R for Data Analysis and Visualization

ECON 396 (Fall 2017)
TR 10:30-11:45, DURP Computer Lab (first floor Saunders)

Jonathan Page

2017-10-29

Contents

Syllabus

Office Hours

Monday 2-3 PM and Tuesday 3-4 PM, or by appointment, Saunders 509, jrpage at hawaii dot edu.

Student Learning Objectives

1. To be familiar with standard techniques for visualizing data, including heat maps, contour plots, etc.
2. To be able to transform raw data into formats suitable for analysis
3. To be able to perform basic exploratory analysis
4. To be able to create data visualizations in R

There is no prerequisite for this course.

Resources

Required

Introductory Statistics with Randomization and Simulation: Available as a free PDF (https://www.openintro.org/stat/textbook.php?stat_book=isrs) or for $8.49 on Amazon.

Recommended:

R Graphics Cookbook

RStudio Cheat Sheets

Course Requirements

Grades for this course will be based on weekly assignments (30%), project assignments (30%), the project proposal (5%), the final project deliverable (20%), and final project presentation participation (15%).

Weekly assignments (30%)

Weekly assignments are short R excercises. Each exercise should take no longer than 15 minutes. You will typically be given time to complete the exercise in class the day the assignment is given. The assign-

ment will be in the form of R Markdown file (*.Rmd). You will submit the completed assignments via classroom.google.com by the following class period.

Individual Project

Project assignments (30%)

Each week, leading up to the project proposal, you will be given an assignment that is designed to provide you with an organized workflow for approaching new data science projects. Project assignments are submitted via classroom.google.com, with the exception of the two presentations

Project proposal presentation (5%)

This presentation should be less than 2 minutes. You simply need to communicate the core question your project seeks to answer and the dataset(s) you will be using to answer this question.

Final project (20%)

The final project will be an R Markdown document which communicates your project question, the data you used, and your results. You will need to deliver both your R Markdown file and any necessary data for running the file.

Final project presentation participation (15%)

Your final project participation grade is based on a combination of your own presentation and the feedback you provide to your classmates.

Schedule

The following schedule is tentative and subject to change. Typically, the Tuesday class will consist of the week's R lecture. Depending on how quickly we get through the material, you will have time to work on your assignment that will be due before the following class period. On Thursdays, we will discuss a relevant topic, but you should have time to work on your project assignment for the week. That assignment will generally be due before the following class period, except for the last several weeks when you are completing your final project.

Week 1

- **R** Intro to R and RStudio; Histograms, scatterplots, summary statistics
  - **Data** R Sample Datasets
- **Topic** Data sources overview
- **Project Assignment** Indentify interesting datasets (include links to datasets) and questions

Week 2

- **R** read_csv, dplyr basics, heatmaps, hexbins
  - **Data** ACS PUMS *[CSV]*
- **Topic** Anscombe's Quartet
- **Project Assignment** Choose question and dataset (with link to your source) for your project

Week 3

- **R** ggplot facets, bubble plots, transparency
  - **Data** Hawaii Tourism Authority *[Excel]*
- **Topic** Probability
- **Project Assignment** Write description of your question

Week 4

- **R** geom_smooth, abline, vline, hline
  - **Data** State of Hawaii Department of Business, Economic Development (DBEDT) *[Excel]*
- **Topic** Distributions
- **Project Assignment** Write description of your dataset(s)

Week 5

- **R** ggplot2 Extensions and Scatterplot Matrices (GGally)
  - **Data** ACS Immigration *[CSV]*
- **Topic** JunkCharts Trifecta Checkup
- **Project Assignment** Create 2 descriptive plots of your datasets(s)

Week 6

- **R** Boxplots, violin plots
  - **Data** SSA *[Excel]*
- **Topic** Intro to Inference
- **Project Assignment** Write a description of the data cleaning required for your project

Week 7

- **R** Spatial Visualizations with geom_spoke, gganimate, and GGally::glyphs
  - **Data** NOAA Wind *[netCDF]*
- **Topic** Confidence Interval
- **Project Assignment** Write a description of your planned approach

Week 8

- **R** geom_area, geom_ribbon
  - **Data** BLS American Time Use Survey (ATUS) *[TSV]*
- **Topic** Project Proposal Description
- **Project Assignment** Work on project proposal presentation

Week 9

- **R** jitter, rug, aesthetics
  - **Data** PSID *[SPS, TXT (Fixed-Width)]*
- **Topic** Present project proposal (<2 Minutes)

Week 10

- **R** Themes, Labels, and Colors
  - **Data** Zillow Real Estate Data *[CSV]*
- **Topic** Inference for Numerical Data
- **Project Assignment** Work on final project

Week 11

- **R** Polar Coordinates
  - **Data** University of Michigan - Survey of Consumers *[CSV]*
- **Topic** Inference for Categorical Data
- **Project Assignment** Work on final project (cont.)

Week 12

- **R** Text Analysis (Natural Language Processing)
- **Topic** Linear Regression
- **Project Assignment** Work on final project (cont.)

Week 13

- **R** networks, geomnet extension
- **Topic** Multiple Regression
- **Project Assignment** Work on final project (cont.)

Week 14

- **R** git and GitHub for R
- **Topic** Putting your work online
- **Project Assignment** Work on final project (cont.)

Week 15

- Final Project presentations

Other Resources

There are many useful resources you should be aware of while going through this course. I will attempt to keep this list updated as I become aware of more useful links:

RStudio's List of Useful R Packages

Visual Tutorial on Histograms

Statistics

Variance Explained

R for Data Science - Grolemund and Wickham

Visualization

FlowingData

Junk Charts

Catalog of Visualization Types by Ferdio

Courses

Gary King - Quantitative Research Methodology

John Stasko - Information Visualization

Jenny Bryan - Data wrangling, exploration, and analysis with R

HarvardX Biomedical Data Science Open Online Training

Books

Econometrics in R

Using R for Data Analysis and Graphics

Papers

Embedded Plots

Weekly Assignments

Creating the R Markdown files

Weekly assignments are R Markdown files. To begin each assignment, create a new R Markdown file. By either

- Use the `File` menu, under the `New File` selection, click on `R Markdown...`, or
- Click on the new file button, then click on `R Markdown...`

In the dialog box, set the title to match the heading for the assignment. For example,

"Assignment 1.3"

Remember to use your name in the space for an author. When saving set the filename to the assignment followed by your first and last name, using dashes as separators. For example, if your name is John Snow, you would save Assignment 1.3 as `assignment-1-3-John-Snow`

Submitting your assignment

After you have completed the assignment and before the start of the next class period, submit your R Markdown file on classroom.google.com.

Individual Project

Project assignments

Each week, leading up to the project proposal, you will be given an assignment that is designed to provide you with an organized workflow for approaching new data science projects. Project assignments are submitted via classroom.google.com, with the exception of the two presentations

Project proposal presentation

This presentation should be less than 2 minutes. You simply need to communicate the core question your project seeks to answer and the dataset(s) you will be using to answer this question.

Final project RMarkdown

The final project will be an R Markdown document which communicates your project question, the data you used, and your results.

Final project presentation participation

Your final project participation grade is based on a combination of your own presentation and the feedback you provide to your classmates.

Part I

R Tutorials

Chapter 1

R Basics

Before we begin, make sure you have R and RStudio installed.

1.1 R Markdown

Throughout this course, R Markdown will make our lives easier. Make sure that the `rmarkdown` library is installed:

```
install.packages("rmarkdown")
```

For each assignment, you will create an R Markdown file (*.Rmd) and submit that file by the following class session using classroom.google.com. Each class has been made using R Markdown, so you can find many examples by going to the GitHub repository for this course github.com/jonpage/r-course

1.2 Working with data already loaded into R

Base R comes with a set of sample data that is useful for illustrating techniques in R. Run the following command to see a list of the datasets in the core library `datasets`:

```
library(help = "datasets")
```

These datasets are accessible automatically. We'll start with the Swiss Fertility and Socioeconomic Inicators (1888) dataset. See a description of the dataset by using the help command, either `?swiss` or `help(swiss)`. This dataset is technically a `data.frame`, which you can see by using the command `class(swiss)`. For more information on `data.frame`s take a look at the documentation(`help(data.frame)`)

1.2.1 Numeric summaries

Here are a few ways we can summarize a dataset:

`head()` shows us the first six rows of a `data.frame`.

```
head(swiss)
```

```
##              Fertility Agriculture Examination Education Catholic
## Courtelary        80.2        17.0          15        12     9.96
## Delemont          83.1        45.1           6         9    84.84
## Franches-Mnt      92.5        39.7           5         5    93.40
```

```
## Moutier           85.8        36.5          12         7    33.77
## Neuveville        76.9        43.5          17        15     5.16
## Porrentruy        76.1        35.3           9         7    90.57
##              Infant.Mortality
## Courtelary               22.2
## Delemont                 22.2
## Franches-Mnt             20.2
## Moutier                  20.3
## Neuveville               20.6
## Porrentruy               26.6
```

`summary()` provides summary statistics for each column in a `data.frame`.

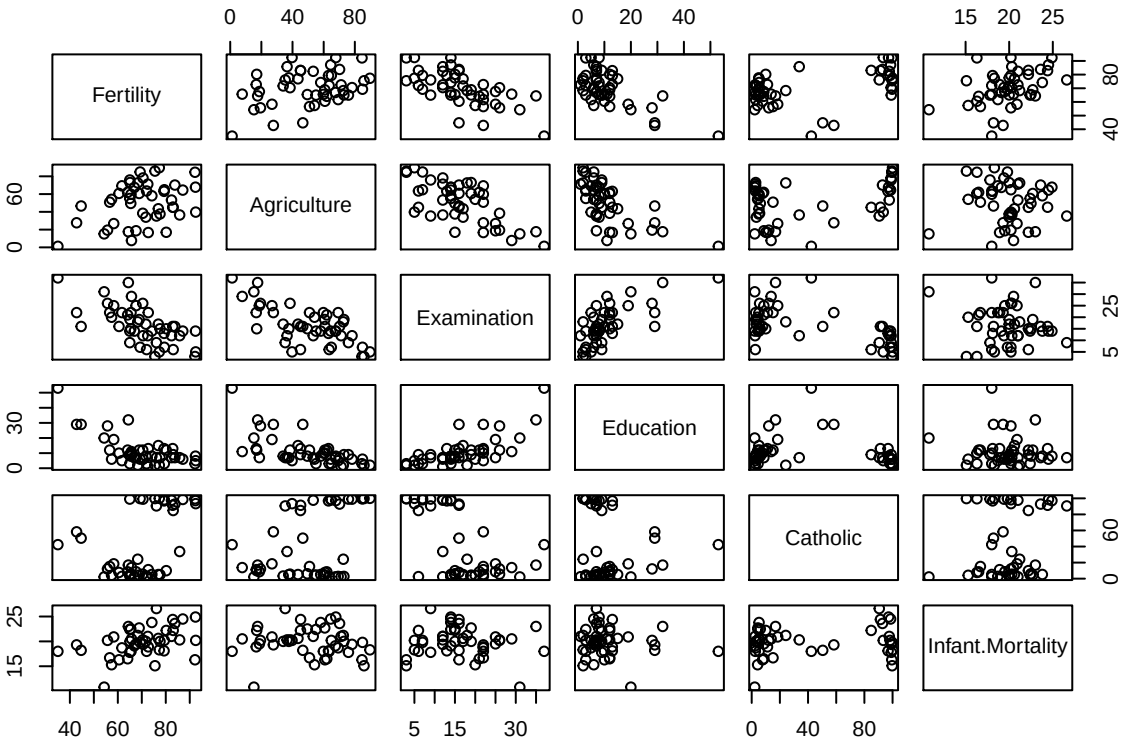
```
summary(swiss)
```

```
##    Fertility      Agriculture     Examination      Education
##  Min.   :35.00   Min.   : 1.20   Min.   : 3.00   Min.   : 1.00
##  1st Qu.:64.70   1st Qu.:35.90   1st Qu.:12.00   1st Qu.: 6.00
##  Median :70.40   Median :54.10   Median :16.00   Median : 8.00
##  Mean   :70.14   Mean   :50.66   Mean   :16.49   Mean   :10.98
##  3rd Qu.:78.45   3rd Qu.:67.65   3rd Qu.:22.00   3rd Qu.:12.00
##  Max.   :92.50   Max.   :89.70   Max.   :37.00   Max.   :53.00
##     Catholic       Infant.Mortality
##  Min.   :  2.150   Min.   :10.80
##  1st Qu.:  5.195   1st Qu.:18.15
##  Median : 15.140   Median :20.00
##  Mean   : 41.144   Mean   :19.94
##  3rd Qu.: 93.125   3rd Qu.:21.70
##  Max.   :100.000   Max.   :26.60
```

1.2.2 Visual summaries

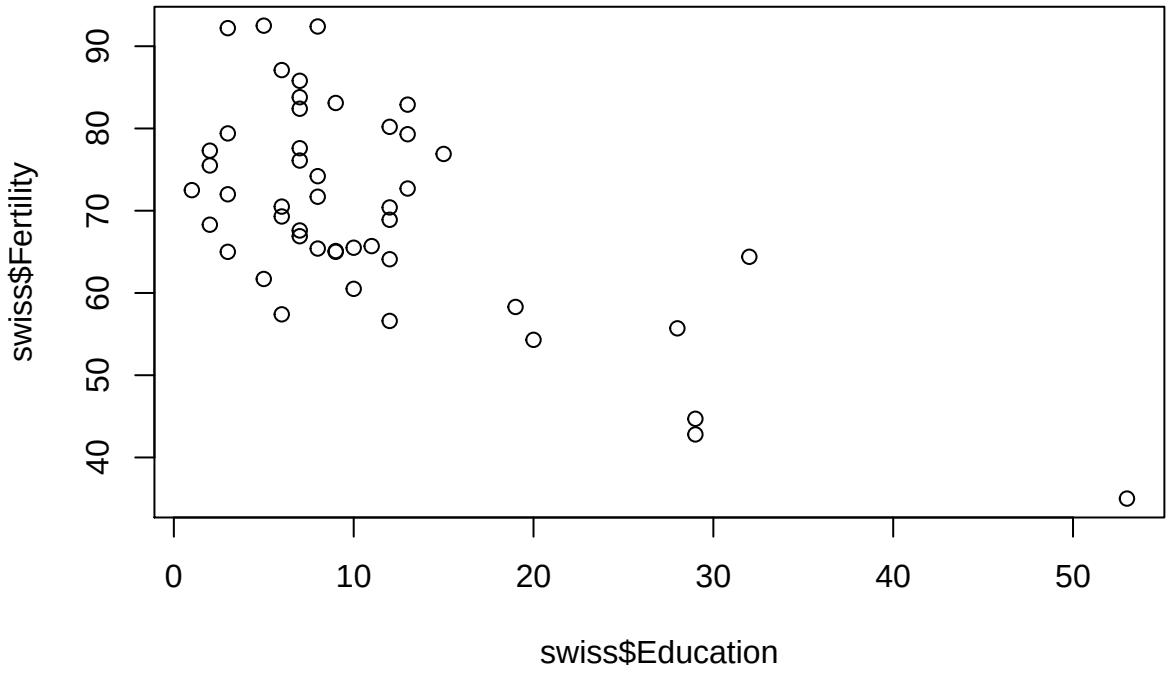
Scatterplot matrix (default plot of a data.frame):

```
plot(swiss)
# or
pairs(swiss)
```



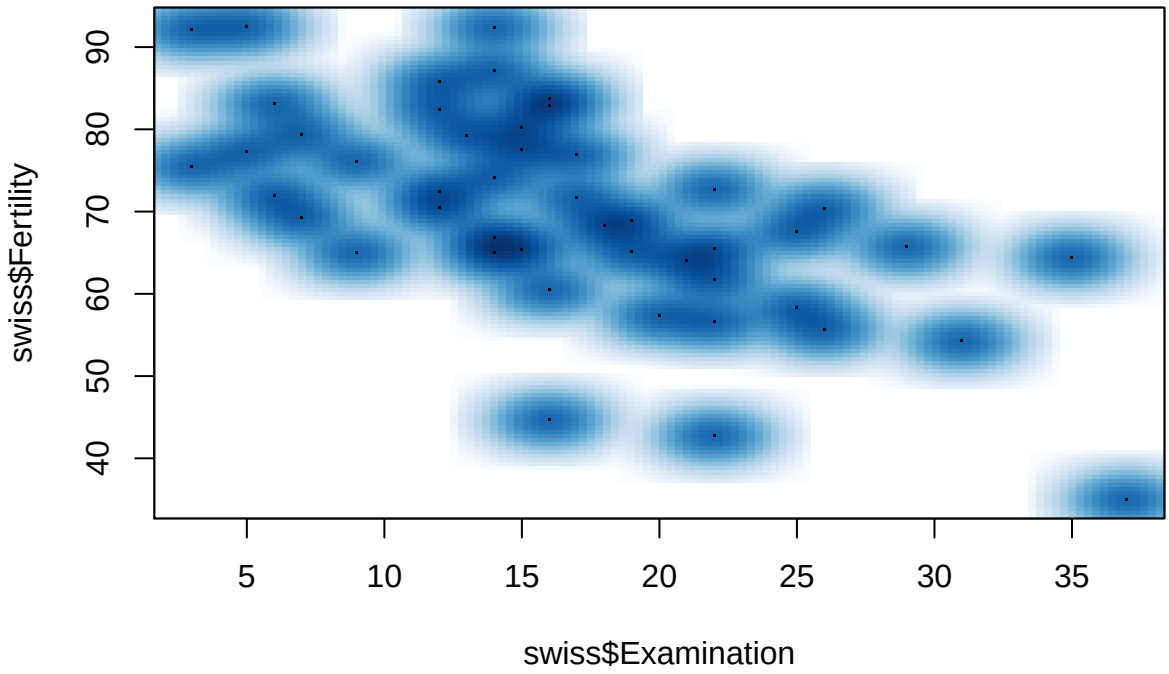
Scatterplot of two dimensions

```
plot(swiss[,c("Education", "Fertility")])
# or
plot(swiss[4,1])
# or
plot(swiss$Education, swiss$Fertility)
# or
plot(swiss$Fertility ~ swiss$Education)
```



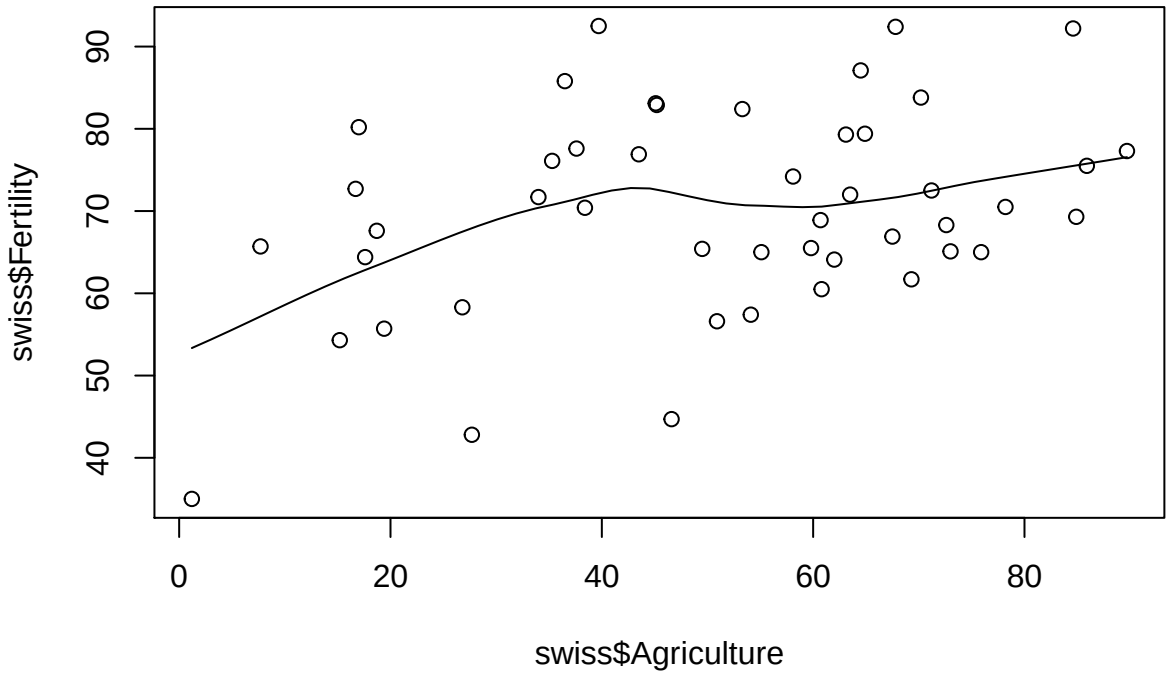
Smoothed Scatterplot of two dimensions

```
smoothScatter(swiss$Fertility ~ swiss$Examination)
```



Scatterplot with a `loess` (locally weighted polynomial regression)

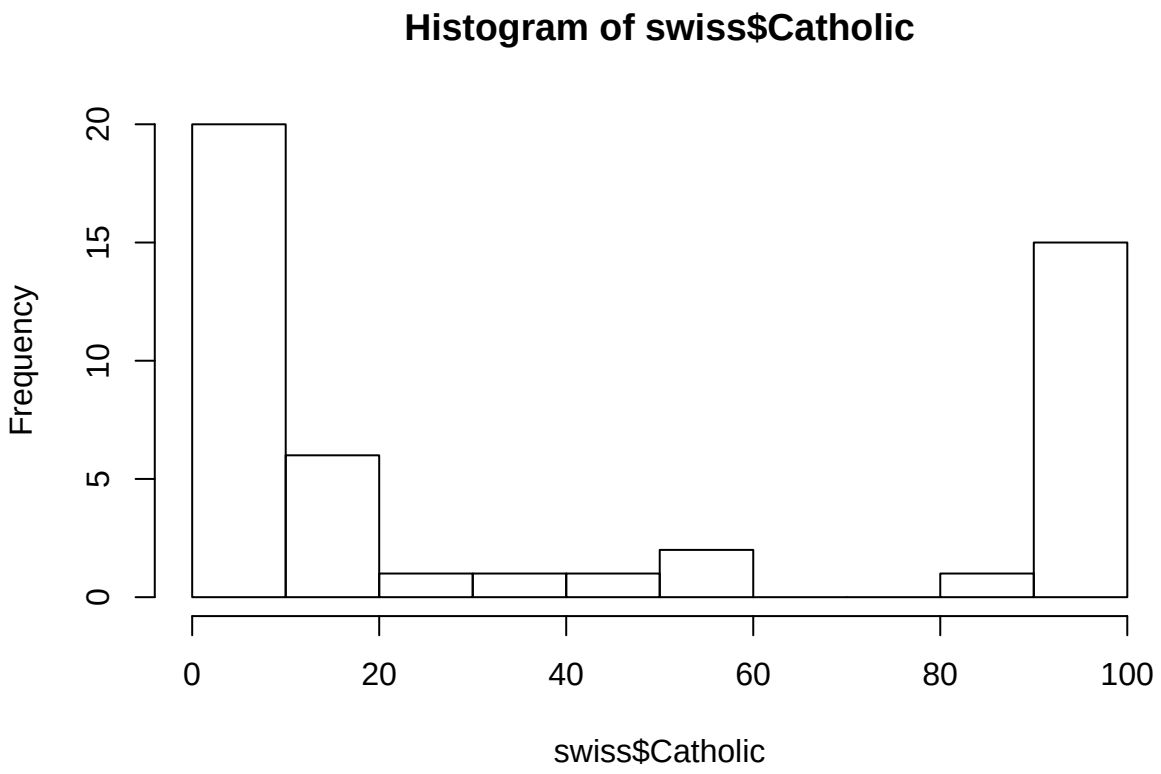
```
scatter.smooth(swiss$Fertility ~ swiss$Agriculture)
```



1.2.3 Distribution plots

Histograms:

```
hist(swiss$Catholic)
```



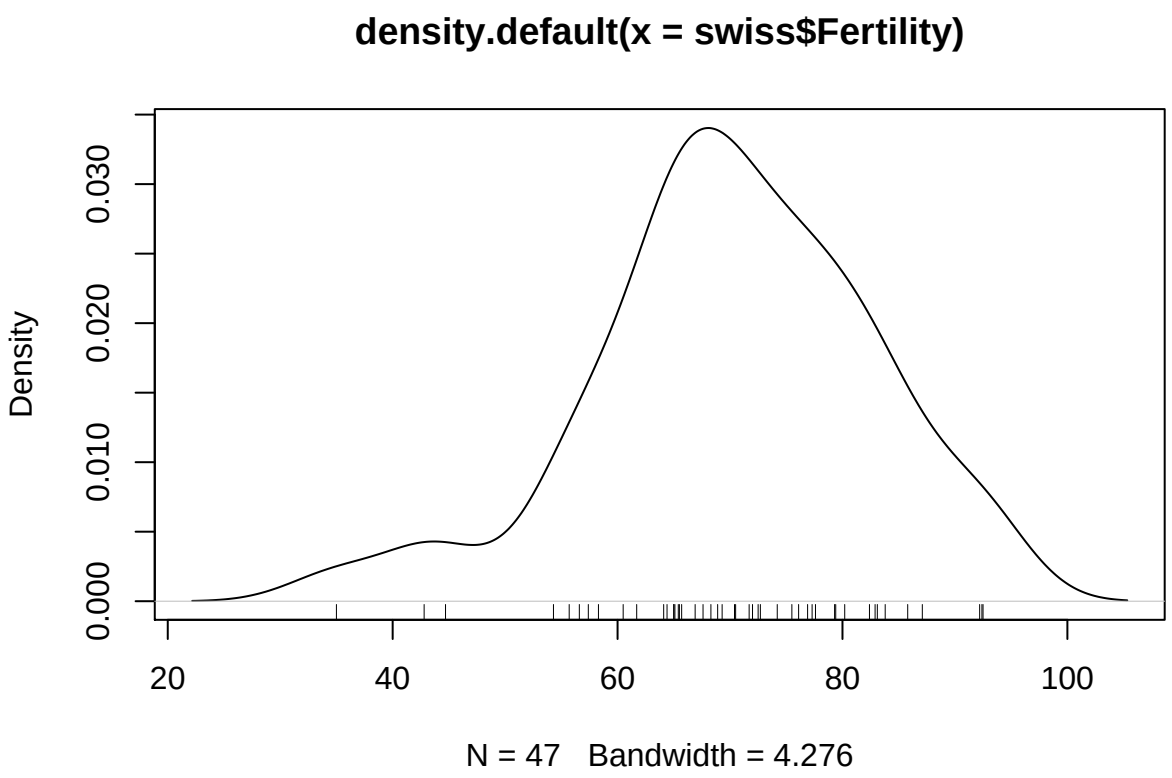
Stem-and-Leaf Plots:

```
stem(swiss$Fertility)
```

```
##
##   The decimal point is 1 digit(s) to the right of the |
##
##   3 | 5
##   4 | 35
##   5 | 46778
##   6 | 124455556678899
##   7 | 01223346677899
##   8 | 0233467
##   9 | 223
```

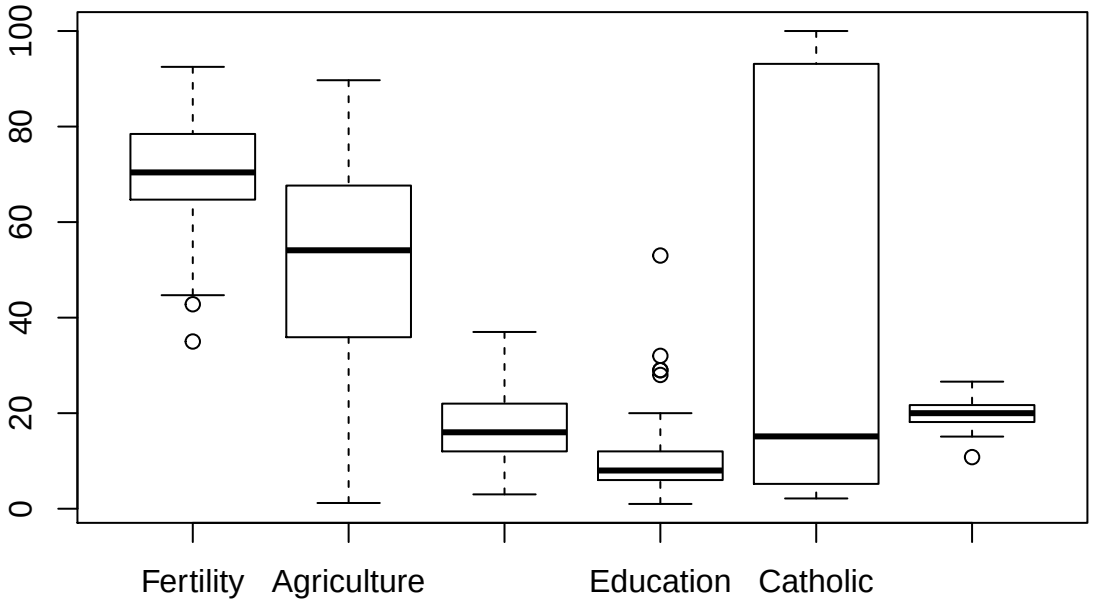
Kernel density plot (and add a rug showing where observation occur):

```
plot(density(swiss$Fertility))
rug(swiss$Fertility)
```



Boxplots:

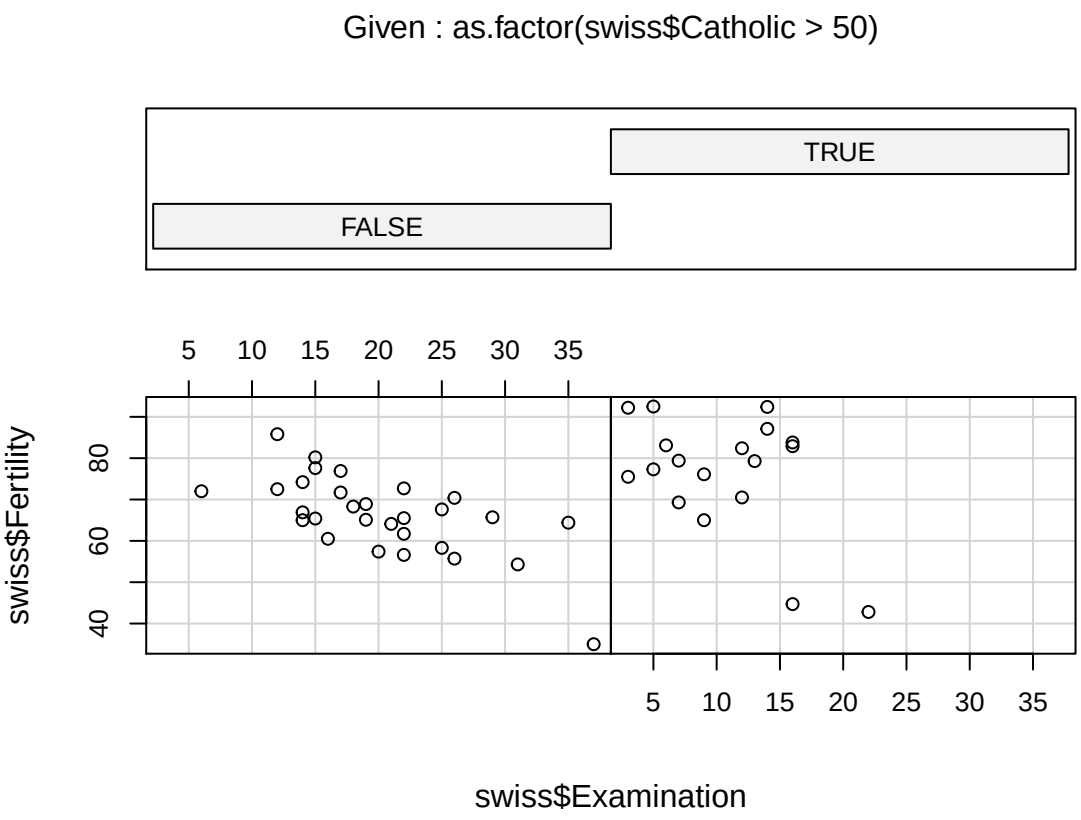
```
boxplot(swiss)
```



1.2.3.1 More complicated charts

Conditioning plots:

```
coplot(swiss$Fertility ~ swiss$Examination | as.factor(swiss$Catholic > 50))
```



Star plots (half-star plots here):

```
stars(swiss, key.loc = c(15,1), flip.labels = FALSE, full = FALSE)
```

1.3 Assignment

Create a new R Markdown file.

Choose a dataset from `datasets` (`library(help = "datasets")` will show you a list) and create 5 charts in an R Markdown file from the example charts above. Run the following command to see what else is available in the base R graphics package:

```
demo(graphics)
```

Chapter 2

Reading data

The first step in analyzing data with R is reading data into it. This lesson focuses on reading data, manipulating it with `dplyr` and a few summary visualizations.

2.1 Data Source:

The US Census Bureau has a large selection of data on the population of the United States. The public-use micro surveys (PUMS) are available from the following link:

https://www.census.gov/programs-surveys/acs/data/pums.html

We'll take a look at the 1-year American Community Survey results for the state of Hawaii.

Hawaii Population Records

The data dictionary

The specific file we are working with is the person record, so not every variable in the data dictionary will be available. Only those under the heading "PERSON RECORD" will be in the `csv_phi.zip` file.

