Module 2: Visualizing your data

Jake Ferguson (jakeferg@hawaii.edu)

Module goals

1. Present data visually.
2. Understand ggplot syntax
3. Identify approach figure types for your datatype

R's package system

Base R can be extended using a package. One popular collection of packages is the tidyverse.

"The tidyverse is an opinionated collection of R packages designed for data science. All packages share an underlying design philosophy, grammar, and data structures."

The tidyverse is designed for "wrangling" and presenting your data.

We are going to use the grammar of graphics package from the tidyverse

What does *gg* in ggplot2 stand for?

It stands for **Grammar of Graphics**:

```
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(
    mapping = aes(<MAPPINGS>),
    stat = <STAT>,
    position = <POSITION>
  ) +
  <COORDINATE_FUNCTION> +
  <FACET_FUNCTION>
```

You can uniquely describe any plot as a combination of these 7 parameters.

Why use ggplot2?

- Easily add colors, size, shape, to illustrate multiple factors.
- Easy superposition, facetting, etc.
- Good animation package
- Automatic legends

Why not use ggplot2?

- Syntax differs from base R
- Colors, size, shape can also be modified in base R
- For publication quality graphics both ggplot2 and base graphics will often require significant coding
- Default theme sucks

Different plots for different thoughts

Keep in mind

some plot characteristics will require **categorical** variables (color, faceting)

some plot charateristics will require **continuous** variables (y-axis)

some plot characteristics can be **either** continuous or categorical (point/line size, x-axis)

How you choose to display your information affects the types of conclusions that a reader will draw from the data.

The car mpg dataset

Data were collecated by the EPA on 38 models of car. The variables are

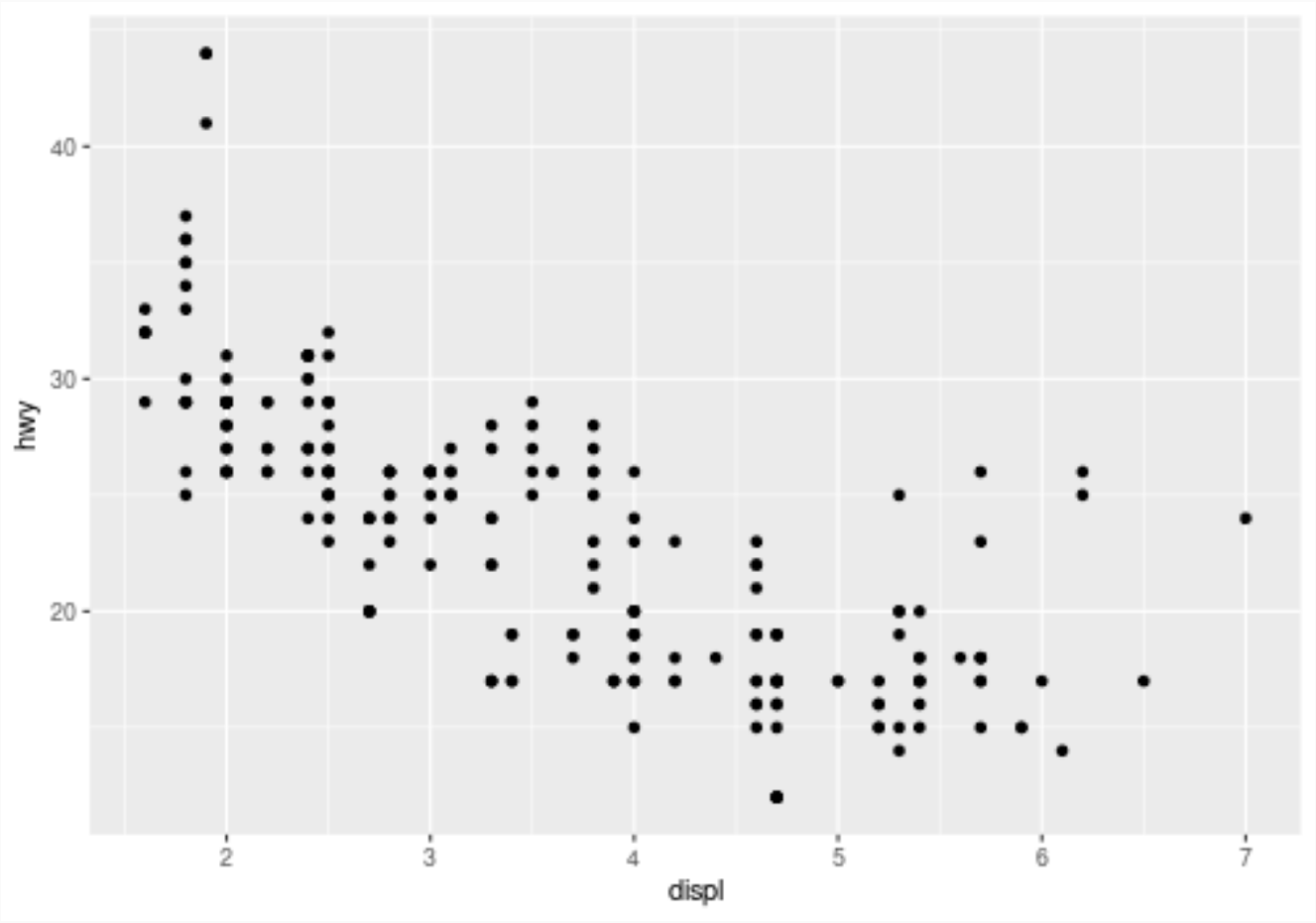
- **displ**, the car's engine size in liters and
- **hwy**, a cars fuel efficiency
- **class**, the type of car
- **year**, the model year

```
str(mpg)
```

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

Plotting miles per gallon

```
library(ggplot2) #loads the ggplot library.
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy))
```



