Session 2

Visualization in R

Youth Impact R Workshop Series

Zezhen Wu

Agenda

- Understanding `ggplot2` syntax
- Getting interactive plots in `plotly` from `ggplot2`
- A primer on interactive visualization in R `shiny`

Learning tools

- A Tableau-like dashboard of plotting in R: `esquisse::esquisser()`
- ggplot2: Elegant Graphics for Data Analysis (3e)
- The R Graph Gallery
- Plotly R Open Source Graphing Library
- Mastering Shiny
- Statistical Inference via Data Science: A ModernDive into R and the Tidyverse
- ChatGPT is particularly good at generating `ggplot2` and `shiny` codes, if you provide it with adequate descriptions.

What is a ggplot?

- `ggplot2` is a powerful and flexible data visualization package for R. It is a part of `tidyverse`.
- It is based on the Grammar of Graphics, a theoretical framework that describes and breaks down graphs into discrete components.
- With `ggplot2`, you can create complex multi-layered graphics from data in a structured and clear manner.
- `ggplot2` is installed as part of `tidyverse`, so you don't need to reinstall it. `library(tidyverse)` also loads `ggplot2` along with other important data wrangling packages.

```
# install.packages("ggplot2")
# library(tidyverse)
library(ggplot2)
```

Load an example dataset

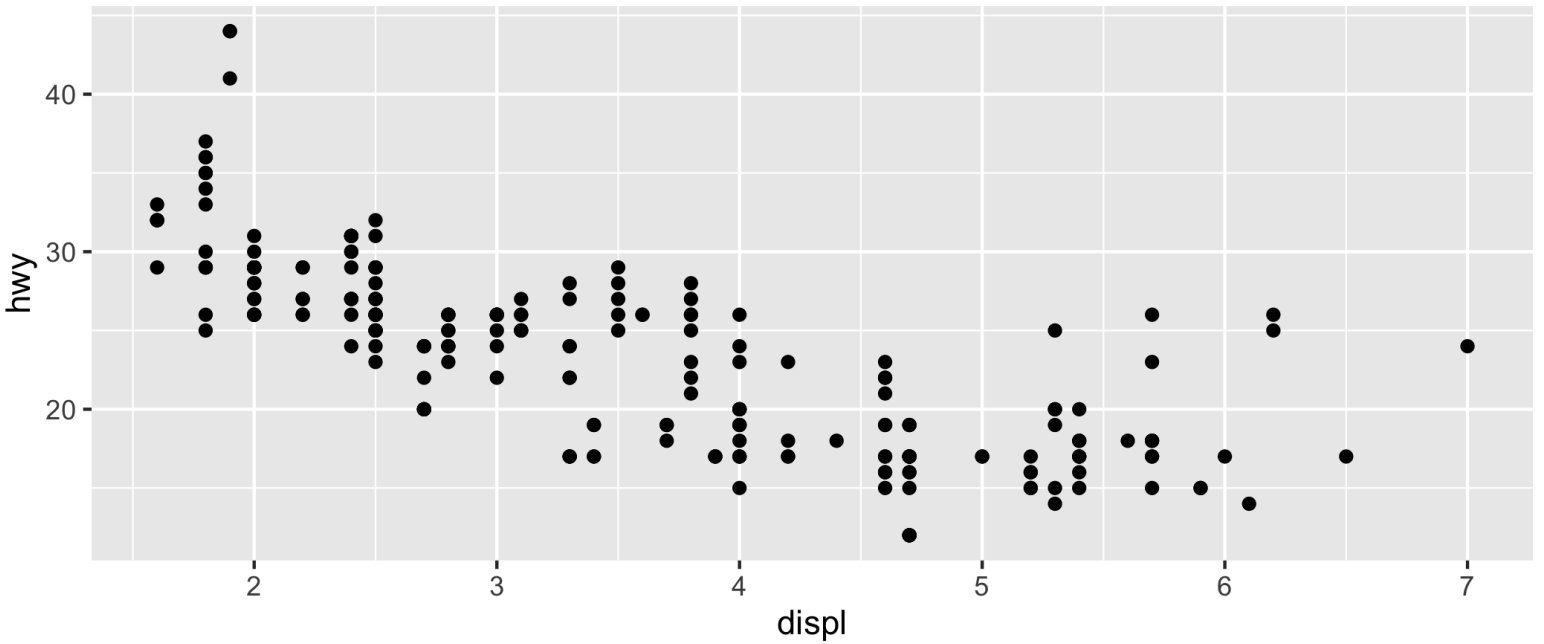
```
# load an R default dataset into your session
mpg <- mpg

# check out the structure of the dataset
str(mpg)
```

```
## tibble [234 × 11] (S3: tbl_df/tbl/data.frame)
##  $ manufacturer: chr [1:234] "audi" "audi" "audi" "audi" ...
##  $ model       : chr [1:234] "a4" "a4" "a4" "a4" ...
##  $ displ       : num [1:234] 1.8 1.8 2 2 2.8 2.8 3.1 1.8 1.8 2 ...
##  $ year        : int [1:234] 1999 1999 2008 2008 1999 1999 2008 1999 1999 2
##  $ cyl         : int [1:234] 4 4 4 4 6 6 6 4 4 4 ...
##  $ trans       : chr [1:234] "auto(l5)" "manual(m5)" "manual(m6)" "auto(av)
##  $ drv         : chr [1:234] "f" "f" "f" "f" ...
##  $ cty         : int [1:234] 18 21 20 21 16 18 18 18 16 20 ...
##  $ hwy         : int [1:234] 29 29 31 30 26 26 27 26 25 28 ...
##  $ fl          : chr [1:234] "p" "p" "p" "p" ...
##  $ class       : chr [1:234] "compact" "compact" "compact" "compact" ...
```

ggplot2 syntax

Create a ggplot

```
# create a plot between engine size (displ) and fuel efficiency (hwy)
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy))
```



Exercise 1

```
# Run ggplot(data = mpg). What do you see?
ggplot(data = mpg)

# How many rows are in mpg? How many columns?

# What does the drv variable describe? Read the help for ?mpg to find ou
?mpg

# Make a scatterplot of hwy vs cyl.

# What happens if you make a scatterplot of class vs drv? Why is the plo
```

Exercise 1 (with answers)

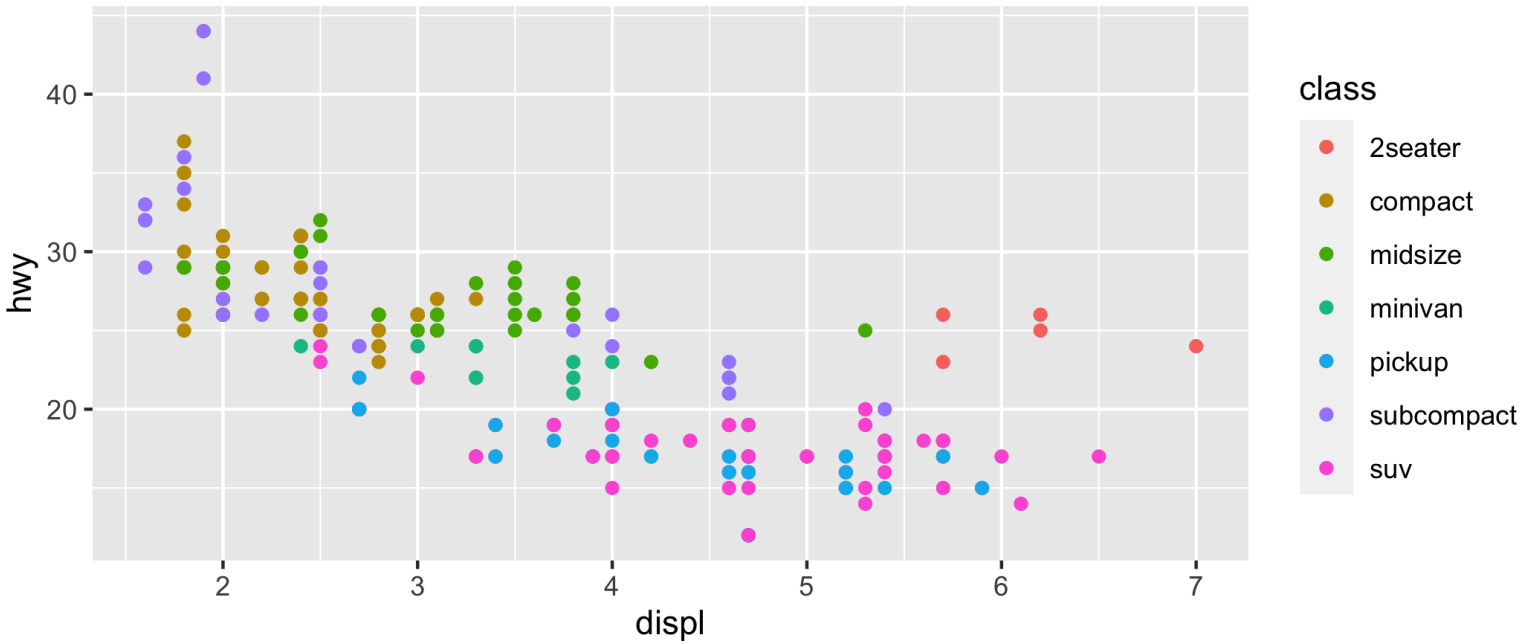
```
# Run ggplot(data = mpg). What do you see?
ggplot(data = mpg)

# How many rows are in mpg? How many columns?
dim(mpg)
nrow(mpg)
ncol(mpg)

# What does the drv variable describe? Read the help for ?mpg to find ou
?mpg

# Make a scatterplot of hwy vs cyl.
ggplot(data = mpg, aes(x = hwy,y = cyl)) + geom_point()

# What happens if you make a scatterplot of class vs drv? Why is the plo
ggplot(data = mpg, aes(x = class, y = drv)) + geom_point()
```

Aesthetic Mapping

- Change the color of an element by group: `color`
- Change the size of an element by group: `size`
- Change the opacity of an element by group: `alpha`
- Change the shape of an element by group: `shape`

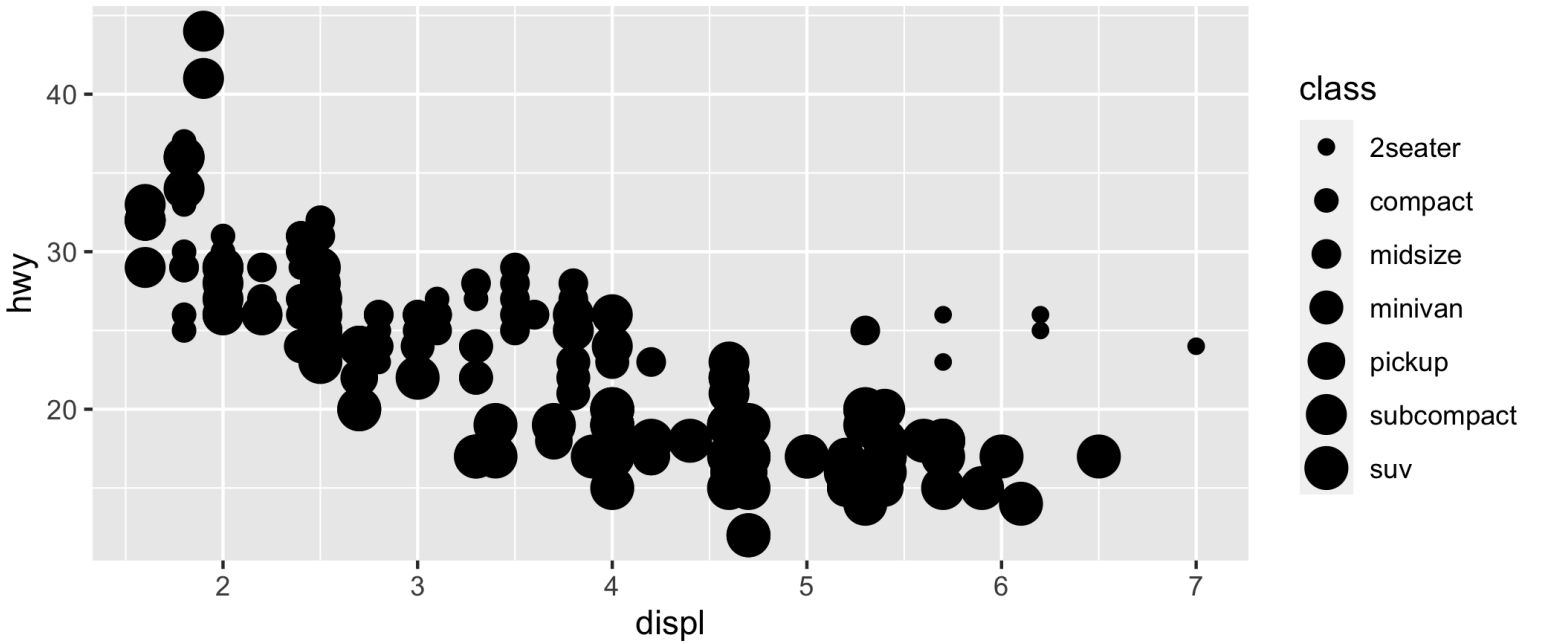
Aesthetic Mapping (color)