2.2 `read_csv`

To read in the downloaded file, we'll use the `readr` package, which you can install by installing `tidyverse`.

```
library(tidyverse)
pop_hi <- read_csv("data/csv_phi.zip")
pop_hi
```

```
## # A tibble: 14,124 x 284
##       RT SERIALNO SPORDER  PUMA    ST  ADJINC PWGTP  AGEP   CIT CITWP
##    <chr>    <int>   <chr> <chr> <int>   <int> <chr> <chr> <int> <int>
##  1     P       21      01 00303    15 1001264 00078    63     1    NA
##  2     P      158      01 00306    15 1001264 00056    54     1    NA
##  3     P      267      01 00100    15 1001264 00059    52     5    NA
##  4     P      267      02 00100    15 1001264 00071    56     5    NA
##  5     P      267      03 00100    15 1001264 00102    25     5    NA
##  6     P      267      04 00100    15 1001264 00155    22     5    NA
##  7     P      267      05 00100    15 1001264 00074    15     5    NA
##  8     P      351      01 00100    15 1001264 00307    32     4  2005
```

```
##  9     P      351      02 00100    15 1001264 00578    02     1    NA
## 10     P      470      01 00200    15 1001264 00041    79     1    NA
## # ... with 14,114 more rows, and 274 more variables: COW <int>,
## #   DDRS <int>, DEAR <int>, DEYE <int>, DOUT <int>, DPHY <int>,
## #   DRAT <int>, DRATX <int>, DREM <int>, ENG <int>, FER <int>, GCL <int>,
## #   GCM <int>, GCR <int>, HINS1 <int>, HINS2 <int>, HINS3 <int>,
## #   HINS4 <int>, HINS5 <int>, HINS6 <int>, HINS7 <int>, INTP <chr>,
## #   JWMNP <chr>, JWRIP <chr>, JWTR <chr>, LANX <int>, MAR <int>,
## #   MARHD <int>, MARHM <int>, MARHT <int>, MARHW <int>, MARHYP <int>,
## #   MIG <int>, MIL <int>, MLPA <int>, MLPB <int>, MLPCD <int>, MLPE <int>,
## #   MLPFG <int>, MLPH <int>, MLPI <int>, MLPJ <int>, MLPK <int>,
## #   NWAB <int>, NWAV <int>, NWLA <int>, NWLK <int>, NWRE <int>, OIP <chr>,
## #   PAP <chr>, RELP <chr>, RETP <chr>, SCH <int>, SCHG <chr>, SCHL <chr>,
## #   SEMP <chr>, SEX <int>, SSIP <chr>, SSP <chr>, WAGP <chr>, WKHP <chr>,
## #   WKL <int>, WKW <int>, WRK <int>, YOEP <int>, ANC <int>, ANC1P <chr>,
## #   ANC2P <chr>, DECADE <int>, DIS <int>, DRIVESP <int>, ESP <int>,
## #   ESR <int>, FHICOVP <int>, FOD1P <int>, FOD2P <int>, HICOV <int>,
## #   HISP <chr>, INDP <chr>, JWAP <chr>, JWDP <chr>, LANP <int>,
## #   MIGPUMA <chr>, MIGSP <chr>, MSP <int>, NAICSP <chr>, NATIVITY <int>,
## #   NOP <int>, OC <int>, OCCP <chr>, PAOC <int>, PERNP <chr>, PINCP <chr>,
## #   POBP <chr>, POVPIP <chr>, POWPUMA <chr>, POWSP <chr>, PRIVCOV <int>,
## #   PUBCOV <int>, QTRBIR <int>, ...
```

2.3 dplyr

Using the data dictionary we can identify some interesting variables.

```
PERSON RECORD

RT      1
    Record Type
        P .Person Record

AGEP        2
    Age
        00  .Under 1 year
        01..99  .1 to 99 years (Top-coded***)

COW     1
    Class of worker
        b .N/A (less than 16 years old/NILF who last
          .worked more than 5 years ago or never worked)
        1 .Employee of a private for-profit company or
          .business, or of an individual, for wages,
          .salary, or commissions
        2 .Employee of a private not-for-profit,
          .tax-exempt, or charitable organization
        3 .Local government employee (city, county, etc.)
        4 .State government employee
        5 .Federal government employee
        6 .Self-employed in own not incorporated
          .business, professional practice, or farm
        7 .Self-employed in own incorporated
```

```
          .business, professional practice or farm
        8 .Working without pay in family business or farm
        9 .Unemployed and last worked 5 years ago or earlier or never
                  .worked

SCHL        2
    Educational attainment
        bb .N/A (less than 3 years old)
        01 .No schooling completed
        02 .Nursery school, preschool
        03 .Kindergarten
        04 .Grade 1
        05 .Grade 2
        06 .Grade 3
        07 .Grade 4
        08 .Grade 5
        09 .Grade 6
        10 .Grade 7
        11 .Grade 8
        12 .Grade 9
        13 .Grade 10
        14 .Grade 11
        15 .12th grade - no diploma
        16 .Regular high school diploma
        17 .GED or alternative credential
        18 .Some college, but less than 1 year
        19 .1 or more years of college credit, no degree
        20 .Associate's degree
        21 .Bachelor's degree
        22 .Master's degree
        23 .Professional degree beyond a bachelor's degree
        24 .Doctorate degree

WAGP        6
    Wages or salary income past 12 months
        bbbbbb      .N/A (less than 15 years old)
        000000      .None
        000001..999999  .$1 to 999999 (Rounded and top-coded)

Note: Use ADJINC to adjust WAGP to constant dollars.


WKHP        2
    Usual hours worked per week past 12 months
        bb   .N/A (less than 16 years old/did not work
             .during the past 12 months)
        01..98   .1 to 98 usual hours
        99   .99 or more usual hours

WKW     1
    Weeks worked during past 12 months
        b .N/A (less than 16 years old/did not work
          .during the past 12 months)
        1 .50 to 52 weeks worked during past 12 months
```

```
        2 .48 to 49 weeks worked during past 12 months
        3 .40 to 47 weeks worked during past 12 months
        4 .27 to 39 weeks worked during past 12 months
        5 .14 to 26 weeks worked during past 12 months
        6 .less than 14 weeks worked during past 12 months

ESR     1
    Employment status recode
        b .N/A (less than 16 years old)
        1 .Civilian employed, at work
        2 .Civilian employed, with a job but not at work
        3 .Unemployed
        4 .Armed forces, at work
        5 .Armed forces, with a job but not at work
        6 .Not in labor force

PERNP       7
    Total person's earnings
        bbbbbbb         .N/A (less than 15 years old)
        0000000         .No earnings
        -010000         .Loss of $10000 or more (Rounded & bottom-coded
                        .components)
        -000001..-009999    .Loss $1 to $9999 (Rounded components)
        0000001         .$1 or break even
        0000002..9999999    .$2 to $9999999 (Rounded & top-coded components)

Note: Use ADJINC to adjust PERNP to constant dollars.

PINCP       7
    Total person's income (signed)
        bbbbbbb         .N/A (less than 15 years old)
        0000000         .None
        -019999         .Loss of $19999 or more (Rounded & bottom-coded
                        .components)
        -000001..-019998    .Loss $1 to $19998 (Rounded components)
        0000001         .$1 or break even
        0000002..9999999    .$2 to $9999999 (Rounded & top-coded components)

Note: Use ADJINC to adjust PINCP to constant dollars.
```

Let's focus on employed civilians (ESR either 1 or 2) working full time (WKHP > 32) for close to the entire year (WKW either 1 or 2).

```
pop_hi <- pop_hi %>%
  filter(
    ESR %in% c(1, 2),
    as.numeric(WKHP) > 32,
    WKW %in% c(1, 2)
  )
```

If you are unsure if a column that you want to treat as numeric contains letters, you can run the following command to get a list of the values containing letters:

```
grep("[[:alpha:]]", pop_hi$WKHP, value = TRUE)
```

```
## character(0)
```

We can use two functions to add new columns (or change existing ones).

1. `mutate()` adds columns and keeps the previous columns
2. `transmute()` adds columns and removes the previous columns

This time we want to drop the columns we don't mention.

```
pop_hi <- pop_hi %>%
  transmute(
    age = as.numeric(AGEP),
    worker_class = factor(COW, labels = c(
      "for-profit",
      "not-for-profit",
      "local government",
      "state government",
      "federal government",
      "self-employed not incorporated",
      "self-employed incorporated",
      "family business no pay"
    )),
    school = SCHL,
    wages = as.numeric(WAGP),
    top_coded_wages = WAGP == 999999
)
```

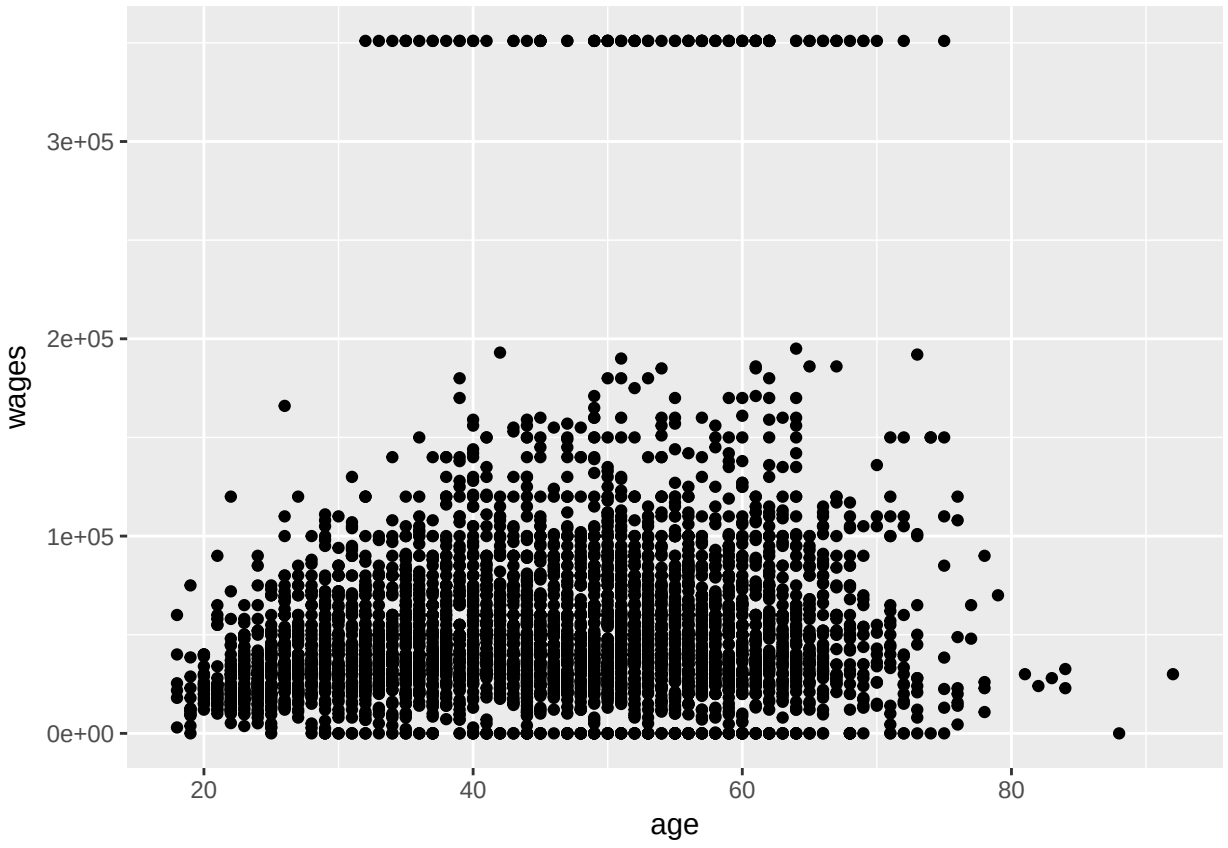
Creating a custom factor variable for educational attainment:

```
education_levels <- c("less than HS", "HS", "associates", "bachelors", "masters", "doctorate")

pop_hi$education <- NA
pop_hi[pop_hi$school < 16,]$education <- "less than HS"
pop_hi[pop_hi$school > 16 & pop_hi$school < 20,]$education <- "HS"
pop_hi[pop_hi$school == 20,]$education <- "associates"
pop_hi[pop_hi$school == 21,]$education <- "bachelors"
pop_hi[pop_hi$school %in% c(22, 23),]$education <- "masters"
pop_hi[pop_hi$school == 24,]$education <- "doctorate"

pop_hi <- pop_hi %>%
  mutate(education = factor(education, levels = education_levels))
```

2.4 First Look at ggplot2

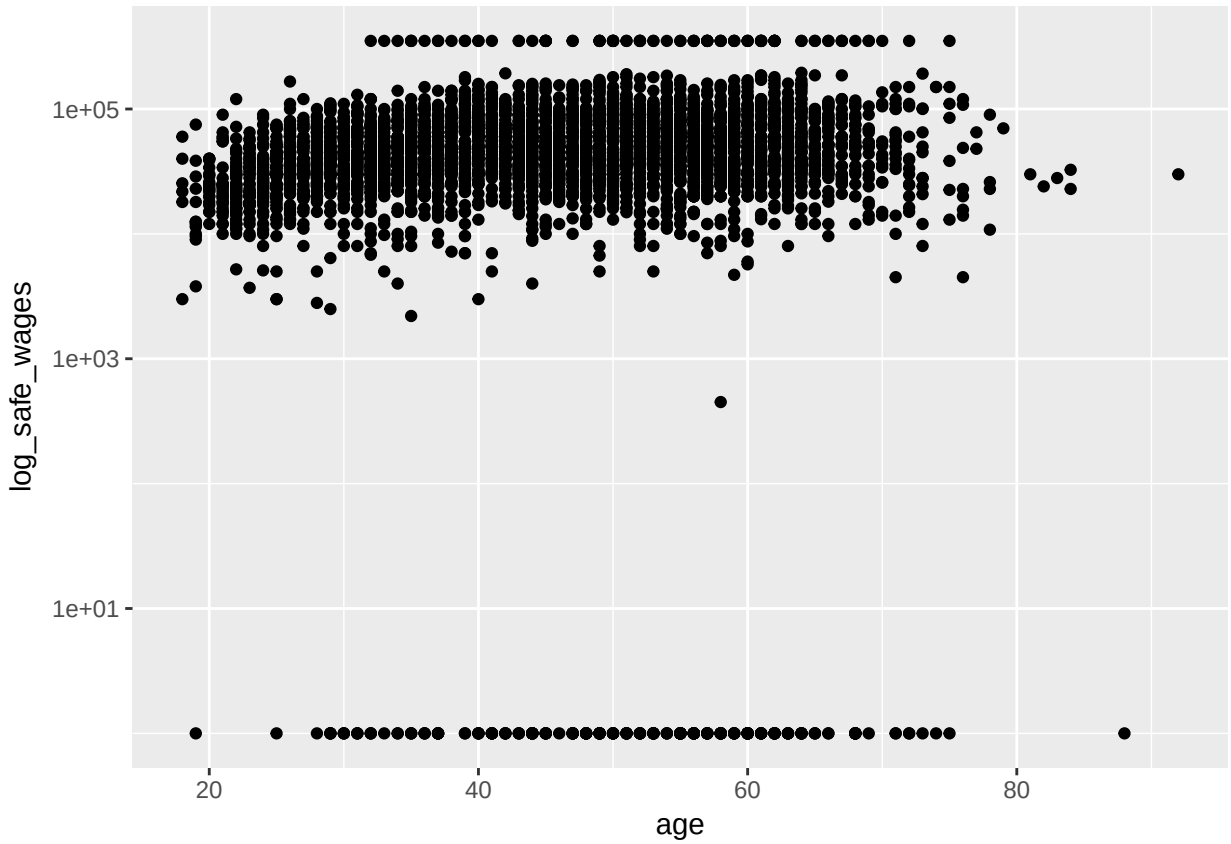
See the ggplot2 documentation for details and inspiration.

```
ggplot(pop_hi, aes(age, wages)) +
  geom_point()
```



Income has a skewed distribution, so it is often presented/analyzed in logs. Here's how to modify the above chart to display income in logs:

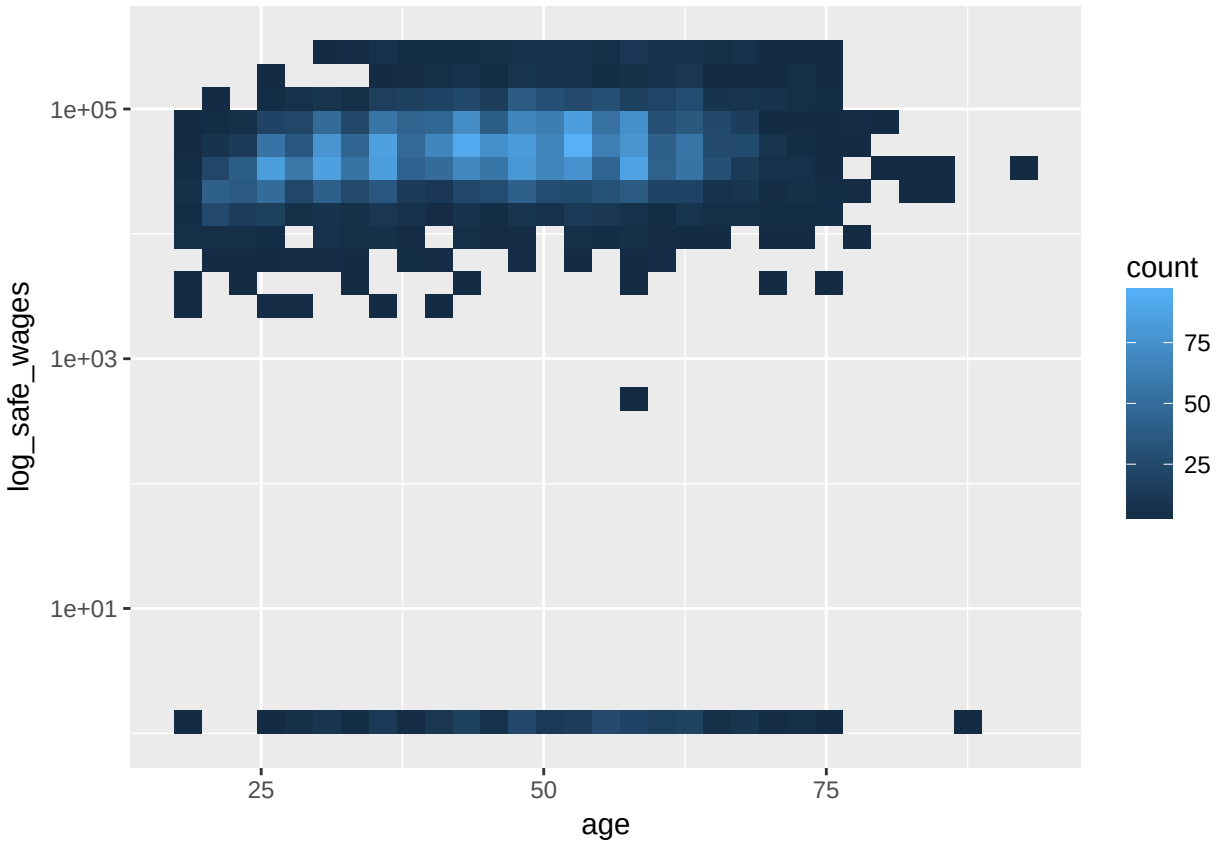
```
pop_hi <- pop_hi %>%
  mutate(log_safe_wages = ifelse(wages == 0, 1, wages))
pop_hi %>%
  ggplot(aes(age, log_safe_wages)) +
  geom_point() +
  scale_y_log10()
```



2.5 Heatmaps

Heatmaps allow you to get a sense of the concetration of observations in regions where there are many overlapping points:

```
ggplot(pop_hi, aes(age, log_safe_wages)) +
  geom_bin2d() +
  scale_y_log10()
```



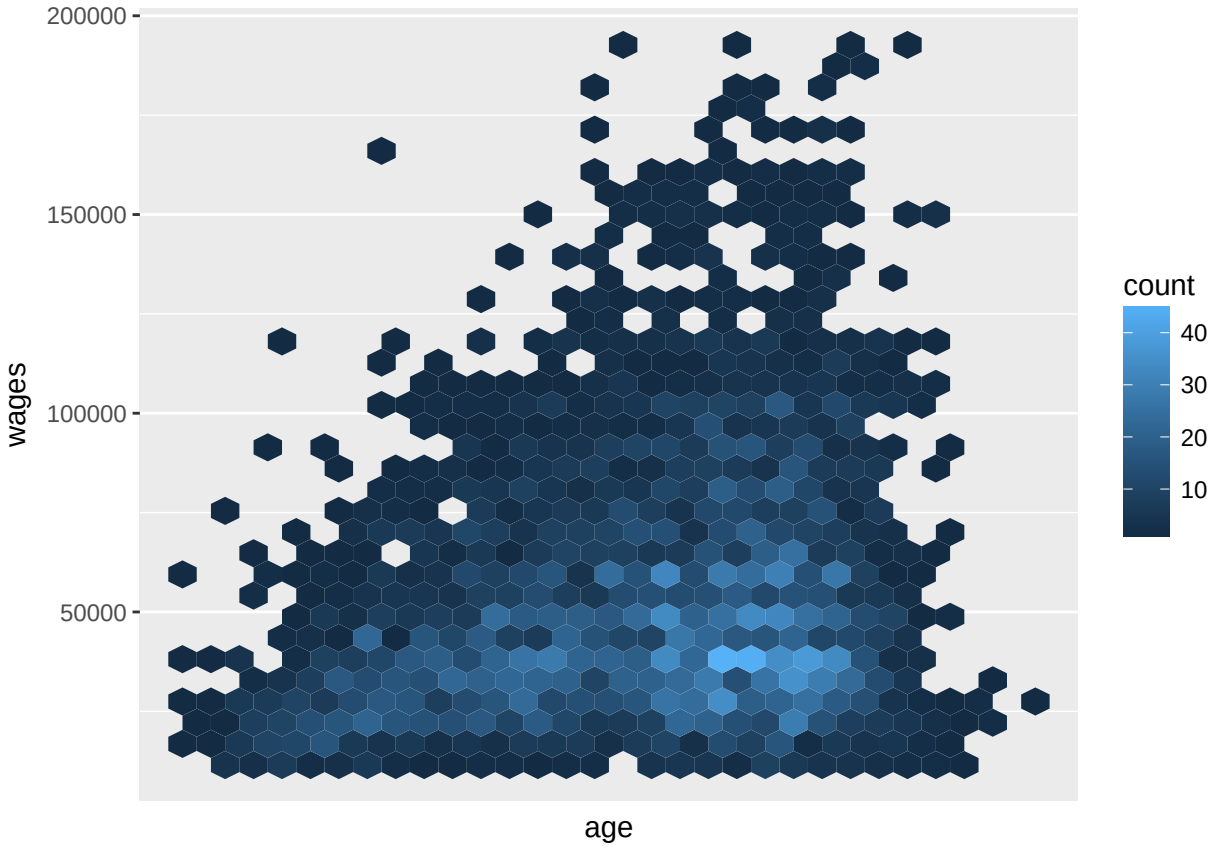
2.6 Hexbins

Hexbins use hexagons instead of squares, which helps avoid the rectangular sections in heatmaps that may misrepresent your data.

You will need to install the `hexbin` package.

```
install.packages("hexbin")
```

```
pop_hi %>%
  filter(wages > 10000, wages < 300000) %>%
  ggplot(aes(age, wages)) +
  geom_hex() +
  scale_x_log10()
```



Here we can see that inequality in wages for workers increases with age.

2.7 Other topics from this dataset

This dataset includes information on the majors for degree holders and the industry codes. You could use that additional information to ask how well targeted majors are to particular industries and how incomes vary across choice of major. Because of the size of the data in the Hawaii sample, it would be better to ask some of these questions at the national level.

Go to the ACS PUMS documentation page for more information.

2.8 Assignment

Choose another pair of variables from the data dictionary and visualize them with a scatterplot, a heatmap, and a hexbin plot.

Chapter 3

Facets, Bubbles, and Transparency

3.1 Data

For this session, we'll explore the Hawaii Tourism Authority (HTA) Air Seat Projection. I'll be working with the Air Seat Projection for 2017 (revised 06/17). Feel free to download the latest available.

3.1.1 Importing non-standard Excel files

The first steps in preparing a non-standard Excel file are (1) identify how many rows to skip and (2) provide column names if the column names are not neatly contained in a single row. You may also want to set the `range` if there is metadata at the end of the table you are importing. `range` overrides any `skip` setting, so we wont have to specify the number of rows to skip.

```
library(tidyverse)
library(readxl)
```

```
seats <- read_excel("data/2017 Air Seat Forecast rev 0617.xls", col_names = c(
  "dep_city",
  "seats2017Q1", "seats2017Q2", "seats2017Q3", "seats2017Q4", "seats2017",
  "seats2016Q1", "seats2016Q2", "seats2016Q3", "seats2016Q4", "seats2016",
  "seatschangeQ1", "seatschangeQ2", "seatschangeQ3", "seatschangeQ4", "seatschange"
), range = "A5:P78")
seats
```

```
## # A tibble: 74 x 16
##     dep_city seats2017Q1 seats2017Q2 seats2017Q3 seats2017Q4 seats2017
##        <chr>       <dbl>       <dbl>       <dbl>       <dbl>     <dbl>
##  1     TOTAL     2987920     3016376     3168233     3050112  12222641
##  2 SCHEDULED     2966915     2996155     3140998     3029794  12133862
##  3  CHARTERS       21005       20221       27235       20318     88779
##  4      <NA>          NA          NA          NA          NA        NA
##  5  US TOTAL     1996549     2108969     2215424     2071513   8392455
##  6 SCHEDULED     1978616     2091981     2200195     2055171   8325963
##  7  CHARTERS       17933       16988       15229       16342     66492
##  8      <NA>          NA          NA          NA          NA        NA
##  9   US WEST     1717254     1837080     1943653     1817441   7315428
## 10 Anchorage       25758       15105       13674       17013     71550
## # ... with 64 more rows, and 10 more variables: seats2016Q1 <dbl>,
```

```
## #   seats2016Q2 <dbl>, seats2016Q3 <dbl>, seats2016Q4 <dbl>,
## #   seats2016 <dbl>, seatschangeQ1 <dbl>, seatschangeQ2 <chr>,
## #   seatschangeQ3 <chr>, seatschangeQ4 <dbl>, seatschange <dbl>
```

Let's add a region identifier

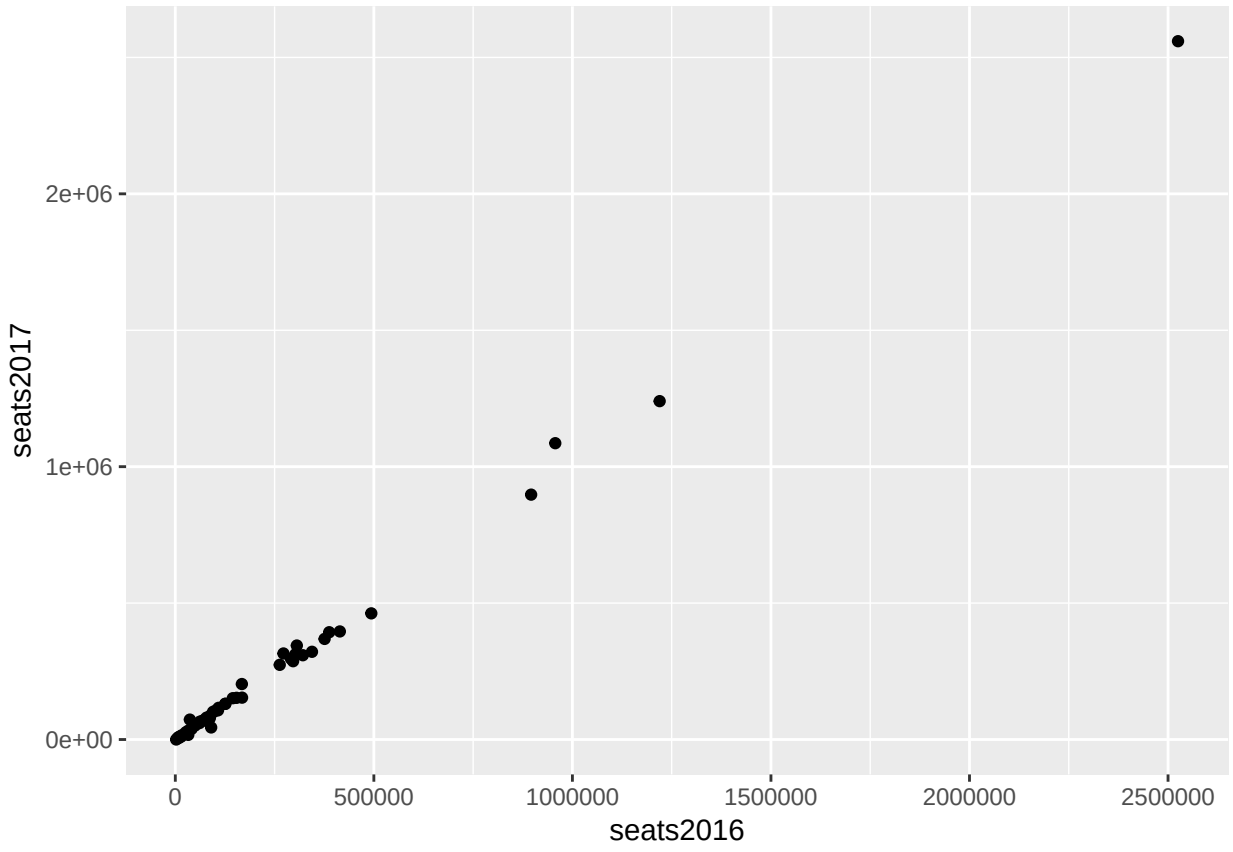
```
us_west_range <- 10:23
us_east_range <- 26:33
japan_range <- 40:45
canada_range <- 48:52
other_asia_range <-55:58
oceania_range <- 61:64
other_range <- 67:74

seats$region <- NA
seats[us_west_range,]$region <- "US West"
seats[us_east_range,]$region <- "US East"
seats[japan_range,]$region <- "Japan"
seats[canada_range,]$region <- "Canada"
seats[other_asia_range,]$region <- "Other Asia"
seats[oceania_range,]$region <- "Oceania"
seats[other_range,]$region <- "Other"

seats <- seats %>%
  filter(!is.na(region))
seats
```

```
## # A tibble: 49 x 17
##          dep_city seats2017Q1 seats2017Q2 seats2017Q3 seats2017Q4
##             <chr>       <dbl>       <dbl>       <dbl>       <dbl>
##  1      Anchorage       25758       15105       13674       17013
##  2     Bellingham       10198         318          NA        6519
##  3         Denver       55803       51654       52585       43290
##  4      Las Vegas       70514       74322       75839       75415
##  5    Los Angeles      548935      647498      715338      647703
##  6        Oakland       84571      104810      116015       90703
##  7        Phoenix      113046      115125      125348      108863
##  8       Portland       90207       71068       65997       81673
##  9     Sacramento       37620       38318       38456       38456
## 10 Salt Lake City       26370       23751       22968       28322
## # ... with 39 more rows, and 12 more variables: seats2017 <dbl>,
## #   seats2016Q1 <dbl>, seats2016Q2 <dbl>, seats2016Q3 <dbl>,
## #   seats2016Q4 <dbl>, seats2016 <dbl>, seatschangeQ1 <dbl>,
## #   seatschangeQ2 <chr>, seatschangeQ3 <chr>, seatschangeQ4 <dbl>,
## #   seatschange <dbl>, region <chr>
```

3.2 Facets

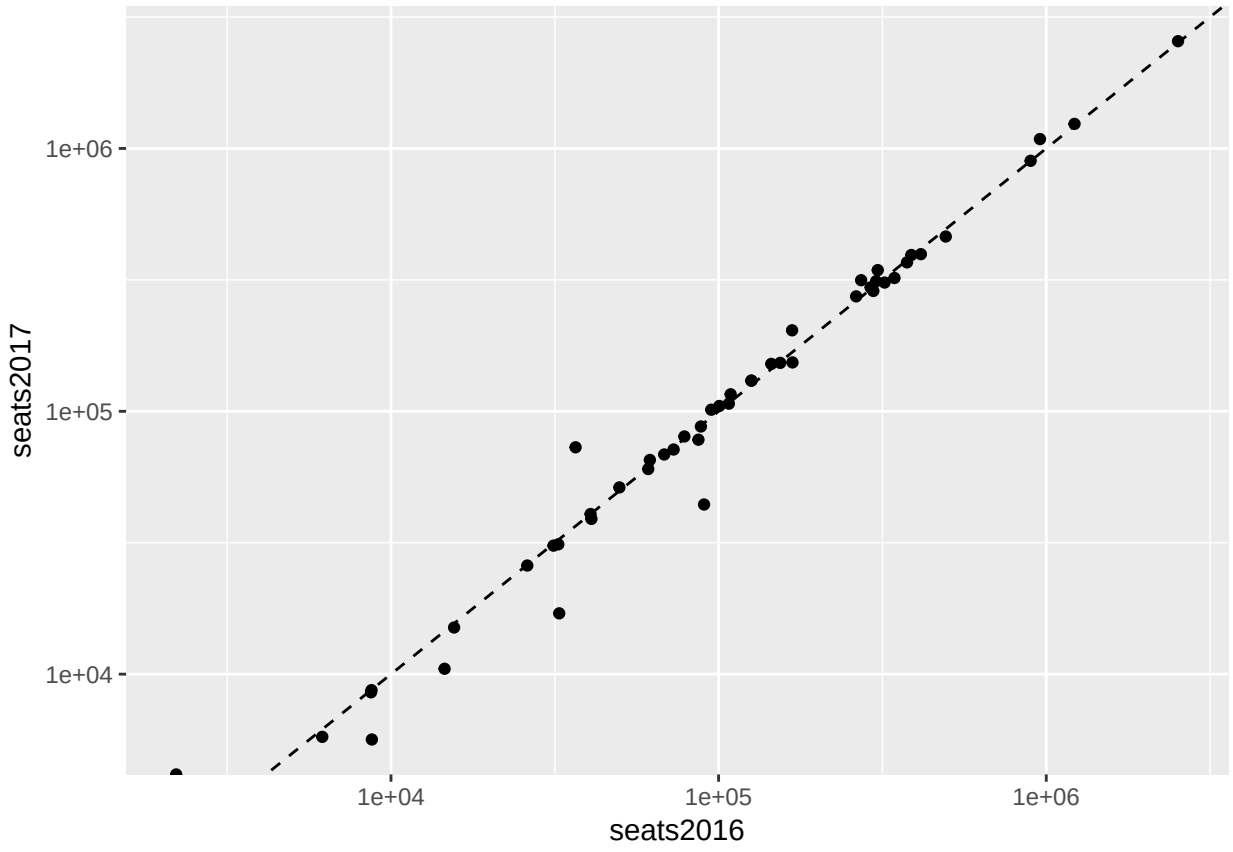
Let's do a simple plot comparing 2017 seats outlook to the 2016 seats outlook.

```
seats %>%
  ggplot(aes(seats2016, seats2017)) +
  geom_point()
```



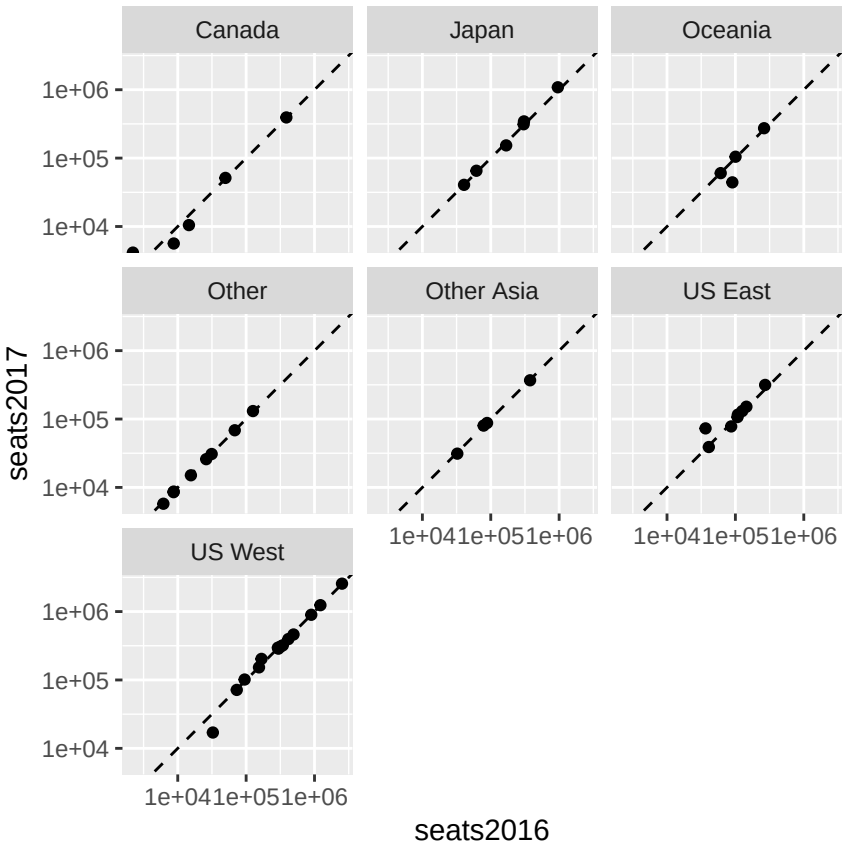
The distribution of this data looks like a good candidate for using the log scale (high concentration in lower values and lower concentration in higher values).

```
seats %>%
  ggplot(aes(seats2016, seats2017)) +
  geom_point() +
  scale_x_log10() +
  scale_y_log10() +
  geom_abline(lty = 2) # dashed line type (lty)
```



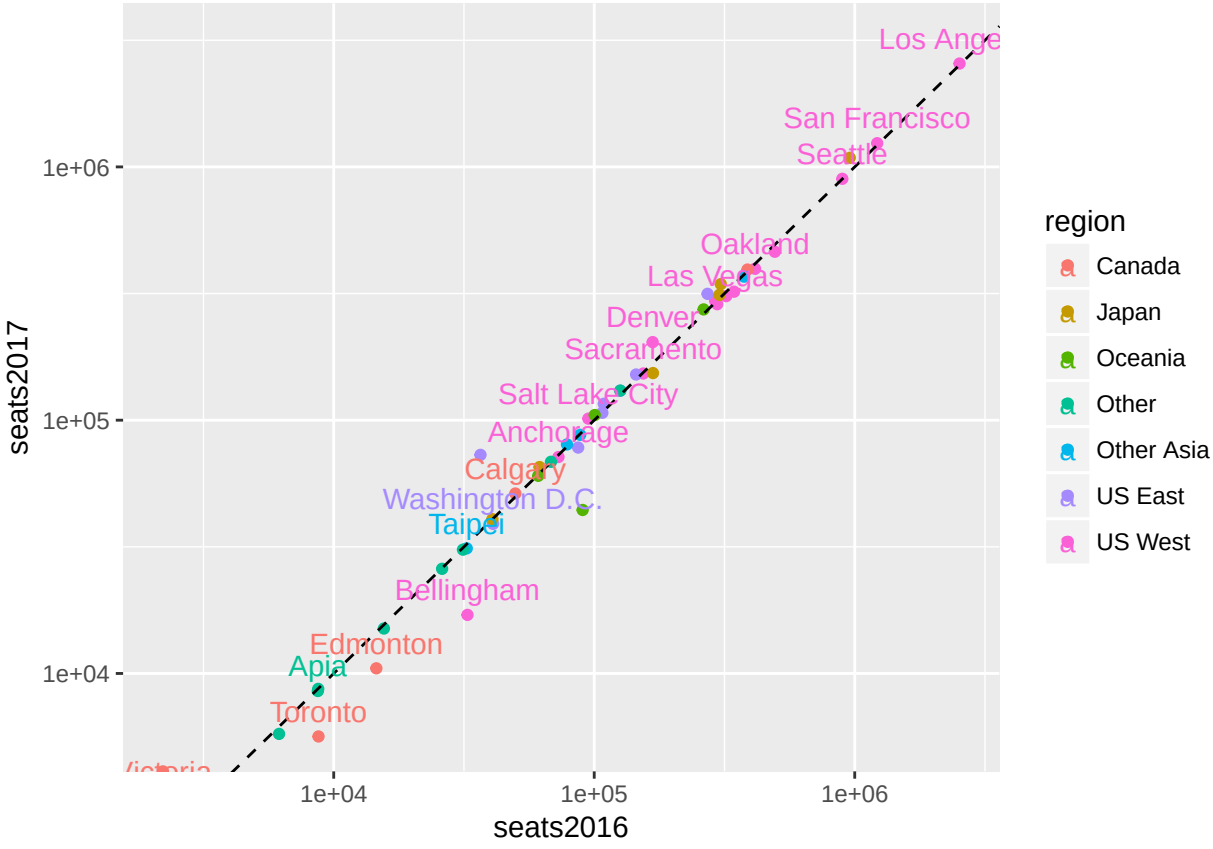
Since we have region identifiers it would be nice to divide our data and see charts of each region side-by-side. Facets allow us to make multiple charts based on a variable or set of variables.

```
seats %>%
  ggplot(aes(seats2016, seats2017)) +
  geom_point() +
  scale_x_log10() +
  scale_y_log10() +
  geom_abline(lty = 2) +
  facet_wrap(~ region) +
  coord_fixed()
```



An alternative representation is to present each region using color:

```
seats %>%
  ggplot(aes(seats2016, seats2017, color = region, label = dep_city)) +
  geom_point() +
  scale_x_log10() +
  scale_y_log10() +
  geom_abline(lty = 2) +
  geom_text(check_overlap = TRUE, nudge_y = 0.1)
```



3.3 Bubbles

Bubble charts are scatter plots (geom_point) with points that vary in size corresponding to the value of a given variable. Let's create a measure of the size of a city's seats relative to its regional total.

