NC STATE UNIVERSITY

Summarizing Data: Graphical Displays via `ggplot2`

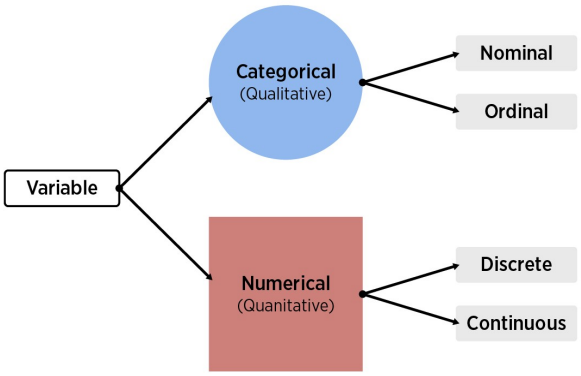
Justin Post

Where are we at?

- Understand types of data and their distributions
- Numerical summaries (across subgroups)
    - Contingency Tables
    - Mean/Median
    - Standard Deviation/Variance/IQR
    - Quantiles/Percentiles
- Graphical summaries (across subgroups)
    - Bar plots (categorical data)
    - Histograms
    - Box plots
    - Scatter plots

Understanding Data

- How to summarize data?
- Depends on data type:
  - Categorical (Qualitative) - variable whose entries are a label or attribute
  - Numeric (Quantitative) - variable whose entries are a numerical value where math can be performed



Plotting

Three major systems for plotting:

- Base R (built-in functions)
  - Use `plot`, `barplot`, `hist`, etc. to start a plot
  - Annotate the plot using functions like `text`, `lines`, `points`, etc.
- Lattice (not covered)
- ggplot2 (sort of part of the tidyverse - Cheat Sheet)
  - `ggplot(data = data_frame)` creates a plot instance
  - Add “layers” to the system (geoms or stats)

Great reference book here!

`ggplot2` Plotting

ggplot2 basics (Cheat Sheet)

- `ggplot(data = data_frame)` creates a plot instance
- Add “layers” to the system (geoms or stats)
    - Creates a visualization of the data

`ggplot2` Plotting

ggplot2 basics (Cheat Sheet)

- `ggplot(data = data_frame)` creates a plot instance
- Add “layers” to the system (geoms or stats)
  - Creates a visualization of the data
- Modify layer “mapping” args (aes)
  - Ex: size, color, and x, y location(s)
- Coordinate system (mostly use Cartesian plane)
- Optional: Titles, etc.

factors

- factor - special class of vector with a `levels` attribute
- Levels define all possible values for that variable
  - Great for variable like `Day` (Monday, Tuesday, …)
  - Not great for variable like `Name` where new values may come up
- Quite useful with plotting
  - Allows for easy labeling of subgroups

factors

- Consider data on titanic passengers in `titanic.csv`

```
titanicData <- read_csv("../datasets/titanic.csv")
#convert survival status to a factor
titanicData$survived <- as.factor(titanicData$survived)
levels(titanicData$survived) #R knows it isn't numeric now
```

```
## [1] "0" "1"
```

- Can’t add value unless it is a level

```
titanicData$survived[1] <- "5"
```

```
## Warning in `[<-.factor`(`*tmp*`, 1, value = structure(c(NA, 2L, 1L, 1L, :
## invalid factor level, NA generated
```

`factor` levels

```
#overwrite with another possible level
levels(titanicData$survived) <- c(levels(titanicData$survived), "5")
levels(titanicData$survived)
```

```
## [1] "0" "1" "5"
```

```
#no error
titanicData$survived[1] <- "5"
```

`factor` levels

- Useful if you want to create better labels

```
levels(titanicData$survived) <- c("Died", "Survived", "Other")
levels(titanicData$survived)
```

```
## [1] "Died"     "Survived" "Other"
```

`factor` levels

- Useful if you want to create better labels

```
levels(titanicData$survived) <- c("Died", "Survived", "Other")
levels(titanicData$survived)
```

```
## [1] "Died"     "Survived" "Other"
```

- Useful if you want to change order of values (say for plotting!)

```
#or use ordered()
titanicData$survived <- factor(titanicData$survived,
                               levels(titanicData$survived)[c(3, 2, 1)])
levels(titanicData$survived)
```

```
## [1] "Other"    "Survived" "Died"
```

`ggplot2` Plotting: Categorical variables

Categorical variable - entries are a label or attribute

Goal: describe relative frequency (or count) in each category

- Generally, use a barplot!

ggplot2 barplots

- Barplots via `ggplot + geom_bar`
- Consider data on titanic passengers in `titanic.csv`

```
titanicData <- read_csv("../datasets/titanic.csv")
titanicData$survived <- as.factor(titanicData$survived)
levels(titanicData$survived) <- c("Died", "Survived")
titanicData
```

```
## # A tibble: 1,310 x 14
##   pclass survived name  sex      age sibsp parch ticket  fare cabin embarked
##    <dbl> <fct>    <chr> <chr>  <dbl> <dbl> <dbl> <chr>  <dbl> <chr> <chr>
## 1      1 Survived Alle~ fema~ 29         0     0 24160   211. B5    S
## 2      1 Survived Alli~ male   0.917     1     2 113781  152. C22 ~ S
## 3      1 Died     Alli~ fema~  2         1     2 113781  152. C22 ~ S
## 4      1 Died     Alli~ male  30         1     2 113781  152. C22 ~ S
## 5      1 Died     Alli~ fema~ 25         1     2 113781  152. C22 ~ S
## # ... with 1,305 more rows, and 3 more variables: boat <chr>, body <dbl>,
## #   home.dest <chr>
```

`ggplot2` barplots

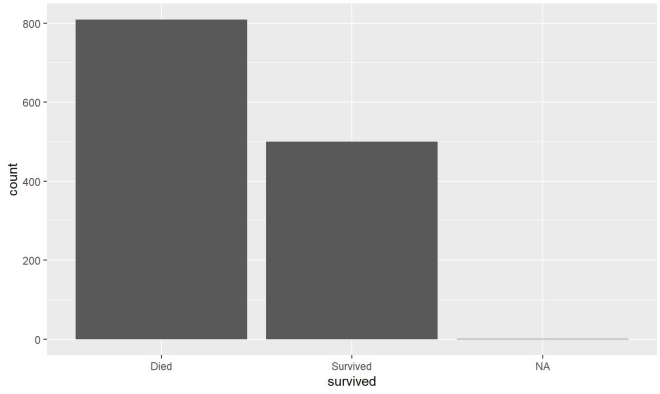
- Barplots via `ggplot + geom_bar`
- Across x-axis we want our categories - specify with `aes(x = ...)`

```
ggplot(data = titanicData, aes(x = survived))
```