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



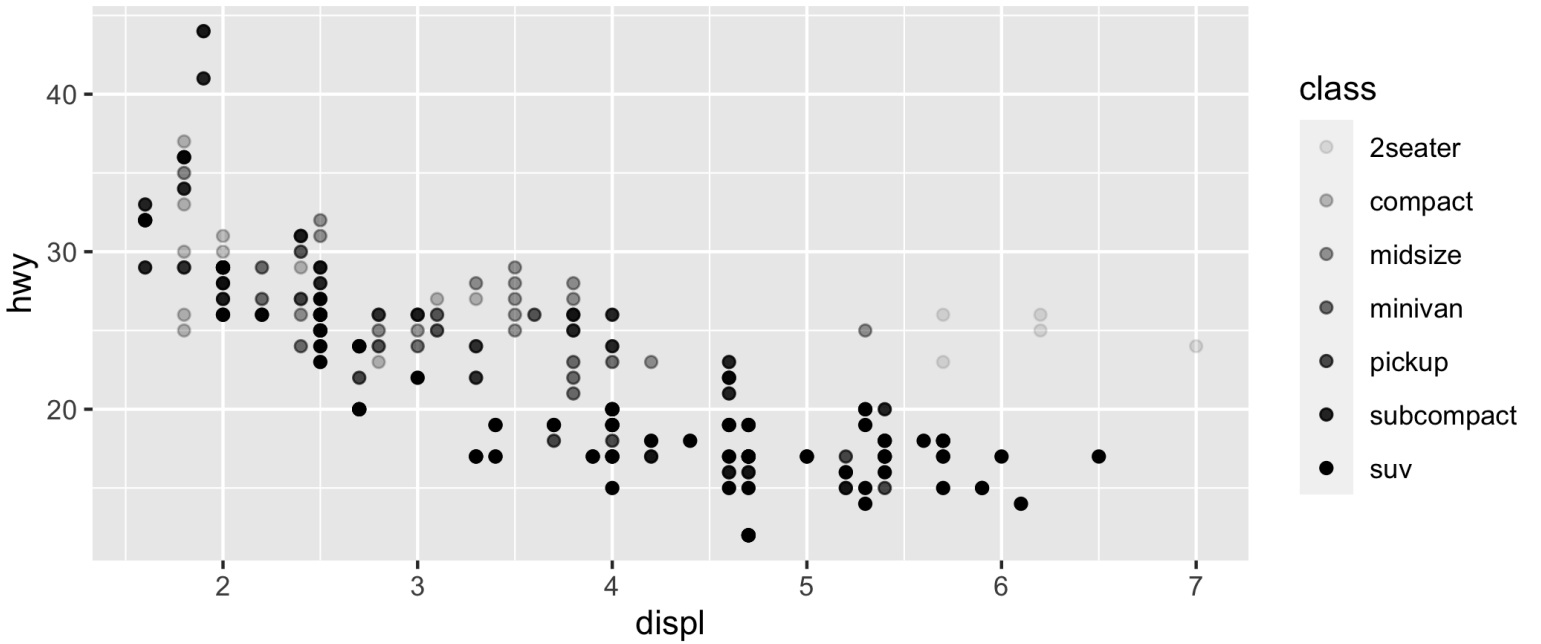
Aesthetic Mapping (size)

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



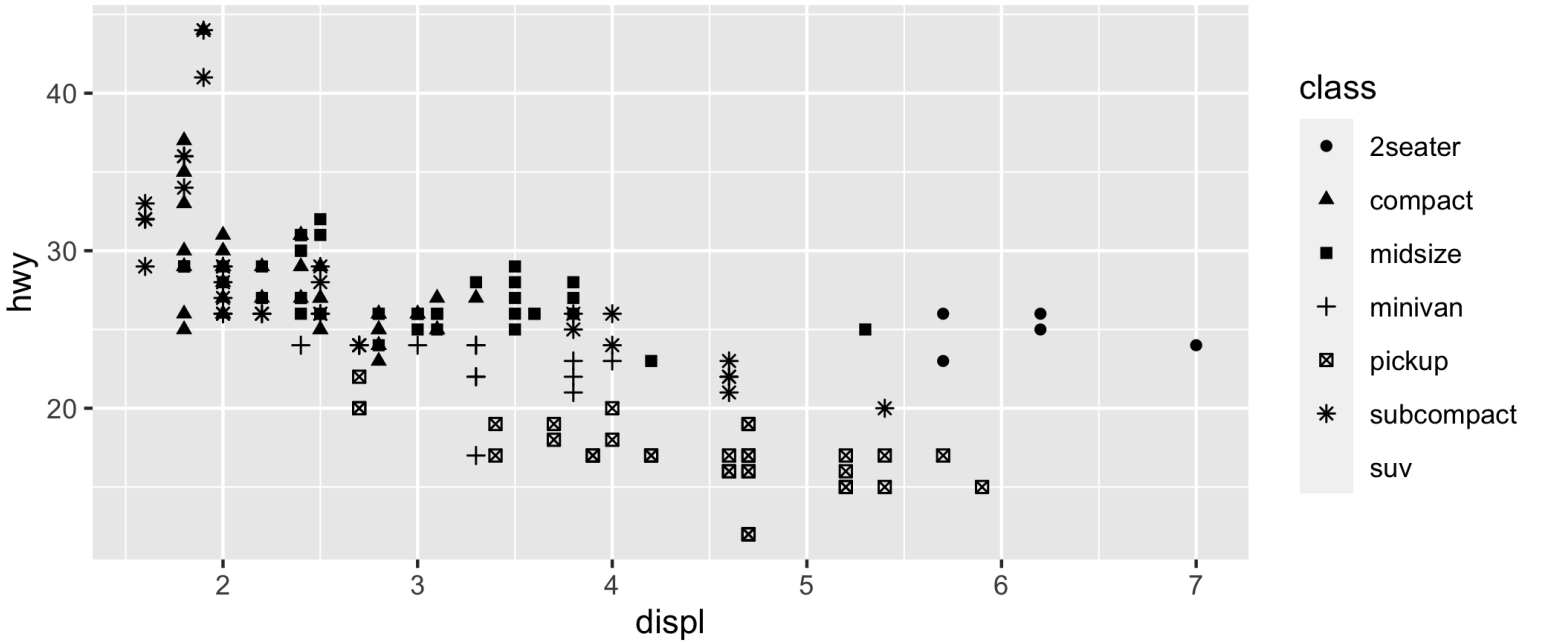
Aesthetic Mapping (alpha)

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



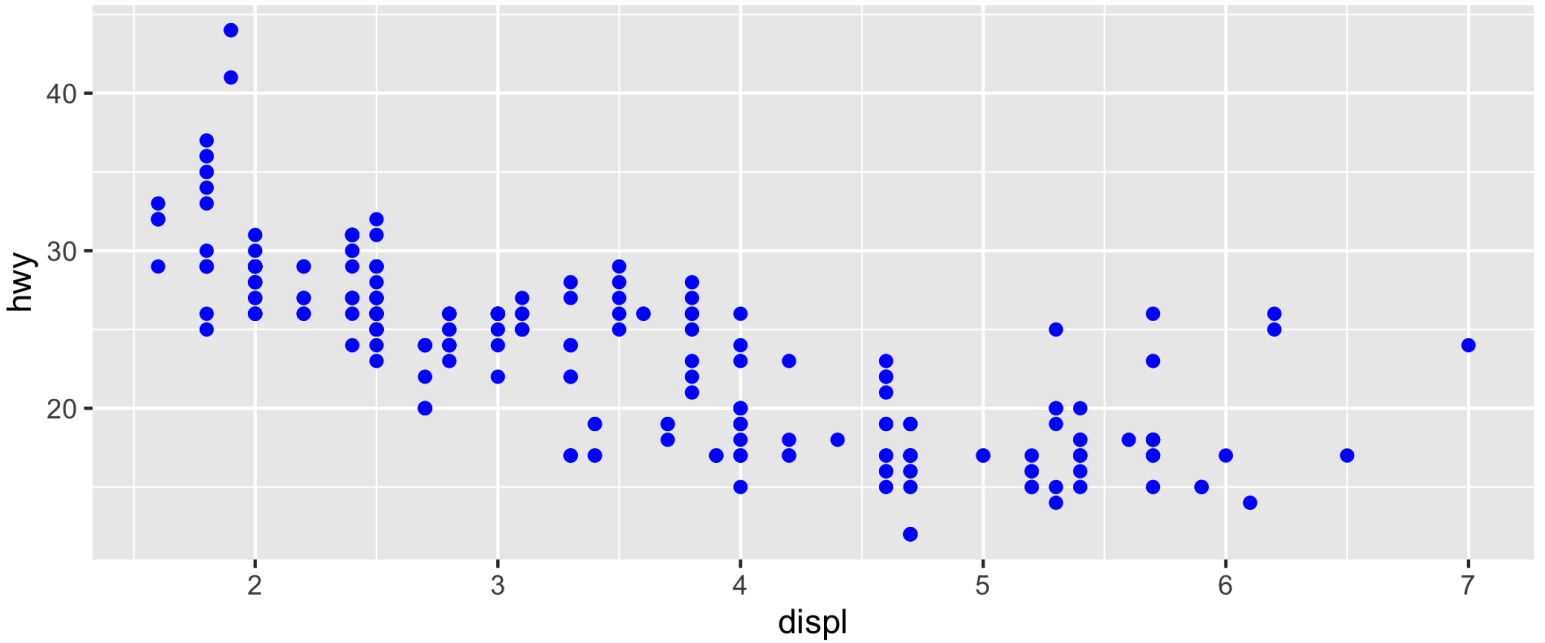
Aesthetic Mapping (shape)

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



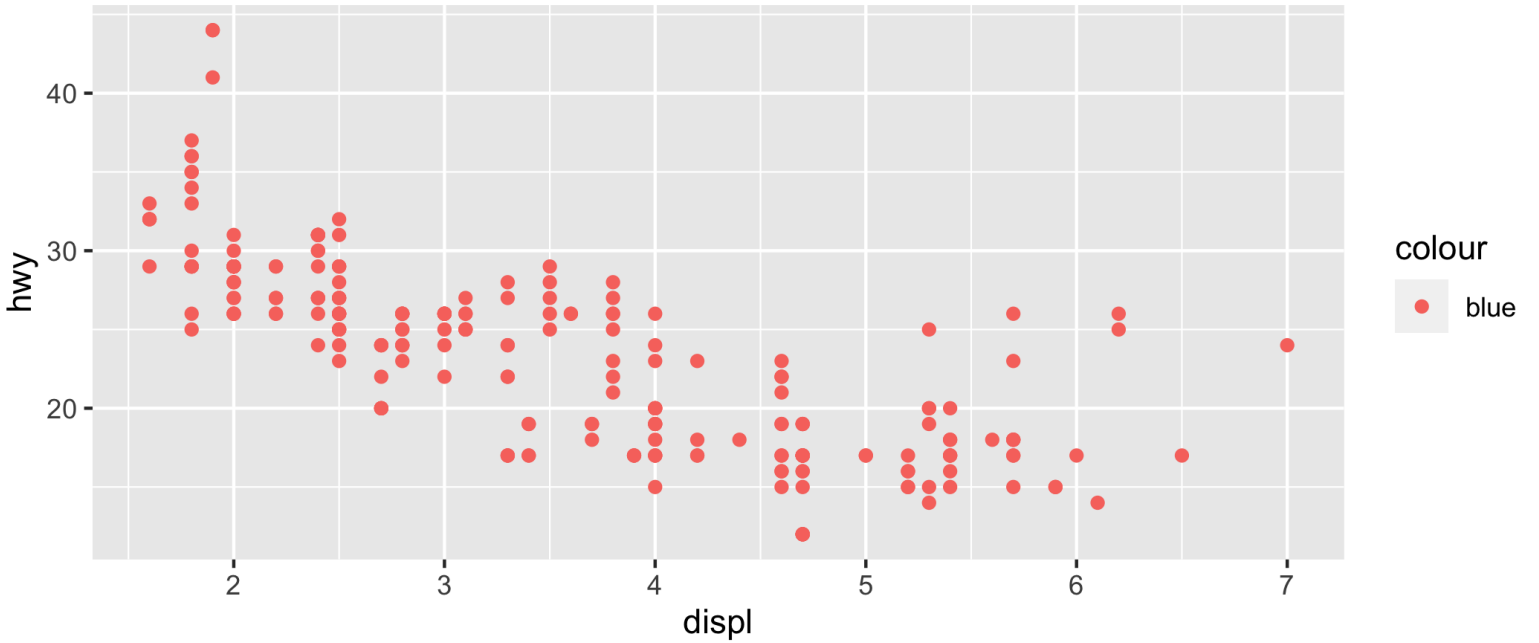
What is wrong with this Aesthetic Mapping?

```
# set aesthetics manually
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy), color = "blue")
```



What are the points red instead of blue?

