Data Visualization with ggplot2

Session 1: The Grammar of Graphics

Nathan Barron
nathanbarron@ou.edu

Graphics are like onions

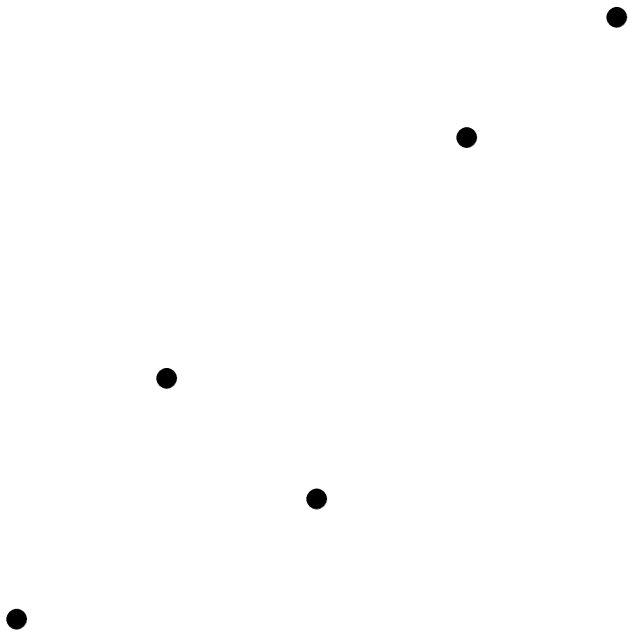
The `ggplot2` package in R is designed with a **grammar**, meaning that there are *core principles* that can be applied to create nearly *any* type of graphic.

- Data: the information you want to visualize
- Mapping: the description of how the data's variables are "mapped" to aesthetic attributes

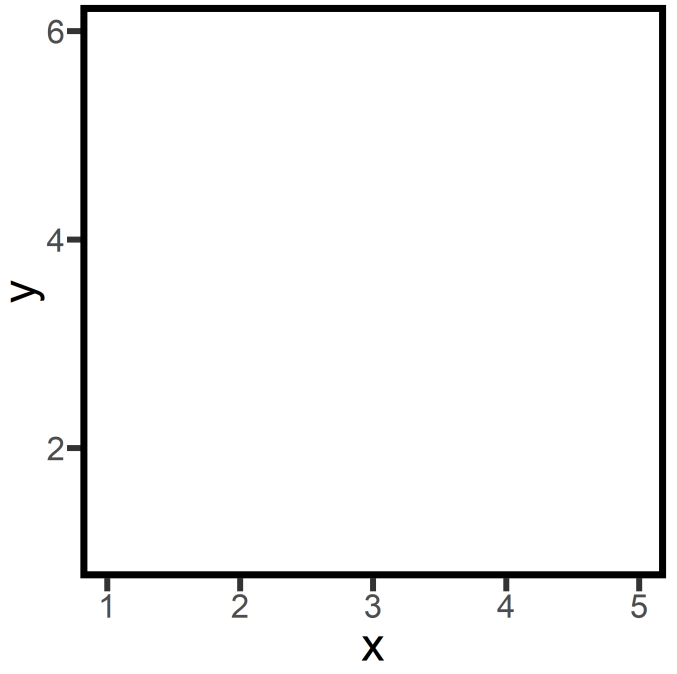
Essential Mapping Components

- Layer: collection of geometric elements (or *geom*s)
- Geom: visual components of a plot (e.g. points, lines, bars, etc.)
- Scales: map values in the data space to values in the aesthetic space (e.g. color, shape, size, etc.)
- Coord: coordinate system (e.g., the axes and gridlines)
- Facet: how to display subsets of data
- Theme: display features (e.g. font size, background color, etc.)

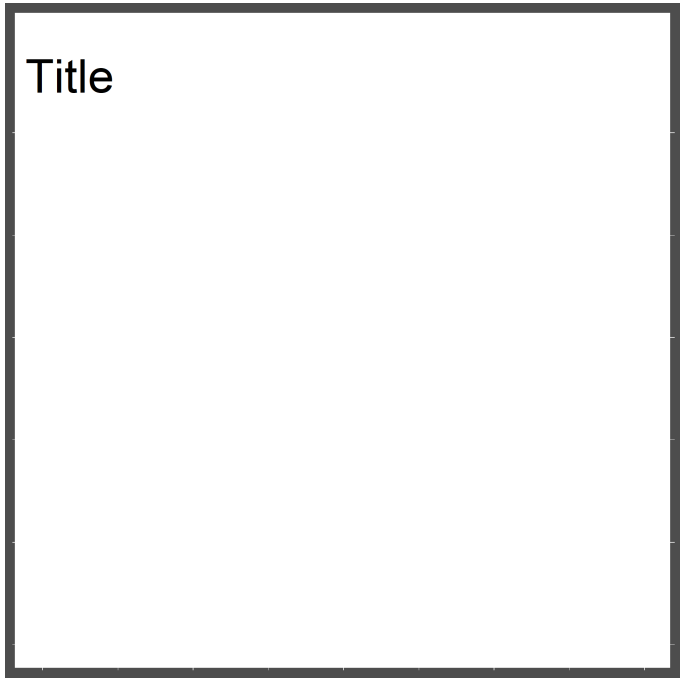
Plot layers



These are the `geom_point` elements. They are the visual representations of your data.



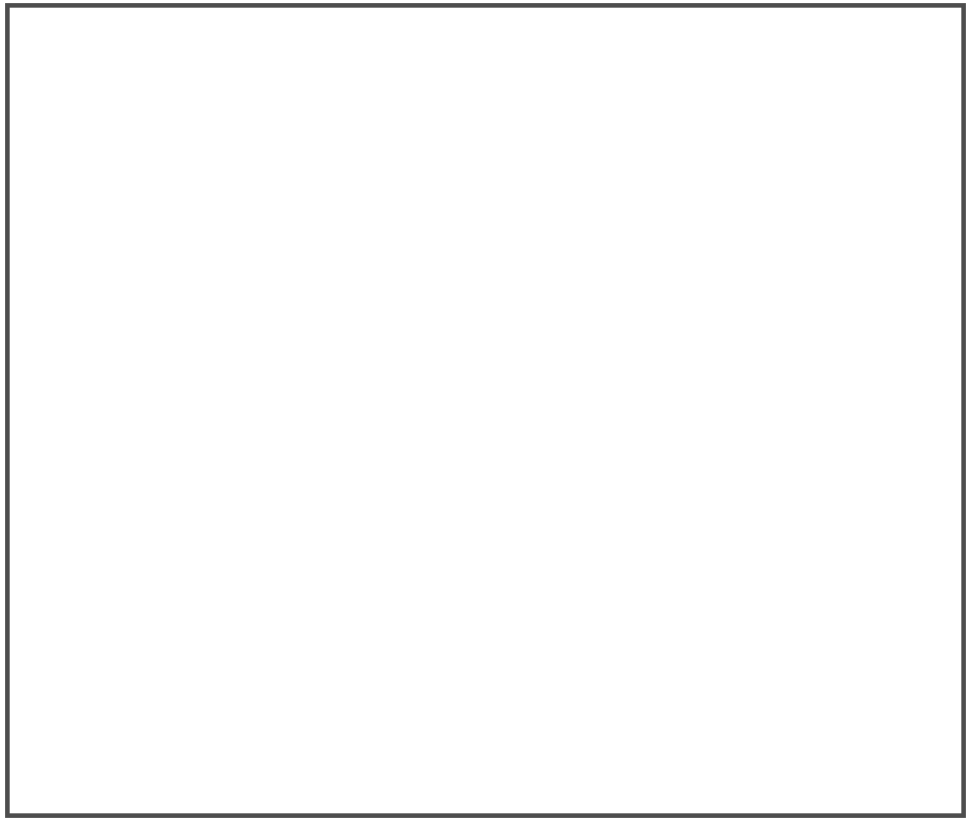
These are the scales and coordinate system. They are the axes and legends so that we can read values from the graph



These are the plot annotations. They provide crucial information for easy interpretation.

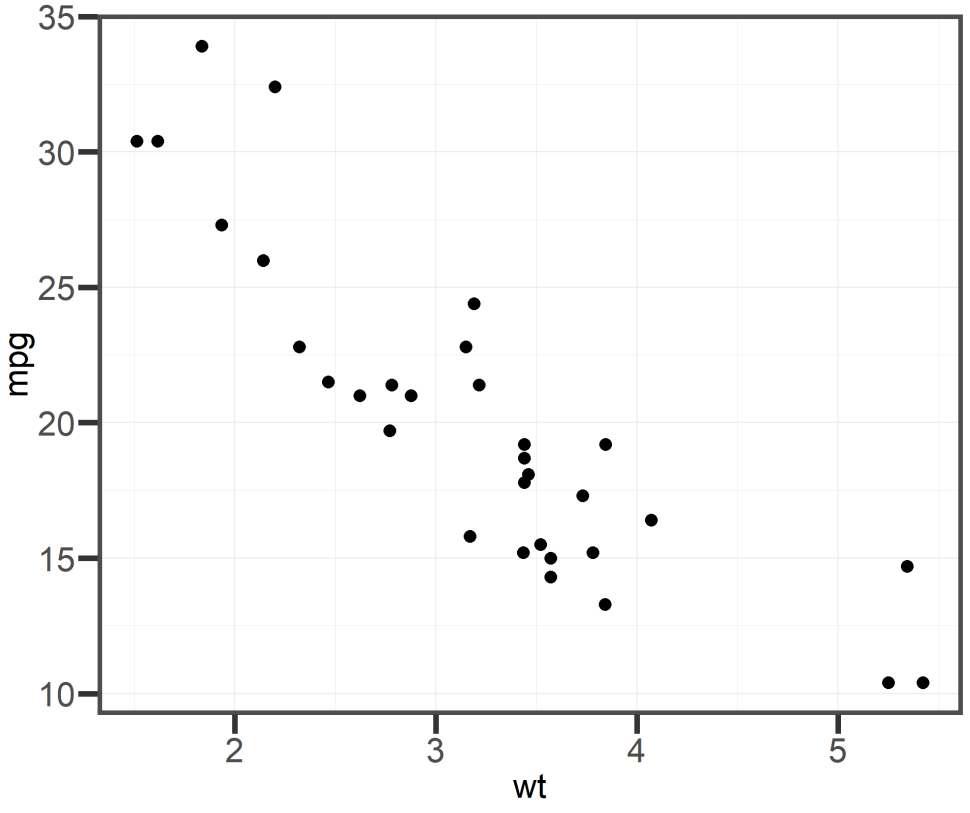
The foundation

```
1  ggplot(mtcars)
```



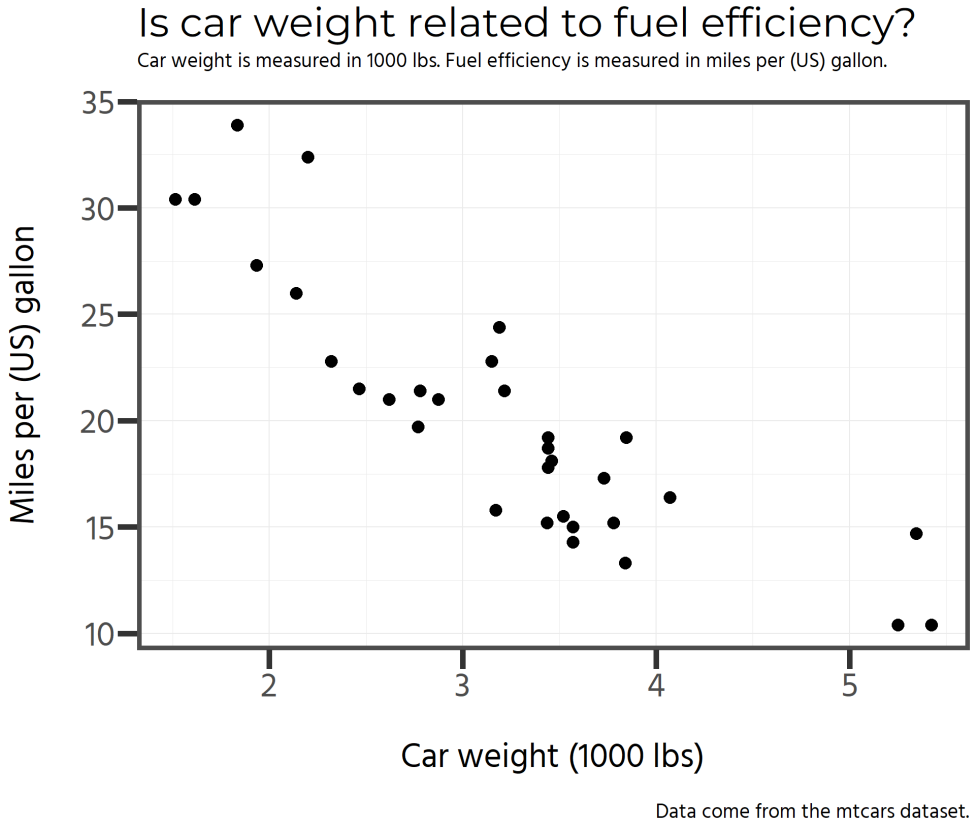
Plotting the data

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



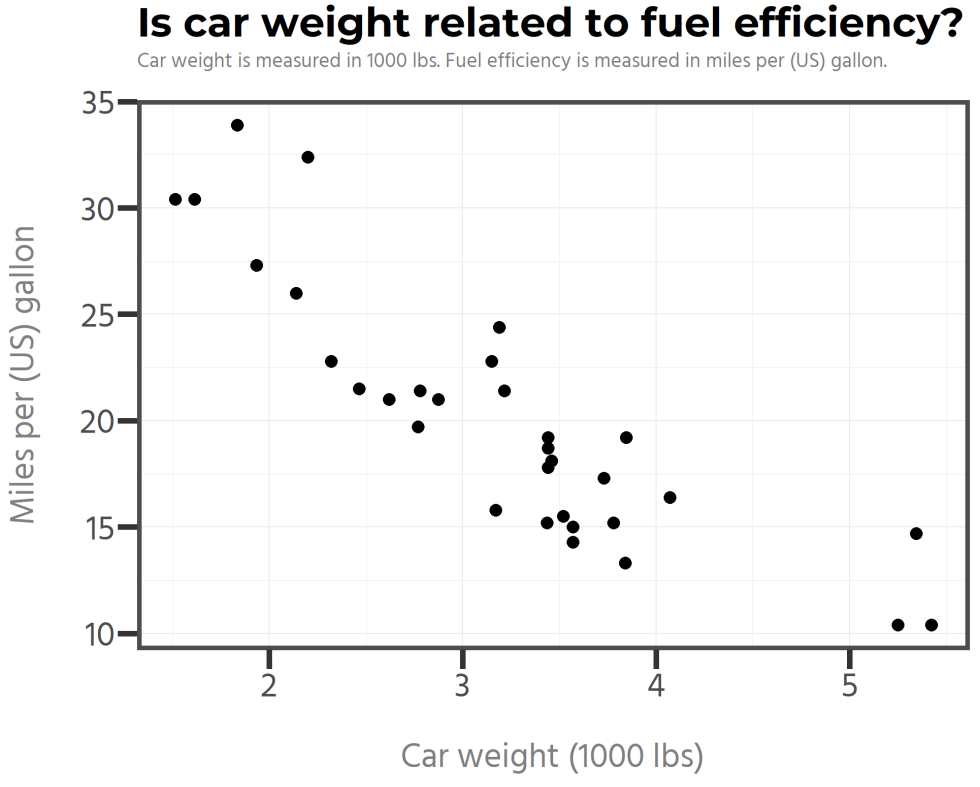
Annotating your plot

```
ggplot(mtcars) +
  geom_point(aes(x=wt, y=mpg)) +
  labs(x='Car weight (1000 lbs)',
       y='Miles per (US) gallon',
       title = 'Is car weight related to fuel efficiency?
       subtitle = 'Car weight is measured in 1000 lbs. Fu
       caption = 'Data come from the mtcars dataset.')
```



