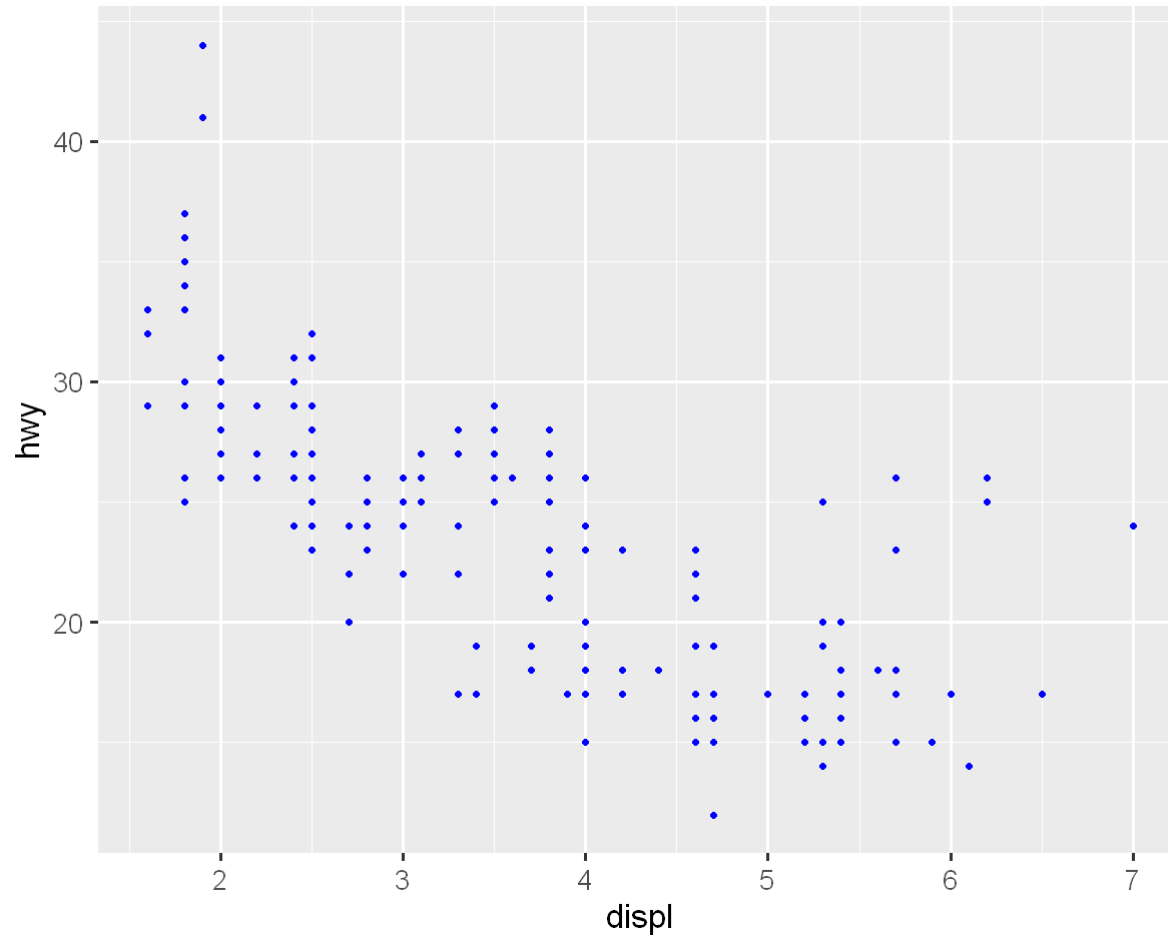
## Exercícios

O que há de errado com este código? Por que os pontos não são azuis?

```
In [133]: ggplot(data = mpg) +  
  geom_point(mapping = aes(x = displ, y = hwy, color = "blue"))
```



```
In [134]: ggplot(data = mpg) +  
  geom_point(mapping = aes(x = displ, y = hwy), color = "blue")
```



Quais variáveis em mpg são categóricas? Quais variáveis são contínuas? (Dica: digite? mpg para ler a documentação do conjunto de dados). Como você pode ver essas informações ao executar o mpg?

In [135]:

mpg

A tibble: 234 × 11

manufacturer	model	displ	year	cyl	trans	drv	cty	hwy	fl	class
<chr>	<chr>	<dbl>	<int>	<int>	<chr>	<chr>	<int>	<int>	<chr>	<chr>
audi	a4	1.8	1999	4	auto(l5)	f	18	29	p	compact
audi	a4	1.8	1999	4	manual(m5)	f	21	29	p	compact
audi	a4	2.0	2008	4	manual(m6)	f	20	31	p	compact
audi	a4	2.0	2008	4	auto(av)	f	21	30	p	compact
audi	a4	2.8	1999	6	auto(l5)	f	16	26	p	compact
audi	a4	2.8	1999	6	manual(m5)	f	18	26	p	compact
audi	a4	3.1	2008	6	auto(av)	f	18	27	p	compact
audi	a4 quattro	1.8	1999	4	manual(m5)	4	18	26	p	compact
audi	a4 quattro	1.8	1999	4	auto(l5)	4	16	25	p	compact
audi	a4 quattro	2.0	2008	4	manual(m6)	4	20	28	p	compact
audi	a4 quattro	2.0	2008	4	auto(s6)	4	19	27	p	compact
audi	a4 quattro	2.8	1999	6	auto(l5)	4	15	25	p	compact
audi	a4 quattro	2.8	1999	6	manual(m5)	4	17	25	p	compact
audi	a4 quattro	3.1	2008	6	auto(s6)	4	17	25	p	compact
audi	a4 quattro	3.1	2008	6	manual(m6)	4	15	25	p	compact
audi	a6 quattro	2.8	1999	6	auto(l5)	4	15	24	p	midsize
audi	a6 quattro	3.1	2008	6	auto(s6)	4	17	25	p	midsize
audi	a6 quattro	4.2	2008	8	auto(s6)	4	16	23	p	midsize
chevrolet	c1500 suburban 2wd	5.3	2008	8	auto(l4)	r	14	20	r	suv
chevrolet	c1500 suburban 2wd	5.3	2008	8	auto(l4)	r	11	15	e	suv
chevrolet	c1500 suburban 2wd	5.3	2008	8	auto(l4)	r	14	20	r	suv
chevrolet	c1500 suburban 2wd	5.7	1999	8	auto(l4)	r	13	17	r	suv
chevrolet	c1500 suburban 2wd	6.0	2008	8	auto(l4)	r	12	17	r	suv
chevrolet	corvette	5.7	1999	8	manual(m6)	r	16	26	p	2seater
chevrolet	corvette	5.7	1999	8	auto(l4)	r	15	23	p	2seater
chevrolet	corvette	6.2	2008	8	manual(m6)	r	16	26	p	2seater
chevrolet	corvette	6.2	2008	8	auto(s6)	r	15	25	p	2seater
chevrolet	corvette	7.0	2008	8	manual(m6)	r	15	24	p	2seater
chevrolet	k1500 tahoe 4wd	5.3	2008	8	auto(l4)	4	14	19	r	suv
chevrolet	k1500 tahoe 4wd	5.3	2008	8	auto(l4)	4	11	14	e	suv

manufacturer	model	displ	year	cyl	trans	drv	cty	hwy	fl	class
toyota	toyota tacoma 4wd	3.4	1999	6	auto(l4)	4	15	18	r	pickup
toyota	toyota tacoma 4wd	4.0	2008	6	auto(l5)	4	16	20	r	pickup
volkswagen	gti	2.0	1999	4	manual(m5)	f	21	29	r	compact
volkswagen	gti	2.0	1999	4	auto(l4)	f	19	26	r	compact
volkswagen	gti	2.0	2008	4	manual(m6)	f	21	29	p	compact
volkswagen	gti	2.0	2008	4	auto(s6)	f	22	29	p	compact
volkswagen	gti	2.8	1999	6	manual(m5)	f	17	24	r	compact
volkswagen	jetta	1.9	1999	4	manual(m5)	f	33	44	d	compact
volkswagen	jetta	2.0	1999	4	manual(m5)	f	21	29	r	compact
volkswagen	jetta	2.0	1999	4	auto(l4)	f	19	26	r	compact
volkswagen	jetta	2.0	2008	4	auto(s6)	f	22	29	p	compact
volkswagen	jetta	2.0	2008	4	manual(m6)	f	21	29	p	compact
volkswagen	jetta	2.5	2008	5	auto(s6)	f	21	29	r	compact
volkswagen	jetta	2.5	2008	5	manual(m5)	f	21	29	r	compact
volkswagen	jetta	2.8	1999	6	auto(l4)	f	16	23	r	compact
volkswagen	jetta	2.8	1999	6	manual(m5)	f	17	24	r	compact
volkswagen	new beetle	1.9	1999	4	manual(m5)	f	35	44	d	subcompact
volkswagen	new beetle	1.9	1999	4	auto(l4)	f	29	41	d	subcompact
volkswagen	new beetle	2.0	1999	4	manual(m5)	f	21	29	r	subcompact
volkswagen	new beetle	2.0	1999	4	auto(l4)	f	19	26	r	subcompact
volkswagen	new beetle	2.5	2008	5	manual(m5)	f	20	28	r	subcompact
volkswagen	new beetle	2.5	2008	5	auto(s6)	f	20	29	r	subcompact
volkswagen	passat	1.8	1999	4	manual(m5)	f	21	29	p	midsize
volkswagen	passat	1.8	1999	4	auto(l5)	f	18	29	p	midsize
volkswagen	passat	2.0	2008	4	auto(s6)	f	19	28	p	midsize
volkswagen	passat	2.0	2008	4	manual(m6)	f	21	29	p	midsize
volkswagen	passat	2.8	1999	6	auto(l5)	f	16	26	p	midsize
volkswagen	passat	2.8	1999	6	manual(m5)	f	18	26	p	midsize
volkswagen	passat	3.6	2008	6	auto(s6)	f	17	26	p	midsize

```
In [136]: names(mpg)
```

```
'manufacturer' · 'model' · 'displ' · 'year' · 'cyl' · 'trans' · 'drv' · 'cty' · 'hwy' ·  
'fl' · 'class'
```

In [137]: `sapply(mpg, class)`

<b>manufacturer:</b>	'character'	<b>model:</b>	'character'	<b>displ:</b>	'numeric'	<b>year:</b>
'integer'	<b>cyl:</b>	'integer'	<b>trans:</b>	'character'	<b>drv:</b>	'character'
'integer'	<b>hwy:</b>	'integer'	<b>fl:</b>	'character'	<b>class:</b>	'character'



## 4. Facetas

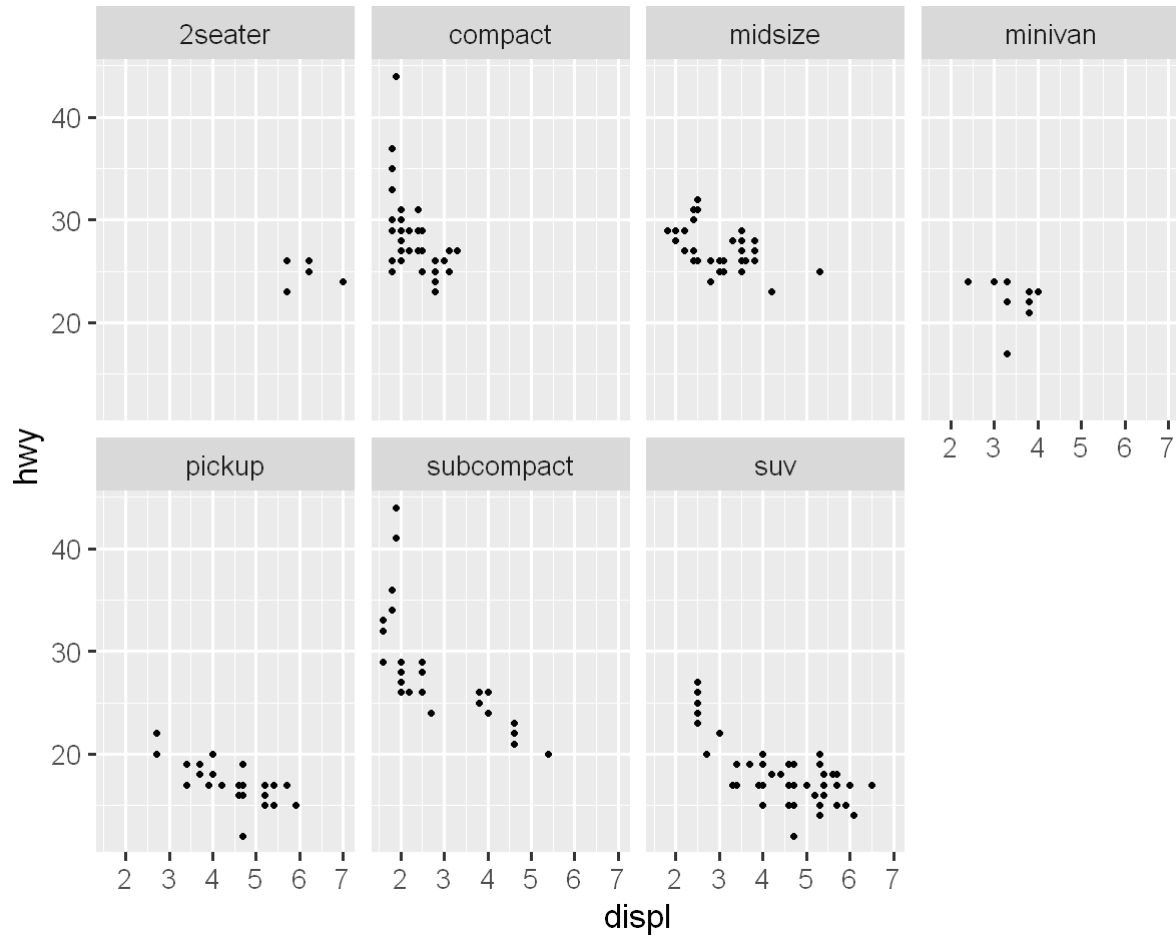
Uma maneira de adicionar variáveis adicionais é com a estética. Outra maneira, particularmente útil para variáveis categóricas, é dividir seu gráfico em facetas, subparcelas em que cada uma exibe um subconjunto dos dados.

Para facetar seu gráfico por uma única variável, use `facet_wrap()`.

O primeiro argumento de `facet_wrap()` deve ser uma fórmula, que você cria com `~` seguido por um nome de variável (aqui “fórmula” é o nome de uma estrutura de dados em R, não um sinônimo para “equação”).

