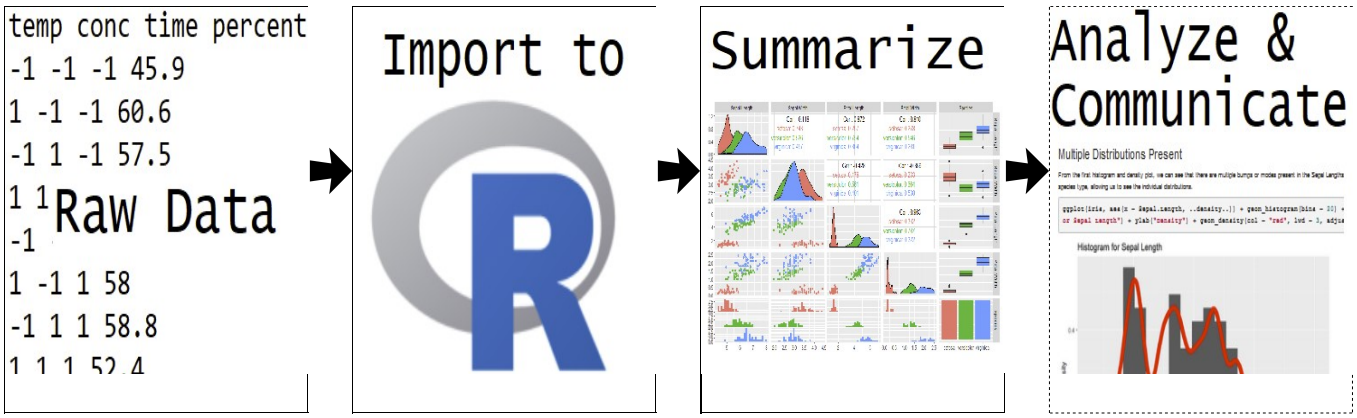
NC STATE UNIVERSITY

Introduction to R for Data Science Part IV

Justin Post

What is this course about?

Basic use of R for reading, manipulating, and plotting data!



Schedule

Day 2

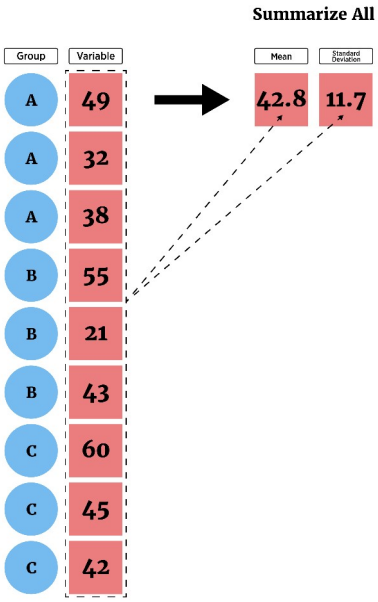
- Logical Statements and Subsetting/Manipulating Data
- **Numerical and Graphical Summaries**
- Basic Analyses

Where do we start?

- Understand types of data and their distributions

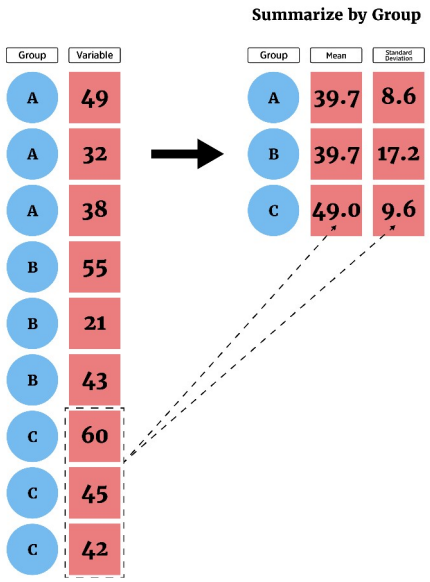
Where do we start?

- Understand types of data and their distributions
- Numerical summaries (across subgroups)



Where do we start?

- Understand types of data and their distributions
- Numerical summaries (across subgroups)



Where do we start?

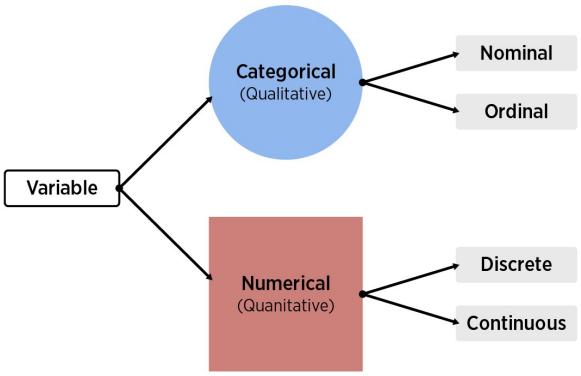
- Understand types of data and their distributions
- Numerical summaries (across subgroups)
    - Contingency Tables
    - Mean/Median
    - Standard Deviation/Variance/IQR
    - Quantiles/Percentiles

Where do we start?

- 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
    - Histograms
    - Box plots
    - Scatter plots

Understanding Data

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



Understanding Data

Common goal: Describe the **distribution** of the variable

- Distribution = pattern and frequency with which you observe a variable
- Categorical variable - describe relative frequency (or count) in each category
- Numeric variable - describe the shape, center, and spread

Contingency tables

Categorical variable - entries are a label or attribute

- Tables (contingency tables) via `table`
    - Show frequency/proportion of categories

Contingency tables

Categorical variable - entries are a label or attribute

- Tables (contingency tables) via `table`
    - Show frequency/proportion of categories
- Consider data on titanic passengers in `titanic.csv`

```
## # A tibble: 1,310 x 14
##   pclass survived name  sex      age sibsp parch ticket  fare cabin embarked
##    <dbl>    <dbl> <chr> <chr>  <dbl> <dbl> <dbl> <chr>  <dbl> <chr> <chr>
## 1      1        1 Alle~ fema~ 29         0     0 24160   211. B5    S
## 2      1        1 Alli~ male   0.917     1     2 113781  152. C22 ~ S
## 3      1        0 Alli~ fema~  2         1     2 113781  152. C22 ~ S
## 4      1        0 Alli~ male  30         1     2 113781  152. C22 ~ S
## 5      1        0 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>
```

Contingency tables

- Create **one-way contingency tables** for each of three categorical variables:
  - embarked (where journey started)
  - survived (survive or not)
  - sex (Male or Female)

```
table(titanicData$embarked)
```

```
##
##   C   Q   S
## 270 123 914
```

```
table(titanicData$survived)
```

```
##
##   0   1
## 809 500
```

```
table(titanicData$sex)
```

```
##
## female   male
##    466    843
```

Two-way contingency tables

- Create **two-way contingency tables** for pairs of categorical variables

```
table(titanicData$survived,
      titanicData$sex)
```

```
##
##     female male
##   0    127  682
##   1    339  161
```

```
table(titanicData$survived,
      titanicData$embarked)
```

```
##
##       C   Q   S
##   0 120  79 610
##   1 150  44 304
```

```
table(titanicData$sex,
      titanicData$embarked)
```

```
##
##             C   Q   S
##   female 113  60 291
##   male   157  63 623
```

Three-way contingency tables

- Create a **three-way contingency table** for three categorical variables

```
table(titanicData$sex, titanicData$embarked, titanicData$survived)
```

```
## , ,  = 0
##
##
##            C   Q   S
##   female  11  23  93
##   male   109  56 517
##
## , ,  = 1
##
##
##            C   Q   S
##   female 102  37 198
##   male    48   7 106
```

Three-way contingency tables

- Create a **three-way contingency table** for three categorical variables (order matters for output!)
- Example of an array! 3 dimensions `[ , , ]`

```
tab <- table(titanicData$sex, titanicData$embarked, titanicData$survived)

str(tab)
```

```
##  'table' int [1:2, 1:3, 1:2] 11 109 23 56 93 517 102 48 37 7 ...
##  - attr(*, "dimnames")=List of 3
##   ..$ : chr [1:2] "female" "male"
##   ..$ : chr [1:3] "C" "Q" "S"
##   ..$ : chr [1:2] "0" "1"
```