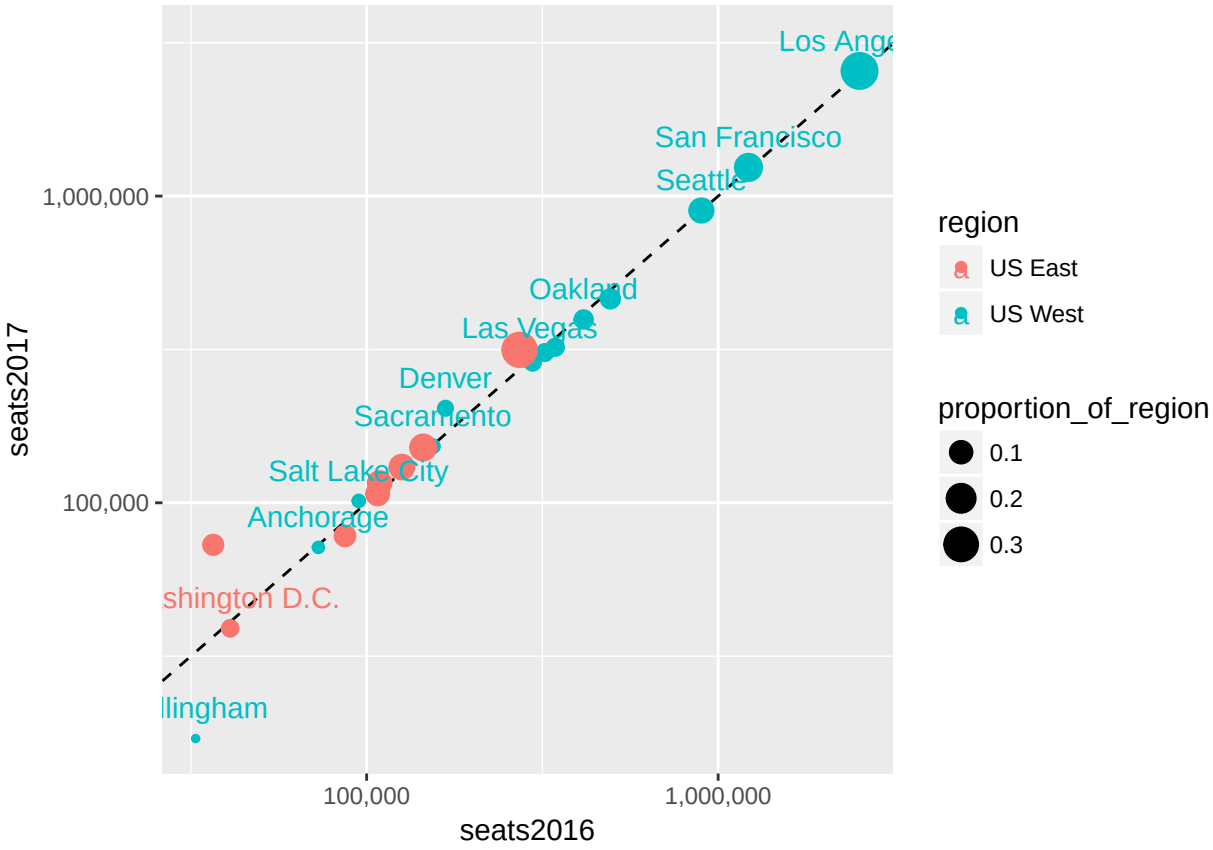
```
seats <- seats %>%
  group_by(region) %>%
  mutate(proportion_of_region = seats2017/sum(seats2017))
seats
```

```
## # A tibble: 49 x 18
## # Groups:   region [7]
##          dep_city seats2017Q1 seats2017Q2 seats2017Q3 seats2017Q4
##             <chr>       <dbl>       <dbl>       <dbl>       <dbl>
##  1      Anchorage       25758       15105       13674       17013
##  2     Bellingham       10198         318          NA        6519
##  3         Denver       55803       51654       52585       43290
##  4      Las Vegas       70514       74322       75839       75415
##  5    Los Angeles      548935      647498      715338      647703
##  6        Oakland       84571      104810      116015       90703
##  7        Phoenix      113046      115125      125348      108863
##  8       Portland       90207       71068       65997       81673
##  9     Sacramento       37620       38318       38456       38456
## 10 Salt Lake City       26370       23751       22968       28322
## # ... with 39 more rows, and 13 more variables: seats2017 <dbl>,
## #   seats2016Q1 <dbl>, seats2016Q2 <dbl>, seats2016Q3 <dbl>,
```

```
## #   seats2016Q4 <dbl>, seats2016 <dbl>, seatschangeQ1 <dbl>,
## #   seatschangeQ2 <chr>, seatschangeQ3 <chr>, seatschangeQ4 <dbl>,
## #   seatschange <dbl>, region <chr>, proportion_of_region <dbl>
```

Now we can modify the chart to show the importance of each city in the context of its region.

```
seats %>%
  filter(region %in% c("US West", "US East")) %>%
  ggplot(aes(seats2016, seats2017, color = region, label = dep_city)) +
  geom_abline(lty = 2) +
  geom_point(aes(size = proportion_of_region)) +
  scale_x_log10(labels = scales::comma) +
  scale_y_log10(labels = scales::comma) +
  geom_text(check_overlap = TRUE, nudge_y = 0.1)
```

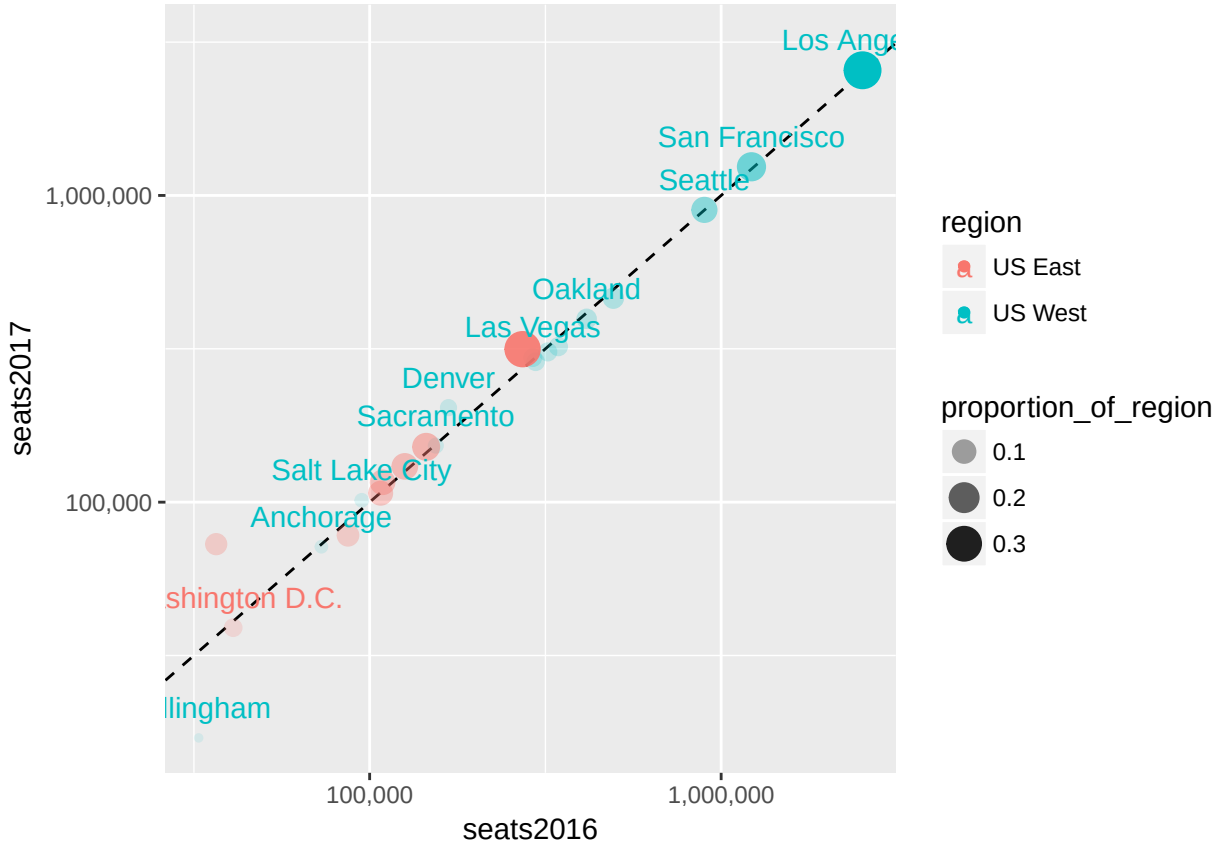


3.4 Transparency

We can also use transparency (or alpha) to make less important points less visible. We do this by setting the `alpha` aesthetic. Let's try adding the alpha setting to the `geom_point()` call first.

```
seats %>%
  filter(region %in% c("US West", "US East")) %>%
  ggplot(aes(seats2016, seats2017, color = region, label = dep_city)) +
  geom_abline(lty = 2) +
  geom_point(aes(size = proportion_of_region, alpha = proportion_of_region)) +
  scale_x_log10(labels = scales::comma) +
```

```
  scale_y_log10(labels = scales::comma) +
  geom_text(check_overlap = TRUE, nudge_y = 0.1)
```

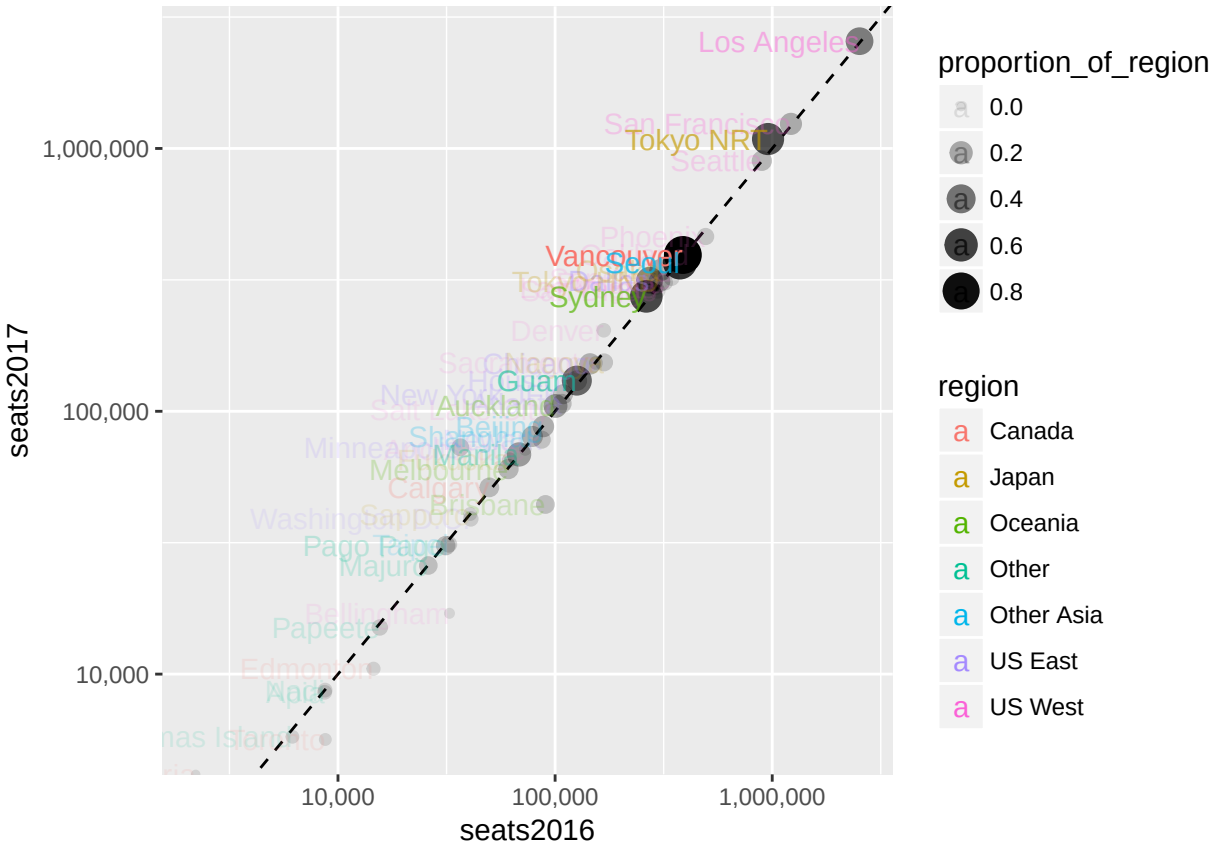


Let's add the alpha to the ggplot-level aesthetic instead, so that it also affects the text labels.

```
seats %>%
  filter(region %in% c("US West", "US East")) %>%
  ggplot(aes(seats2016, seats2017, color = region, label = dep_city, alpha = proportion_of_region)) +
  geom_abline(lty = 2) +
  geom_point(aes(size = proportion_of_region)) +
  scale_x_log10(labels = scales::comma) +
  scale_y_log10(labels = scales::comma) +
  geom_text(nudge_y = 0.1)
```

We can combine all the regions now and use transparency to help us see how many cities are in the same area on the plot by how dark a region is.

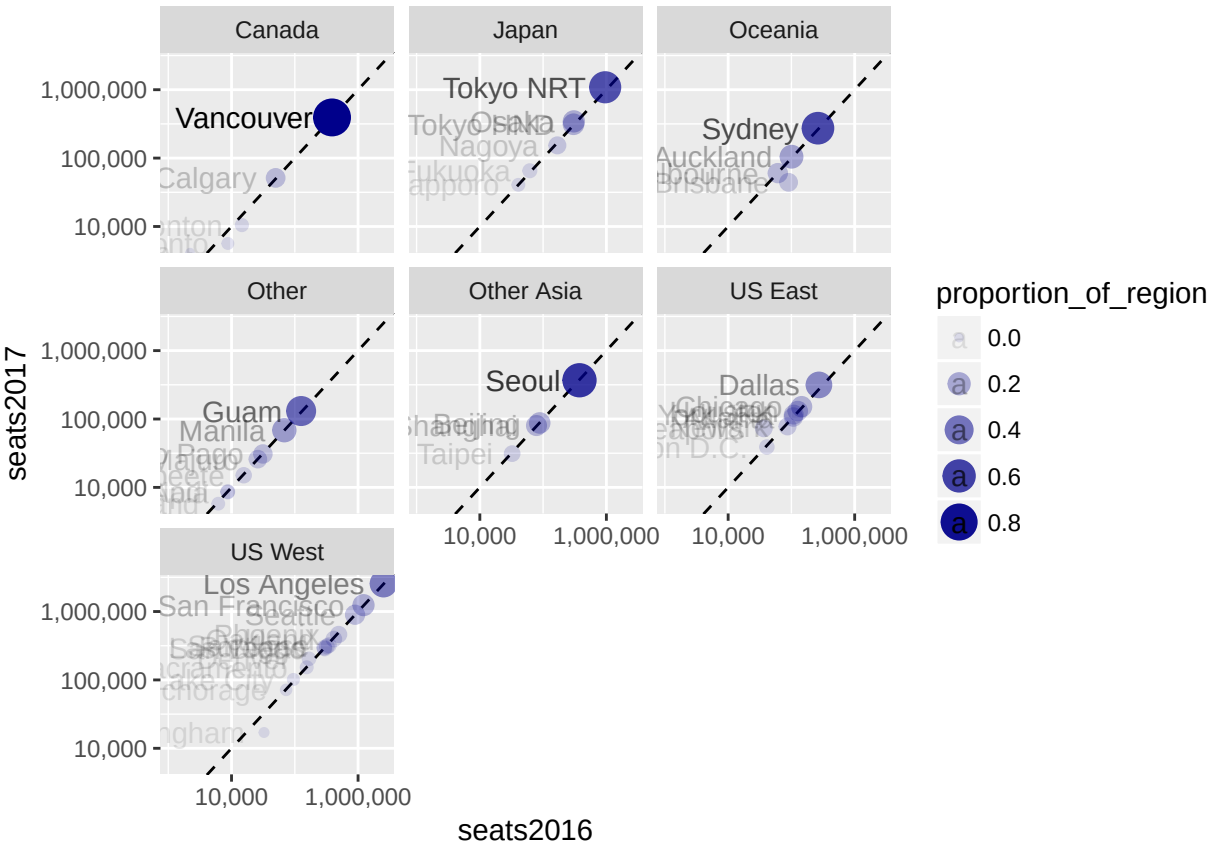
```
seats %>%
  ggplot(aes(seats2016, seats2017, label = dep_city, alpha = proportion_of_region)) +
  geom_abline(lty = 2) +
  geom_point(aes(size = proportion_of_region)) +
  scale_x_log10(labels = scales::comma) +
  scale_y_log10(labels = scales::comma) +
  geom_text(aes(color = region), hjust = "right", vjust = "center")
```



3.5 Facets

Let's use facets so we can combine everything we've done so far.

```
seats %>%
  ggplot(aes(seats2016, seats2017, label = dep_city, alpha = proportion_of_region)) +
  geom_abline(lty = 2) +
  geom_point(aes(size = proportion_of_region), color = "darkblue") +
  scale_x_log10(labels = scales::comma) +
  scale_y_log10(labels = scales::comma) +
  geom_text(hjust = "right", vjust = "center", nudge_x = -0.3) +
  facet_wrap(~ region)
```



3.6 Assignment

Create a bubble plot highlighting the change in year-on-year growth rates for different quarters. Plot `seatschangeQ3` on the x axis and `seatschangeQ4` on the y axis. Use `seats2017` to determine the size of each bubble. Facet by region.

Chapter 4

Lines and Curves

4.1 Data

We will use data on the number of active duty personnel in Hawaii. The first dataset is an Excel file pulled from the State of Hawaii Department of Business, Economic Development, and Tourism (DBEDT) 2015 State of Hawaii Data Book. See the line listed as, "10.03 - Active Duty Personnel, by Service: 1953 to 2015." The data is originally from the US Defense Manpower Data Center

```
library(tidyverse)
library(readxl)
```

```
mil_personnel <- read_excel("data/100315.xls", range = "A5:L38", col_types = "numeric")
mil_personnel <- bind_rows(
  mil_personnel %>% select(1:6) %>% magrittr::set_colnames(c("Year", "Total", "Army", "Navy", "Marine C
  mil_personnel %>% select(7:12) %>% magrittr::set_colnames(c("Year", "Total", "Army", "Navy", "Marine
)
mil_personnel
```

```
## # A tibble: 66 x 6
##     Year Total  Army  Navy `Marine Corps` `Air Force`
##    <dbl> <dbl> <dbl> <dbl>          <dbl>       <dbl>
##  1    NA    NA    NA    NA             NA          NA
##  2  1953 24785  5872  7657           6040        5216
##  3  1954 23654  7957  6443           4155        5099
##  4  1955 40258 19821  5211           9677        5549
##  5  1956 37470 16531  5237           9490        6212
##  6  1957 40683 17511  5466           9608        8098
##  7  1958 35076 14672  4908           8670        6826
##  8  1959 36310 15438  5309           8470        7093
##  9  1960 35412 15492  5687           7756        6477
## 10  1961 39474 16945  5774           9679        7076
## # ... with 56 more rows
```

Notice that the `Year` 2015 was turned into `NA`. This happened because the value in the corresponding cell was '2/ 2015'. Let's remove the final row of `NA`s and replace the remaining `NA` with 2015.

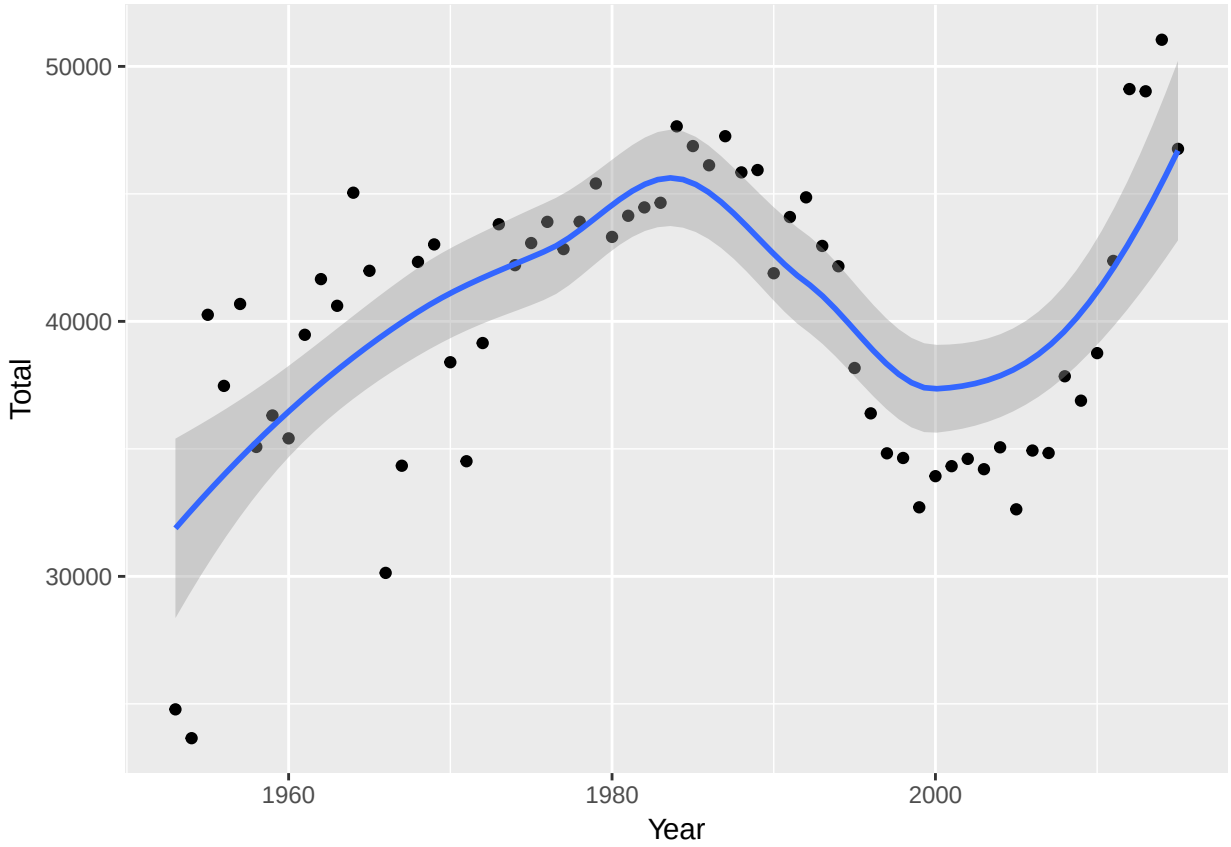
```
mil_personnel <- mil_personnel %>% filter(!is.na(Total))
mil_personnel[is.na(mil_personnel$Year),]$Year <- 2015
mil_personnel
```

```
## # A tibble: 63 x 6
##     Year Total  Army  Navy `Marine Corps` `Air Force`
##    <dbl> <dbl> <dbl> <dbl>          <dbl>       <dbl>
##  1  1953 24785  5872  7657           6040        5216
##  2  1954 23654  7957  6443           4155        5099
##  3  1955 40258 19821  5211           9677        5549
##  4  1956 37470 16531  5237           9490        6212
##  5  1957 40683 17511  5466           9608        8098
##  6  1958 35076 14672  4908           8670        6826
##  7  1959 36310 15438  5309           8470        7093
##  8  1960 35412 15492  5687           7756        6477
##  9  1961 39474 16945  5774           9679        7076
## 10  1962 41657 17645  6664           9903        7445
## # ... with 53 more rows
```

4.2 geom_smooth

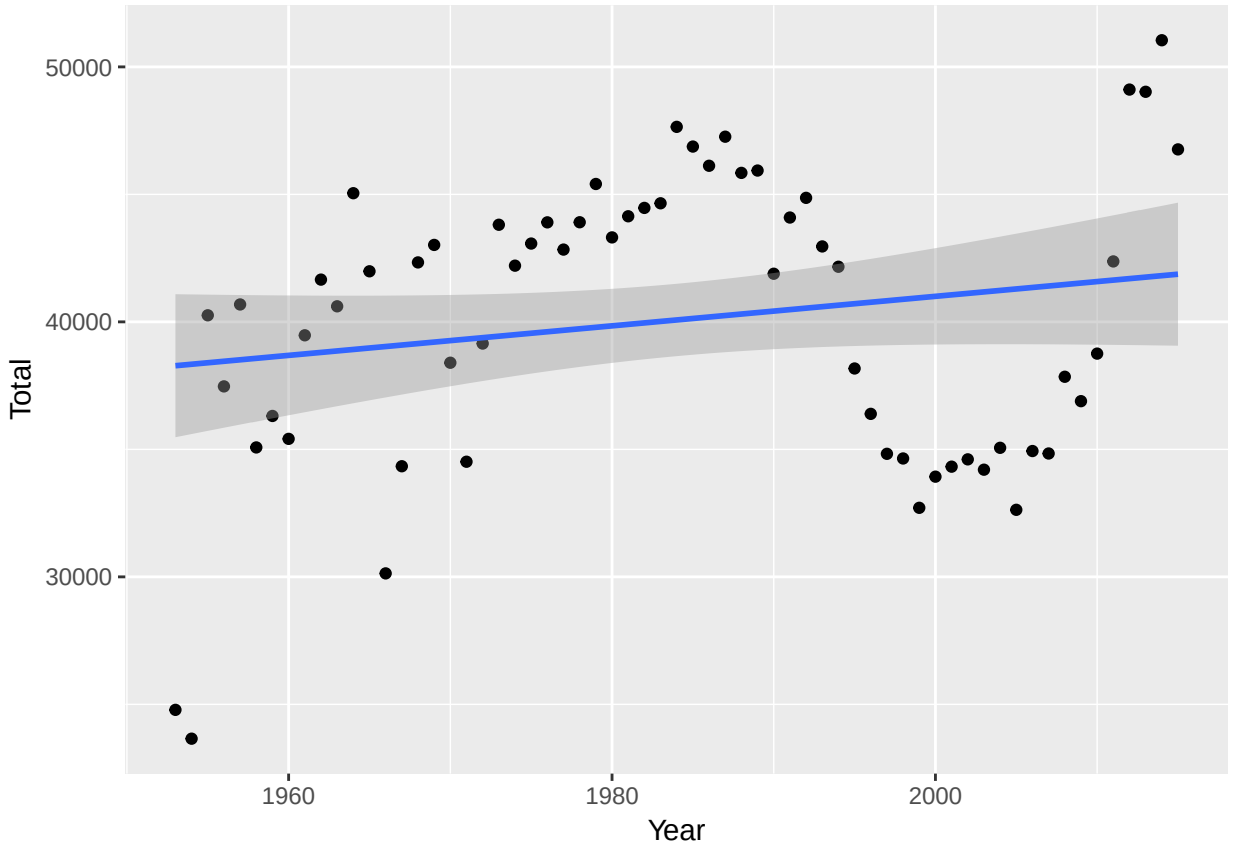
`geom_smooth` allows you to have smooth lines appear in your chart. With no argument, it will choose `loess` for series shorter than 1,000 observations. It shows a shaded confidence interval.

```
mil_personnel %>%
  ggplot(aes(Year, Total)) +
  geom_point() +
  geom_smooth()
```



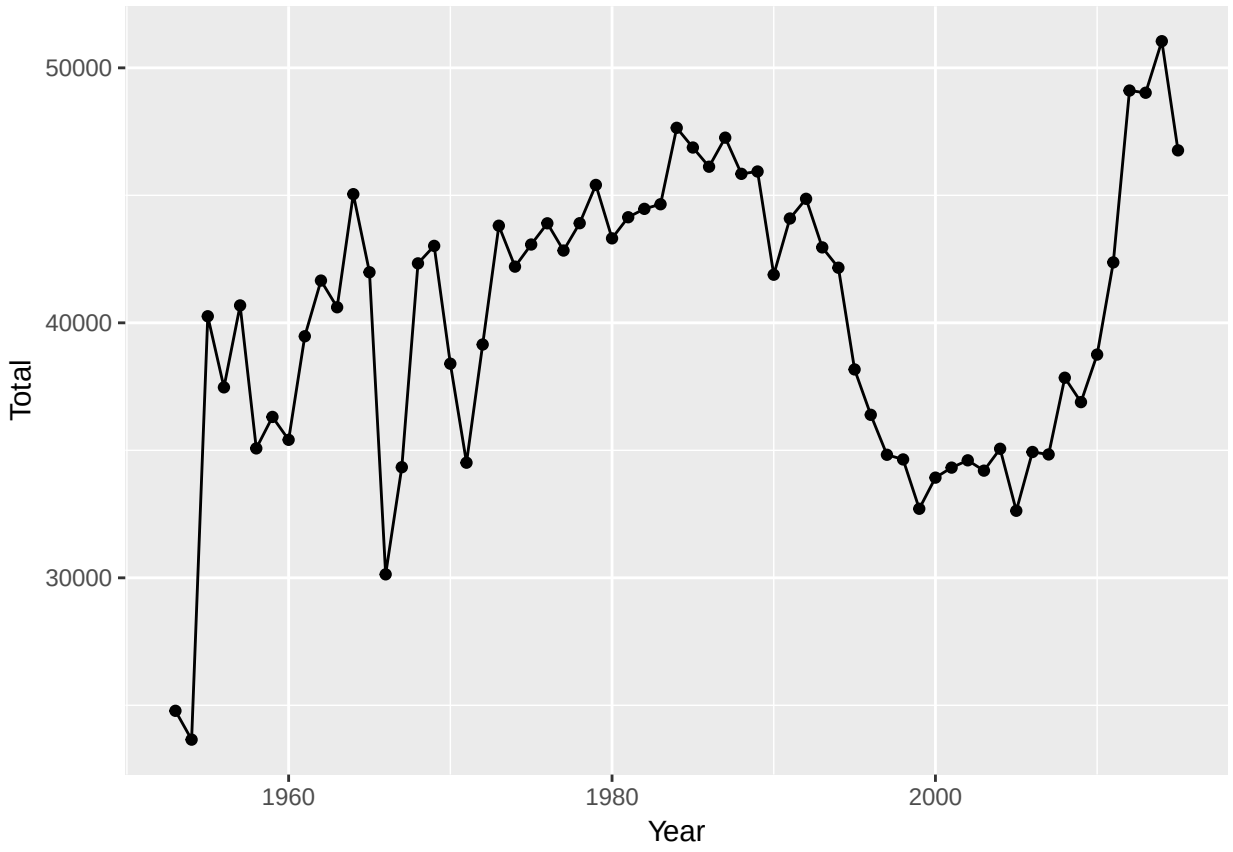
Here's what it looks like if we fit a linear model instead:

```
mil_personnel %>%
  ggplot(aes(Year, Total)) +
  geom_point() +
  geom_smooth(method = "lm")
```



We can also just have a line chart that connects the points:

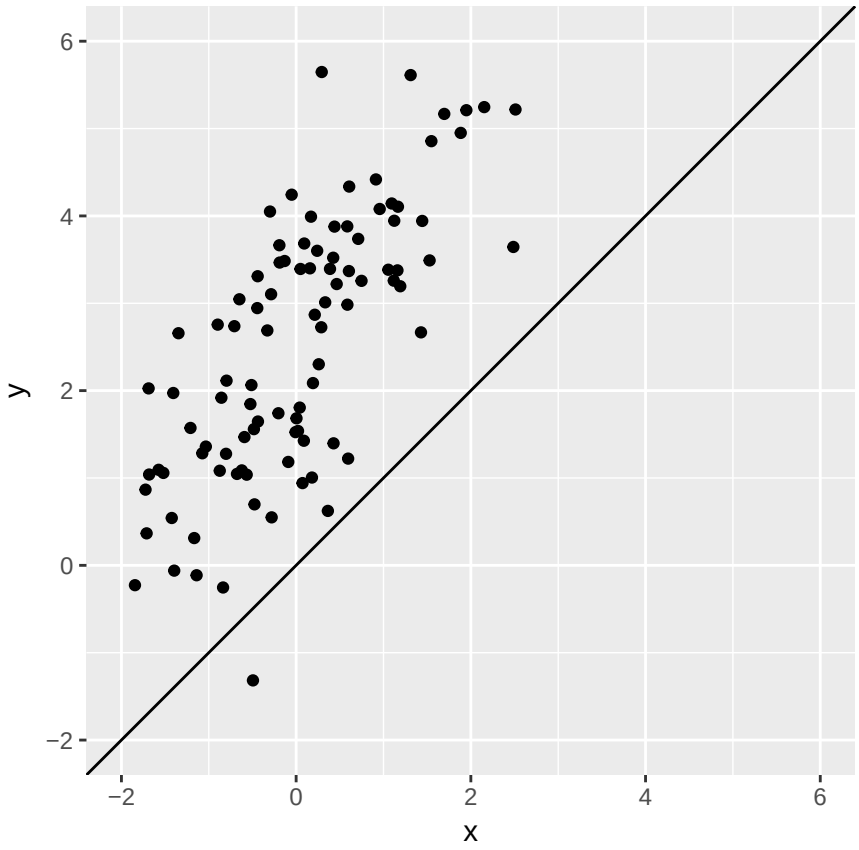
```
mil_personnel %>%
  ggplot(aes(Year, Total)) +
  geom_point() +
  geom_line()
```



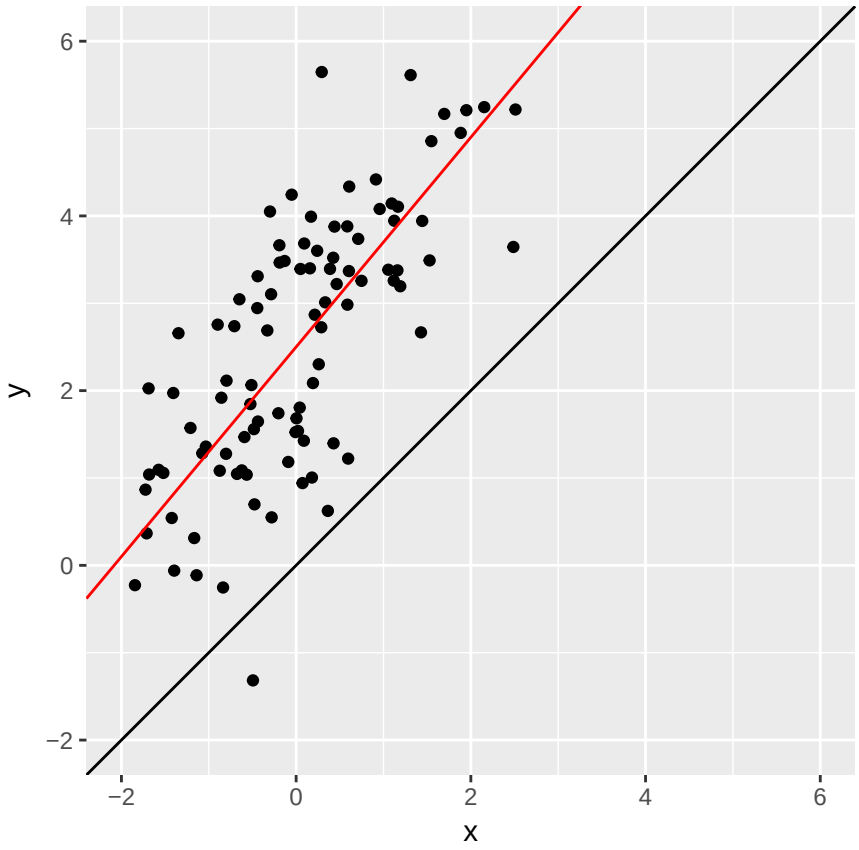
4.3 geom_abline

`geom_abline` allows you to display lines with a specific intercept and slope. If no intercept or slope is provided, a 45-degree line will be shown.

```
x = rnorm(100)
y = 2.5 + 1.2 * x + rnorm(100)
test_data <- data_frame(x, y)

test_data %>%
  ggplot(aes(x, y)) +
  geom_point() +
  xlim(-2, 6) + ylim(-2, 6) +
  coord_fixed() +
  geom_abline()
```



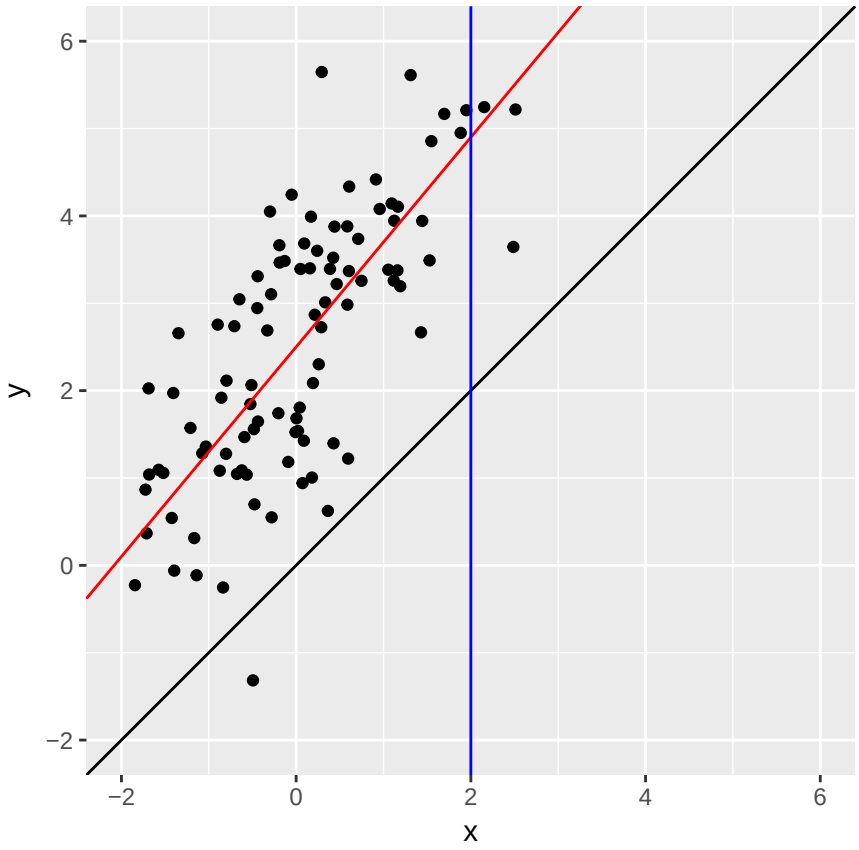

```
test_data %>%
  ggplot(aes(x, y)) +
  geom_point() +
  xlim(-2, 6) + ylim(-2, 6) +
  coord_fixed() +
  geom_abline() +
  geom_abline(intercept = 2.5, slope = 1.2, color = "red")
```



4.4 geom_vline

`geom_vline` allows you to draw vertical lines by specifying an x intercept.