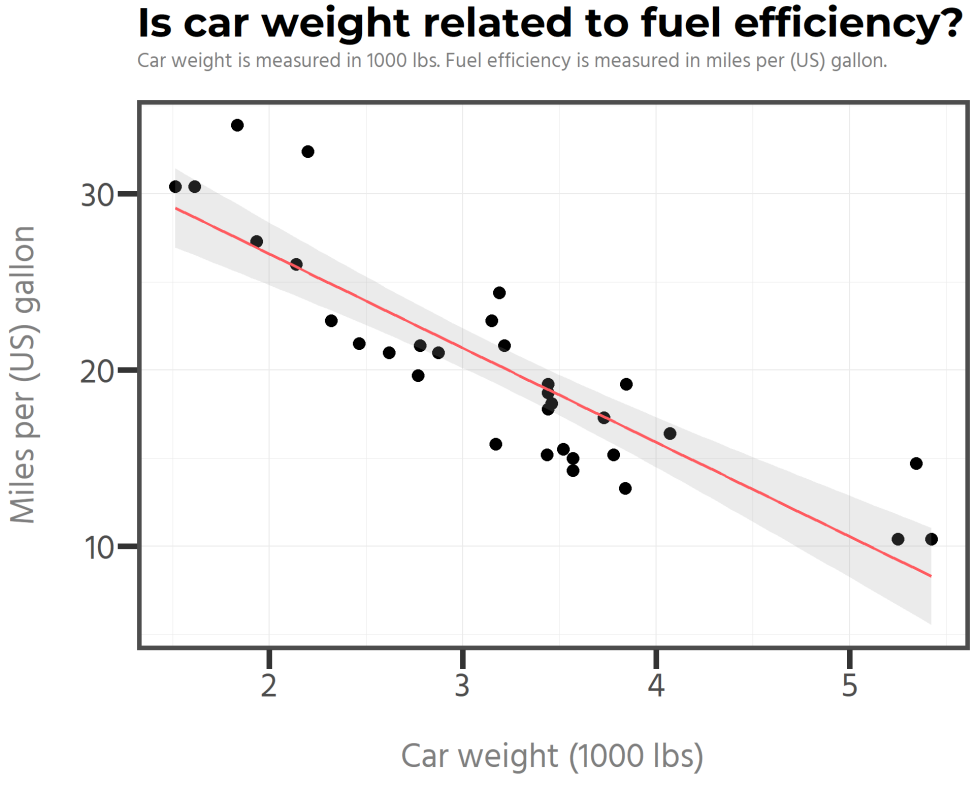
Changing the theme

```
ggplot(mtcars) +
  geom_point(aes(x=wt, y=mpg)) +
  labs(x='Car weight (1000 lbs)',
       y='Miles per (US) gallon',
       title = 'Is car weight related to fuel efficiency?
       subtitle = 'Car weight is measured in 1000 lbs. Fu
       caption = 'Data come from the mtcars dataset.') +
  theme(axis.title = element_text(color = 'grey50'),
        plot.title = element_text(face = 'bold'),
        plot.subtitle = element_text(color = 'grey50'),
        plot.caption = element_text(face = 'italic'))
```



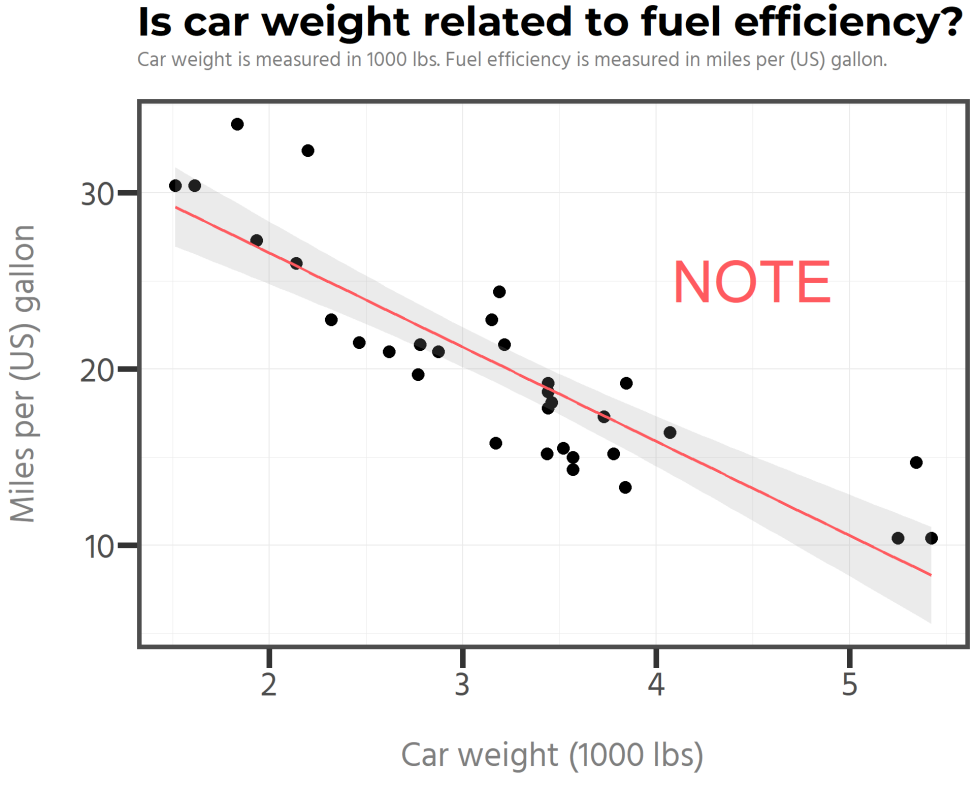
Adding new layers (fitted line)

```
ggplot(mtcars) +
  geom_point(aes(x=wt, y=mpg)) +
  geom_smooth(aes(x=wt, y=mpg),
              method = 'lm',
              alpha = .20,
              color = '#FF5A5F')+
  labs(x='Car weight (1000 lbs)',
       y='Miles per (US) gallon',
       title = 'Is car weight related to fuel efficiency?
       subtitle = 'Car weight is measured in 1000 lbs. Fu
       caption = 'Data come from the mtcars dataset.') +
  theme(axis.title = element_text(color = 'grey50'),
        plot.title = element_text(face = 'bold'),
        plot.subtitle = element_text(color = 'grey50'),
        plot.caption = element_text(face = 'italic'))
```



Adding new layers (annotations)

```
ggplot(mtcars) +
  geom_point(aes(x=wt, y=mpg)) +
  geom_smooth(aes(x=wt, y=mpg),
              method = 'lm',
              alpha = .20,
              color = '#FF5A5F') +
  annotate("text", x=4.5, y=25, label = 'NOTE', color = '
  labs(x='Car weight (1000 lbs)',
       y='Miles per (US) gallon',
       title = 'Is car weight related to fuel efficiency?
       subtitle = 'Car weight is measured in 1000 lbs. Fu
       caption = 'Data come from the mtcars dataset.') +
  theme(axis.title = element_text(color = 'grey50'),
        plot.title = element_text(face = 'bold'),
        plot.subtitle = element_text(color = 'grey50'),
        plot.caption = element_text(face = 'italic'))
```

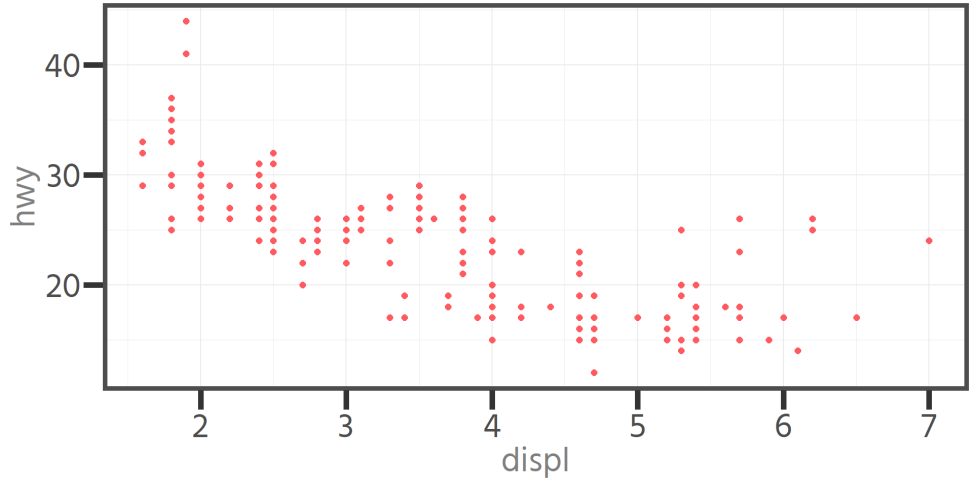


Aesthetic attributes

Aesthetics are parameters that can change the appearance of a plot.

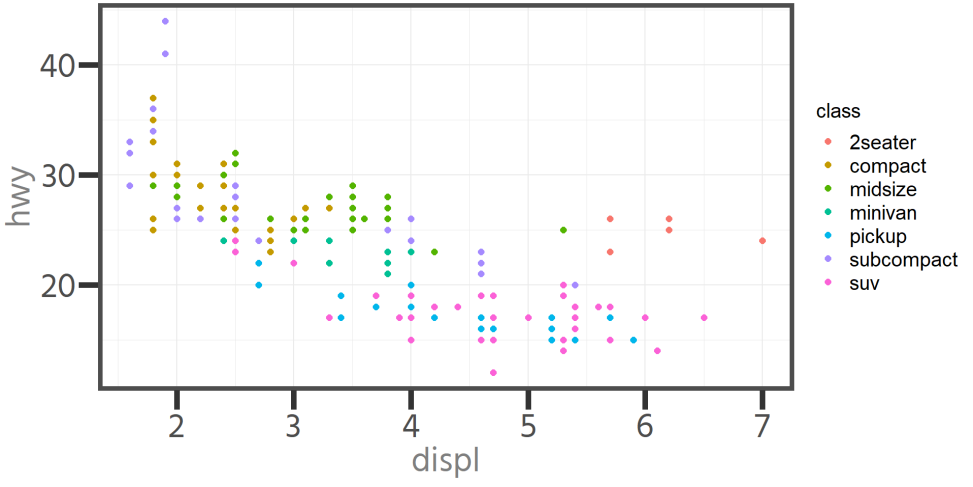
Global aesthetics affect all data at the same time.

```
ggplot(mpg) +
  geom_point(aes(displ, hwy),
             color = 'red')
```

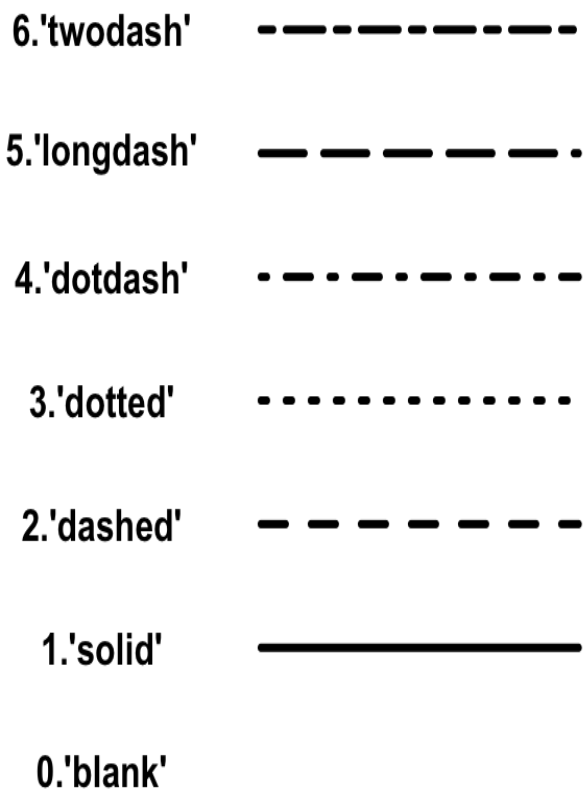


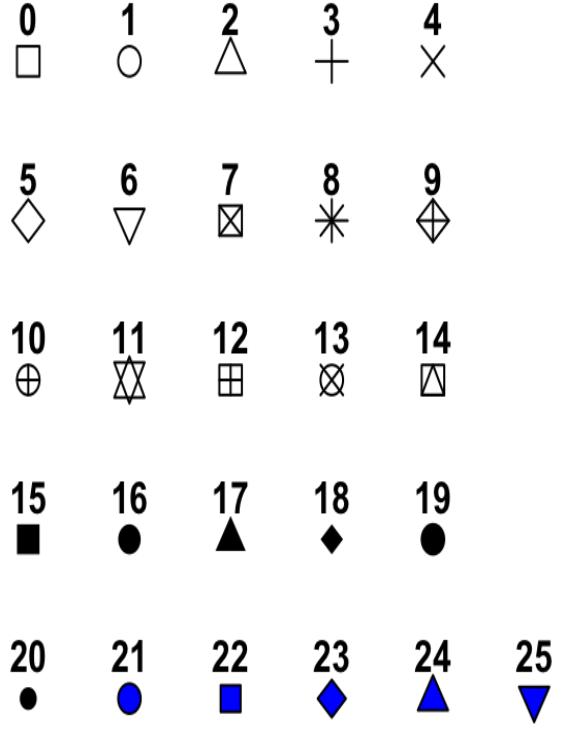
Mapped aesthetics affect data based on the values of some specified variable.

```
ggplot(mpg) +
  geom_point(aes(displ, hwy, color = class))
# Notice that 'color = ____' is now inside aes()
```



Linetype and shape aesthetics





Scales

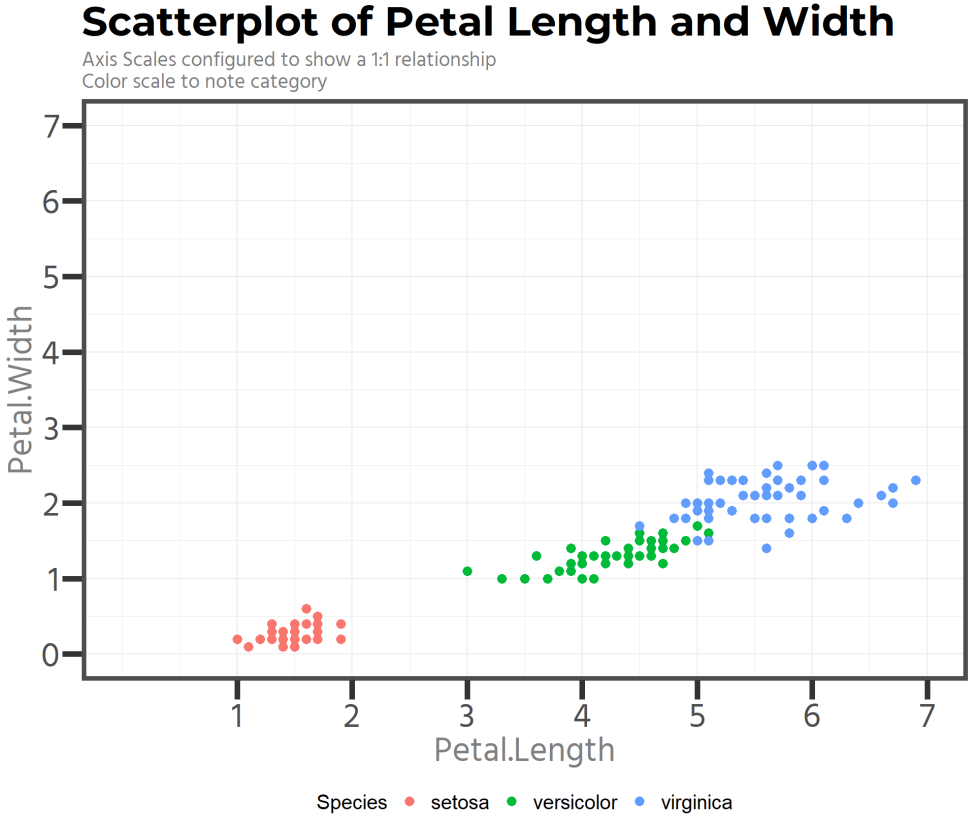
Scales refer to how data are translated into visual representations. This includes the x-axis, y-axis, colors, shapes, etc. Scales are essential components of the mapping process.