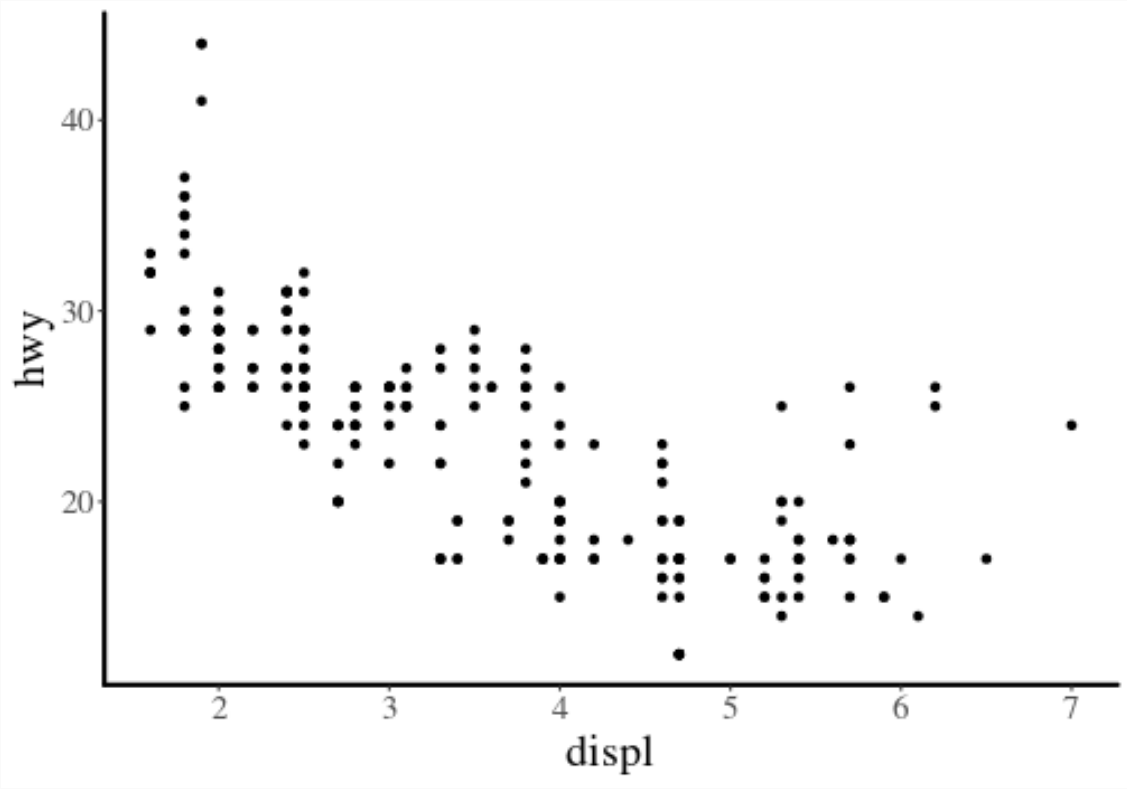
Excercise 2A

Build a scatterplot

Theming your plot

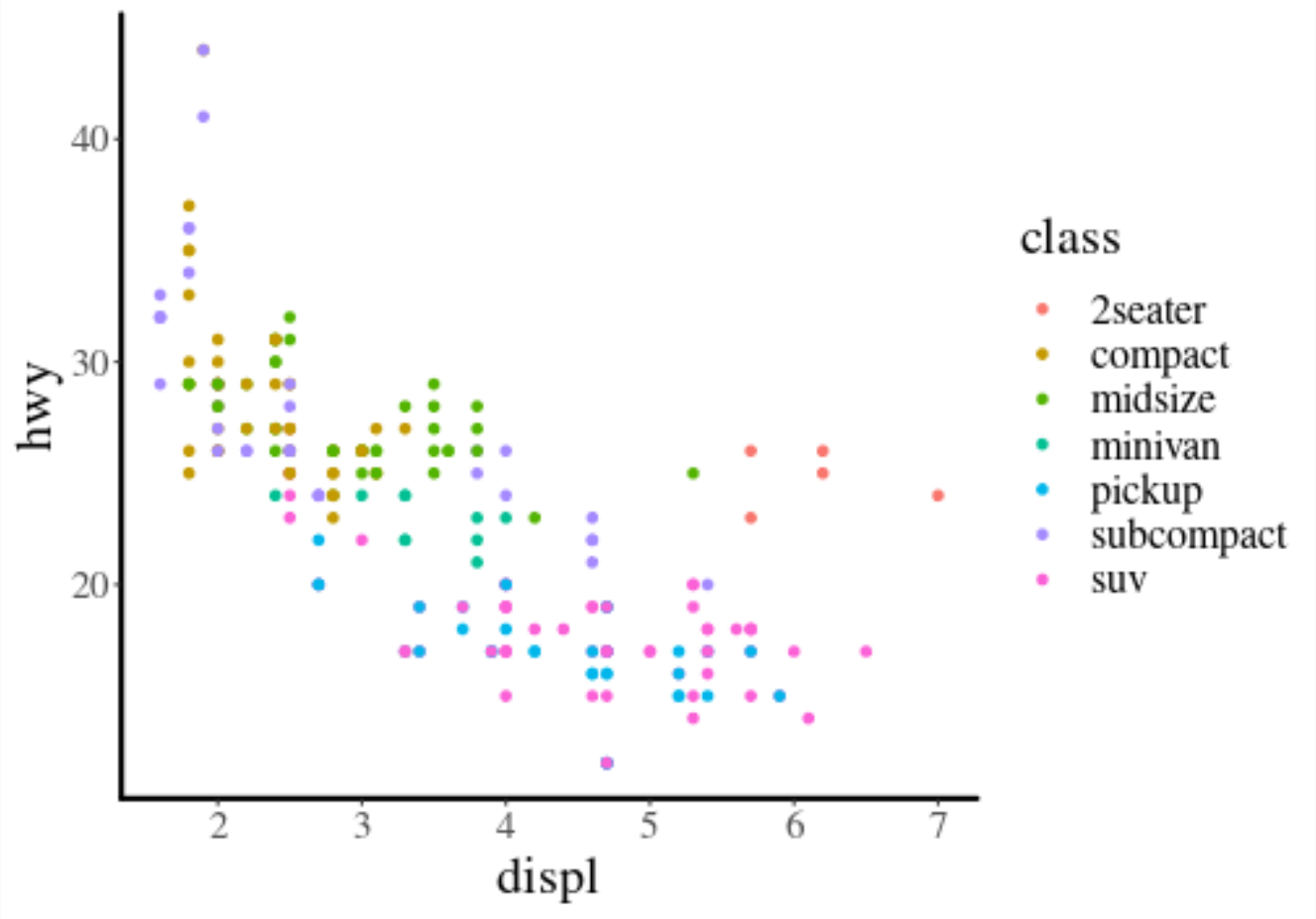
```
library(ggthemes) #an R library with more ggplot themes
theme_set(theme_tufte()) # a simple theme I like.
```