`ggplot2` barplots

- Barplots via `ggplot + geom_bar`
- Must add geom (or stat) layer!

```
ggplot(data = titanicData, aes(x = survived)) + geom_bar()
```



ggplot2 barplots

- Generally: Save base object, then “add layers”

```
g <- ggplot(data = titanicData, aes(x = survived))
g + geom_bar()
```

`ggplot2` barplots

- Generally: Save base object, then “add layers”

```
g <- ggplot(data = titanicData, aes(x = survived))
g + geom_bar()
```

- `aes()` defines visual properties of objects in the plot

  `x = , y = , size = , shape = , color = , alpha = , ...`

- Cheat Sheet gives most common properties for a given `geom`

`ggplot2` barplots

- Generally: Save base object, then “add layers”

```
g <- ggplot(data = titanicData, aes(x = survived))
g + geom_bar()
```

- `aes()` defines visual properties of objects in the plot

  `x = , y = , size = , shape = , color = , alpha = , ...`

- Cheat Sheet gives most common properties for a given `geom`

  `d + geom_bar()`

  `x, alpha, color, fill, linetype, size, weight`

`ggplot2` global and local aesthetics

`data` and `aes` can be set in two ways;

- 'globally' (for all layers) via the `ggplot` statement
- 'locally' (for just that layer) via the `geom`, `stat`, etc. layer

`ggplot2` global and local aesthetics

`data` and `aes` can be set in two ways;

- 'globally' (for all layers) via the `ggplot` statement
- 'locally' (for just that layer) via the `geom`, `stat`, etc. layer

```
#global
ggplot(data = titanicData, aes(x = survived)) + geom_bar()
#local
ggplot() + geom_bar(data = titanicData, aes(x = survived))
```

`ggplot2` global and local aesthetics

`data` and `aes` can be set in two ways;

- 'globally' (for all layers) via the `ggplot` statement
- 'locally' (for just that layer) via the `geom`, `stat`, etc. layer

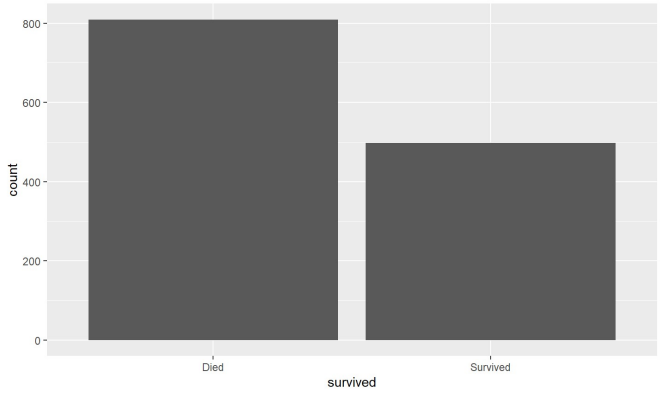
```
#global
ggplot(data = titanicData, aes(x = survived)) + geom_bar()
#local
ggplot() + geom_bar(data = titanicData, aes(x = survived))
```

- To set an attribute that doesn't depend on the data (i.e. `color = 'blue'`), generally place these outside of the `aes`

ggplot2 barplots

- Improve plot by removing `NA` category

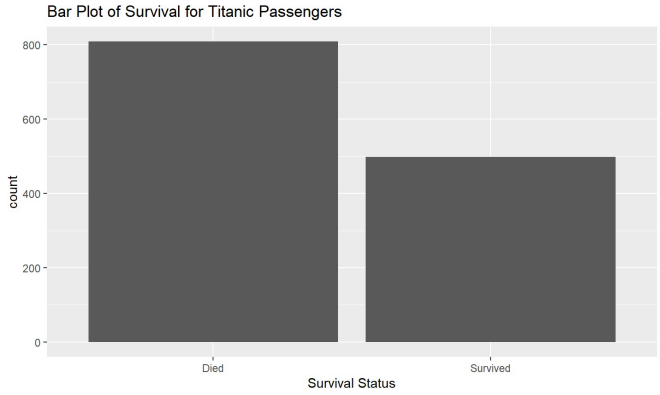
```
titanicData <- titanicData %>% drop_na(survived, sex, embarked)
g <- ggplot(data = titanicData, aes(x = survived))
g + geom_bar()
```



ggplot2 barplots

- Add better labels and a title (new layers, see cheat sheet!)

```
ggplot(data = titanicData, aes(x = as.character(survived))) +
  geom_bar() +
  labs(x = "Survival Status", title = "Bar Plot of Survival for Titanic Passengers")
```



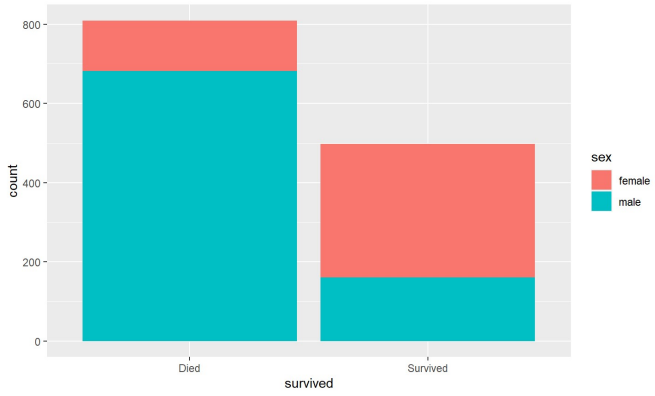
`ggplot2` stacked barplots

- **Stacked barplot** created by via `fill` aesthetic and same process
    - Create base object
    - Add geoms
    - Use aes to specify aspects of the plot

`ggplot2` stacked barplots

- **Stacked barplot** created by via `fill` aesthetic
- Automatic assignment of colors, creation of legends, etc. for `aes` elements (except with `group`)