Key Elements

Name: a basic, though often neglected, description of scale

Limits: the range of a scale

Breaks: the subdivisions of range used to mark gradations



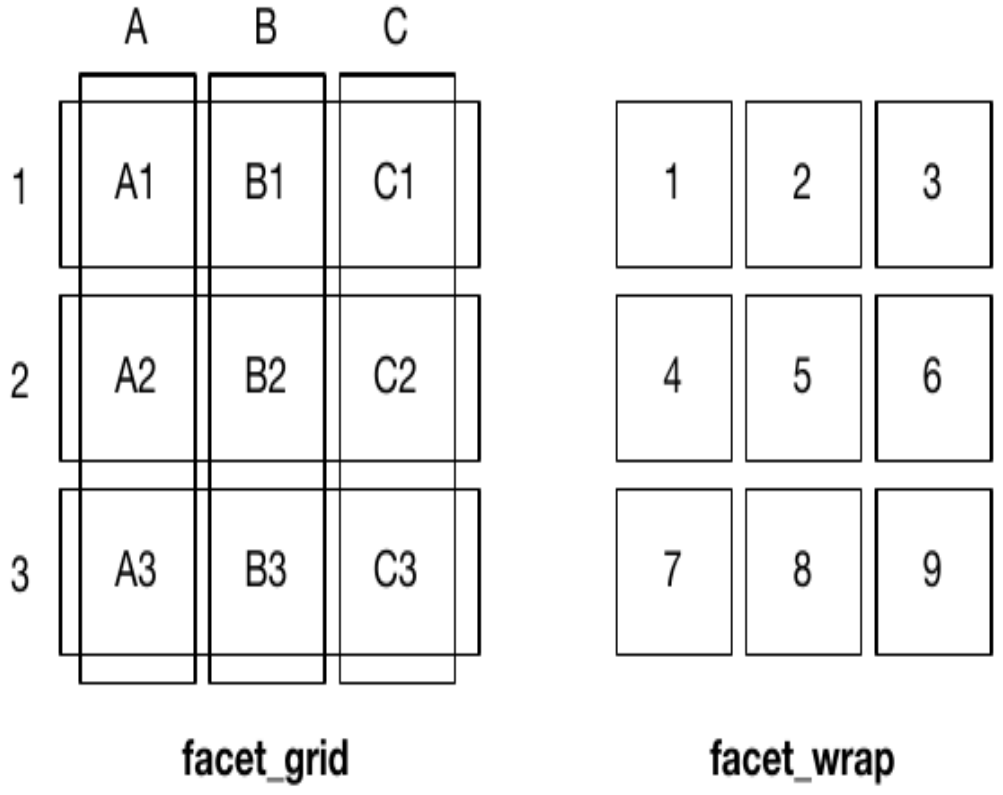
Facets

Facets show subsets of the data in a sequential, side-by-side format.

Two types of facets:

`facet_wrap`:
Subsetting data on one variable

`facet_grid`:
Subsetting data on more than one variable



Exercise #1

Data Visualization with ggplot2

Session 2: Building a Custom Theme

Nathan Barron
nathanbarron@ou.edu

The `theme` layer

The `theme()` function allows you to override default graphics settings. There are two main components involved:

Elements: non-data elements of the graphic that you can control

Examples:

- `plot.title`
- `axis.ticks.x`
- `legend.key.height`
- etc.

Element functions: the description of element features

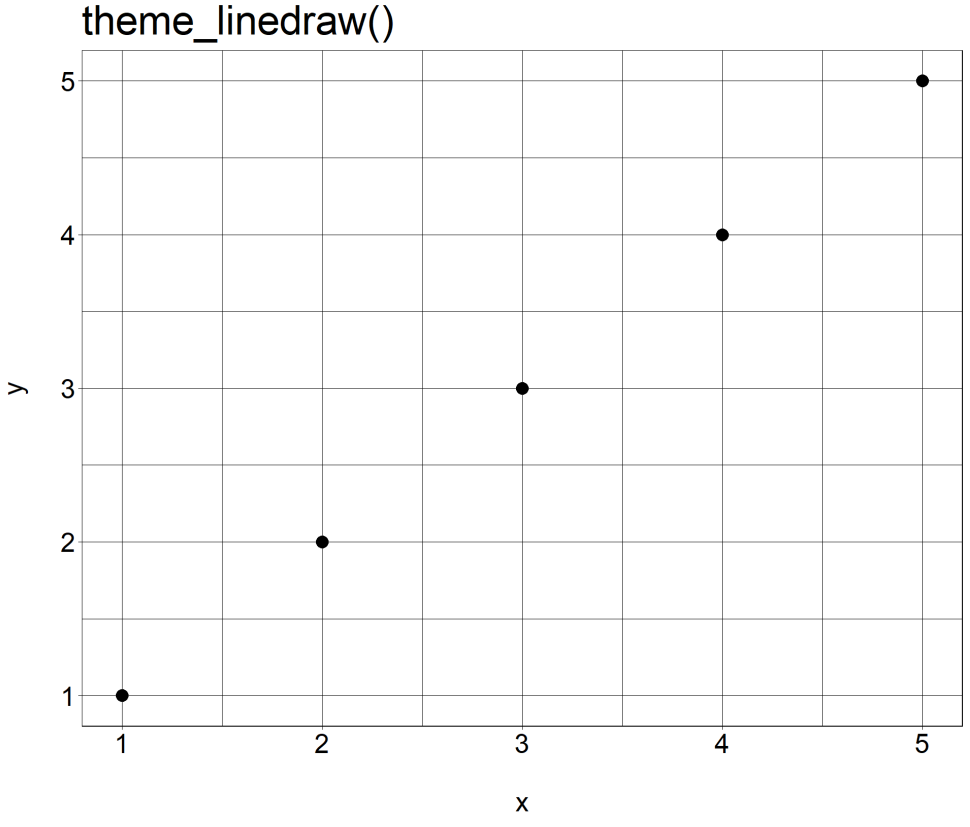
Examples:

- `element_text()`
- `element_rect()`
- `element_line()`
- etc.

Complete themes

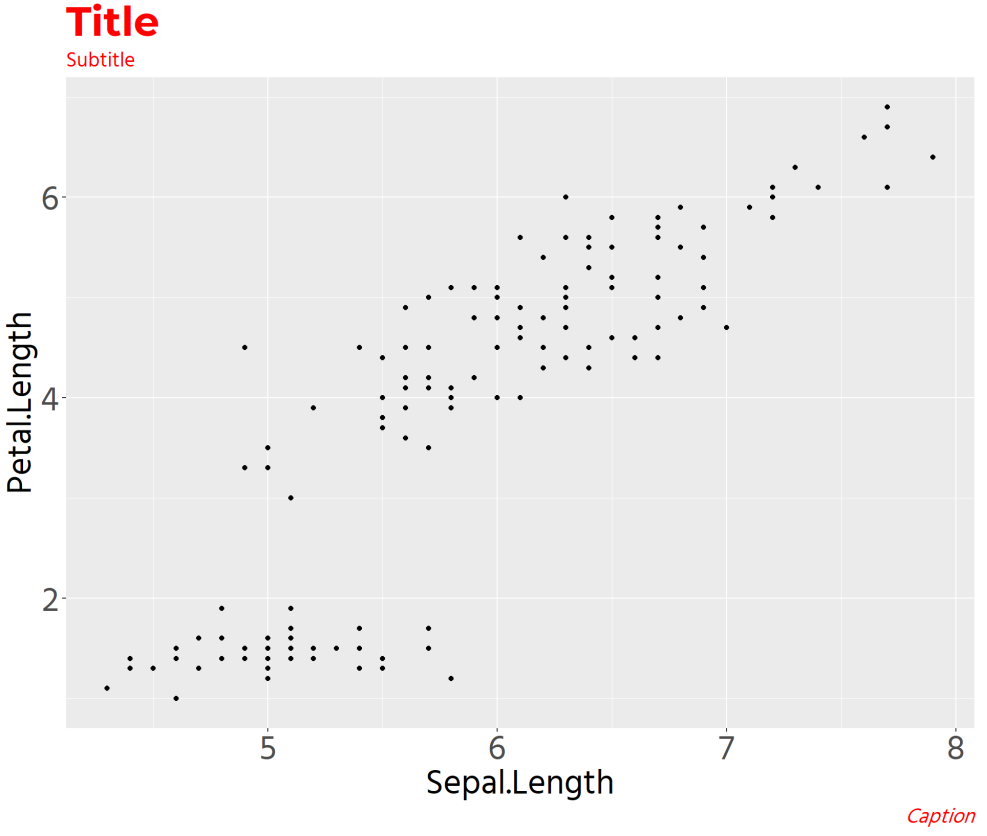
`ggplot2` has eight pre-made themes built in:

- `theme_grey()`
  - `theme_bw()`
- `theme_linedraw()`
  - `theme_light()`
  - `theme_dark()`
- `theme_minimal()`
- `theme_classic()`
- `theme_void()`



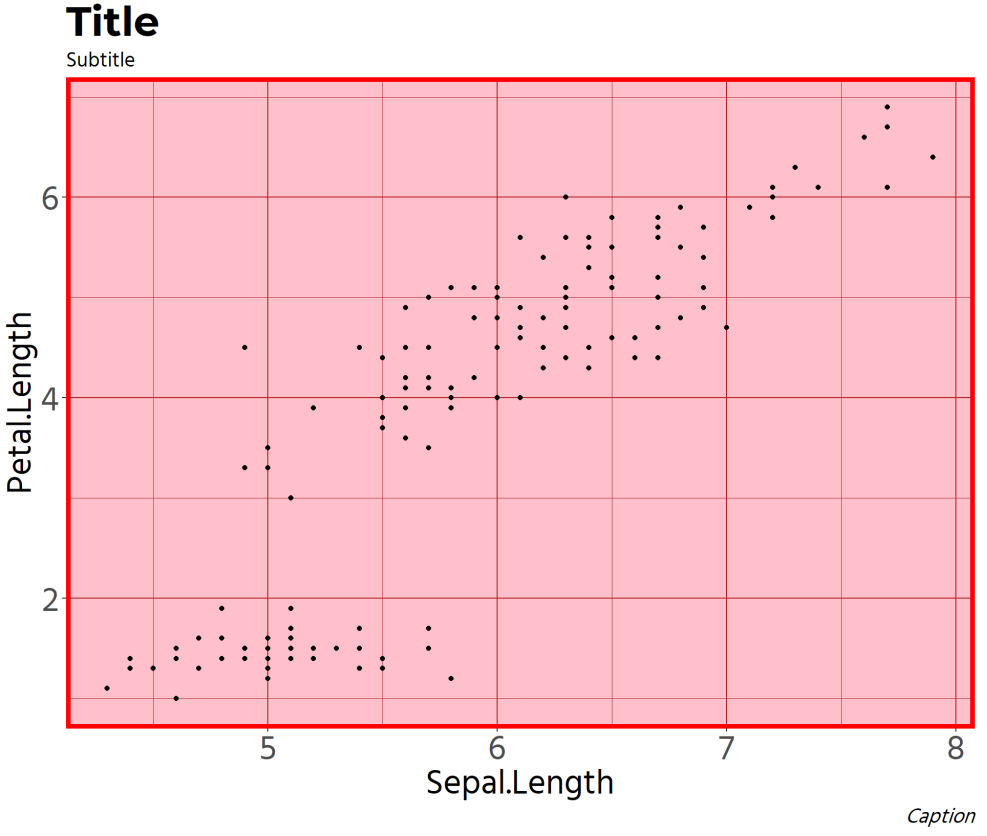
Plot elements

- `plot.background`
- `plot.title`
- `plot.title.position`
- `plot.subtitle`
- `plot.caption`
- `plot.caption.position`
- `plot.tag`
- `plot.tag.position`



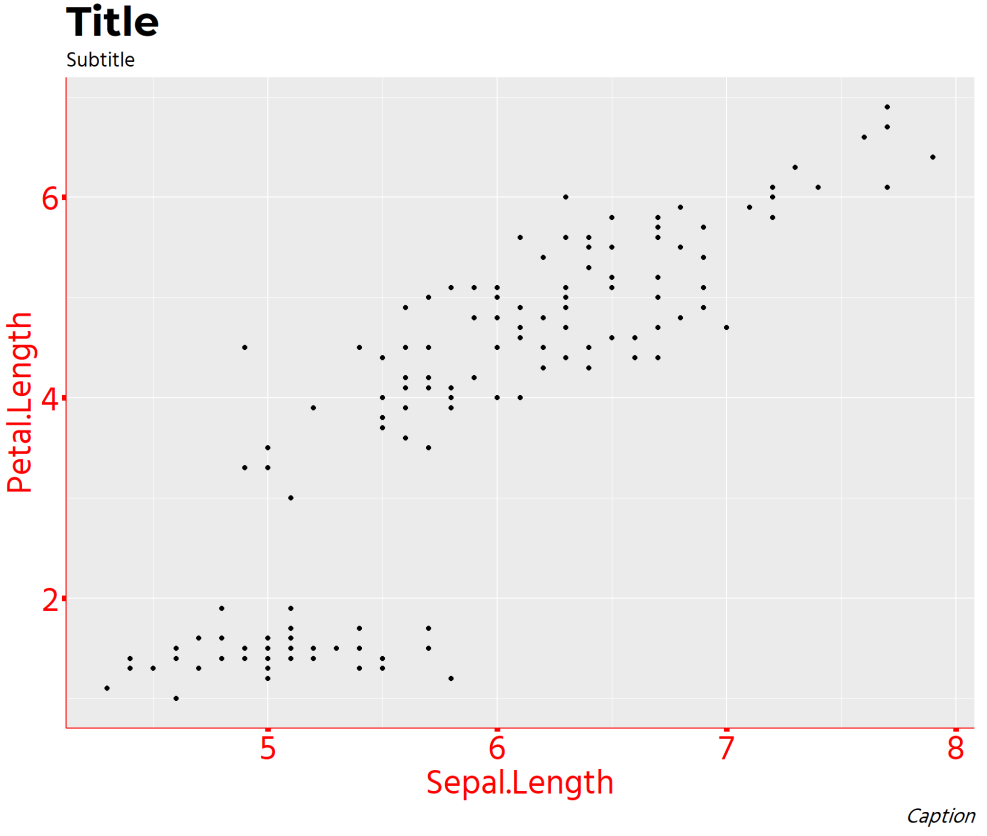
Panel elements

- panel.background
- panel.border
- panel.grid
  - panel.grid.major
    - panel.grid.major.x
    - panel.grid.major.y
  - panel.grid.minor
    - panel.grid.minor.x
    - panel.grid.minor.y



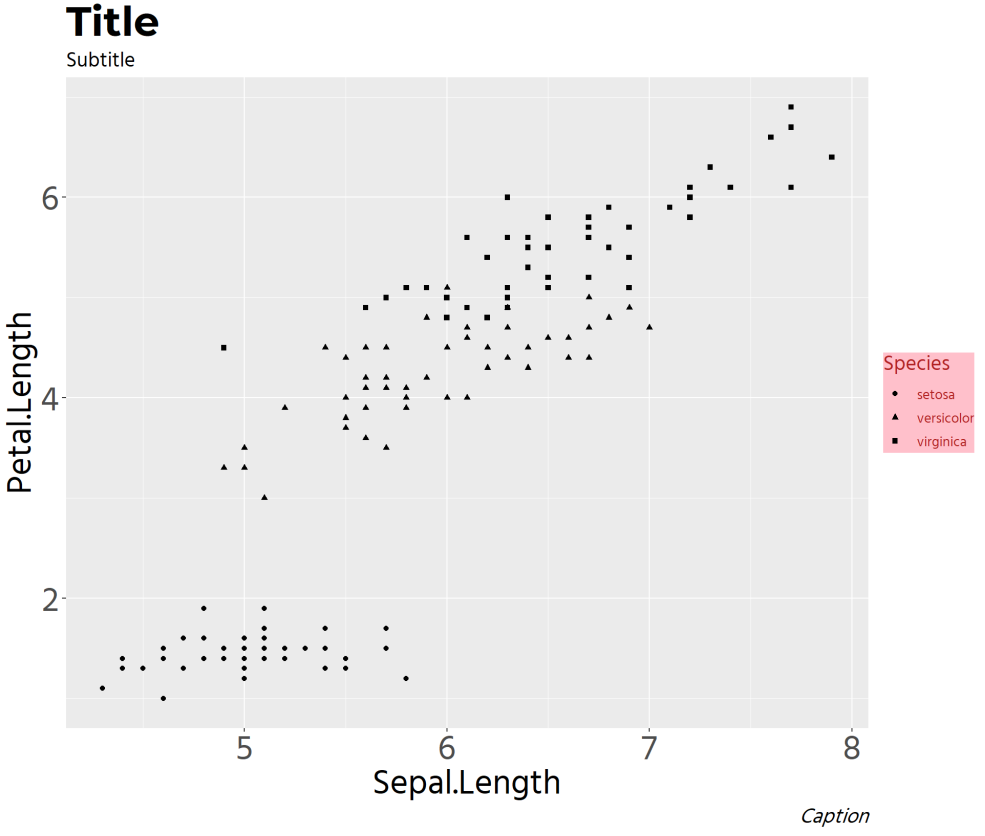
Axis elements

- `axis.title`
  - `axis.title.x` (.top/.bottom)
  - `axis.title.y` (.left/.right)
- `axis.text`
  - `axis.text.x` (.top/.bottom)
  - `axis.text.y` (.left/.right)`
- `axis.ticks`
  - `axis.ticks.x` (.top/.bottom)
  - `axis.ticks.y` (.left/.right)
- `axis.ticks.length`
  - `axis.ticks.length.x` (.top/.bottom)
  - `axis.ticks.length.y` (.left/.right)
- `axis.line`
  - `axis.line.x` (.top/.bottom)
  - `axis.line.y` (.left/.right)



