Outline

1. Background
2. Wrangling, Plotting, and Modeling
3. Essential Functionality
4. Advanced Functionality
5. Additional Resources

Part I: Background

Why use R: Accessibility

-
  - A script documents all your work, from data access to reporting, and can instantly be re-run at any time
-
  - As an open-source project, you can use R free of charge: no worries about subscription fees, license managers, or user limits.
-
  - All of the standard data analysis tools are built right into the R language (and many others are available via “packages”)
-
  - One of the design principles of R was that visualization of data through charts and graphs is an essential part of the data analysis process, so it has excellent tools for creating graphics

Why use R: Community

-
  - Leading academics and researches from around the world use R to develop the latest methods in statistics, machine learning, and predictive modeling
-
  - There's a wealth of community resources for R available on the Web, for help in just about every domain
-
  - Available Linux, Mac, and Windows
-
  - R users come from myriad academic departments and industries

Part II: Wrangling, Plotting, and Modeling

Focus on data frames (and tidyverse)

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

df <- nycflights13::flights
```

Explore your data: Descriptive statistics

```
df_ss <- select(df, dep_delay, arr_delay, air_time, distance,
carrier)
psych::describe(df_ss)
```

```
          vars      n    mean     sd median trimmed    mad min  max
range
dep_delay    1 328521   12.64  40.21     -2    3.32   5.93 -43 1301
1344
arr_delay    2 327346    6.90  44.63     -5   -1.03  20.76 -86 1272
1358
air_time     3 327346  150.69  93.69    129  140.03  75.61  20  695
675
distance     4 336776 1039.91 733.23    872  955.27 569.32  17 4983
4966
carrier*     5 336776    9.00   0.00      9    9.00   0.00   9    9
0
          skew kurtosis   se
dep_delay 4.80    43.95 0.07
arr_delay 3.72    29.23 0.08
air_time  1.07     0.86 0.16
distance  1.13     1.19 1.26
carrier*   NaN      NaN 0.00
```

What's going on here?

```
df %>%
    select(dep_delay, arr_delay, air_time, distance, carrier) %>%
    group_by(carrier) %>%
    summarize(dep_delay_mean = mean(dep_delay, na.rm = T))
```

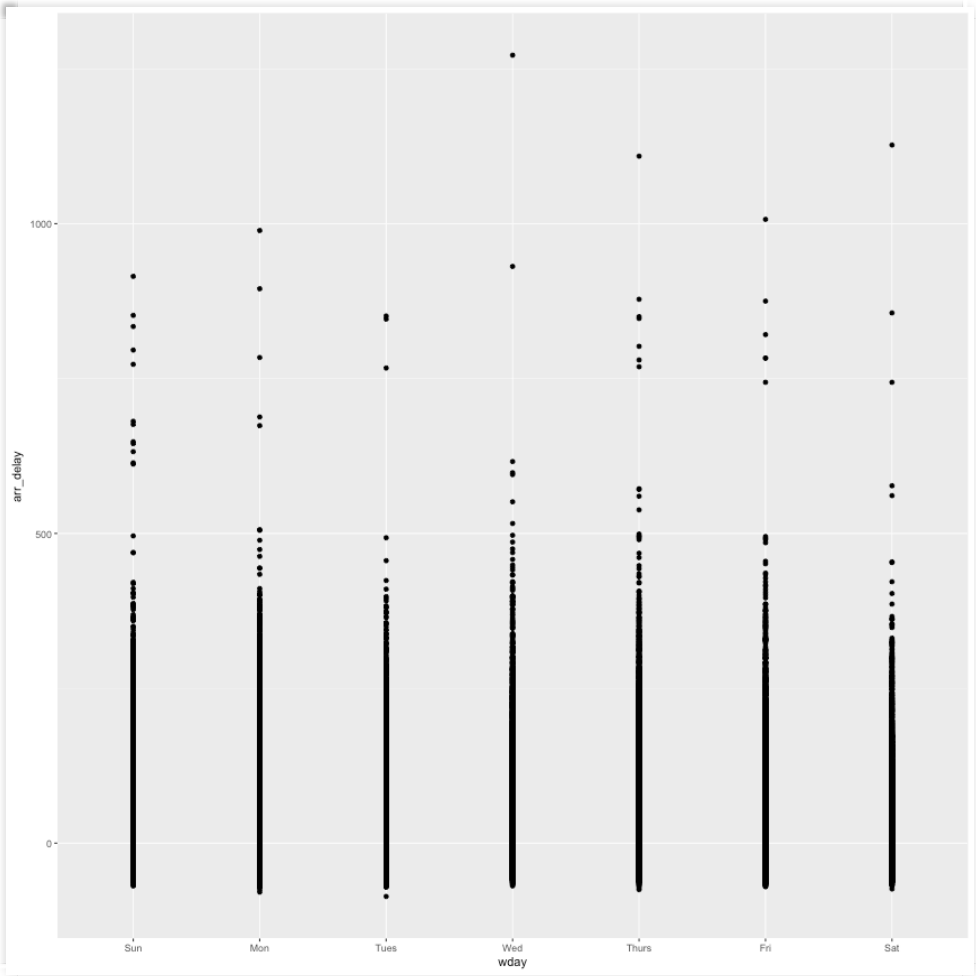
```
df %>%
    select(dep_delay, arr_delay, air_time, distance, carrier) %>%
    group_by(carrier) %>%
    summarize(dep_delay_mean = mean(dep_delay, na.rm = T))
```