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



Exercise 2

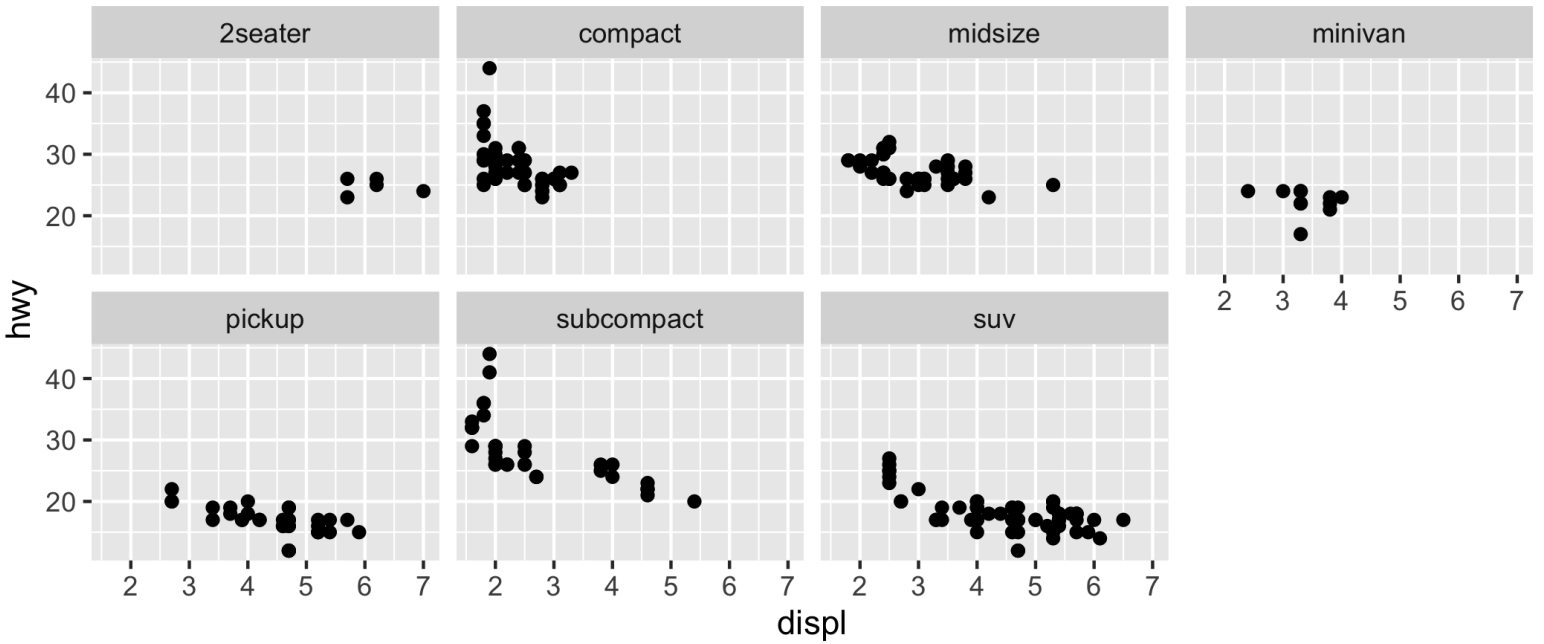
```
# What's gone wrong with this code? Why are the points not green?

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

# Which variables in mpg are categorical?
# Which variables are continuous? (Hint: type ?mpg to read the documentation for the dataset).
# How can you see this information when you run mpg?
sapply(mpg, class)

# Map a continuous variable to color, size, and shape.
# How do these aesthetics behave differently for categorical vs. continuous variables?

# What happens if you map the same variable to multiple aesthetics?

# What does the stroke aesthetic do? What shapes does it work with? (Hint: use ?geom_point)

# What happens if you map an aesthetic to something other than a variable name,
# like aes(colour = displ < 5)?
```

Facets

- Plot for each level of a variable: `facet_wrap()`
- Plot for each level of a combination of two variables: `facet_wrap()`
- Combine two or more plots by row/column: `gridExtra::grid.arrange()`

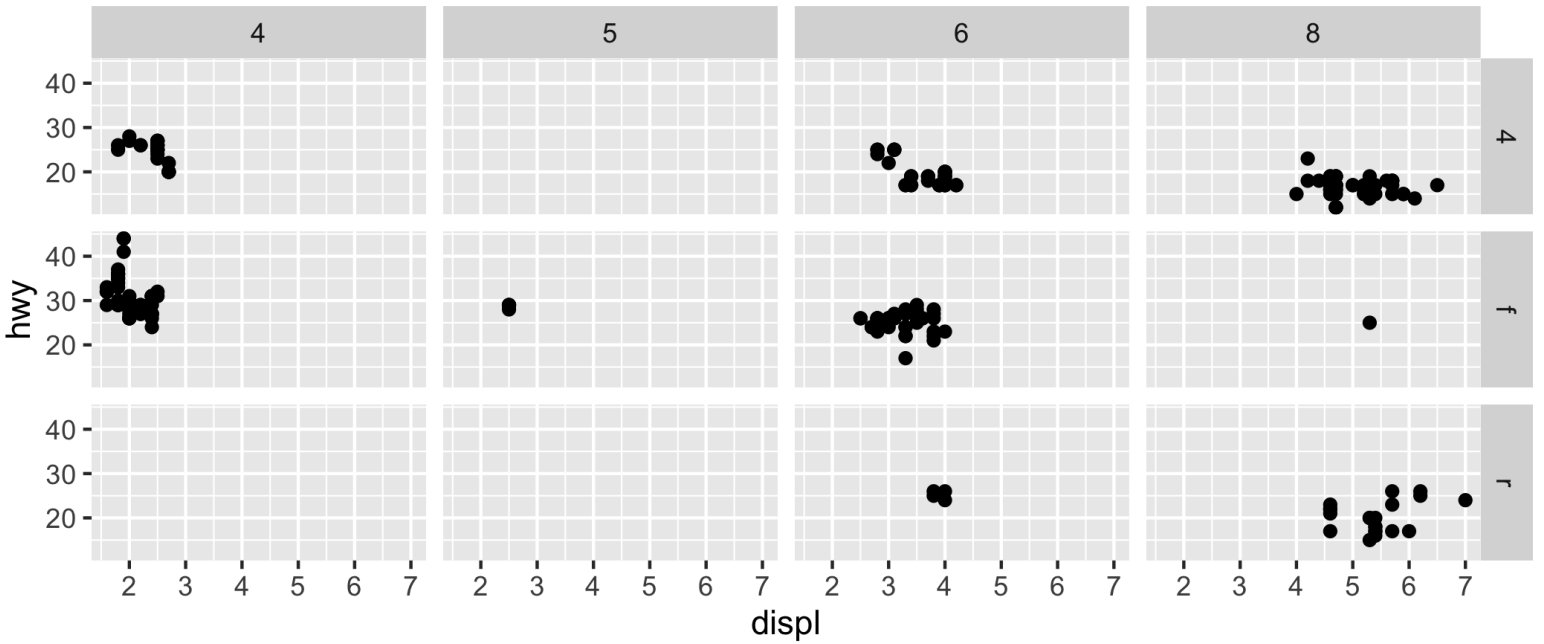
Facets (one group)

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



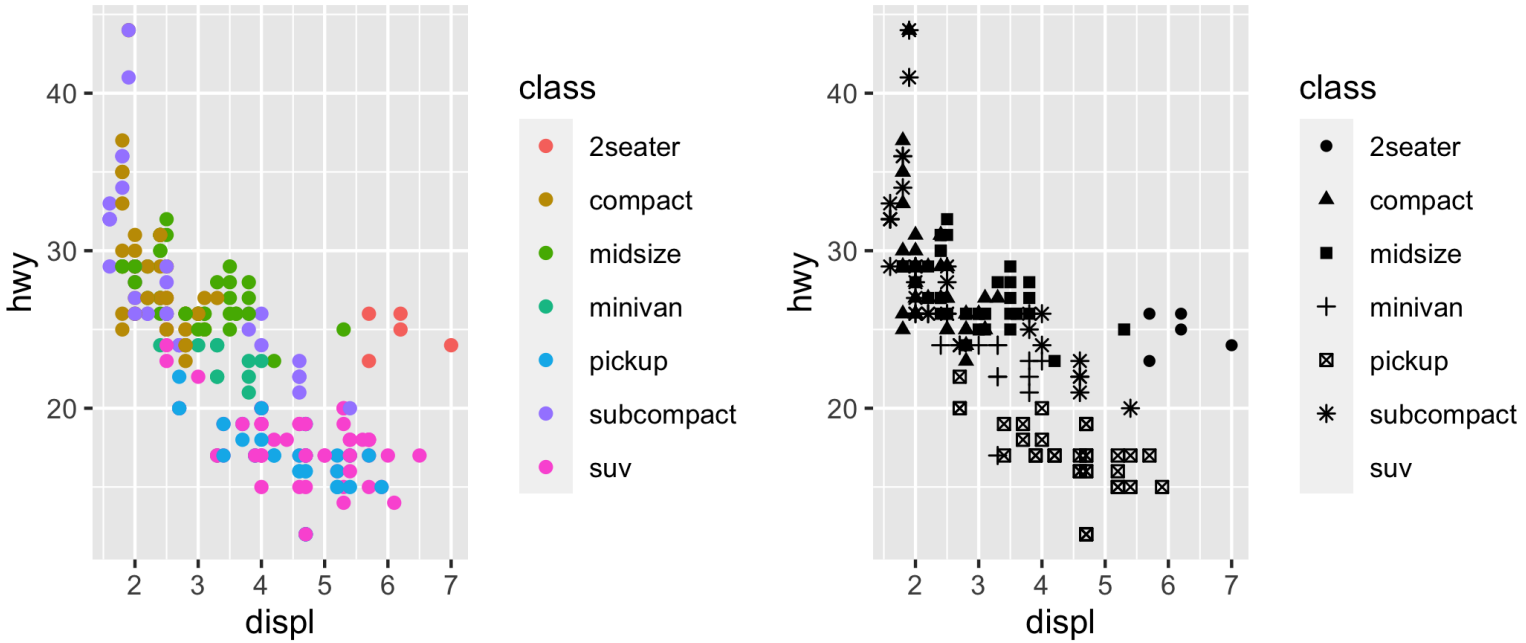
Facets (two groups)

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



Combining multiple plots

```
p1 <- ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, color = class))

p2 <- ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, shape = class))

library(gridExtra)
grid.arrange(p1, p2, nrow = 1)
```



Exercise 3

```
# What happens if you facet on a continuous variable?

ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  facet_wrap(~ cty, nrow = 2)

# What do the empty cells in plot with facet_grid(drv ~ cyl) mean? How do they relate to this plot?

ggplot(data = mpg) +
  geom_point(mapping = aes(x = drv, y = cyl))

# What plots does the following code make? What does . do?

ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  facet_grid(drv ~ .)

ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  facet_grid(. ~ cyl)

# Take the first faceted plot in this section:

ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  facet_wrap(~ class, nrow = 2)


# What are the advantages to using faceting instead of the colour aesthetic? What are the disadvantages?
# How might the balance change if you had a larger dataset?

# Read ?facet_wrap. What does nrow do? What does ncol do?
# What other options control the layout of the individual panels?
# Why doesn't facet_grid() have nrow and ncol arguments?

# When using facet_grid() you should usually put the variable with more unique levels in the columns. Why?
```