Conditional contingency tables

- Can obtain **conditional** bivariate info!

```
##  'table' int [1:2, 1:3, 1:2] 11 109 23 56 93 517 102 48 37 7 ...
##  - attr(*, "dimnames")=List of 3
##   ..$ : chr [1:2] "female" "male"
##   ..$ : chr [1:3] "C" "Q" "S"
##   ..$ : chr [1:2] "0" "1"
```

```
#returns embarked vs survived table for females
tab[1, , ]
```

```
##
##       0   1
##   C  11 102
##   Q  23  37
##   S  93 198
```

Conditional contingency tables

- Can obtain **conditional** bivariate info!

```
##  'table' int [1:2, 1:3, 1:2] 11 109 23 56 93 517 102 48 37 7 ...
##  - attr(*, "dimnames")=List of 3
##   ..$ : chr [1:2] "female" "male"
##   ..$ : chr [1:3] "C" "Q" "S"
##   ..$ : chr [1:2] "0" "1"
```

```
#returns embarked vs survived table for males
tab[2, , ]
```

```
##
##       0   1
##   C 109  48
##   Q  56   7
##   S 517 106
```

Conditional contingency tables

- Can obtain **conditional** bivariate info!

```
##  'table' int [1:2, 1:3, 1:2] 11 109 23 56 93 517 102 48 37 7 ...
##  - attr(*, "dimnames")=List of 3
##   ..$ : chr [1:2] "female" "male"
##   ..$ : chr [1:3] "C" "Q" "S"
##   ..$ : chr [1:2] "0" "1"
```

```
#returns survived vs sex table for embarked "C"
tab[, 1, ]
```

```
##
##            0   1
##   female  11 102
##   male   109  48
```

Conditional contingency tables

- Can obtain **conditional** univariate info too!

```
##  'table' int [1:2, 1:3, 1:2] 11 109 23 56 93 517 102 48 37 7 ...
##  - attr(*, "dimnames")=List of 3
##   ..$ : chr [1:2] "female" "male"
##   ..$ : chr [1:3] "C" "Q" "S"
##   ..$ : chr [1:2] "0" "1"
```

```
#Survived status for males that embarked at "Q"
tab[2, 2, ]
```

```
##  0  1
## 56  7
```

Numerical summaries: Numeric variables

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

Single variable:

- Shape: Histogram or Density plot
- Measures of center: Mean, Median
- Measures of spread: Variance, Standard Deviation, Quartiles, IQR

Numerical summaries: Numeric variables

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

Single variable:

- Shape: Histogram or Density plot
- Measures of center: Mean, Median
- Measures of spread: Variance, Standard Deviation, Quartiles, IQR

Two Variables:

- Shape: Scatter plot
- Measures of linear relationship: Covariance, Correlation

Numerical summaries: 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
```

Measures of center

Mean & Median

```
mean(CO2$uptake)
```

```
## [1] 27.2131
```

```
#note you can easily get a trimmed mean
mean(CO2$uptake, trim = 0.05) #5% trimmed mean
```

```
## [1] 27.25263
```

```
median(CO2$uptake)
```

```
## [1] 28.3
```

Measures of spread

Variance, Standard Deviation, Quartiles, & IQR

```
#quartiles and mean
summary(CO2$uptake)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    7.70   17.90   28.30   27.21   37.12   45.50
```

```
var(CO2$uptake)
```

```
## [1] 116.9515
```

```
sd(CO2$uptake)
```

```
## [1] 10.81441
```

```
IQR(CO2$uptake)
```

```
## [1] 19.225
```

```
quantile(CO2$uptake, probs = c(0.1, 0.2))
```

```
##   10%   20%
## 12.36 15.64
```

Measures of linear relationship

Covariance & Correlation

```
cov(CO2$conc, CO2$uptake)
```

```
## [1] 1552.687
```

```
cor(CO2$conc, CO2$uptake)
```

```
## [1] 0.4851774
```

Numerical summaries: Numeric variables

Usually want summaries for different **subgroups of data**

- Ex: Get similar uptake summaries for each **Treatment**

Numerical summaries: Numeric variables

Usually want summaries for different **subgroups of data**

- Ex: Get similar uptake summaries for each **Treatment**
- `dplyr` easy to use but can only return one value

Numerical summaries: Numeric variables

Usually want summaries for different **subgroups of data**

- Ex: Get similar uptake summaries for each **Treatment**
- `dplyr` easy to use but can only return one value

Idea:

- Use `group_by` to create subgroups associated with the data frame
- Use `summarize` to create basic summaries for each subgroup

Summarizing across groups

- Ex: Get similar uptake summaries for each **Treatment**

```
CO2 %>% group_by(Treatment) %>%
    summarise(avg = mean(uptake), med = median(uptake), var = var(uptake))
```

```
## # A tibble: 2 x 4
##   Treatment    avg   med   var
##   <fct>      <dbl> <dbl> <dbl>
## 1 nonchilled  30.6  31.3  94.2
## 2 chilled     23.8  19.7 118.
```

Summarizing across groups

- Ex: Get similar uptake summaries for each **Treatment** and **Concentration**

```
CO2 %>% group_by(Treatment, conc) %>%
        summarise(avg = mean(uptake), med = median(uptake), var = var(uptake))
```

```
## # A tibble: 14 x 5
## # Groups:   Treatment [2]
##    Treatment   conc   avg   med    var
##    <fct>      <dbl> <dbl> <dbl>  <dbl>
##  1 nonchilled    95  13.3  12.8   5.75
##  2 nonchilled   175  25.1  24.6  32.6
##  3 nonchilled   250  32.5  32.7  35.1
##  4 nonchilled   350  35.1  34.5  37.4
##  5 nonchilled   500  35.1  33.8  31.9
##  6 nonchilled   675  36.0  35.8  40.2
##  7 nonchilled  1000  37.4  37.6  49.8
##  8 chilled       95  11.2  10.6   8.18
##  9 chilled      175  19.4  19.5  34.7
## 10 chilled      250  25.3  24.2 112.
## 11 chilled      350  26.2  26.4 117.
## 12 chilled      500  26.6  26   131.
## 13 chilled      675  27.9  28.8 120.
## 14 chilled     1000  29.8  30.3 154.
```

Summarizing across groups

`dplyr` has variations on `summarise` that can be used:

- `summarise_all()` - Apply functions to every column
- `summarise_at()` - Apply functions to specific columns
- `summarise_if()` - Apply functions to all columns of one type

Summarizing across groups

- Ex: Get similar uptake summaries for each **Treatment**
- Built-in `aggregate()` function more general

Summarizing across groups

- Ex: Get similar uptake summaries for each **Treatment**
- Built-in `aggregate()` function more general
- Basic use gives response (`x`) and a `list` of variables to group `by`

```
aggregate(x = CO2$uptake, by = list(CO2$Treatment), FUN = summary)
```

```
##      Group.1   x.Min. x.1st Qu. x.Median   x.Mean x.3rd Qu.   x.Max.
## 1 nonchilled 10.60000  26.47500 31.30000 30.64286  38.70000 45.50000
## 2    chilled  7.70000  14.52500 19.70000 23.78333  34.90000 42.40000
```

Summarizing across groups

- Ex: Get similar uptake summaries for each **Treatment**
- Built-in `aggregate()` function more general
- Commonly used with `formula` notation!

```
aggregate(uptake ~ Treatment, data = CO2, FUN = summary)
```

```
##    Treatment uptake.Min. uptake.1st Qu. uptake.Median uptake.Mean
## 1 nonchilled    10.60000       26.47500      31.30000    30.64286
## 2    chilled     7.70000       14.52500      19.70000    23.78333
##   uptake.3rd Qu. uptake.Max.
## 1       38.70000    45.50000
## 2       34.90000    42.40000
```

Summarizing across groups

- Ex: Get similar uptake summaries for each **Treatment**
- Built-in `aggregate()` function more general
- Commonly used with `formula` notation!

```
aggregate(uptake ~ Treatment, data = CO2, FUN = summary)
```

`uptake ~ Treatment` - formula notation in R

- Idea: uptake (LHS) modeled by Treatment levels (RHS)

Summarizing across groups

- Ex: Get similar uptake summaries for each **Treatment** and **Concentration**
- Built-in `aggregate()` function more general
- Commonly used with `formula` notation!