```
# A tibble: 16 × 2
   carrier dep_delay_mean
     <chr>          <dbl>
1       9E      16.725769
2       AA       8.586016
3       AS       5.804775
4       B6      13.022522
5       DL       9.264505
6       EV      19.955390
7       F9      20.215543
8       FL      18.726075
9       HA       4.900585
10      MQ      10.552041
11      OO      12.586207
12      UA      12.106073
13      US       3.782418
14      VX      12.869421
15      WN      17.711744
16      YV      18.996330
```

Explore your data: Plotting Distributions

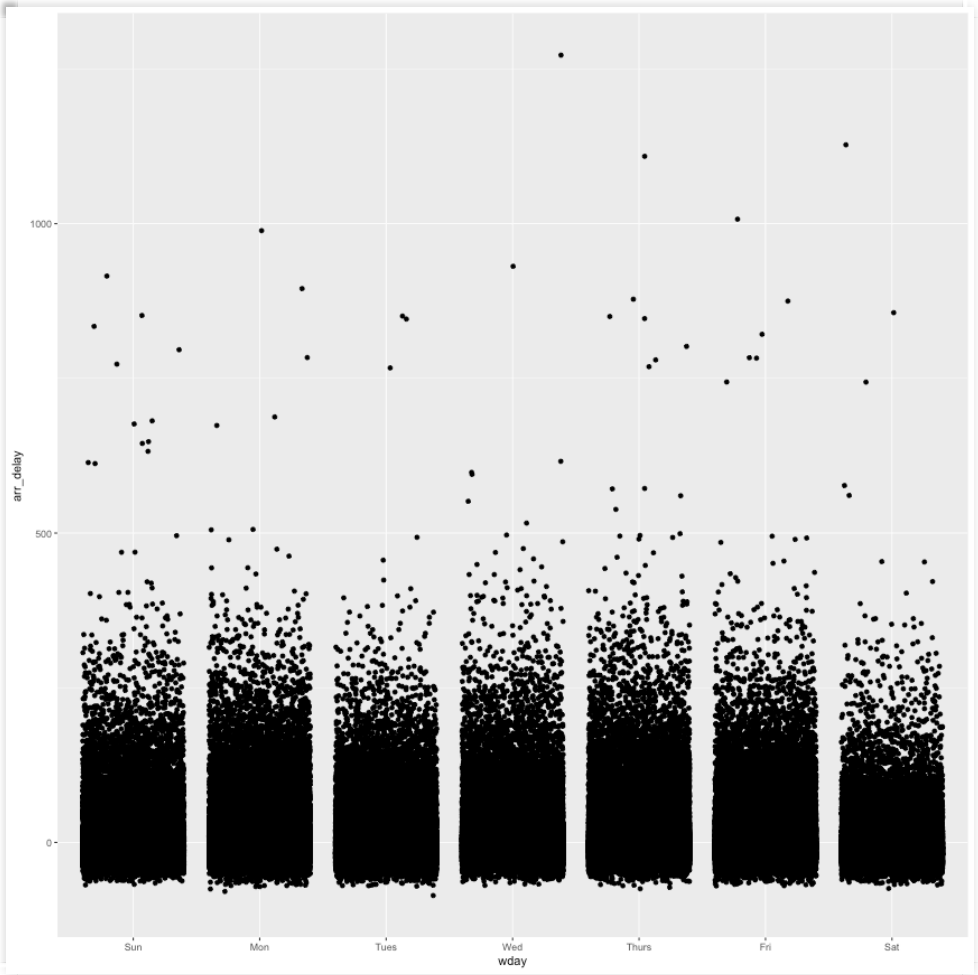
```
df$wday <- lubridate::wday(df$time_hour, label = T)