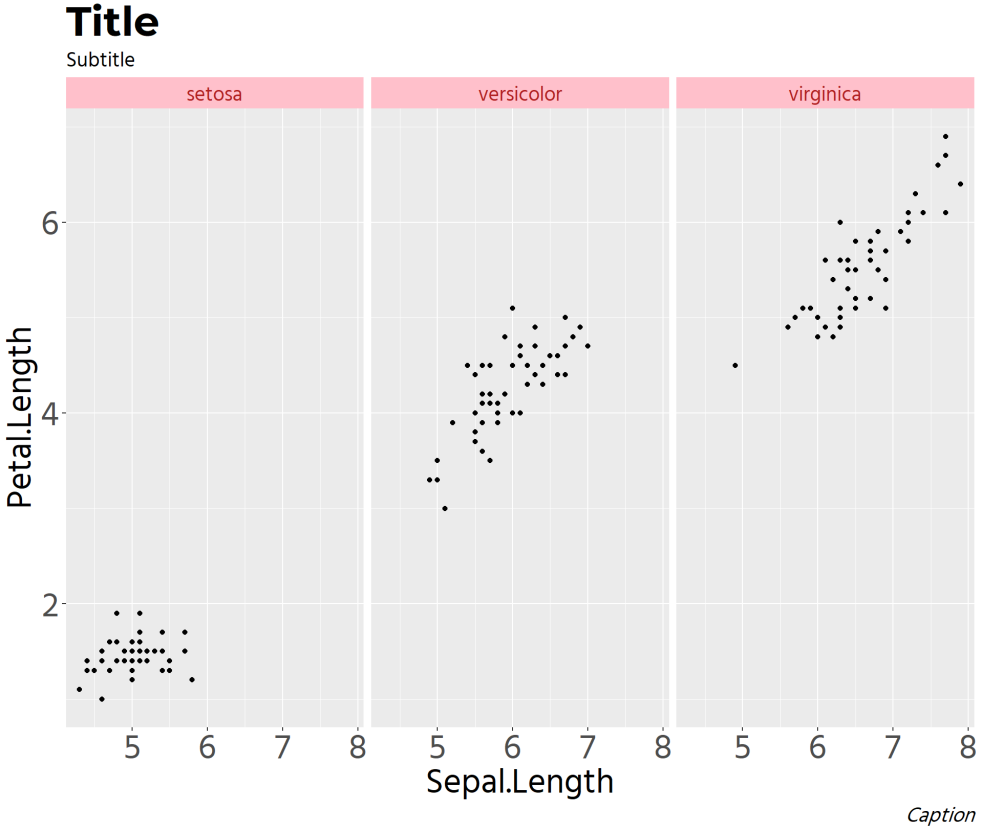
Legend elements

- `legend.background`
- `legend.margin`
- `legend.spacing`
  - `legend.spacing.x`
  - `legend.spacing.y`
- `legend.key`
  - `legend.key.size`
  - `legend.key.height`
  - `legend.key.width`
- `legend.text`
  - `legend.text.align`
- `legend.title`
  - `legend.title.align`
- `legend.position`
- `legend.direction`
- `legend.justification`
- `legend.box`
  - `legend.box.just`
  - `legend.box.margin`
  - `legend.box.background`
  - `legend.box.spacing`



Faceting elements

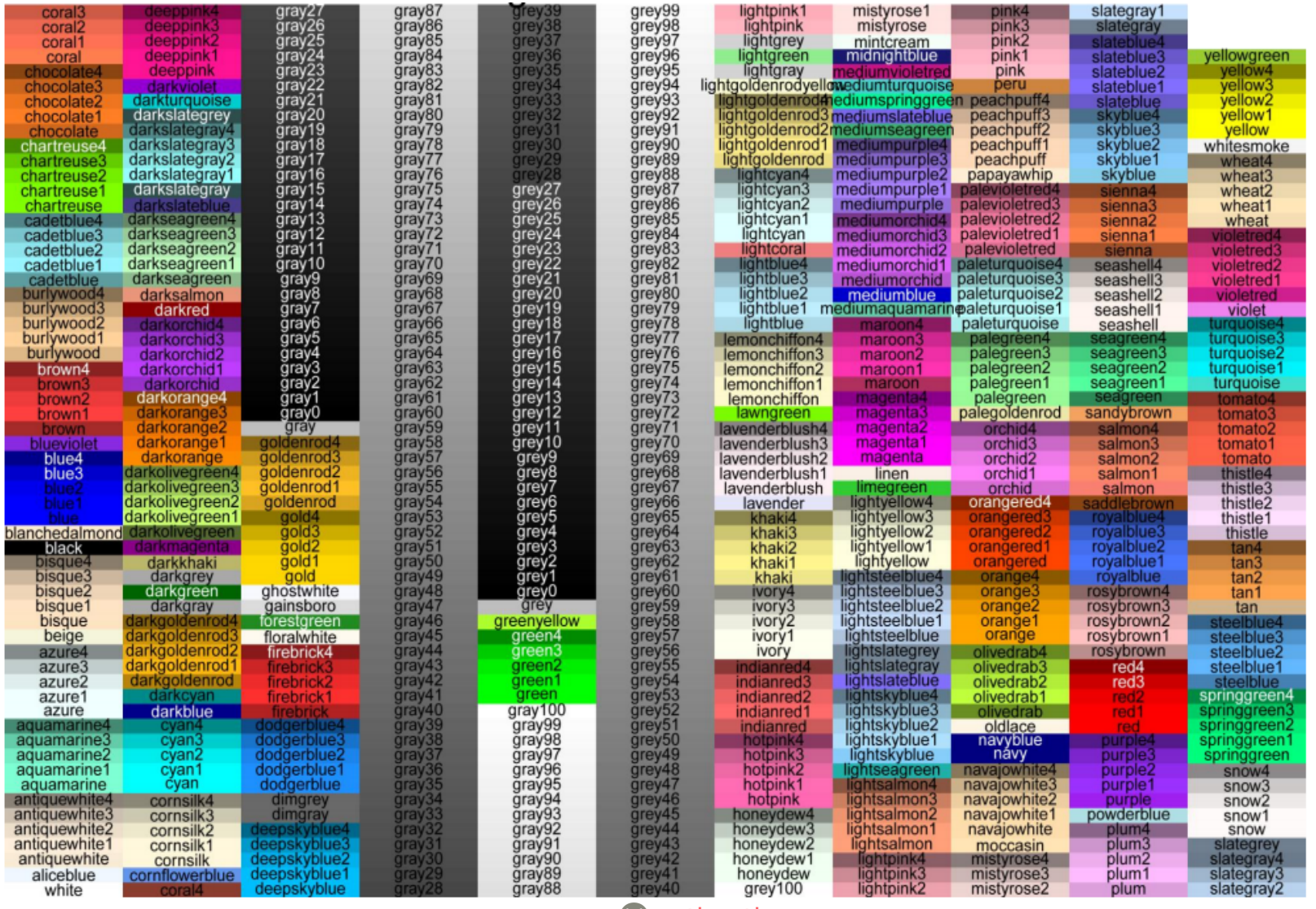
- strip.background
  - strip.background.x
  - strip.background.y
- strip.placement
- strip.text
  - strip.text.x
    - strip.text.x.bottom
    - strip.text.x.top
  - strip.text.y
    - strip.text.y.left
    - strip.text.y.right
- panel.spacing
  - panel.spacing.x
  - panel.spacing.y



Colors in R

Colors can be easily referenced in with a hex code, rbg code, or by name (if applicable).

Following colors are available by name:



nathantbarron

Fonts in R

You can use any font available on your computer in a `ggplot2` graphic.

For Windows users

```
install.packages('extrafont')
library('extrafont')

font_import()
# You will be prompted to continue [y/n]
# Type 'y' and press enter

loadfonts(device="win")
```

For Mac users

```
install.packages('extrafont')
library('extrafont')

font_import()
# You will be prompted to continue [y/n]
# Type 'y' and press enter

loadfonts()
```

Afterwards, you should be able to utilize fonts:

```
ggplot() +
  ... +
  theme(...
        ... = element_text(family = "font-name",
                           face = "type-face-name")
        )
```

Be sure to know which font families and font faces you have available.

Building a custom theme

Instead of re-using all of your `theme()` code (which can be lengthy), try saving the code as an object. Then, you can add the object to a ggplot.

```
# You have a pre-theme plot saved as 'existing_plot'
existing_plot <- ggplot() + ...

# Save your theme settings as an object ('my_theme')
my_theme <- theme(plot.title = element_text(...),
                  ...
                  ...
                  )

# Add your custom theme to the existing plot
existing_plot + my_theme
```

Exercise #2

Data Visualization with ggplot2

Session 3: Visualizing Diverse Data

Nathan Barron
nathanbarron@ou.edu

Different data need different visualizations

data | mpg | cyl | am | cyl x am

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4

Types of data

Levels of Data Measurement

- Categorical
  - Binary
  - Non-binary
- Continuous

. Cat.: Binary Cat.: Non-binary

Continuous

Click through each header to read more about each type

Storing Data in R

- Boolean
- Character
- Numerical
- Factor

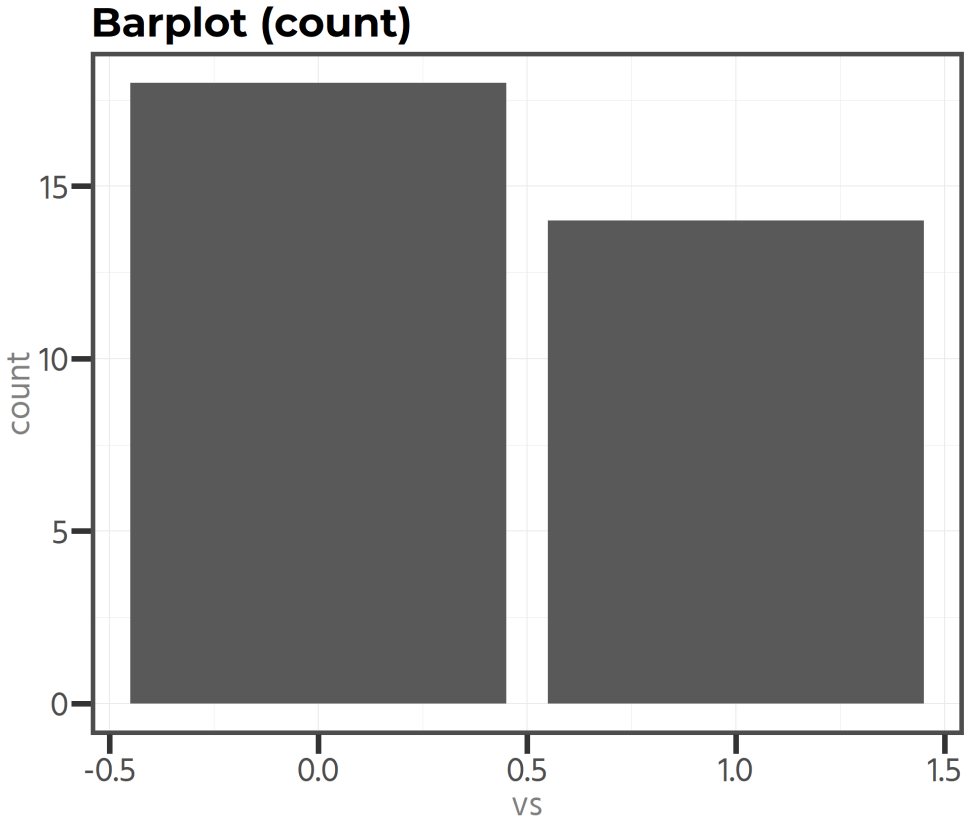
. Boolean Character Numerical

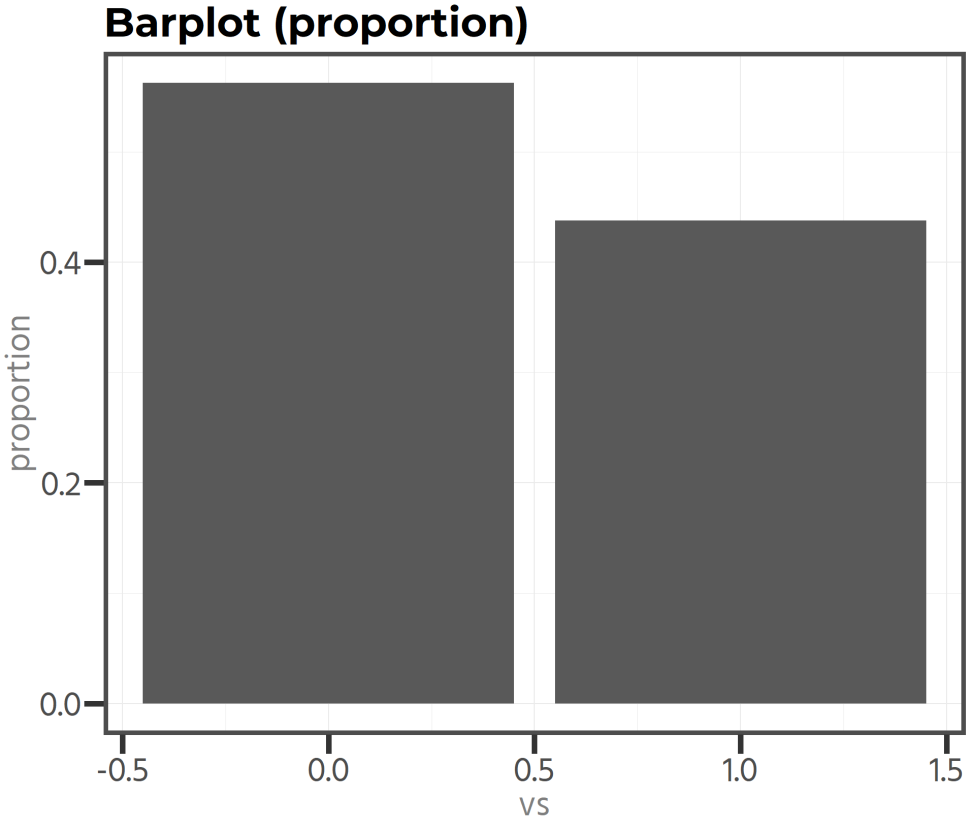
Factor Dates

Click through each header to read more about each type

Visualizing a single variable

Binary



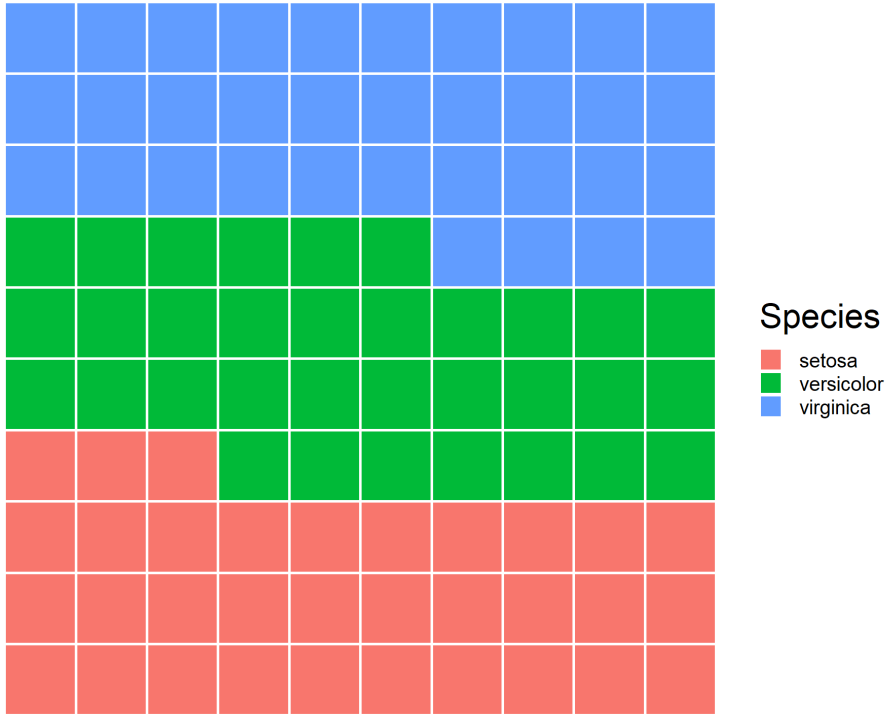



```r
# Barplot with frequency on y-axis
ggplot(mtcars) +
  geom_bar(aes(x=vs))
```

```r
# Barplot with proportion on y-axis
ggplot(mtcars) +
  geom_bar(aes(x=vs, y=..count../sum(..count..)))
```