```
aggregate(uptake ~ Treatment + conc, data = CO2, FUN = summary)
```

`uptake ~ Treatment + conc` model uptake by levels of Treatment and conc

Summarizing across groups

- Ex: Get similar uptake summaries for each **Treatment** and **Concentration**

```
aggregate(uptake ~ Treatment + conc, data = CO2, FUN = summary)
```

```
##     Treatment conc uptake.Min. uptake.1st Qu. uptake.Median uptake.Mean
## 1  nonchilled   95    10.60000       11.47500      12.80000    13.28333
## 2     chilled   95     7.70000        9.60000      10.55000    11.23333
## 3  nonchilled  175    19.20000       20.05000      24.65000    25.11667
## 4     chilled  175    11.40000       15.67500      19.50000    19.45000
## 5  nonchilled  250    25.80000       27.30000      32.70000    32.46667
## 6     chilled  250    12.30000       17.95000      24.20000    25.28333
## 7  nonchilled  350    27.90000       30.45000      34.50000    35.13333
## 8     chilled  350    13.00000       18.15000      26.45000    26.20000
## 9  nonchilled  500    28.50000       31.27500      33.85000    35.10000
## 10    chilled  500    12.50000       18.30000      26.00000    26.65000
## 11 nonchilled  675    28.10000       31.42500      35.80000    36.01667
## 12    chilled  675    13.70000       19.72500      28.80000    27.88333
## 13 nonchilled 1000    27.80000       32.50000      37.60000    37.38333
## 14    chilled 1000    14.40000       20.40000      30.30000    29.78333
##    uptake.3rd Qu. uptake.Max.
## 1        15.40000    16.20000
## 2        13.30000    15.10000
## 3        29.62500    32.40000
## 4        23.32500    27.30000
## 5        36.52500    40.30000
## 6        33.82500    38.10000
## 7        40.65000    42.10000
## 8        34.45000    38.80000
## 9        39.27500    42.90000
## 10       37.07500    38.90000
## 11       40.85000    43.90000
## 12       36.97500    39.60000
## 13       43.15000    45.50000
## 14       40.72500    42.40000
```

Recap/Next Up!

- Understand types of data and their distributions
- Numerical summaries
  - Contingency Tables: `table`
  - Mean/Median: `mean`, `median`
  - Standard Deviation/Variance/IQR: `sd`, `var`, `IQR`
  - Quantiles/Percentiles: `quantile`
- Across subgroups with `dplyr::group_by` and `dplyr::summarize` or `aggregate`
- Graphical summaries (across subgroups)

Activity

- **Numerical Summaries Activity** instructions available on web
- We'll send you to breakout rooms
- One TA or instructor in each room to help out
- Feel free to ask questions about anything you didn't understand as well!

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
- 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"
```

`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

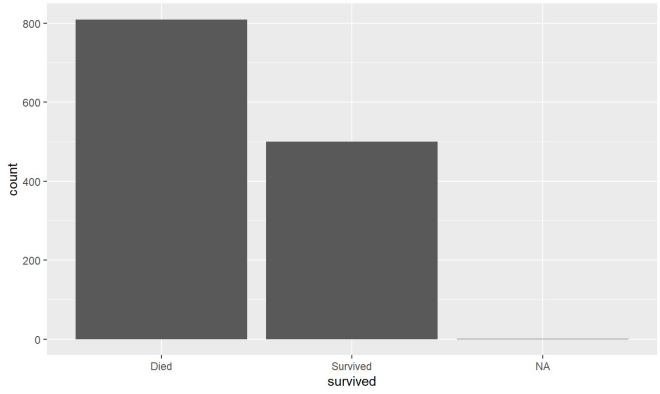
- Barplots via `ggplot + geom_bar`
- Across x-axis we want our categories - specify with `aes(x = ...)`

```
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

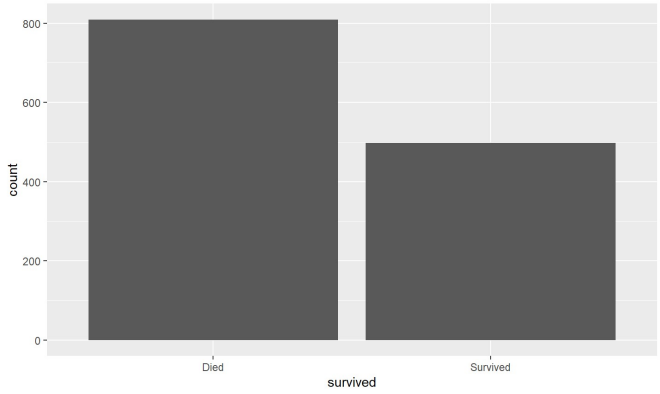
```
#global
ggplot(data = titanicData, aes(x = survived)) + geom_bar()
#local
ggplot() + geom_bar(data = titanicData, aes(x = survived))
```

- 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

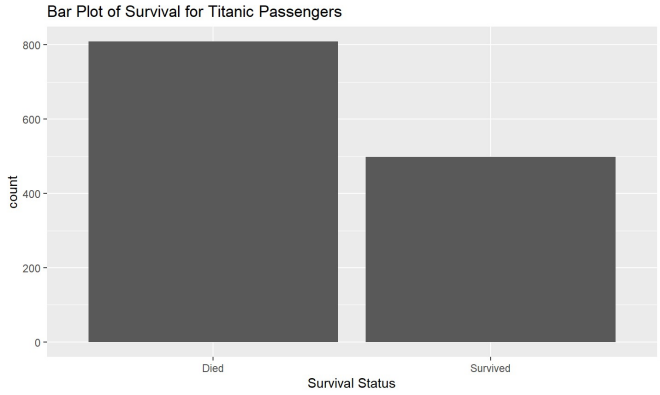
```
titanicData <- titanicData %>% drop_na(survived, sex, embarked)
g <- ggplot(data = titanicData, aes(x = survived))
g + geom_bar()
```



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")
```



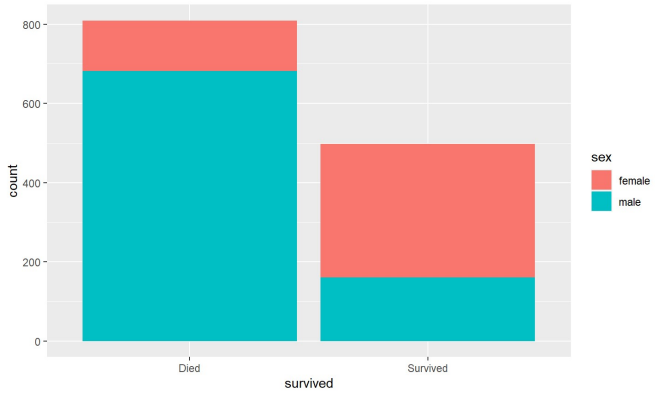
`ggplot2` stacked barplots

- **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`)

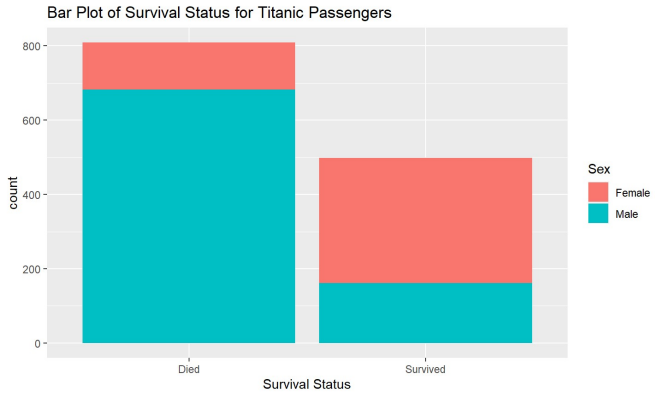
```
ggplot(data = titanicData, aes(x = survived, fill = sex)) + geom_bar()
```