ggplot(df, aes(x = wday, y = arr_delay)) +
    geom_point()
```



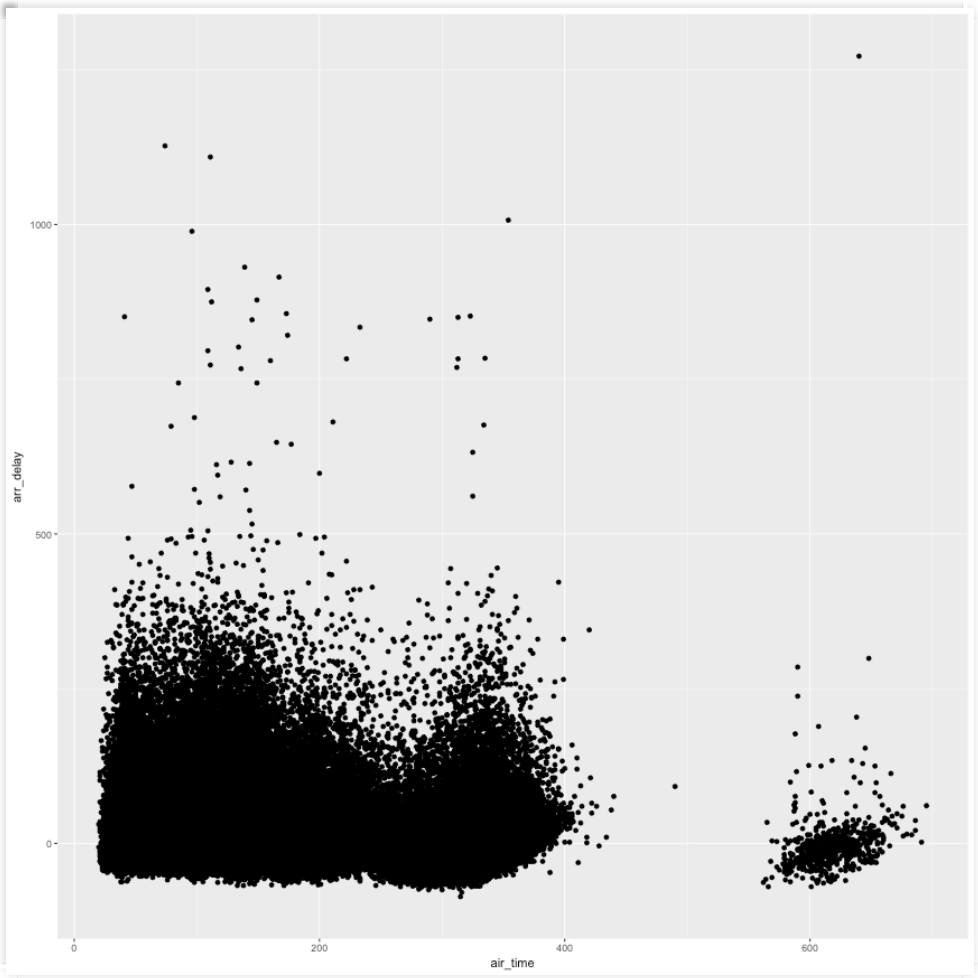
Explore your data: Plotting Distributions (With Some Random Noise)