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



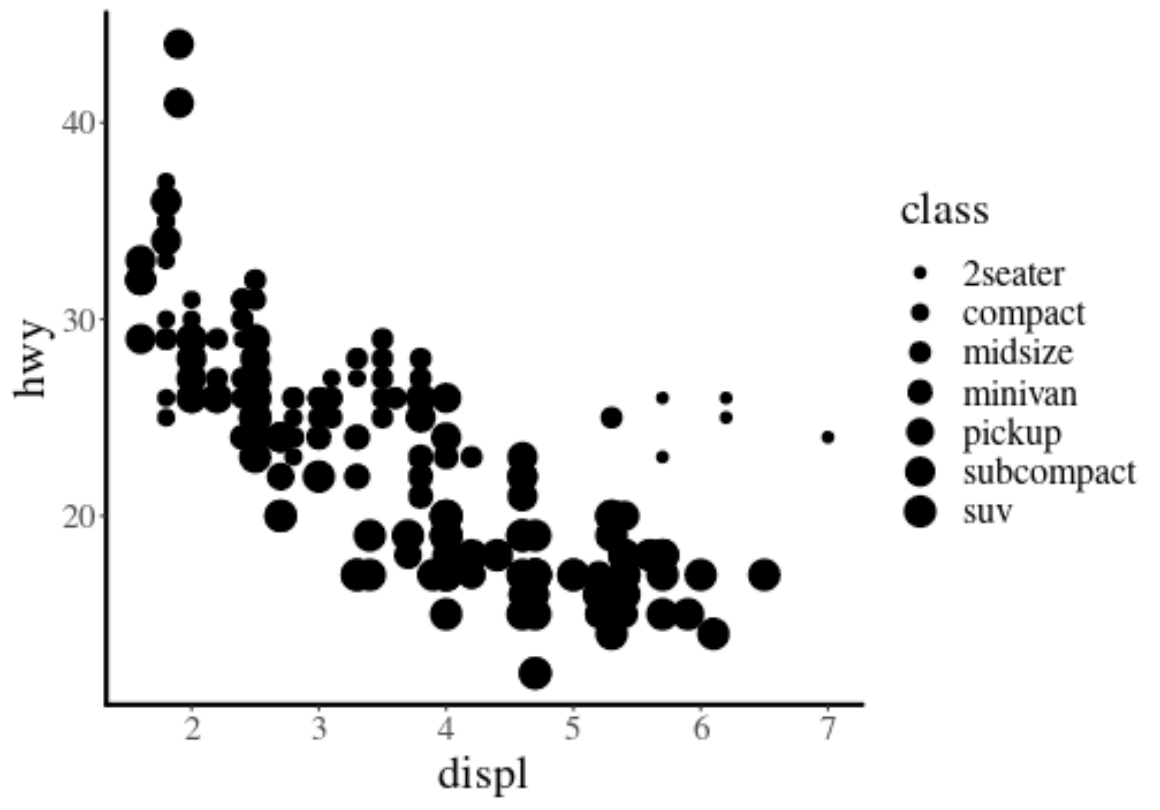
Adding `color`

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



Adding `size`

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

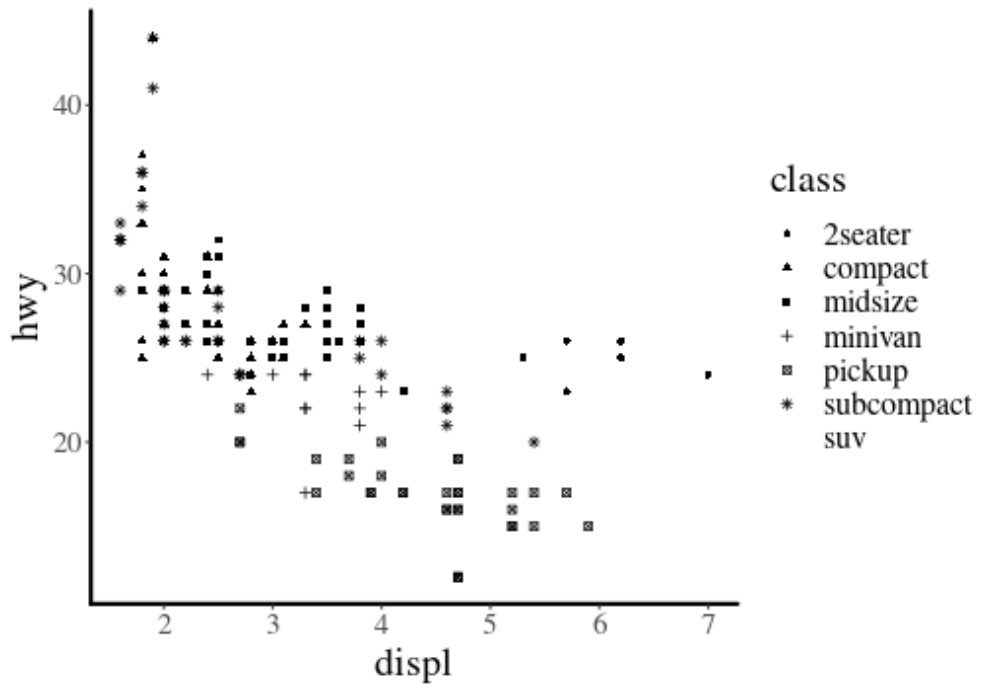


Adding shape

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

```
## Warning: 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: Removed 62 rows containing missing values (geom_point).
```

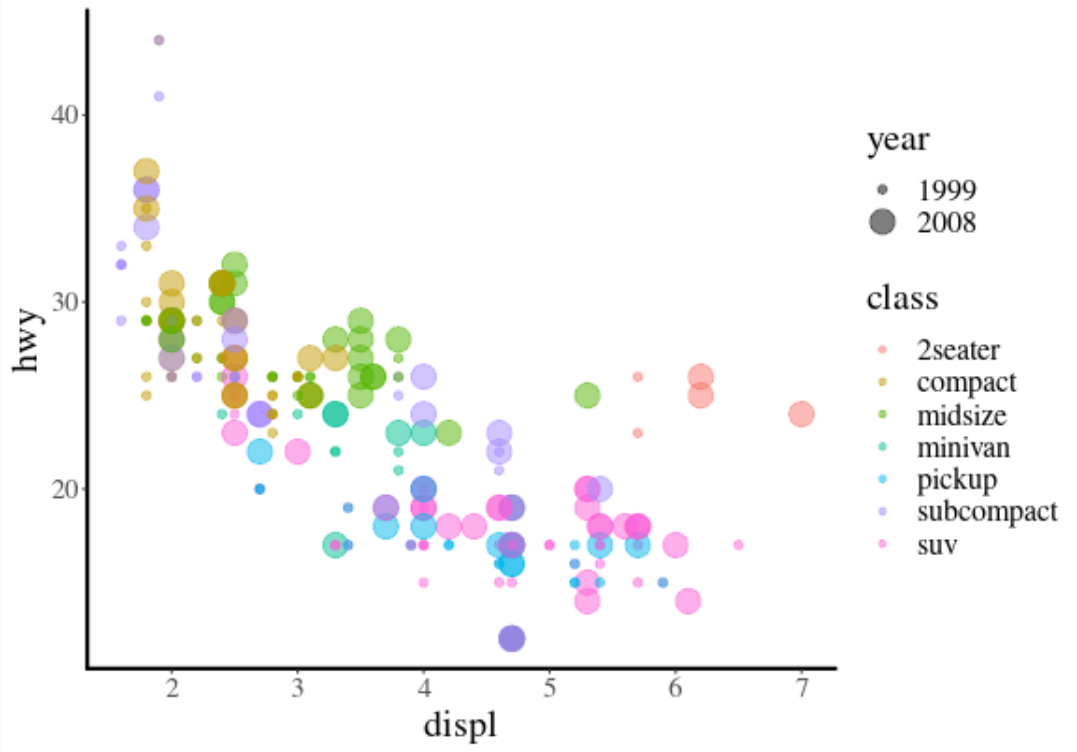


Adding transparency

`alpha` controls transparency

Editing arguments outside of `aes` will modify the all observations

```
ggplot(data=mpg) +
  geom_point(mapping=aes(x=displ, y=hwy, color=class, size=year), alpha=0.5)
```



Aspect & axis labels

```
ggplot(data=mpg) +
  geom_point(mapping=aes(x=displ, y=hwy, color=class, size=year)) +
  theme(aspect.ratio = 0.5) +
  xlab("Engine size") + ylab("Miles per gallon")
```