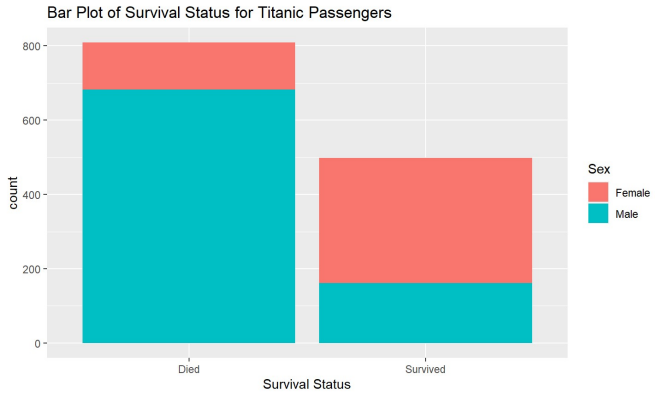
```
ggplot(data = titanicData, aes(x = survived, fill = sex)) + geom_bar()
```



ggplot2 labeling

- Add custom labels by adding more layers

```
ggplot(data = titanicData, aes(x = survived, fill = sex)) +
  geom_bar() +
  labs(x = "Survival Status",
       title = "Bar Plot of Survival Status for Titanic Passengers") +
  scale_fill_discrete(name = "Sex", labels = c("Female", "Male"))
```



ggplot2 labeling

- Adjusting appropriate labeling via `scale_*_discrete`

```
aes(x = survived, fill = sex)
```

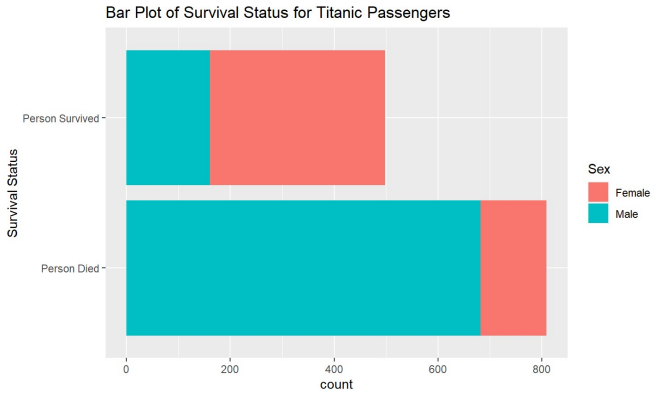
```
scale_x_discrete(labels = c("Person Died", "Person Survived"))
```

```
scale_fill_discrete(name = "Sex", labels = c("Female","Male"))
```

ggplot2 horizontal barplots

- Easy to rotate a plot with `coord_flip`

```
ggplot(data = titanicData, aes(x = survived, fill = sex)) + geom_bar() +
  labs(x = "Survival Status",
       title = "Bar Plot of Survival Status for Titanic Passengers") +
  scale_x_discrete(labels = c("Person Died", "Person Survived")) +
  scale_fill_discrete(name = "Sex", labels = c("Female", "Male")) +
  coord_flip()
```



ggplot2 stat vs geom layers

Note: Most geoms have a corresponding stat that can be used

```
geom_bar(mapping = NULL, data = NULL, stat = "count", position =
"stack", ..., width = NULL, binwidth = NULL, na.rm = FALSE,
show.legend = NA, inherit.aes = TRUE)
```

- Equivalent plots via:

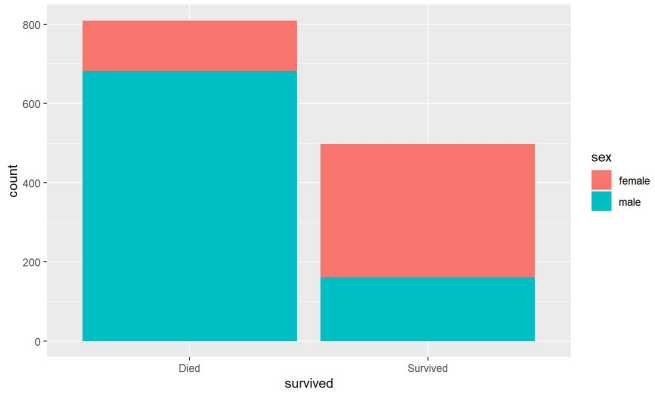
```
ggplot(data = titanicData, aes(x = survived, fill = sex)) + geom_bar()
ggplot(data = titanicData, aes(x = survived, fill = sex)) + stat_count() ```
```

ggplot2 stat vs geom layers

Note: Most geoms have a corresponding stat that can be used

- Can modify the stat: if you have summary data, use `identity`

```
sumData <- titanicData %>% group_by(survived, sex) %>% summarize(count = n())
ggplot(sumData, aes(x = survived, y = count, fill = sex)) +
  geom_bar(stat = "identity")
```



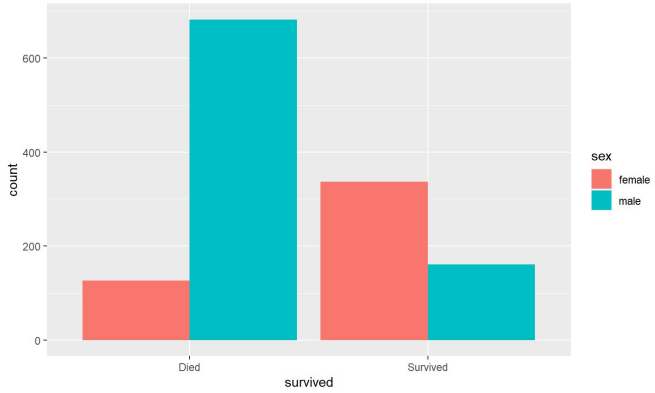
`ggplot2` side-by-side barplots

- **Side-by-side barplot** created by via `position` aesthetic
  - `dodge` for side-by-side bar plot
  - `jitter` for continuous data with many points at same values
  - `fill` stacks bars and standardises each stack to have constant height
  - `stack` stacks bars on top of each other

`ggplot2` side-by-side barplots

- **Side-by-side barplot** created by via `position` aesthetic

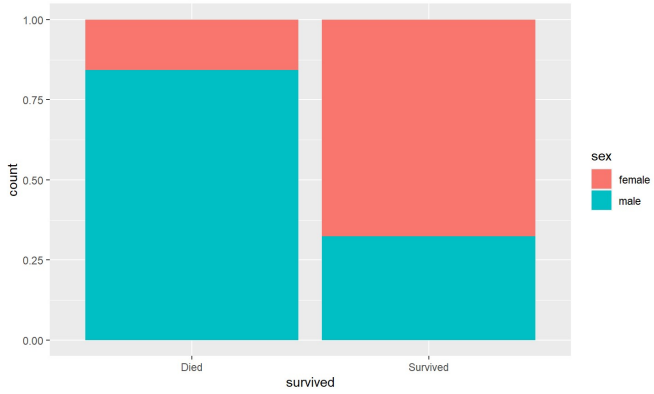
```
ggplot(data = titanicData, aes(x = survived, fill = sex)) +
  geom_bar(position = "dodge")
```



ggplot2 filled barplots

- `position = fill` stacks bars and standardises each stack to have constant height (especially useful with equal group sizes)

```
ggplot(data = titanicData, aes(x = survived, fill = sex)) +
  geom_bar(position = "fill")
```



`ggplot2` faceting

How to create same plot for each `embarked` value? Use **faceting**!