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)
```

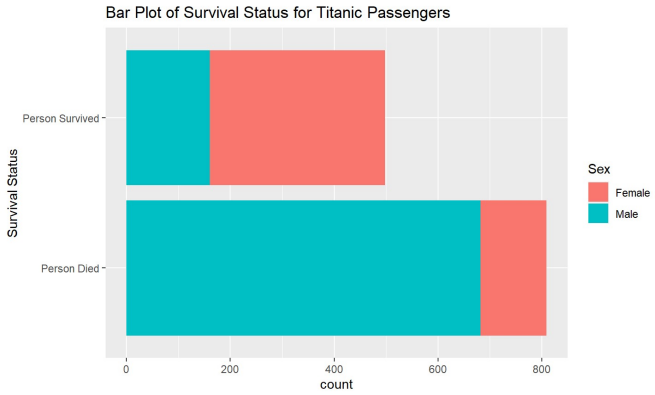
```
scale_x_discrete(labels = c("Person Died", "Person Survived"))
```

```
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:

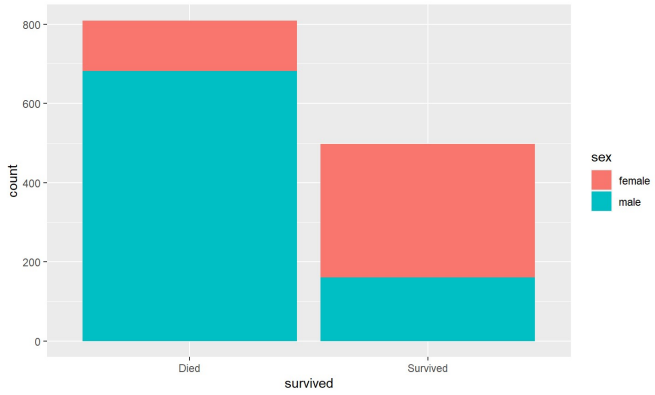
```
ggplot(data = titanicData, aes(x = survived, fill = sex)) + geom_bar()
ggplot(data = titanicData, aes(x = survived, fill = sex)) + stat_count() ```
```

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")
```



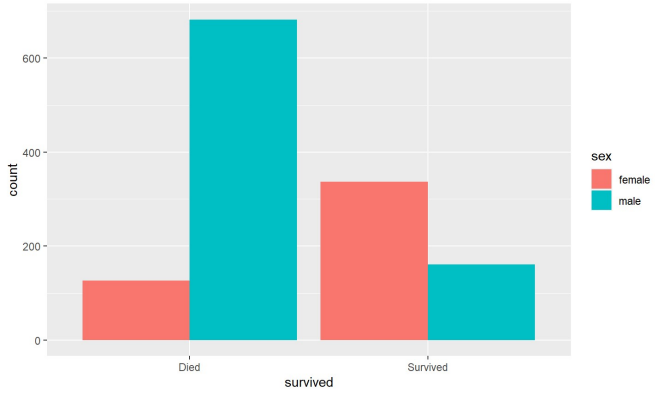
`ggplot2` side-by-side barplots

- **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

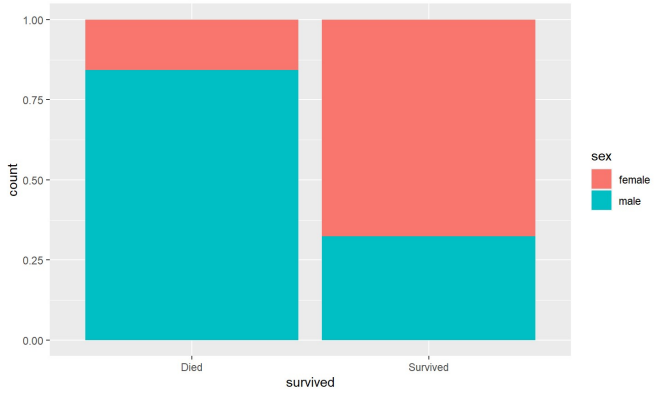
```
ggplot(data = titanicData, aes(x = survived, fill = sex)) +
  geom_bar(position = "dodge")
```



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`

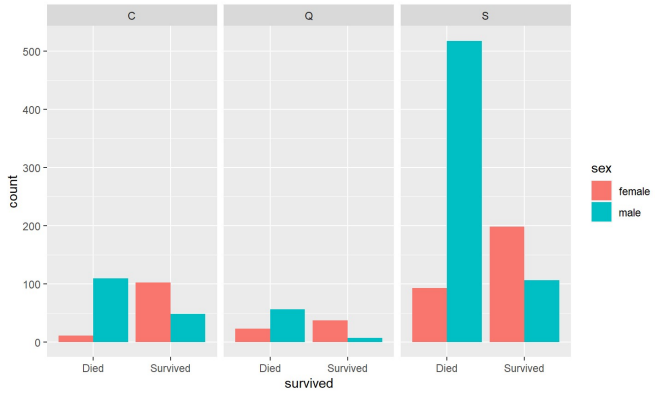
- `var1` values across rows
- `var2` values across columns
- Use `. ~ var2` or `var1 ~ .` to have only one row or column

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)
```

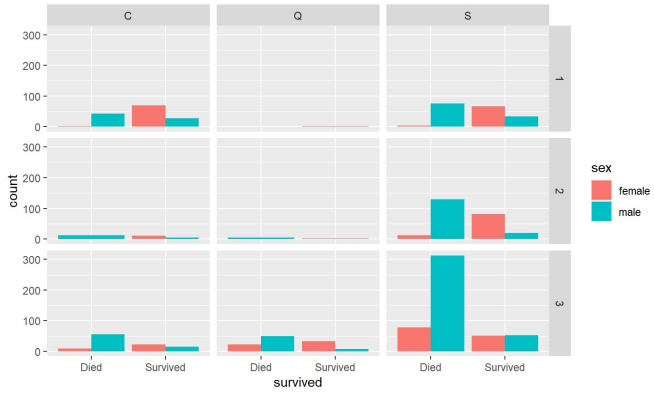


ggplot2 faceting

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

Activity

- **Basic Plotting Activity** instructions available on web
- We'll send you to breakout rooms
- One TA or instructor in each room to help out
- Feel free to ask questions about anything you didn't understand as well!

`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)
```

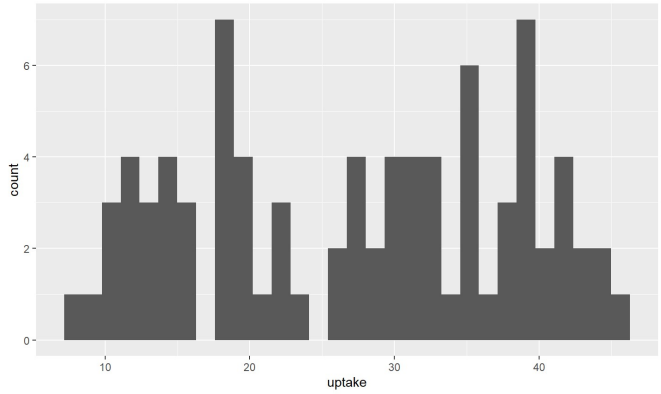
```
x, y, alpha, color, fill, linetype, size, weight
```

- only `x` is really needed

ggplot2 histograms

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

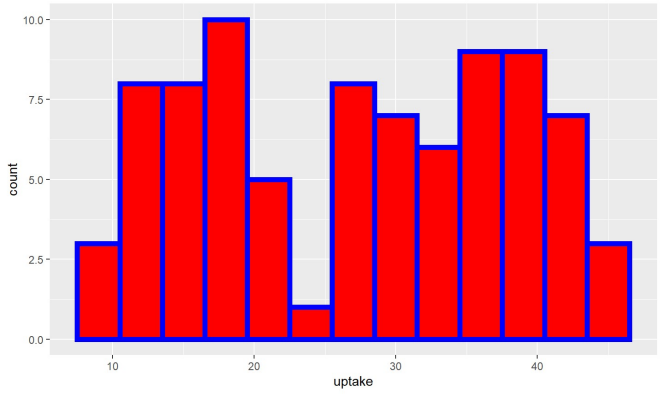
```
g <- ggplot(CO2, aes(x = uptake))
g + geom_histogram()
```



ggplot2 histograms

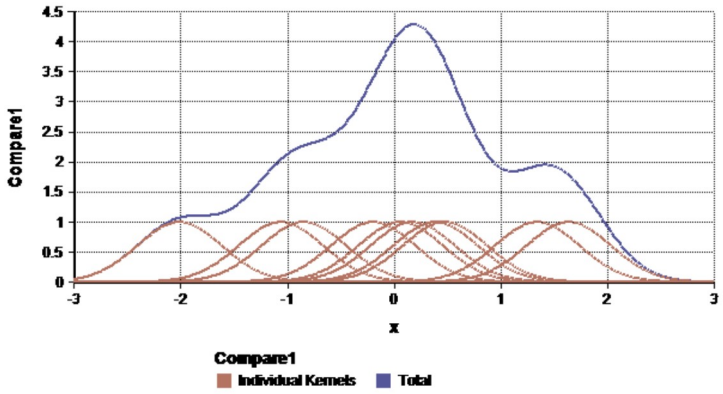
- **Histogram** - Bins data to show distribution of observations
- Attributes of the plot not dependent on data generally set outside of `aes`