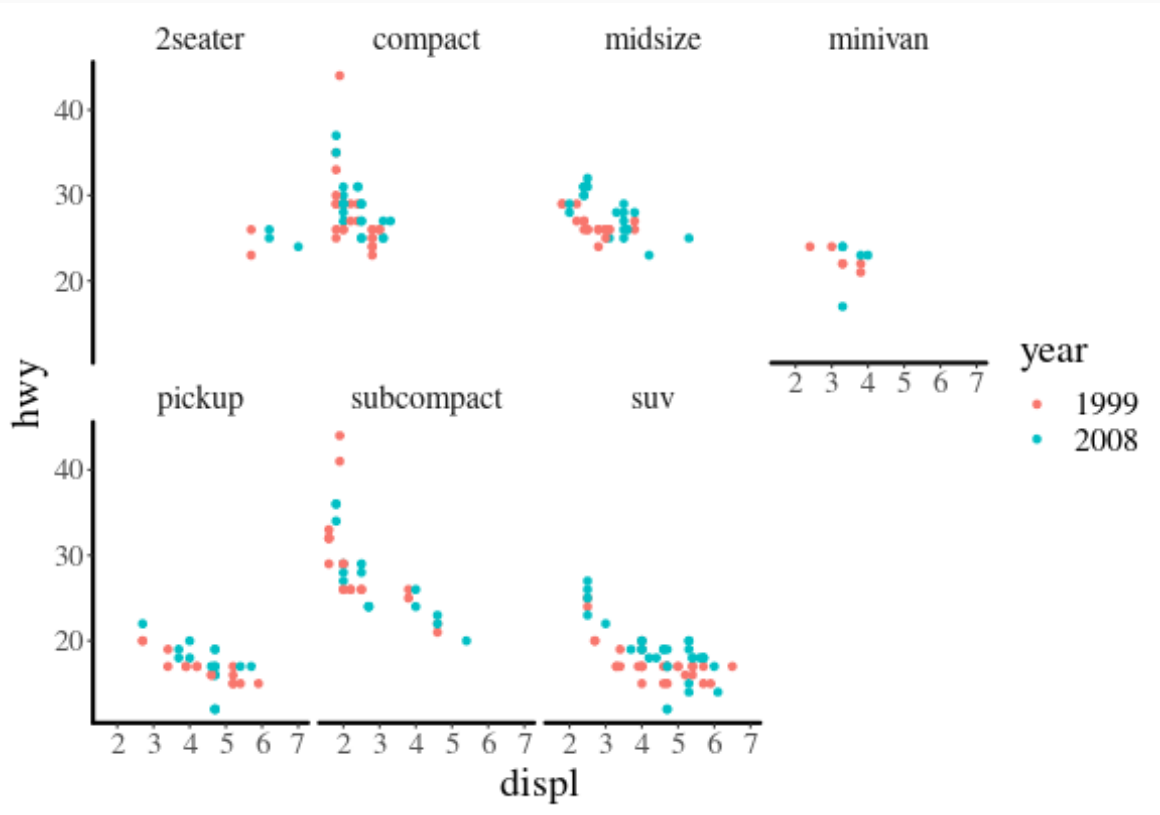
Excercise 2B

Polish your scatterplots

Faceting (multipanel figures)

Use `facet_wrap` to split your plot into facets that each display a subset of the data.

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



Other geometries

Fisher's iris data

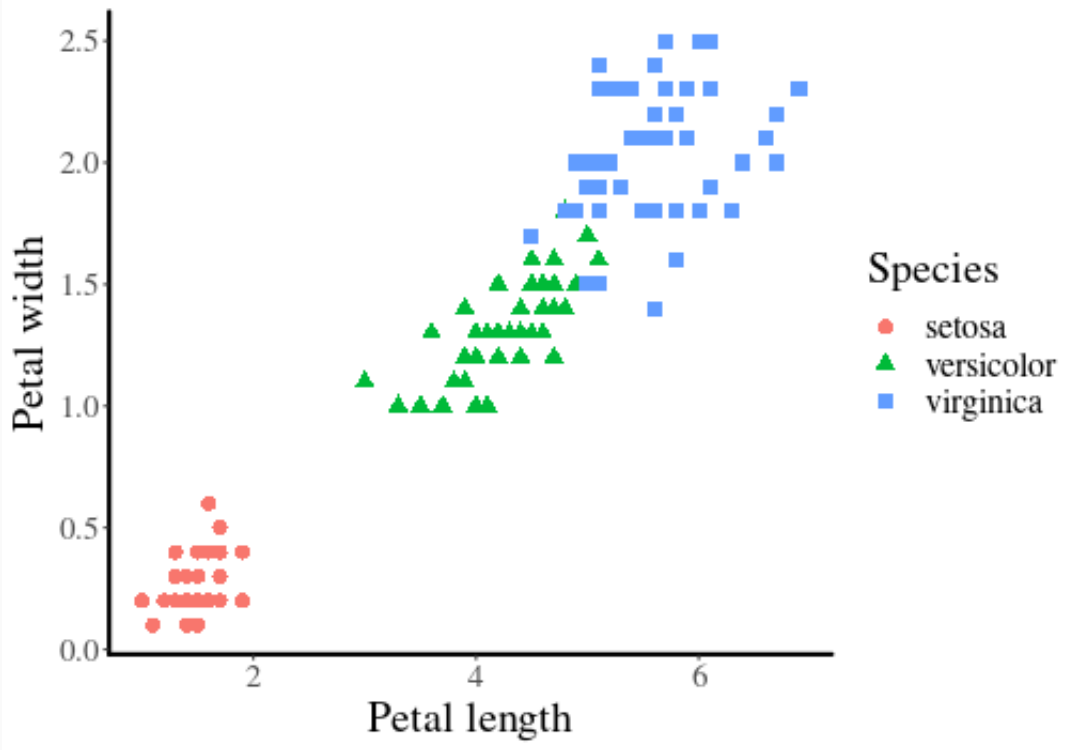
```
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

https://en.wikipedia.org/wiki/Iris_flower_data_set

Look at your base-layer first

```
ggplot(data=iris) +
  geom_point(mapping=aes(x=Petal.Length, y=Petal.Width, color = Species, shape = Spec
  xlab("Petal length") + ylab("Petal width")
```



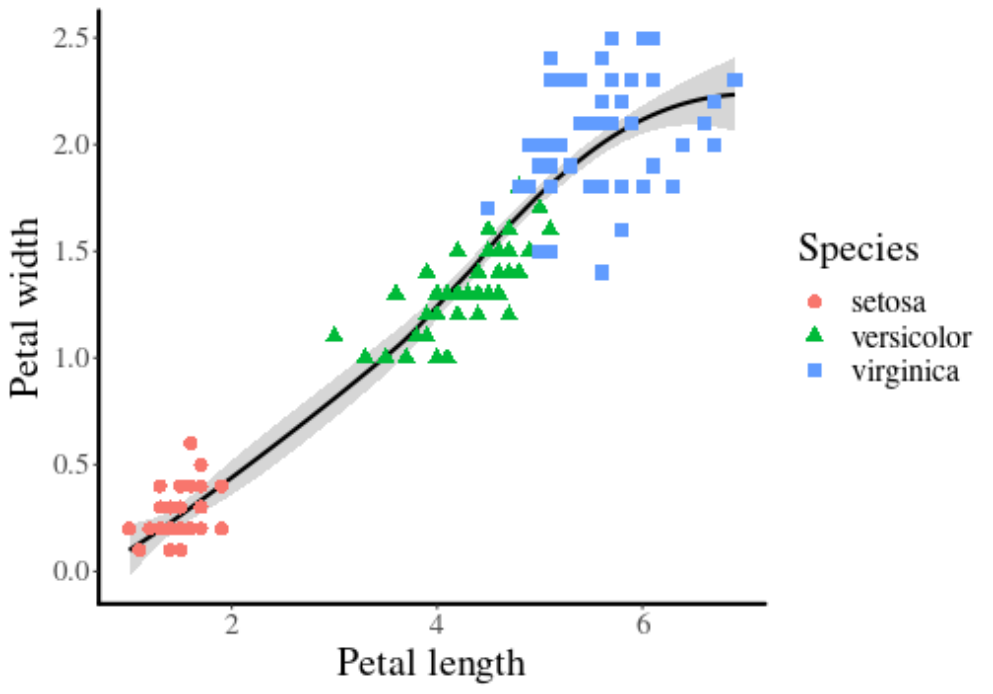
A smoother

The smooth geometry: `geom_smooth`