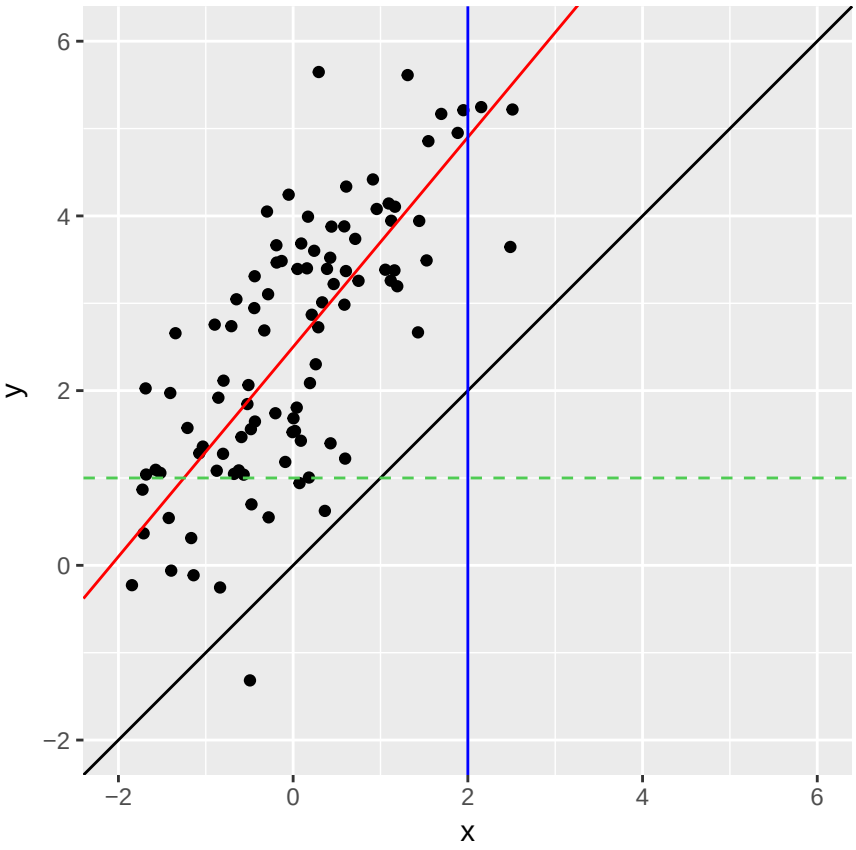
```
test_data %>%
  ggplot(aes(x, y)) +
  geom_point() +
  xlim(-2, 6) + ylim(-2, 6) +
  coord_fixed() +
  geom_abline() +
  geom_abline(intercept = 2.5, slope = 1.2, color = "red") +
  geom_vline(xintercept = 2, color = "blue")
```



4.5 hline

`geom_vline` allows you to draw vertical lines by specifying an x intercept.

```
test_data %>%
  ggplot(aes(x, y)) +
  geom_point() +
  xlim(-2, 6) + ylim(-2, 6) +
  coord_fixed() +
  geom_abline() +
  geom_abline(intercept = 2.5, slope = 1.2, color = "red") +
  geom_vline(xintercept = 2, color = "blue") +
  geom_hline(yintercept = 1, color = "#4FCC53", lty = 2)
```



4.6 Assignment

Create a visualization of the military data by branch (i.e., `Army`, `Navy`, etc.) using `facet_wrap()`. Plot both the points and a smooth line.

The data we have been working with is not yet tidy. Each row contains multiple observations (observations for Army, Navy, etc.). To make this tidy we should have one column with the personnel counts and one column that indicates the branch.

```
tidy_mil <- mil_personnel %>%
  gather(branch, personnel, -Year)
tidy_mil
```

```
## # A tibble: 315 x 3
##     Year branch personnel
##    <dbl>  <chr>     <dbl>
##  1  1953  Total     24785
##  2  1954  Total     23654
##  3  1955  Total     40258
##  4  1956  Total     37470
##  5  1957  Total     40683
##  6  1958  Total     35076
##  7  1959  Total     36310
##  8  1960  Total     35412
##  9  1961  Total     39474
## 10  1962  Total     41657
```

```
## # ... with 305 more rows
```

Chapter 5

Scatter Plot Matrices and Extensions

5.1 Data

For this section, we'll look at data from the American Community Survey (ACS) on immigration. To download the data,

1. Go to the American FactFinder website.
2. Click on the "Download Center" section, then click the "DOWNLOAD CENTER" button.
3. Click the "NEXT" button, since we know the table we want to download.
4. Select "American Community Survey" from the Program dropdown.
5. Select "2015 ACS 5-year estimates", click the "ADD TO YOUR SELECTIONS" button, then click "NEXT"
6. Select "County - 050" from the geographic type dropdown, then select "All Counties within United States", click the "ADD TO YOUR SELECTIONS" button, then click "NEXT"
7. Type `income mobility` in the "topic or table name"" search box, then select the option that reads:

"B07011: MEDIAN INCOME IN THE PAST 12 MONTHS (IN 2015 INFLATION-ADJUSTED DOLLARS) BY GEOGRAPHICAL MOBILITY IN THE PAST YEAR FOR CURRENT RESIDENCE IN THE UNITED STATES"

Click "GO", then check the checkbox beside the table we found. Now click on the "Download" button, and uncheck the option that says, "Include descriptive data element names." Click "Ok" to create your zip file. Once the file has been created, click "DOWNLOAD" to download the zip file.

The following will assume you moved the following files within the zip to your `data` folder:

- ACS_15_5YR_B07011_with_ann.csv
- ACS_15_5YR_B07011_metadata.csv

The file `ACS_15_5YR_B07011.txt` tells us how to interpret codes within our data. It is possible for median values to be followed by a `+` or `-` if they are in the upper or lower open-ended interval. In our dataset we don't have any medians in the upper open-ended interval, but we do have entries in the lower open-ended interval.

```
library(tidyverse)
```

```
acs <- read_csv("data/ACS_15_5YR_B07011_with_ann.csv", col_types = strrep("c", 15), na = c("-", "(X)"))
meta <- read_csv("data/ACS_15_5YR_B07011_metadata.csv")
meta
```

```
## # A tibble: 14 x 2
##               GEO.id
```

```
##                <chr>
##  1           GEO.id2
##  2 GEO.display-label
##  3         HD01_VD02
##  4         HD02_VD02
##  5         HD01_VD03
##  6         HD02_VD03
##  7         HD01_VD04
##  8         HD02_VD04
##  9         HD01_VD05
## 10         HD02_VD05
## 11         HD01_VD06
## 12         HD02_VD06
## 13         HD01_VD07
## 14         HD02_VD07
## # ... with 1 more variables: Id <chr>
```

Let's keep only the variables we care about, using more informative variable names.

```
acs_mobility <- acs %>%
  transmute(
    geo = `GEO.display-label`,
    same_house = HD01_VD03,
    same_county = HD01_VD04,
    same_state = HD01_VD05,
    same_country = HD01_VD06,
    different_country = HD01_VD07
  )
acs_mobility
```

```
## # A tibble: 1,949 x 6
##                         geo same_house same_county same_state same_country
##                       <chr>      <chr>       <chr>      <chr>        <chr>
##  1  Autauga County, Alabama      27553       18655      27643        35870
##  2  Barbour County, Alabama      17263       17363       8165        12667
##  3     Bibb County, Alabama      21489       16112       8804         <NA>
##  4   Butler County, Alabama      19499       11734      20769        23887
##  5 Chambers County, Alabama      20708       14522      21218        18516
##  6  Chilton County, Alabama      23668       20646      15739        33464
##  7     Clay County, Alabama      19201       18836      20596        11350
##  8 Cleburne County, Alabama      21888       11583      20380        16475
##  9   Coffee County, Alabama      24325       18957      11906        31702
## 10  Colbert County, Alabama      22419       18389      17330        20093
## # ... with 1,939 more rows, and 1 more variables: different_country <chr>
```

Now we can add an indicator for whether the median value is in the lowest available interval. This would mean that the median value presented has been bottom-coded.

```
acs_mobility <- acs_mobility %>%
  mutate(
    same_country_bc = grepl("[0-9]*-", same_country),
    different_country_bc = grepl("[0-9]*-", different_country)
  )
acs_mobility
```

```
## # A tibble: 1,949 x 8
```

```
##                         geo same_house same_county same_state same_country
##                       <chr>      <chr>       <chr>      <chr>        <chr>
##  1  Autauga County, Alabama      27553       18655      27643        35870
##  2  Barbour County, Alabama      17263       17363       8165        12667
##  3     Bibb County, Alabama      21489       16112       8804         <NA>
##  4   Butler County, Alabama      19499       11734      20769        23887
##  5 Chambers County, Alabama      20708       14522      21218        18516
##  6  Chilton County, Alabama      23668       20646      15739        33464
##  7     Clay County, Alabama      19201       18836      20596        11350
##  8 Cleburne County, Alabama      21888       11583      20380        16475
##  9   Coffee County, Alabama      24325       18957      11906        31702
## 10  Colbert County, Alabama      22419       18389      17330        20093
## # ... with 1,939 more rows, and 3 more variables: different_country <chr>,
## #   same_country_bc <lgl>, different_country_bc <lgl>
```

Let's see how many counties have observations that are bottom coded:

```
acs_mobility %>%
  summarize(same_country_bc = sum(same_country_bc), different_country_bc = sum(different_country_bc), co
```

```
## # A tibble: 1 x 3
##   same_country_bc different_country_bc counties
##             <int>                <int>    <int>
## 1              15                   64     1949
```

Let's see what the typical bottom-coded values are:

```
acs_mobility %>%
  filter(same_country_bc) %>%
  select(same_country) %>%
  table()
```

```
## .
## 2,500-
##     15
```

```
acs_mobility %>%
  filter(different_country_bc) %>%
  select(different_country) %>%
  table()
```

```
## .
## 2,500-
##     64
```

In both cases the bottom-coded interval is the range from zero to 2,500. Since this is a small number of counties given the entire range, let's simply set the bottom-coded values to equal the upper-bound of their interval (i.e., 2,500).

```
acs_mobility <- acs_mobility %>%
  transmute(
    geo = geo,
    same_house = if_else(grepl("[0-9]*-", same_house), 2500L, as.integer(same_house)),
    same_county = if_else(grepl("[0-9]*-", same_county), 2500L, as.integer(same_county)),
    same_state = if_else(grepl("[0-9]*-", same_state), 2500L, as.integer(same_state)),
    same_country = if_else(same_country_bc, 2500L, as.integer(same_country)),
    different_country = if_else(different_country_bc, 2500L, as.integer(different_country))
  )
```

```
acs_mobility
```

```
## # A tibble: 1,949 x 6
##                          geo same_house same_county same_state same_country
##                        <chr>      <int>       <int>      <int>        <int>
##  1  Autauga County, Alabama      27553       18655      27643        35870
##  2  Barbour County, Alabama      17263       17363       8165        12667
##  3     Bibb County, Alabama      21489       16112       8804           NA
##  4   Butler County, Alabama      19499       11734      20769        23887
##  5 Chambers County, Alabama      20708       14522      21218        18516
##  6  Chilton County, Alabama      23668       20646      15739        33464
##  7     Clay County, Alabama      19201       18836      20596        11350
##  8 Cleburne County, Alabama      21888       11583      20380        16475
##  9   Coffee County, Alabama      24325       18957      11906        31702
## 10  Colbert County, Alabama      22419       18389      17330        20093
## # ... with 1,939 more rows, and 1 more variables: different_country <int>
```

Let's rearrange the data into `tidy` format (one observation per row).

```
tidy_acs <- acs_mobility %>%
  gather(location_last_year, median_income, -geo, factor_key = TRUE)
tidy_acs
```

```
## # A tibble: 9,745 x 3
##                          geo location_last_year median_income
##                        <chr>             <fctr>         <int>
##  1  Autauga County, Alabama         same_house         27553
##  2  Barbour County, Alabama         same_house         17263
##  3     Bibb County, Alabama         same_house         21489
##  4   Butler County, Alabama         same_house         19499
##  5 Chambers County, Alabama         same_house         20708
##  6  Chilton County, Alabama         same_house         23668
##  7     Clay County, Alabama         same_house         19201
##  8 Cleburne County, Alabama         same_house         21888
##  9   Coffee County, Alabama         same_house         24325
## 10  Colbert County, Alabama         same_house         22419
## # ... with 9,735 more rows
```

5.2 ggplot2 extensions

There are many extensions the community have made that build on ggplot2. The following link provides a gallery of many of these extensions:

ggplot2 extensions

Some others that are usefull are `ggjoy` and `GGally`.

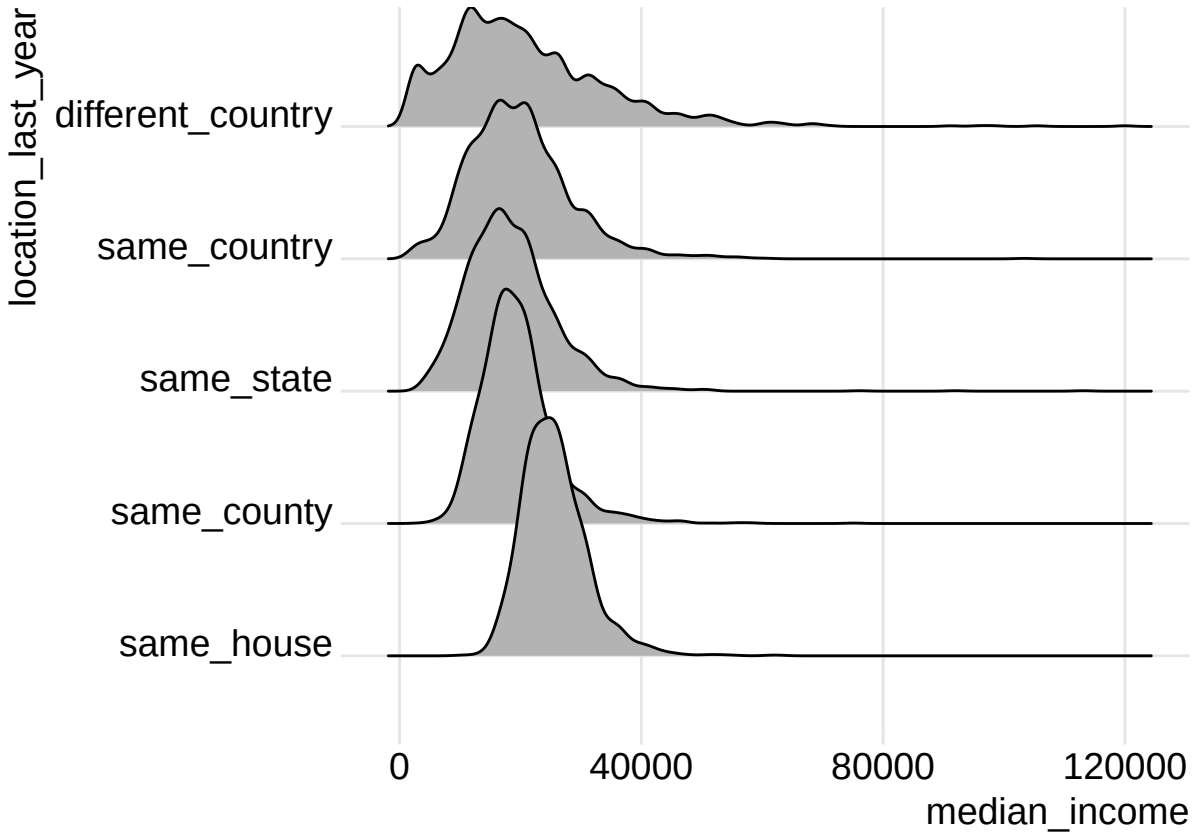
5.3 ggjoy

Make sure `ggjoy` is installed.

```
install.packages("ggjoy")
```

`ggjoy` gives us the ability to stack kernel density plots.

```
ggplot(tidy_acs, aes(x = median_income, y = location_last_year, group(location_last_year))) +
  ggjoy::geom_joy() +
  ggjoy::theme_joy()
```



This plot shows us that, on average, the distance moved in the past year is inversely related to median income.

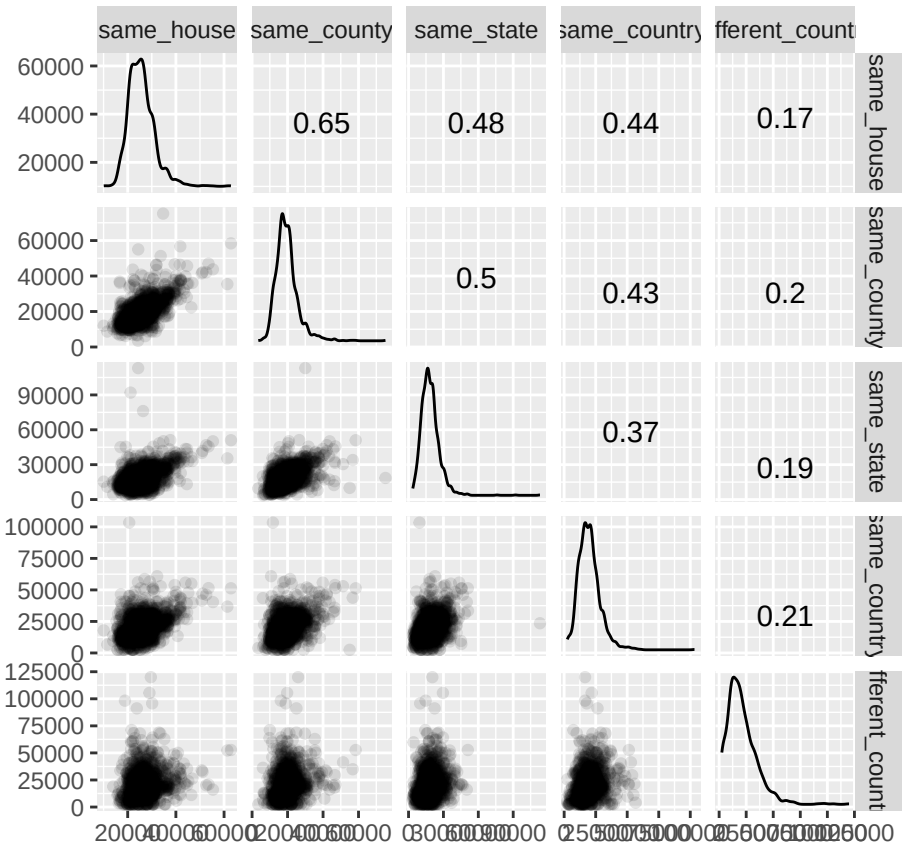
5.4 scatterplot matrix (GGally::ggscatmat)

Make sure you have `GGally` installed.

```
install.packages("GGally")
```

One particular library, GGally, has a great set of visualizations to extend those that come prebuilt with `ggplot`. One common visualization tool that is missing from ggplot is the scatterplot matrix. While base R provides `splom()` in the `lattice` library, `GGally::ggpairs` and `GGally::ggscatmat` pr ovide an easy tool to create a scatterplot matrix with ggplot2.

```
acs_mobility %>%
  as.data.frame() %>%
  GGally::ggscatmat(columns = 2:ncol(.), alpha = 0.1)
```



5.5 Assignment

Go to American FactFinder. Follow the steps above up until the point where we typed out "income mobility". This time pick another keyword to search for and select a different table to analyze (make sure this will give you more than two columns of data you would like to compare). Use `ggscatmat()` to visualize the variables that interest you. Keep the filenames as they are provided by the Census, so I can run your R Markdown file.

Chapter 6

Boxplots and Violin Plots

Boxplots and violin plots are two important tools for visualizing the distribution of data within a dataset. The boxplot highlights the median, key percentiles, and outliers within a dataset. The violin plot takes a kernel density plot, rotates it 90 degrees, then mirrors it about the axis to create a shape that sometimes resembles a violin.

6.1 Data

The Social Security Administration releases data on earnings and employment each year. We'll take a look at the data for 2014:

https://www.ssa.gov/policy/docs/statcomps/eedata_sc/2014/index.html

We're going to download Table 1: "Number of persons with Social Security (OASDI) taxable earnings, amount taxable, and contributions, by state or other area, sex, and type of earnings, 2014"

Save that file as 'ssa_earnings.xlsx' in the `data` folder

```
library(tidyverse)
library(readxl)
```

```
ssa <- read_xlsx("data/ssa_earnings.xlsx", range = "A7:J159",
                 col_names = c("state", "gender", "other", "other2", "number.total", "number.wage", "nu
                               "earnings.total", "earnings.wage", "earnings.self"))
ssa
```

```
## # A tibble: 153 x 10
##       state gender other other2 number.total number.wage number.self
##       <chr>  <chr> <lgl>  <lgl>        <dbl>       <dbl>       <dbl>
##  1  Alabama   <NA>    NA     NA      2355477     2215535      255253
##  2     <NA>    Men    NA     NA      1200468     1116458      138895
##  3     <NA>  Women    NA     NA      1155009     1099077      116357
##  4   Alaska   <NA>    NA     NA       400007      375833       47696
##  5     <NA>    Men    NA     NA       223464      209694       27884
##  6     <NA>  Women    NA     NA       176543      166140       19812
##  7  Arizona   <NA>    NA     NA      3189785     2997567      334292
##  8     <NA>    Men    NA     NA      1660088     1551488      185753
##  9     <NA>  Women    NA     NA      1529697     1446079      148539
## 10 Arkansas   <NA>    NA     NA      1468898     1376249      163320
## # ... with 143 more rows, and 3 more variables: earnings.total <dbl>,
```

```
## #   earnings.wage <dbl>, earnings.self <dbl>
```

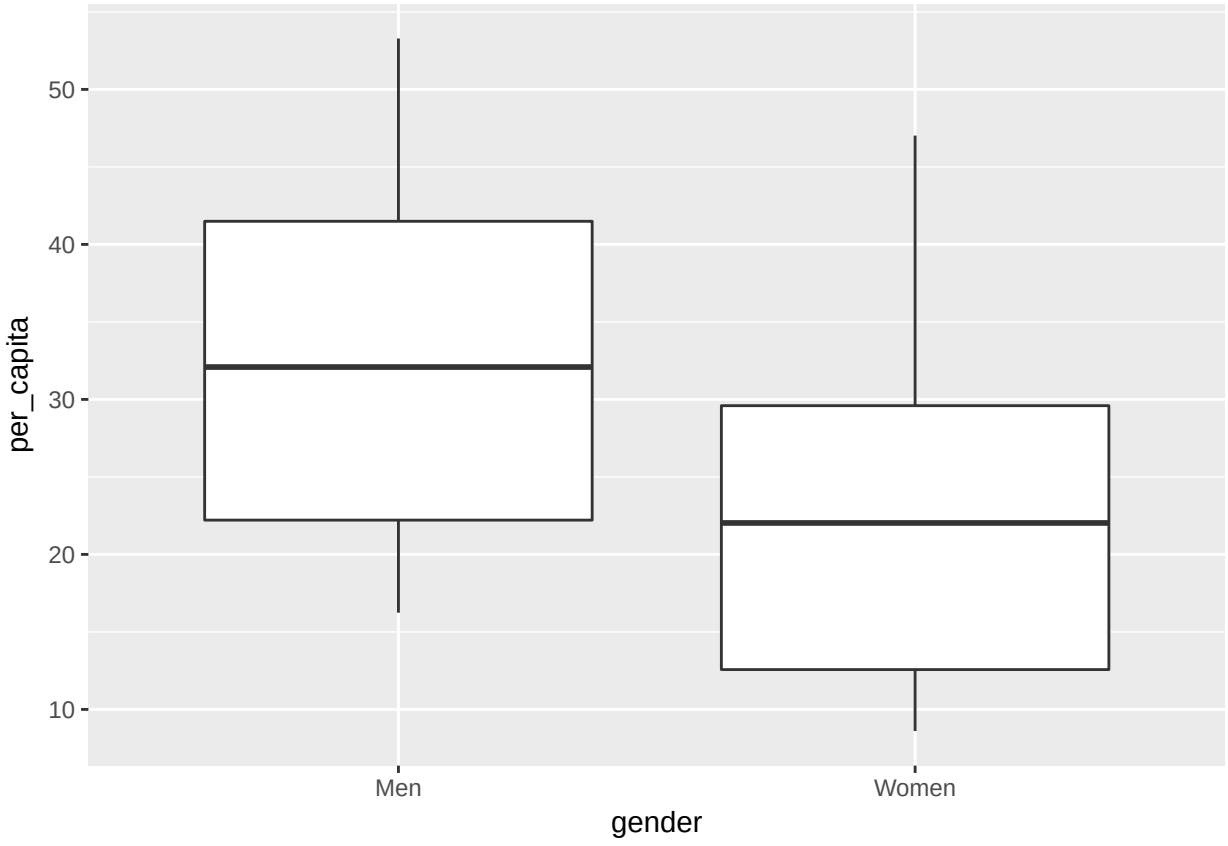
The starting format is far from ideal. Each row should represent one group, so we don't need any of the rows with totals.

It's important to always read any footnotes and documentation that comes with the data you plan to use. Footnote `c` for this table indicates that individuals with both wage and salary employment will be counted in both groups, but only once in the total. It is important to be aware of this double counting.

```
ssa_long <- ssa %>%
  fill(state) %>%
  filter(!is.na(gender)) %>%
  reshape(varying = 5:10, direction = "long", timevar = "earnings_type") %>%
  select(state, gender, earnings_type, number, earnings) %>%
  mutate(per_capita = earnings / number)
```

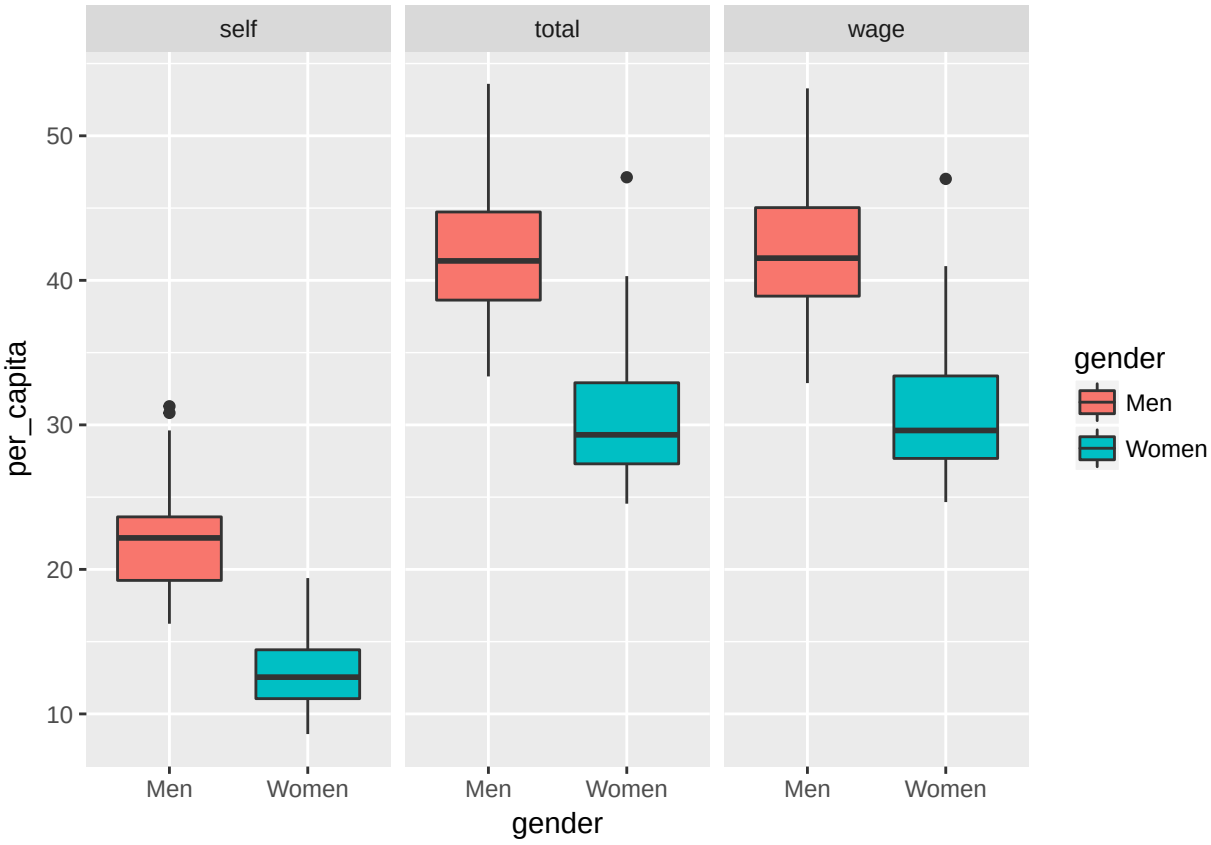
6.2 Boxplots

```
ssa_long %>%
  filter(earnings_type != "total") %>%
  ggplot(aes(gender, per_capita)) +
  geom_boxplot()
```




```
ssa_long %>%
  ggplot(aes(gender, per_capita, fill = gender)) +
  geom_boxplot() +
```

```
  facet_grid(~ earnings_type)
```

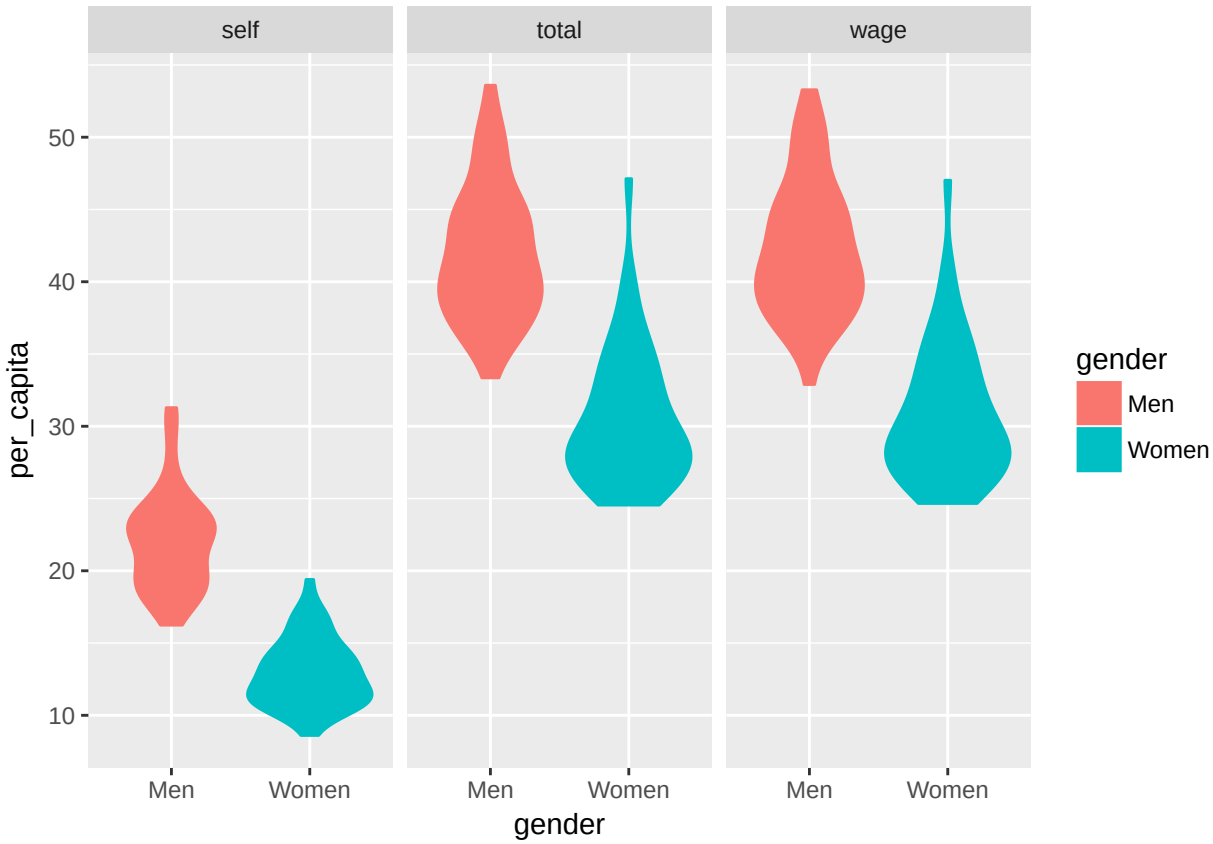


6.3 Violin Plots

Let's repeat the above plots using the violin plot type.

```
ssa_long %>%
  filter(earnings_type != "total") %>%
  ggplot(aes(gender, per_capita)) +
  geom_violin()
```

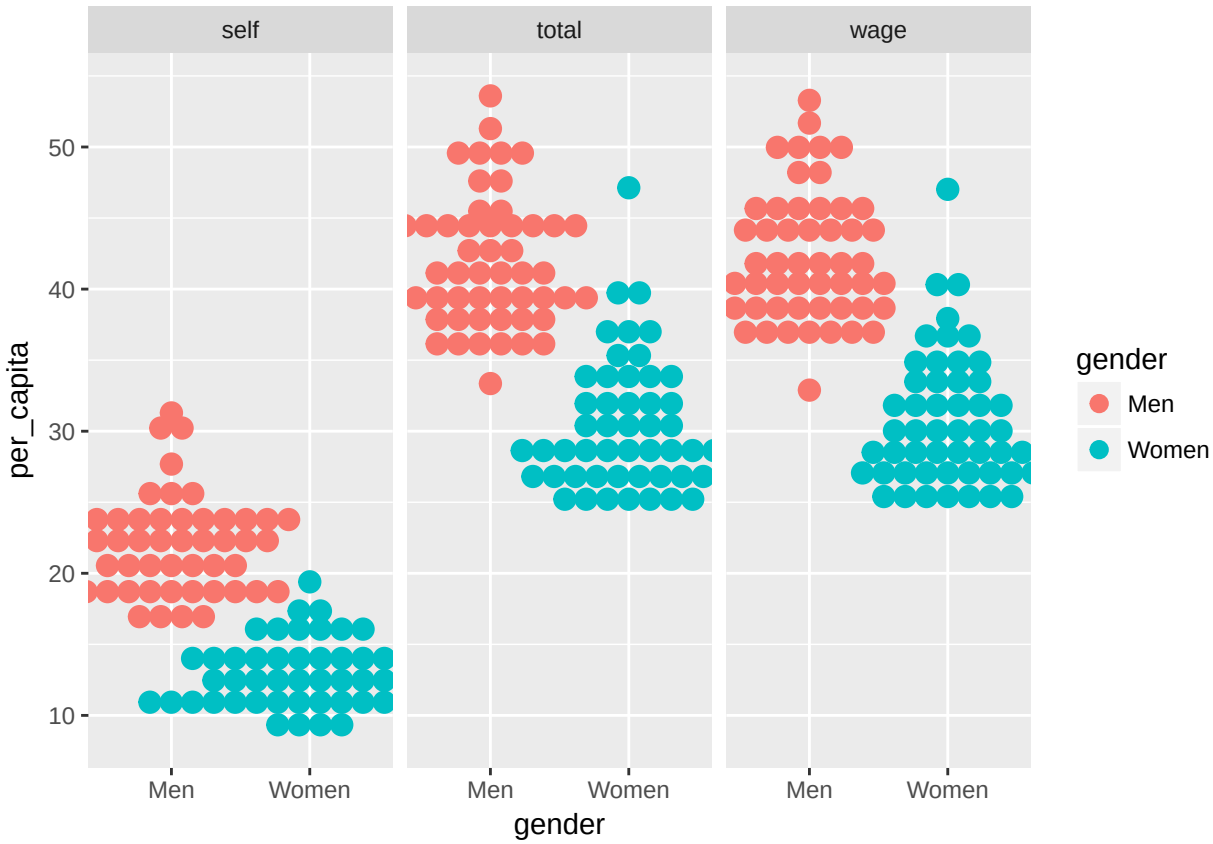

```
ssa_long %>%
  ggplot(aes(gender, per_capita, color = gender, fill = gender)) +
  geom_violin() +
  facet_grid(~ earnings_type)
```



6.4 Dot Plots

Dot plots appear similar to violin plots, but dot plots may be easier to interpret:

```
ssa_long %>%
  ggplot(aes(gender, per_capita, color = gender, fill = gender)) +
  geom_dotplot(binaxis = "y", stackdir = "center", position = "dodge") +
  facet_grid(~ earnings_type)
```



6.5 Assignment

Create your own visualizations of the distribution of the `earnings` and `number` variables.