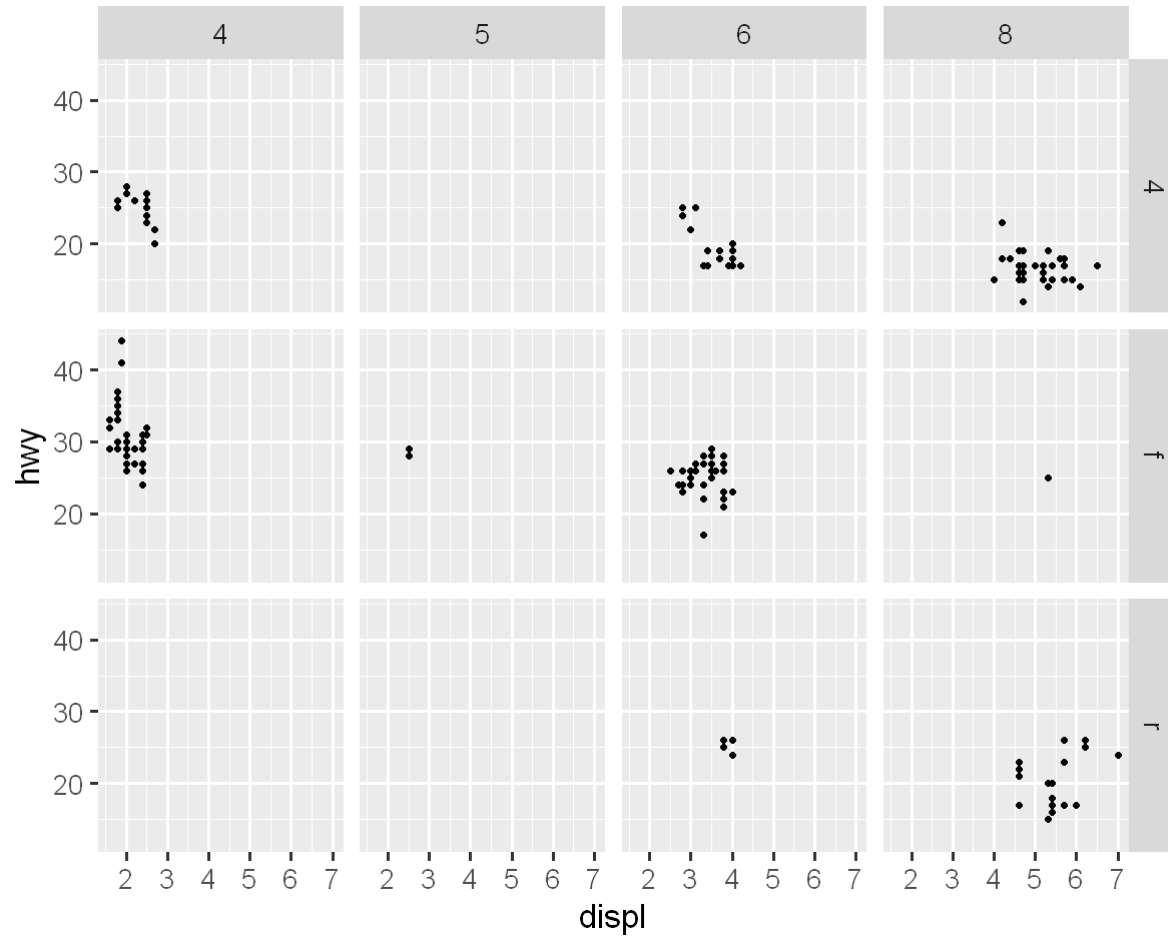
A variável que você passa para `facet_wrap()` deve ser discreta.

```
In [138]: ggplot(data = mpg) +  
  geom_point(mapping = aes(x = displ, y = hwy)) +  
  facet_wrap(~ class, nrow = 2)
```



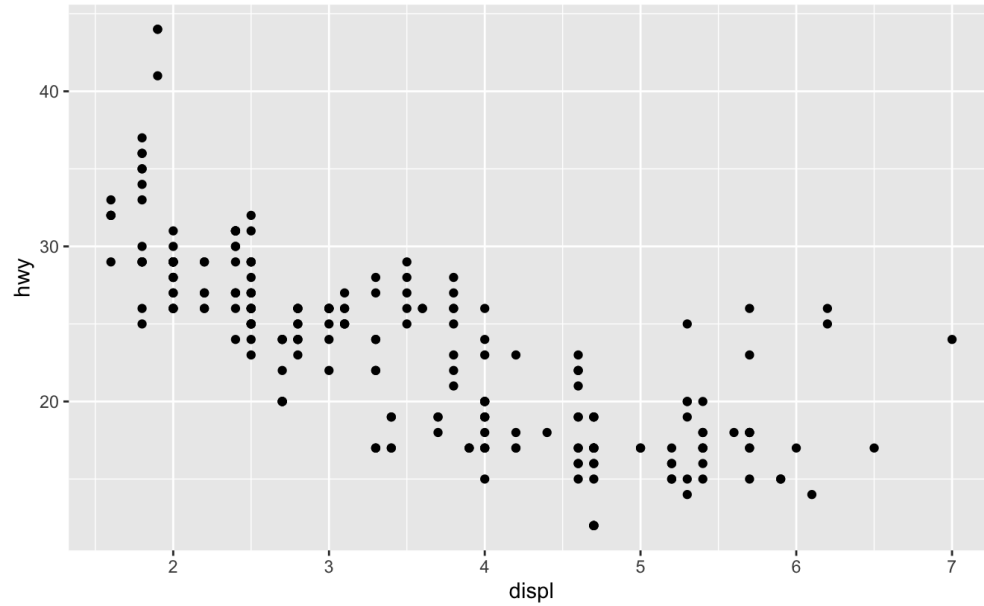
Para facetar seu gráfico na combinação de duas variáveis, adicione `facet_grid()` à sua chamada de gráfico. O primeiro argumento de `facet_grid()` também é uma fórmula. Desta vez, a fórmula deve conter dois nomes de variáveis separados por um `~`.

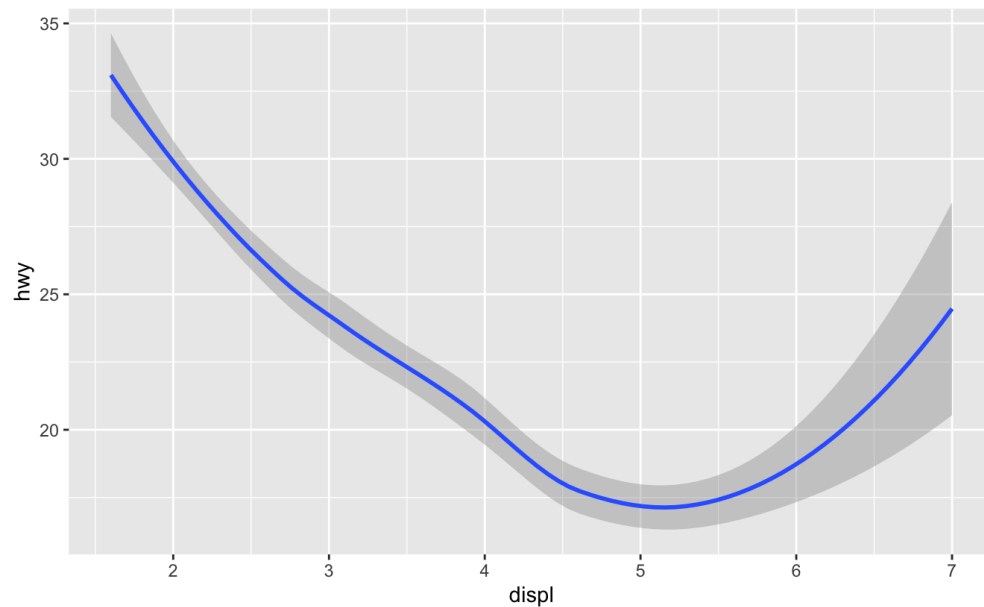
```
In [139]: ggplot(data = mpg) +  
  geom_point(mapping = aes(x = displ, y = hwy)) +  
  facet_grid(drv ~ cyl)
```



# 5 objetos geométricos (Geometric objects)

Como esses dois gráficos são semelhantes?





***Ambos os gráficos contêm a mesma variável x, a mesma variável y e ambos descrevem os mesmos dados. Mas os enredos não são idênticos.***

***Cada gráfico usa um objeto visual diferente para representar os dados. Na sintaxe ggplot2, dizemos que eles usam geoms diferentes.***

Um **geom** é o objeto geométrico que um gráfico usa para representar dados. Muitas vezes as pessoas descrevem plotagens pelo tipo de geom que a plotagem usa.

Por exemplo, gráficos de barras usam **geoms** de barra (bar geoms), gráficos de linha usam **geoms** de linha (line geoms), boxplots usam geom de boxplot e assim por diante.

Os gráficos de dispersão quebram a tendência; eles usam o ponto geom.

Como vimos acima, você pode usar diferentes geoms para plotar os mesmos dados.

O gráfico acima usa o point geom, e o gráfico abaixo usa o geom smooth, uma linha suave ajustada aos dados.

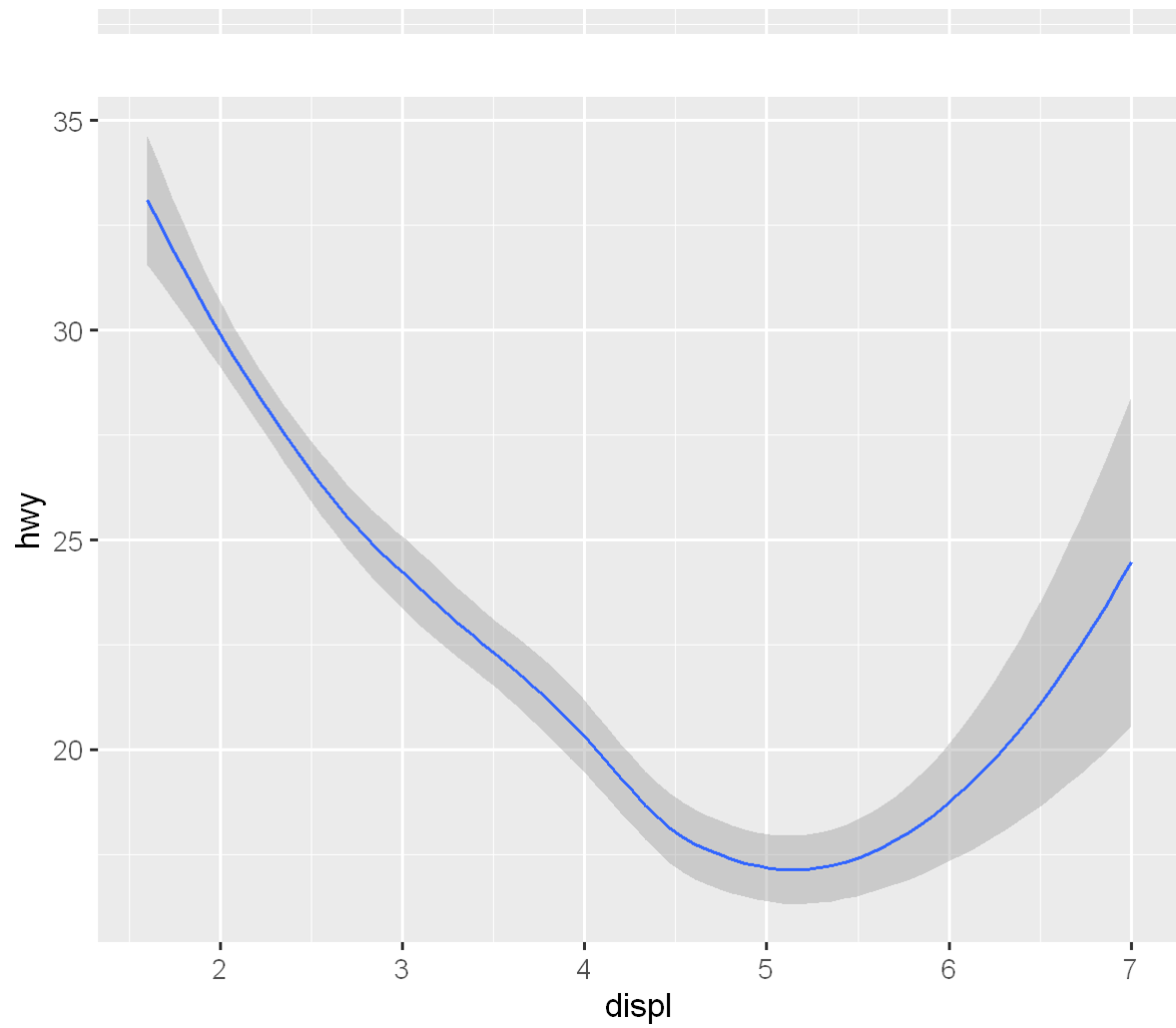
Para alterar o geom em seu gráfico, altere a função geom que você adiciona ao **ggplot()**. Por exemplo, para fazer os gráficos acima, você pode usar este código

```
In [140]: # superior
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy))

# inferior
ggplot(data = mpg) +
  geom_smooth(mapping = aes(x = displ, y = hwy))

`geom_smooth()` using method = 'loess' and formula 'y ~ x'
```





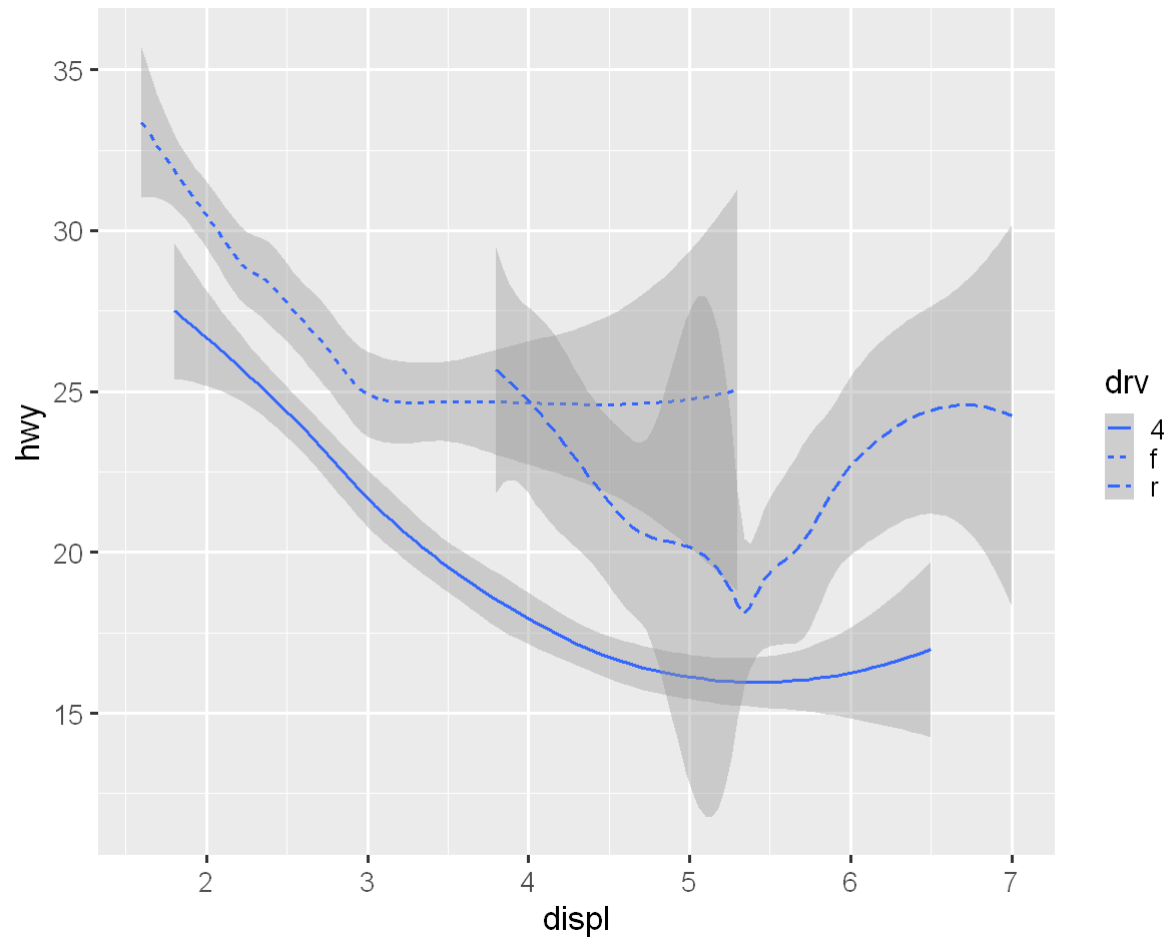
Cada função **geom** em **ggplot2** recebe um argumento de mapeamento.

No entanto, nem toda estética funciona com todos os geom. Você pode definir a forma de um ponto, mas não pode definir a "forma" de uma linha.

Por outro lado, você pode definir o tipo de linha de uma linha. `geom_smooth()` desenhara uma linha diferente, com um tipo de linha diferente, para cada valor único da variável que você mapeia para o tipo de linha.

```
In [141]: ggplot(data = mpg) +  
  geom_smooth(mapping = aes(x = displ, y = hwy, linetype = drv))
```

`geom\_smooth()` using method = 'loess' and formula 'y ~ x'



Aqui `geom_smooth ()` separa os carros em três linhas com base em seu valor `drv`, que descreve o trem de força de um carro.