```
library(ggplot2)
ggplot(df, aes(x = wday, y = arr_delay)) +
    geom_jitter()
```



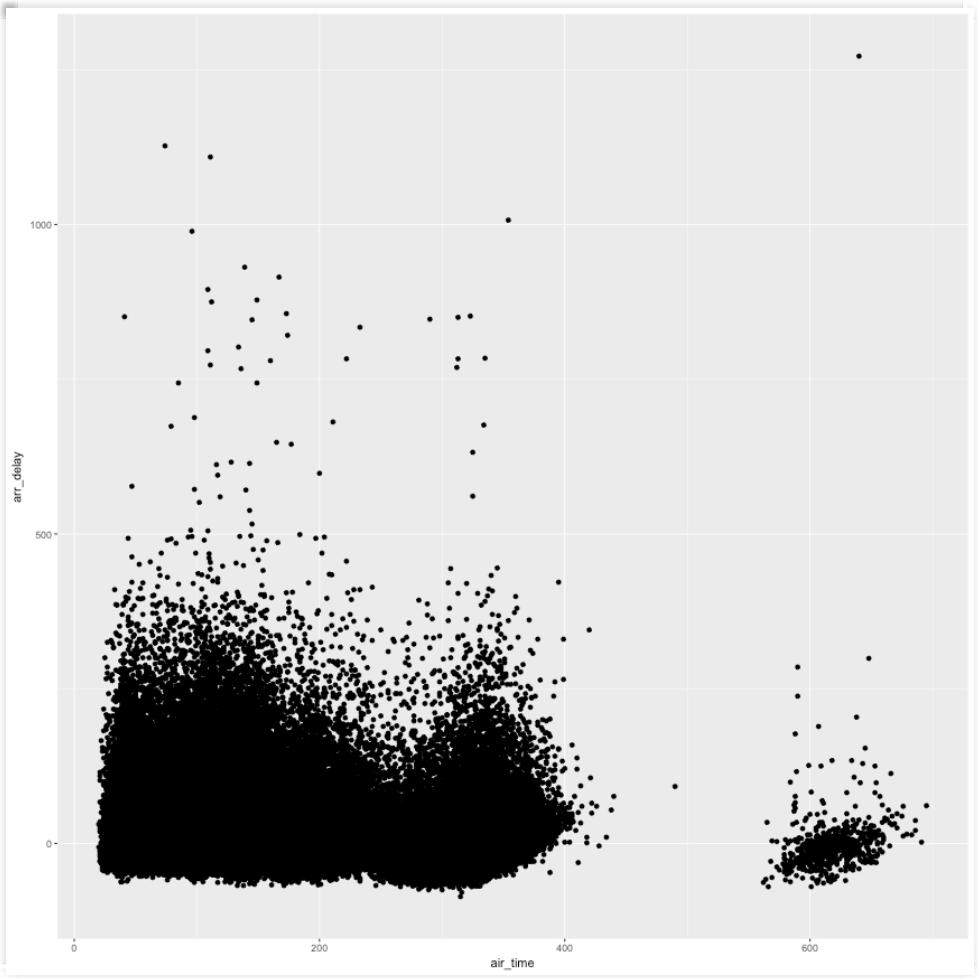
Explore your data: Plotting relationships

```
ggplot(df, aes(x = air_time, y = arr_delay)) +
    geom_point()
```



Explore your data: Plotting relationships

```
ggplot(df, aes(x = air_time, y = arr_delay)) +
    geom_point()
```



Explore your data: Plotting relationships

```
ggplot(df_ss, aes(x = air_time, y = arr_delay, color = carrier)) +
    geom_point()
```

Explore your data: Manipulate data (for plots)