```
ggplot(data=iris) +
  geom_smooth(mapping=aes(x=Petal.Length, y=Petal.Width,
                color = Species), size = 1) +
    xlab("Petal length") + ylab("Petal width")
```

Combining geometries

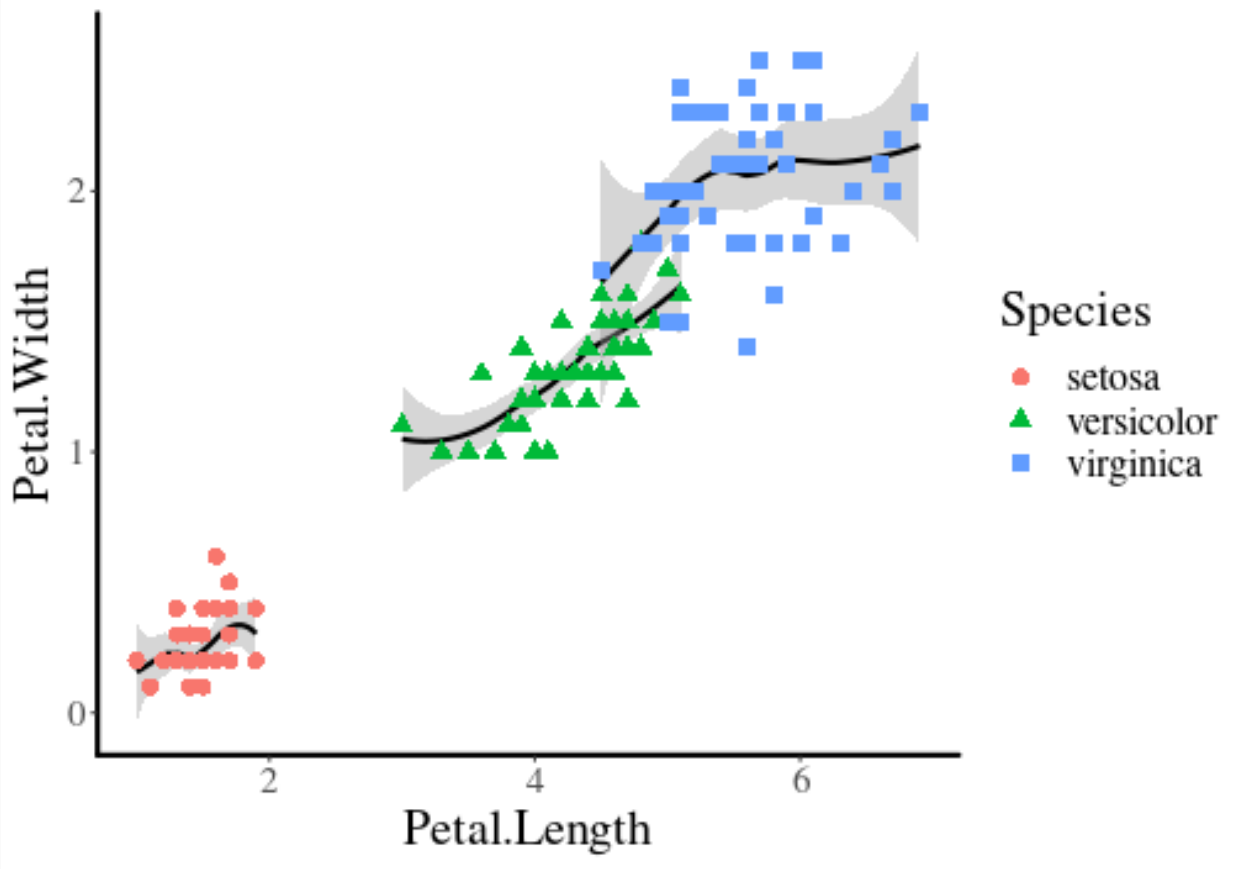
```
ggplot(data=iris) +
  geom_smooth(color = "black", mapping=aes(x=Petal.Length, y=Petal.Width)) +
  geom_point(mapping=aes(x=Petal.Length, y=Petal.Width, color = Species, shape = Spec
    xlab("Petal length") + ylab("Petal width")
```



Note that the order you give the geometries affects the order they are plotted.

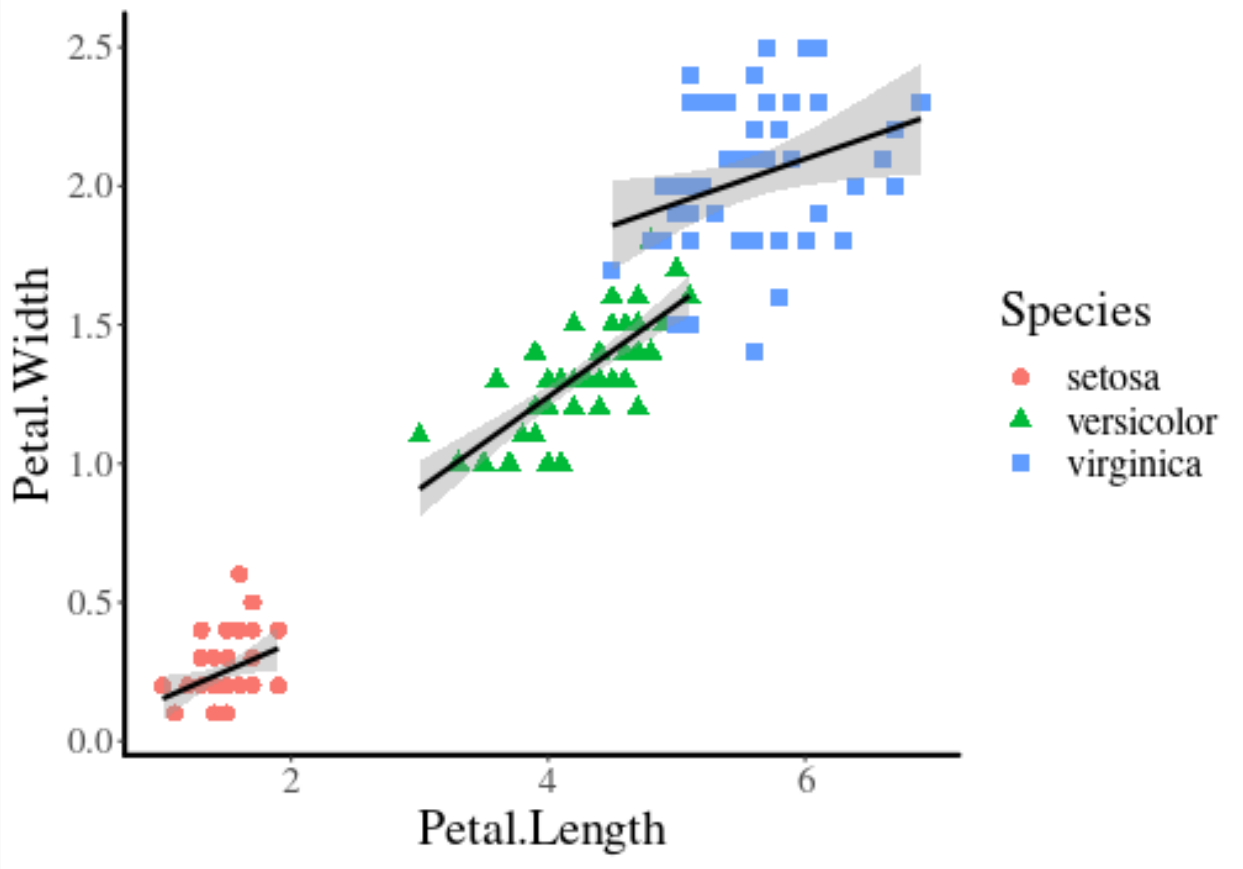
Smooth by group