Chapter 7

Spatial Visualizations

7.1 Data

The data for this class will come from the National Oceanic and Atmospheric Administration (NOAA) U.S. Wind Climatology datasets (https://www.ncdc.noaa.gov/societal-impacts/wind/).

Download the files for both the u-component and the v-component of the wind data. To open these files in R, we'll need to install the ncdf4 package, which provides an interface to Unidata's netCDF data file format:

```
install.packages(c("ncdf4", "ncdf4.helpers", "PCICt"))
```

Let's load up the u-component file first:

```
library(ncdf4)

uwnd_nc <- nc_open("data/uwnd.sig995.2017.nc")
uwnd_nc
```

```
## File data/uwnd.sig995.2017.nc (NC_FORMAT_NETCDF4_CLASSIC):
##
##      2 variables (excluding dimension variables):
##         float uwnd[lon,lat,time]
##             long_name: mean Daily u-wind at sigma level 995
##             units: m/s
##             precision: 2
##             least_significant_digit: 1
##             GRIB_id: 33
##             GRIB_name: UGRD
##             var_desc: u-wind
##             dataset: NCEP Reanalysis Daily Averages
##             level_desc: Surface
##             statistic: Mean
##             parent_stat: Individual Obs
##             missing_value: -9.96920996838687e+36
##             valid_range: -102.199996948242
##              valid_range: 102.199996948242
##             actual_range: -26.9250011444092
##              actual_range: 29.8999996185303
##         double time_bnds[nbnds,time]
##
```

```
##      4 dimensions:
##         lat  Size:73
##             units: degrees_north
##             actual_range: 90
##              actual_range: -90
##             long_name: Latitude
##             standard_name: latitude
##             axis: Y
##         lon  Size:144
##             units: degrees_east
##             long_name: Longitude
##             actual_range: 0
##              actual_range: 357.5
##             standard_name: longitude
##             axis: X
##         time  Size:198   *** is unlimited ***
##             long_name: Time
##             delta_t: 0000-00-01 00:00:00
##             standard_name: time
##             axis: T
##             units: hours since 1800-01-01 00:00:0.0
##             avg_period: 0000-00-01 00:00:00
##             coordinate_defines: start
##             actual_range: 1902192
##              actual_range: 1906920
##         nbnds  Size:2
##
##     7 global attributes:
##         Conventions: COARDS
##         title: mean daily NMC reanalysis (2014)
##         history: created 2013/12 by Hoop (netCDF2.3)
##         description: Data is from NMC initialized reanalysis
## (4x/day).  These are the 0.9950 sigma level values.
##         platform: Model
##         References: http://www.esrl.noaa.gov/psd/data/gridded/data.ncep.reanalysis.html
##         dataset_title: NCEP-NCAR Reanalysis 1
```

Let's store the uwnd observations in the netCDF file for the u-component:

```
library(ncdf4.helpers)
library(PCICt)
```

```
## Loading required package: methods
```

```
uwnd <- ncvar_get(uwnd_nc, "uwnd")
uwnd_time <- nc.get.time.series(uwnd_nc, v = "uwnd", time.dim.name = "time")
uwnd_lon <- ncvar_get(uwnd_nc, "lon")
uwnd_lat <- ncvar_get(uwnd_nc, "lat")
nc_close(uwnd_nc)

library(tidyverse)
```

```
## Loading tidyverse: ggplot2
## Loading tidyverse: tibble
## Loading tidyverse: tidyr
## Loading tidyverse: readr
```

```
## Loading tidyverse: purrr
## Loading tidyverse: dplyr

## Conflicts with tidy packages ----------------------------------------------

## filter(): dplyr, stats
## lag():    dplyr, stats
```

```
uwnd_df <- uwnd %>%
  as.data.frame.table(responseName = "uwnd", stringsAsFactors = FALSE) %>%
  rename(a = Var1, b = Var2, c = Var3) %>%
  cbind.data.frame(expand.grid(uwnd_lon, uwnd_lat, uwnd_time)) %>%
  rename(lon = Var1, lat = Var2, time = Var3) %>%
  dplyr::select(lon, lat, time, uwnd)
uwnd_df %>% as.tibble()
```

```
## # A tibble: 2,081,376 x 4
##      lon   lat        time        uwnd
##    <dbl> <dbl> <S3: PCICt>       <dbl>
##  1   0.0    90  2017-01-01 -2.29999971
##  2   2.5    90  2017-01-01 -1.99999964
##  3   5.0    90  2017-01-01 -1.69999957
##  4   7.5    90  2017-01-01 -1.34999967
##  5  10.0    90  2017-01-01 -1.02499962
##  6  12.5    90  2017-01-01 -0.72499961
##  7  15.0    90  2017-01-01 -0.39999962
##  8  17.5    90  2017-01-01 -0.04999962
##  9  20.0    90  2017-01-01  0.27500039
## 10  22.5    90  2017-01-01  0.60000038
## # ... with 2,081,366 more rows
```

Now we need to do the same for the v-component of the wind vectors. Since we know the lat, lon, and time dimensions are repeated, we can join directly to the previous data.frame:

```
vwnd_nc <- nc_open("data/vwnd.sig995.2017.nc")
vwnd <- ncvar_get(vwnd_nc, "vwnd")
vwnd_time <- nc.get.time.series(vwnd_nc, v = "vwnd", time.dim.name = "time")
vwnd_lon <- ncvar_get(vwnd_nc, "lon")
vwnd_lat <- ncvar_get(vwnd_nc, "lat")
nc_close(vwnd_nc)

wind <- vwnd %>%
  as.data.frame.table(responseName = "vwnd", stringsAsFactors = FALSE) %>%
  cbind.data.frame(uwnd_df) %>%
  rename(lon2 = Var1, lat2 = Var2, time2 = Var3) %>%
  select(lon, lat, time, vwnd, uwnd)
wind %>% as.tibble()
```

```
## # A tibble: 2,081,376 x 5
##      lon   lat        time     vwnd        uwnd
##    <dbl> <dbl> <S3: PCICt>    <dbl>       <dbl>
##  1   0.0    90  2017-01-01 7.150002 -2.29999971
##  2   2.5    90  2017-01-01 7.250002 -1.99999964
##  3   5.0    90  2017-01-01 7.350002 -1.69999957
##  4   7.5    90  2017-01-01 7.375001 -1.34999967
##  5  10.0    90  2017-01-01 7.475002 -1.02499962
##  6  12.5    90  2017-01-01 7.475002 -0.72499961
```

```
##  7  15.0    90  2017-01-01 7.525002 -0.39999962
##  8  17.5    90  2017-01-01 7.550002 -0.04999962
##  9  20.0    90  2017-01-01 7.550002  0.27500039
## 10  22.5    90  2017-01-01 7.525002  0.60000038
## # ... with 2,081,366 more rows
```

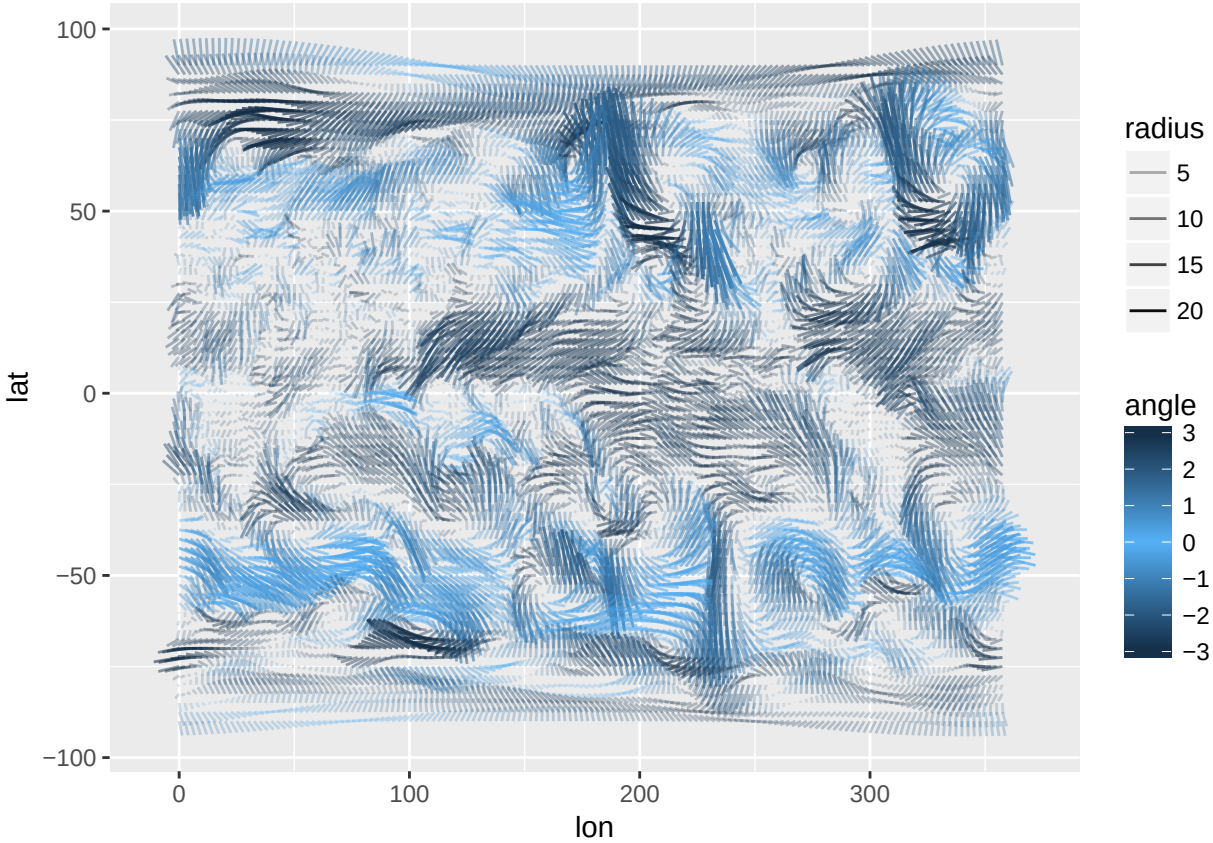
Otherwise, we would need to merge these data.frames to get `uwnd` and `vwnd` together with the following, which takes long time to run:

```
wind <- merge(uwnd_df, vwnd_df)
```

7.2 geom_spoke

To represent these wind vectors we'll use the `geom_spoke()`. We'll start just plotting wind patterns for January 1, 2017:

```
wind <- wind %>%
  mutate(angle = atan2(vwnd, uwnd), radius = sqrt(uwnd^2 + vwnd^2), time = as.POSIXct(time))
wind %>%
  filter(time == as.POSIXct("2017-01-01", tz = "GMT")) %>%
  ggplot(aes(lon, lat)) +
  geom_spoke(aes(angle = angle, radius = radius, alpha = radius, color = angle)) +
  scale_color_gradient2(low = "#132B43", mid = "#56B1F7", high = "#132B43")
```

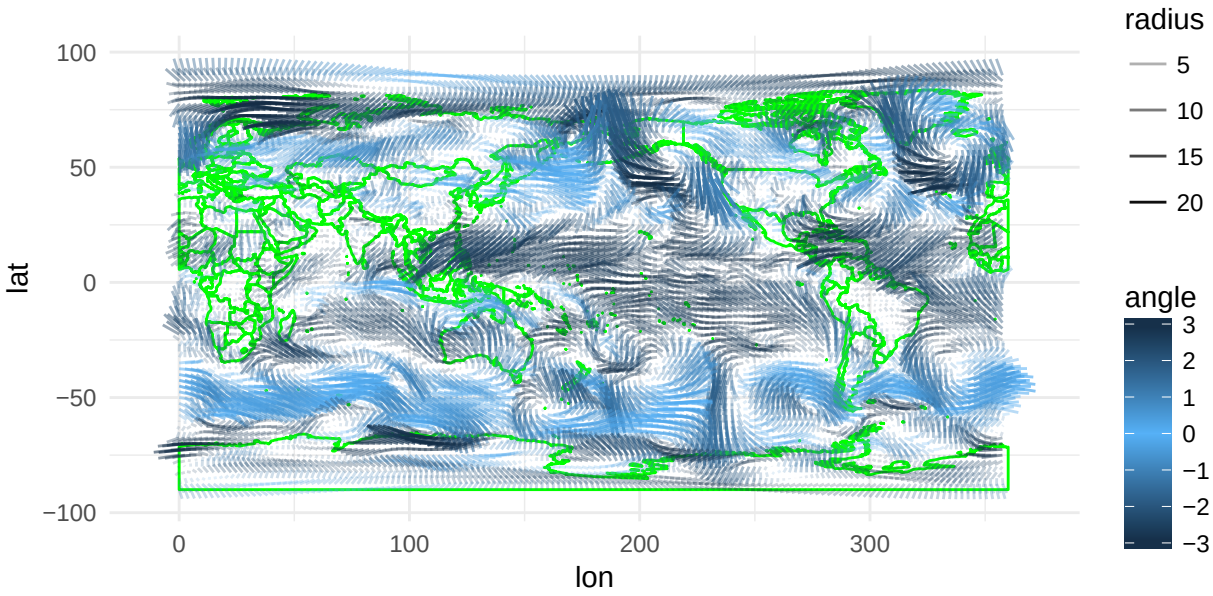


7.3 maps

```
install.packages("maps")
```

Map data will help to provide some context to this wind figure. We'll use `geom_polygon` to plot the world centered on the Pacific Ocean (`world2`) using the `map_data()` function.

```
world <- map_data("world2")
wind %>%
  filter(time == as.POSIXct("2017-01-01", tz = "GMT")) %>%
  ggplot(aes(lon, lat)) +
  geom_polygon(data = world, aes(x=long, y = lat, group = group), color = "green", fill = NA) +
  coord_fixed(1) +
  geom_spoke(aes(angle = angle, radius = radius, alpha = radius, color = angle)) +
  scale_color_gradient2(low = "#132B43", mid = "#56B1F7", high = "#132B43") +
  theme_minimal()
```



7.4 gganimate

The `gganimate` package lets us animate the above chart. If you want to be able to save animations as an mp4, you will need install `ffmpeg` (https://www.ffmpeg.org/download.html). If you are running macOS, you will need also need ImageMagick (http://www.imagemagick.org/script/binary-releases.php#macosx).

You can install `gganimate` with `devtools`:

```
devtools::install_github("dgrtwo/gganimate")
```

```
library(gganimate)
f <- wind %>%
  ggplot(aes(lon, lat)) +
  geom_polygon(data = world, aes(x=long, y = lat, group = group), color = "green", fill = NA) +
  coord_fixed(1) +
  geom_spoke(aes(angle = angle, radius = radius, alpha = radius, color = angle, frame = time)) +
  scale_color_gradient2(low = "#132B43", mid = "#56B1F7", high = "#132B43") +
```

```
  theme_minimal()
gganimate(f)
```

7.5 glyphs

`glyphs` provide another useful way of analyzing spatial data with a time dimesion. This shows a tiny line charts representing the north-south component of the wind at each longitude/latitude combination.

```
library(GGally)
wind$day <- as.numeric(julian(wind$time, as.POSIXct("2017-01-01", tz = "GMT")))
wind$day_flip <- -wind$day
vwnd_gly <- glyphs(wind, "lon", "day", "lat", "vwnd", height=2.5)
uwnd_gly <- glyphs(wind, "lon", "day", "lat", "uwnd", height=2.5)

ggplot(vwnd_gly, aes(gx, gy, group = gid)) +
  add_ref_lines(vwnd_gly, color = "grey90") +
  add_ref_boxes(vwnd_gly, color = "grey90") +
  geom_path() +
  theme_bw() +
  labs(x = "", y = "")
```

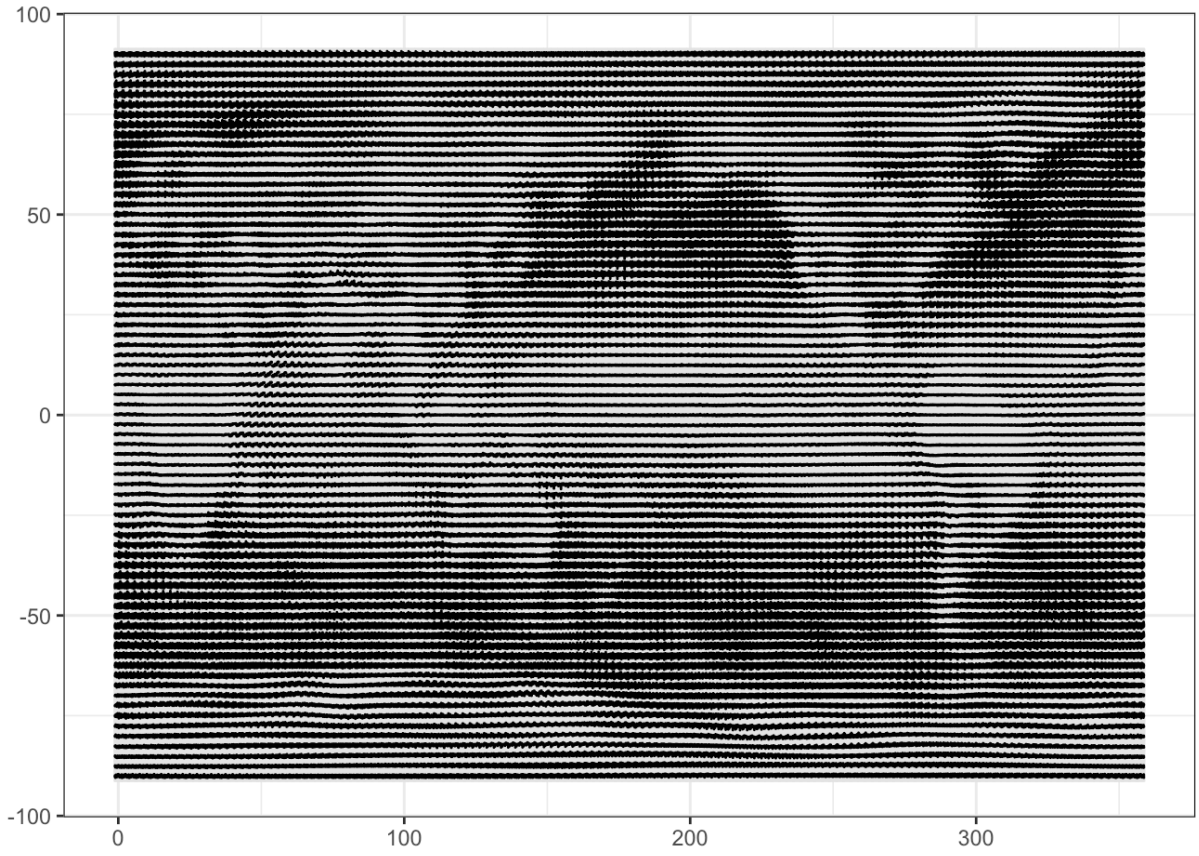
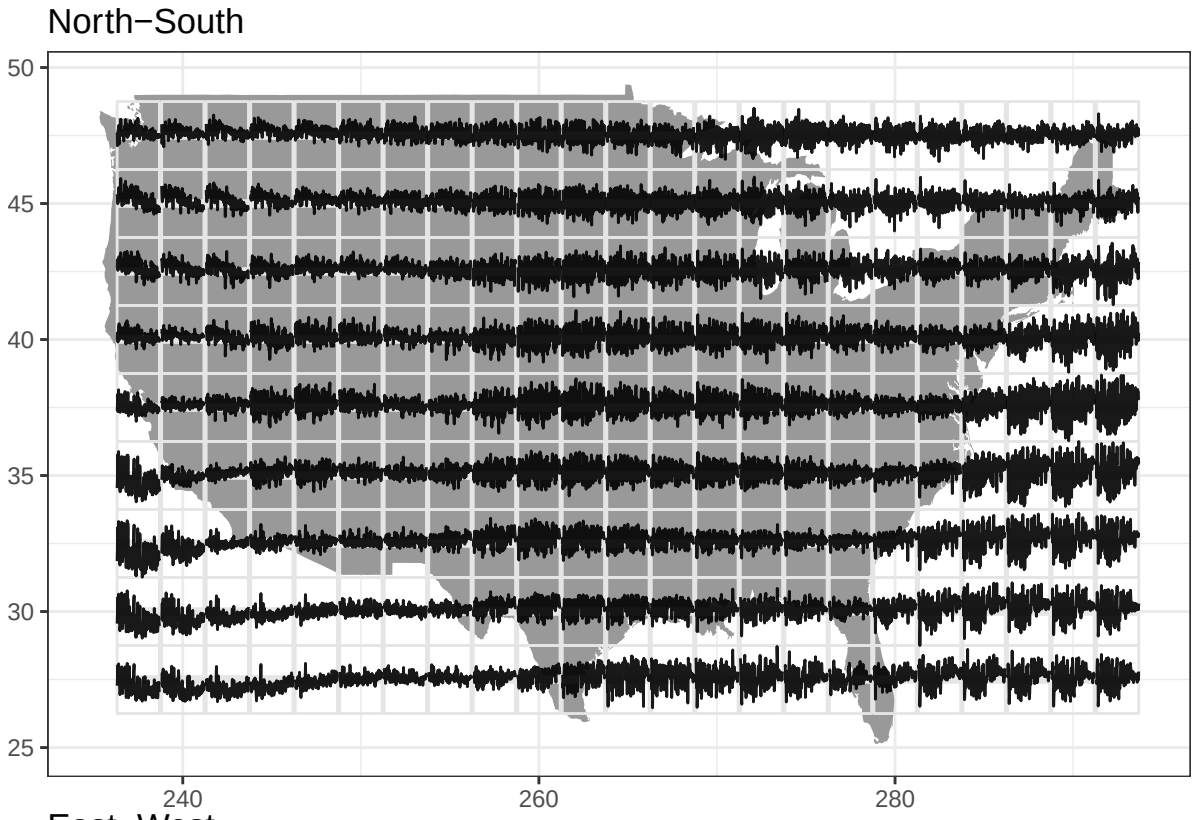
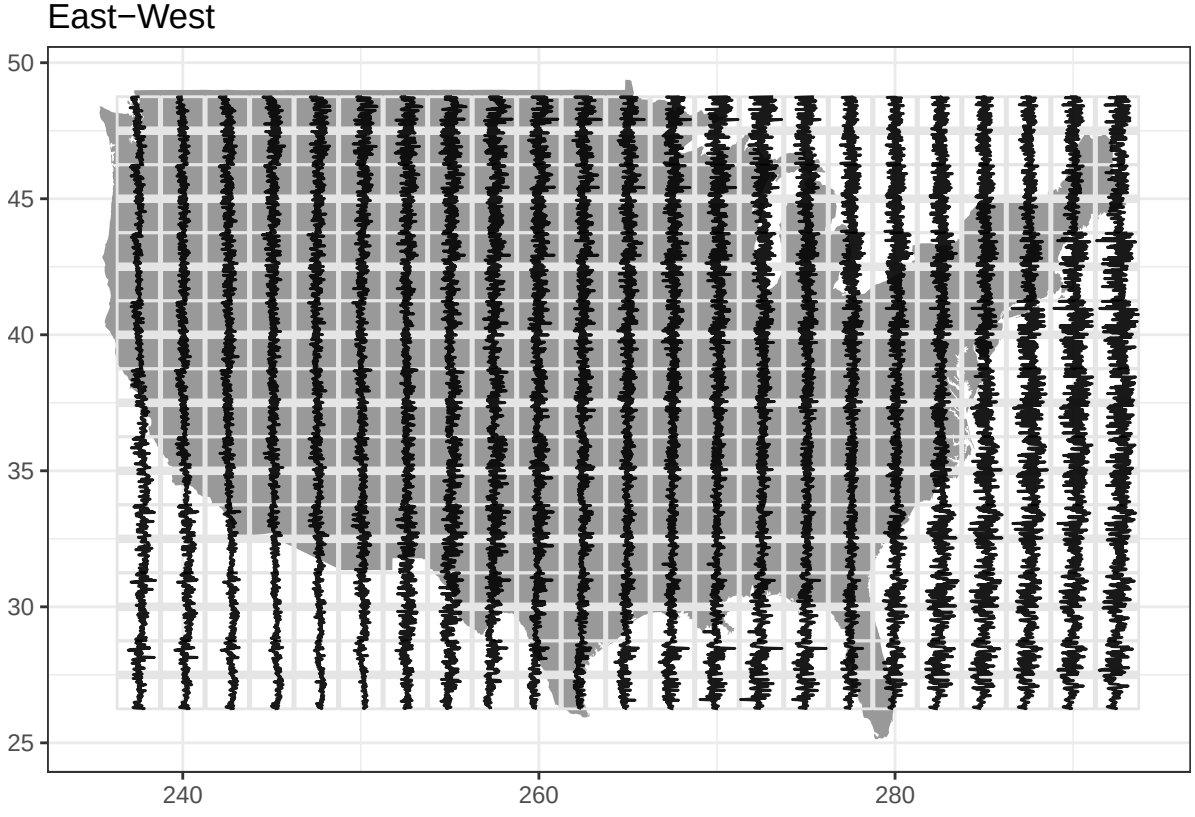


Figure 7.1:

Let's focus in on just the continental US:

```
library(GGally)

usa <- map_data("usa")
usa_long_range <- range(usa$long)
usa_lat_range <- range(usa$lat)

usa_wind <- wind %>%
  filter(lon >= (usa_long_range[1] %% 360) & lon <= (usa_long_range[2] %% 360) &
           lat >= usa_lat_range[1] & lat <= usa_lat_range[2])
usa_wind$day <- as.numeric(julian(usa_wind$time, as.POSIXct("2017-01-01", tz = "GMT")))
usa_wind$day_flip <- -usa_wind$day
usa_vwnd_gly <- glyphs(usa_wind, "lon", "day", "lat", "vwnd", height=2.5)
usa_uwnd_gly <- glyphs(usa_wind, "lon", "uwnd", "lat", "day_flip", height=2.5)

ggplot(usa_vwnd_gly, aes(gx, gy, group = gid)) +
  geom_polygon(data = usa, aes(x = long %% 360, y = lat %% 360, group = group), fill = "grey60") +
  add_ref_lines(usa_vwnd_gly, color = "grey90") +
  add_ref_boxes(usa_vwnd_gly, color = "grey90") +
  geom_path(alpha = 0.9) +
  theme_bw() +
  labs(x = "", y = "", title = "North-South")
ggplot(usa_uwnd_gly, aes(gx, gy, group = gid)) +
  geom_polygon(data = usa, aes(x = long %% 360, y = lat %% 360, group = group), fill = "grey60") +
  add_ref_lines(usa_uwnd_gly, color = "grey90") +
  add_ref_boxes(usa_uwnd_gly, color = "grey90") +
  geom_path(alpha = 0.9) +
  theme_bw() +
  labs(x = "", y = "", title = "East-West")
```


North−South
50
45
40
35
30
25
240
260
280


East−West
50
45
40
35
30
25
240
260
280

7.6 Assignment

Create heatmaps of `uwnd` and `vwnd` values on March 31, 2017. Each heatmap should be 90 degrees longitude by 90 degrees lattitude. Hint: Use `facet_grid` and create two new variables to help with faceting. The plot should end up being 5 facets wide by 3 facets tall.

Chapter 8

geom_area and geom_ribbon

8.1 Data

The US Bureau of Labor Statistics (BLS) conducts the American Time Use Survey (ATUS). You can download the text form of the ATUS by going to the BLS data page, finding the section labelled Spending & Time Use, then clicking on the "Text Files" button on the row for the ATUS. Or by using the following link:

https://download.bls.gov/pub/time.series/tu/

8.1.1 Downloading a file from the internet

While you can manually download the files from the above URL, `download.file()` lets you download files from within R. The first argument is the URL of the resource you want to download. The second argument is the destination for the file. The following requests will require you to create the `tu` folder.

```
download.file("https://download.bls.gov/pub/time.series/tu/tu.txt", "data/tu/tu.txt")
download.file("https://download.bls.gov/pub/time.series/tu/tu.series", "data/tu/tu.series")
download.file("https://download.bls.gov/pub/time.series/tu/tu.data.0.Current", "data/tu/tu.data.0.Curre
```

The file `tu.txt` contains the documentation for the time use (tu) survey data. Section 2 of that file provides descriptions of each of the files in the `pub/time.series/tu` folder. From that list we can see that `tu.series` will give us a list of the available series.

```
library(readr)
series_defn <- read_tsv("data/tu/tu.series")
series_defn
```

```
## # A tibble: 85,277 x 43
##             series_id seasonal stattype_code datays_code sex_code
##                 <chr>    <chr>         <int>       <chr>    <int>
##  1 TUU10100AA01000007        U         10100          01        0
##  2 TUU10100AA01000013        U         10100          01        0
##  3 TUU10100AA01000014        U         10100          01        0
##  4 TUU10100AA01000015        U         10100          01        0
##  5 TUU10100AA01000018        U         10100          01        0
##  6 TUU10100AA01000019        U         10100          01        0
##  7 TUU10100AA01000025        U         10100          01        0
##  8 TUU10100AA01000035        U         10100          01        0
##  9 TUU10100AA01000036        U         10100          01        0
```

```
## 10 TUU10100AA01000037        U         10100          01        0
## # ... with 85,267 more rows, and 38 more variables: region_code <chr>,
## #   lfstat_code <chr>, educ_code <chr>, maritlstat_code <chr>,
## #   age_code <chr>, orig_code <chr>, race_code <chr>, mjcow_code <chr>,
## #   nmet_code <int>, where_code <chr>, sjmj_code <int>,
## #   timeday_code <chr>, actcode_code <chr>, industry_code <chr>,
## #   occ_code <chr>, prhhchild_code <chr>, earn_code <chr>,
## #   disability_code <chr>, who_code <chr>, hhnscc03_code <chr>,
## #   schenr_code <int>, prownhhchild_code <chr>, work_code <int>,
## #   elnum_code <chr>, ecage_code <chr>, elfreq_code <int>,
## #   eldur_code <chr>, elwho_code <chr>, ecytd_code <int>,
## #   elder_code <int>, lfstatw_code <chr>, pertype_code <chr>,
## #   series_title <chr>, footnote_codes <chr>, begin_year <int>,
## #   begin_period <chr>, end_year <int>, end_period <chr>
```

There is a lot here to process. The columns we care most about for now are `series_id` and `series_title`. Using `select()` from the `dplyr` library, we can show just the columns we care about.

```
library(dplyr)
series_defn %>%
  select(series_id, series_title)
```

```
## # A tibble: 85,277 x 2
##             series_id
##                 <chr>
##  1 TUU10100AA01000007
##  2 TUU10100AA01000013
##  3 TUU10100AA01000014
##  4 TUU10100AA01000015
##  5 TUU10100AA01000018
##  6 TUU10100AA01000019
##  7 TUU10100AA01000025
##  8 TUU10100AA01000035
##  9 TUU10100AA01000036
## 10 TUU10100AA01000037
## # ... with 85,267 more rows, and 1 more variables: series_title <chr>
```

8.1.2 Pairing down the list of variables

Let's look for variables on sleep, work, and leisure:

```
series_defn %>%
  select(series_title) %>%
  filter(grepl("sleep", series_title, ignore.case = TRUE))
```

```
## # A tibble: 1,310 x 1
##                                                                 series_title
##                                                                        <chr>
##  1                                                 Avg hrs per day - Sleeping
##  2                       Avg hrs per day - Sleeping, Weekend days and holidays
##  3                            Avg hrs per day - Sleeping, Nonholiday weekdays
##  4                                      Avg hrs per day - Sleeping, Employed
##  5             Avg hrs per day - Sleeping, Weekend days and holidays, Employed
##  6                   Avg hrs per day - Sleeping, Nonholiday weekdays, Employed
##  7                        Avg hrs per day - Sleeping, Employed, on days worked
```

```
##  8 Avg hrs per day - Sleeping, Weekend days and holidays, Employed, on days wo
##  9   Avg hrs per day - Sleeping, Nonholiday weekdays, Employed, on days worked
## 10                              Avg hrs per day - Sleeping, Employed full time
## # ... with 1,300 more rows
```

Since this simple search returns a ton of results, let's further filter by 'employed' and 'per day':

```
series_defn %>%
  select(series_title) %>%
  filter(grepl("per day.*sleep.*employed", series_title, ignore.case = TRUE))
```

```
## # A tibble: 154 x 1
##                                                                  series_title
##                                                                         <chr>
##  1                                        Avg hrs per day - Sleeping, Employed
##  2             Avg hrs per day - Sleeping, Weekend days and holidays, Employed
##  3                   Avg hrs per day - Sleeping, Nonholiday weekdays, Employed
##  4                        Avg hrs per day - Sleeping, Employed, on days worked
##  5 Avg hrs per day - Sleeping, Weekend days and holidays, Employed, on days wo
##  6   Avg hrs per day - Sleeping, Nonholiday weekdays, Employed, on days worked
##  7                              Avg hrs per day - Sleeping, Employed full time
##  8   Avg hrs per day - Sleeping, Weekend days and holidays, Employed full time
##  9         Avg hrs per day - Sleeping, Nonholiday weekdays, Employed full time
## 10              Avg hrs per day - Sleeping, Employed full time, on days worked
## # ... with 144 more rows
```

Now let's filter further by 'employed full time', 'nonholiday weekdays', and 'on days worked':

```
series_defn %>%
  select(series_title) %>%
  filter(grepl("per day.*sleep.*nonholiday weekdays.*employed full time.*on days worked", series_title,
```

```
## # A tibble: 6 x 1
##                                                                    series_title
##                                                                           <chr>
## 1 Avg hrs per day - Sleeping, Nonholiday weekdays, Employed full time, on day
## 2 Avg hrs per day - Sleeping, Nonholiday weekdays, Employed full time, on day
## 3 Avg hrs per day - Sleeping, Nonholiday weekdays, Employed full time, on day
## 4 Avg hrs per day for participants - Sleeping, Nonholiday weekdays, Employed
## 5 Avg hrs per day for participants - Sleeping, Nonholiday weekdays, Employed
## 6 Avg hrs per day for participants - Sleeping, Nonholiday weekdays, Employed
```

Finally, let's filter that to exclude the 'participants only' group and only get the Men/Women values (not the combined totals):

```
series_defn %>%
  select(series_title) %>%
  filter(grepl("per day -.*sleep.*nonholiday weekdays.*employed full time.*on days worked,", series_titl
```

```
## # A tibble: 2 x 1
##                                                                    series_title
##                                                                           <chr>
## 1 Avg hrs per day - Sleeping, Nonholiday weekdays, Employed full time, on day
## 2 Avg hrs per day - Sleeping, Nonholiday weekdays, Employed full time, on day
```

8.1.3 Adding more activity categories

Now let's add 'work' and 'leisure' to our search:

```
activity <- series_defn %>%
  select(series_id, series_title) %>%
  filter(grepl("per day -.*(sleep|work|leisure).*nonholiday weekdays.*employed full time.*on days worke
activity
```

```
## # A tibble: 26 x 2
##             series_id
##                 <chr>
##  1 TUU10101AA01000344
##  2 TUU10101AA01000423
##  3 TUU10101AA01000962
##  4 TUU10101AA01001041
##  5 TUU10101AA01003012
##  6 TUU10101AA01003097
##  7 TUU10101AA01003307
##  8 TUU10101AA01003378
##  9 TUU10101AA01003947
## 10 TUU10101AA01004011
## # ... with 16 more rows, and 1 more variables: series_title <chr>
```

Now we should create a variable that codes each of these as either work, sleep, or leisure:

```
activity <- activity %>%
  mutate(
    activity_type = case_when(
      grepl("leisure", activity$series_title, ignore.case = TRUE) ~ "Leisure",
      grepl("sleep", activity$series_title, ignore.case = TRUE) ~ "Sleep",
      TRUE ~ "Work"
    ),
    sex = ifelse(grepl("Men", series_title), "Men", "Women")
  )
activity
```

```
## # A tibble: 26 x 4
##             series_id
##                 <chr>
##  1 TUU10101AA01000344
##  2 TUU10101AA01000423
##  3 TUU10101AA01000962
##  4 TUU10101AA01001041
##  5 TUU10101AA01003012
##  6 TUU10101AA01003097
##  7 TUU10101AA01003307
##  8 TUU10101AA01003378
##  9 TUU10101AA01003947
## 10 TUU10101AA01004011
## # ... with 16 more rows, and 3 more variables: series_title <chr>,
## #   activity_type <chr>, sex <chr>
```

Now we can join the activity data.frame with the current data and create time series of each activity type we created.