Non-binary Categorical

Waffle plot

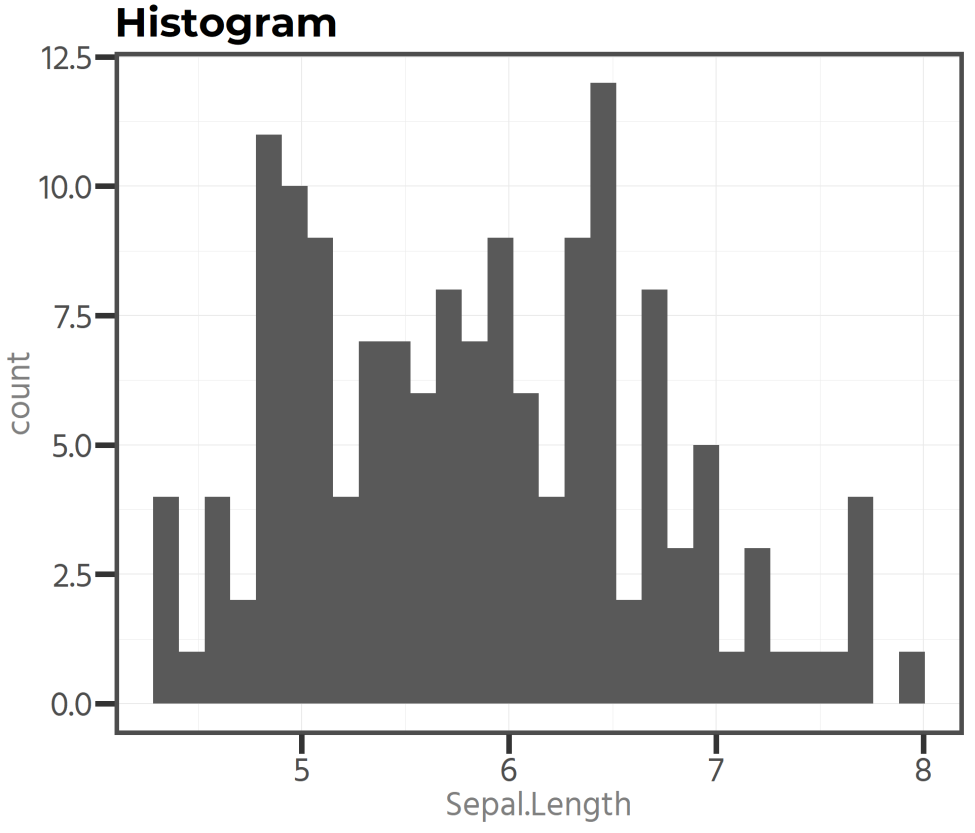


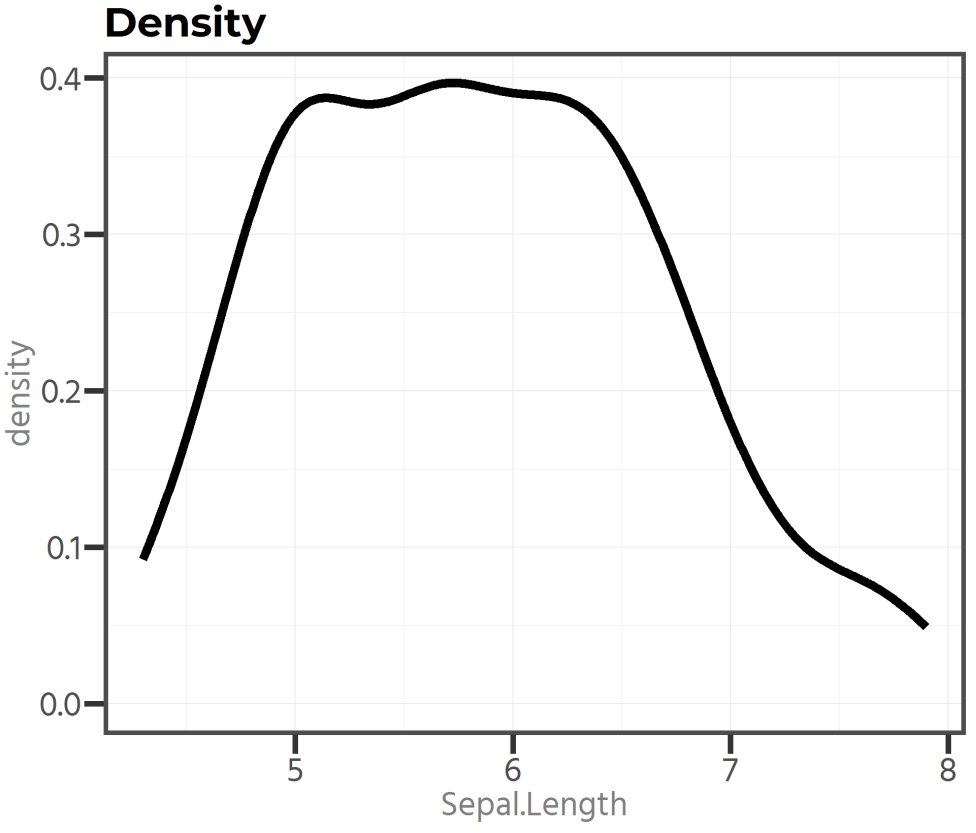

```r
# install.packages("waffle",
#                  repos = "https://cinc.rud.is")

library(waffle)
library(dplyr)

iris %>%
  count(Species) %>%
  ggplot(aes(fill = Species, values = n)) +
  geom_waffle(size = 1,
              colour = "white",
              na.rm=TRUE,
              flip = TRUE,
              make_proportional = TRUE) +
  theme_void() +
  coord_equal()
```

Continuous



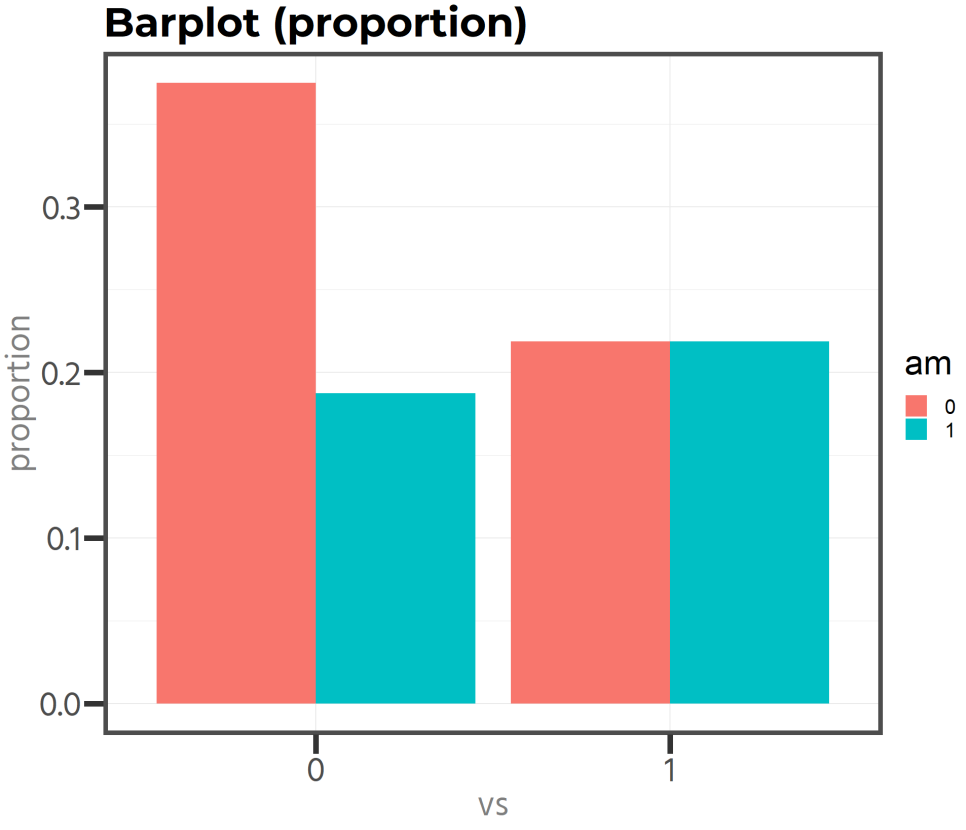



```
ggplot(iris) +
  geom_histogram(aes(x=Sepal.Length))
```

```
ggplot(iris) +
  geom_density(aes(x=Sepal.Length))
```

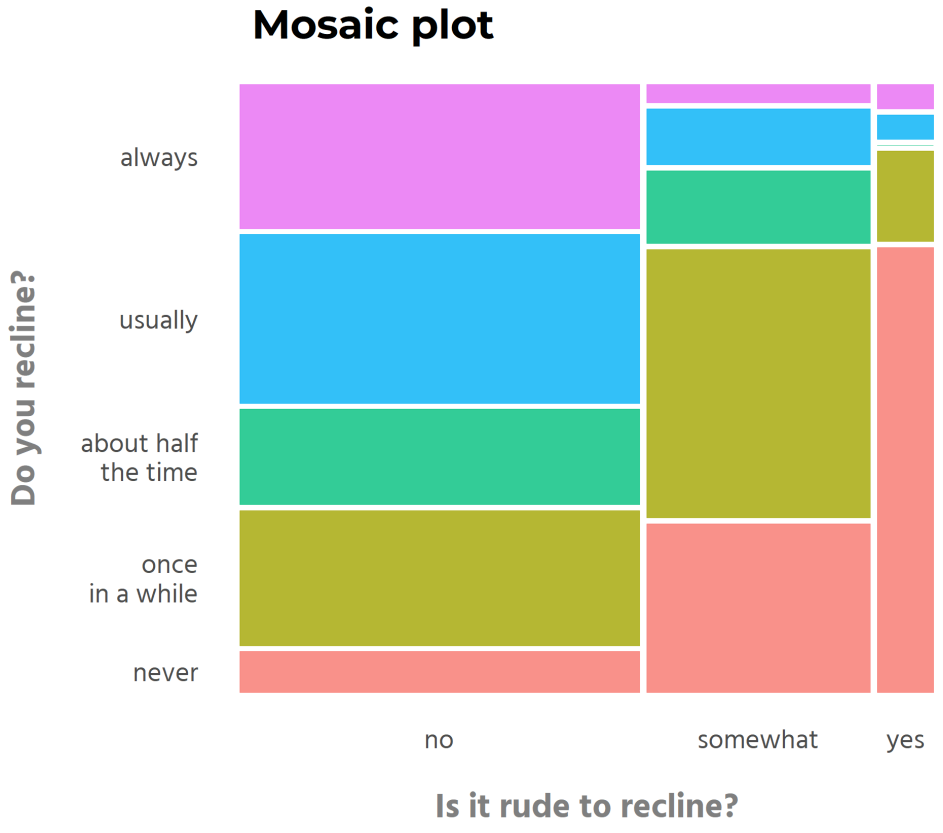
Visualizing multiple variables

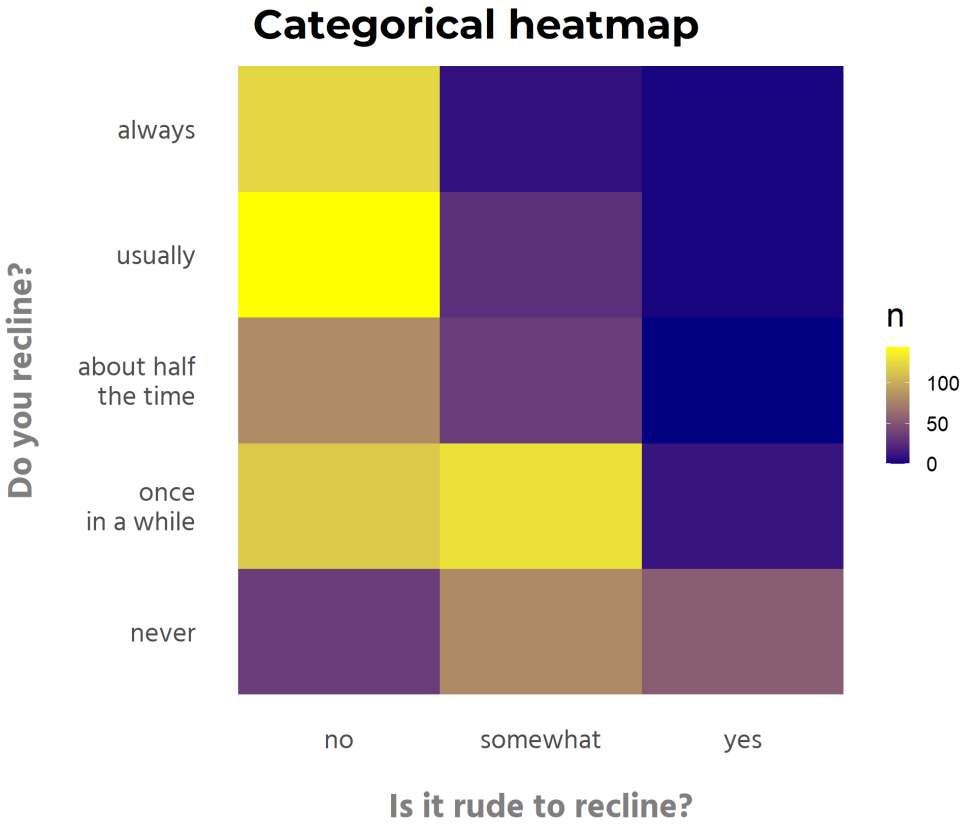
Multiple categorical variables




```
mtcars %>%
  mutate(vs = factor(vs), am = factor(am)) %>%
  ggplot(.) +
  geom_bar( aes(x=vs,
                y=..count../sum(..count..),
                group=am,
                fill=am),
            position = 'dodge')
```

Multiple categorical variables II



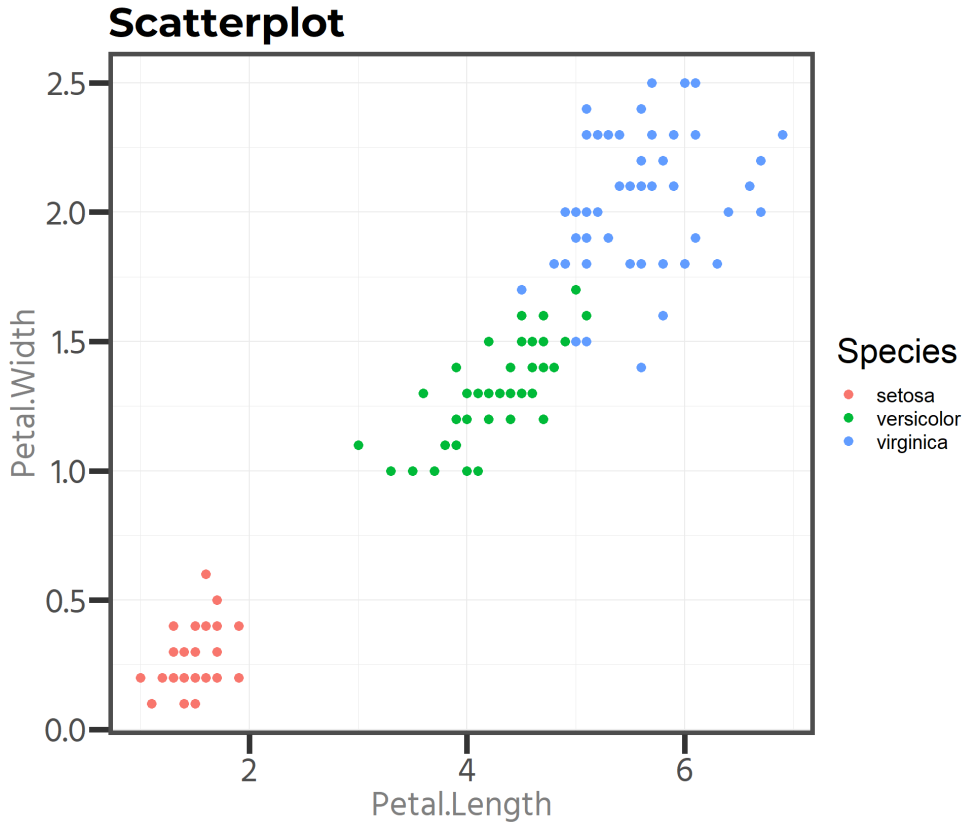


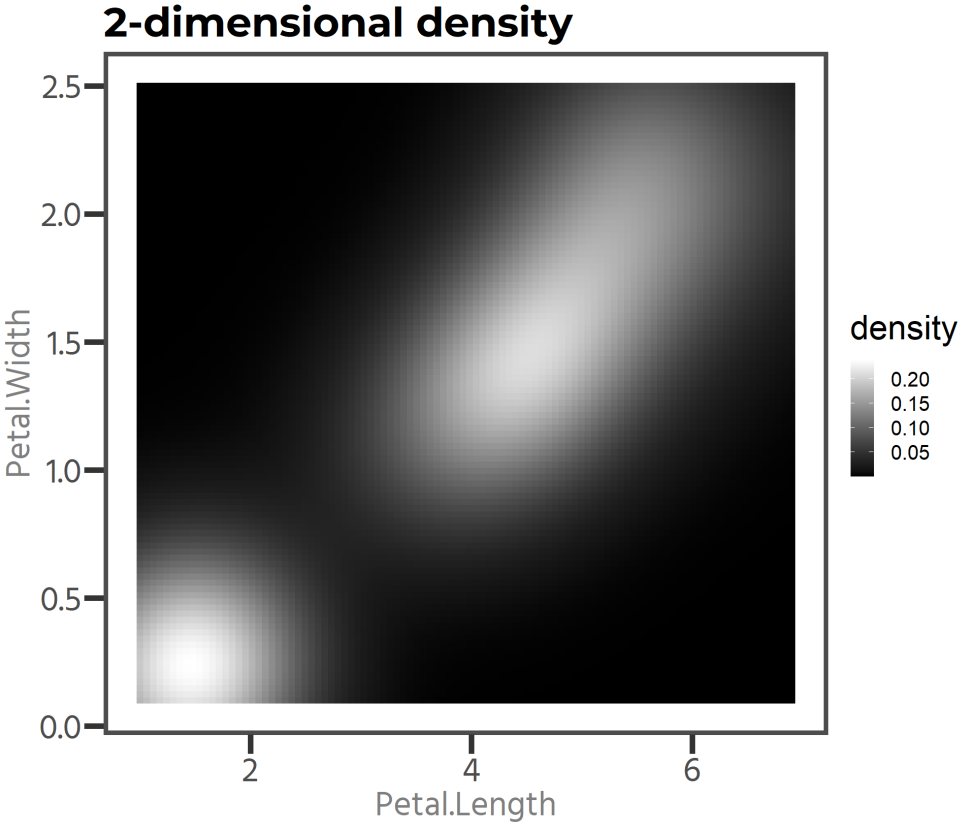

```
# library(ggmosaic)
# library(tidyr)

ggplot(ggmosaic::fly) +
  geom_mosaic(aes(x = product(do_you_recline, rude_to_rec
  theme(panel.grid = element_blank(),
        legend.position = 'none')
```

```
ggmosaic::fly %>%
  count(rude_to_recline, do_you_recline, .drop = FALSE) %
  ggplot(.) +
  geom_tile(aes(x = rude_to_recline,
                y = do_you_recline,
                fill = n)) +
  scale_fill_gradient(low = 'navy', high = "yellow")
```