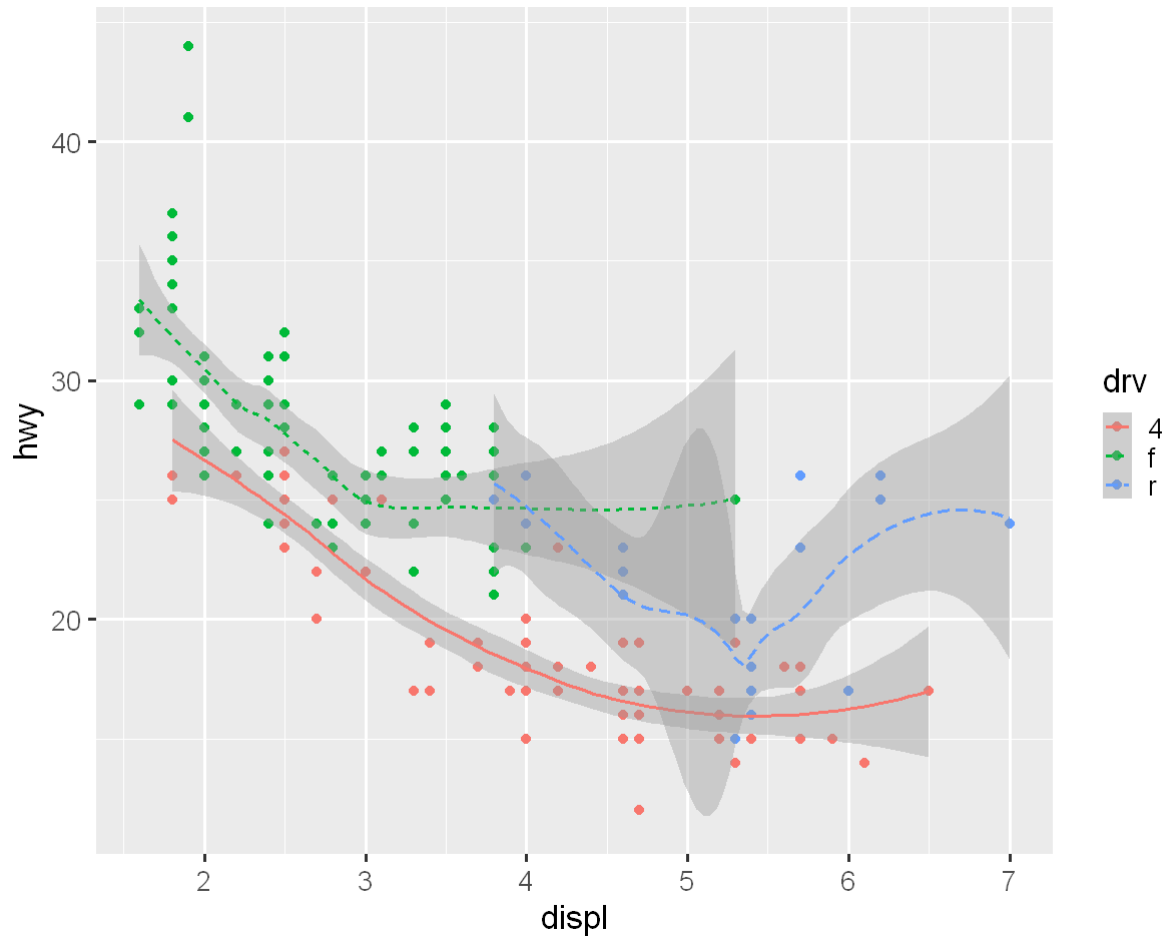
Uma linha descreve todos os pontos com um valor `4`, uma linha descreve todos os pontos com um valor `f` e uma linha descreve todos os pontos com um valor `r`.

Aqui, `4` significa tração nas quatro rodas, `f` significa tração nas rodas dianteiras e `r` para tração nas rodas traseiras.

Se isso soar estranho, podemos deixar mais claro sobrepondo as linhas sobre os dados brutos e, em seguida, colorindo tudo de acordo com `drv`.

```
In [142]: ggplot(data = mpg) +  
  geom_point(mapping = aes(x = displ, y = hwy, color = drv), size=2.5) +  
  geom_smooth(mapping = aes(x = displ, y = hwy, color = drv, linetype = drv))
```

`geom\_smooth()` using method = 'loess' and formula 'y ~ x'



ggplot2 fornece mais de 40 geoms e pacotes de extensão fornecem ainda mais (consulte [este link \(https://exts.ggplot2.tidyverse.org/gallery/\)](https://exts.ggplot2.tidyverse.org/gallery/) para obter uma amostra. A melhor maneira de obter uma visão geral abrangente é o cheatsheet ggplot2, que você pode encontrar [neste link \(http://rstudio.com/resources/cheatsheets\)](http://rstudio.com/resources/cheatsheets). Para aprender mais sobre geom use help: ?geom\_smooth .

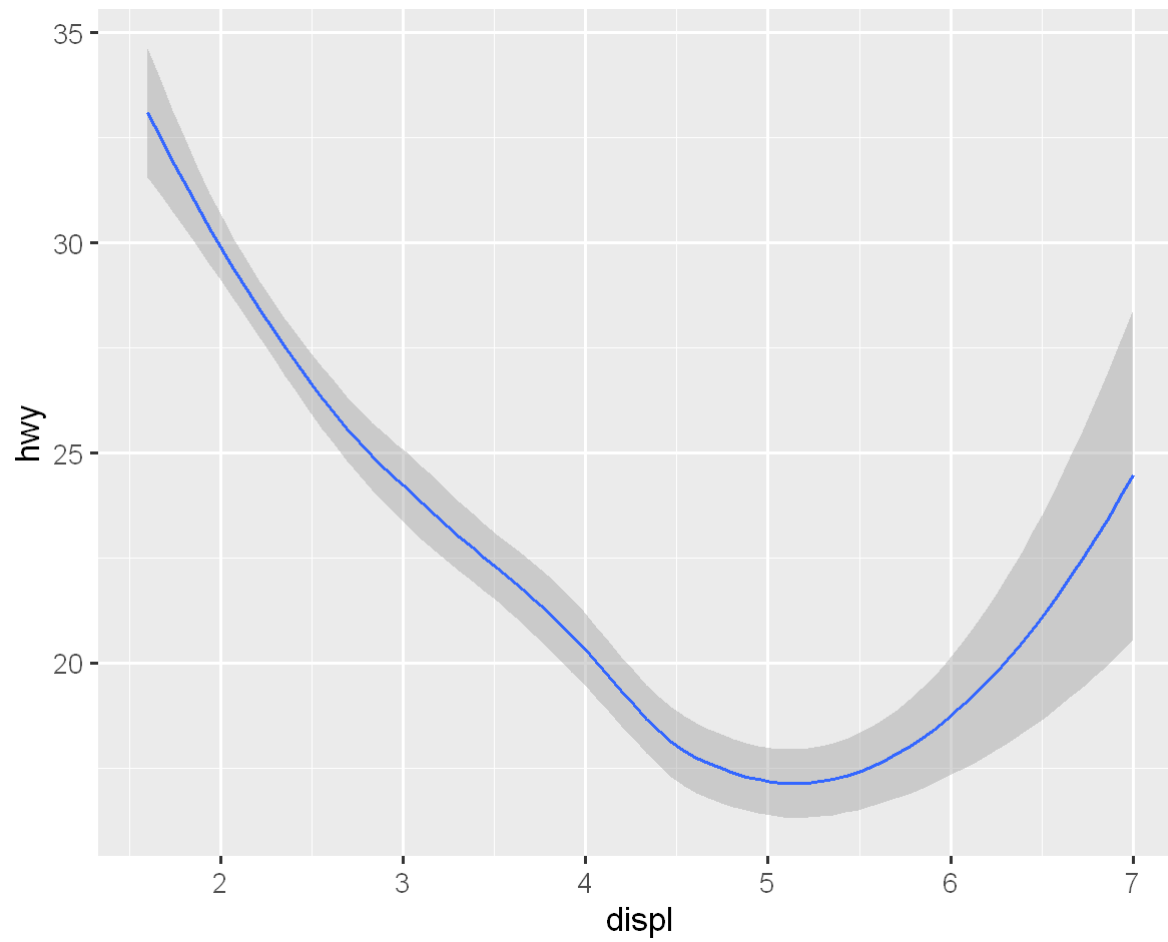
**Usando um Geom por gráfico**

```
In [143]: ggplot(data = mpg) +  
  geom_smooth(mapping = aes(x = displ, y = hwy))  
  
ggplot(data = mpg) +  
  geom_smooth(mapping = aes(x = displ, y = hwy, group = drv))  
  
ggplot(data = mpg) +  
  geom_smooth(  
    mapping = aes(x = displ, y = hwy, color = drv),  
    show.legend = FALSE  
  )
```

`geom\_smooth()` using method = 'loess' and formula 'y ~ x'

`geom\_smooth()` using method = 'loess' and formula 'y ~ x'



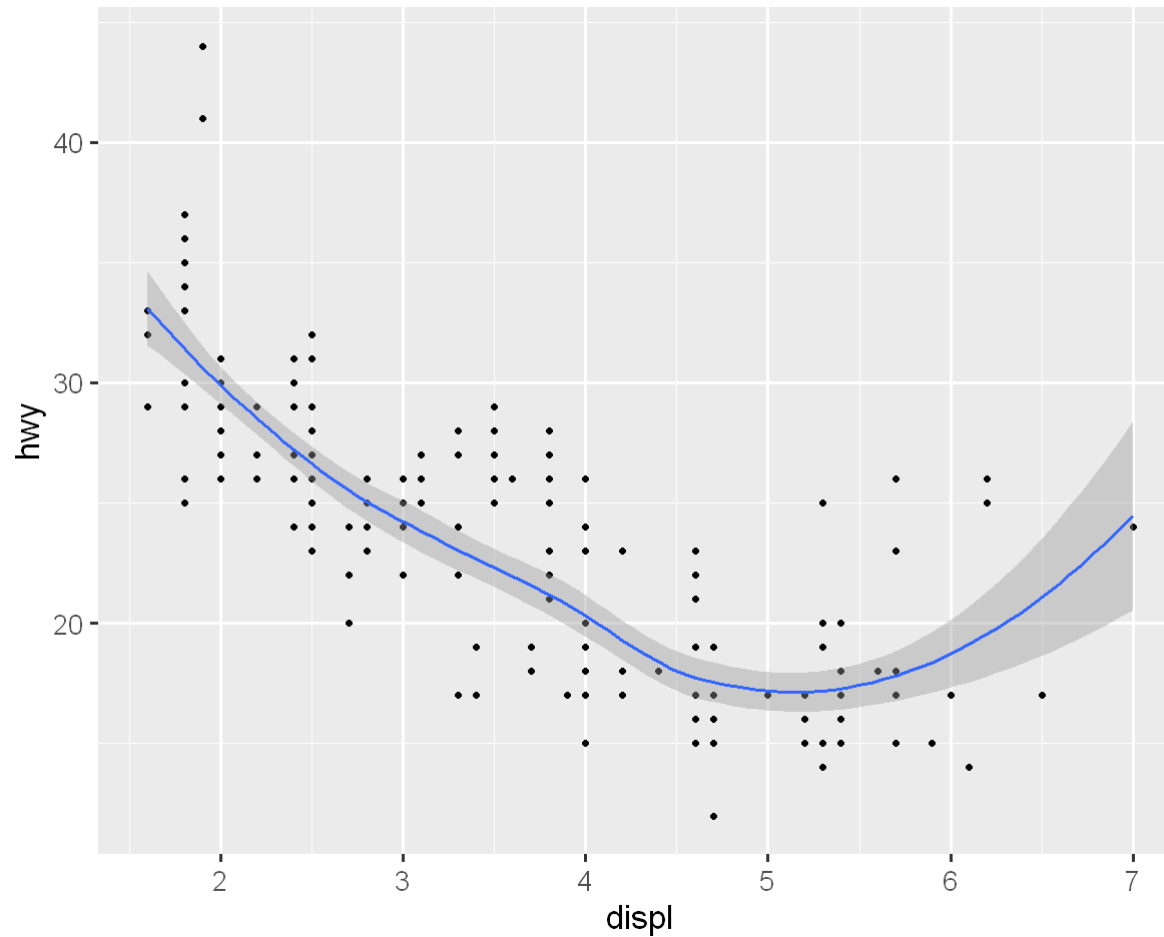


``geom_smooth()`` using method = 'loess' and formula 'y ~ x'

**Usando mais de um Geom por gráfico**

```
In [144]: ggplot(data = mpg) +  
  geom_point(mapping = aes(x = displ, y = hwy)) +  
  geom_smooth(mapping = aes(x = displ, y = hwy))
```

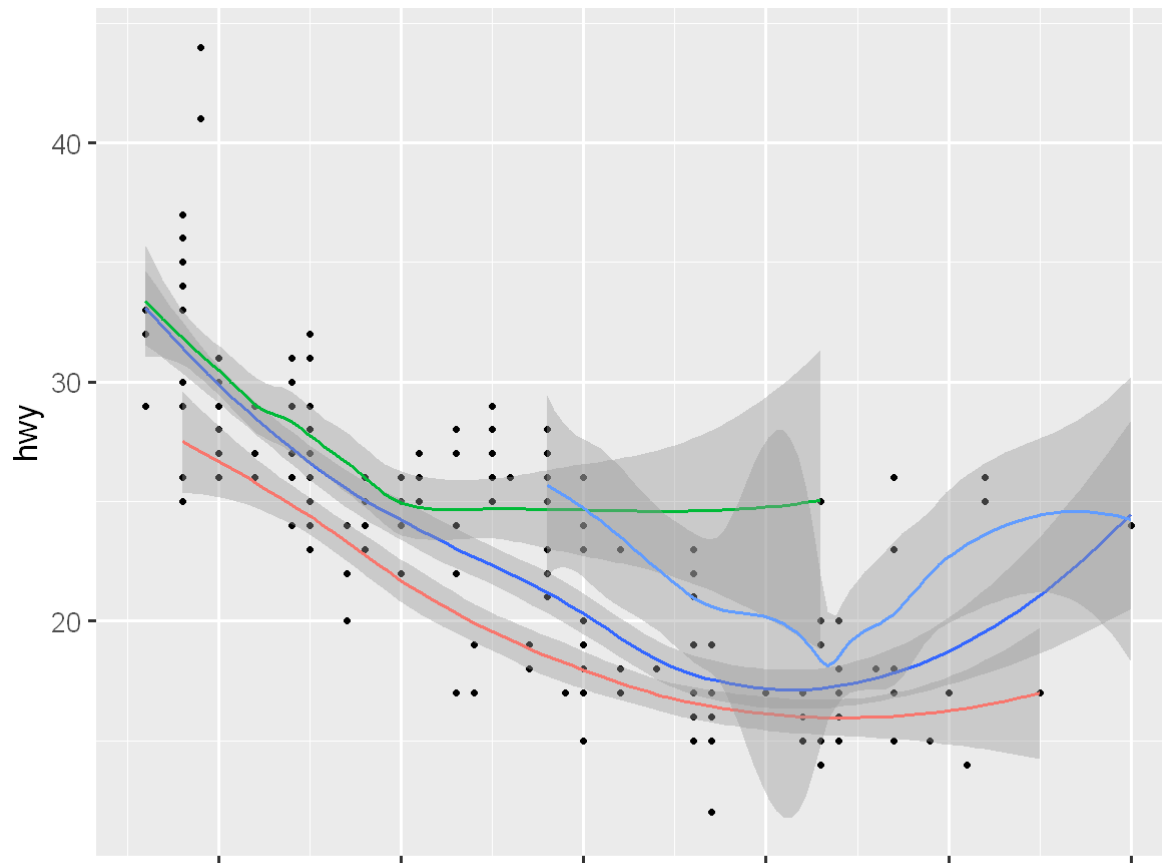
`geom\_smooth()` using method = 'loess' and formula 'y ~ x'



```
In [145]: ggplot(data = mpg) +  
  geom_point(mapping = aes(x = displ, y = hwy)) +  
  geom_smooth(mapping = aes(x = displ, y = hwy))+  
  geom_smooth(  
    mapping = aes(x = displ, y = hwy, color = drv),  
    show.legend = FALSE  
  )
```

`geom\_smooth()` using method = 'loess' and formula 'y ~ x'

`geom\_smooth()` using method = 'loess' and formula 'y ~ x'



2

3

4

displ

5

6

7

## 6 Transformações estatísticas

A seguir, vamos dar uma olhada em um gráfico de barras. Os gráficos de barras parecem simples, mas são interessantes porque revelam algo sutil sobre os gráficos.