`ggplot2` faceting

How to create same plot for each `embarked` value? Use **faceting**!

`facet_wrap(~ var)` - creates a plot for each setting of `var`

- Can specify `nrow` and `ncol` or let R figure it out

ggplot2 faceting

How to create same plot for each `embarked` value? Use **faceting**!

`facet_wrap(~ var)` - creates a plot for each setting of `var`

- Can specify `nrow` and `ncol` or let R figure it out

`facet_grid(var1 ~ var2)` - creats a plot for each combination of `var1` and `var2`

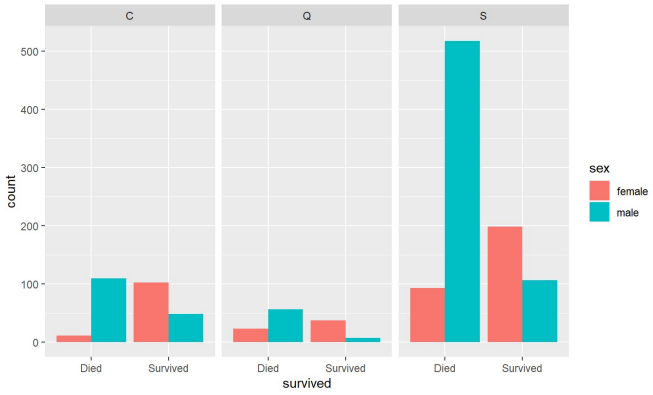
- `var1` values across rows
- `var2` values across columns
- Use `. ~ var2` or `var1 ~ .` to have only one row or column

ggplot2 faceting

How to create same plot for each `embarked` value? Use **faceting**!

- `facet_wrap(~ var)` - creates a plot for each setting of `var`

```
ggplot(data = titanicData, aes(x = survived)) +
  geom_bar(aes(fill =sex), position = "dodge") +
  facet_wrap(~ embarked)
```

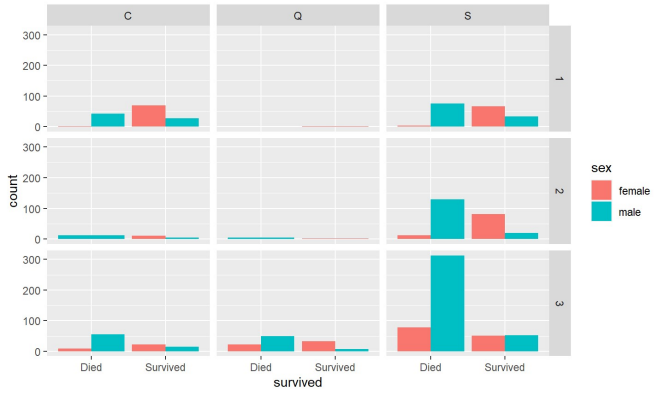


ggplot2 faceting

How to create same plot for each `pclass` and `embarked` combination?

- `facet_grid(var1 ~ var2)` - creats a plot for each combination of `var1` and `var2`

```
ggplot(data = titanicData, aes(x = survived)) +
  geom_bar(aes(fill =sex), position = "dodge") +
  facet_grid(pclass ~ embarked)
```



`ggplot2` Plotting: Categorical variables

Recap!

Categorical variable - entries are a label or attribute

- Most common plot is barplot: `ggplot + geom_bar(aes(x = ...))`
- With summary data, use `geom_bar(stat = identity)`
- Stacked barplot via `aes(fill = )`
- Side-by-side barplot via `position = dodge`
- Filled barplot via `position = fill`

`ggplot2` Plotting Recap

General `ggplot` things:

- Can set local or global `aes`
- Modify titles/labels by adding more layers
- Use either stat or geom layer
- Faceting (multiple plots) via `facet_grid` or `facet_wrap`
- Only need `aes` if setting a mapping value that is dependent on the data (or you want to create a custom legend!)

`ggplot2` Plotting Recap

General `ggplot` things:

- Can set local or global `aes`
- Modify titles/labels by adding more layers
- Use either stat or geom layer
- Faceting (multiple plots) via `facet_grid` or `facet_wrap`
- Only need `aes` if setting a mapping value that is dependent on the data (or you want to create a custom legend!)

Next: Numeric variable plotting

`ggplot2` Plotting: Numeric Variables

Numeric variable - entries are a numerical value where math can be performed

Goal: describe the shape, center, and spread

- Generally, via a histogram or boxplot!

Same process:

- Create base object
- Add geoms
- Use aes to specify aspects of the plot

`ggplot2` Plotting: Numeric Variables

- Look at carbon dioxide (CO2) uptake data set
  - Response recorded: `uptake` CO2 uptake rates in grass plants
  - Environment manipulated: `Treatment` - chilled/nonchilled
  - Ambient CO2 specified and measured: `conc`

```
CO2 <- as_tibble(CO2)
CO2
```

```
## # A tibble: 84 x 5
##   Plant Type   Treatment   conc uptake
##   <ord> <fct>  <fct>      <dbl>  <dbl>
## 1 Qn1   Quebec nonchilled    95   16
## 2 Qn1   Quebec nonchilled   175   30.4
## 3 Qn1   Quebec nonchilled   250   34.8
## 4 Qn1   Quebec nonchilled   350   37.2
## 5 Qn1   Quebec nonchilled   500   35.3
## # ... with 79 more rows
```

`ggplot2` histograms

- **Histogram** - Bins data to show distribution of observations
- Common `aes` values (from cheat sheet):

```
c + geom_histogram(binwidth = 5)
```