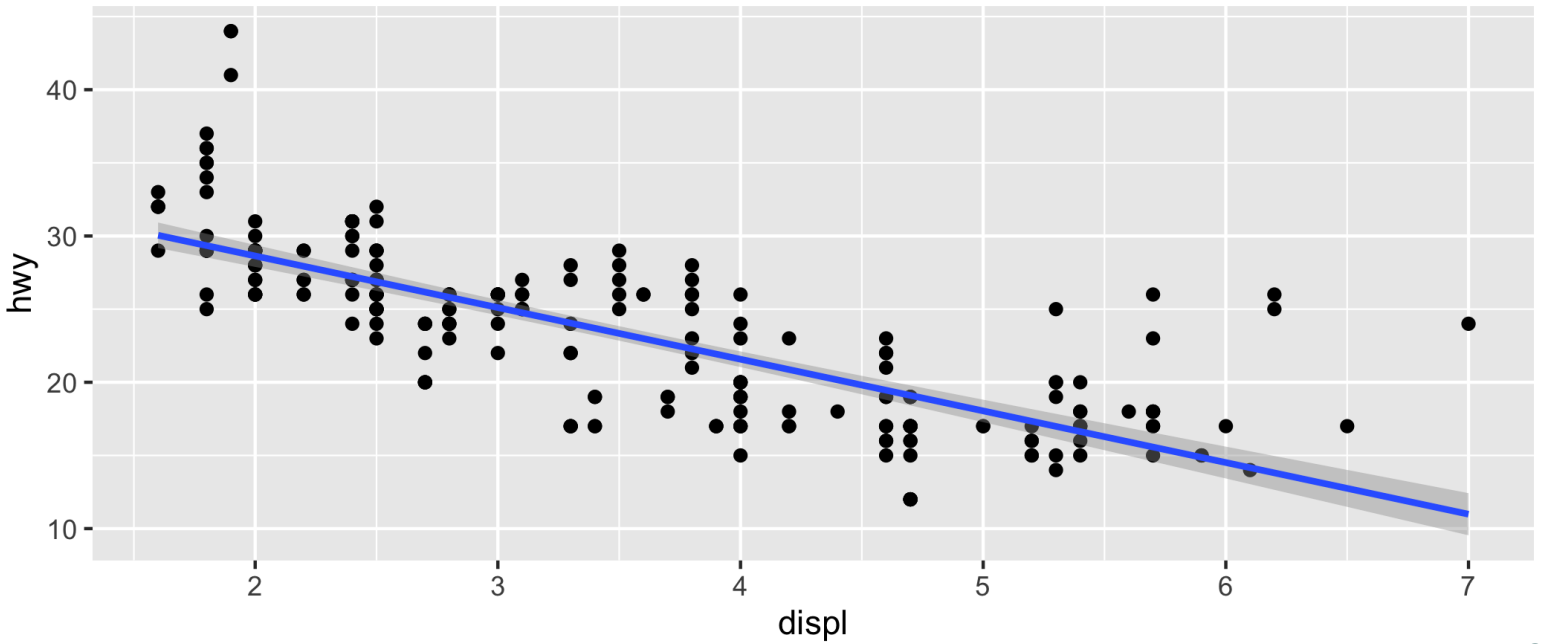
Geometric Objects

- scatterplots: `geom_point()`
- linegraphs: `geom_line()`
- histograms: `geom_histogram()`
- boxplots: `geom_boxplot()`
- barplots: `geom_bar()` & `geom_col()`

scatterplots: geom_point()

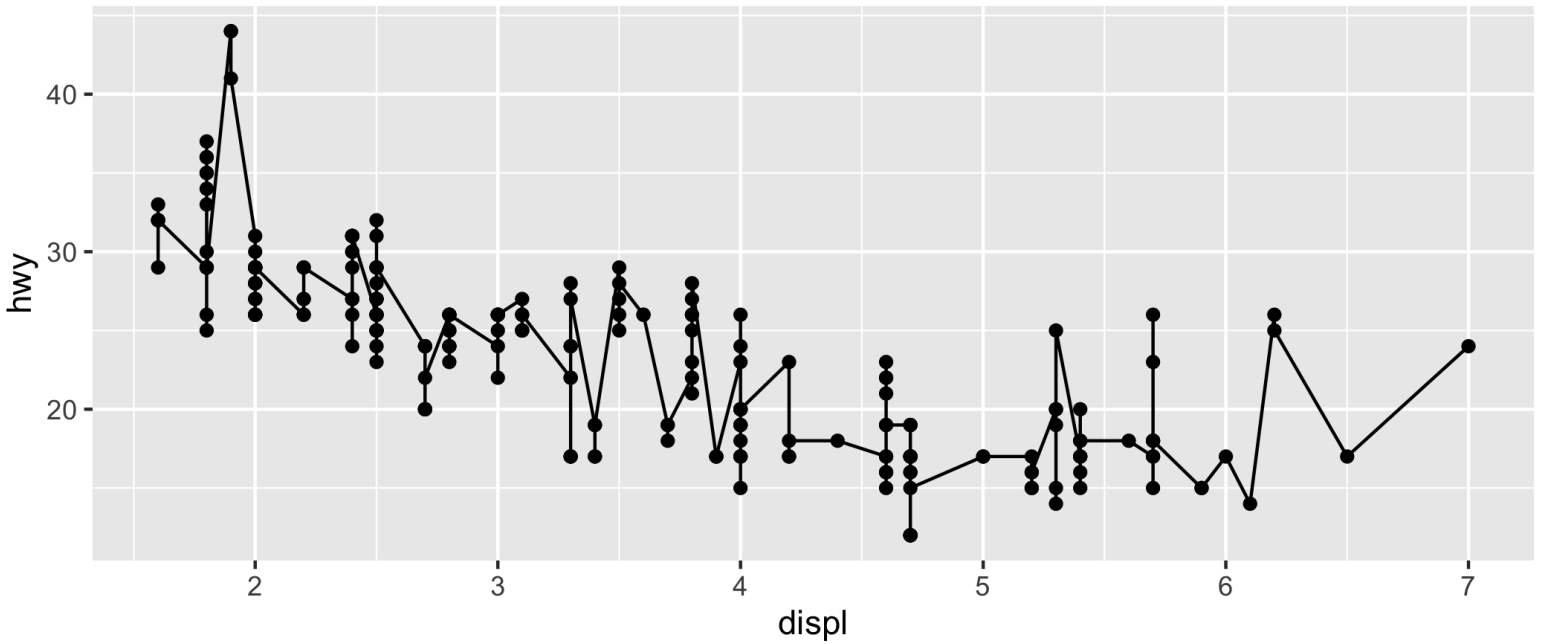
```
# apply global (instead of local) mapping to a scatterplot with a best f
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_point() +
  geom_smooth(method = "lm")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



linegraphs: geom_line()

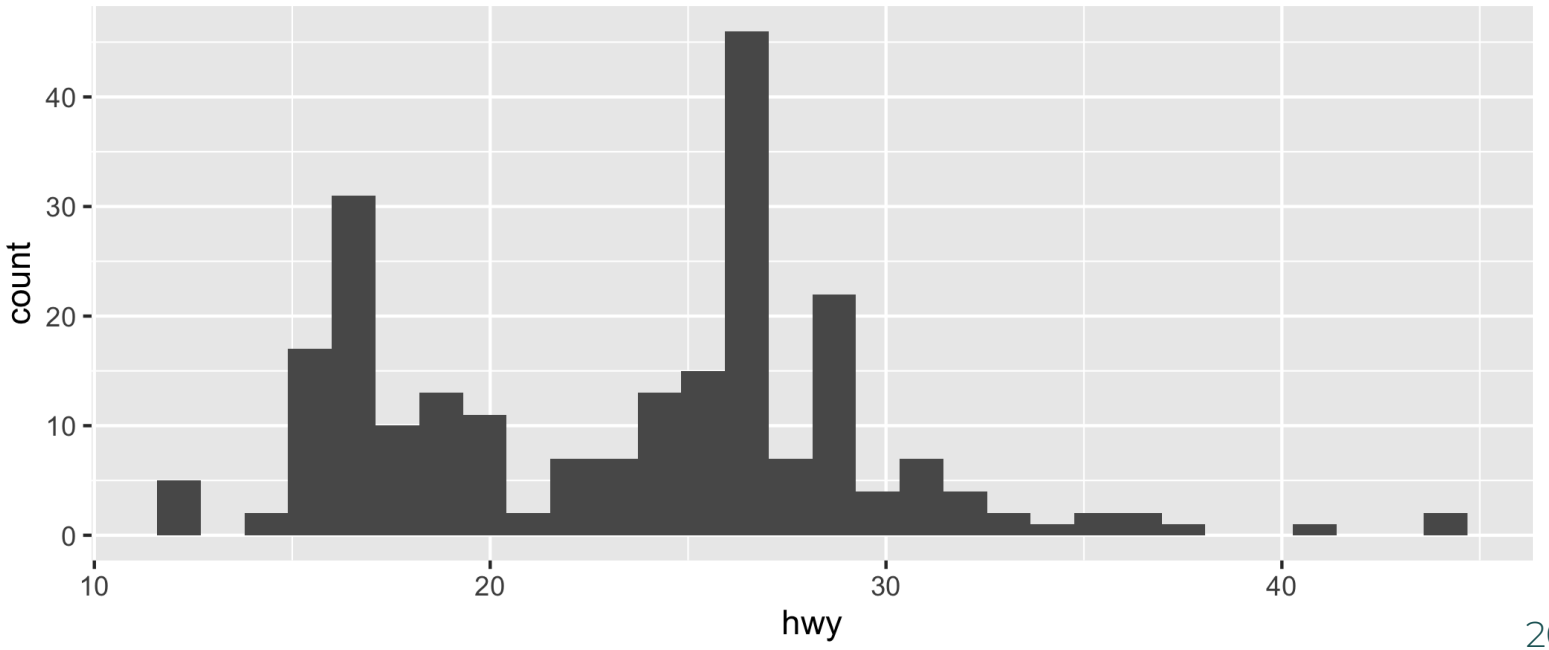
```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_point() +
  geom_line()
```



histograms: geom_histogram()

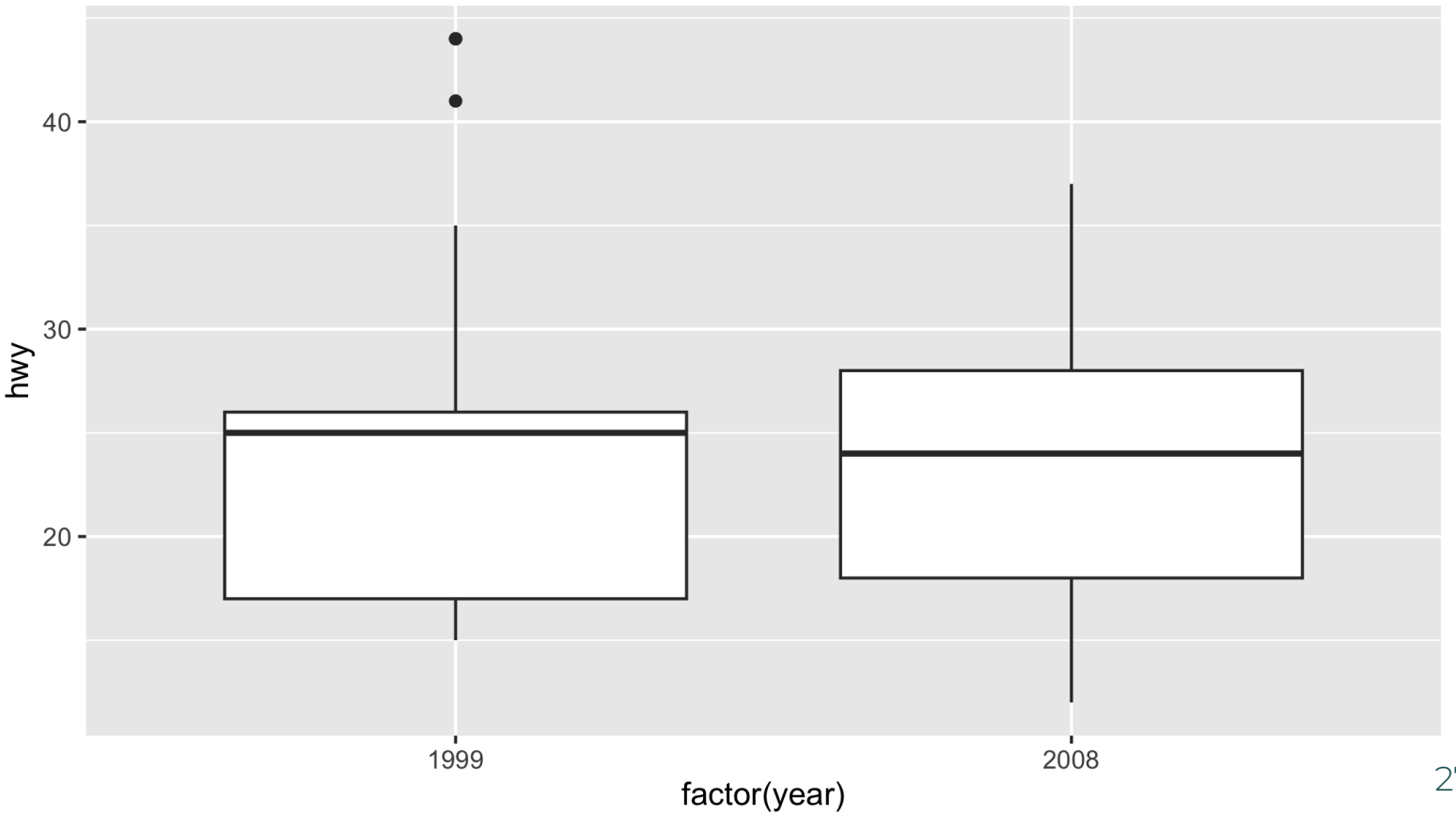
```
ggplot(data = mpg, mapping = aes(x = hwy)) +
  geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