Considere um gráfico de barras básico, desenhado com `geom_bar()`. O gráfico a seguir exibe o número total de diamantes no conjunto de dados de diamantes, agrupados por corte.

O conjunto de dados de diamantes vem em `ggplot2` e contém informações sobre ~ 54.000 diamantes, incluindo o preço, quilate, cor, clareza e corte de cada diamante.

O gráfico mostra que mais diamantes estão disponíveis com cortes de alta qualidade do que com cortes de baixa qualidade.

In [146]: diamonds

A tibble: 53940 × 10

carat	cut	color	clarity	depth	table	price	x	y	z
<dbl>	<ord>	<ord>	<ord>	<dbl>	<dbl>	<int>	<dbl>	<dbl>	<dbl>
0.23	Ideal	E	SI2	61.5	55	326	3.95	3.98	2.43
0.21	Premium	E	SI1	59.8	61	326	3.89	3.84	2.31
0.23	Good	E	VS1	56.9	65	327	4.05	4.07	2.31
0.29	Premium	I	VS2	62.4	58	334	4.20	4.23	2.63
0.31	Good	J	SI2	63.3	58	335	4.34	4.35	2.75
0.24	Very Good	J	VVS2	62.8	57	336	3.94	3.96	2.48
0.24	Very Good	I	VVS1	62.3	57	336	3.95	3.98	2.47
0.26	Very Good	H	SI1	61.9	55	337	4.07	4.11	2.53
0.22	Fair	E	VS2	65.1	61	337	3.87	3.78	2.49
0.23	Very Good	H	VS1	59.4	61	338	4.00	4.05	2.39
0.30	Good	J	SI1	64.0	55	339	4.25	4.28	2.73
0.23	Ideal	J	VS1	62.8	56	340	3.93	3.90	2.46
0.22	Premium	F	SI1	60.4	61	342	3.88	3.84	2.33
0.31	Ideal	J	SI2	62.2	54	344	4.35	4.37	2.71
0.20	Premium	E	SI2	60.2	62	345	3.79	3.75	2.27
0.32	Premium	E	I1	60.9	58	345	4.38	4.42	2.68
0.30	Ideal	I	SI2	62.0	54	348	4.31	4.34	2.68
0.30	Good	J	SI1	63.4	54	351	4.23	4.29	2.70
0.30	Good	J	SI1	63.8	56	351	4.23	4.26	2.71
0.30	Very Good	J	SI1	62.7	59	351	4.21	4.27	2.66
0.30	Good	I	SI2	63.3	56	351	4.26	4.30	2.71
0.23	Very Good	E	VS2	63.8	55	352	3.85	3.92	2.48
0.23	Very Good	H	VS1	61.0	57	353	3.94	3.96	2.41
0.31	Very Good	J	SI1	59.4	62	353	4.39	4.43	2.62
0.31	Very Good	J	SI1	58.1	62	353	4.44	4.47	2.59
0.23	Very Good	G	VVS2	60.4	58	354	3.97	4.01	2.41
0.24	Premium	I	VS1	62.5	57	355	3.97	3.94	2.47
0.30	Very Good	J	VS2	62.2	57	357	4.28	4.30	2.67
0.23	Very Good	D	VS2	60.5	61	357	3.96	3.97	2.40
0.23	Very Good	F	VS1	60.9	57	357	3.96	3.99	2.42

carat	cut	color	clarity	depth	table	price	x	y	z
0.70 ct>	Premium	E	SI1	60.5	58	2753	5.74	5.77	3.48
0.57	Premium	E	IF	59.8	60	2753	5.43	5.38	3.23
0.61	Premium	F	VVS1	61.8	59	2753	5.48	5.40	3.36
0.80	Good	G	VS2	64.2	58	2753	5.84	5.81	3.74
0.84	Good	I	VS1	63.7	59	2753	5.94	5.90	3.77
0.77	Ideal	E	SI2	62.1	56	2753	5.84	5.86	3.63
0.74	Good	D	SI1	63.1	59	2753	5.71	5.74	3.61
0.90	Very Good	J	SI1	63.2	60	2753	6.12	6.09	3.86
0.76	Premium	I	VS1	59.3	62	2753	5.93	5.85	3.49
0.76	Ideal	I	VVS1	62.2	55	2753	5.89	5.87	3.66
0.70	Very Good	E	VS2	62.4	60	2755	5.57	5.61	3.49
0.70	Very Good	E	VS2	62.8	60	2755	5.59	5.65	3.53
0.70	Very Good	D	VS1	63.1	59	2755	5.67	5.58	3.55
0.73	Ideal	I	VS2	61.3	56	2756	5.80	5.84	3.57
0.73	Ideal	I	VS2	61.6	55	2756	5.82	5.84	3.59
0.79	Ideal	I	SI1	61.6	56	2756	5.95	5.97	3.67
0.71	Ideal	E	SI1	61.9	56	2756	5.71	5.73	3.54
0.79	Good	F	SI1	58.1	59	2756	6.06	6.13	3.54
0.79	Premium	E	SI2	61.4	58	2756	6.03	5.96	3.68
0.71	Ideal	G	VS1	61.4	56	2756	5.76	5.73	3.53
0.71	Premium	E	SI1	60.5	55	2756	5.79	5.74	3.49
0.71	Premium	F	SI1	59.8	62	2756	5.74	5.73	3.43
0.70	Very Good	E	VS2	60.5	59	2757	5.71	5.76	3.47
0.70	Very Good	E	VS2	61.2	59	2757	5.69	5.72	3.49
0.72	Premium	D	SI1	62.7	59	2757	5.69	5.73	3.58
0.72	Ideal	D	SI1	60.8	57	2757	5.75	5.76	3.50
0.72	Good	D	SI1	63.1	55	2757	5.69	5.75	3.61
0.70	Very Good	D	SI1	62.8	60	2757	5.66	5.68	3.56
0.86	Premium	H	SI2	61.0	58	2757	6.15	6.12	3.74
0.75	Ideal	D	SI2	62.2	55	2757	5.83	5.87	3.64



Um dataframe com 53940 linhas e 8 variáveis:

preço - preço em dólares americanos (326— 18.823)

quilate - m peso do diamante (0,2-5,01)

corte - qualidade do corte (razoável, bom, muito bom, premium, ideal)

cor - cor do diamante, de D (melhor) a J (pior)

clareza - uma medição de quão claro é o diamante (I1 (pior), SI2, SI1, VS2, VS1, VVS2, VVS1, IF (melhor))

x - comprimento em mm (0–10,74)

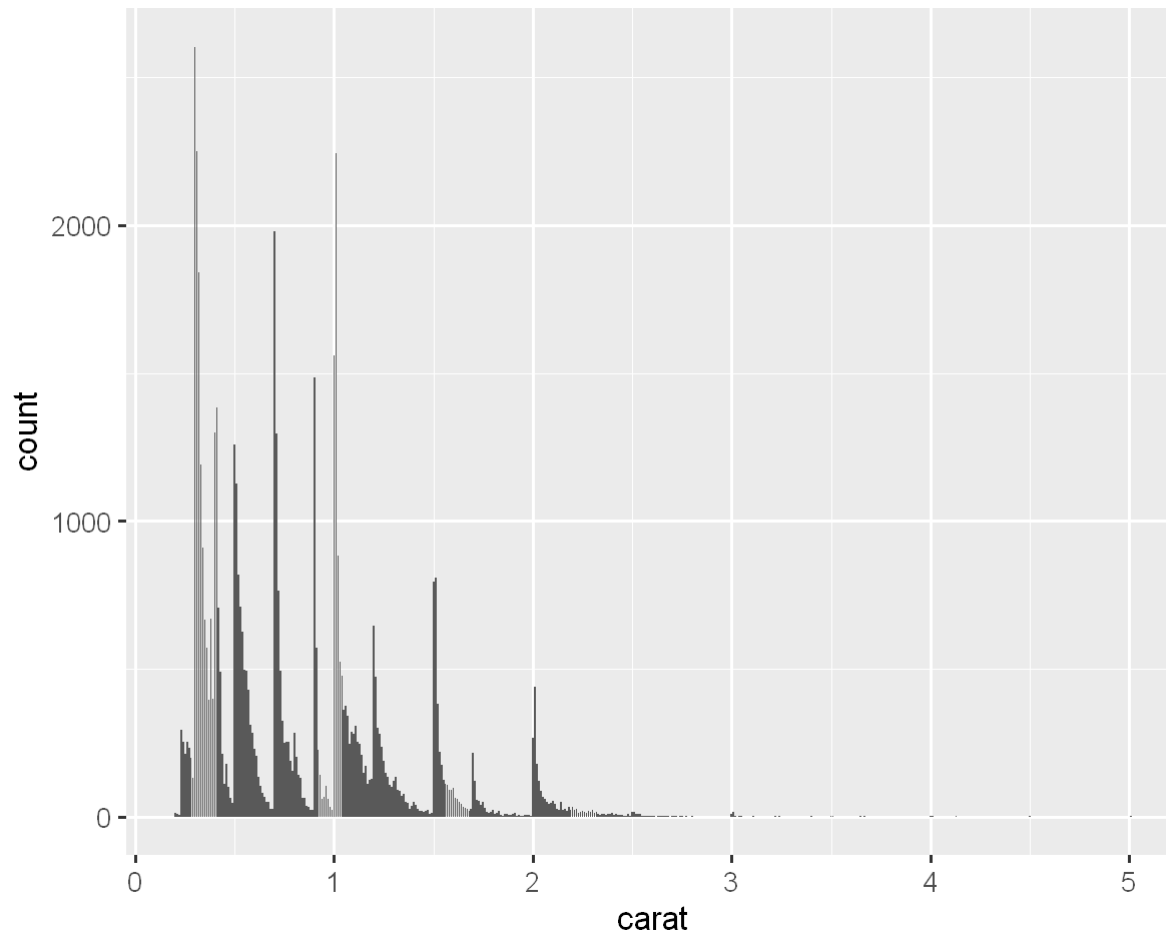
y - largura em mm (0–58,9)

z - profundidade em mm (0-31,8)

depth - porcentagem de profundidade total =  $z / \text{média}(x, y) = 2 * z / (x + y)$  (43-79)

table - largura do topo do diamante em relação ao ponto mais largo (43-95)

```
In [147]: ggplot(data = diamonds) +  
  geom_bar(mapping = aes(x = carat))
```



No eixo x, o gráfico exibe corte, uma variável de diamantes.

No eixo y, ele exibe a contagem, mas a contagem não é uma variável em diamantes! De onde vem a contagem? Muitos gráficos, como diagramas de dispersão, representam os valores brutos de seu conjunto de dados. Outros gráficos, como gráficos de barras, calculam novos valores para plotar:

Gráficos de barras, histogramas e polígonos de frequência armazenam seus dados e, a seguir, traçam as contagens de bin, o número de pontos que caem em cada bin.

Os smoths ajustam um modelo aos seus dados e, em seguida, representam as previsões do modelo.

Os boxplots computam um resumo robusto da distribuição e, em seguida, exibem uma caixa especialmente formatada.

O algoritmo usado para calcular novos valores para um gráfico é chamado de stat, abreviação de transformação estatística. A figura abaixo descreve como esse processo funciona com `geom_bar()`.

1. **geom\_bar()** begins with the **diamonds** data set

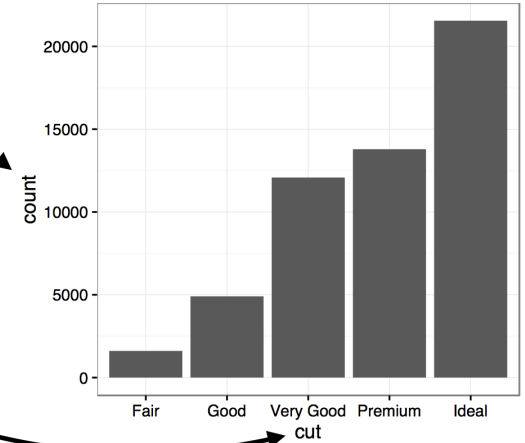
carat	cut	color	clarity	depth	table	price	x	y	z
0.23	Ideal	E	SI2	61.5	55	326	3.95	3.98	2.43
0.21	Premium	E	SI1	59.8	61	326	3.89	3.84	2.31
0.23	Good	E	VS1	56.9	65	327	4.05	4.07	2.31
0.29	Premium	I	VS2	62.4	58	334	4.20	4.23	2.63
0.31	Good	J	SI2	63.3	58	335	4.34	4.35	2.75
...	...	...	...	...	...	...	...	...	...

stat\_count()

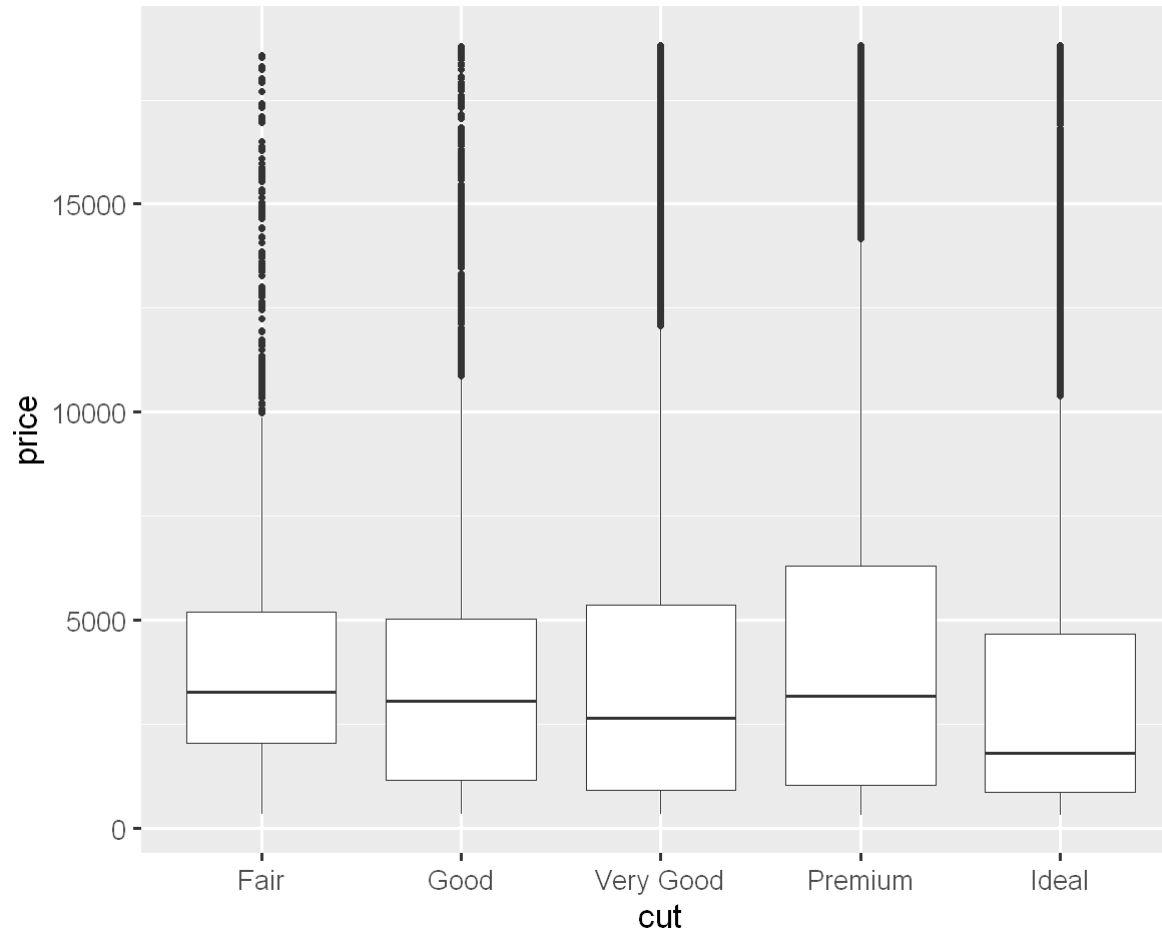
2. **geom\_bar()** transforms the data with the "count" stat, which returns a data set of cut values and counts.