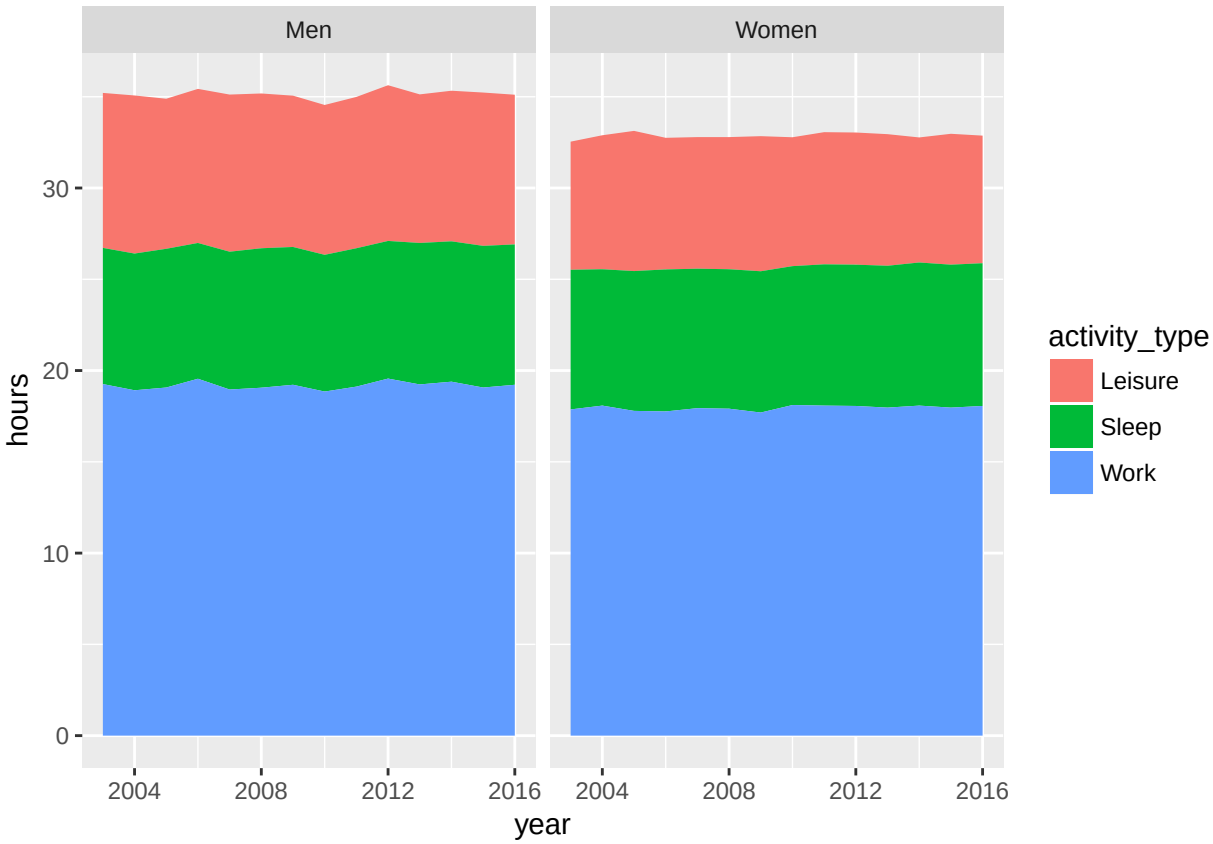
```
data <- read_tsv("data/tu/tu.data.0.Current")
data <- data %>%
  inner_join(activity) %>%
  group_by(year, sex, activity_type) %>%
  summarize(hours = sum(as.numeric(value), na.rm = TRUE))
data
```

```
## # A tibble: 84 x 4
## # Groups:   year, sex [?]
##     year   sex activity_type hours
##    <int> <chr>         <chr> <dbl>
##  1  2003   Men       Leisure  8.49
##  2  2003   Men         Sleep  7.46
##  3  2003   Men          Work 19.26
##  4  2003 Women       Leisure  7.02
##  5  2003 Women         Sleep  7.65
##  6  2003 Women          Work 17.87
##  7  2004   Men       Leisure  8.66
##  8  2004   Men         Sleep  7.49
##  9  2004   Men          Work 18.92
## 10  2004 Women       Leisure  7.34
## # ... with 74 more rows
```

8.2 geom_area

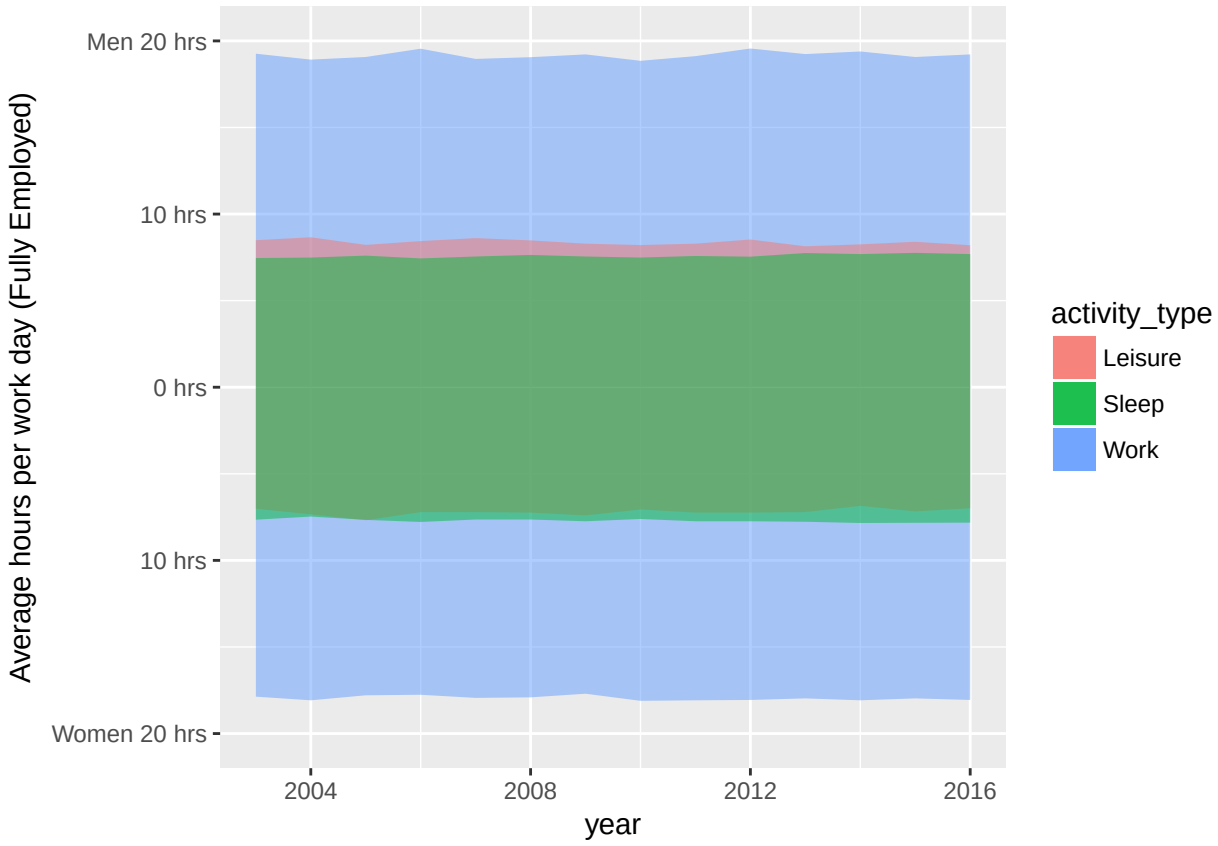
geom_area is useful when components that naturally add to each other:

```
library(ggplot2)
ggplot(data, aes(year, hours, fill= activity_type)) + geom_area() + facet_wrap(~ sex)
```



8.3 geom_ribbon

```
data %>%
  ggplot(aes(x = year, group = sex, fill = activity_type)) +
  geom_ribbon(mapping = aes(ymin = -hours * (sex == "Women"), ymax = hours * (sex == "Men")), data = . %
  geom_ribbon(mapping = aes(ymin = -hours * (sex == "Women"), ymax = hours * (sex == "Men")), data = . %
  geom_ribbon(mapping = aes(ymin = -hours * (sex == "Women"), ymax = hours * (sex == "Men")), data = . %
  scale_y_continuous(
    name = "Average hours per work day (Fully Employed)",
    breaks = c(-20, -10, 0, 10, 20),
    labels = c("Women 20 hrs", "10 hrs", "0 hrs", "10 hrs", "Men 20 hrs"),
    limits = c(-20, 20)
    )
```



8.4 Assignment

Plot leisure computer use over time using separate lines for men and women. The y axis should display the amount of use in minutes. The plot should look like the following image (the aspect ratio can be different).

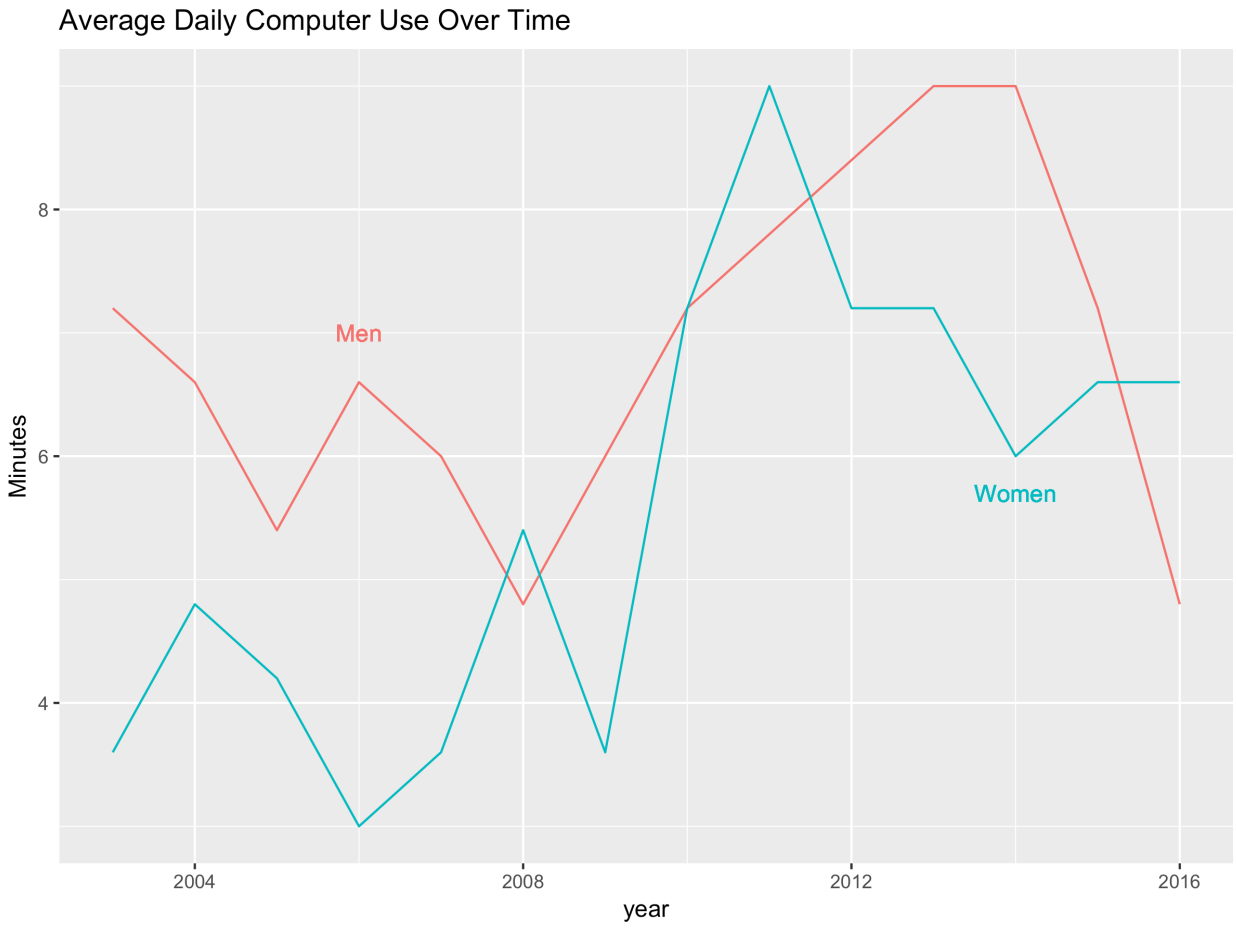

Average Daily Computer Use Over Time
Men
Women
Minutes
8
6
4
2004
2008
2012
2016
year

Figure 8.1:

Chapter 9

Jitter, Rug, and Aesthetics

9.1 Data

The Panel Study of Income Dynamics (PSID) is the longest running longitudinal household survey in the world.

From the Data page, you can use the `Data Center` to create customized datasets. We'll use the Packaged Data option. Click the Main and Supplemental Studies link. Under the `Supplemental Studies > Transition into Adulthood Supplement` section, select the download for `2015`.

To download the supplement you will need to sign in or register for a new account (by clicking the "New User?" link). Once you have logged in you should be able to download the zip file:

- ta2015.zip

9.1.1 Codebook

The `TA2015_codebook.pdf` is the perfect place for us to identify some key variables of interest. The following is an excerpt listing the variables we will use:

```
TA150003  "2015 PSID FAMILY IW (ID) NUMBER"
2015 PSID Family Interview Number

TA150004  "2015 INDIVIDUAL SEQUENCE NUMBER"
2015 PSID Sequence Number
This variable provides a means of identifying an individual's status with regard to the
PSID family unit at the time of the 2015 PSID interview.

Value/Range   Code Value/Range Text
1 - 20        Individuals in the family at the time of the 2015 PSID
              interview
51 - 59       Individuals in institutions at the time of the 2015 PSID
              interview

TA150005 "CURRENT STATE"
Current State (FIPS state codes)

TA150015 "A1_1 HOW SATISFIED W/ LIFE AS A WHOLE"
A1_1. We'd like to start by asking you about life in general. Please think about your
```

```
life-as-a-whole. How satisfied are you with it? Are you completely satisfied, very
satisfied, somewhat satisfied, not very satisfied, or not at all satisfied?

Value/Range   Code Value/Range Text
1             Completely satisfied
2             Very satisfied
3             Somewhat satisfied
4             Not very satisfied
5             Not at all satisfied
8             DK

TA150092 "D28A NUMBER OF CHILDREN"
D28a. How many (biological,) adopted, or step-children do you have?

TA150128 "E1 EMPLOYMENT STATUS 1ST MENTION"
E1. Now we have some questions about employment. We would like to know about what you do -
- are you working now, looking for work, keeping house, a student, or what?--1ST MENTION

Value/Range   Code Value/Range Text
1             Working now, including military
2             Only temporarily laid off; sick or maternity leave
3             Looking for work, unemployed
5             Disabled, permanently or temporarily
6             Keeping house
7             Student

TA150512 "F1 HOW MUCH EARN LAST YEAR"
F1. We try to understand how people all over the country are getting along financially, so
now I have some questions about earnings and income. How much did you earn altogether
from work in 2014, that is, before anything was deducted for taxes or other things,
including any income from bonuses, overtime, tips, commissions, military pay or any other
source?

Value/Range     Code Value/Range Text
0 - 5,000,000   Actual amount
    9,999,998   DK
    9,999,999   NA; refused
```

9.1.1.1 FIPS

In preparation for working with these variables, we can setup arrays to take the place of the codebook. The `tigris` package will give us the FIPS codes for each state:

```
install.packages(tigris)
```

```
library(tidyverse)

state_fips <- tigris::fips_codes %>%
  group_by(state) %>%
  summarize(fips = as.numeric(first(state_code)))
fips2state <- array()
fips2state[state_fips$fips] <- state_fips$state
fips2state
```

```
##  [1] "AL" "AK" NA   "AZ" "AR" "CA" NA   "CO" "CT" "DE" "DC" "FL" "GA" NA
## [15] "HI" "ID" "IL" "IN" "IA" "KS" "KY" "LA" "ME" "MD" "MA" "MI" "MN" "MS"
## [29] "MO" "MT" "NE" "NV" "NH" "NJ" "NM" "NY" "NC" "ND" "OH" "OK" "OR" "PA"
## [43] NA   "RI" "SC" "SD" "TN" "TX" "UT" "VT" "VA" NA   "WA" "WV" "WI" "WY"
## [57] NA   NA   NA   "AS" NA   NA   NA   NA   NA   "GU" NA   NA   "MP" NA
## [71] NA   "PR" NA   "UM" NA   NA   NA   "VI"
```

9.1.1.2 Satisfaction

```
satisfaction <- array()
satisfaction_levels <- c("Completely satisfied", "Very satisfied", "Somewhat satisfied", "Not very satis
satisfaction[c(1, 2, 3, 4, 5, 8)] <- satisfaction_levels
satisfaction
```

```
## [1] "Completely satisfied" "Very satisfied"       "Somewhat satisfied"
## [4] "Not very satisfied"   "Not at all satisfied" NA
## [7] NA                     "DK"
```

9.1.1.3 Employment Status

We can also specify the array elements one by one:

```
employment <- array()
employment[1] <- "Working now, including military"
employment[2] <- "Only temporarily laid off; sick or maternity leave"
employment[3] <- "Looking for work, unemployed"
employment[5] <- "Disabled, permanently or temporarily"
employment[6] <- "Keeping house"
employment[7] <- "Student"
employment
```

```
## [1] "Working now, including military"
## [2] "Only temporarily laid off; sick or maternity leave"
## [3] "Looking for work, unemployed"
## [4] NA
## [5] "Disabled, permanently or temporarily"
## [6] "Keeping house"
## [7] "Student"
```

9.1.2 Preprocessing the SPSS File

If you don't have them already, open the zip file and move the `TA2015.txt` and `TA2015.sps` files into the `data` folder. For our import to work, We need to find a line that needs to be removed from the top of the sps file. The line we want to remove should look like the following line:

```
FILE HANDLE PSID / NAME = '[PATH]\TA2015.TXT' LRECL = 2173 .
```

To find this line we can output the first 20 lines of the `TA2015.sps` file:

```
readLines("data/TA2015.sps", n = 10)
```

```
##  [1] ""
##  [2] "************************************************************************"
##  [3] "   Label           : Transition to Adulthood Study 2015"
```

```
##  [4] "   Rows            : 1641"
##  [5] "   Columns         : 1304"
##  [6] "   ASCII File Date : July 5, 2017"
##  [7] "*****************************************************************************."
##  [8] ""
##  [9] "FILE HANDLE PSID / NAME = '[PATH]\\TA2015.TXT' LRECL = 2173 ."
## [10] "DATA LIST FILE = PSID FIXED /"
```

Now we know the line to remove is line number 9, we can write a new file to be used in the processing step below.

```
input <- file("data/TA2015.sps")
output <- file("data/TA2015_clean.sps")

open(input, type = "r")
open(output, open = "w")

writeLines(readLines(input, n = 8), output)
invisible(readLines(input, n = 1))
writeLines(readLines(input), output)

close(input)
close(output)

readLines("data/TA2015_clean.sps", n = 10)
```

```
##  [1] ""
##  [2] "*****************************************************************************"
##  [3] "   Label           : Transition to Adulthood Study 2015"
##  [4] "   Rows            : 1641"
##  [5] "   Columns         : 1304"
##  [6] "   ASCII File Date : July 5, 2017"
##  [7] "*****************************************************************************."
##  [8] ""
##  [9] "DATA LIST FILE = PSID FIXED /"
## [10] "      TA150001        1 - 1         TA150002        2 - 6         TA150003        7 - 11    "
```

9.1.3 Importing with the SPSS file using memisc

The `memisc` package has useful tools for importing SPSS and Stata files that augment what already exists in `base`. Unfortunately, one of its dependencies, `MASS`, will mask the select method from `dplyr`. To avoid this, instead of loading `memisc` with `library(memisc)`, we can prefix all `memisc` functions with `memisc::`.

```
install.packages("memisc")
```

```
ta_importer <- memisc::spss.fixed.file("data/TA2015.txt", columns.file = "data/TA2015_clean.sps", varlab
ta_full <- memisc::as.data.set(ta_importer)
ta_full
```

```
##
## Data set with 1641 observations and 1304 variables
##
##    TA150001 TA150002 TA150003 TA150004 TA150005 TA150006 TA150007 ...
##  1        1        1     4893        1       37       55        9 ...
##  2        1        2     2967        1       48       57        9 ...
##  3        1        3     6095        5       37       96        9 ...
```

```
##  4        1        4     3738        3        8       98        9 ...
##  5        1        5     6741        3       16      104        9 ...
##  6        1        6     4839        1       28       68        9 ...
##  7        1        7     3828        1       48       67        9 ...
##  8        1        8     5640       51       21       88        9 ...
##  9        1        9     5210       52        6       80        9 ...
## 10        1       10      339        3       18       93        9 ...
## 11        1       11     3192        2       26       66        9 ...
## 12        1       12      561        2       26       91        9 ...
## 13        1       13     2500       51       13      100        9 ...
## 14        1       14     5283        2       39       68        9 ...
## 15        1       15     6679        1       24       62        9 ...
## 16        1       16     3286        2        8       67        9 ...
## 17        1       17     3266        2        6      155        9 ...
## 18        1       18     3720        2       48       61        9 ...
## 19        1       19     3714        2       48       82        9 ...
## 20        1       20     6244        1       48       89        9 ...
## 21        1       21     4199        1       13       80        9 ...
## 22        1       22     3962        2       51       86        9 ...
## 23        1       23     2878        1       41       85        9 ...
## 24        1       24     3835        1       37       85        9 ...
## 25        1       25      487       51       13       91        9 ...
## .. ........ ........ ........ ........ ........ ........ ........ ...
## (25 of 1641 observations shown)
```

9.1.4 Transform Data

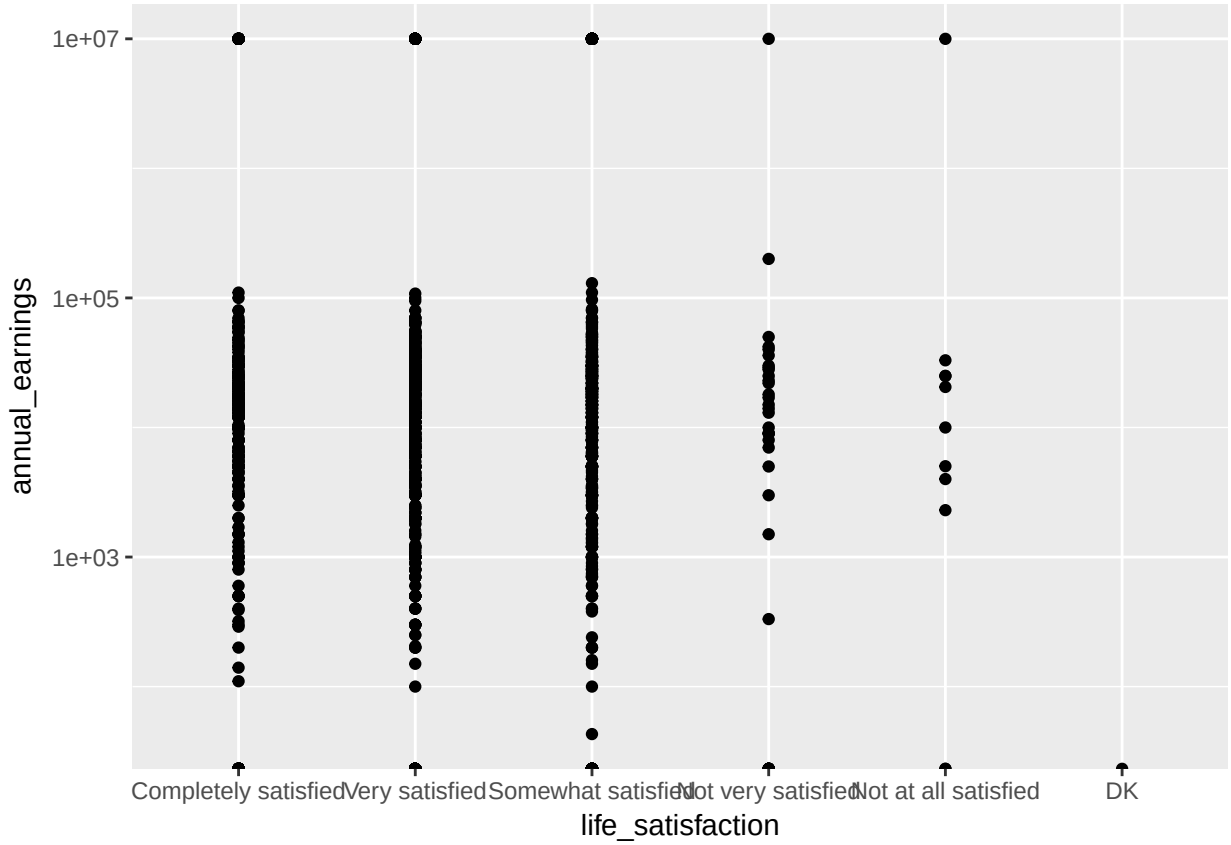
Take the arrays we created above to process the file down to the variables we selected.

```
ta <- ta_full %>%
  as.data.frame() %>%
  as.tbl() %>%
  filter(TA150005 > 0) %>% # get rid of the 1 non-US response
  transmute(
    family_id         = TA150003,
    in_institution    = TA150004 > 50,
    state             = fips2state[TA150005],
    life_satisfaction = factor(satisfaction[TA150015], levels = satisfaction_levels, ordered = TRUE),
    children          = TA150092,
    employment_status = employment[TA150128],
    annual_earnings   = TA150512 %>% na_if(9999999) %>% na_if(9999999)
  )
ta
```

```
## # A tibble: 1,640 x 7
##    family_id in_institution state    life_satisfaction children
##        <dbl>          <lgl> <chr>                <ord>    <dbl>
##  1      4893          FALSE    NC   Somewhat satisfied        0
##  2      2967          FALSE    TX   Somewhat satisfied        0
##  3      6095          FALSE    NC   Somewhat satisfied        0
##  4      3738          FALSE    CO   Somewhat satisfied        0
##  5      6741          FALSE    ID       Very satisfied        0
##  6      4839          FALSE    MS Completely satisfied        1
##  7      3828          FALSE    TX   Somewhat satisfied        0
```

```
##  8      5640           TRUE    KY Completely satisfied        0
##  9      5210           TRUE    CA       Very satisfied        0
## 10       339          FALSE    IN   Somewhat satisfied        0
## # ... with 1,630 more rows, and 2 more variables: employment_status <chr>,
## #   annual_earnings <dbl>
```

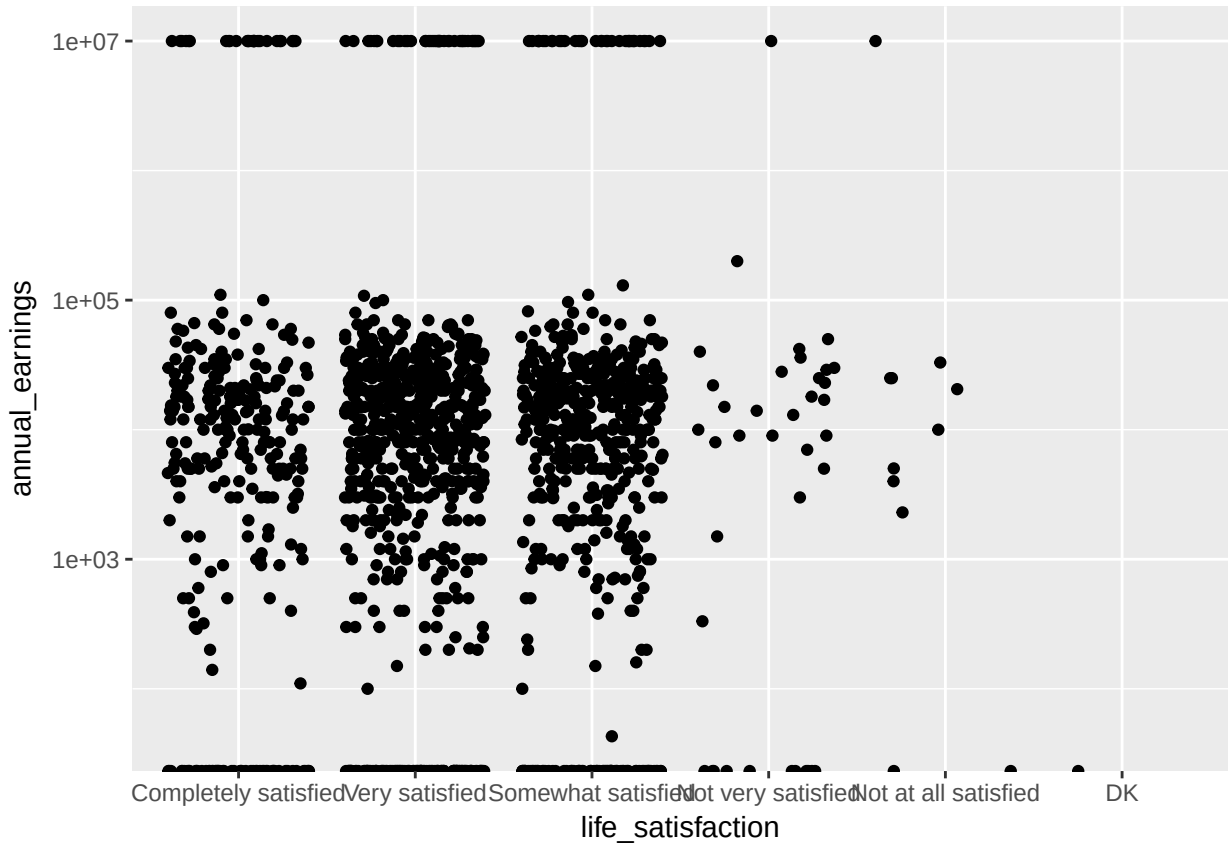
```
library(ggplot2)
base_plot <- ggplot(ta, aes(life_satisfaction, annual_earnings)) + scale_y_log10()
base_plot + geom_point()
```



9.2 Jitter

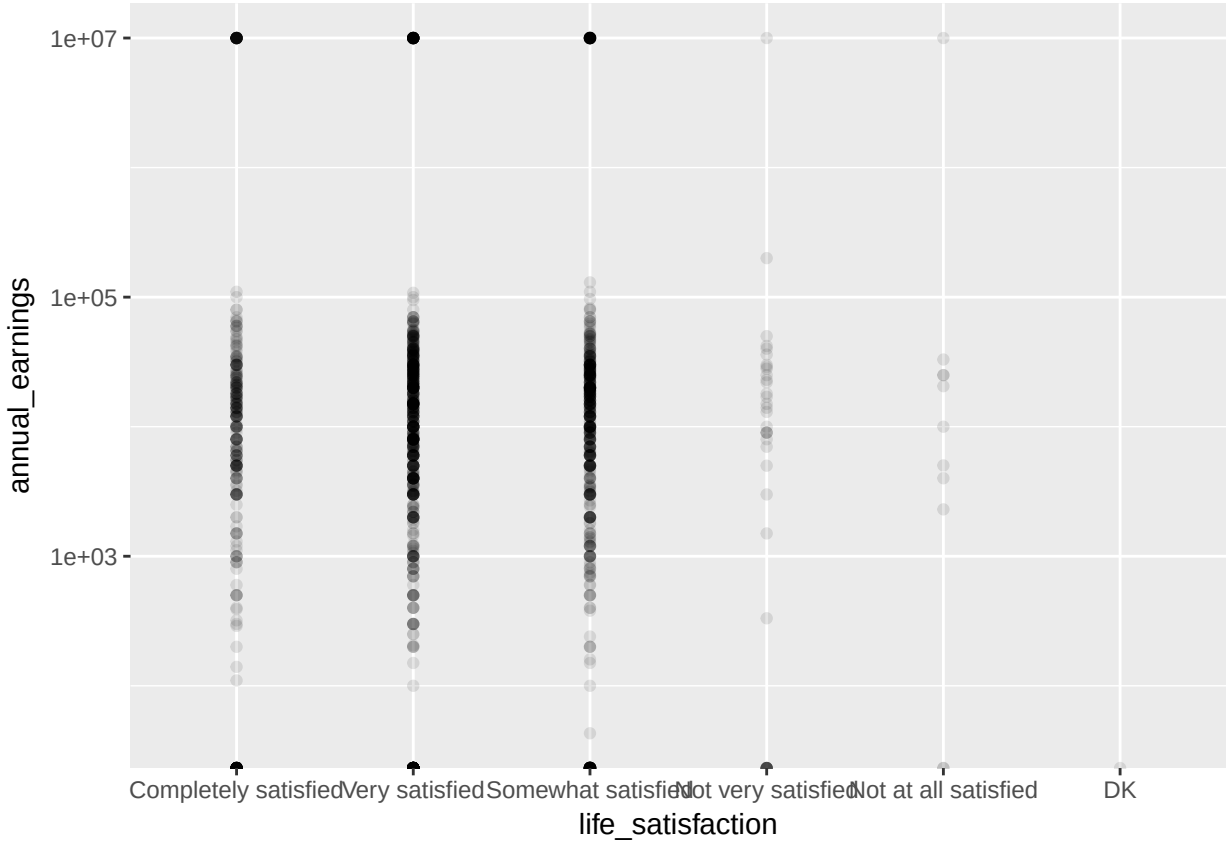
When many points overlap, using `geom_jitter` adjusts the position of each point to minimize overlap.

```
base_plot + geom_jitter()
```



See how this compares to using `alpha` (i.e., opacity) to see how many points are in a given position:

```
base_plot + geom_point(alpha = 0.1)
```



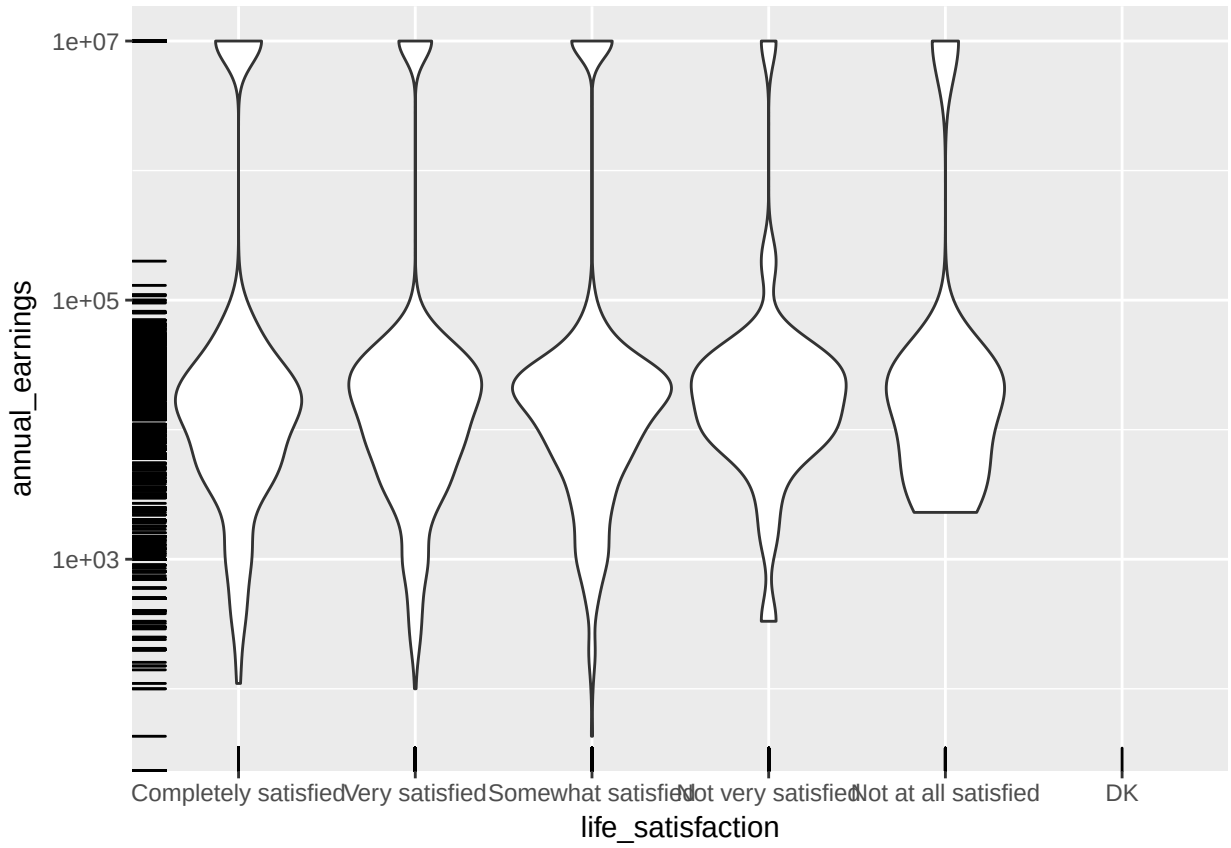
9.3 Rug

Often when there are many points, we want to plot a summary that presents the general shape of the data.

```
base_plot + geom_violin()
```