```
ggplot(data=iris) +
  geom_smooth(mapping=aes(x=Petal.Length, y=Petal.Width, group = Species), color = "bl
  geom_point(mapping=aes(x=Petal.Length, y=Petal.Width, color = Species, shape = Speci
```



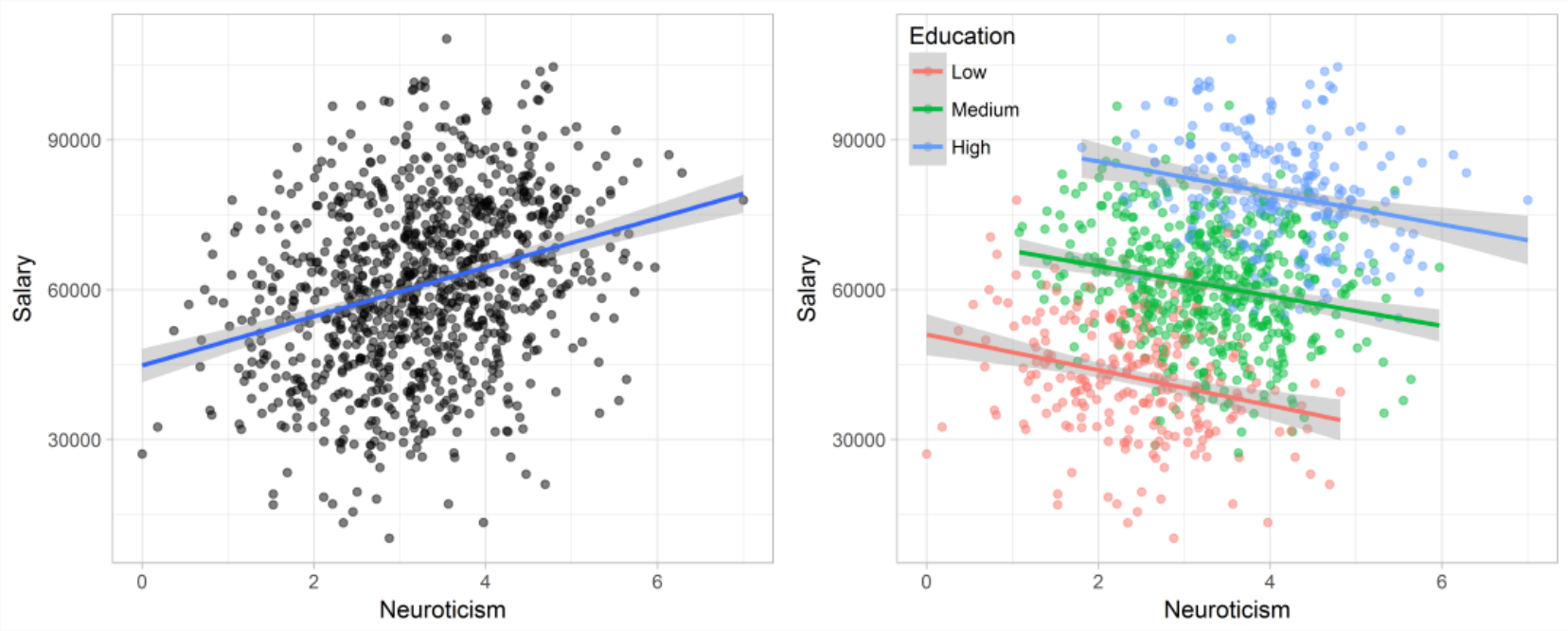
Use a linear model instead

```
ggplot(data=iris) +
   geom_point(mapping=aes(x=Petal.Length, y=Petal.Width, shape = Species, color=Specie
  geom_smooth(mapping=aes(x=Petal.Length, y=Petal.Width, group = Species), color="blac
```



An important application

Simpson's Paradox



Source: https://goo.gl/GycYod

Excercise 2C

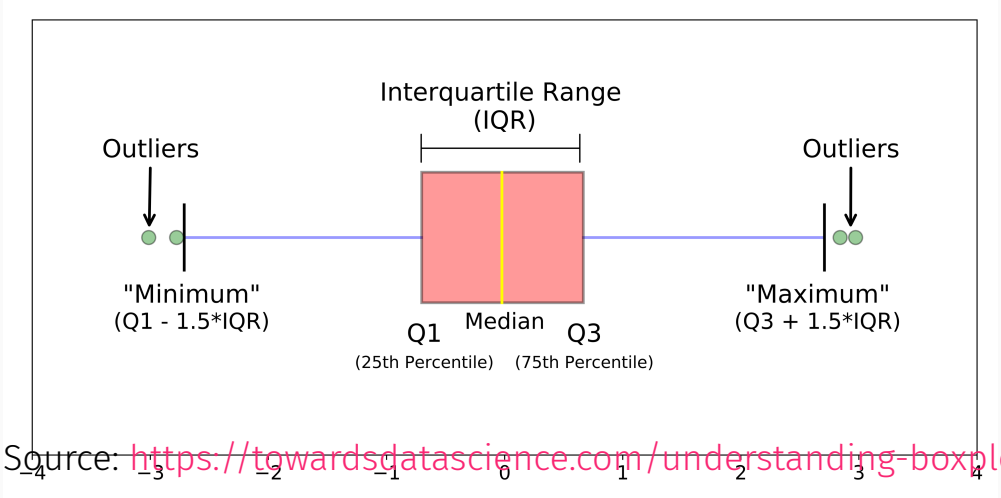
Combining geometries

The boxplot

Numerical response, categorical covariate

The box - first, second (median), and third quartiles

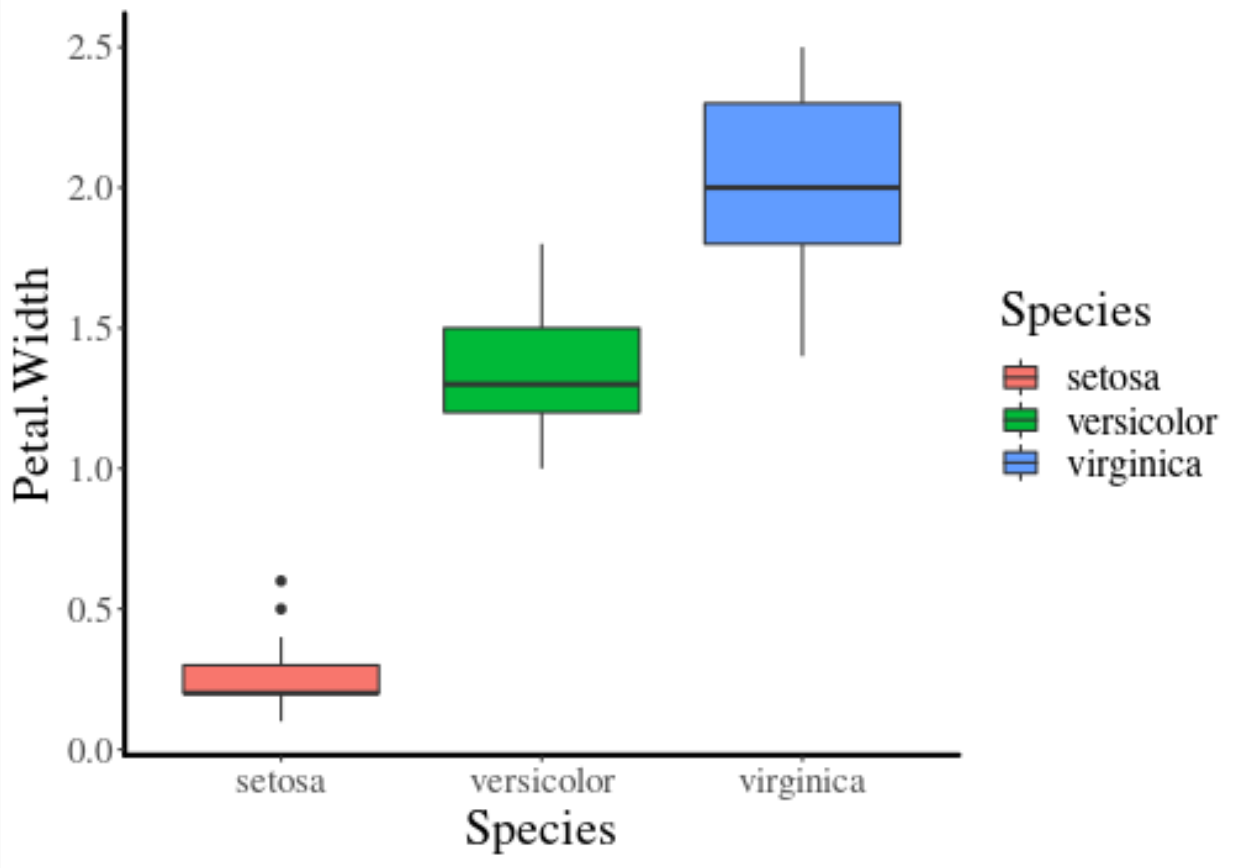
The whiskers - "The upper whisker extends from the hinge to the largest value no further than 1.5 *IQR from the hinge (where IQR is the inter-quartile range, or distance between the first and third quartiles). The lower whisker extends from the hinge to the smallest value at most 1.5* IQR of the hinge. Data beyond the end of the whiskers are called "outlying" points and are plotted individually."



Source: https://towardsdatascience.com/understanding-boxplots-5e2df7bcbd51

The `geom_boxplot` geometry