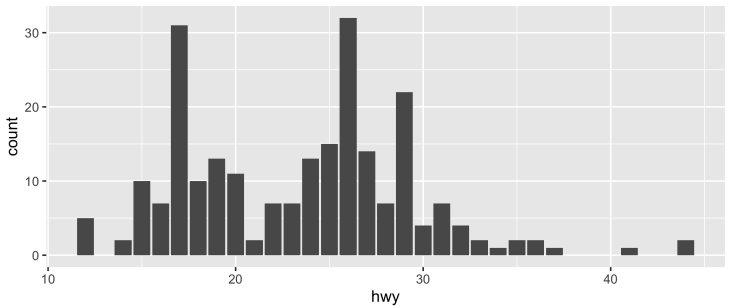
boxplots: geom_boxplot()

```
ggplot(data = mpg, mapping = aes(x = factor(year), y = hwy)) +
  geom_boxplot()
```

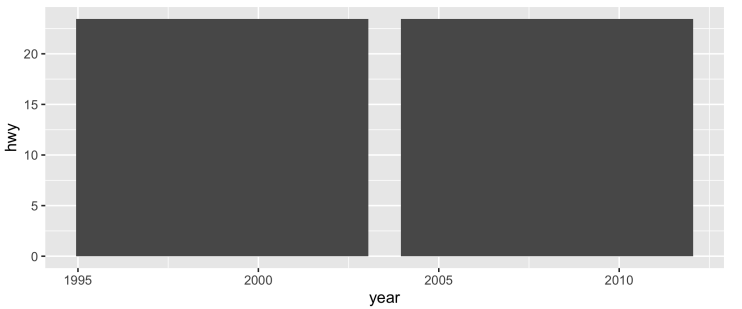


barplots: geom_bar() & geom_col()

```
# bar plot based on one variable
ggplot(data = mpg,
       mapping = aes(x = hwy)) +
  geom_bar()
```

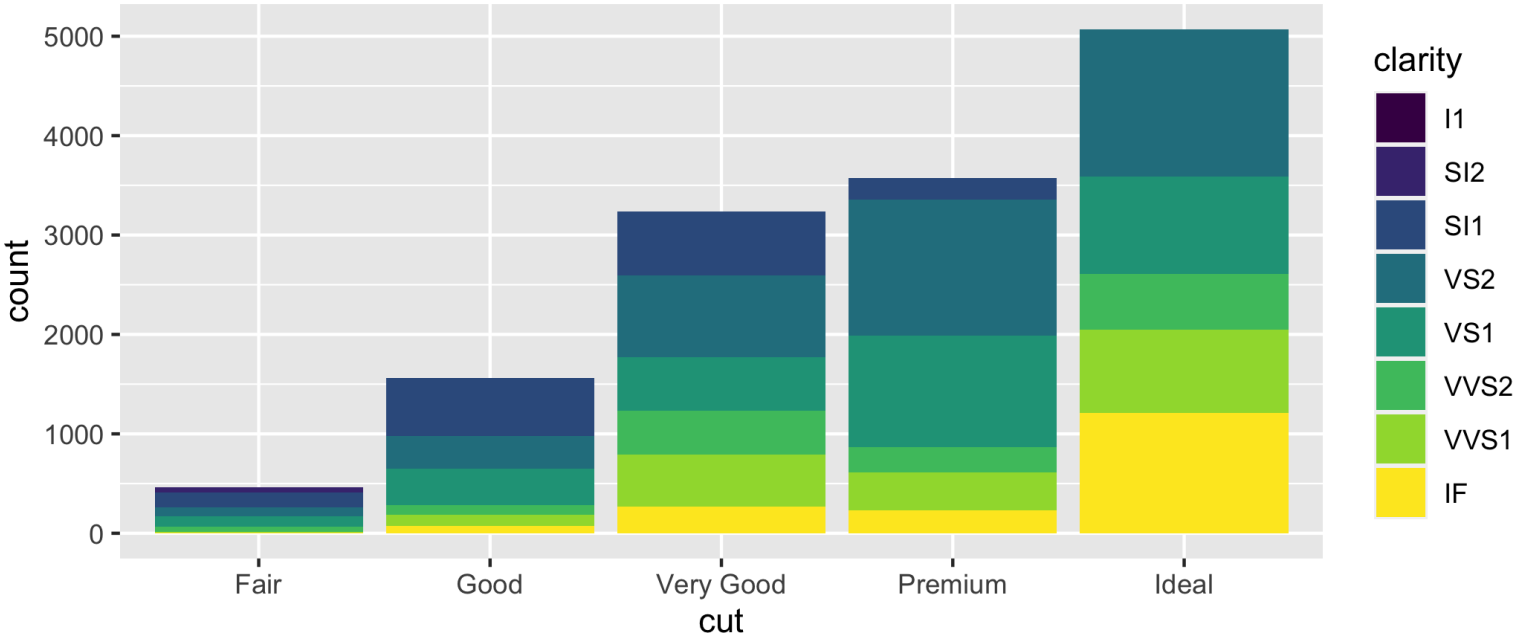



```
# bar plot based on two variables
df <- mpg %>%
  group_by(year) %>%
  summarize(hwy = mean(hwy,
                       na.rm = T)

ggplot(data = df,
       mapping = aes(x = year,
                     y = hwy)) +
  geom_col()
```



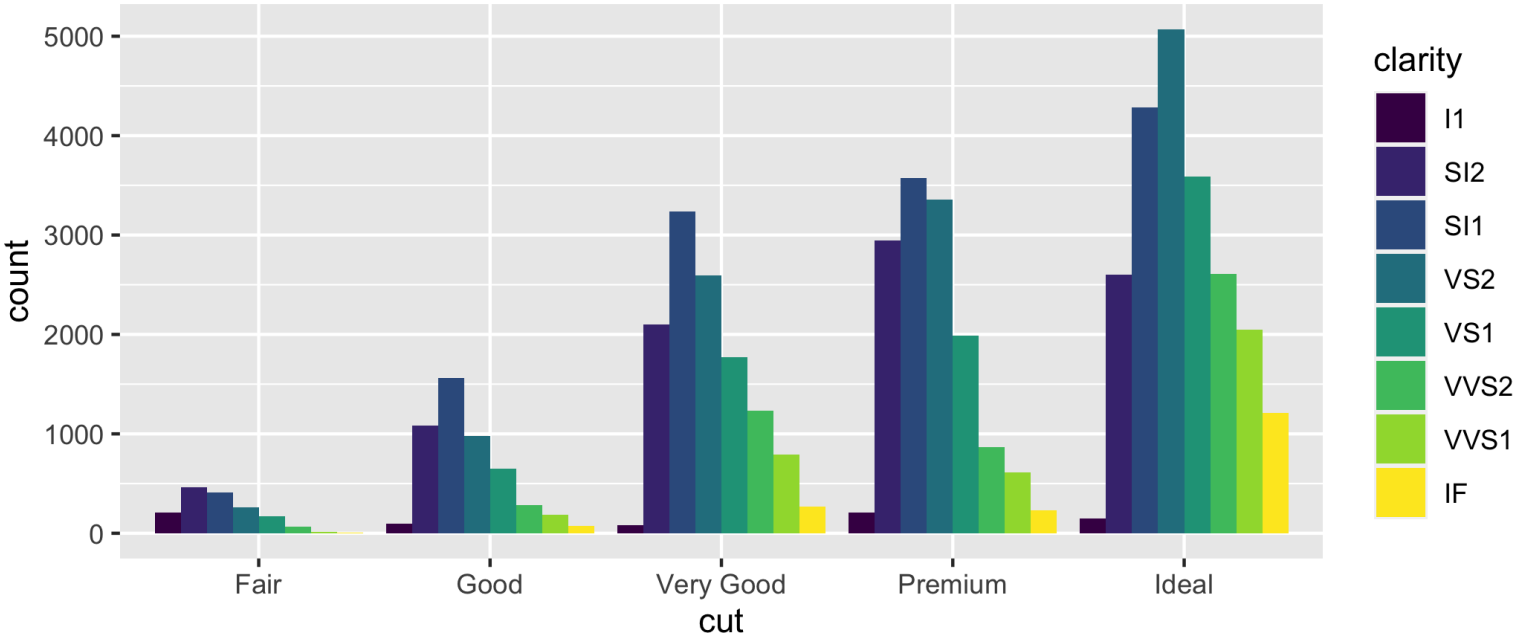
barplots by group

```
# a new default dataset
diamonds <- diamonds

# stacked bars
ggplot(data = diamonds, mapping = aes(x = cut, fill = clarity)) +
  geom_bar(position = "identity")
```



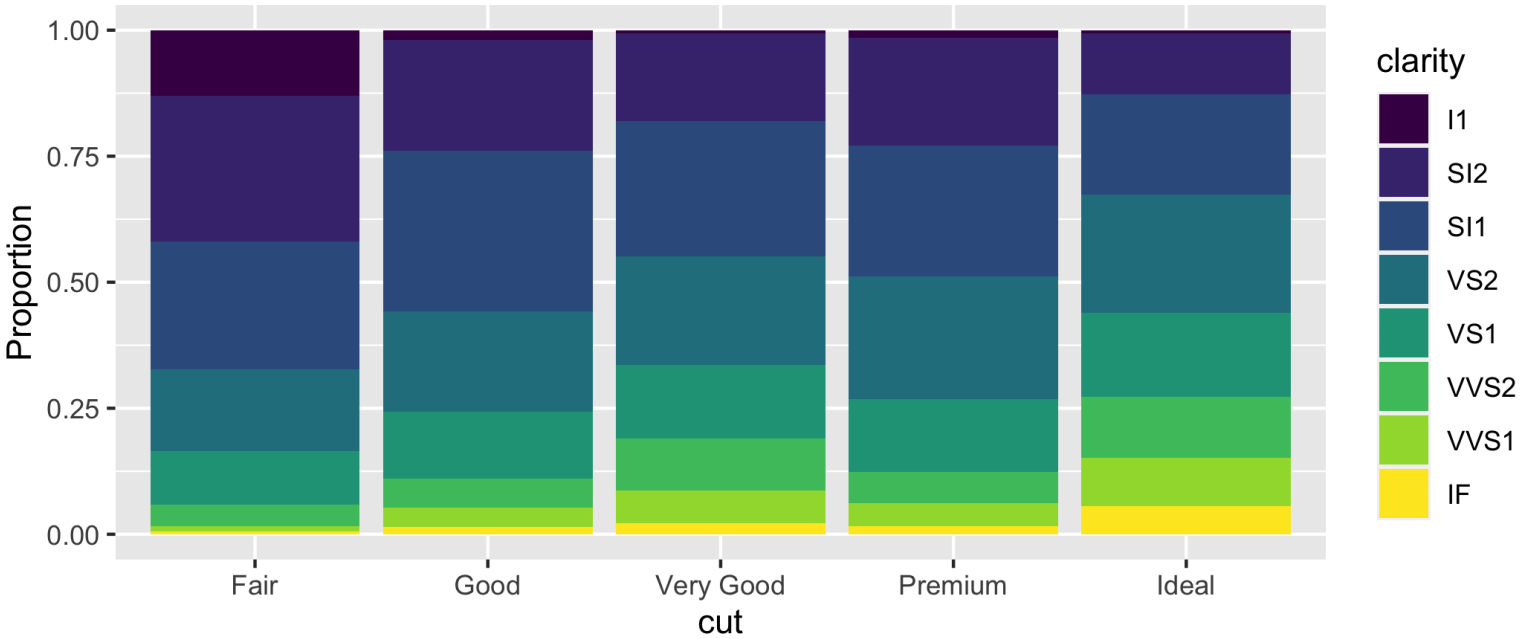
barplots by group

```
# bars side by side
ggplot(data = diamonds, mapping = aes(x = cut, fill = clarity)) +
  geom_bar(position = "dodge")
```



barplots by group

```
# bars as proportions
ggplot(data = diamonds, mapping = aes(x = cut, fill = clarity)) +
  geom_bar(position = "fill") +
  labs(y = "Proportion")
```