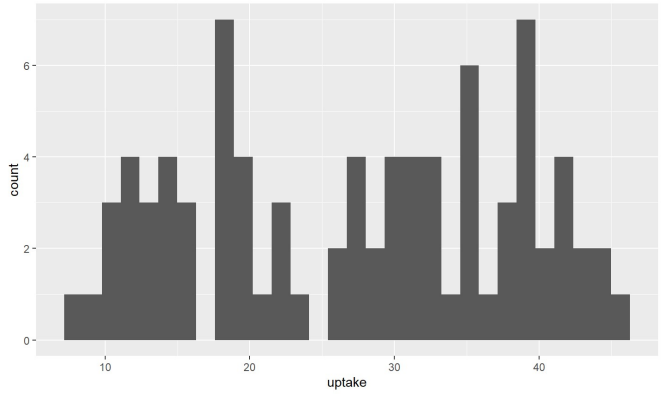
```
x, y, alpha, color, fill, linetype, size, weight
```

- only `x` is really needed

ggplot2 histograms

- **Histogram** - Bins data to show distribution of observations

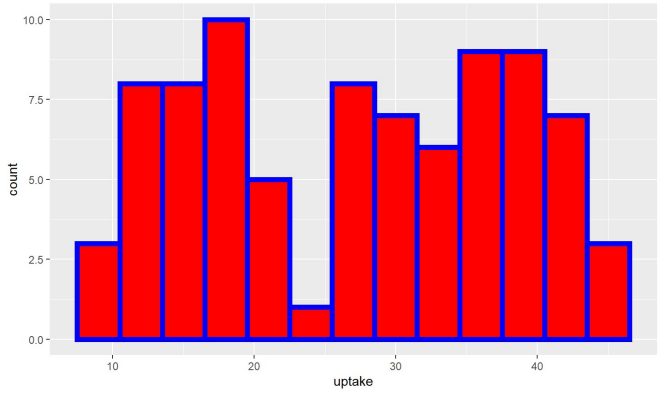
```
g <- ggplot(CO2, aes(x = uptake))
g + geom_histogram()
```



ggplot2 histograms

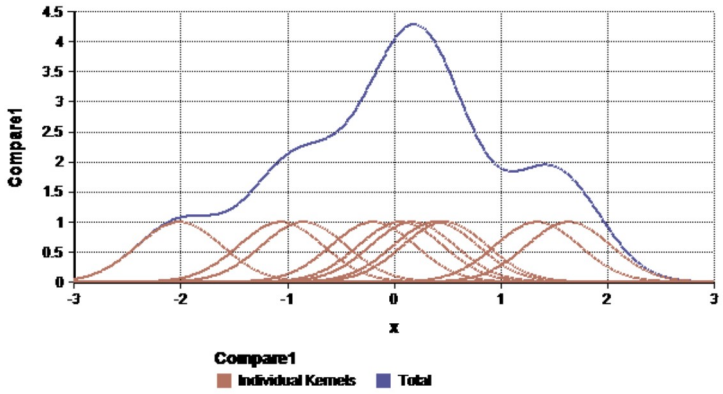
- **Histogram** - Bins data to show distribution of observations
- Attributes of the plot not dependent on data generally set outside of `aes`

```
g + geom_histogram(color = "blue", fill = "red", size = 2, binwidth = 3)
```



`ggplot2` smoothed histogram

- **Kernel Smoother** - Smoothed version of a histogram
- 'Kernel' determines weight given to nearby points



`ggplot2` smoothed histogram

- **Kernel Smoother** - Smoothed version of a histogram
- Common `aes` values (from cheat sheet):

```
c + geom_density(kernel = "gaussian")
```

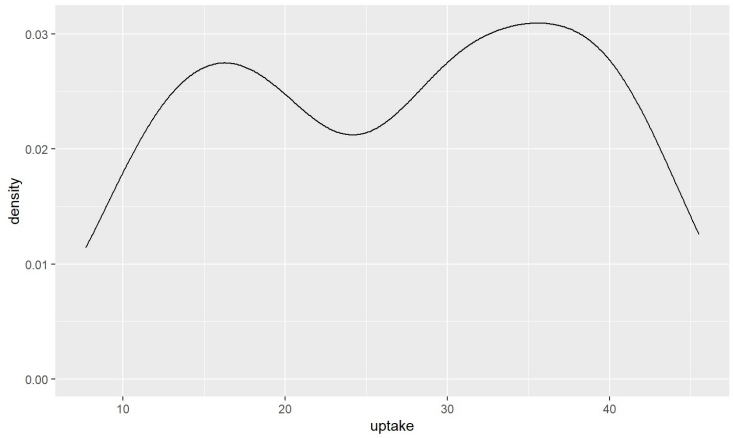
```
x, y, alpha, color, fill, group, linetype, size, weight
```

- Only `x =` is really needed

ggplot2 smoothed histogram

- **Kernel Smoother** - Smoothed version of a histogram

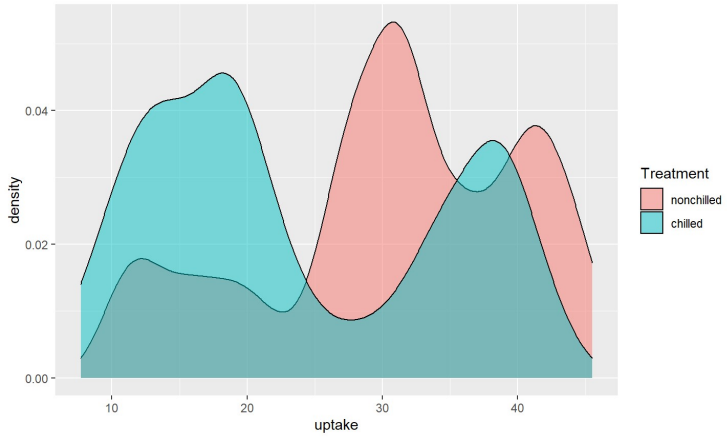
```
ggplot(CO2, aes(x = uptake)) + geom_density()
```



`ggplot2` smoothed histogram

- **Kernel Smoother** - Smoothed version of a histogram
- `fill` a useful aesthetic!