cut	count	prop
Fair	1610	1
Good	4906	1
Very Good	12082	1
Premium	13791	1
Ideal	21551	1

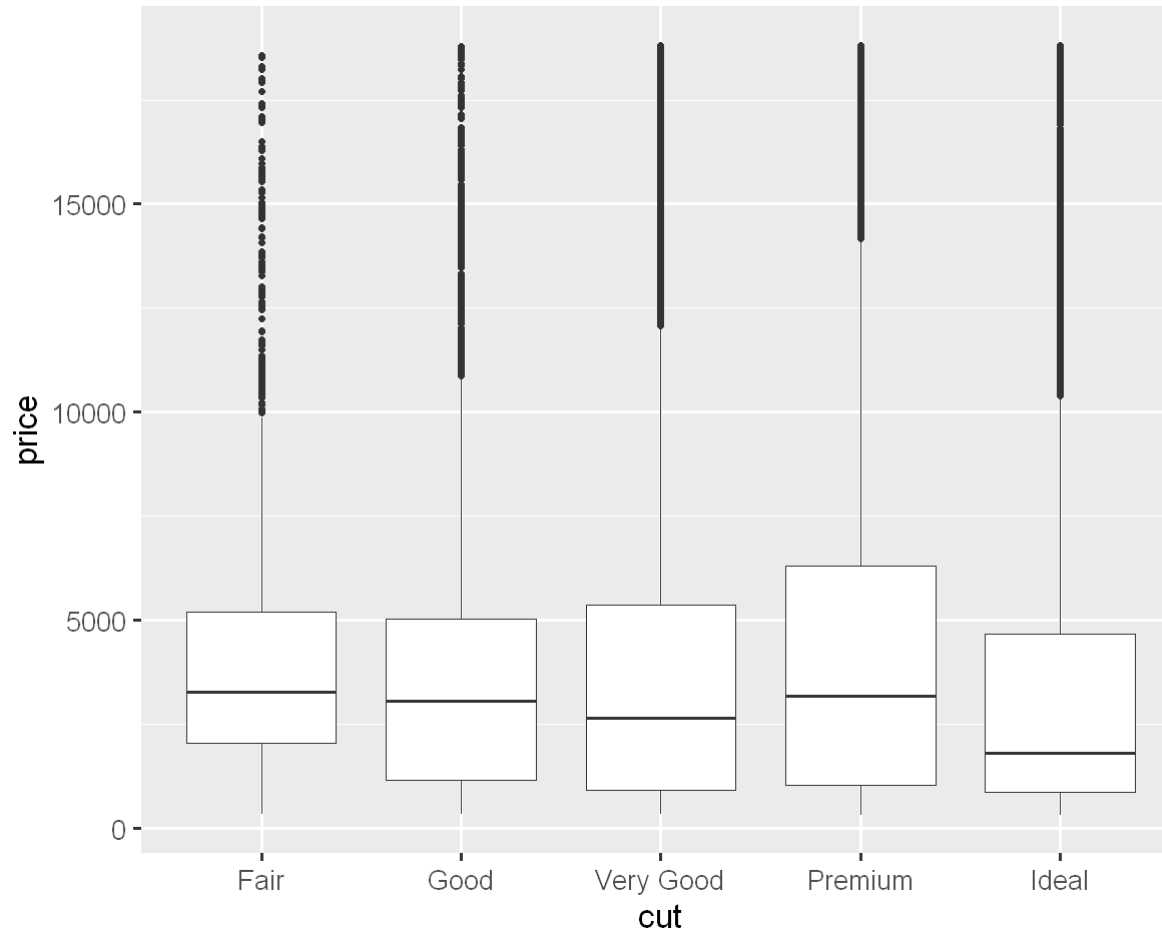
3. **geom\_bar()** uses the transformed data to build the plot. cut is mapped to the x axis, count is mapped to the y axis.



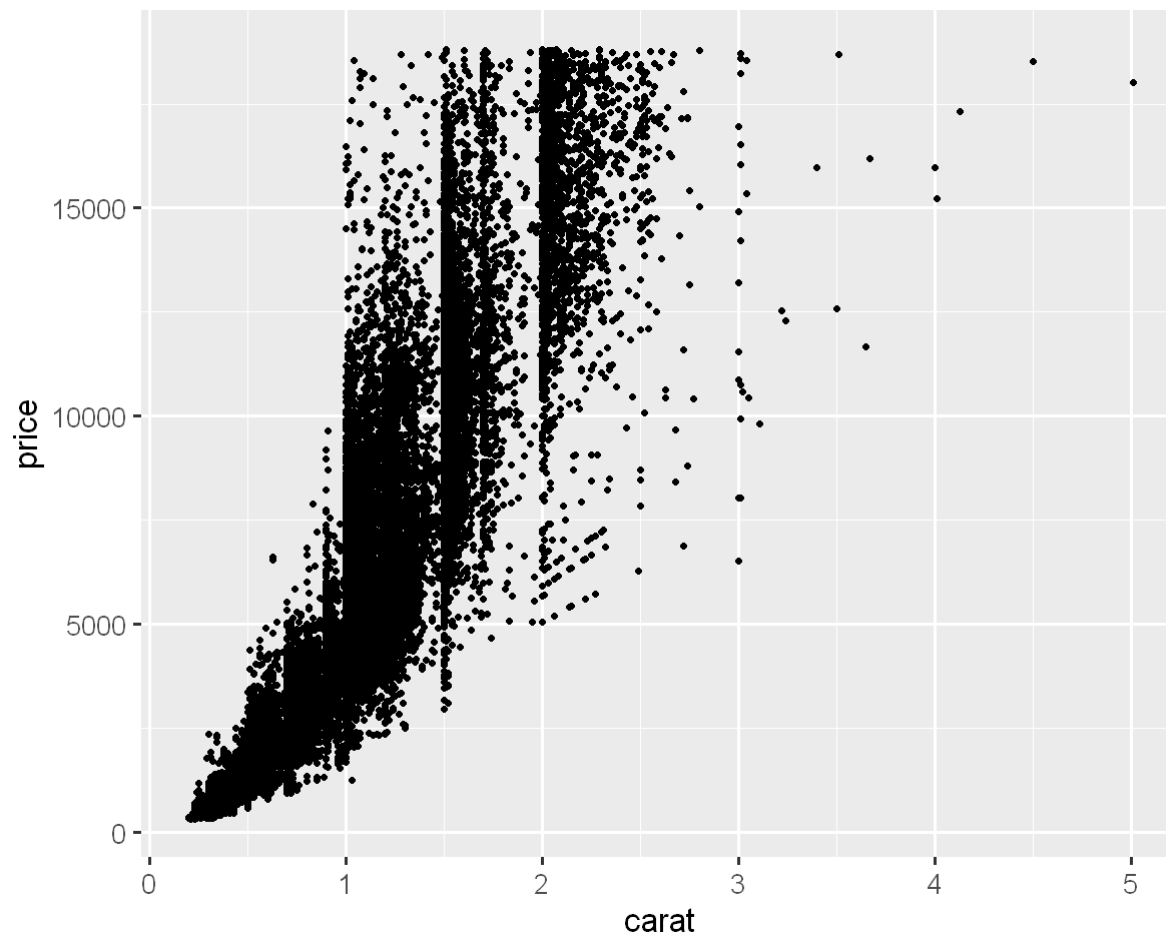
```
In [148]: ggplot(data = diamonds) +  
  geom_boxplot(mapping = aes(x = cut, y=price))
```



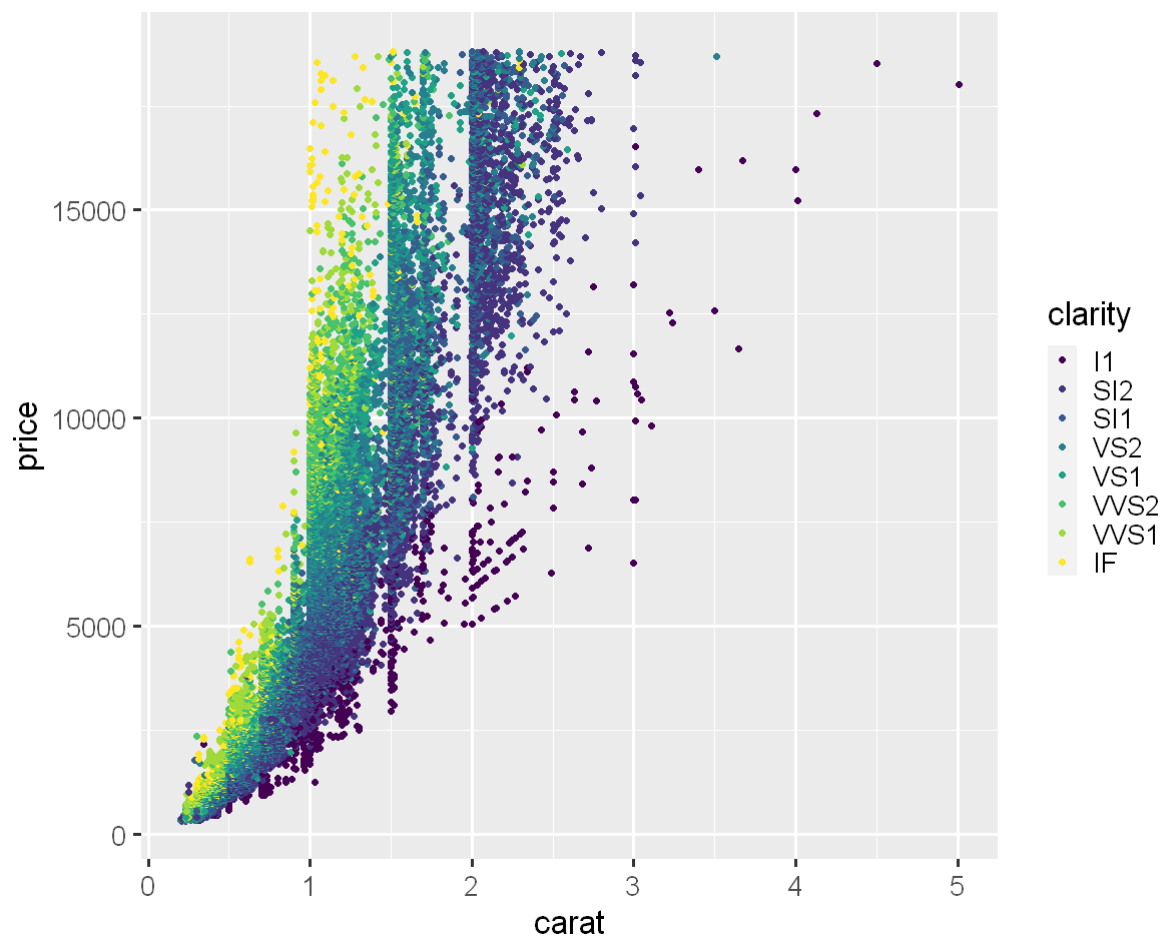
```
In [149]: ggplot(data = diamonds) +  
  geom_boxplot(mapping = aes(x = cut, y=price))
```



```
In [150]: ggplot(diamonds, aes(x=carat, y=price)) + geom_point()
```

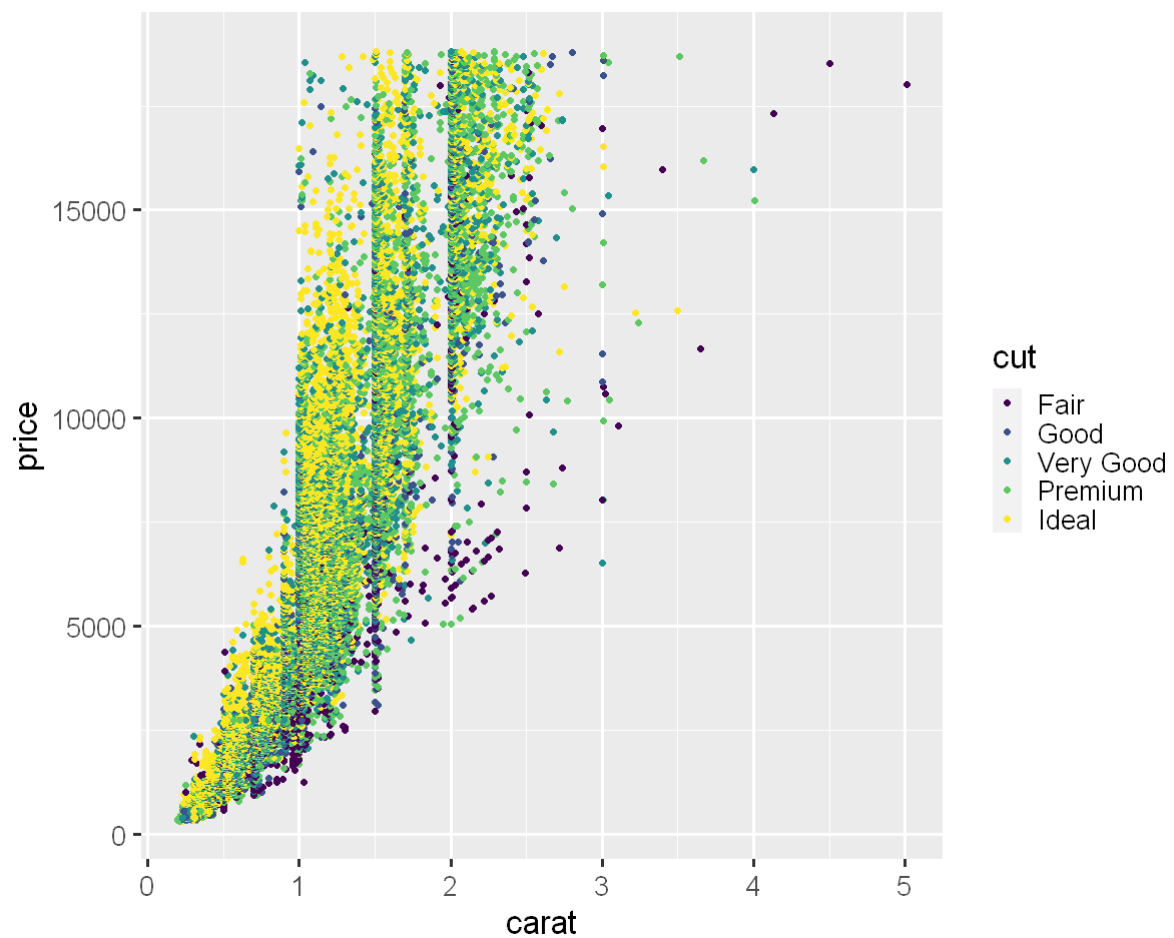


```
In [151]: ggplot(diamonds, aes(x=carat, y=price, color=clarity)) + geom_point(size=1.5)
```