```
ggplot(data=iris) +
  geom_boxplot(mapping=aes(x=Species, y=Petal.Width, fill=Species))
```



Excercise 2D

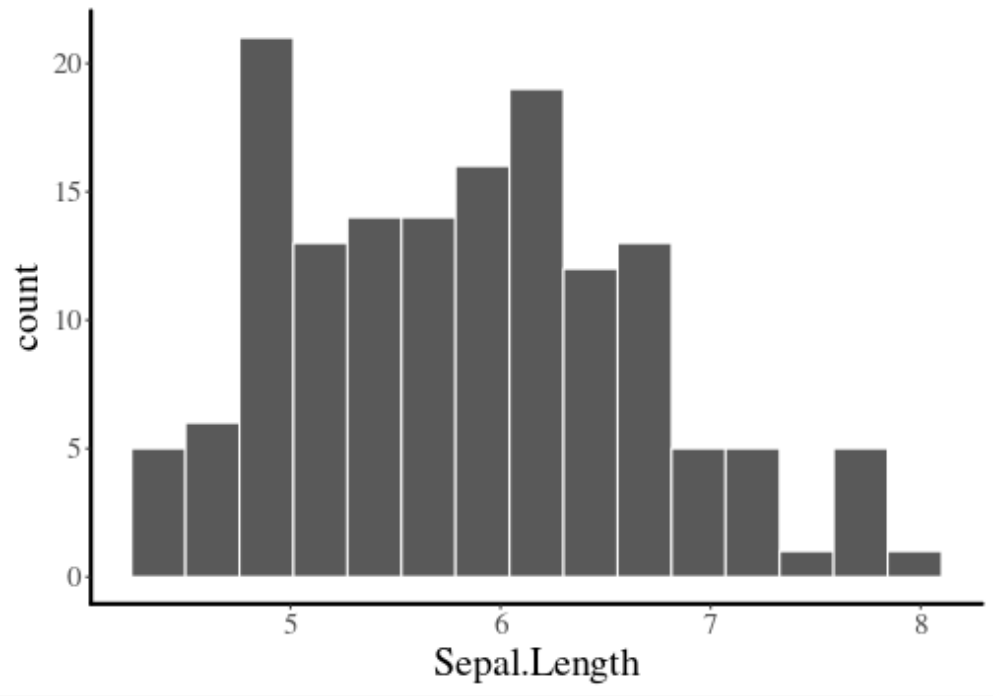
Build a boxplot

A single continuous variable

histogram: `geom_histogram`

Histograms pile observations into predetermined bins

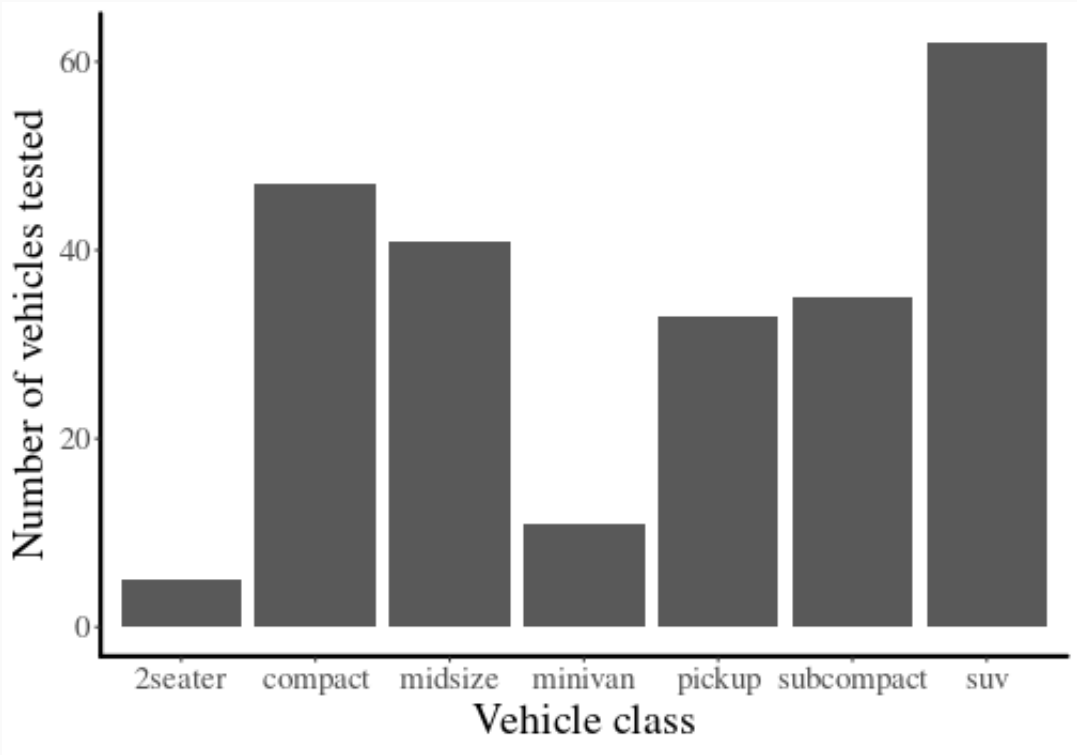
```
ggplot(data=iris) +
  geom_histogram(mapping=aes(x=Sepal.Length), bins=15, color="white")
```



A single categorical variable

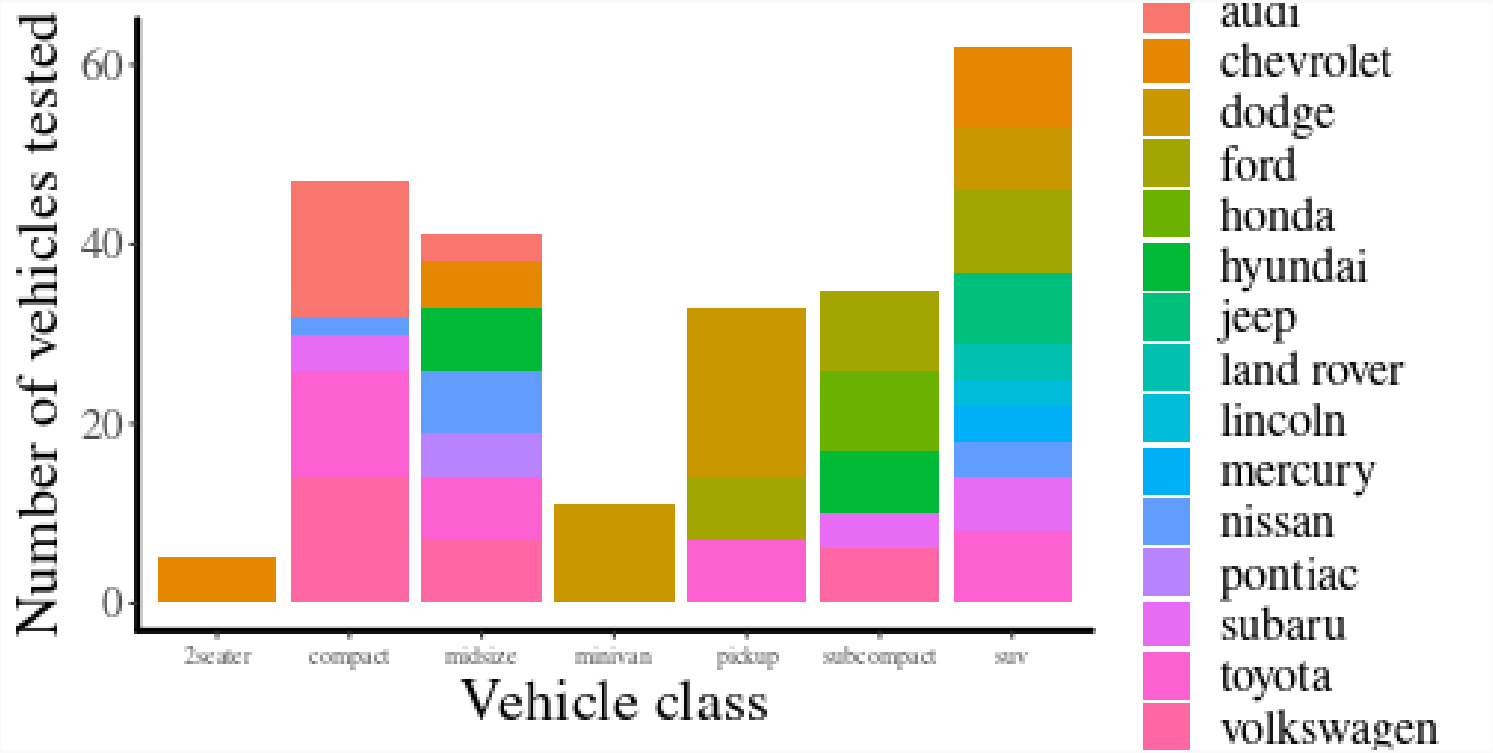
bar charts: `geom_bar()`