```
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")
```

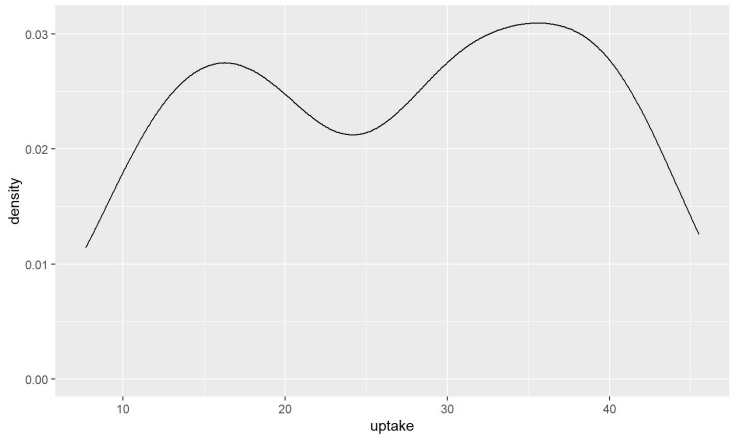
```
x, y, alpha, color, fill, group, linetype, size, weight
```

- Only `x =` is really needed

ggplot2 smoothed histogram

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

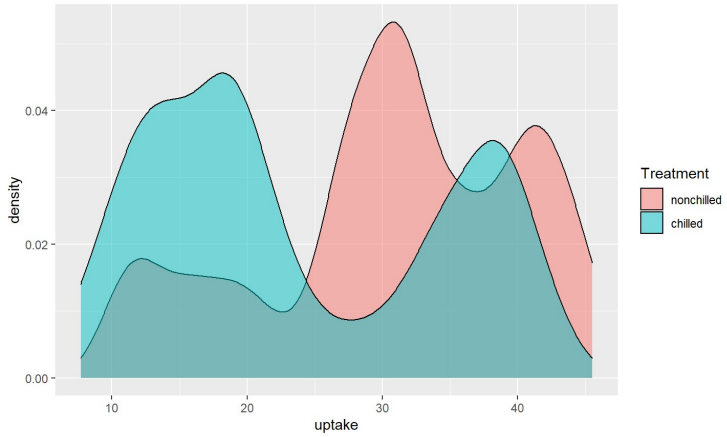
```
ggplot(CO2, aes(x = uptake)) + geom_density()
```



`ggplot2` smoothed histogram

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

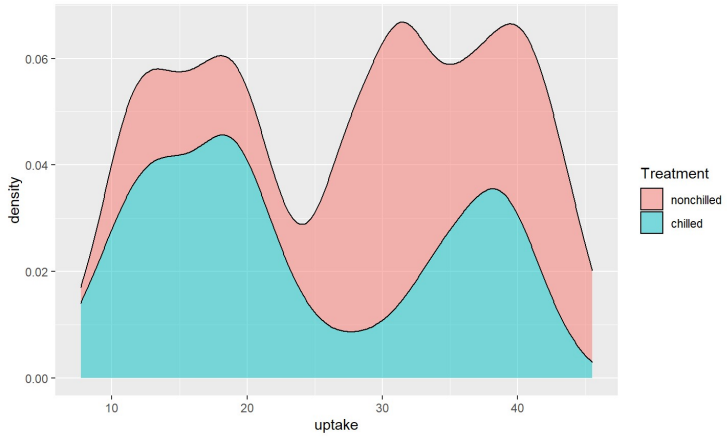
```
ggplot(CO2, aes(x = uptake)) +
  geom_density(adjust = 0.5, alpha = 0.5, aes(fill = Treatment))
```



`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))
```