The `geom_rug` gives us a rug plot that we can use to highlight where actual observations occured when we create these summary plots.

```
base_plot + geom_violin() + geom_rug()
```



Both `alpha` and `jitter` can be applied to the rug as well.

```
base_plot + geom_violin() + geom_rug(alpha = 0.1, position = "jitter")
```

Now we can see that rug did in fact create a line on the x-axis for each observation. We can use the `sides` property to only display the rug on the left and right (`"lr"`)

```
base_plot + geom_violin() + geom_rug(alpha = 0.1, position = "jitter", sides = "lr")
```

9.4 Aesthetics

Aesthetics are the visual properties that define each graph. In ggplot, the `aes()` function is used to create a mapping from your data to these visual properties. We most often use `aes()` to map our variables to the `x` and `y` dimension. Above, we used the fact that the first two arguments to `aes()` are `x` and `y`. That is,

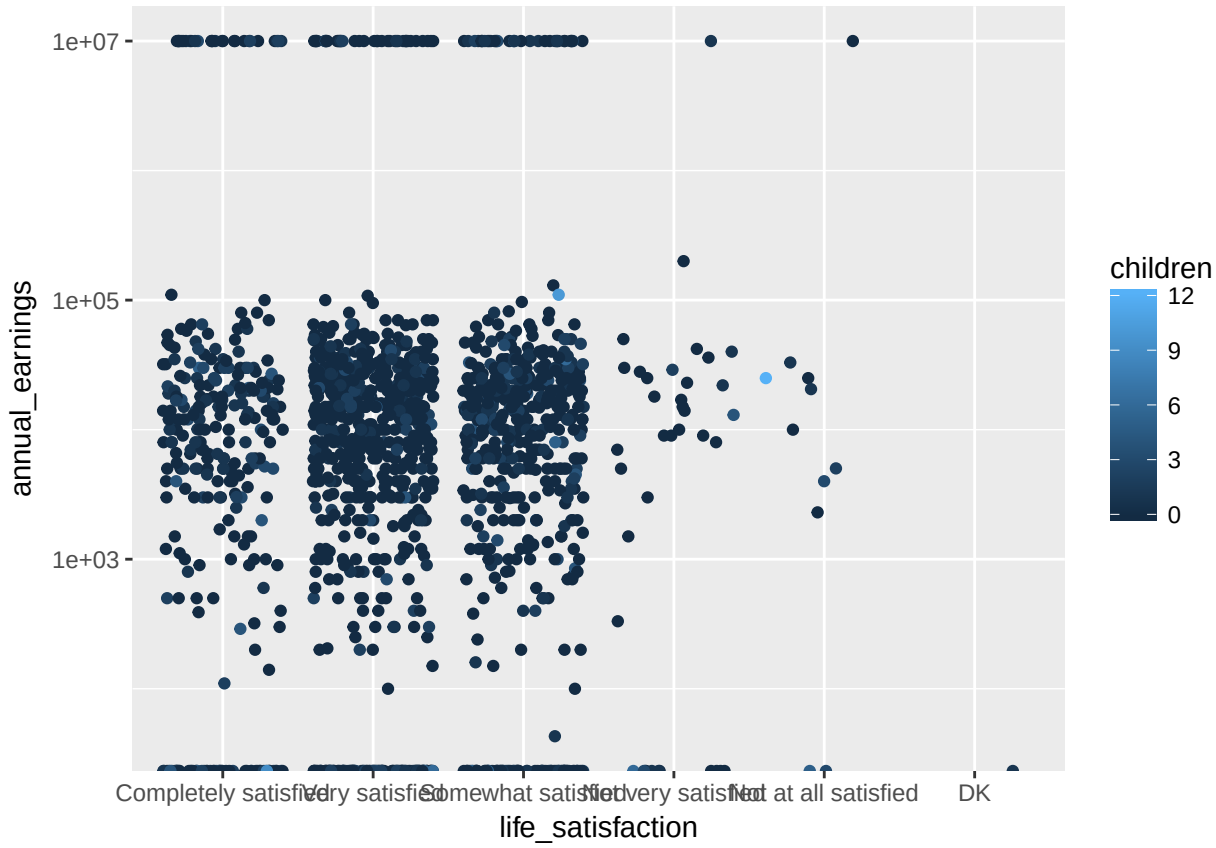
```
aes(x = life_satisfaction, y = annual_earnings)
```

gives the same result as

```
aes(life_satisfaction, annual_earnings)
```

Anytime we want more than two dimensions of our data displayed, it's useful to map the extra variables to other features of our graph (e.g., size, color, alpha, etc.). Let's add number of children to our graph assigning children to color. (In ggplot both American and British spellings are supported, so you can use `color` or `colour`.)

```
ggplot(ta, aes(life_satisfaction, annual_earnings, color = children)) +
  scale_y_log10() +
  geom_jitter()
```

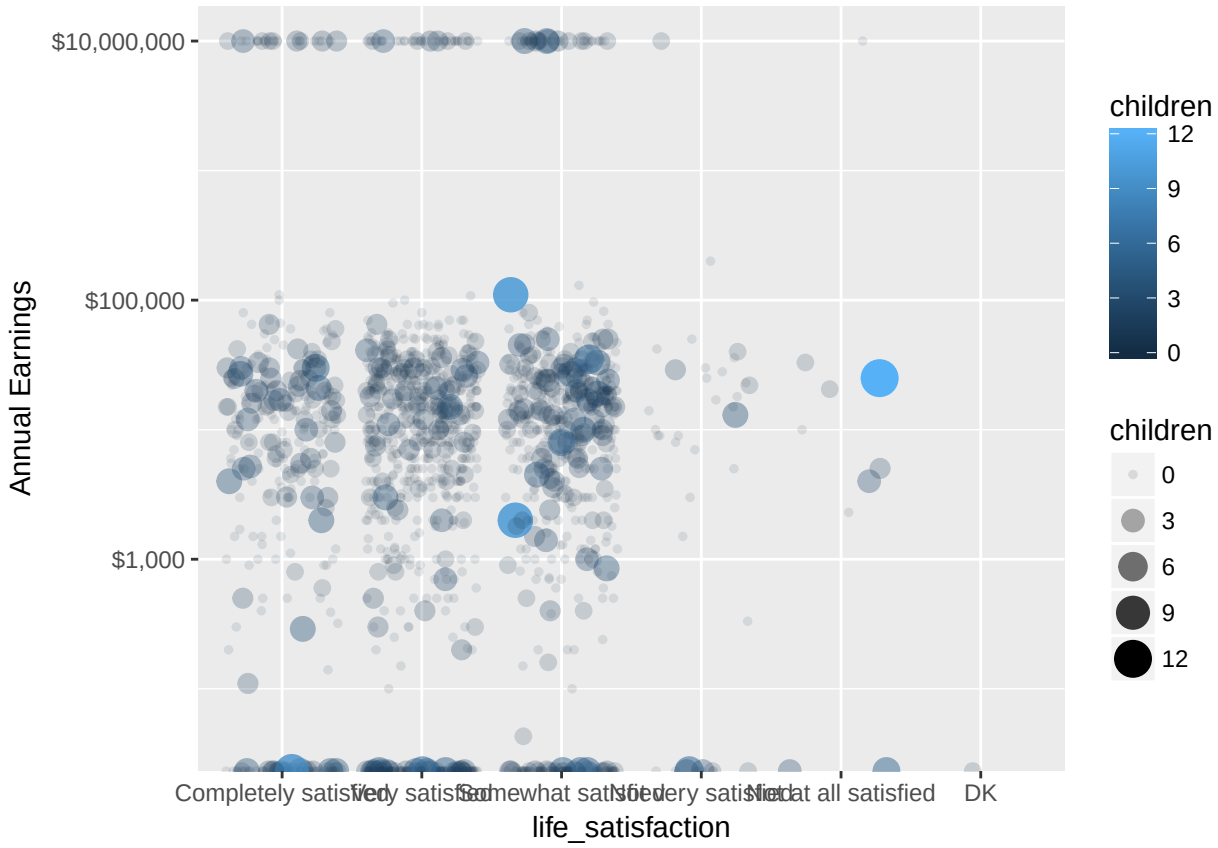


If the `children` variable was a factor, more distinct colors would have been chosen. We can see this by coloring by `employment_status` instead.

```
ggplot(ta, aes(life_satisfaction, annual_earnings, color = employment_status)) +
  scale_y_log10() +
  geom_jitter()
```

You can emphasize a variable by encoding in more than one visual aesthetic. Let's map `children` to `color`, `size`, and `alpha`.

```
ggplot(ta, aes(life_satisfaction, annual_earnings, color = children, size = children, alpha = children)
  scale_y_log10("Annual Earnings", labels = scales::dollar) +
  geom_jitter()
```



9.5 Assignment

Open the codebook (`data/TA2015_codebook.pdf`) and search for two new variables to visualize. Create a plot that makes use of both jitter and rug.

Chapter 10

Themes, Labels, and Colors

This lecture uses the following packages:

```
tidyverse
lubridate
ggthemes
grid
gridExtra
```

10.1 Data

10.1.1 Zillow Real Estate Data

Zillow is an online marketplace for real estate. It facilitates connection between buyers and sellers, and in the process collects a large amount of useful economic statistics.

From Zillow's research data page, we will download the Age of Inventory (Days) CSV. Their list of data definitions at the bottom of the page includes the following entry:

Age of Inventory: Each Wednesday, age of inventory is calculated as the median number of days all active listings as of that Wednesday have been current. These medians are then aggregated into the number reported by taking the median across weekly values.

```
library(readr)
raw_inventory <- read_csv("data/AgeOfInventory_Metro_Public.csv")
head(raw_inventory)[,1:6]
```

```
## # A tibble: 6 x 6
##              RegionName RegionType StateFullName DataTypeDescription
##                   <chr>      <chr>         <chr>               <chr>
## 1         United States    Country          <NA>           All Homes
## 2          New York, NY        Msa      New York           All Homes
## 3           Chicago, IL        Msa      Illinois           All Homes
## 4 Dallas-Fort Worth, TX        Msa         Texas           All Homes
## 5      Philadelphia, PA        Msa  Pennsylvania           All Homes
## 6           Houston, TX        Msa         Texas           All Homes
## # ... with 2 more variables: `2012-01` <int>, `2012-02` <int>
```

10.1.2 Reshaping data with `tidyr`

Since this data set uses a separate column for each time period, the data is not yet tidy. Let's fix that. We'll use the `gather()` function from the `tidyr` package (http://tidyr.tidyverse.org/reference/gather.html) to identify the name for the new column that stores the old column names (`month`), the name for the new column that stores the values in the columns being reshaped (`age`), and a column selector to identify which columns to reshape (`matches()` takes a regular expression, see the `regex` documentation). Lastly, we need to `mutate()` the `month` using `ymd()` from the `lubridate` package. Into `ymd()`, we place the original date character (in "YYYY-MM" format) and add a piece for the day (using the "-DD" format), so that each of our month time markers are interpreted as the first day of the corresponding month.

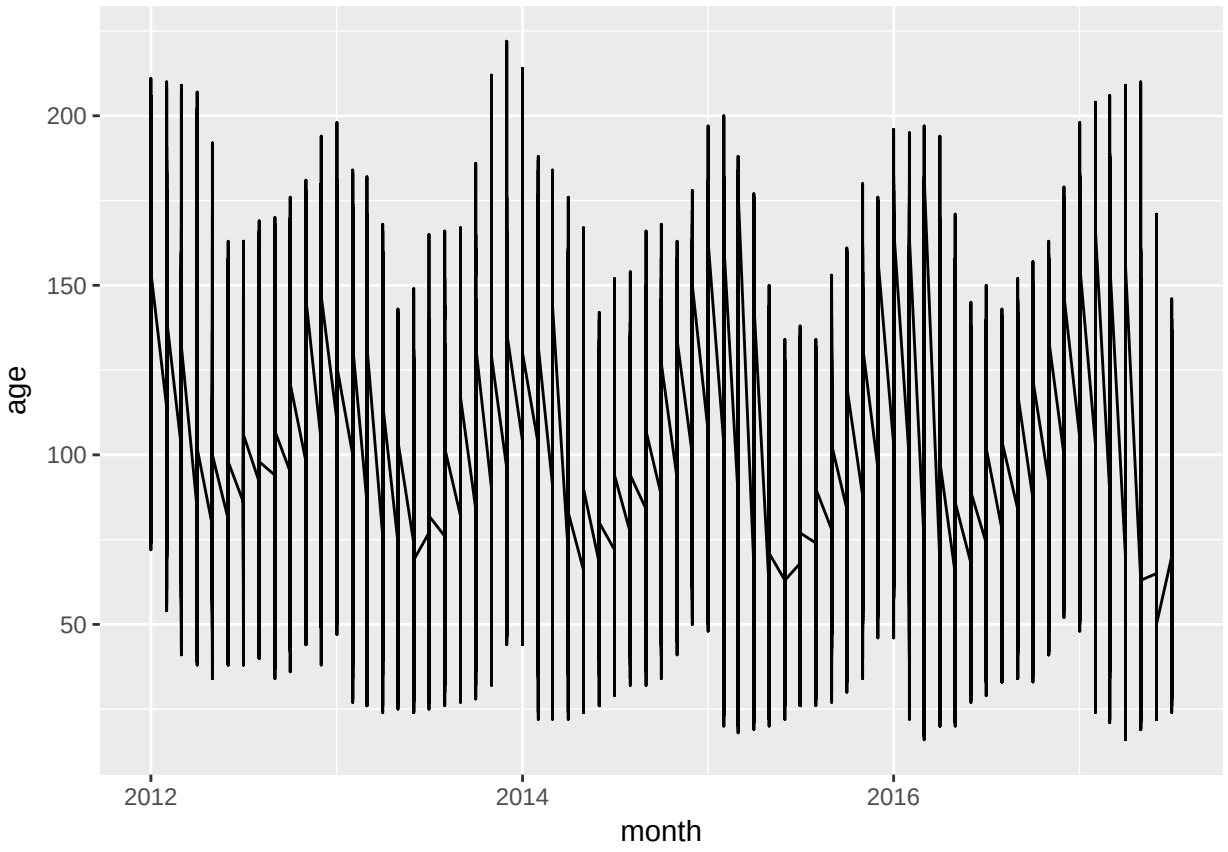
```
library(lubridate)
if("package:dplyr" %in% search()) detach("package:dplyr", unload=TRUE)
library(dplyr)
library(tidyr)
inventory <- raw_inventory %>% select(RegionName, matches("[[:digit:]]")) %>%
  gather(month, age, matches("[[:digit:]]")) %>%
  mutate(month = ymd(paste0(month, "-01")))
inventory
```

```
## # A tibble: 22,713 x 3
##                    RegionName      month   age
##                         <chr>     <date> <int>
##  1             United States 2012-01-01   120
##  2              New York, NY 2012-01-01   136
##  3               Chicago, IL 2012-01-01   141
##  4     Dallas-Fort Worth, TX 2012-01-01   109
##  5          Philadelphia, PA 2012-01-01   132
##  6               Houston, TX 2012-01-01   112
##  7            Washington, DC 2012-01-01   105
##  8 Miami-Fort Lauderdale, FL 2012-01-01    89
##  9               Atlanta, GA 2012-01-01    98
## 10                Boston, MA 2012-01-01   116
## # ... with 22,703 more rows
```

10.2 Starting Plot

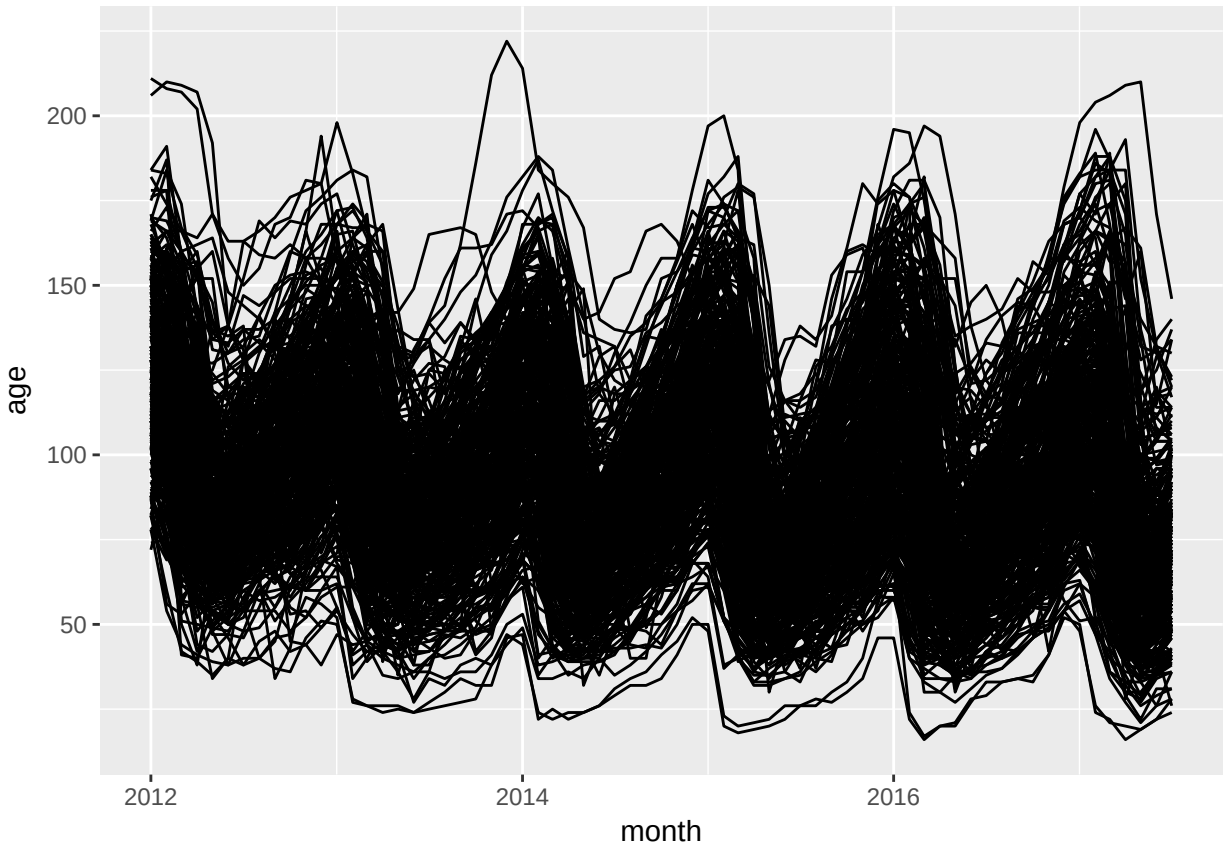
Let's start with a simple line plot of all these series. Let's show what happens if we leave out the group and color aesthetic.

```
library(ggplot2)
inventory %>% ggplot(aes(month, age)) + geom_line()
```



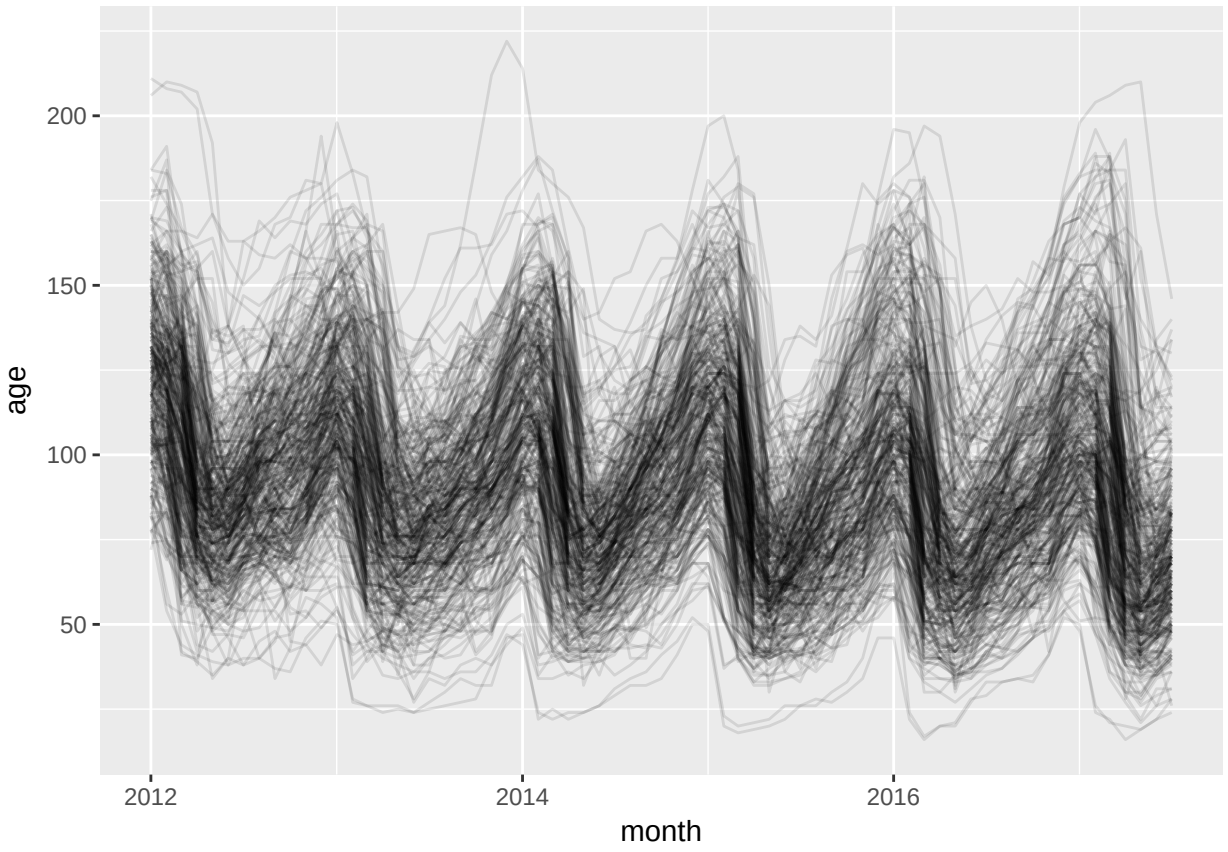
Now, let's add in the `group` `aes`.

```
inventory %>% ggplot(aes(month, age, group = RegionName)) + geom_line()
```



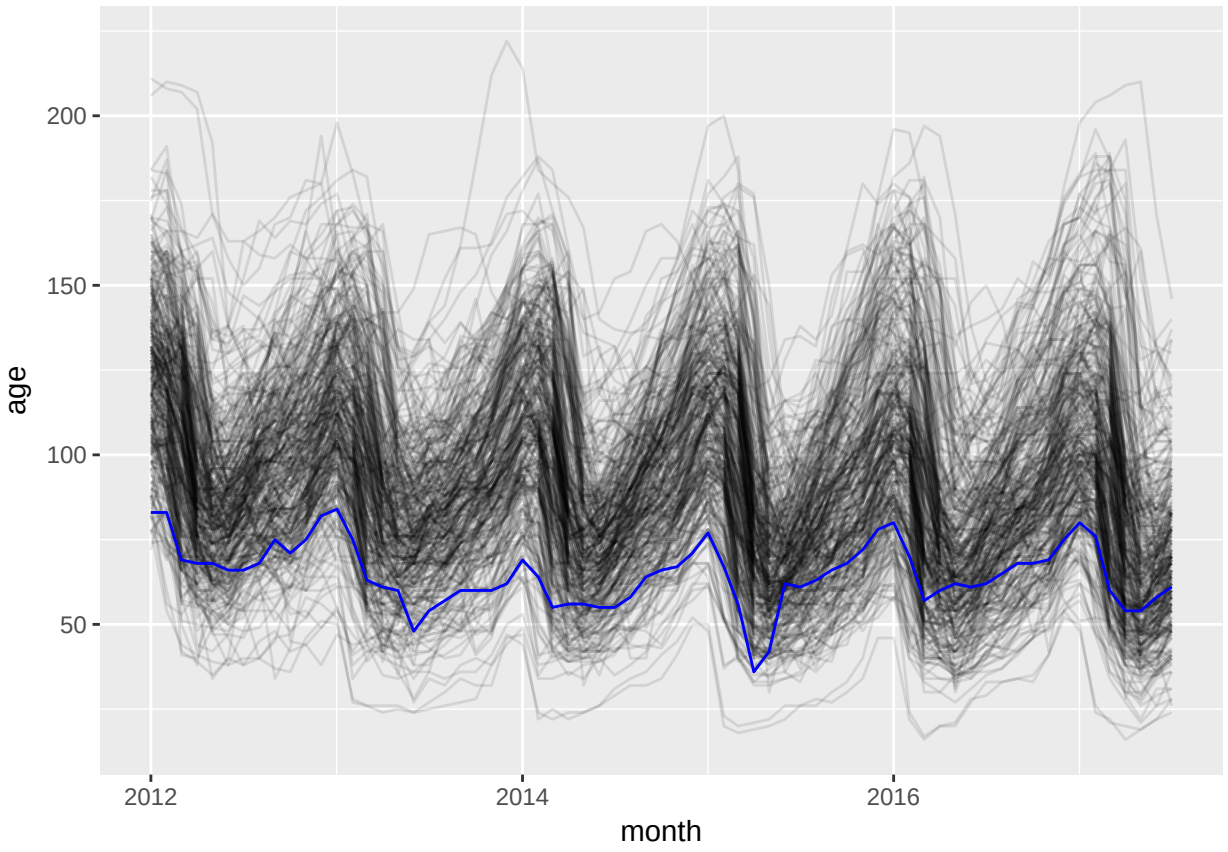
It's hard to see what is happening in this tangled mess. Setting `alpha` to 0.1 will make this easier to untangle.

```
basic_plot <- inventory %>% ggplot(aes(month, age, group = RegionName)) + geom_line(alpha = 0.1)
basic_plot
```



Let's emphasize the line for Honolulu.

```
basic_plot +
  geom_line(data = inventory %>% filter(grepl("Honolulu", RegionName)), aes(month, age), color = "blue"
```

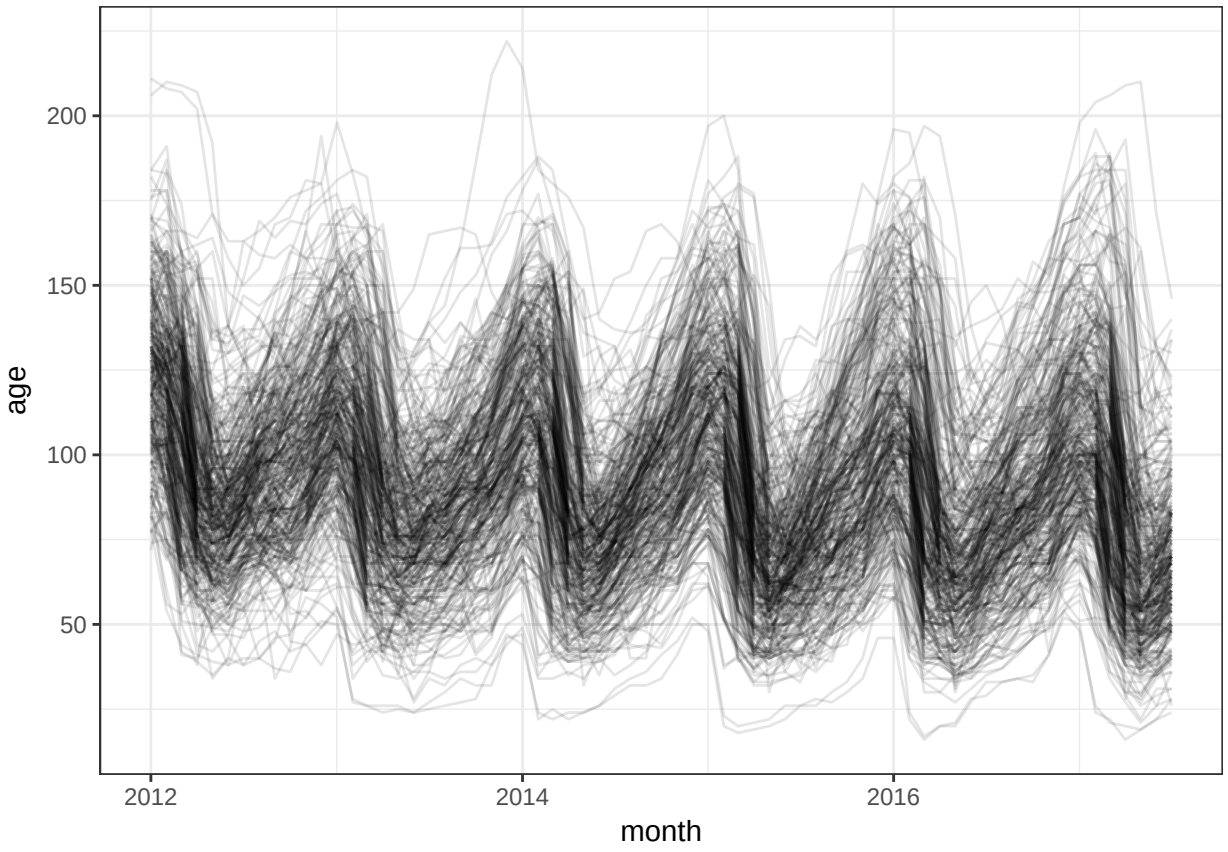


10.3 Themes

10.3.1 Complete Themes

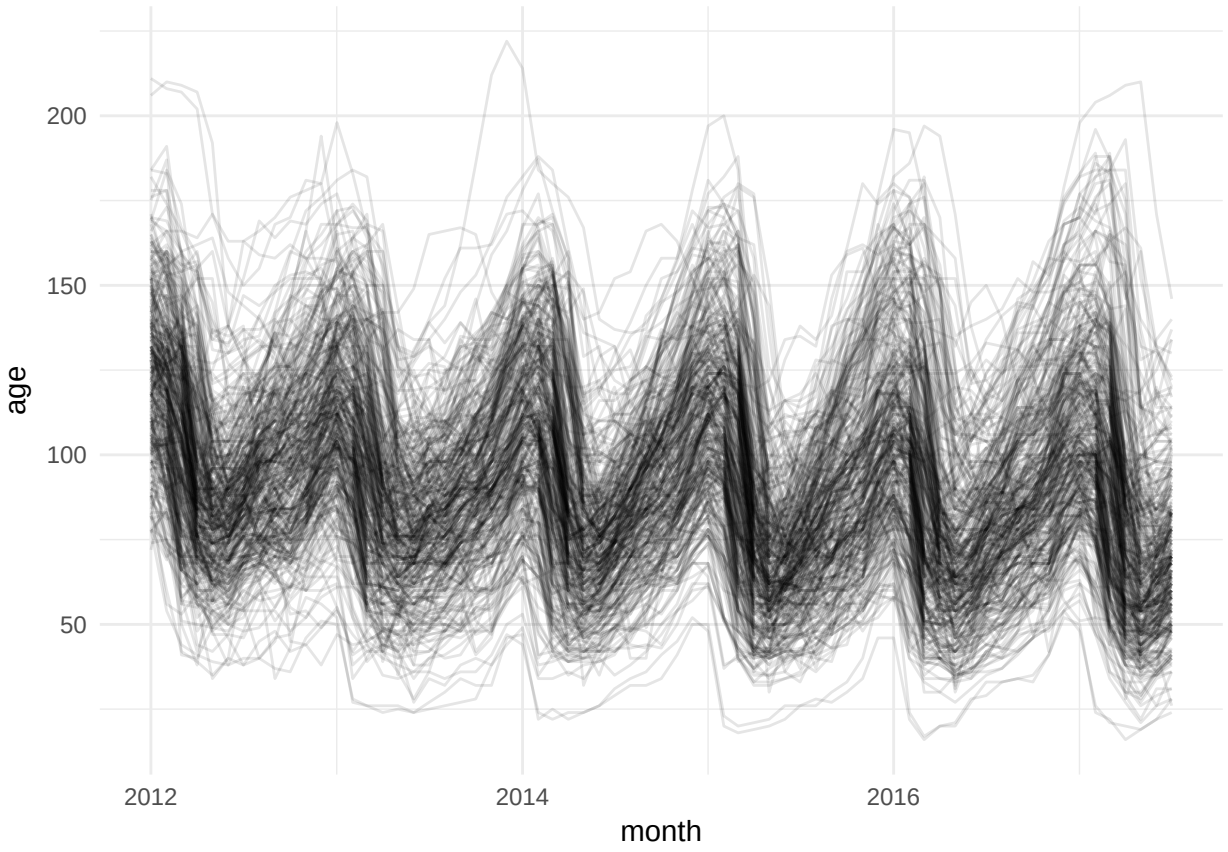
There are a variety of pre-made themes that can make our figures look cleaner (http://ggplot2.tidyverse.org/reference/ggtheme.html). `theme_bw()` is good if you don't want to print all the grey from the default, but you still want the same basic structure.

```
basic_plot + theme_bw()
```



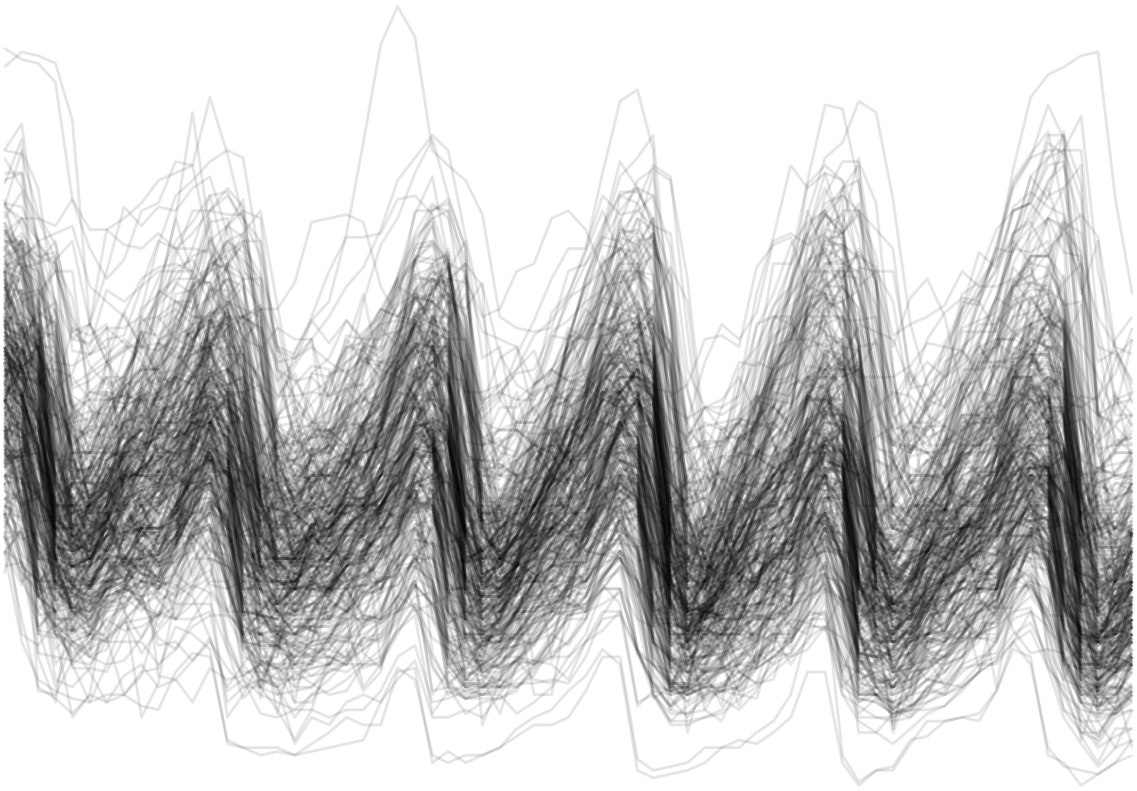
`theme_minimal()` removes some of the visual clutter, removing the plot border and the axis ticks.

```
basic_plot + theme_minimal()
```



`theme_void()` goes the whole way and removes everything, but the data. It even removes the axis labels.

```
basic_plot + theme_void()
```

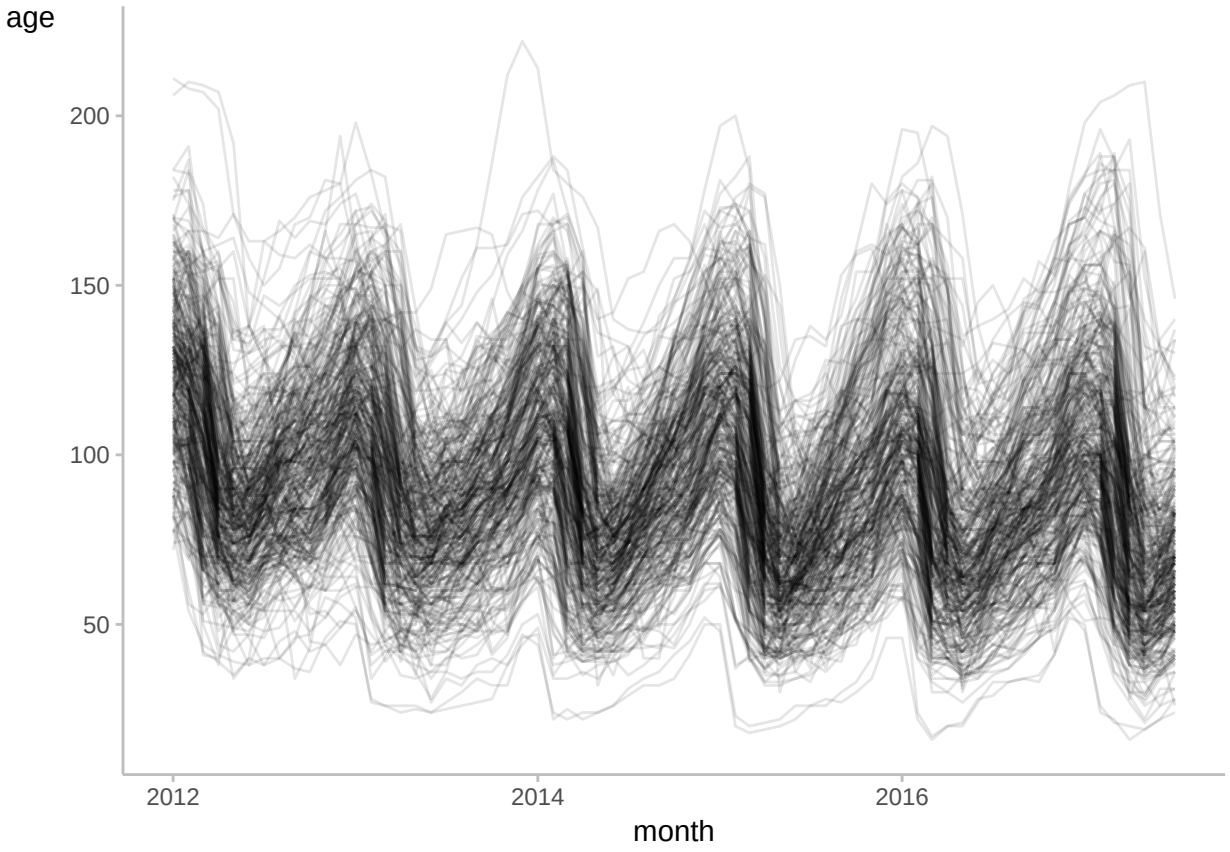


10.3.2 Modifying a theme

To modify a theme, we just add a call to the `theme()` function and assign new values to the parts of the plot we want to change (see the `theme()` reference for more examples: http://ggplot2.tidyverse.org/reference/theme.html).