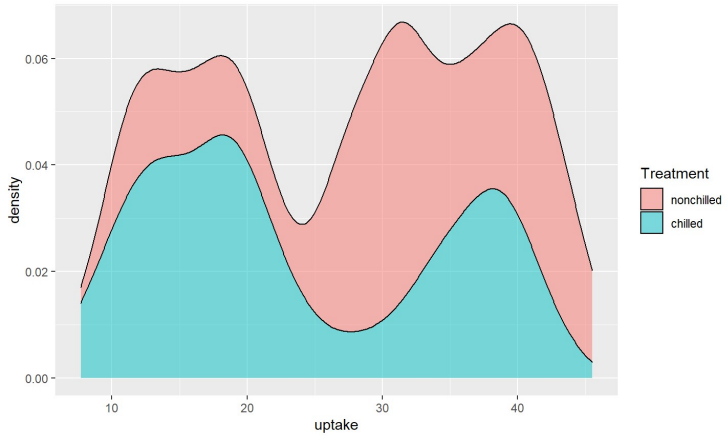
```
ggplot(CO2, aes(x = uptake)) +
  geom_density(adjust = 0.5, alpha = 0.5, aes(fill = Treatment))
```



`ggplot2` smoothed histogram

- **Kernel Smoother** - Smoothed version of a histogram
- recall `position` choices of `dodge`, `jitter`, `fill`, and `stack`

```
ggplot(CO2, aes(x = uptake)) +
  geom_density(adjust = 0.5, alpha = 0.5, position = "stack", aes(fill = Treatment))
```



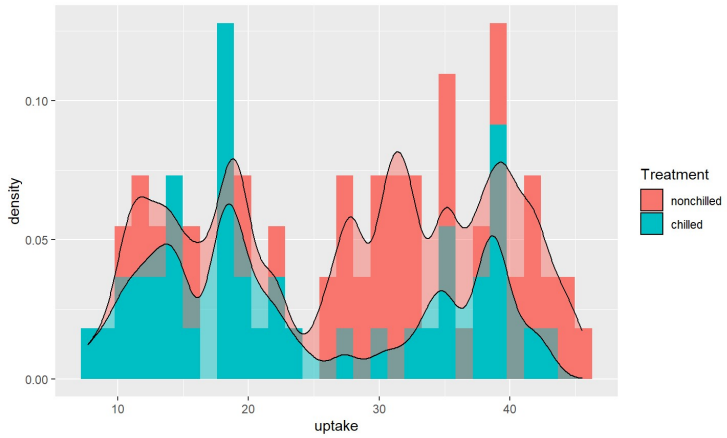
`ggplot2` overlaying plots

- **Histogram + Kernel Smoother**
- Just add both layers! (Works for any compatible layers)
- Use `y = ..density..` to put histogram on same scale

ggplot2 overlaying plots

- Histogram + Kernel Smoother

```
ggplot(CO2) +
  geom_histogram(aes(y = ..density.., x = uptake, fill = Treatment)) +
  geom_density(adjust = 0.25, alpha = 0.5, position = "stack",
               aes(x = uptake, fill = Treatment))
```



`ggplot2` overlaying plots

- **Histogram + Kernel Smoother**
- Better to set global `aes()` options here!
- Can easily change `fill` variable for all layers

```
ggplot(CO2, aes(x = uptake, fill = Treatment)) +
  geom_histogram(aes(y = ..density..)) +
  geom_density(adjust = 0.25, alpha = 0.5, position = "stack")
```

`ggplot2` boxplots

- **Boxplot** - Provides the five number summary in a graph
  - Often used with one numerical and one categorical variable
  - Min, Q1, Median, Q3, Max
  - Often show possible outliers as well

`ggplot2` boxplots

- **Boxplot** - Provides the five number summary in a graph
- Common `aes` values (from cheat sheet):

```
f + geom_boxplot()
```

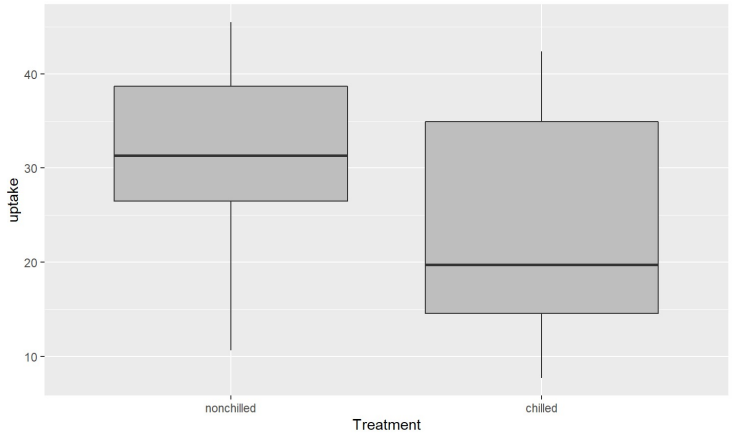
`x, y, lower, middle, upper, ymax, ymin, alpha, color, fill, group, linetype, shape, size, weight`

- Only `x =, y =` are really needed

ggplot2 boxplots

- **Boxplot** - Provides the five number summary in a graph

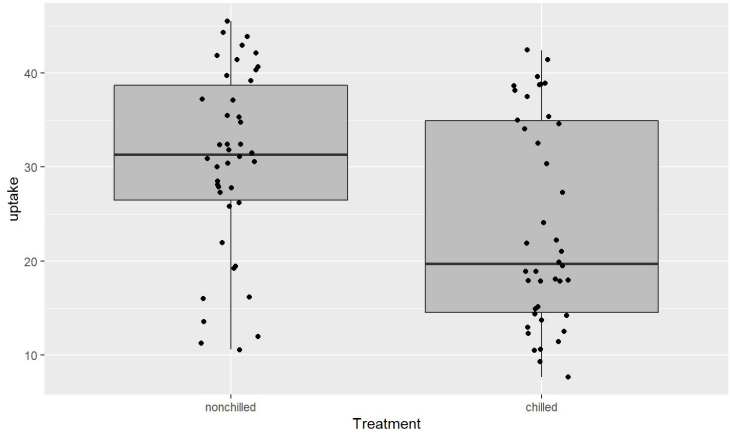
```
g <- ggplot(CO2, aes(x = Treatment, y = uptake))
g + geom_boxplot(fill = "grey")
```



`ggplot2` boxplots with points

- **Boxplot** - Provides the five number summary in a graph
- Can add data points (jittered) to see shape of data better