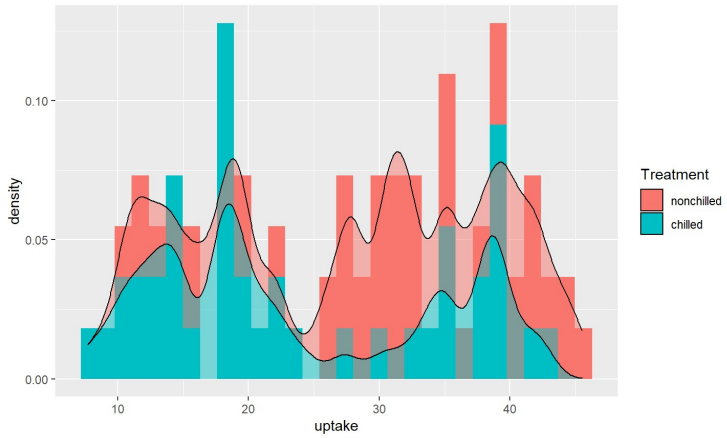
`ggplot2` overlaying plots

- **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()
```

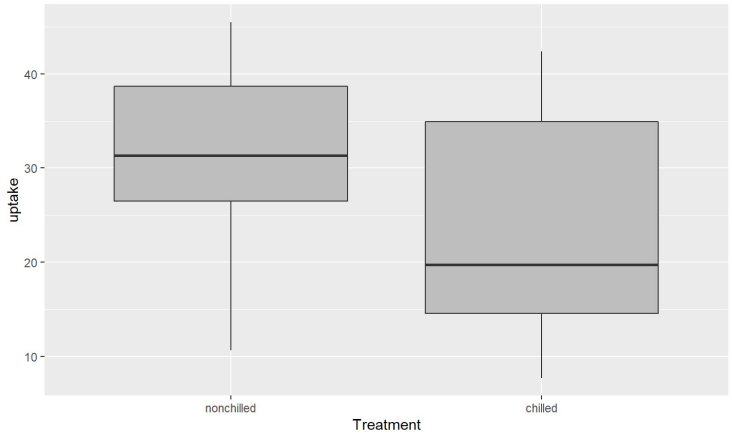
`x, y, lower, middle, upper, ymax, ymin, alpha, color, fill, group, linetype, shape, size, weight`

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

ggplot2 boxplots

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

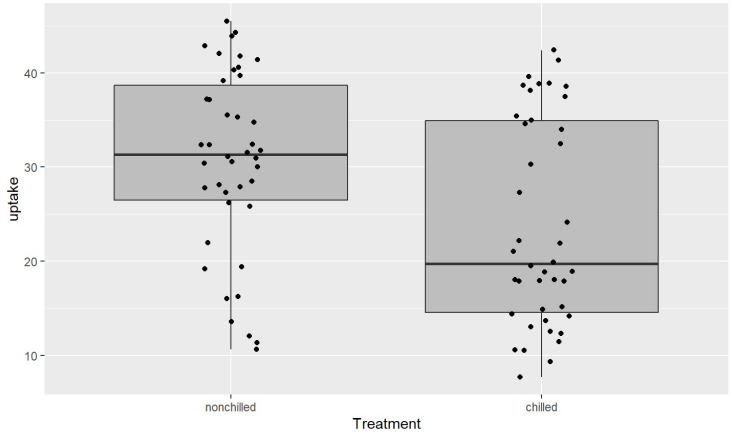
```
g <- ggplot(CO2, aes(x = Treatment, y = uptake))
g + geom_boxplot(fill = "grey")
```



`ggplot2` boxplots with points

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

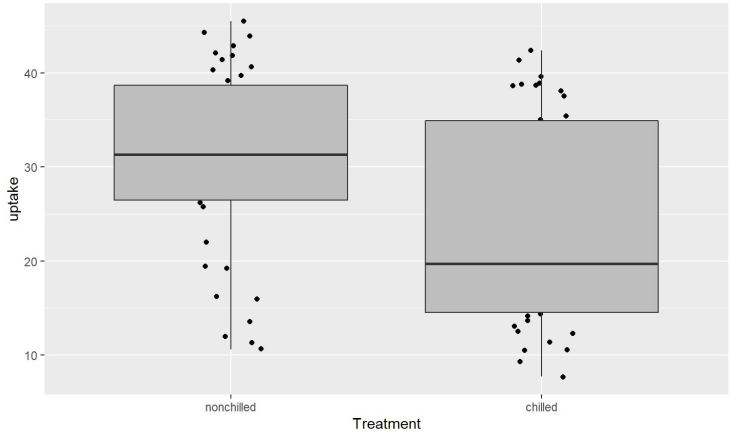
```
ggplot(CO2, aes(x = Treatment, y = uptake)) +
  geom_boxplot(fill = "grey") +
  geom_jitter(width = 0.1)
```



`ggplot2` boxplots with points

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

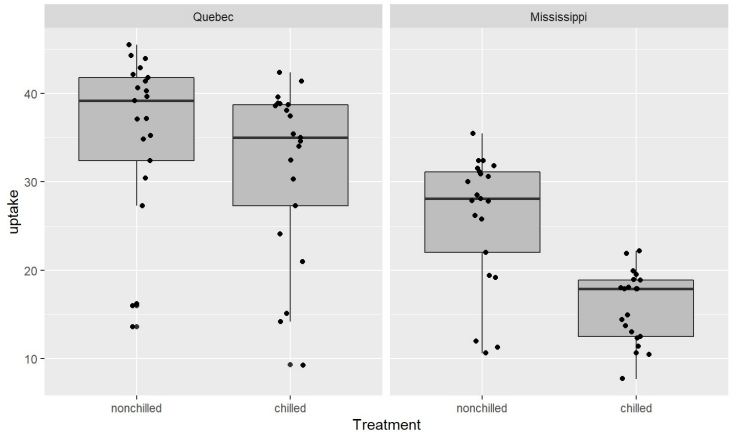
```
ggplot(CO2, aes(x = Treatment, y = uptake)) +
  geom_jitter(width = 0.1) +
  geom_boxplot(fill = "grey")
```



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()
```

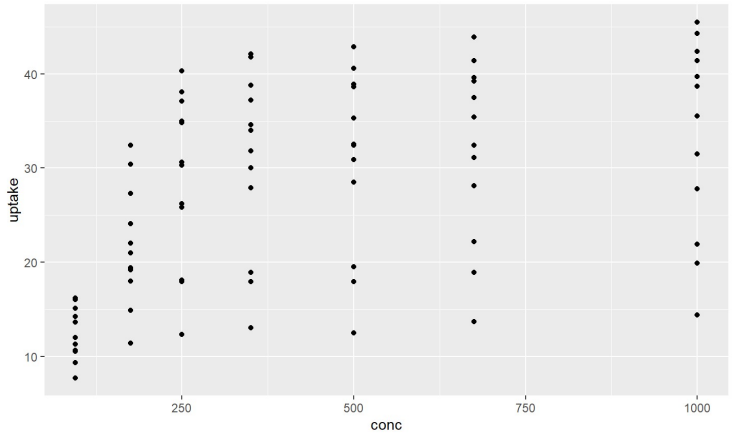
```
x, y, alpha, color, fill, shape, size, stroke
```

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

ggplot2 scatter plots

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

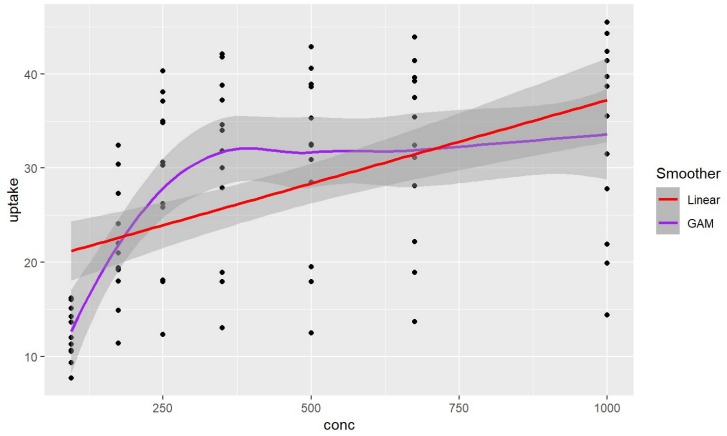
```
g <- ggplot(CO2, aes(x = conc, y = uptake))
g + geom_point()
```



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 = ".")
```

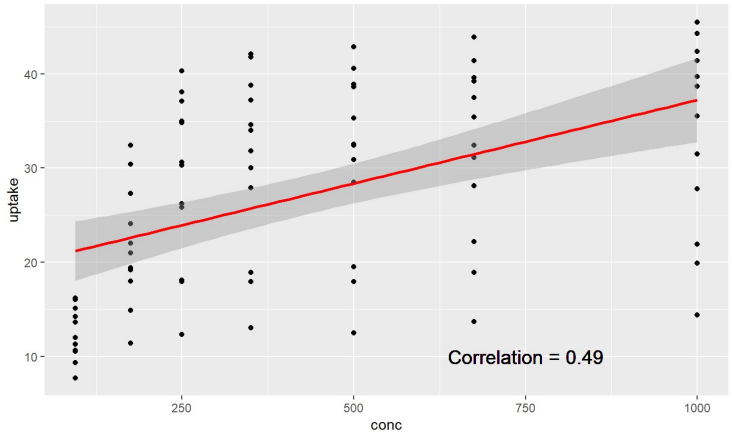
```
## [1] "Hi.What.Is.Going.On.?"
```

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

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

ggplot2 scatter plots with text

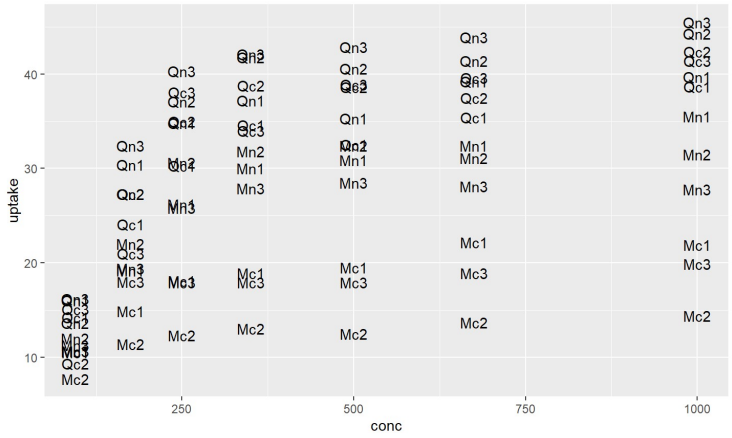
```
correlation <- cor(CO2$conc, CO2$uptake)
ggplot(CO2, aes(x = conc, y = uptake)) + geom_point() +
  geom_smooth(method = lm, col = "Red") +
  geom_text(x = 750, y = 10, size = 5,
            label = paste0("Correlation = ", round(correlation, 2)))
```



`ggplot2` scatter plots with text points

- Text for points with `geom_text`

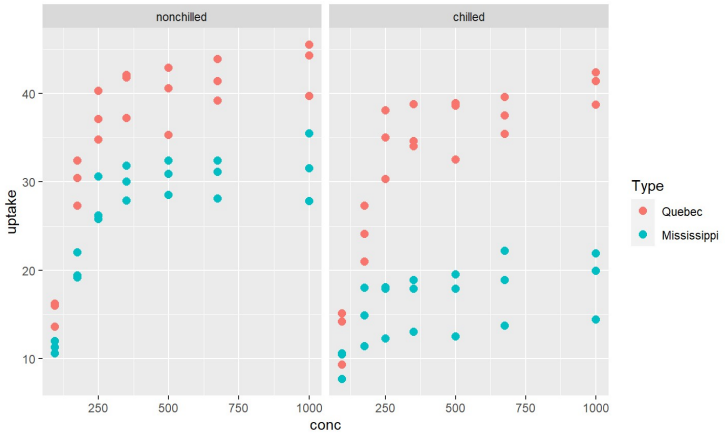
```
ggplot(CO2, aes(x = conc, y = uptake)) +
  geom_text(aes(label = Plant))
```



ggplot2 faceting

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

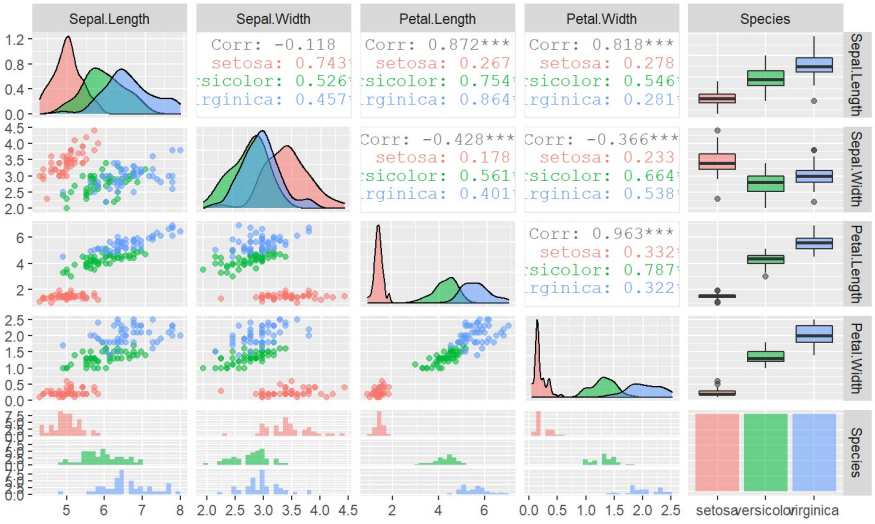
```
ggplot(CO2, aes(x = conc, y = uptake)) +
  geom_point(aes(color = Type), size = 2.5) +
  facet_wrap(~ Treatment)
```



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`

Activity

- **ggplot Activity** instructions available on web
- We'll send you to breakout rooms
- One TA or instructor in each room to help out
- Feel free to ask questions about anything you didn't understand as well!

Schedule

Day 2

- Logical Statements and Subsetting/Manipulating Data?
- Numerical and Graphical Summaries
- **Basic Analyses**

Basic Analysis (Linear Regression)

- With multiple quantitative varibles can look at many types of relationships
- Investigated linear relationship between two with correlation
- Scatter Plot gives useful visualization
- Consider data on voting preferences by county
    - Inspect relationship with pct with college degree and income

Basic Analysis (Linear Regression)

- With multiple quantitative varibles can look at many types of relationships
- Investigated linear relationship between two with correlation
- Scatter Plot gives useful visualization
- Consider data on voting preferences by county
    - Inspect relationship with pct with college degree and income