```
to_plot <- df %>%
    group_by(carrier) %>%
    summarize(dep_delay_mean = mean(dep_delay, na.rm = T),
              n = n()) %>%
    filter(n > 10000) %>%
    arrange(desc(dep_delay_mean))
```

- EV: Express Jet
- WN: Southwest Airlines
- AA: American Airlines
- US: US Airways

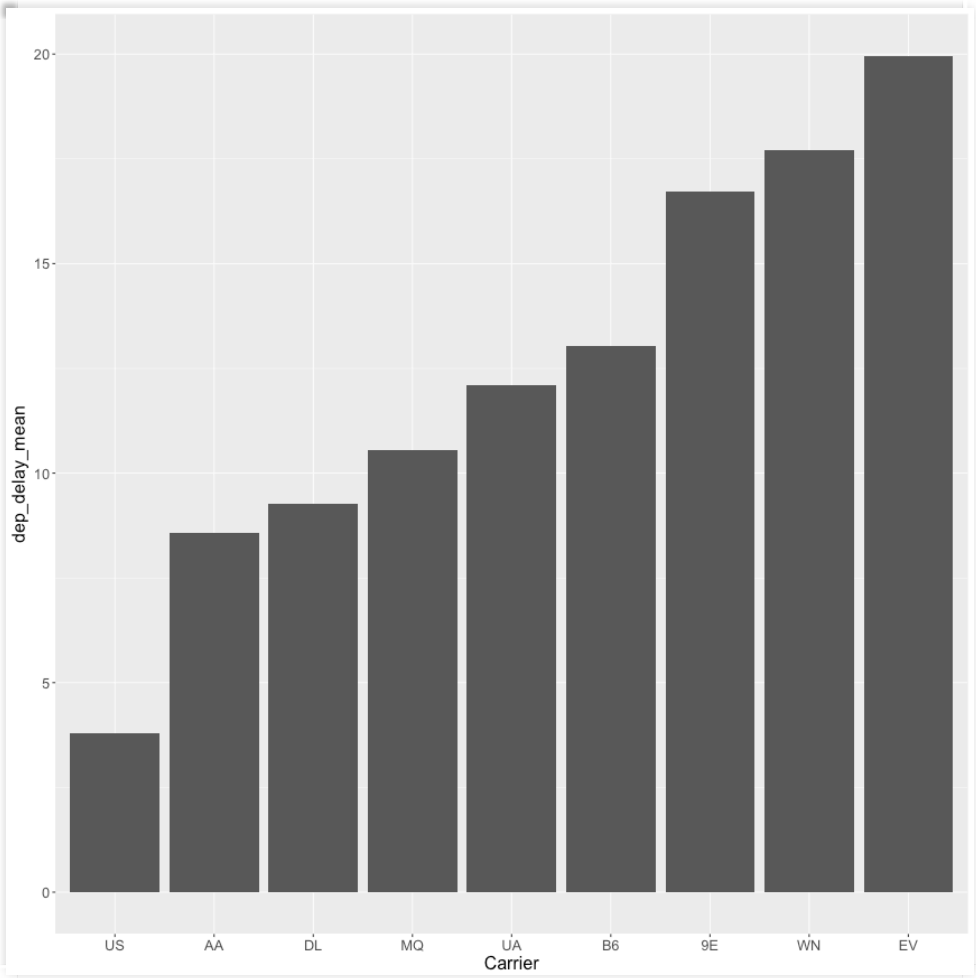
Explore your data: Manipulate data (for plots)

```
to_plot
```

```
# A tibble: 9 × 3
  carrier dep_delay_mean     n
    <chr>          <dbl> <int>
1      EV      19.955390 54173
2      WN      17.711744 12275
3      9E      16.725769 18460
4      B6      13.022522 54635
5      UA      12.106073 58665
6      MQ      10.552041 26397
7      DL       9.264505 48110
8      AA       8.586016 32729
9      US       3.782418 20536
```

Explore your data: Plotting Means

```
ggplot(to_plot, aes(x = reorder(carrier, dep_delay_mean), y =
dep_delay_mean)) +
    geom_col() +
    theme(text = element_text(size = 16)) +
    xlab("Carrier")
```



Model your data: Linear models