Multiple continuous variables





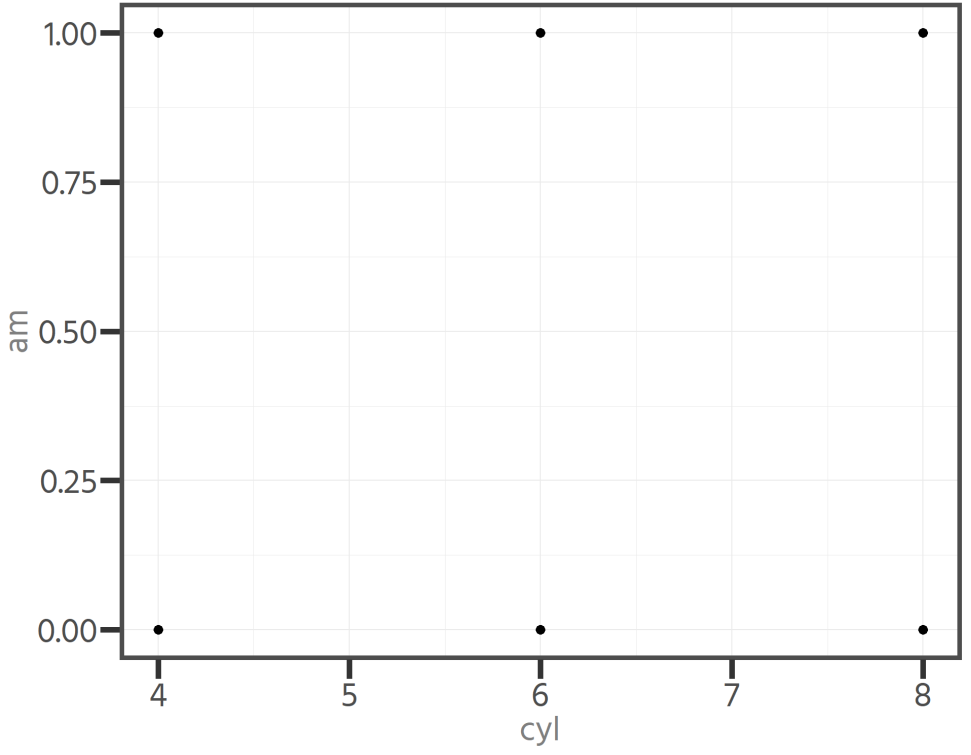

```
ggplot(iris) +
  geom_point(aes(x=Petal.Length,
                 y=Petal.Width,
                 color=Species))
```

```
ggplot(iris, aes(x=Petal.Length,y=Petal.Width)) +
  stat_density2d(aes(fill = ..density..),
                 geom = "raster", contour = FALSE) +
  scale_fill_gradient(low = "black", high = "white")
```

Hybrid (some categorical, some continuous)

Overplotting

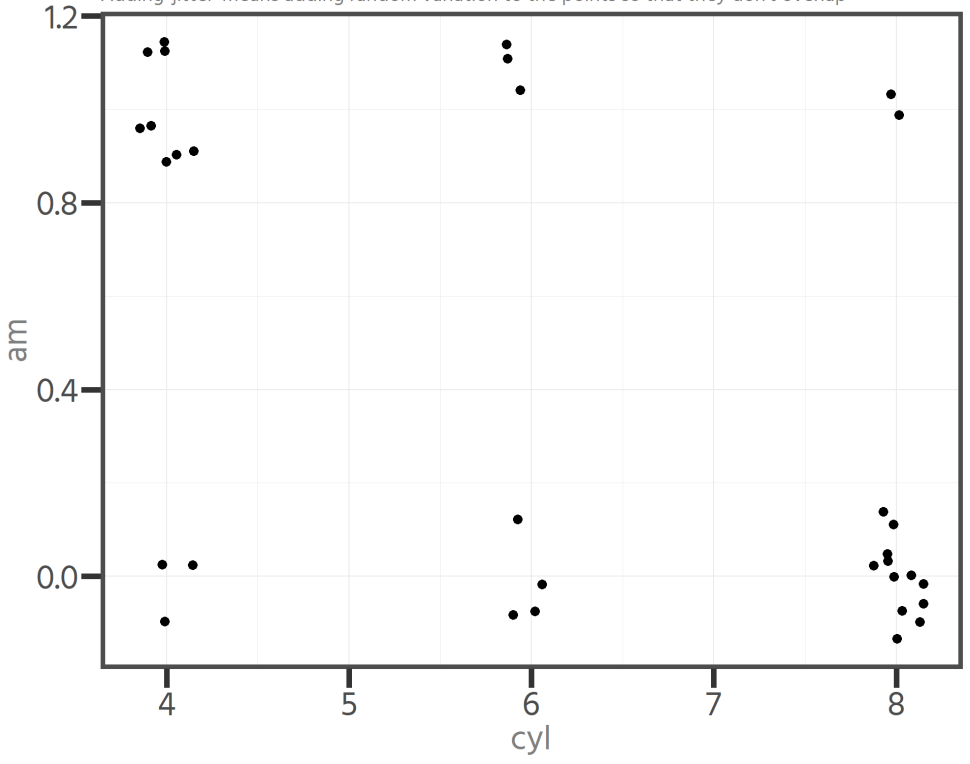
Overplotting is when data or labels overlap, meaning that the viewer loses important information




```
ggplot(mtcars) +
  geom_point(aes(x=cyl,y=am))
```

Jitterplot

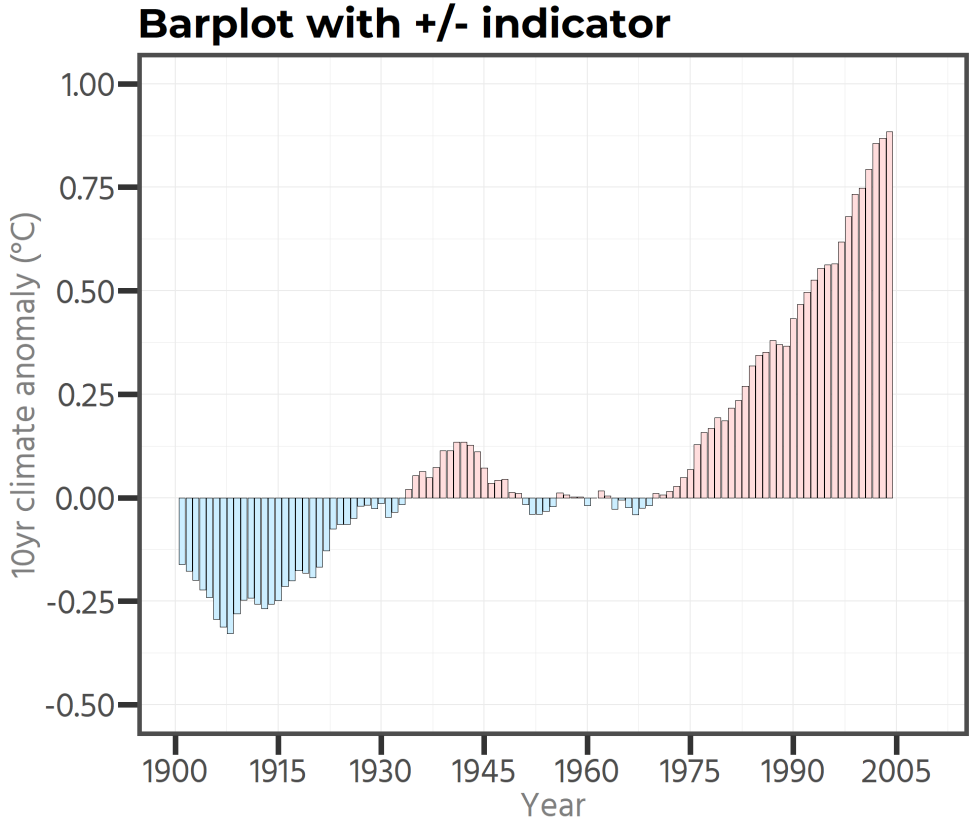
Adding 'jitter' means adding random variation to the points so that they don't overlap

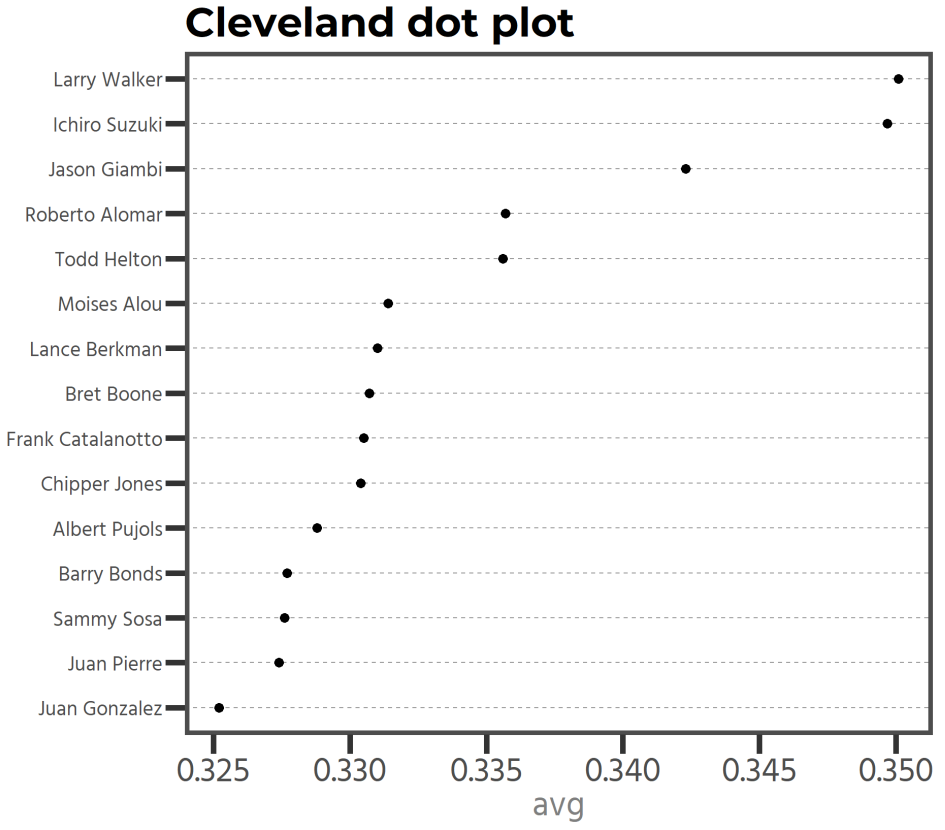



```
ggplot(mtcars) +
  geom_jitter(aes(x=cyl,y=am))

# You can also specify 'width' and 'height'
# as additional arguments to refine the
# jitter. Remember, jitter is random:
# jitter plots will change whenever you
# re-run the code.
```

Hybrid II






```r
library(gcookbook)
climate_sub <- climate %>%
  filter(Source == "Berkeley" & Year >= 1900) %>%
  mutate(pos = Anomaly10y >= 0)

ggplot(climate_sub, aes(x = Year, y = Anomaly10y, fill =
  geom_col(position = "identity", colour = "black", size
  scale_fill_manual(values = c("#CCEEFF", "#FFDDDD"), gui
```

```r
library(gcookbook)
tophit <- tophitters2001[1:15, ]

ggplot(tophit) +
  geom_point(aes(x = avg, y = reorder(name, avg)))

# The `reorder` function is helpful
# when making plots of a non-binary categorical
# variable and a corresponding continuous variable.
```

There is no “one size fits all”

Data viz is creative ...

and creativity requires inspiration.

Check out these cool resources:

- R graphics gallery
- Nightingale: The Magazine of the Data Visualization Society
- Daydreaming Numbers Blog

Data viz is also programming ...

and programming requires code.

Check out these cool resources:

- R Graphics Cookbook
- R Graph Gallery

Data Visualization with ggplot2

Session 4: Making Good Visualizations

Nathan Barron
nathanbarron@ou.edu

4 Principles of Good Data Visualization

Know your data

- You can't visualize what you don't understand
- A deep knowledge of the data, its structure, and the insights to be drawn are essential to data visualization
- When preparing your visualization, **form follows function**

Data for price of groceries since 1970

year	item	unit	price	type
1970	milk	gallon	0.49	namebrand
1970	eggs	dozen	0.60	namebrand
1970	butter	lb	0.87	namebrand
1970	flour	5lbs	0.59	namebrand
1970	sugar	5lbs	0.67	namebrand
1971	milk	gallon	0.50	namebrand
1971	eggs	dozen	0.53	namebrand
1971	butter	lb	0.88	namebrand
1971	flour	5lbs	0.60	namebrand
1971	sugar	5lbs	0.79	namebrand

Sketch

The creative and `> programming` aspects of data visualization require different parts of your brain.

1. Start with *What do I want?*
2. Draw it
3. Follow up with *How do I get that?*
4. Code it!

Visualize the price of groceries since 1970

Simple Visuals

Don't try to include too much or irrelevant information.

Do provide detailed captions that offer additional context and information to the graphic

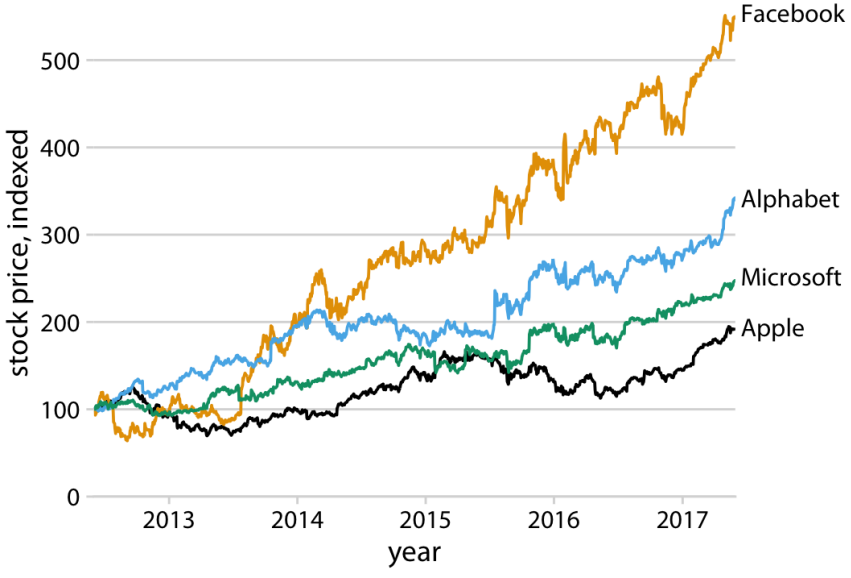
Use Color Effectively

- Only use colors when:
  - There are fewer than 6 categories; or
  - Visualizing continuous data with color gradient
- Use monotonic scales
- Use accessible colors: Color Blindness Simulator
- Use colors and shading to highlight important information

Color Palette Generator

Legends

- If there is a clear visual ordering in your data, make sure to match it in the legend.
- Whenever possible, design your figures so they don't need a legend.