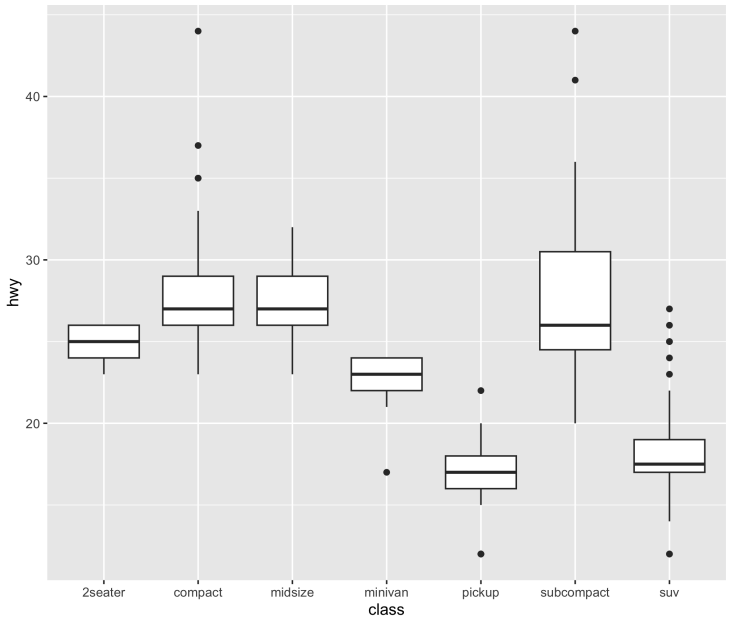


Exercise 4

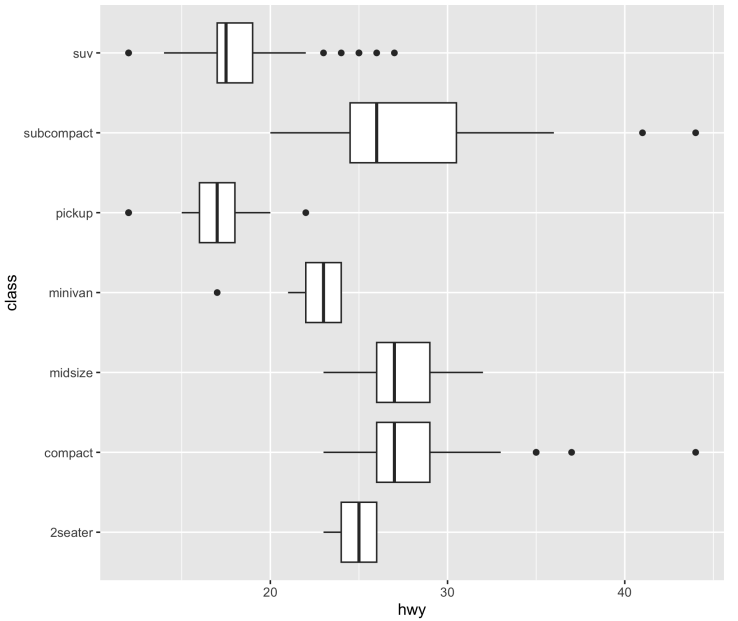
```
# What geom would you use to draw a line chart? A boxplot? A histogram?

# Run this code in your head and predict what the output will look like.

ggplot(data = mpg, mapping = aes(x = displ, y = hwy, color = drv)) +
  geom_point() +
  geom_smooth(method = "lm",
              se = F)
# What does show.legend = FALSE do? What happens if you remove it?
# Why do you think I used it earlier in the chapter?

# What does the se argument to geom_smooth() do?

# Will these two graphs look different? Why/why not?

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

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

Coordinate System

```
ggplot(data = mpg,
       mapping = aes(x = class,
                     y = hwy)) +
  geom_boxplot()
```




```
ggplot(data = mpg,
       mapping = aes(x = class,
                     y = hwy)) +
  geom_boxplot() +
  coord_flip()
```

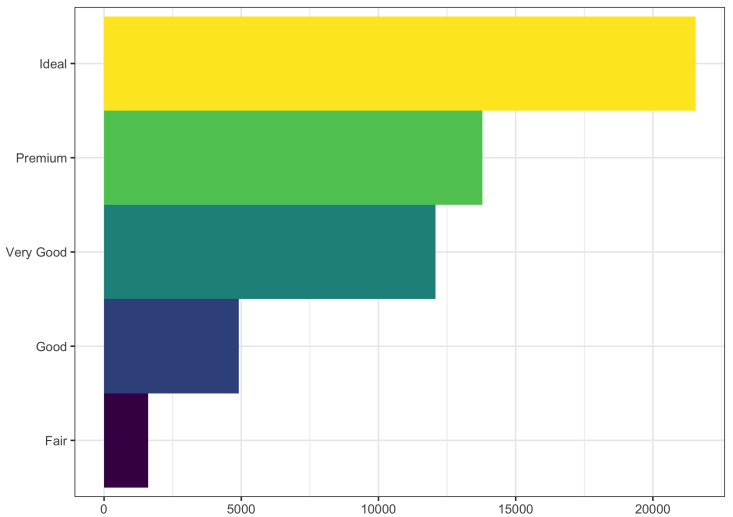


Coordinate System

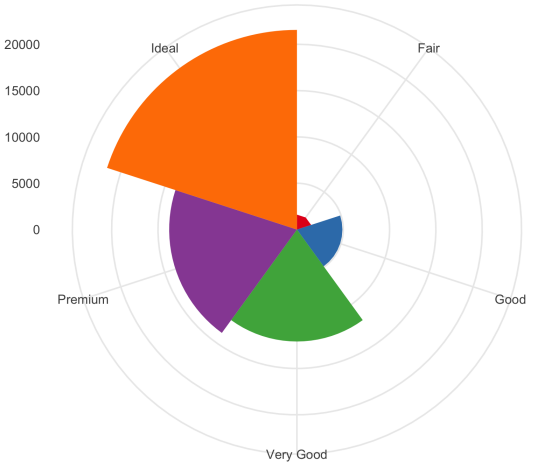
- `coord_flip()`
- `coord_polar()`
- `labs(x = ..., y = ...)`
- `theme()`

Coordinate System

```
ggplot(data = diamonds) +
  geom_bar(
    mapping = aes(x = cut, fill = cut),
    show.legend = FALSE,
    width = 1
  ) +
  theme(aspect.ratio = 1) +
  theme_bw() +
  labs(x = NULL, y = NULL) +
  coord_flip()
```

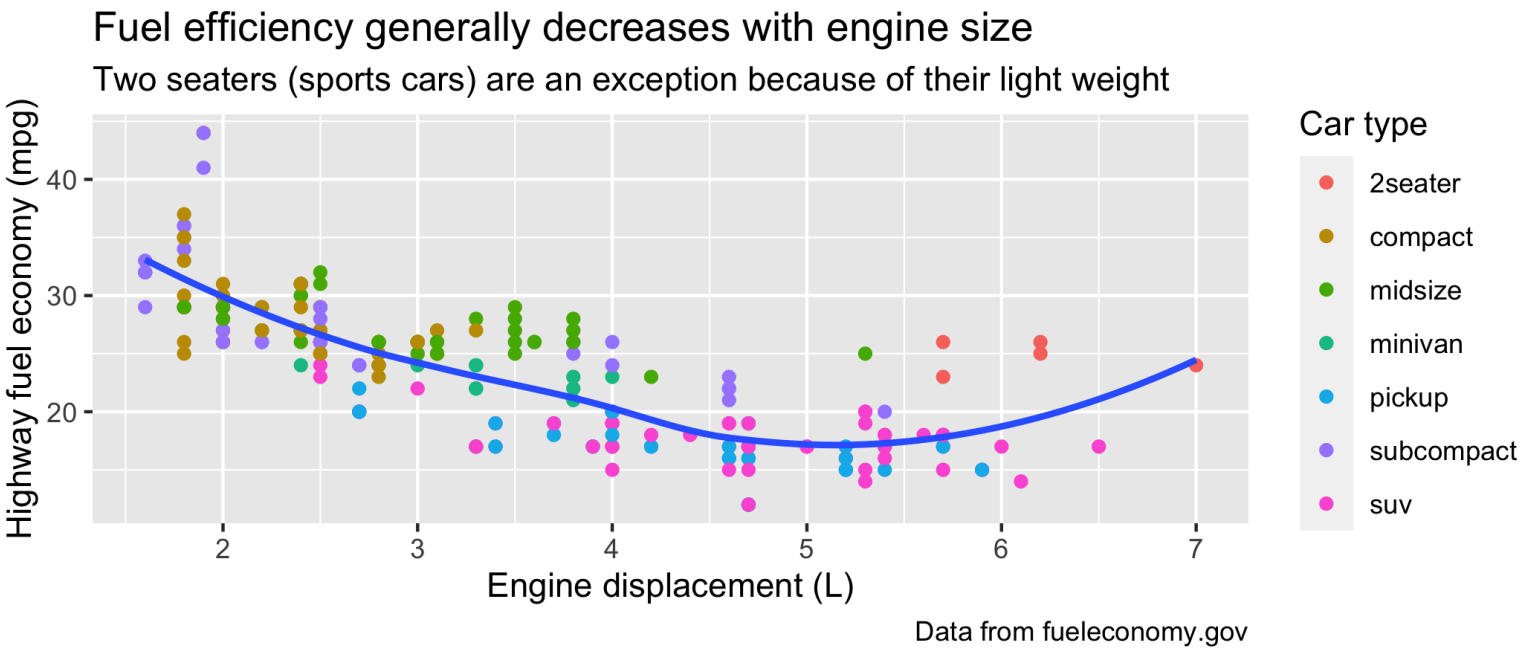