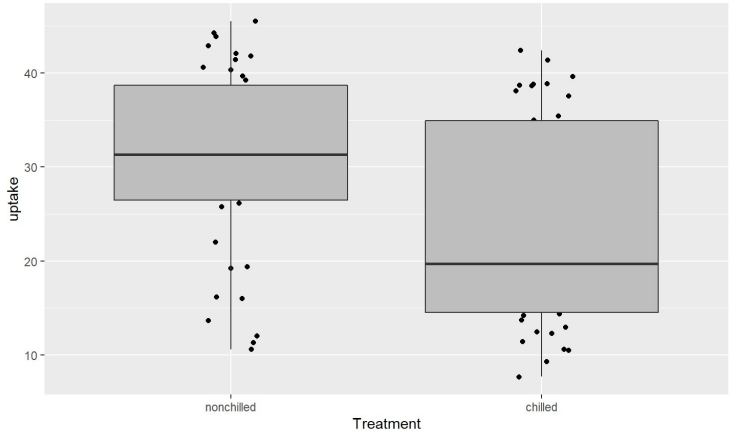
```
ggplot(CO2, aes(x = Treatment, y = uptake)) +
  geom_boxplot(fill = "grey") +
  geom_jitter(width = 0.1)
```



`ggplot2` boxplots with points

- **Boxplot** - Provides the five number summary in a graph
- Order of layers important!

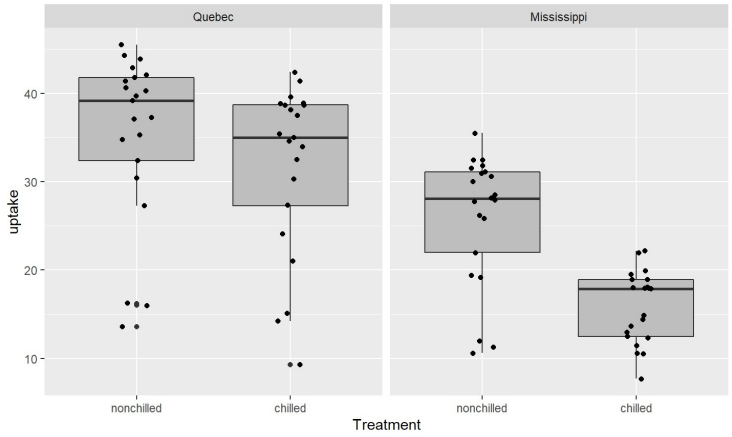
```
ggplot(CO2, aes(x = Treatment, y = uptake)) +
  geom_jitter(width = 0.1) +
  geom_boxplot(fill = "grey")
```



ggplot2 faceting

- **Boxplot** - Provides the five number summary in a graph
- Can `facet` easily!

```
ggplot(CO2, aes(x = Treatment, y = uptake)) + geom_boxplot(fill = "grey") +
  geom_jitter(width = 0.1) + facet_wrap(~ Type)
```



`ggplot2` scatter plots

Two numerical variables

- **Scatter Plot** - graphs points corresponding to each observation
- Common `aes` values (from cheat sheet):

```
e + geom_point()
```

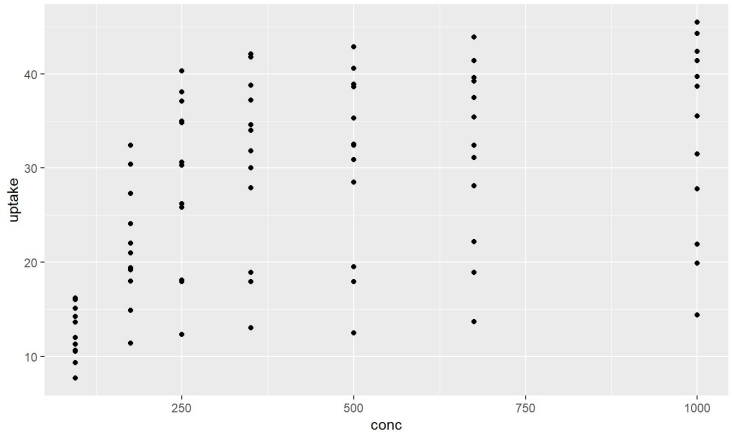
```
x, y, alpha, color, fill, shape, size, stroke
```

- Only `x =, y =` are really needed

ggplot2 scatter plots

- **Scatter Plot** - graphs points corresponding to each observation

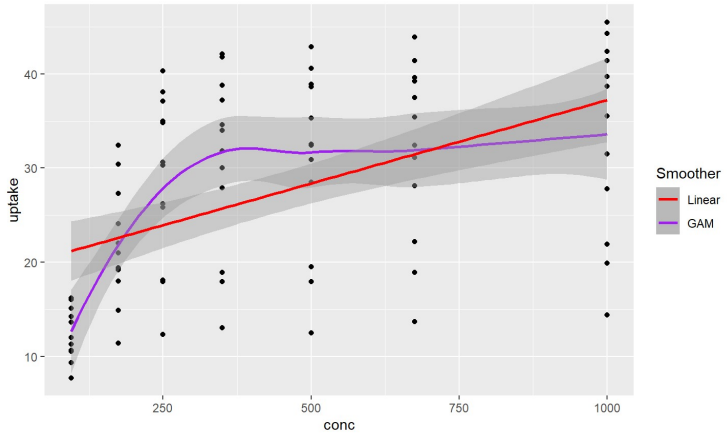
```
g <- ggplot(CO2, aes(x = conc, y = uptake))
g + geom_point()
```



ggplot2 scatter plots with trend line

- Add trend lines easily (linear and loess - a smoother)

```
ggplot(CO2, aes(x = conc, y = uptake)) + geom_point() +
  geom_smooth(aes(col = "loess")) +
  geom_smooth(method = lm, aes(col = "linear")) +
  scale_colour_manual(name = 'Smoother', values =c('linear'='red', 'loess'='purple'),
                      labels = c('Linear','GAM'), guide = 'legend')
```



`ggplot2` scatter plots with text

- May want to add value of correlation to plot
- `paste()` or `paste0()` handy

```
paste("Hi", "What", "Is", "Going", "On", "?", sep = " ")
```

```
## [1] "Hi What Is Going On ?"
```

```
paste("Hi", "What", "Is", "Going", "On", "?", sep = ".")
```