```
m1 <- lm(arr_delay ~ air_time, data = df)
arm::display(m1)
```

```
lm(formula = arr_delay ~ air_time, data = df)
            coef.est coef.se
(Intercept)  9.43     0.15
air_time    -0.02     0.00
---
n = 327346, k = 2
residual sd = 44.61, R-Squared = 0.00
```

Model your data: Linear models

```
m2 <- lm(arr_delay ~ air_time + distance, data = df)
arm::display(m2)
```

```
lm(formula = arr_delay ~ air_time + distance, data = df)
            coef.est coef.se
(Intercept) -1.46     0.17
air_time     0.67     0.01
distance    -0.09     0.00
---
n = 327346, k = 3
residual sd = 43.73, R-Squared = 0.04
```

Model your data: Linear models

```
m3 <- lm(arr_delay ~ air_time*distance, data = df)
arm::display(m3)
```

```
lm(formula = arr_delay ~ air_time * distance, data = df)
                  coef.est coef.se
(Intercept)       -0.07     0.24
air_time           0.66     0.01
distance          -0.09     0.00
air_time:distance  0.00     0.00
---
n = 327346, k = 4
residual sd = 43.72, R-Squared = 0.04
```

Model your data: Linear models

```
m3 <- lm(arr_delay ~ air_time*distance + carrier, data = df)
arm::display(m3)
```

Part III: Essential Functionality

Vectors

```
my_vector <- c(1:10)
```

```
my_vector
```

```
 [1]  1  2  3  4  5  6  7  8  9 10
```

```
mean(my_vector)
```

```
[1] 5.5
```

Base R functions

```
- ? # this is to find out what a function does
- str() # this is to find out the 'structure' of anything
- View() # this allows you to view a data frame (think spreadsheet)
or matrix
- class() # this tells you what kind of object this is
```

```
my_data[1, ] # just the first row of data frame
my_data[, 1] # just the first column of data frame
head(my_data) # first six rows of data frame
tail(my_data) # last six rows of data frame
```

Loading data (CSV)

```
setwd("~/documents") # this sets the working directory
my_data <- readr::read_csv("r_introduction_data.csv") # loads a CSV
and saves it to `my_data`
my_data
```

```
# A tibble: 212 × 57
     Int    StudentID Grade   Age Gender ClassTeacher SciTeacher
PrevIQWST
   <int>        <chr> <int> <int>  <int>        <int>      <int>
<int>
1      1 A.J. Miranda     1    11      0            5          2
NA
2      1      Abby B.     0    10      1            0          0
0
3      0       Abby E     0    10      0            1          0
0
4      0       Abi N.     1    11      1            4          2
NA
5      1   Abigail D.     1    11      1            7          3
NA
6      0       Adam F     1    12      0            5          2
NA
7     NA      Adam L.     1    11      0            4          2
NA
8      0      Adam T.     0     9      0            1          0
0
9      1   Addison D.     1    11      1            6          3
NA
10     1    Adelle S.     1    11      1            4          2
NA
```

```
# ... with 202 more rows, and 49 more variables: PreEff1 <int>,
#   PreEff2 <int>, PreEff3 <int>, PreEff4 <int>, PreEff5 <int>,
#   PreInt1 <int>, PreInt2Rev <int>, PreInt3 <int>, PreInt4 <int>,
#   PreInt5 <int>, PreVal1 <int>, PreVal2 <int>, PreVal3 <int>,
#   PreVal4 <int>, PreVal5 <int>, PreVal6 <int>, PreVal7 <int>,
#   PreVal8 <int>, PreEff_Ave <dbl>, PreInt_Ave <dbl>, PreVal_Ave
<dbl>,
#   PostEff1 <int>, PostEff2 <int>, PostEff3 <int>, PostEff4 <int>,
#   PostEff5 <int>, PostInt1 <int>, PostInt2 <int>, PostInt3 <int>,
#   PostInt4 <int>, PostInt5 <int>, PostVal1 <int>, PostVal2 <int>,
#   PostVal3 <int>, PostVal4 <int>, PostVal5 <int>, PostVal6 <int>,
#   PostVal7 <int>, PostVal8 <int>, PostMaintInt1 <int>,
#   PostMaintInt2 <int>, PostMaintInt3 <int>, PostMaintInt4 <int>,
#   PostEff_Ave <dbl>, PostInt_Ave <dbl>, PostVal_Ave <dbl>,
#   PostMaintInt_Ave <dbl>, PreAchievement <int>, PostAchievement
<int>
```

Loading data (Tab-delimited, Excel, and SPSS)