```
ggplot(data = diamonds) +
  geom_bar(
    mapping = aes(x = cut, fill = cut),
    show.legend = FALSE,
    width = 1
  ) +
  theme(aspect.ratio = 1) +
  theme_minimal() +
  labs(x = NULL, y = NULL) +
  coord_polar() +
  scale_fill_brewer(palette = "Set1")
```



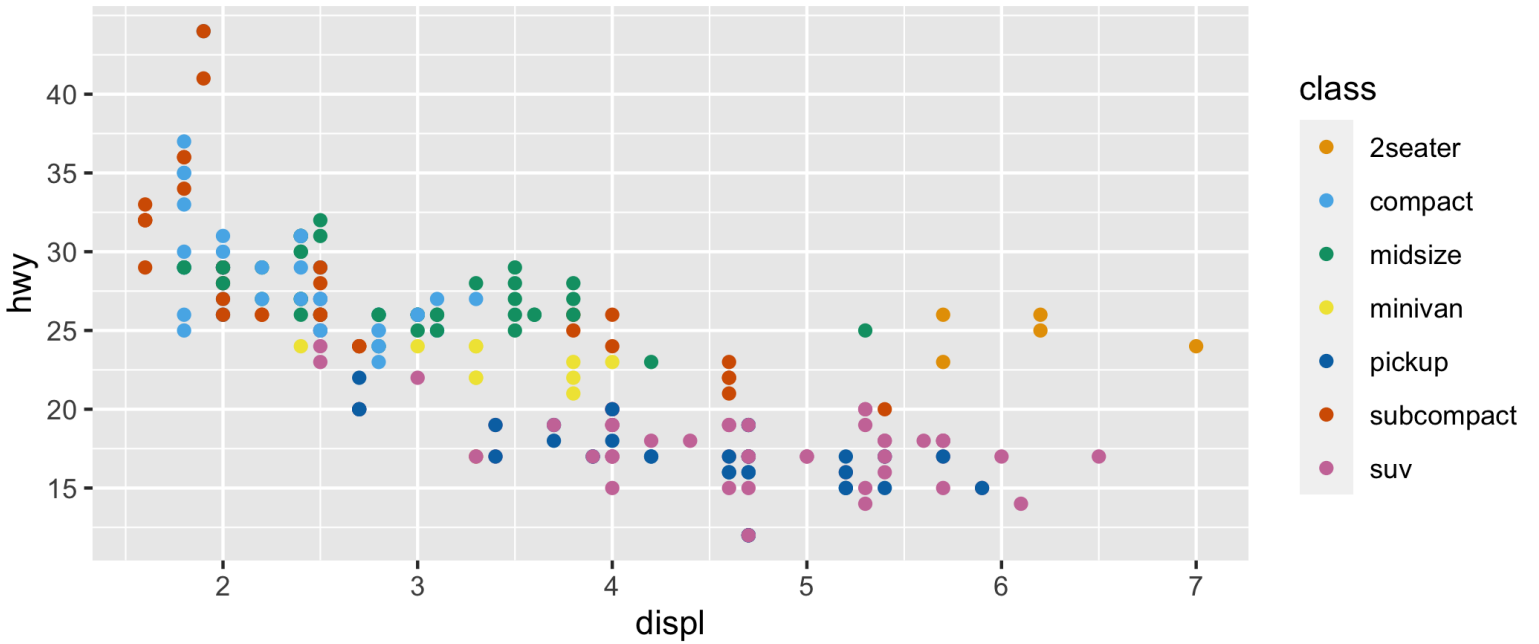
Labels and Annotations

```
ggplot(mpg, aes(displ, hwy)) +
  geom_point(aes(color = class)) +
  geom_smooth(se = FALSE) +
  labs(
    title = "Fuel efficiency generally decreases with engine size",
    subtitle = "Two seaters (sports cars) are an exception because of their light weight",
    caption = "Data from fueleconomy.gov",
    x = "Engine displacement (L)",
    y = "Highway fuel economy (mpg)",
    colour = "Car type")
```



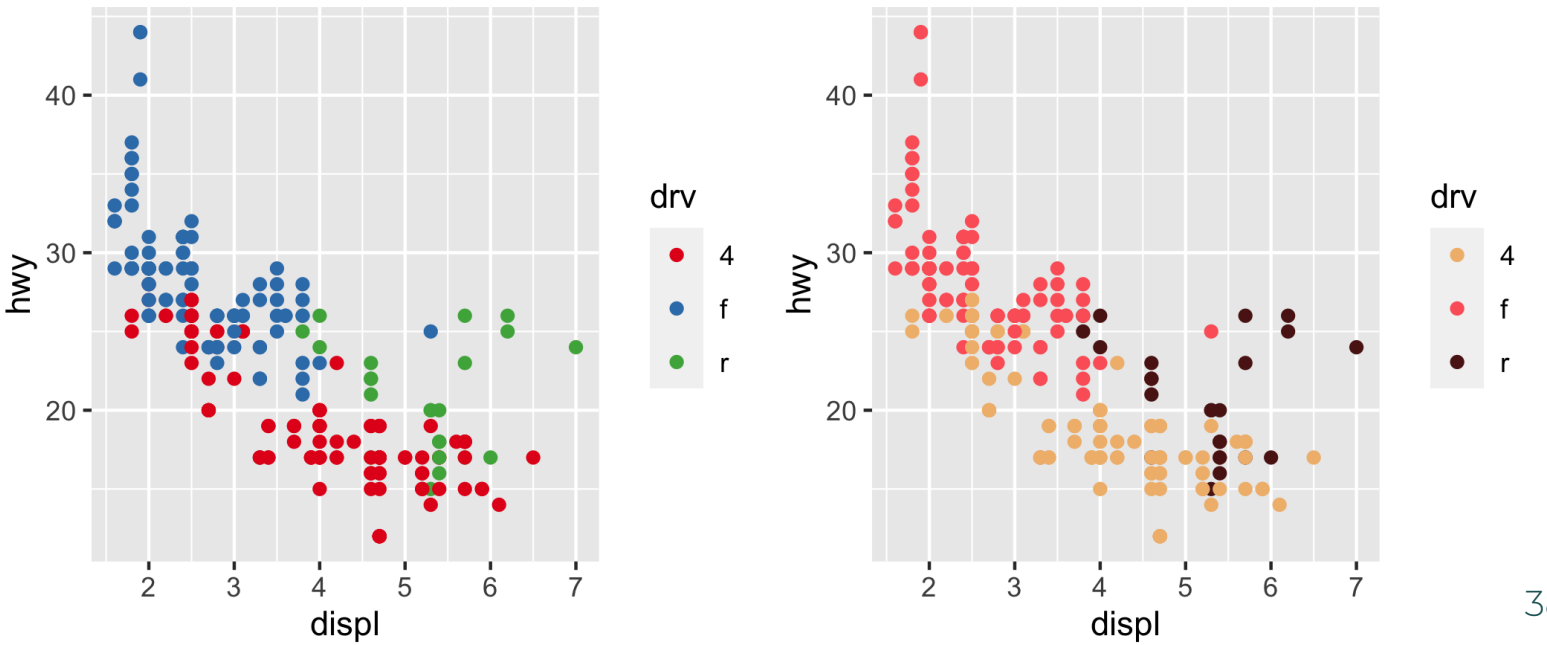
Scales

```
ggplot(mpg, aes(displ, hwy)) +
  geom_point(aes(color = class)) +
  scale_x_continuous(breaks = seq(1, 10, by = 1)) +
  scale_y_continuous(breaks = seq(15, 40, by = 5)) +
  scale_color_manual(values = c("#E69F00", "#56B4E9", "#009E73",
                                "#F0E442", "#0072B2", "#D55E00", "#CC79A7"))
```



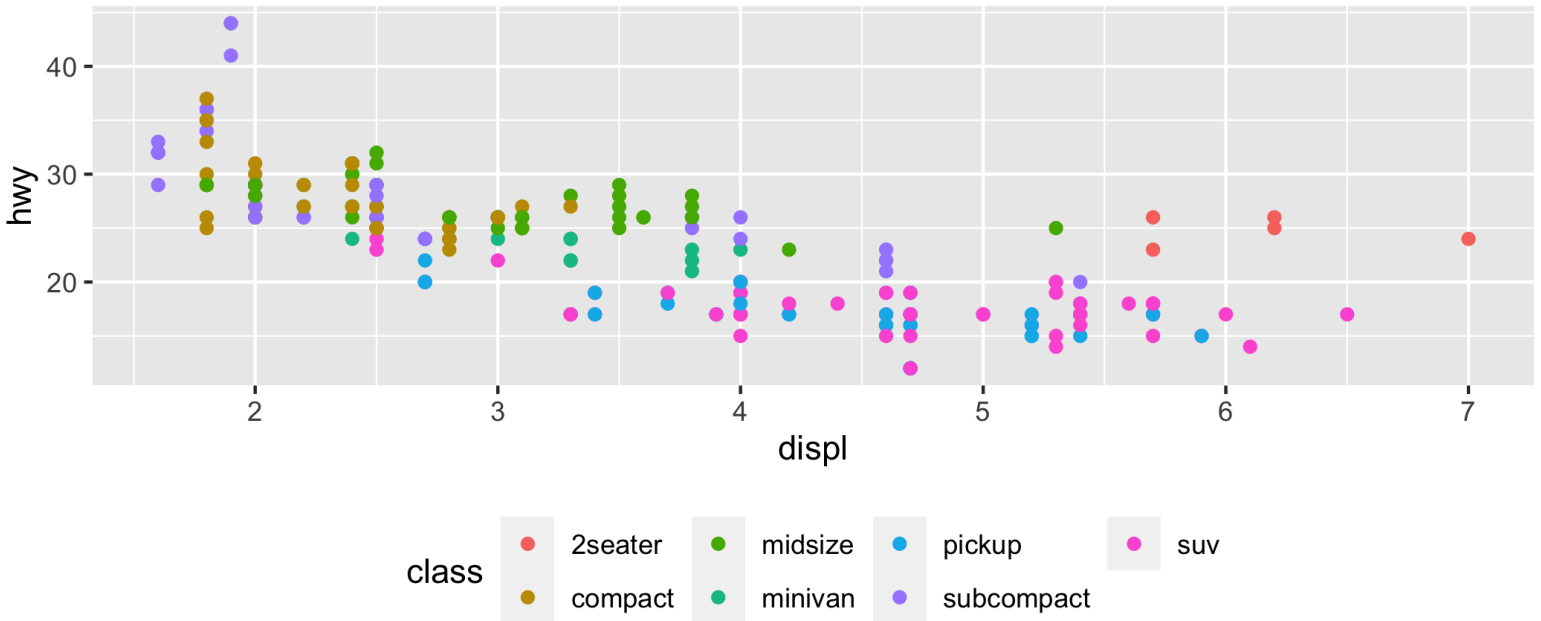
Colors

```
p <- ggplot(mpg, aes(displ, hwy)) +
  geom_point(aes(color = drv))
p1 <- p + scale_color_brewer(palette = "Set1")

library(wesanderson)
p2 <- p + scale_color_manual(values = wes_palette("GrandBudapest1"))

gridExtra::grid.arrange(p1, p2, nrow = 1)
```



Legends

```
p <- ggplot(mpg, aes(displ, hwy)) +
  geom_point(aes(colour = class))

# p + theme(legend.position = "right") # the default
# p + theme(legend.position = "left")
# p + theme(legend.position = "top")
p + theme(legend.position = "bottom")
```



Saving the plot

```
p <- ggplot(mpg, aes(displ, hwy)) +
  geom_point(aes(color = class)) +
  geom_smooth(se = FALSE, method = "lm") +
  theme_bw()

# save the ggplot under the working directory
ggsave(p, "my-plot.pdf",
       width = 6, height = 8,
       dpi = 600)
```

ggplot2 to plotly

Install the `plotly` package

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

library(plotly)
```

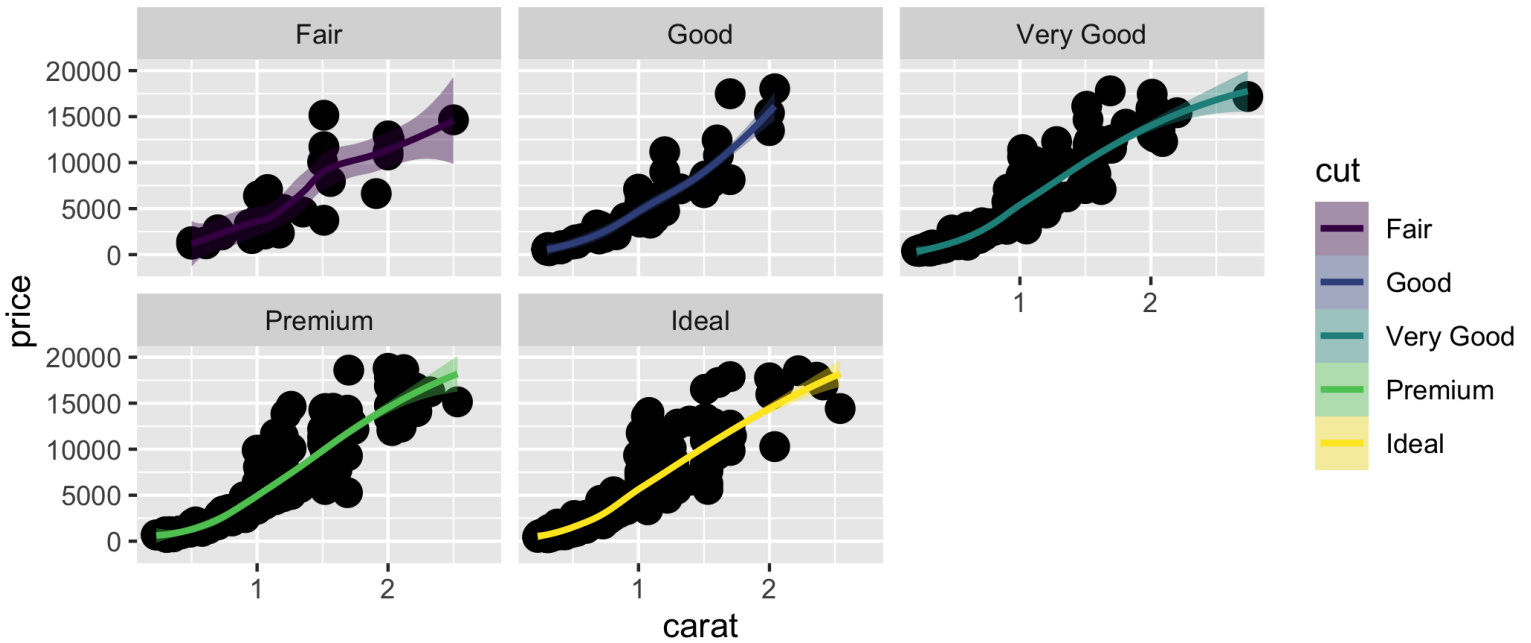
```
##
## Attaching package: 'plotly'

## The following object is masked from 'package:ggplot2':
##
##     last_plot

## The following object is masked from 'package:stats':
##
##     filter

## The following object is masked from 'package:graphics':
##
##     layout
```

Watch out for the masked functions!