Tell a story well

Don’t try to tell too much in a single visualization

Do know *what* you want to convey

Do design for the ‘aha’ moment

Code-along #1

```
library(ggplot2)

# 1. Plotting layers ----
## 1.1 Base layer ----
ggplot(mtcars)


## 1.2 Adding geom and scale layers ----
ggplot(mtcars) +
  geom_point(aes(x=wt, y=mpg))


## 1.3 Adding title annotations ----
ggplot(mtcars) +
  geom_point(aes(x=wt, y=mpg)) +
  labs(x='Car weight (1000 lbs)',
       y='Miles per (US) gallon',
       title = 'Is car weight related to fuel efficiency?',
       subtitle = 'Car weight is measured in 1000 lbs. Fuel efficiency is measured in
miles per (US) gallon.',
       caption = 'Data come from the mtcars dataset.')


## 1.4 Adjusting the theme layer ----
ggplot(mtcars) +
  geom_point(aes(x=wt, y=mpg)) +
  labs(x='Car weight (1000 lbs)',
       y='Miles per (US) gallon',
       title = 'Is car weight related to fuel efficiency?',
       subtitle = 'Car weight is measured in 1000 lbs. Fuel efficiency is measured in
miles per (US) gallon.',
       caption = 'Data come from the mtcars dataset.') +
  theme(axis.title = element_text(color = 'grey50'),
        plot.title = element_text(face = 'bold'),
        plot.subtitle = element_text(color = 'grey50'),
        plot.caption = element_text(face = 'italic'))


## 1.5 Adding another geom layer (line of best fit) ----
ggplot(mtcars) +
  geom_point(aes(x=wt, y=mpg)) +
  labs(x='Car weight (1000 lbs)',
       y='Miles per (US) gallon',
       title = 'Is car weight related to fuel efficiency?',
       subtitle = 'Car weight is measured in 1000 lbs. Fuel efficiency is measured in
miles per (US) gallon.',
       caption = 'Data come from the mtcars dataset.') +
  theme(axis.title = element_text(color = 'grey50'),
        plot.title = element_text(face = 'bold'),
        plot.subtitle = element_text(color = 'grey50'),
        plot.caption = element_text(face = 'italic')) +
  geom_smooth(aes(x=wt, y=mpg),
              method = 'lm')


## 1.6 Adding another geom layer (text annotation) ----
ggplot(mtcars) +
  geom_point(aes(x=wt, y=mpg)) +
  labs(x='Car weight (1000 lbs)',
       y='Miles per (US) gallon',
       title = 'Is car weight related to fuel efficiency?',
       subtitle = 'Car weight is measured in 1000 lbs. Fuel efficiency is measured in
miles per (US) gallon.',
       caption = 'Data come from the mtcars dataset.') +
```

```
  theme(axis.title = element_text(color = 'grey50'),
        plot.title = element_text(face = 'bold'),
        plot.subtitle = element_text(color = 'grey50'),
        plot.caption = element_text(face = 'italic')) +
  geom_smooth(aes(x=wt, y=mpg),
              method = 'lm') +
  annotate("text", x=4.5, y=25, label = 'NOTE')


# 2. Aesthetics ----
## 2.1 Example of Global Aesthetics ----
ggplot(mpg) +
  geom_point(aes(displ, hwy),
             color = 'red')


## 2.2 Example of Mapped Aesthetics ----
ggplot(mpg) +
  geom_point(aes(displ, hwy, color = class))


# 3. Linetypes and shapes ----
## 3.1 Linetype as global aesthetic ----
ggplot(mtcars) +geom_smooth(aes(x=wt, y=mpg), method = 'lm', linetype = 2)


## 3.2 Linetype as mapped aesthetic ----
ggplot(mtcars) +geom_smooth(aes(x=wt, y=mpg, linetype=as.factor(cyl)), method = 'lm')


## 3.3 Shape as global aesthetic ----
ggplot(mtcars) +geom_point(aes(x=wt, y=mpg), shape = 2)


## 3.4 Shape as mapped aesthetic ----
ggplot(mtcars) +geom_point(aes(x=wt, y=mpg, shape = as.factor(cyl)))


# 4. Scales ----
# 4.1 Adjusting axis scales (continuous) ----
ggplot(iris) +
  geom_point(aes(Petal.Length,Petal.Width, color=Species)) +
  scale_x_continuous(limits=c(0,7), breaks = seq(1,7,1)) +
  scale_y_continuous(limits=c(0,7), breaks = seq(0,7,1))


# 4.2 Adjusting axis scales (discrete) ----
ggplot(iris) +
  geom_boxplot(aes(Species,Petal.Length)) +
  scale_x_discrete(limits = c('virginica','versicolor','setosa'),
                   labels = c('VIRGINICA','VERSICOLOR','SETOSA'))

# Caution: In many cases, the ordering of discrete values on the X axis
#          is better manipulated by factoring the variable while cleaning
#          prior to plotting.


# 4.3 Adjusting color scales (continuous) ----
ggplot(iris) +
  geom_point(aes(Petal.Length,Petal.Width, color=Species))


### 4.3a Low/High Gradient ----
ggplot(iris) +
  geom_point(aes(Petal.Length,Petal.Width, color=Petal.Width)) +
```

```
  scale_color_gradient(low = 'grey40', high = 'orange')


### 4.3b Low/High Steps ----
ggplot(iris) +
  geom_point(aes(Petal.Length,Petal.Width, color=Petal.Width)) +
  scale_color_steps(low = 'grey40', high = 'orange', n.breaks = 3)


### 4.3c Color Brewer Palettes ----
ggplot(iris) +
  geom_point(aes(Petal.Length,Petal.Width, color=Petal.Width)) +
  scale_color_fermenter(palette = 'Set2')

# Use scale_color_brewer() for gradient.
# Use scale_color_fermenter() for steps.
# See more palettes: https://r-graph-gallery.com/38-rcolorbrewers-palettes.html


### 4.3d Viridis Palettes ----
ggplot(iris) +
  geom_point(aes(Petal.Length,Petal.Width, color=Petal.Width)) +
  scale_color_viridis_c()

# Use scale_color_viridis_c() for gradient.
# Use scale_color_viridis_b() for steps.
# See more palettes: https://cran.r-project.org/web/packages/viridis/vignettes/intro-to-
viridis.html


# 4.4 Adjusting color scales (discrete) ----
### 4.4a Manually ----
ggplot(iris) +
  geom_point(aes(Petal.Length,Petal.Width, color=Species)) +
  scale_color_manual(values = c('grey40','firebrick','orange'))

### 4.4b Greyscale ----
ggplot(iris) +
  geom_point(aes(Petal.Length,Petal.Width, color=Species)) +
  scale_color_grey()


### 4.4c Color Brewer Palettes ----
ggplot(iris) +
  geom_point(aes(Petal.Length,Petal.Width, color=Species)) +
  scale_color_brewer(palette = 'Set2')

# See more palettes: https://r-graph-gallery.com/38-rcolorbrewers-palettes.html


### 4.4d Viridis Palettes ----
ggplot(iris) +
  geom_point(aes(Petal.Length,Petal.Width, color=Species)) +
  scale_color_viridis_d(option = 'turbo')

# See more palettes: https://cran.r-project.org/web/packages/viridis/vignettes/intro-to-
viridis.html


# 5. Faceting ----
## 5.1 Example of facet_wrap ----
ggplot(mtcars) +
  geom_point(aes(hp, mpg)) +
  facet_wrap(~as.factor(cyl))
```

```
## 5.2 Example of facet_grid ----
ggplot(mtcars) +
  geom_point(aes(hp, mpg)) +
  facet_grid(as.factor(cyl) ~ as.factor(gear))
```

Code-along #2

```
library(ggplot2)

# 1. Complete themes ----
## 1.1 theme_grey() ----
ggplot(mtcars) + geom_point(aes(wt,mpg)) + theme_grey()


## 1.2 theme_bw() ----
ggplot(mtcars) + geom_point(aes(wt,mpg)) + theme_bw()


## 1.3 theme_linedraw() ----
ggplot(mtcars) + geom_point(aes(wt,mpg)) + theme_linedraw()


## 1.4 theme_light() ----
ggplot(mtcars) + geom_point(aes(wt,mpg)) + theme_light()


## 1.5 theme_dark() ----
ggplot(mtcars) + geom_point(aes(wt,mpg)) + theme_dark()


## 1.6 theme_minimal() ----
ggplot(mtcars) + geom_point(aes(wt,mpg)) + theme_minimal()


## 1.7 theme_classic() ----
ggplot(mtcars) + geom_point(aes(wt,mpg)) + theme_classic()


## 1.7 theme_void() ----
ggplot(mtcars) + geom_point(aes(wt,mpg)) + theme_void()


# 2. Plot elements ----
## 2.1 plot.background ----
ggplot(mtcars) + geom_point(aes(wt,mpg)) +
  labs(title = 'Title', subtitle = 'Subtitle') +
  theme(plot.background = element_rect(fill='pink'))

## 2.2 plot.title and plot.subtitle ----
ggplot(mtcars) + geom_point(aes(wt,mpg)) +
  labs(title = 'Title', subtitle = 'Subtitle',
       caption = 'Caption') +
  theme(plot.title = element_text(color='red', size = 25),
        plot.subtitle = element_text(color='grey40', face = 'italic'),
        plot.caption = element_text(color='grey40'))


# 3. Panel elements ----
## 3.1 panel.background ----
ggplot(mtcars) + geom_point(aes(wt,mpg)) +
  labs(title = 'Title', subtitle = 'Subtitle',
       caption = 'Caption') +
  theme(panel.background = element_rect(fill = 'pink'))


## 3.2 panel.border ----
ggplot(mtcars) + geom_point(aes(wt,mpg)) +
  labs(title = 'Title', subtitle = 'Subtitle',
       caption = 'Caption') +
  theme(panel.border = element_rect(color = 'red',fill=NA))
```

```
## 3.3 panel.grid ----
ggplot(mtcars) + geom_point(aes(wt,mpg)) +
  labs(title = 'Title', subtitle = 'Subtitle',
       caption = 'Caption') +
  theme(panel.grid = element_line(color = 'red'))


# 4. Axis elements ----
## 4.1 axis.title ----
ggplot(mtcars) + geom_point(aes(wt,mpg)) +
  labs(title = 'Title', subtitle = 'Subtitle',
       caption = 'Caption') +
  theme(axis.title = element_text(color = 'red'))

ggplot(mtcars) + geom_point(aes(wt,mpg)) +
  labs(title = 'Title', subtitle = 'Subtitle',
       caption = 'Caption') +
  theme(axis.title.x = element_text(color = 'forestgreen'),
        axis.title.y = element_text(color = 'blue'))


## 4.2 axis.text ----
ggplot(mtcars) + geom_point(aes(wt,mpg)) +
  labs(title = 'Title', subtitle = 'Subtitle',
       caption = 'Caption') +
  theme(axis.text = element_text(size=25, angle = 40),
        axis.text.y = element_text(color = 'forestgreen'))

## 4.3 axis.ticks and axis.ticks.length ----
ggplot(mtcars) + geom_point(aes(wt,mpg)) +
  labs(title = 'Title', subtitle = 'Subtitle',
       caption = 'Caption') +
  theme(axis.ticks = element_line(linewidth=8),
        axis.ticks.length = unit(18, 'points'))


# 5. Fonts ----
## 5.1 For Windows users ----
install.packages('extrafont')
library('extrafont')

font_import()
# You will be prompted to continue [y/n]
# Type 'y' and press enter

loadfonts(device="win")


## 5.2 For Mac users ----
install.packages('extrafont')
library('extrafont')

font_import()
# You will be prompted to continue [y/n]
# Type 'y' and press enter

loadfonts()


# 5.3 Check which fonts are available ----
fonts()


# 6. Custom theme ----
```

```
# You have a pre-theme plot saved as 'existing_plot'
existing_plot <- ggplot(mtcars) + geom_point(aes(wt,mpg)) + labs(title =
'TITLE',subtitle='Subtitle')

# Save your theme settings as an object ('my_theme')
my_theme <- theme(plot.title = element_text(size=25, family = 'Urbanist', face = 'bold',
color = 'blue'),
                  plot.subtitle = element_text(size=15, family = 'Urbanist', face =
'italic', color = 'grey40'),
                  axis.title = element_text(size = 10, family = 'Urbanist'),
                  axis.text = element_text(size = 8, family = 'Urbanist'),
)

# Add your custom theme to the existing plot
existing_plot + my_theme
```

Code-along #3

```
library(ggplot2)

# 1. Visualizing a single variable ----
## 1.1 Binary varialbes ----
### 1.1a Barplot (count) ----
ggplot(mtcars) + geom_bar(aes(x=vs))


### 1.1b Barplot (proportion) ----
ggplot(mtcars) + geom_bar(aes(x=vs, y=..count../sum(..count..)))


## 1.2 Non-binary categorical ----
### 1.2a Waffle plot ----
# install.packages("waffle", repos = "https://cinc.rud.is")
library(waffle)
library(dplyr)

iris %>%
  count(Species) %>%
  ggplot(aes(fill = Species, values = n)) +
  geom_waffle(size = 1,
              colour = "white",
              na.rm=TRUE,
              flip = TRUE,
              make_proportional = TRUE) +
  theme_void() +
  coord_equal()


## 1.3 Continuous ----
### 1.3a Histogram ----
ggplot(iris) + geom_histogram(aes(x=Sepal.Length))


### 1.3b Density ----
ggplot(iris) + geom_density(aes(x=Sepal.Length))


# 2. Visualizing multiple variables ----
## 2.1 Multiple categorical variables ----
### 2.1a Barplot (proportion) ----
mtcars %>%
  mutate(vs = factor(vs), am = factor(am)) %>%
  ggplot(.) +
  geom_bar( aes(x=vs,
                y=..count../sum(..count..),
                group=am,
                fill=am),
            position = 'dodge')


### 2.1b Mosaic plot ----
# library(ggmosaic)
# library(tidyr)

ggplot(ggmosaic::fly) +
  geom_mosaic(aes(x = product(do_you_recline, rude_to_recline), fill=do_you_recline)) +
  theme(panel.grid = element_blank(),
        legend.position = 'none')


### 2.1c Categorical heatmap ----
ggmosaic::fly %>%
```

```
  count(rude_to_recline, do_you_recline, .drop = FALSE) %>%
  ggplot(.) +
  geom_tile(aes(x = rude_to_recline,
                y = do_you_recline,
                fill = n)) +
  scale_fill_gradient(low = 'navy', high = "yellow")


## 2.2 Multiple continuous variables ----
### 2.2a Scatterplot ----
ggplot(iris) +
  geom_point(aes(x=Petal.Length,
                 y=Petal.Width,
                 color=Species))


### 2.2b Two-dimensional density ----
ggplot(iris, aes(x=Petal.Length,y=Petal.Width)) +
  stat_density2d(aes(fill = ..density..),
                 geom = "raster", contour = FALSE) +
  scale_fill_gradient(low = "black", high = "white")


## 2.3 Hybrid ----
### 2.3a Jitterplot ----
ggplot(mtcars) +
  geom_jitter(aes(x=cyl,y=am))

# You can also specify 'width' and 'height'
# as additional arguments to refine the
# jitter. Remember, jitter is random:
# jitter plots will change whenever you
# re-run the code.


### 2.3b Barplot with +/- indicator ----
library(gcookbook)
climate_sub <- climate %>%
  filter(Source == "Berkeley" & Year >= 1900) %>%
  mutate(pos = Anomaly10y >= 0)

ggplot(climate_sub, aes(x = Year, y = Anomaly10y, fill = pos)) +
  geom_col(position = "identity", colour = "black", size = 0.25) +
  scale_fill_manual(values = c("#CCEEFF", "#FFDDDD"), guide = 'none')


### 2.3c Cleveland dot plot ----
library(gcookbook)
tophit <- tophitters2001[1:15, ]

ggplot(tophit) +
  geom_point(aes(x = avg, y = reorder(name, avg)))

# The `reorder` function is helpful
# when making plots of a non-binary categorical
# variable and a corresponding continuous variable.
```