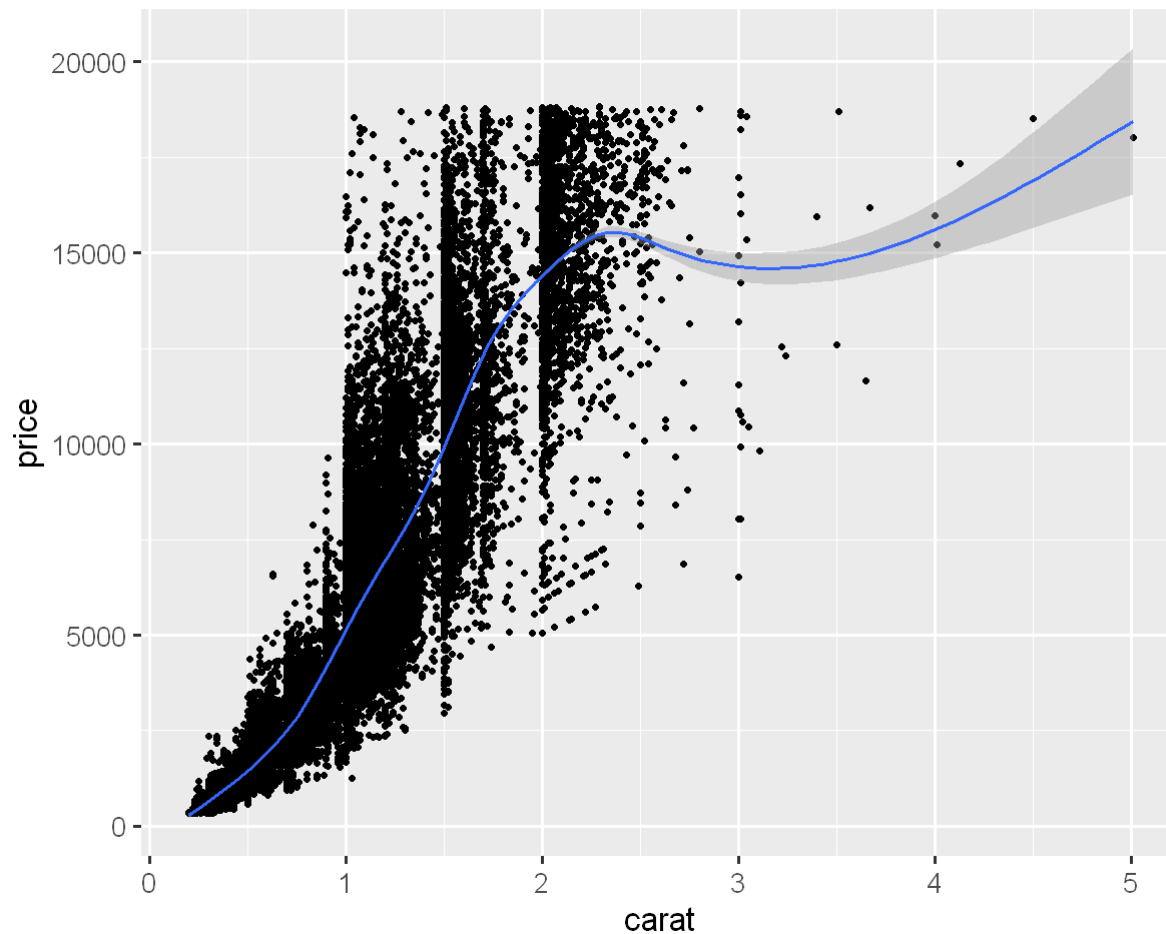


```
In [152]: ggplot(diamonds, aes(x=carat, y=price, color=cut)) + geom_point(size=1.5)
```



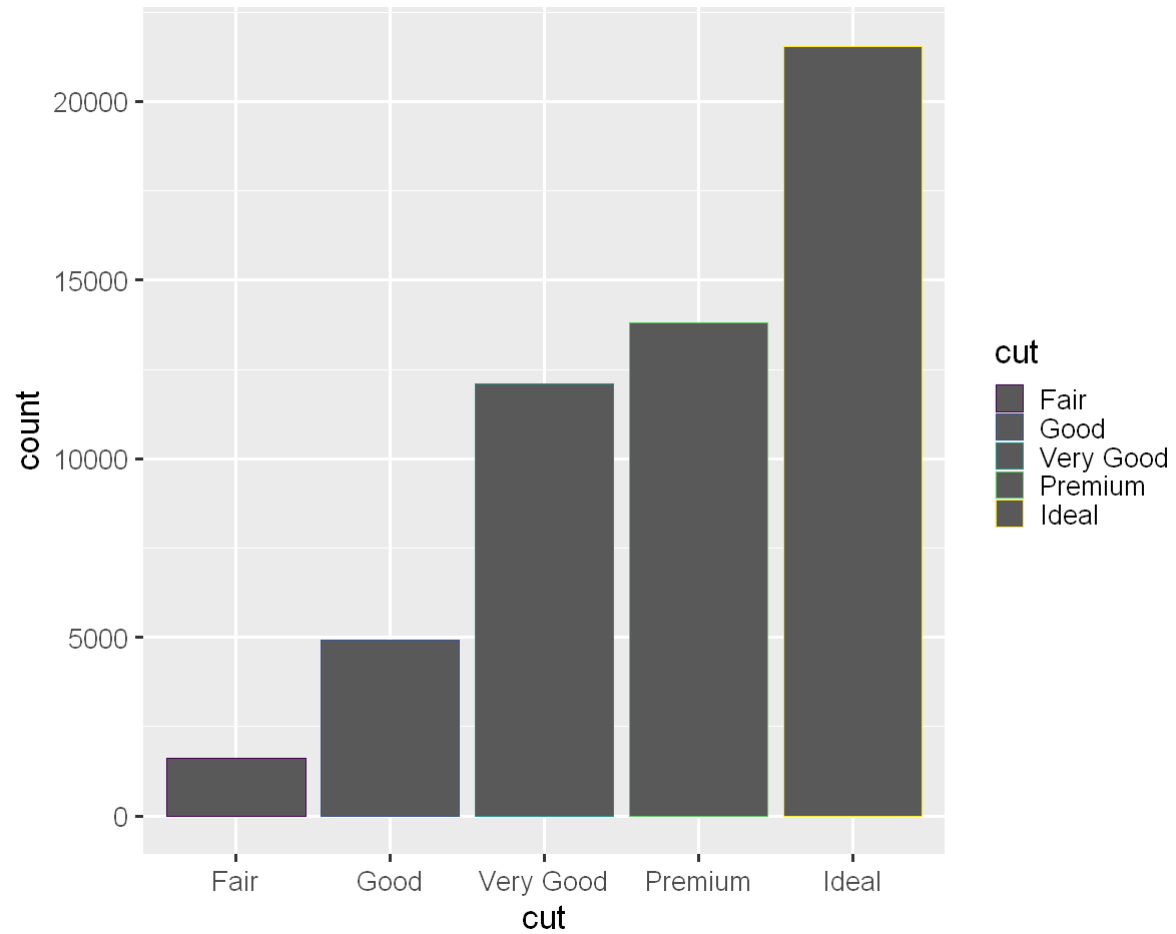
```
In [153]: ggplot(diamonds, aes(x=carat, y=price)) + geom_point() + geom_smooth()
```

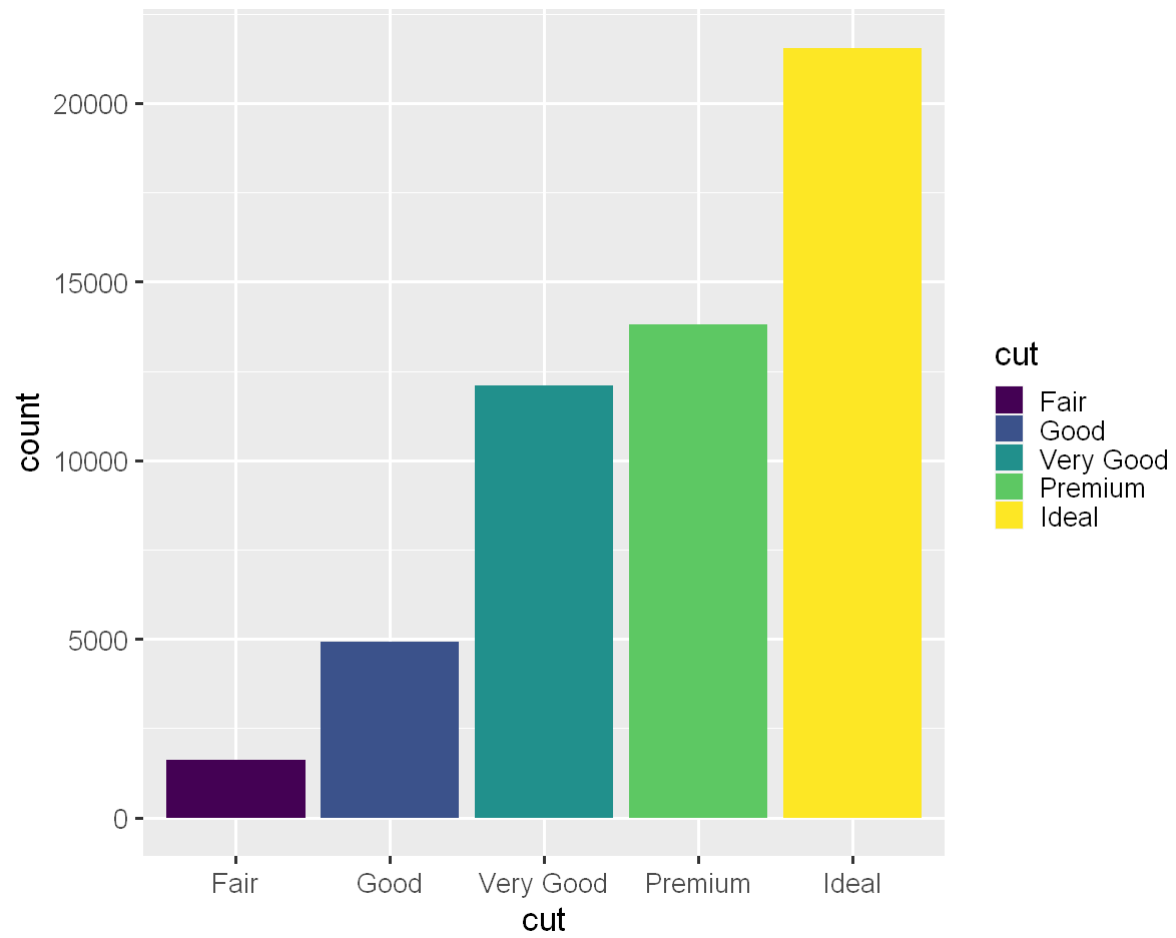
``geom_smooth()`` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'



7 ajustes de posição Há mais uma mágica associada aos gráficos de barras. Você pode colorir um gráfico de barras usando a estética de cores ou, mais útil, `fill` :

```
In [154]: ggplot(data = diamonds) +  
  geom_bar(mapping = aes(x = cut, colour = cut))  
ggplot(data = diamonds) +  
  geom_bar(mapping = aes(x = cut, fill = cut))
```

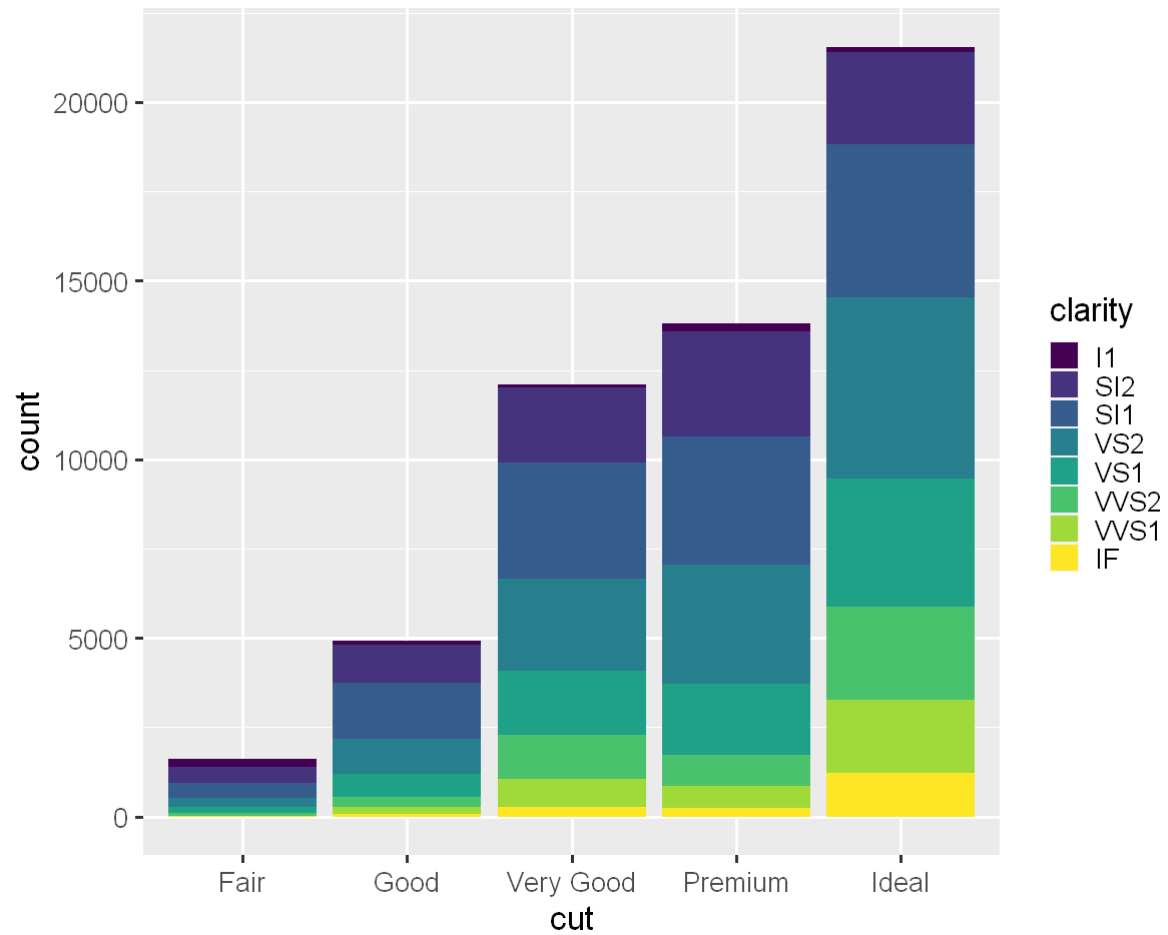




Observe o que acontece se você mapear a estética de preenchimento para outra variável, como clareza: as barras são empilhadas automaticamente. Cada retângulo colorido representa uma combinação de cut e clarify.