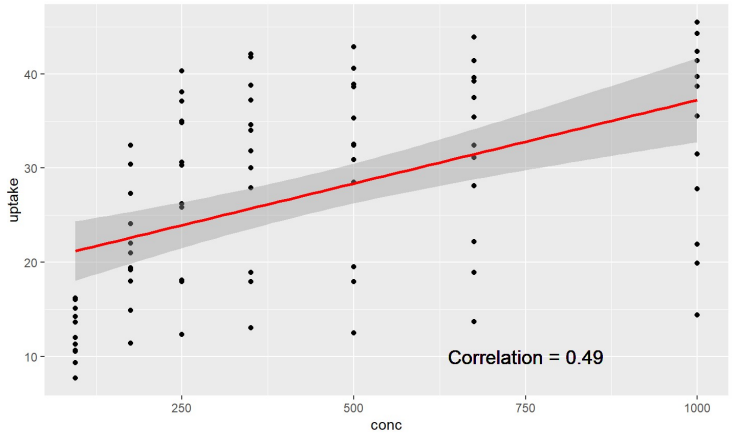
```
## [1] "Hi.What.Is.Going.On.?"
```

```
paste0("Hi", "What", "Is", "Going", "On", "?")
```

```
## [1] "HiWhatIsGoingOn?"
```

ggplot2 scatter plots with text

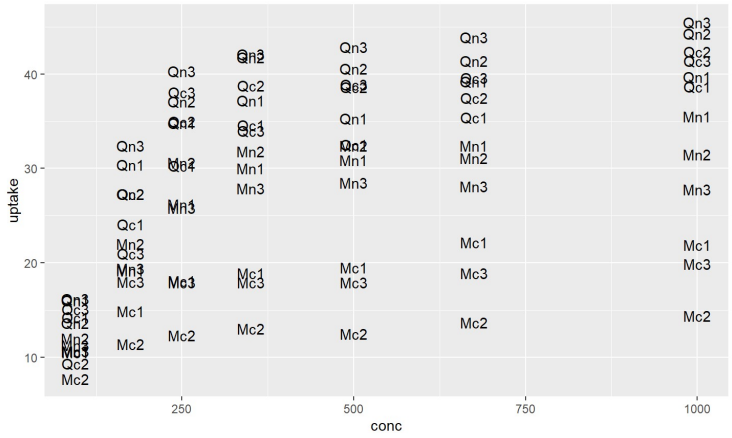
```
correlation <- cor(CO2$conc, CO2$uptake)
ggplot(CO2, aes(x = conc, y = uptake)) + geom_point() +
  geom_smooth(method = lm, col = "Red") +
  geom_text(x = 750, y = 10, size = 5,
            label = paste0("Correlation = ", round(correlation, 2)))
```



`ggplot2` scatter plots with text points

- Text for points with `geom_text`

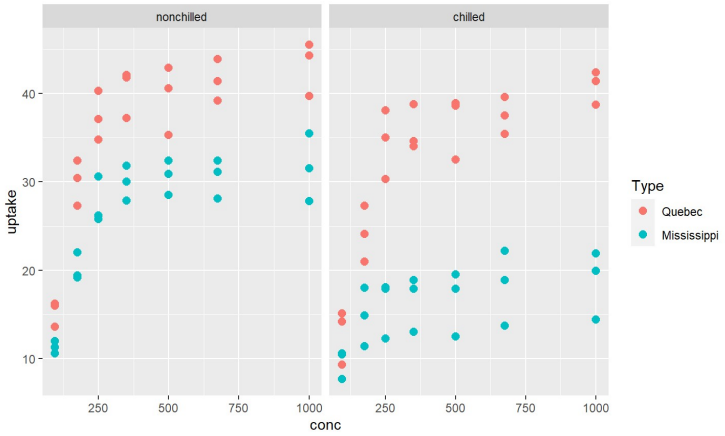
```
ggplot(CO2, aes(x = conc, y = uptake)) +
  geom_text(aes(label = Plant))
```



ggplot2 faceting

- Easily facet! (Note: `cut()` is useful for categorizing a numeric variable)

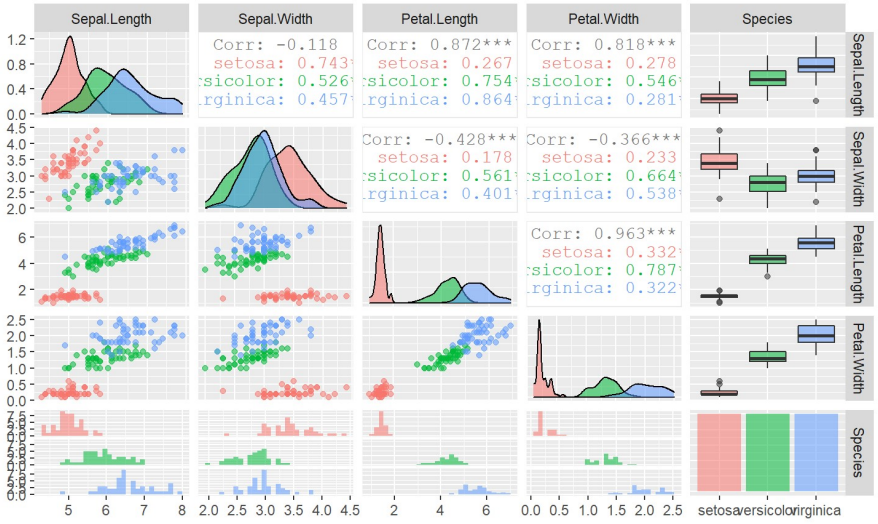
```
ggplot(CO2, aes(x = conc, y = uptake)) +
  geom_point(aes(color = Type), size = 2.5) +
  facet_wrap(~ Treatment)
```



ggpairs

- Saw `pairs`, better with `GGally::ggpairs()`

```
library(GGally) #install GGally if needed
ggpairs(iris, aes(colour = Species, alpha = 0.4))
```



`ggplot2` Plotting: Numeric variables

Recap!

Numeric variable - entries are a numerical value where math can be performed

Most common plots:

- Histogram (`geom_hist`), Density (`geom_density`)
- Boxplot (`geom_boxplot`)
- Scatter plot (`geom_point`), Smoothers (`geom_smooth`)
- Jittered points (`geom_jitter`)
- Text on plot (`geom_text`)

`ggplot2` Plotting Recap

General `ggplot` things:

- Can set local or global `aes`
- Modify titles/labels by adding more layers
- Use either stat or geom layer
- Faceting (multiple plots) via `facet_grid` or `facet_wrap`
- Only need `aes` if setting a mapping value that is dependent on the data (or you want to create a custom legend!)

Recap!

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

- Lots of extensions to ggplot!
    - `gganimate`
    - `ggthemes`
    - `ggmap`
    - `ggrepel`