```
read.delim("filename.txt") # Tab-delimited
readxl::read_excel("filename.xlsx") # Excel
haven::read_sav("filename.sav") # SPSS
```

Calculating summary statistics

```
my_data %>% count(SciTeacher) # counts frequencies and creates a
table
```

```
# A tibble: 4 × 2
  SciTeacher     n
       <int> <int>
1          0    53
2          1    55
3          2    53
4          3    51
```

```
my_data_ss <- select(my_data, contains("_Ave")) # this selects any
variables containing "Ave"
summary(my_data_ss) # creates summary statistics for continuous
variables
```

```
   PreEff_Ave      PreInt_Ave      PreVal_Ave     PostEff_Ave
 Min.   :2.200   Min.   :1.000   Min.   :1.875   Min.   :1.200
 1st Qu.:5.000   1st Qu.:4.800   1st Qu.:5.125   1st Qu.:4.800
 Median :5.600   Median :6.200   Median :6.125   Median :5.600
 Mean   :5.466   Mean   :5.739   Mean   :5.780   Mean   :5.354
 3rd Qu.:6.200   3rd Qu.:6.800   3rd Qu.:6.625   3rd Qu.:6.200
 Max.   :7.000   Max.   :7.000   Max.   :7.000   Max.   :7.000
                                                 NA's   :16
  PostInt_Ave     PostVal_Ave    PostMaintInt_Ave
 Min.   :1.000   Min.   :1.500   Min.   :1.00
 1st Qu.:4.400   1st Qu.:4.875   1st Qu.:3.50
 Median :5.800   Median :6.250   Median :5.00
 Mean   :5.394   Mean   :5.691   Mean   :4.72
 3rd Qu.:6.600   3rd Qu.:6.750   3rd Qu.:6.00
 Max.   :7.000   Max.   :7.000   Max.   :7.00
```

```
NA's   :16      NA's   :16      NA's   :18
```

dplyr for data manipulation

```
dplyr::select(my_data, PreAchievement, PostAchievement) # Select
only certain columns

dplyr::filter(my_data, PreAchievement >= 3) # select only certain
rows

dplyr::arrange(my_data, PostAchievement) # arrange data by a
variable
```

```
my_data %>%
    filter(PreAchievement >= 3) %>%
    group_by(SciTeacher) %>%
    summarize(SciTeacher_mean = mean(SciTeacher))
```

tidyr for reshaping and tidying data

```
stocks <- data_frame(
  time = as.Date('2009-01-01') + 0:9,
  X = rnorm(10, 0, 1),
  Y = rnorm(10, 0, 2),
  Z = rnorm(10, 0, 4)
)

stocks
```

```
# A tibble: 10 × 4
         time           X          Y          Z
       <date>       <dbl>      <dbl>      <dbl>
1  2009-01-01  0.55486073  2.2439625  0.6818610
2  2009-01-02 -1.20030507 -3.1660577  2.3646694
3  2009-01-03 -0.44413959  1.7341190  0.6827402
4  2009-01-04 -0.18554758  1.7967152 -0.3609197
5  2009-01-05  0.96336125  1.3014036  6.3654195
6  2009-01-06 -0.69798296  2.2115162 -4.6764298
7  2009-01-07 -0.50668007  0.7689348  1.0283452
8  2009-01-08  1.96965999 -4.6243631  4.8479906
9  2009-01-09  0.01062281  3.0295156  2.9525947
10 2009-01-10  0.99066158 -0.0639218 -3.2256691
```

gather() for reshaping from "wide" to "long" format