Observe o que acontece se você mapear a estética de preenchimento para outra variável, como clareza: as barras são empilhadas automaticamente. Cada retângulo colorido representa uma combinação de corte e clareza.

```
In [155]: ggplot(data = diamonds) +  
          geom_bar(mapping = aes(x = cut, fill = clarity))
```

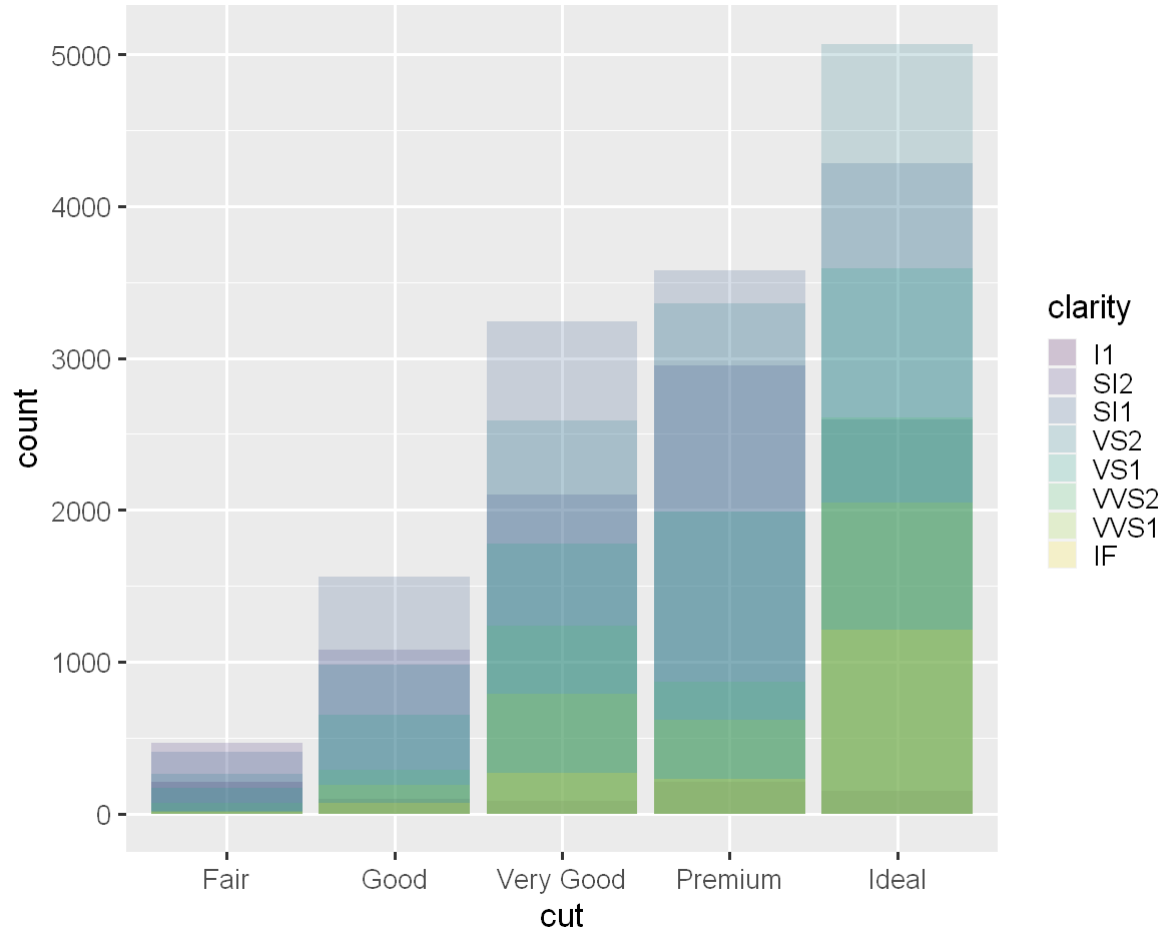


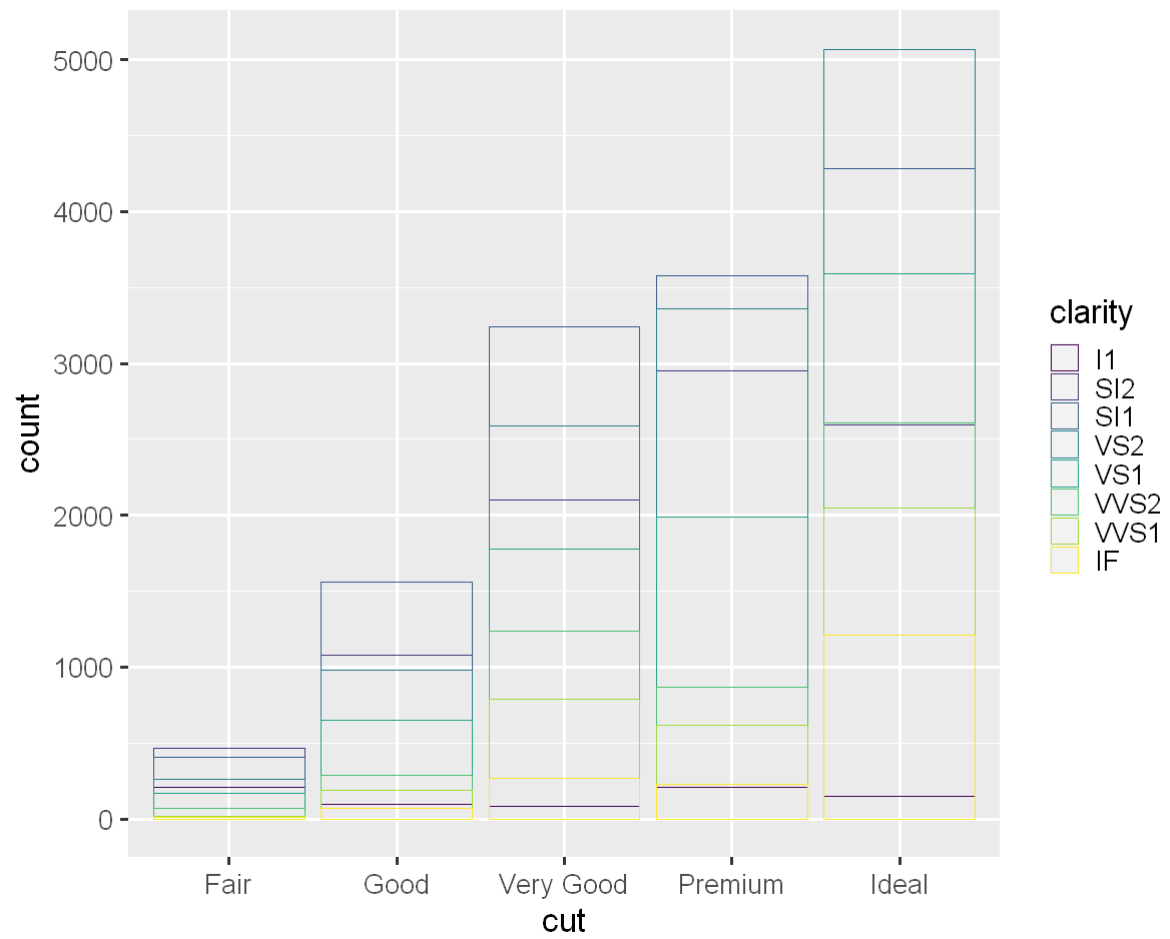


O empilhamento é executado automaticamente pelo ajuste de posição especificado pelo argumento de posição. Se você não quiser um gráfico de barras empilhadas, pode usar uma das três outras opções: "identify", "dodge" ou "fill".

`position = "identity"` colocará cada objeto exatamente onde ele se encaixa no contexto do gráfico. Isso não é muito útil para barras, porque as sobrepõe. Para ver essa sobreposição, precisamos tornar as barras ligeiramente transparentes, definindo `alfa` para um valor pequeno, ou completamente transparentes, definindo `fill = NA`.

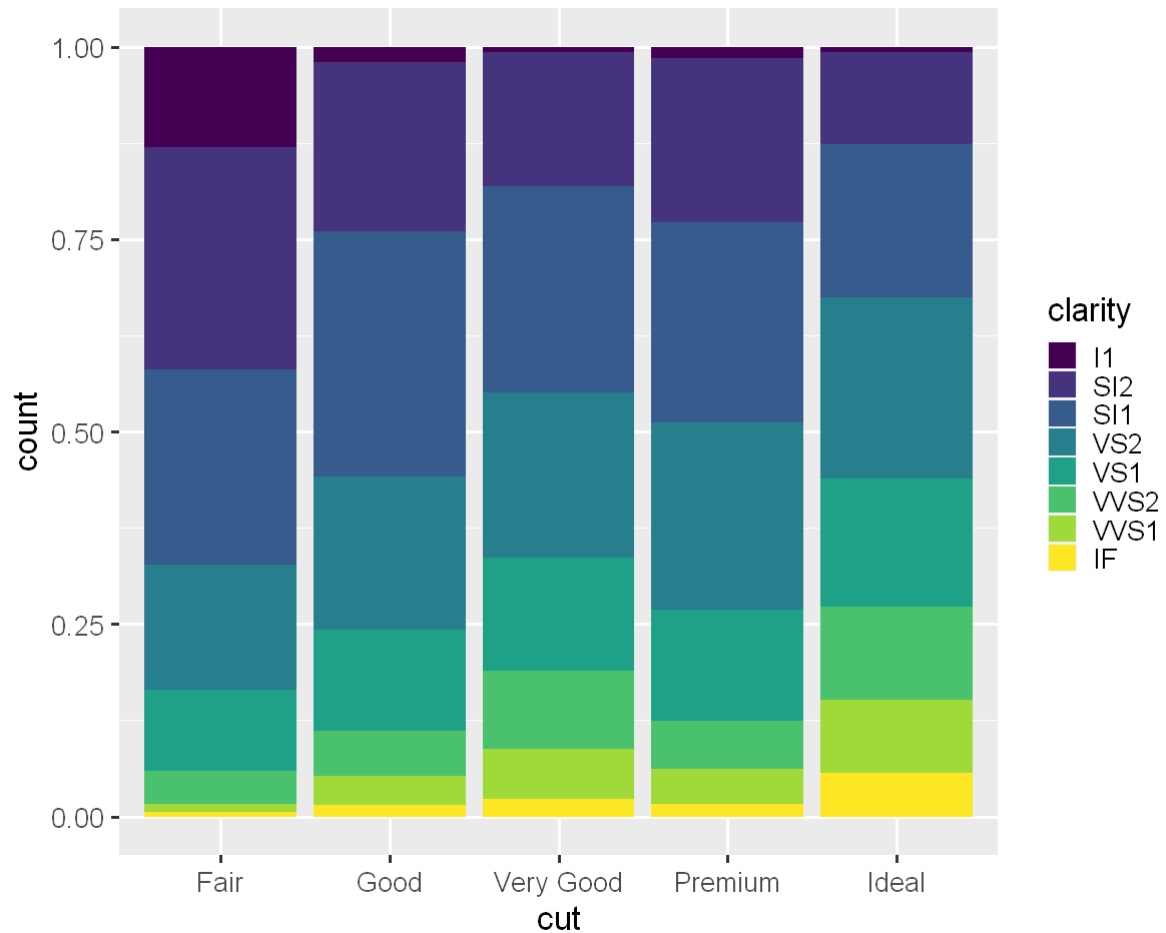
```
In [156]: ggplot(data = diamonds, mapping = aes(x = cut, fill = clarity)) +
  geom_bar(alpha = 1/5, position = "identity")
ggplot(data = diamonds, mapping = aes(x = cut, colour = clarity)) +
  geom_bar(fill = NA, position = "identity")
```





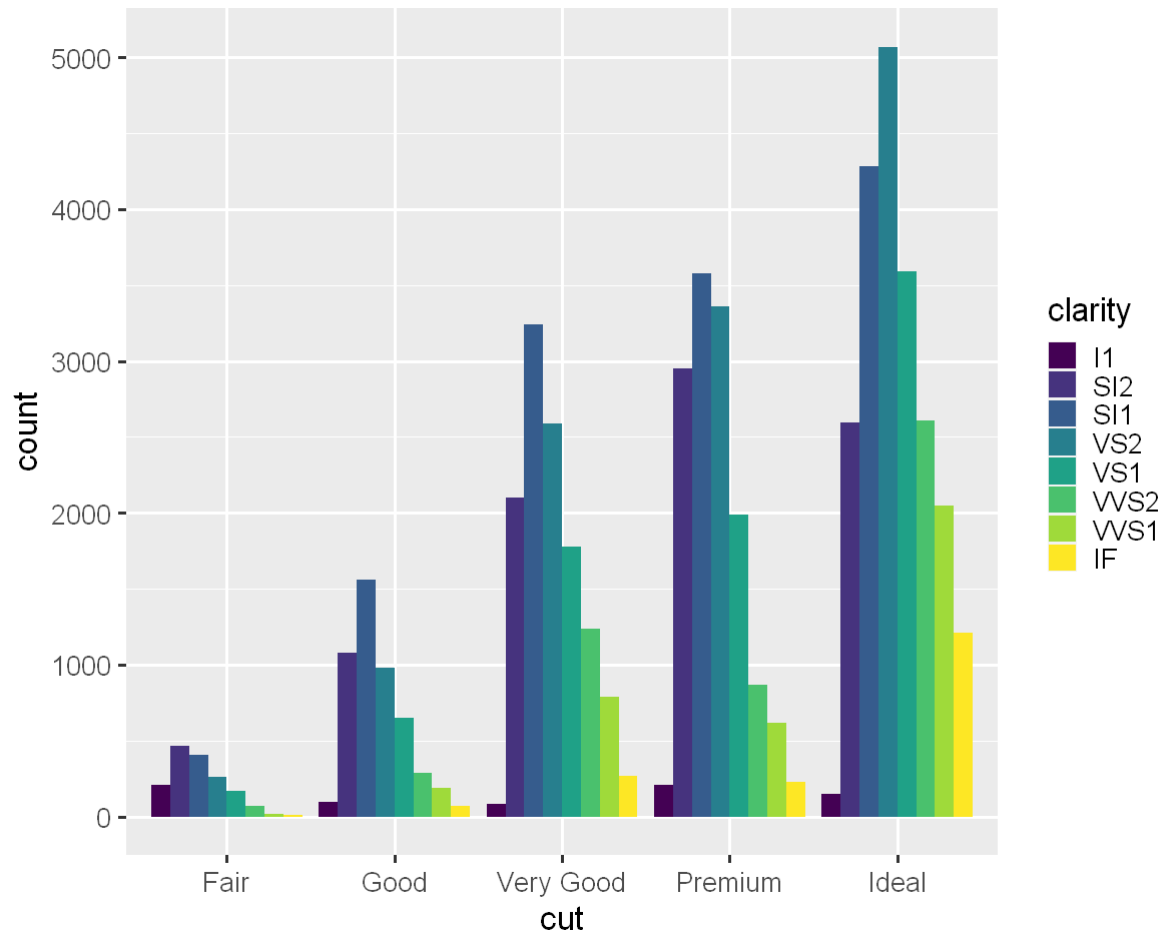
`position = "fill"` funciona como empilhamento, mas faz com que cada conjunto de barras empilhadas tenha a mesma altura. Isso torna mais fácil comparar proporções entre grupos.

```
In [157]: ggplot(data = diamonds) +  
  geom_bar(mapping = aes(x = cut, fill = clarity), position = "fill")
```



`position = "dodge"` coloca objetos sobrepostos diretamente um ao lado do outro. Isso torna mais fácil comparar valores individuais.

```
In [158]: ggplot(data = diamonds) +  
  geom_bar(mapping = aes(x = cut, fill = clarity), position = "dodge")
```



## 8 Sistemas de coordenadas

Os sistemas de coordenadas são provavelmente a parte mais complicada do ggplot2.

O sistema de coordenadas padrão é o sistema de coordenadas cartesianas, onde as posições x e y atuam independentemente para determinar a localização de cada ponto.