```
voting <- read_csv("datasets/counties.csv")
```

Basic Analysis (Linear Regression)

```
voting
```

```
## # A tibble: 3,141 x 20
##   region county state   msa  pmsa pop.density   pop pop.change age6574 age75
##   <chr>  <chr>  <chr> <dbl> <dbl>       <dbl> <dbl>      <dbl>   <dbl> <dbl>
## 1 South  Autau~ AL     5240    NA          61 34222      11.9     5.70  4.10
## 2 South  Baldw~ AL     5160    NA          67 98280      35.4     9.20  6
## 3 South  Barbo~ AL       NA    NA          29 25417       2       8.20  6.40
## 4 South  Bibb   AL       NA    NA          28 16576       9.20    6.70  6
## 5 South  Blount AL     1000    NA          62 39248      10.6     7.40  5.60
## # ... with 3,136 more rows, and 10 more variables: crime <dbl>, college <dbl>,
## #   income <dbl>, farm <dbl>, democrat <dbl>, republican <dbl>, Perot <dbl>,
## #   white <dbl>, black <dbl>, turnout <dbl>
```

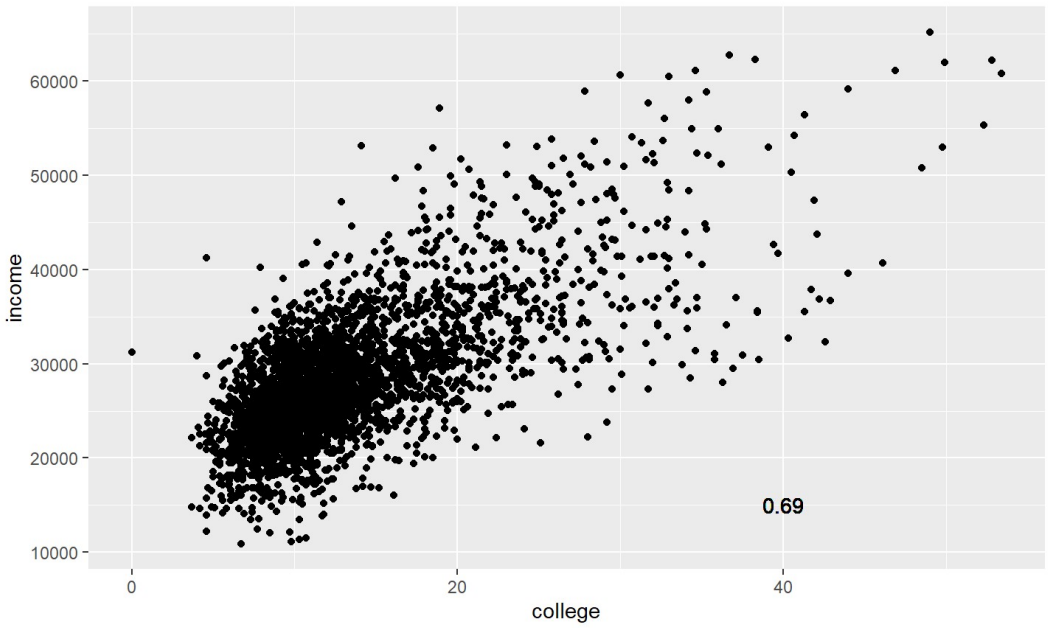
Basic Analysis (Linear Regression)

- Create a scatter plot, add correlation

```
votePlot <- ggplot(voting, aes(x = college, y = income))


votePlot +
  geom_point()+
  geom_text(x = 40, y = 15000,
            label = round(cor(voting$college, voting$income), 2))
```

Basic Analysis (Linear Regression)



Basic Analysis (Linear Regression)

- Often want to predict y variable using a value of x
- **Simple Linear Regression** allows for this
- Linear model for predicting income based on college:
- cty_income = int + slope * (% college in county) + random_error
- Generally, response (Y) = function of predictors (features, X's) + error

$$Y_i = \beta_0 + \beta_1 x_{1i} + \ldots + \beta_p x_{pi} + E_i$$

Basic Analysis (Linear Regression)

- Fit simple linear regression model using `lm()`

```
lm(income ~ college, data = voting)
```

```
##
## Call:
## lm(formula = income ~ college, data = voting)
##
## Coefficients:
## (Intercept)      college
##     18305.5        752.6
```

Basic Analysis (Linear Regression)

- Save as object, check attributes

```
fit <- lm(income ~ college, data = voting)
attributes(fit)
```

```
## $names
##  [1] "coefficients"  "residuals"     "effects"       "rank"
##  [5] "fitted.values" "assign"        "qr"            "df.residual"
##  [9] "xlevels"       "call"          "terms"         "model"
##
## $class
## [1] "lm"
```

Basic Analysis (Linear Regression)

- Three main ways to get at *many* of these attributes

```
fit[[1]]
```

```
## (Intercept)     college
##  18305.5294    752.6481
```

```
coefficients(fit)
```

```
## (Intercept)     college
##  18305.5294    752.6481
```

```
fit$coefficients
```

```
## (Intercept)     college
##  18305.5294    752.6481
```

Basic Analysis (Linear Regression)

- Three main ways to get at *many* of these attributes