Create a ggplot first

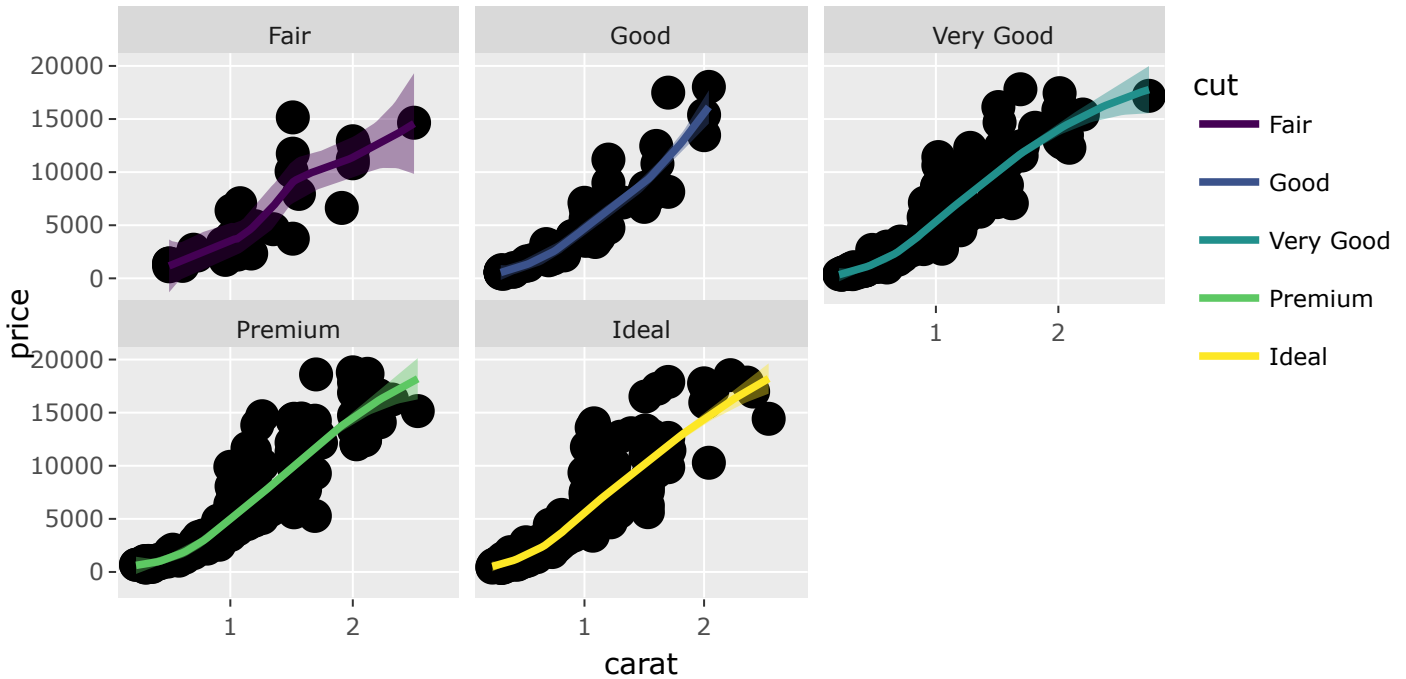
```
# Sample 1000 rows (the random seed makes the sampling reproducible)
set.seed(100)
d <- diamonds %>% sample_n(1000)

p <- ggplot(data = d, aes(x = carat, y = price)) +
  geom_point(aes(text = paste("Clarity:", clarity)), size = 4) +
  geom_smooth(aes(colour = cut, fill = cut)) +
  facet_wrap(~ cut)
p
```

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



Create a ggplotly plot

```
ggplotly(p)
```



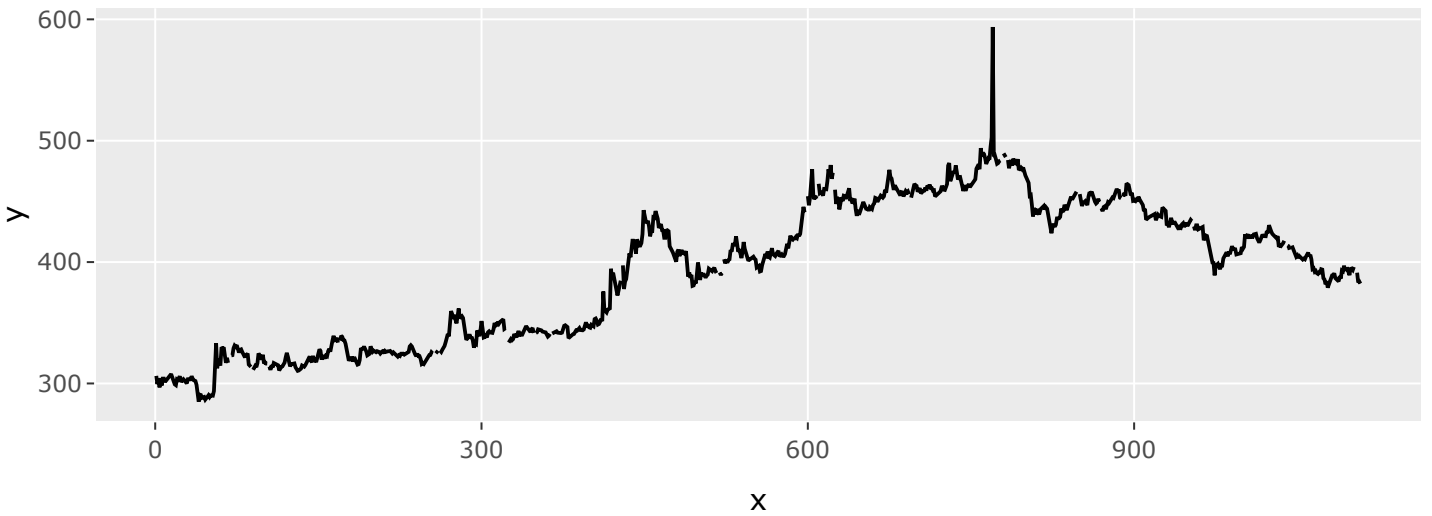
`plotly` plot is a list

```
names(p)
```

```
## [1] "data"        "layers"      "scales"      "mapping"
## [5] "theme"       "coordinates" "facet"       "plot_env"
## [9] "labels"
```

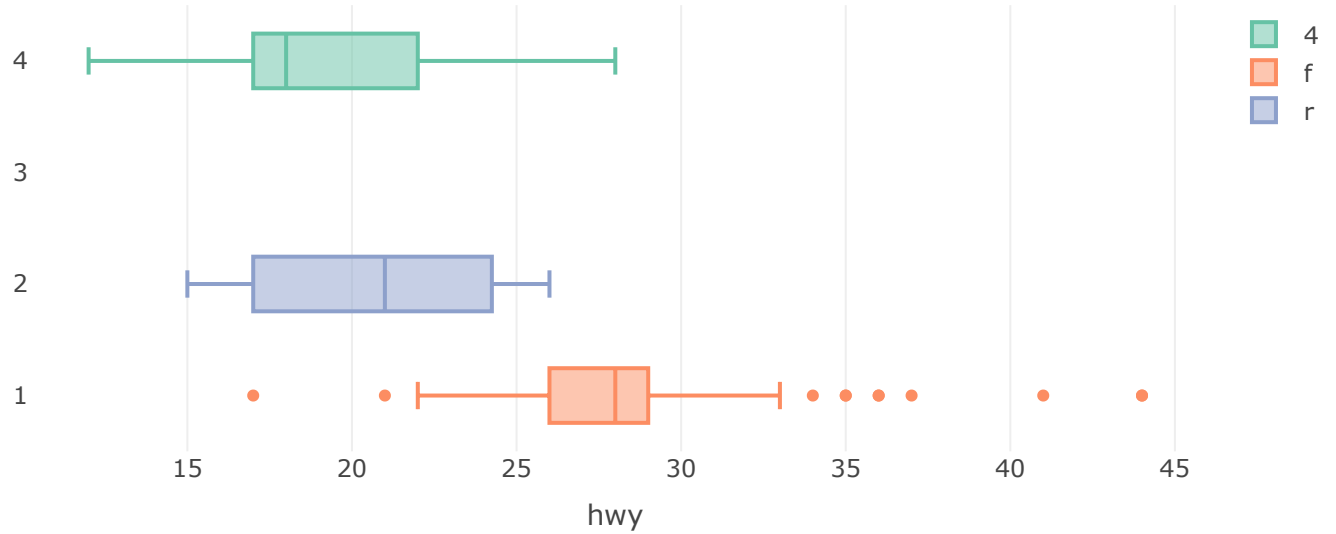
Try `ggplotly` with a time-series dataset

```
# install.packages("forecast")

p <- ggplot(fortify(forecast::gold), aes(x, y)) +
  geom_line()

ggplotly(p) %>% layout(dragmode = "pan")
```



Directly build a `plotly` plot

```
plot_ly(mpg, x = ~hwy, color = ~drv, type = "box")
```



Tuning **`plotly`** for your specific task can be time-consuming.

Please refer to the official website (https://plotly.com/r/plotly-fundamentals/) for guidance.

Exercise 5

- Take any ggplot that you have built above and change it to a `plotly` plot. Do you find the interactive plot more useful than the static plot?

A primer on R shiny

Install the shiny package

```
# install.packages("shiny")

library(shiny)
```

UI (User Interface)

```
# User Interface
ui <- fluidPage(
  selectInput("xVariable", "X-axis variable:",
              choices = c("displ", "hwy", "cty", "cyl"),
              selected = "displ"),
  selectInput("yVariable", "Y-axis variable:",
              choices = c("hwy", "cty", "cyl", "displ"),
              selected = "hwy"),
  selectInput("colorVariable", "Color points by:",
              choices = c("class", "drv", "fl"),
              selected = "class"),
  plotlyOutput("mpgPlot")
)
```

Server

```
# Server logic
server <- function(input, output) {
  output$mpgPlot <- renderPlotly({
    p <- ggplot(mpg,
                aes(x = .data[[input$xVariable]],
                    y = .data[[input$yVariable]],
                    color = .data[[input$colorVariable]])) +
      geom_point() +
      theme_minimal()

    ggplotly(p)
  })
}
```

Define the App

```
# Run the application
shinyApp(ui = ui, server = server)
```

Running the App

Store your ui, server, and shinyApp in an R script called `app.R` in a folder called my_app in your working directory, you can create the shiny website by running the line below.

Or you can just click on "Run App" at the right side of the lower tool bar.

```
runApp("my_app")
```

A list of examples

```
runExample("01_hello")      # a histogram
runExample("02_text")       # tables and data frames
runExample("03_reactivity") # a reactive expression
runExample("04_mpg")        # global variables
runExample("05_sliders")    # slider bars
runExample("06_tabsets")    # tabbed panels
runExample("07_widgets")    # help text and submit buttons
runExample("08_html")       # Shiny app built from HTML
runExample("09_upload")     # file upload wizard
runExample("10_download")   # file download wizard
runExample("11_timer")      # an automated timer
```

Let's learn these in future workshops.

Or just ask ChatGPT!