```
ggplot(data=mpg) +
  geom_bar(mapping=aes(x=class)) +
  xlab("Vehicle class") + ylab("Number of vehicles tested")
```



Multiple categorical variables

```
ggplot(data=mpg) +
  geom_bar(mapping=aes(x=class, fill=manufacturer)) +
  xlab("Vehicle class") + ylab("Number of vehicles tested") +
  theme(axis.text.x = element_text(size = 8))
```



Proportions

Use `position="fill"` to turn counts into proportions

```
ggplot(data=mpg) +
  geom_bar(mapping=aes(x=manufacturer, fill=class), position="fill") +
  xlab("Manufacturer") + ylab("Number of vehicles tested") +
  theme(axis.text.x = element_text(size = 8), aspect.ratio=0.5)
```

Excercise 2E

Univariate plots

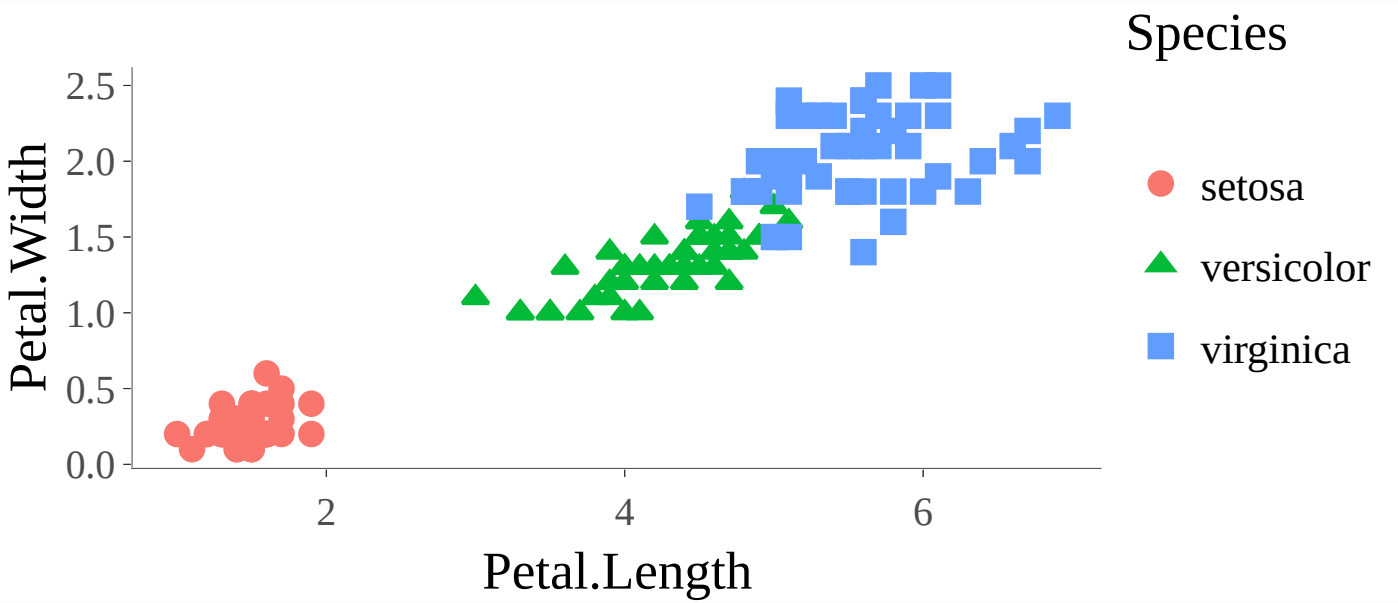
Interactive plots

1. Build your `ggplot`

```
p ← ggplot(data=iris) + geom_point(mapping=aes(x=Petal.Length, y=Petal.Width, color=S
```

2. convert your ggplot to plotly

```
library(plotly, quietly=F)
ggplotly(width = 700, height = 300)
```



Resources

- Cheatsheet
- Spatial Visualization
- Heatmaps
- Cookbook for R - Graphs
- Top 50 ggplot2 Visualizations
- Using color brewer palettes
- Publication Ready Plots