```
fit[[2]]
residuals(fit)
fit$residuals
```

Basic Analysis (Linear Regression)

- Three main ways to get at *many* of these attributes

```
#no generic function for some things
rank(fit)
```

```
## Error in rank(fit): unimplemented type 'list' in 'greater'
```

```
y(fit)
```

```
## Error in y(fit): could not find function "y"
```

```
fit$rank
```

```
## [1] 2
```

Basic Analysis (Linear Regression)

- Statistical analysis found using `anova()` or `summary()`

```
#ANOVA table (F tests)
anova(fit)
```

```
## Analysis of Variance Table
##
## Response: income
##             Df     Sum Sq    Mean Sq F value    Pr(>F)
## college      1 7.6911e+10 7.6911e+10  2865.4 < 2.2e-16 ***
## Residuals 3139 8.4254e+10 2.6841e+07
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Basic Analysis (Linear Regression)

- Statistical analysis found using `anova()` or `summary()`

```
#coefficient type III tests
summary(fit)
```

```
##
## Call:
## lm(formula = income ~ college, data = voting)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -18062.3  -3146.8   -288.8   2890.7  24569.4
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 18305.53     211.29   86.64   <2e-16 ***
## college       752.65      14.06   53.53   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5181 on 3139 degrees of freedom
## Multiple R-squared:  0.4772, Adjusted R-squared:  0.4771
## F-statistic:  2865 on 1 and 3139 DF,  p-value: < 2.2e-16
```

Basic Analysis (Linear Regression)

- Diagnostic plots easily found using `plot()`
- Run this code in console

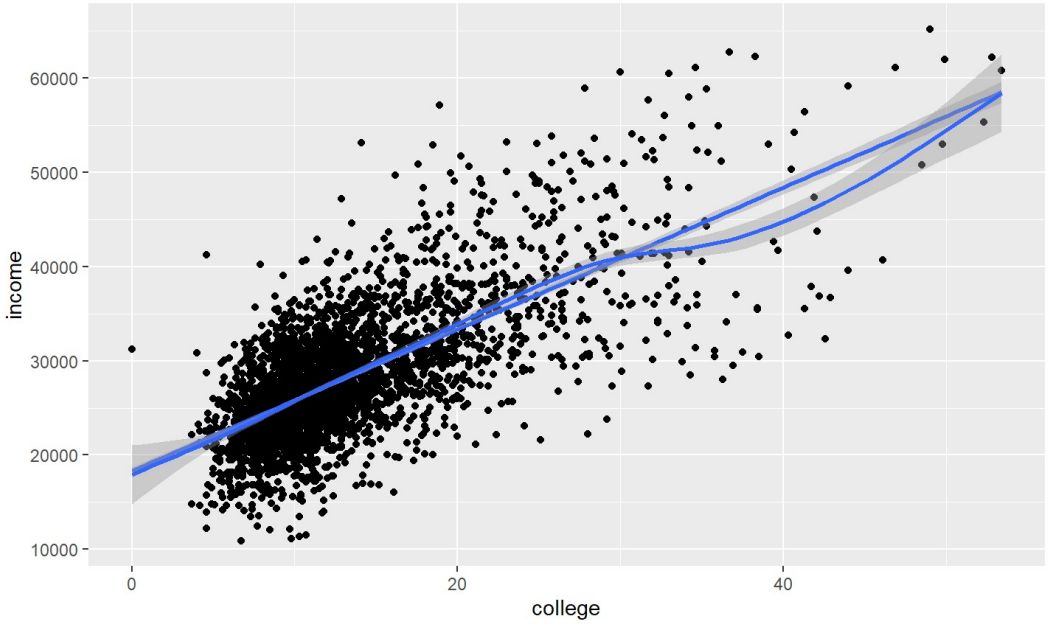
```
plot(fit)
```

Basic Analysis (Linear Regression)

- Add fit to our scatter plot
- Compare to a smoothed fit

```
votePlot +
  geom_point() +
  geom_smooth(method = "lm") +
  geom_smooth()
```

Basic Analysis (Linear Regression)



Basic Analysis (Linear Regression)

- Now can predict y for a given x
- Check `help(predict)`, particularly `predict.lm()`

```
predict(fit, newdata = data.frame(college = c(40, 10)))
```

```
##        1        2
## 48411.45 25832.01
```

Basic Analysis (Linear Regression)

- Get SE for prediction

```
predict(fit, newdata = data.frame(college = c(40, 10)), se.fit = TRUE)
```

```
## $fit
##        1        2
## 48411.45 25832.01
##
## $se.fit
##        1        2
## 383.7192 104.8104
##
## $df
## [1] 3139
##
## $residual.scale
## [1] 5180.841
```

Basic Analysis (Linear Regression)

- Get confidence interval for mean response

```
predict(fit, newdata = data.frame(college = c(40, 10)),
        se.fit = TRUE, interval = "confidence")
```

```
## $fit
##        fit      lwr      upr
## 1 48411.45 47659.09 49163.82
## 2 25832.01 25626.51 26037.51
##
## $se.fit
##        1        2
## 383.7192 104.8104
##
## $df
## [1] 3139
##
## $residual.scale
## [1] 5180.841
```

Basic Analysis (Linear Regression)

- Get prediction interval for new response

```
predict(fit, newdata = data.frame(college = c(40, 10)),
        se.fit = TRUE, interval = "prediction")
```

```
## $fit
##        fit      lwr      upr
## 1 48411.45 38225.45 58597.45
## 2 25832.01 15671.75 35992.27
##
## $se.fit
##        1        2
## 383.7192 104.8104
##
## $df
## [1] 3139
##
## $residual.scale
## [1] 5180.841
```

Basic Analysis (Linear Regression)

- Multiple Linear Regression (more than one x)
- In `lm()`
  - add “main effect” terms with `+ name`
  - interactions with `+ name1:name2`
  - all possible combinations with `+ name1*name2`

```
fit2<-lm(income ~ college + Perot, data = voting)
```

Basic Analysis (Linear Regression)

- Multiple Linear Regression (more than one x)

```
anova(fit2)
```

```
## Analysis of Variance Table
##
## Response: income
##             Df     Sum Sq    Mean Sq  F value    Pr(>F)
## college      1 7.4771e+10 7.4771e+10 3013.718 < 2.2e-16 ***
## Perot        1 2.0739e+09 2.0739e+09   83.591 < 2.2e-16 ***
## Residuals 3111 7.7185e+10 2.4810e+07
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Basic Analysis (Linear Regression)

```
summary(fit2)
```

```
##
## Call:
## lm(formula = income ~ college + Perot, data = voting)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -17192  -3205   -234   2909  21077
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 16167.90     310.46  52.077   <2e-16 ***
## college       728.46      13.71  53.136   <2e-16 ***
## Perot         119.73      13.10   9.143   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4981 on 3111 degrees of freedom
##   (27 observations deleted due to missingness)
## Multiple R-squared:  0.4989, Adjusted R-squared:  0.4986
## F-statistic:  1549 on 2 and 3111 DF,  p-value: < 2.2e-16
```

Basic Analysis (Linear Regression)

- Access elements of object just as before

```
coef(fit2)
```

```
## (Intercept)     college       Perot
##  16167.9032    728.4587    119.7262
```

```
fit2$rank
```

```
## [1] 3
```

Basic Analysis (Linear Regression)

- Basic diagnostic plots (run in console)

```
plot(fit2)
```

Basic Analysis (Linear Regression)

- Predict new for new values

```
predict(fit2, newdata = data.frame(college = c(40, 30), Perot = c(20, 25)))
```

```
##        1        2
## 47700.77 41014.82
```

##Recap!

- Basic linear regression
  - Use `lm()`
- Statistical analysis
  - `anova()`
  - `summary()`
- Predict using `predict()`
- Add fits to scatter plot with `geom_smooth(method = "lm")`

Activity

- **Basic Analysis Activity** instructions available on web
- We'll send you to breakout rooms
- One TA or instructor in each room to help out
- Feel free to ask questions about anything you didn't understand as well!
- Thanks for coming!!