Data Visualization with ggplot2 in R

Activity #1: Creating Base Plots

0.0 Getting Started

Install and load the **palmerpenguins** package. You should notice that you now have access to the `penguins` dataset, even though it does not appear in the global environment.

```
install.packages("palmerpenguins")
```

```
library(ggplot2)
library(palmerpenguins)
head(penguins)
```

```
# A tibble: 6 × 8
  species island    bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
  <fct>   <fct>              <dbl>         <dbl>             <int>       <int>
1 Adelie  Torgersen           39.1          18.7               181        3750
2 Adelie  Torgersen           39.5          17.4               186        3800
3 Adelie  Torgersen           40.3          18                 195        3250
4 Adelie  Torgersen           NA            NA                  NA          NA
5 Adelie  Torgersen           36.7          19.3               193        3450
6 Adelie  Torgersen           39.3          20.6               190        3650
# ℹ 2 more variables: sex <fct>, year <int>
```

1.0 Creating a Base Plot

Using the `penguins` dataset, create a scatterplot (using `geom_point()`) of flipper_length_mm (Y axis) by body_mass_g (X axis). Your plot should include *each* of the following:

- Title
- Subtitle
- X-axis title
- Y-axis title

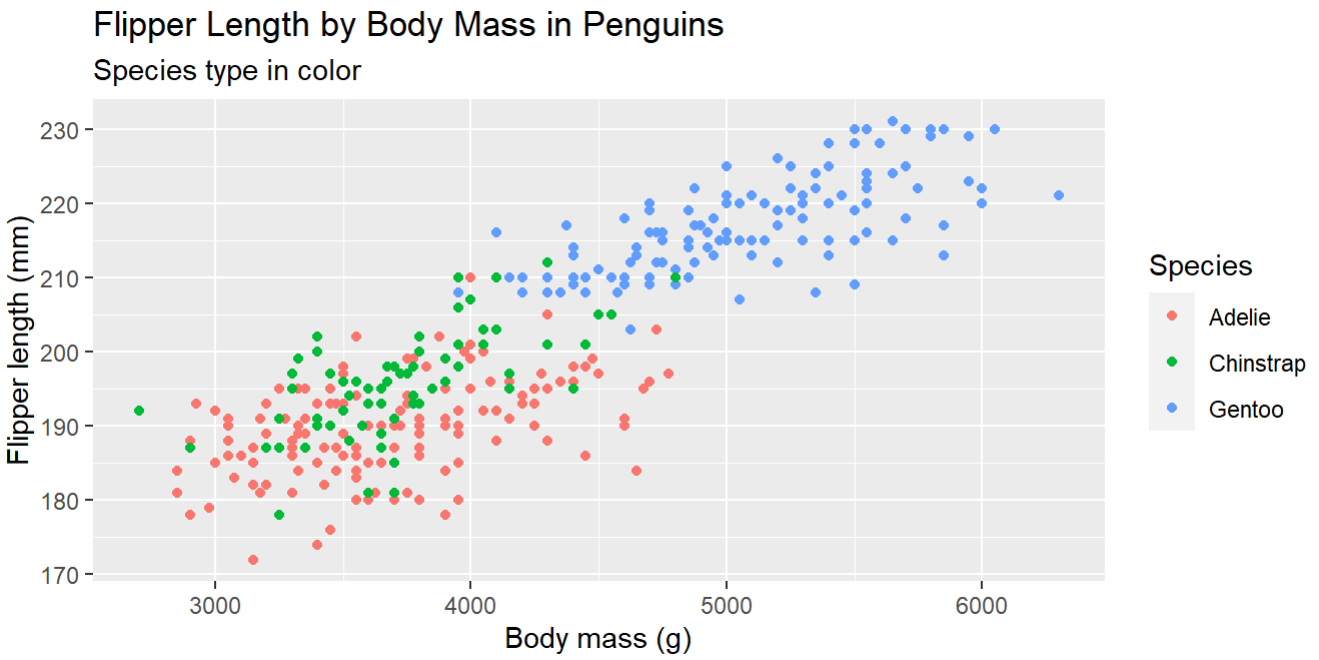
NOTE: Reference the code available to you in the slides or the code-along .R file.

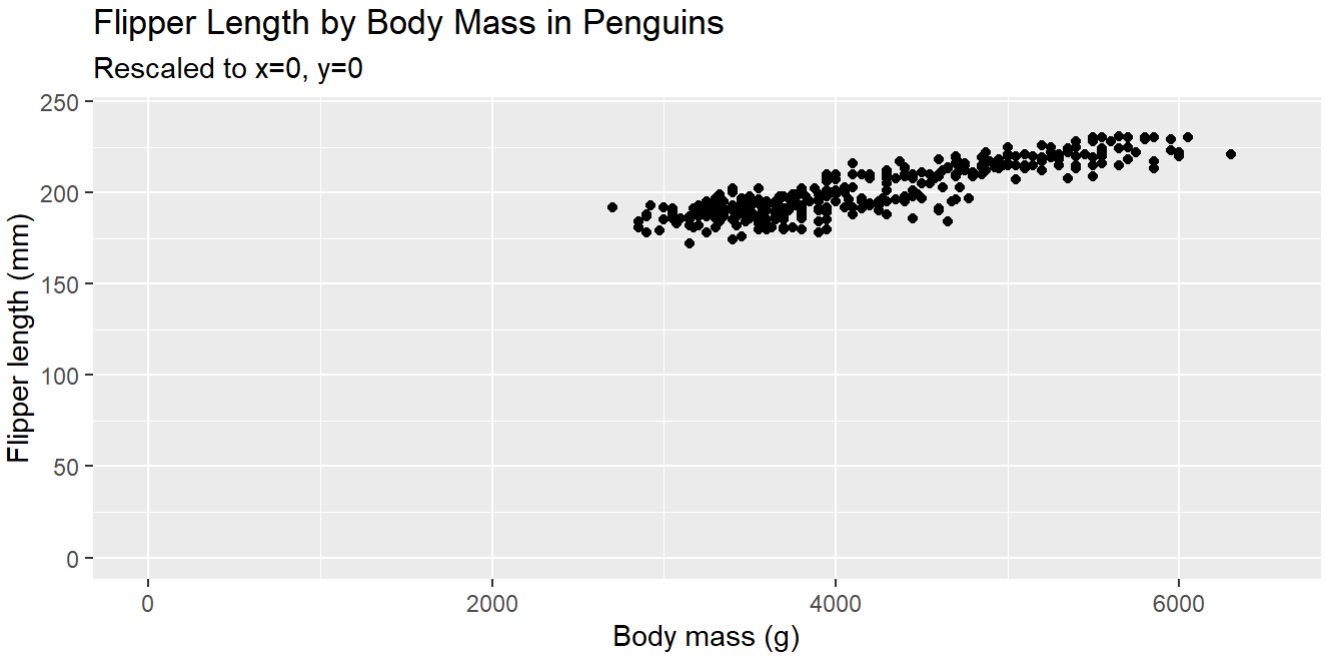
2.0 Adding a Layer

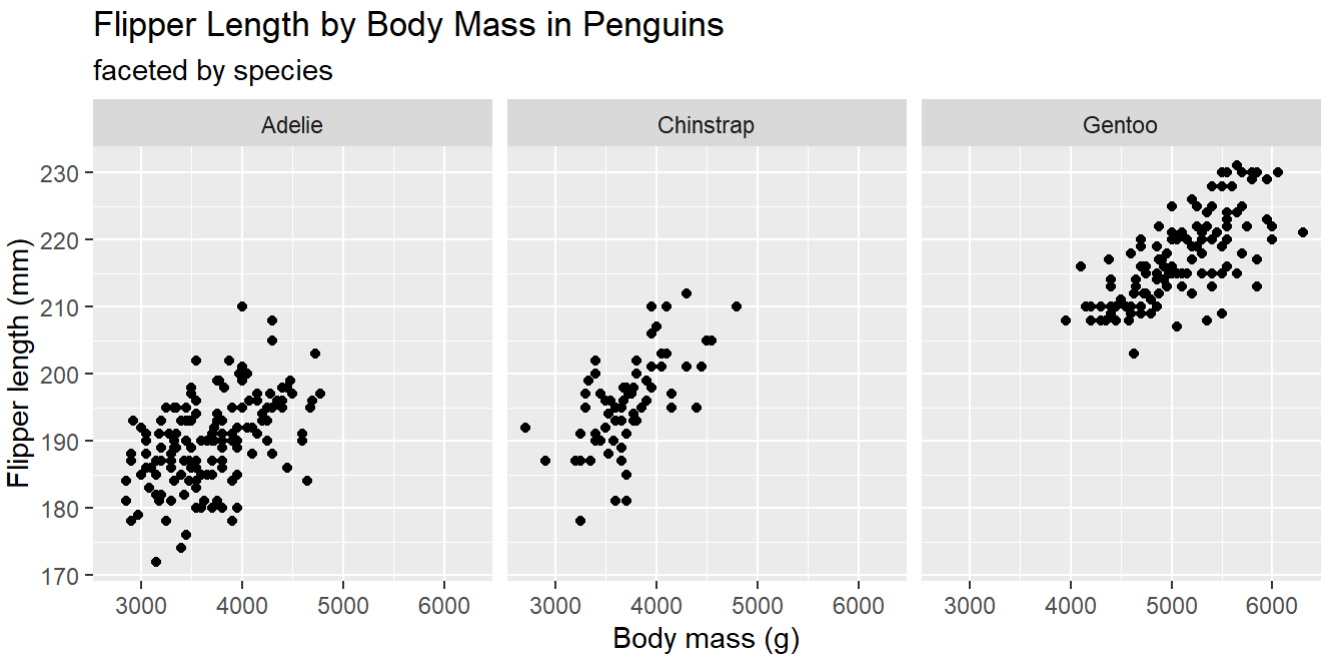
Add a layer to your base plot with any *one* of the following:

- Mapped aesthetic
- Scale change (axis, color, size, etc.)
- Facet

3.0 Examples


Flipper Length by Body Mass in Penguins
Species type in color
Flipper length (mm)
230
220
210
200
190
180
170
3000
4000
5000
6000
Body mass (g)
Species
Adelie
Chinstrap
Gentoo


Flipper Length by Body Mass in Penguins
Rescaled to x=0, y=0
Flipper length (mm)
250
200
150
100
50
0
0
2000
4000
6000
Body mass (g)


Flipper Length by Body Mass in Penguins
faceted by species
Adelie
Chinstrap
Gentoo
Flipper length (mm)
230
220
210
200
190
180
170
3000
4000
5000
6000
Body mass (g)

Data Visualization with ggplot2 in R

Activity #2: Customizing Themes

0.0 Getting Started

Load the appropriate packages. Recreate the base plot you generated in Activity #1. Store the plot as `plot`.

```
library(ggplot2)
library(palmerpenguins)
```

1.0 Using pre-packaged themes

Add a pre-packaged theme to your base plot:

- theme_grey()
- theme_bw()
- theme_linedraw()
- theme_light()
- theme_dark()
- theme_minimal()
- theme_classic()

2.0 Customizing your own theme

2.1 Add fonts

Add fonts that R can use for customizing plots.

```
#### For Windows Users ####
install.packages('extrafont')
library('extrafont')

font_import()
# You will be prompted to continue [y/n]
# Type 'y' and press enter

loadfonts(device="win")


#### For Mac Users ####
install.packages('extrafont')
library('extrafont')
```

```
font_import()
# You will be prompted to continue [y/n]
# Type 'y' and press enter

loadfonts()
```

Check what fonts are available with `fonts()`.

2.2 Customizing theme elements

Using the `theme()` function, customize the following theme elements:

- plot.title
- plot.subtitle
- panel.background
- panel.border
- panel.background
- panel.grid
- axis.title
- axis.text

You should use some combination of the following adjustments in your theme:

- color
- fill
- size
- linewidth
- family
- face (bold, italic, etc.)

Be creative. Effective data communication requires important information to be interpretable *and* visually inviting.

2.3 Saving your theme

Store your theme as `my_theme`.

2.4 Reusing your theme

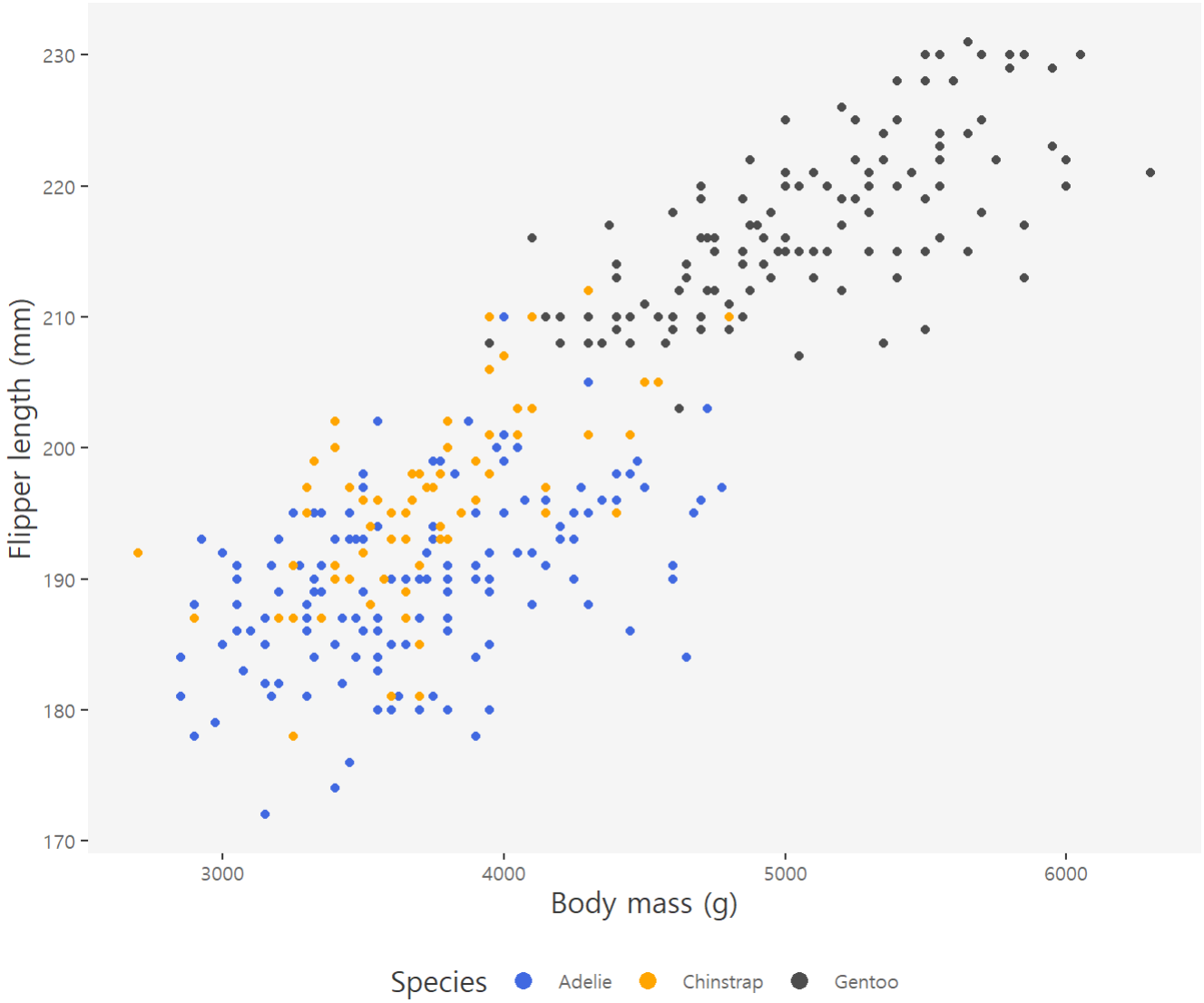
Run the following code:

```
plot + my_theme
```

3.0 Examples

Flipper Length by Body Mass in Penguins

Species type in color



Data Visualization with ggplot2 in R

Activity #3: Creating and Customizing Complex Graphics

0.0 Getting Started

Load the appropriate packages.

```
library(ggplot2)
library(palmerpenguins)
```

1.0 Visualizing Data, from start to finish

Choose one of the following challenges. You must perform the following:

1. Decide which type of graphic is most appropriate for the graphic.
2. Hand sketch the desired plot.
3. Determine what additional features/aesthetics are required.
4. Build the base plot.
5. Customize the theme.

There is no single correct approach, though you should be mindful of the guidelines discussed during the workshop. You will have an opportunity to present your graphic and hear feedback from the instructors and fellow workshop participants.

Alternatively, you can create a visualization using your own data. You will still have an opportunity to present and receive feedback.

Challenge 1 (Difficulty : Low)

Plot how many penguins were studied by island and species.

Challenge 2 (Difficulty : Med.)

Show the relationship between body mass (x-axis) and bill length and bill depth (both y-axis) for each species. Provide both the line of best fit and the underlying data points.

Challenge 3 (Difficulty : High)

Plot the change in the following for each species from 2007 to 2009:

- bill_length_mm
- bill_depth_mm
- flipper_length_mm
- body_mass_g