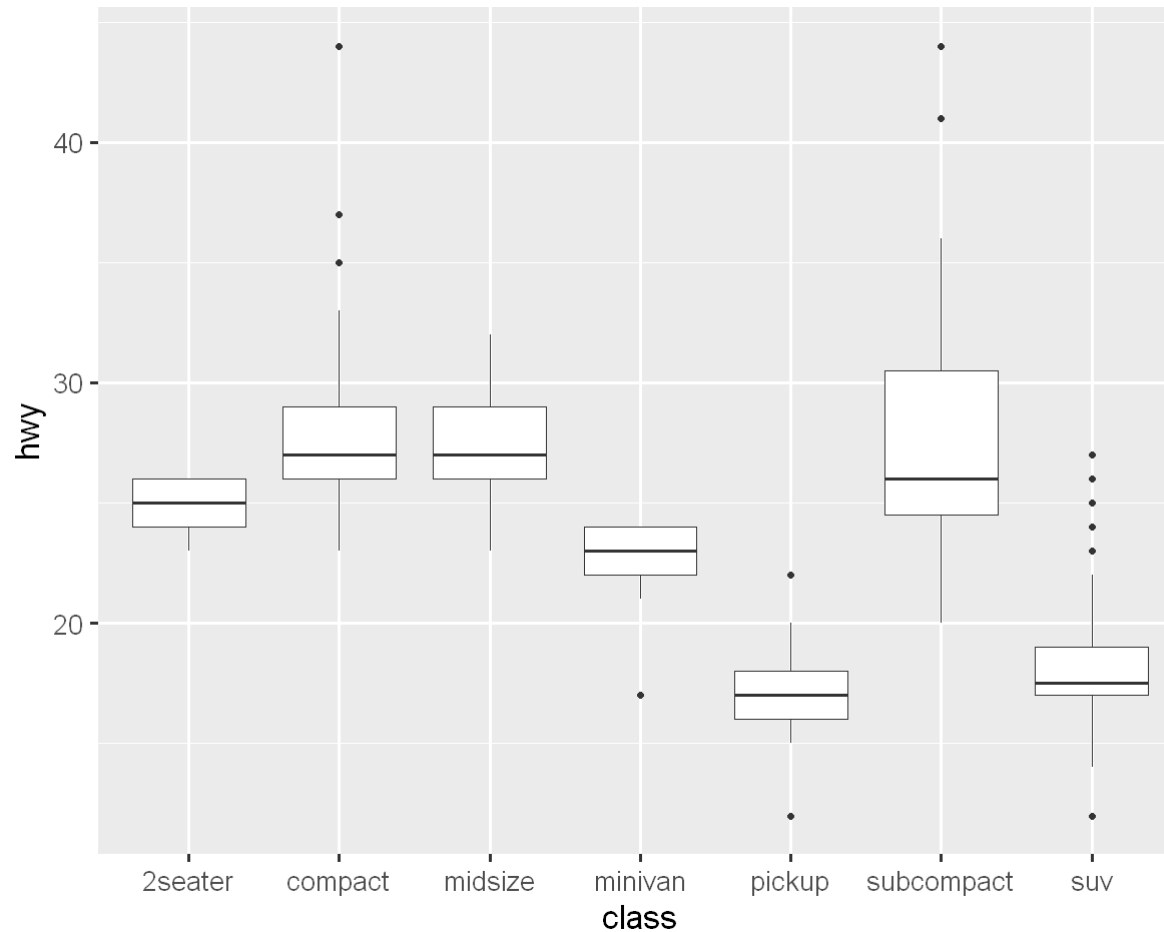
Existem vários outros sistemas de coordenadas que ocasionalmente são úteis.

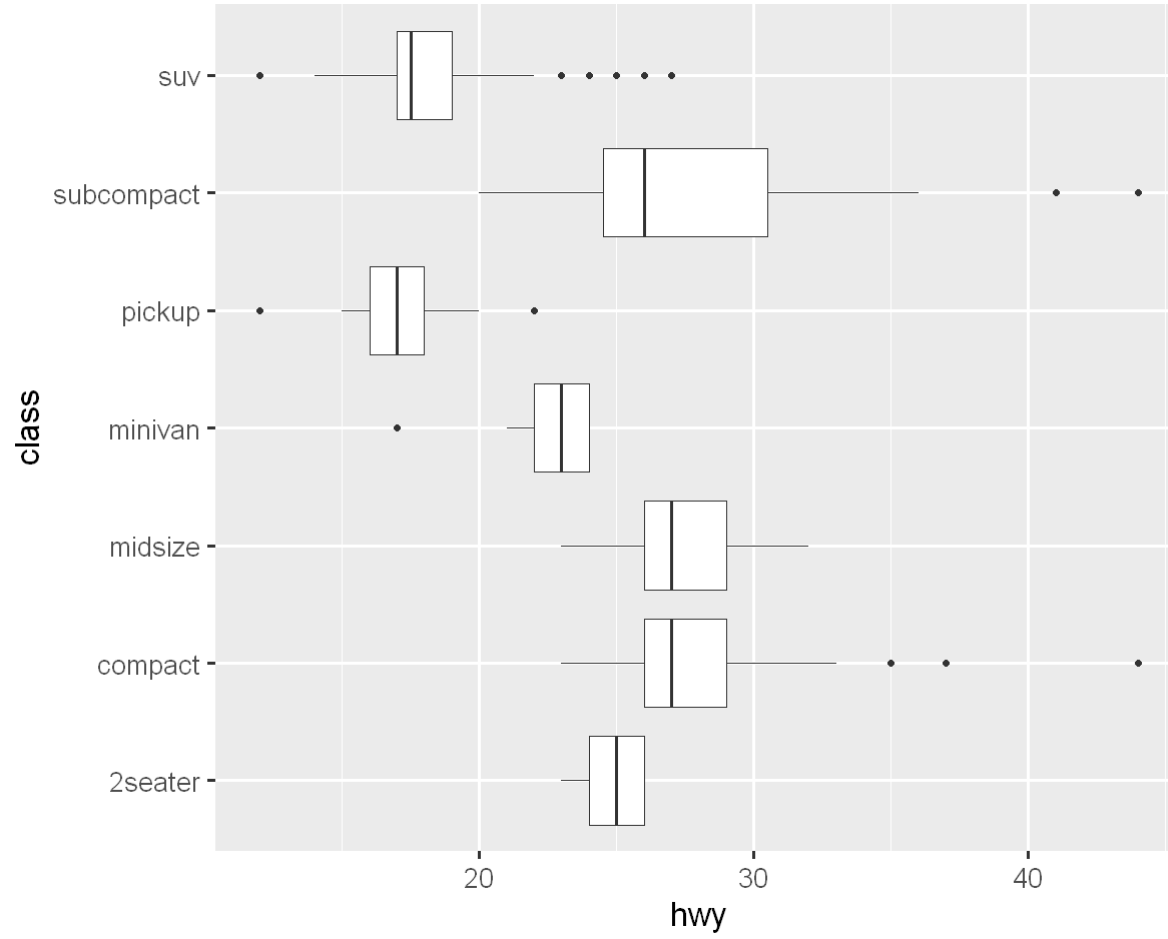
`coord_flip()` muda os eixos x e y. Isso é útil (por exemplo), se você quiser boxplots horizontais.

Também é útil para rótulos longos: é difícil fazer com que eles se encaixem sem se sobrepor no eixo x.



```
In [110]: ggplot(data = mpg, mapping = aes(x = class, y = hwy)) +  
  geom_boxplot()  
ggplot(data = mpg, mapping = aes(x = class, y = hwy)) +  
  geom_boxplot() +  
  coord_flip()
```





`coord_quickmap()` define a relação de aspecto corretamente para mapas. Isso é muito importante se você estiver traçando dados espaciais com `ggplot2` (que infelizmente não temos espaço para cobrir neste curso). mas vale como curiosidade

```
In [159]: install.packages('maps')
```

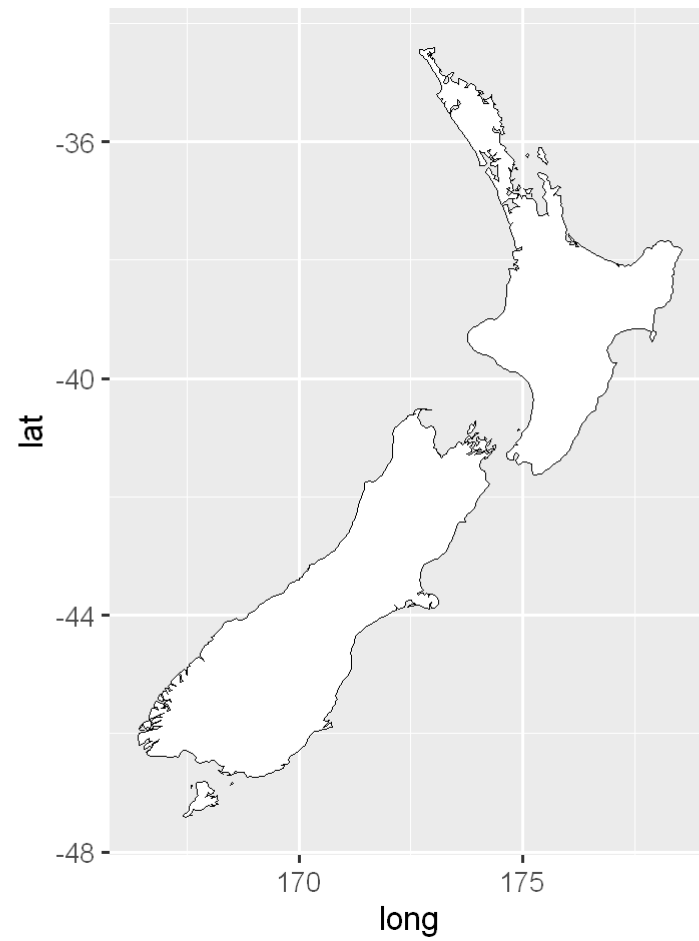
Warning message:

"package 'maps' is in use and will not be installed"

```
In [161]: library("maps")
          nz <- map_data("nz")

          ggplot(nz, aes(long, lat, group = group)) +
            geom_polygon(fill = "white", colour = "black")

          ggplot(nz, aes(long, lat, group = group)) +
            geom_polygon(fill = "white", colour = "black") +
            coord_quickmap()
```

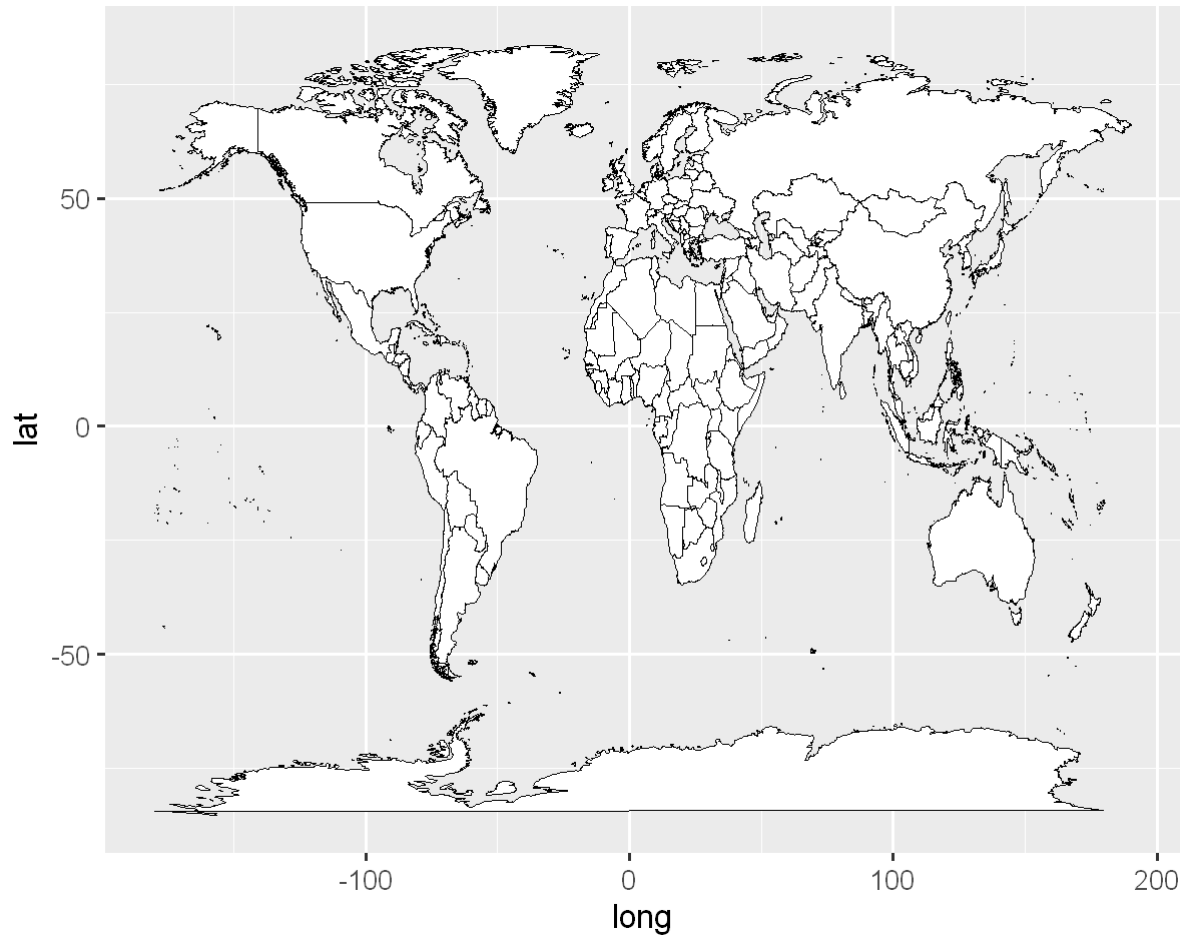


```
In [162]: help(map_data)
```

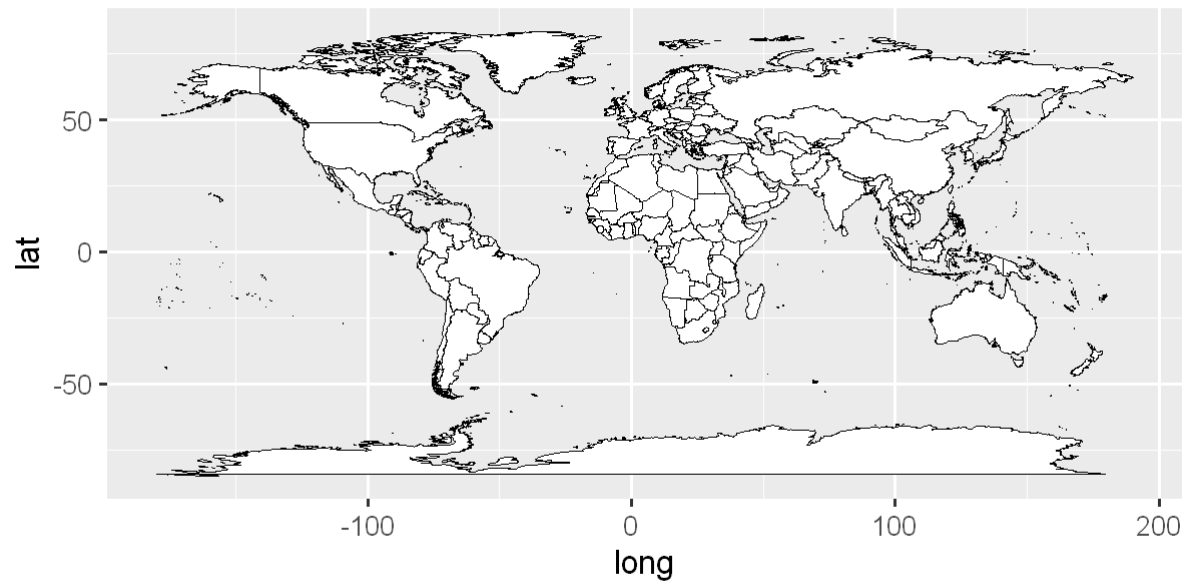
```
In [163]: wd <- map_data("world")

ggplot(wd, aes(long, lat, group = group)) +
  geom_polygon(fill = "white", colour = "black")

ggplot(wd, aes(long, lat, group = group)) +
  geom_polygon(fill = "white", colour = "black") +
  coord_quickmap()
```







`coord_polar()` uses polar coordinates. Polar coordinates reveal an interesting connection between a bar chart and a Coxcomb chart.

```
In [164]: bar <- ggplot(data = diamonds) +  
  geom_bar(  
    mapping = aes(x = cut, fill = cut),  
    show.legend = FALSE,  
    width = 1  
  ) +  
  theme(aspect.ratio = 1) +  
  labs(x = NULL, y = NULL)  
  
bar + coord_flip()  
bar + coord_polar()
```