```
gather(stocks, stock, price, -time)
```

```
# A tibble: 30 × 3
         time stock       price
       <date> <chr>       <dbl>
1  2009-01-01     X  0.55486073
2  2009-01-02     X -1.20030507
3  2009-01-03     X -0.44413959
4  2009-01-04     X -0.18554758
5  2009-01-05     X  0.96336125
6  2009-01-06     X -0.69798296
7  2009-01-07     X -0.50668007
8  2009-01-08     X  1.96965999
9  2009-01-09     X  0.01062281
10 2009-01-10     X  0.99066158
# ... with 20 more rows
```

spread() for reshaping from "long"" to "wide" format

```
stocks_long <- gather(stocks, stock, price, -time)
spread(stocks_long, stock, price)
```

```
# A tibble: 10 × 4
         time           X          Y          Z
*      <date>       <dbl>      <dbl>      <dbl>
1  2009-01-01  0.55486073  2.2439625  0.6818610
2  2009-01-02 -1.20030507 -3.1660577  2.3646694
3  2009-01-03 -0.44413959  1.7341190  0.6827402
4  2009-01-04 -0.18554758  1.7967152 -0.3609197
5  2009-01-05  0.96336125  1.3014036  6.3654195
6  2009-01-06 -0.69798296  2.2115162 -4.6764298
7  2009-01-07 -0.50668007  0.7689348  1.0283452
8  2009-01-08  1.96965999 -4.6243631  4.8479906
9  2009-01-09  0.01062281  3.0295156  2.9525947
10 2009-01-10  0.99066158 -0.0639218 -3.2256691
```

Part IV: Advanced functionality

Packages

- Linear mixed models modeling: `lme4`, `nlme`
- Latent variable modeling: `lavaan`, `OpenMx`
- Social Network Analysis: `igraph`, `statnet`
- Text analysis: `quanteda`, `tidytext`

Linear mixed effects (multi-level) models

```
library(lme4)

model_1 <- lmer(engagement ~ challenge + percomp + (1 |
program_ID), data = df)

summary(model1)
```

Structural equation modeling

```
library(lavaan)

model <- '
  # measurement model
    ind60 =~ x1 + x2 + x3
    dem60 =~ y1 + y2 + y3 + y4
    dem65 =~ y5 + y6 + y7 + y8
  # regressions
    dem60 ~ ind60
    dem65 ~ ind60 + dem60
  # residual correlations
    y1 ~~ y5
    y2 ~~ y4 + y6
    y3 ~~ y7
    y4 ~~ y8
    y6 ~~ y8
'

fit <- sem(model, data = PoliticalDemocracy)
summary(fit, standardized = TRUE, fit.measures = T)
```

Social network analysis

```
library(igraph)
g <- graph.edgelist(edgematrix) # loads two column matrix with ties
V(g)$size = log(degree(g)) # changes size of vertices
V(g)$label <- NA # removes vertex names
V(g)$color[year_1_cohort_bool] <- "blue" # changes vertex color
based on logical index
l <- layout.fruchterman.reingold(g) # selects layout
E(g)$weight <- 1 # weights each edge equal to 1
g <- simplify(g, edge.attr.comb = list(weight = "sum")) # sums
edgeweights for equivalent ties

plot(g, layout = l,
     edge.width = E(g)weight)
```

Text analysis

```
library(quanteda)
my_corpus <- corpus(inaugTexts)
summary(my_corpus, n = 3)

my_dfm <- dfm(my_corpus, ignoredFeatures = stopwords("english"))
```

What are some other things we can do?

```
library(matchit) # propensity score matching
library(mgcv) # generalized additive models
library(modelr) # helper functions for modeling
library(rvest) # web scraping
library(caret) # machine learning framework
```

Part V: Additional resources

Rmarkdown & knitr

-
-
-
-

Shiny

-
- http://shinyapps.io
-
- Example: SETHs

Packages

-
-
-
- Example: prcr

Additional resources

-
  - Quick-R
  - R Studio Cheat Sheets
  - Stack Overflow
  - #rstats
  - RBloggers
-
  - Gelman & Hill (2006)
  - Grolemund & Wickham (2014)
  - Wickham & Grolemund (2017)

Thank you!

- Email: **jrosen@msu.edu**
- Web: **http://jmichaelrosenberg.com**
- Twitter: @jrosenberg6432