Let's start with the `theme_bw()` and make the chart more minimal.

```
basic_plot + theme_bw() + theme(
  panel.border = element_blank(),
  panel.grid = element_blank(),
  axis.line = element_line(color = "grey"),
  axis.ticks = element_line(color = "grey"),
  axis.title.y = element_text(angle = 0)
)
```



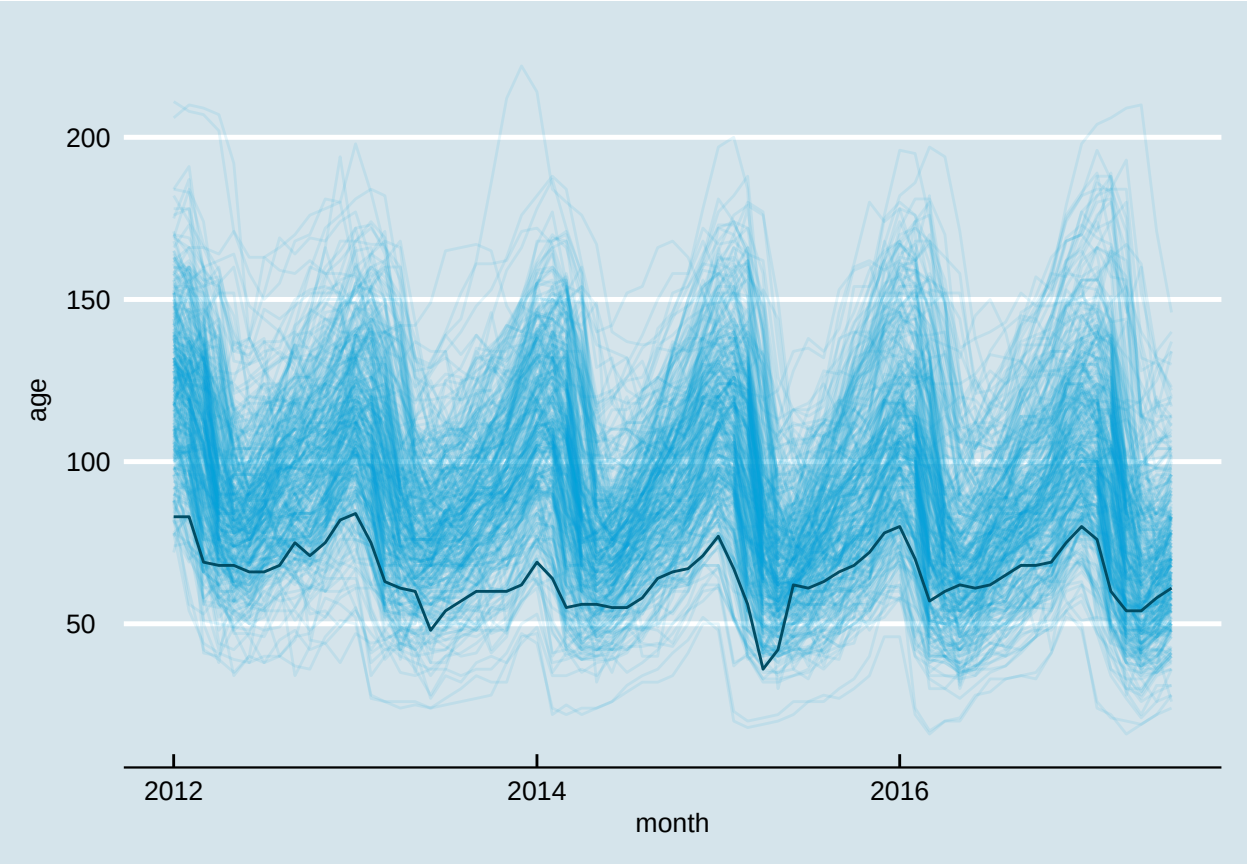
10.3.3 ggthemes

The ggthemes package adds a large set of fun themes. See the vignette at https://cran.r-project.org/web/packages/ggthemes/vignettes/ggthemes.html or enter the following command locally after installing the package

```
install.packages("ggthemes")
vignette("ggthemes", package = "ggthemes")
```

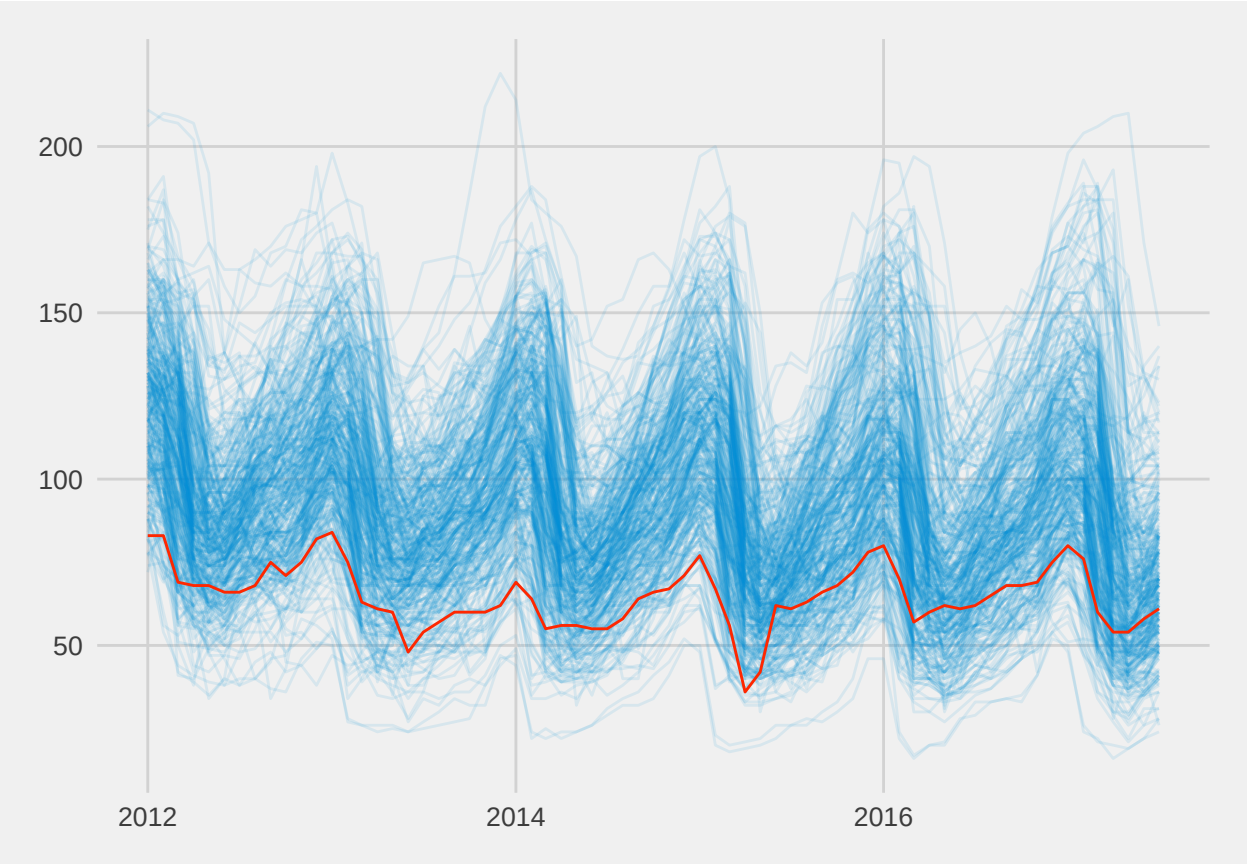
To make our chart look like it came out of the economist, let's use `theme_economist()`. To make the line colors work, we'll use the `economist_pal()` color pal

```
library(ggthemes)
theme_colors <- economist_pal()(2)
inventory %>% ggplot(aes(month, age, group = RegionName)) +
  geom_line(color = theme_colors[1], alpha = 0.1) +
  geom_line(data = inventory %>% filter(grepl("Honolulu", RegionName)),
            aes(month, age),
            color = theme_colors[2]) +
  theme_economist()
```



Now let's try out the theme named for http://fivethirtyeight.com/.

```
theme_colors <- fivethirtyeight_pal()(2)
five38 <- inventory %>% ggplot(aes(month, age, group = RegionName)) +
  geom_line(color = theme_colors[1], alpha = 0.1) +
  geom_line(data = inventory %>% filter(grepl("Honolulu", RegionName)),
            aes(month, age),
            color = theme_colors[2]) +
  theme_fivethirtyeight()
five38
```



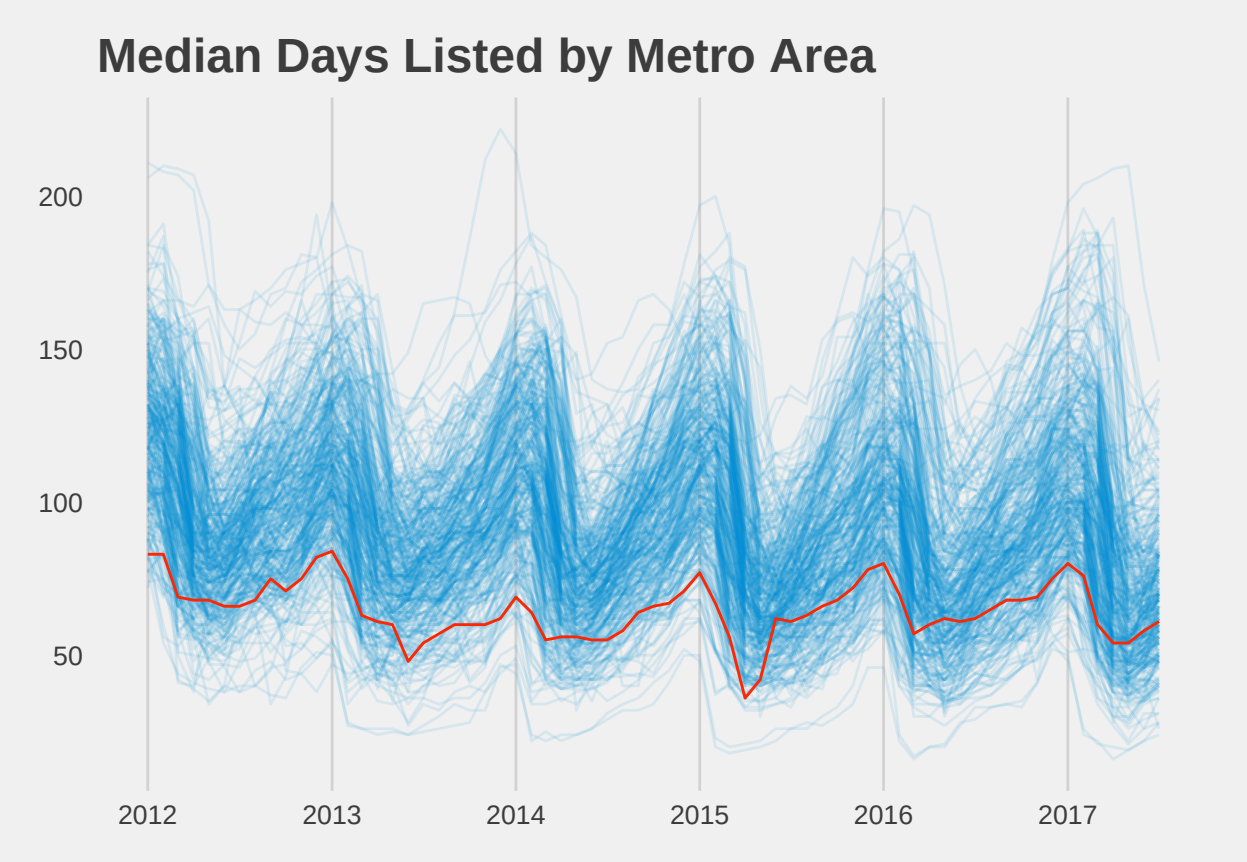
10.4 Labels

Now that we have a nice looking basic chart, we need to make sure our labels are in the right places and give enough information.

10.4.1 Title

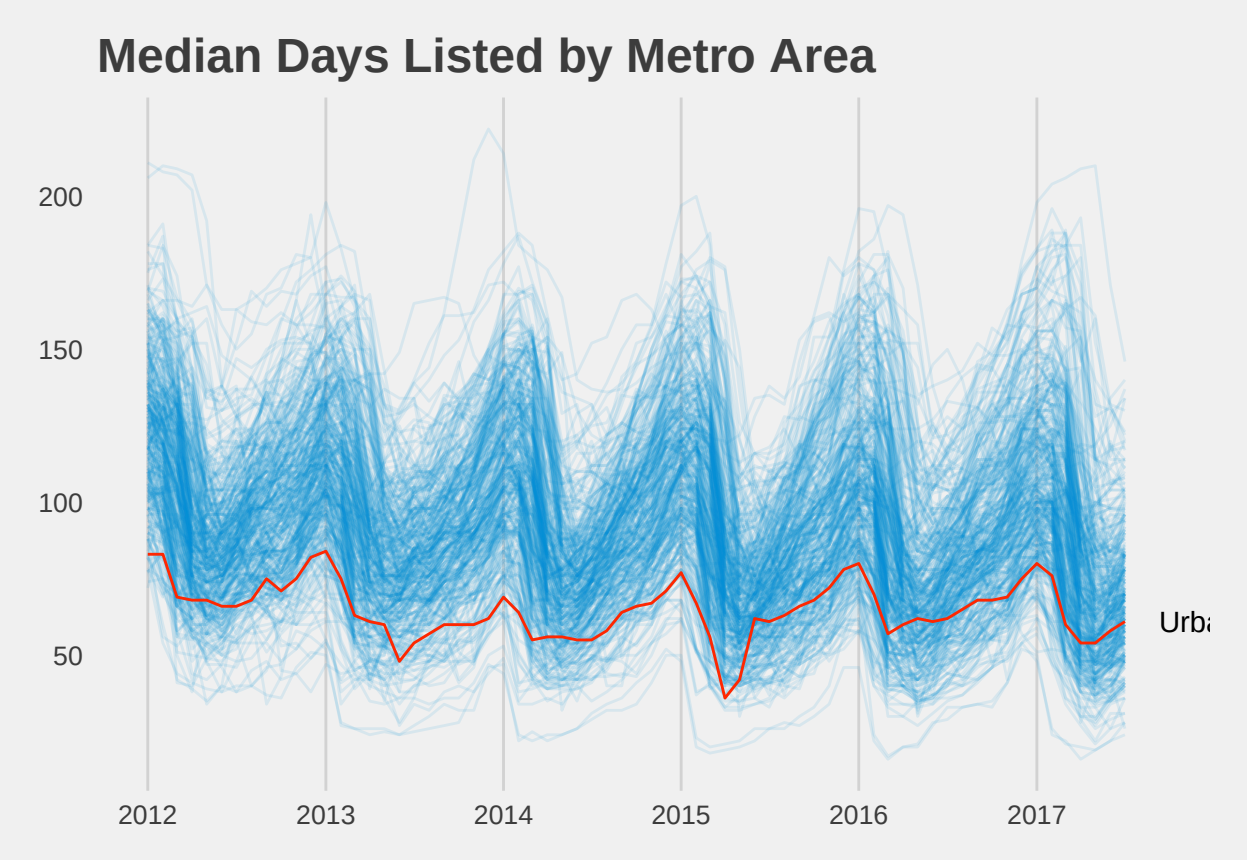
Let's start by adding a title. For a time series like this, using the name of the variable on the x-axis is a good start. We can also change the x-axis to break at each year, which makes the seasonality of this series even easier to pick out. With these added vertical lines, our chart will be more readable if we remove the horizontal gridlines (`panel.grid.major.y`).

```
five38_with_title <- five38 +
  ggtitle("Median Days Listed by Metro Area") +
  scale_x_date(date_breaks = "1 year", date_labels = "%Y") +
  theme(panel.grid.major.y = element_blank())
five38_with_title
```



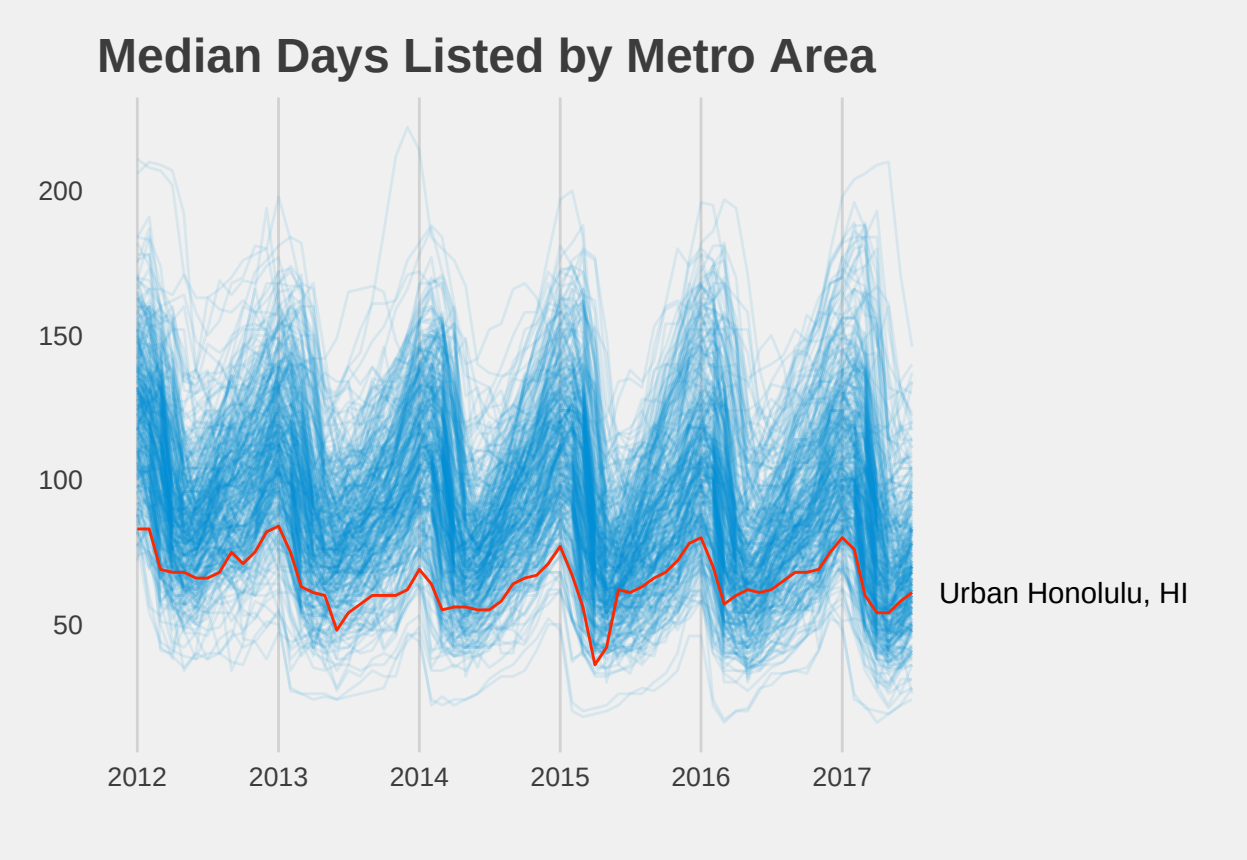
Since we used our title wisely we don't need to add a y-axis title. The x-axis is time and this is fairly obvious, so we can also leave off the x-axis title. What we should do is label the highlighted series.

```
last_hnl <- inventory %>%
  filter(grepl("Honolulu", RegionName)) %>%
  top_n(1, month)
gg <- five38_with_title +
  geom_text(data = last_hnl, label = last_hnl$RegionName,
            hjust = "left", nudge_x = 70)
gg
```



To adjust the margins, we have to drill deeper than ggplot. I found the following approach through searching for `ggplot clipping` (https://rud.is/b/2015/08/27/coloring-and-drawing-outside-the-lines-in-ggplot/).

```
library(gridExtra)
library(grid)
gb <- ggplot_build(gg + theme(plot.margin = unit(c(1, 7, 2, 1), "lines")))
gt <- ggplot_gtable(gb)

gt$layout$clip[gt$layout$name=="panel"] <- "off"

grid.draw(gt)
```



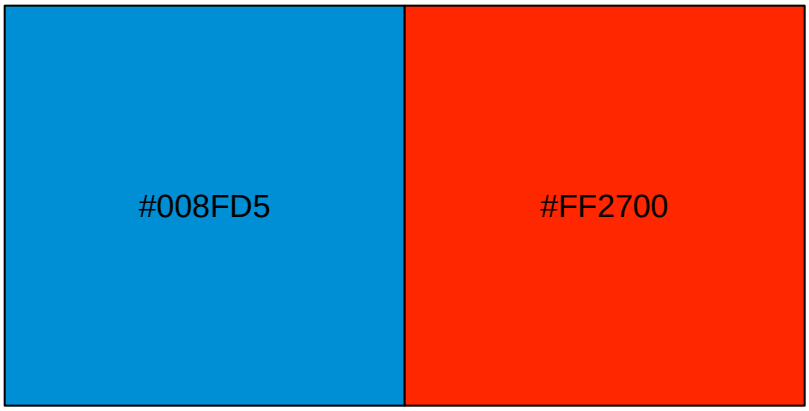
10.5 Colors

Color is an important tool in creating engaging and informative visualizations. Color is often used to encode a dimension not already displayed in a chart (e.g., adding a third dimension to a scatter plot). Above, we used color to highlight a specific set of data points. We used a highlight color for Urban Honolulu and set the other metro areas to a blue with transparency.

10.5.1 Color Scales

An easy way to see the colors within a given color scheme is by using the `show_col()` function in the `scales` package. We can use it to show the colors in the `theme_colors` variable we created above.

```
library(scales)
show_col(theme_colors)
```

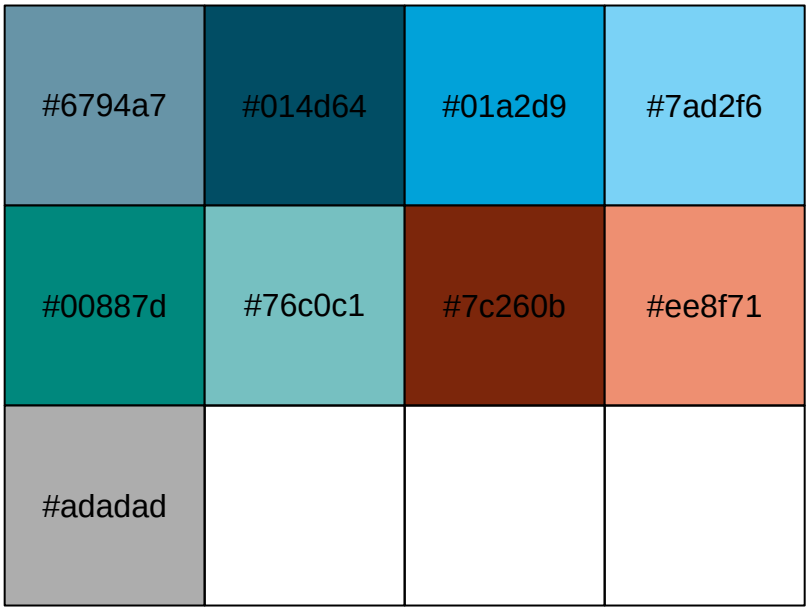


The combinations of letters and numbers in the color squares is the hex representation of the red, green, and blue color values that make up the given color (e.g., `#008FD5`). Adobe has a fun color chooser where you can paste these hex values and create your own color scheme:

https://color.adobe.com/

There are not many colors in the `fivethirtyeight_pal()` color palette (only 3). The `economist_pal()` palette has 11, which is a bit better for categorical data:

```
show_col(economist_pal()(11))
```



10.5.2 Color Brewer

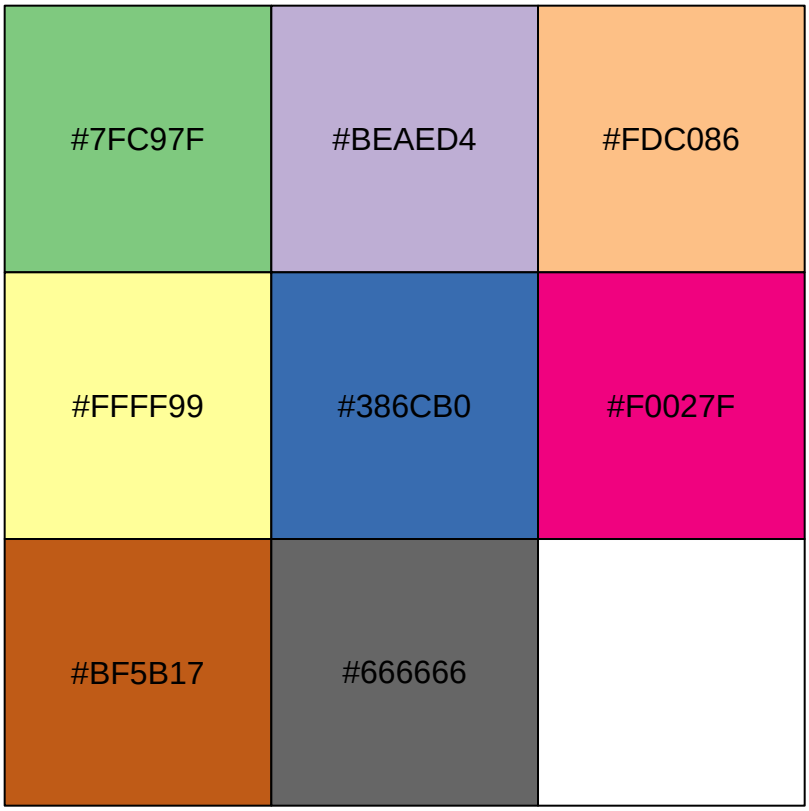
Color Brewer (http://colorbrewer2.org/) is the gold standard for color choice in maps. The online tool allows you to preview and export color schemes that are designed for accessibility (i.e., color-blind safe) and for the main strategies for encoding data using color (sequential, diverging, and qualitative). Most of these scales are available within ggplot (http://ggplot2.tidyverse.org/reference/scale_brewer.html).

```
RColorBrewer::brewer.pal.info
```

```
##          maxcolors category colorblind
## BrBG            11      div       TRUE
```

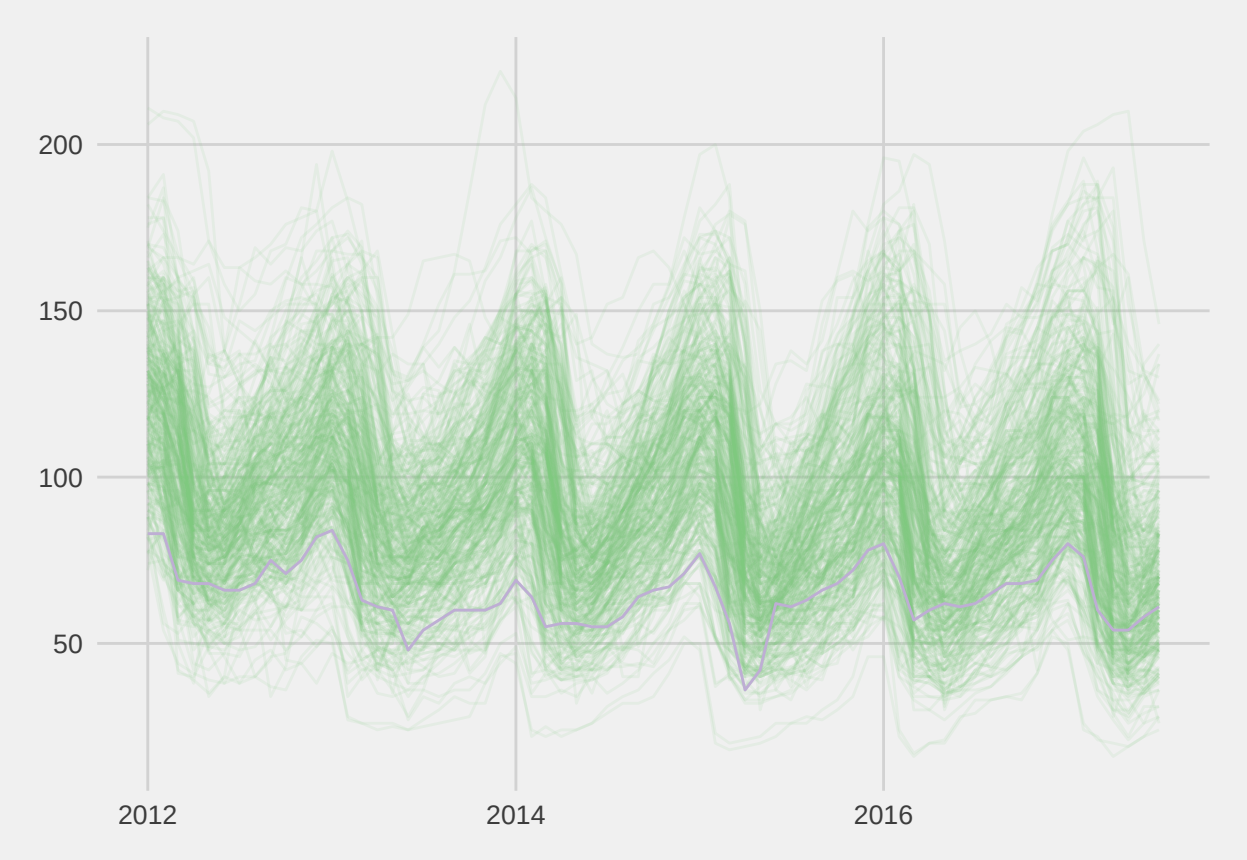
```
## PiYG            11      div       TRUE
## PRGn            11      div       TRUE
## PuOr            11      div       TRUE
## RdBu            11      div       TRUE
## RdGy            11      div      FALSE
## RdYlBu          11      div       TRUE
## RdYlGn          11      div      FALSE
## Spectral        11      div      FALSE
## Accent           8     qual      FALSE
## Dark2            8     qual       TRUE
## Paired          12     qual       TRUE
## Pastel1          9     qual      FALSE
## Pastel2          8     qual      FALSE
## Set1             9     qual      FALSE
## Set2             8     qual       TRUE
## Set3            12     qual      FALSE
## Blues            9      seq       TRUE
## BuGn             9      seq       TRUE
## BuPu             9      seq       TRUE
## GnBu             9      seq       TRUE
## Greens           9      seq       TRUE
## Greys            9      seq       TRUE
## Oranges          9      seq       TRUE
## OrRd             9      seq       TRUE
## PuBu             9      seq       TRUE
## PuBuGn           9      seq       TRUE
## PuRd             9      seq       TRUE
## Purples          9      seq       TRUE
## RdPu             9      seq       TRUE
## Reds             9      seq       TRUE
## YlGn             9      seq       TRUE
## YlGnBu           9      seq       TRUE
## YlOrBr           9      seq       TRUE
## YlOrRd           9      seq       TRUE
```

```
show_col(brewer_pal(palette = "Accent")(8))
```



Here's how to take this color palette and apply it to our previous chart:

```
theme_colors <- brewer_pal(palette = "Accent")(8)
inventory %>% ggplot(aes(month, age, group = RegionName)) +
  geom_line(color = theme_colors[1], alpha = 0.1) +
  geom_line(data = inventory %>% filter(grepl("Honolulu", RegionName)),
            aes(month, age),
            color = theme_colors[2]) +
  theme_fivethirtyeight()
```



10.6 Assignment

Using the same file, pick a different time series to emphasize (with a different color) and choose a different theme.

Chapter 11

Polar Coordinates

This lecture uses the following packages:

```
tidyverse
lubridate
forcats
```

11.1 Data

Survey of Consumers

The University of Michigan conducts the Survey of Consumers. This monthly survey takes the pulse of consumers to help predict the state of the economy in the near future. We will be using a few responses from this monthly survey to highlight how you can visualize periodic data. For full definitions of the variables we will work with take a look at the online codebook: https://data.sca.isr.umich.edu/subset/codebook.php

To download the data

1. Go to the Survey of Consumers' data page: https://data.sca.isr.umich.edu/subset/subset.php
2. In the **Frequency and Range** section, set the **Starting year** to 1998, since that is the first year with complete data on "Probability of Losing a Job During the Next 5 Years" (PJOB)
3. In the **Demographics** section, check all Income groups (y13 = *Bottom 33%*, y23 = *Middle 33%*, and y33 = *Top 33%*)
4. In the **Variables** section, check *PEXP* and *PJOB* in the **Personal Finances** subsection, and check *UMEX* in the **Unemployment, Interest Rates, Prices, Government Expectations** subsection
5. Click on the "Download CSV" button.

Load the downloaded dataset.

```
library(tidyverse)
survey <- read_csv("data/scaum-814.csv")
```

Divide the date column into year and month columns.

```
library(lubridate)
survey <- survey %>%
  mutate(date = parse_date(yyyymm, format = "%Y%m"),
         year = year(date),
         month = month(date)
  ) %>%
```

```
  select(-yyyymm)
survey
```

```
## # A tibble: 235 x 12
##    pexp_r_y13 pexp_r_y23 pexp_r_y33 pjob_mean_y13 pjob_mean_y23
##         <int>      <int>      <int>         <dbl>         <dbl>
##  1        128        153        148          15.3          16.6
##  2        147        143        149          18.9          19.8
##  3        125        141        137          14.4          16.4
##  4        133        134        149          14.2          18.5
##  5        126        129        150          15.8          16.8
##  6        128        139        137          14.4          14.8
##  7        132        142        146          17.0          18.6
##  8        131        143        145          15.8          19.3
##  9        126        135        135          18.1          17.2
## 10        132        134        136          19.4          16.6
## # ... with 225 more rows, and 7 more variables: pjob_mean_y33 <dbl>,
## #   umex_r_y13 <int>, umex_r_y23 <int>, umex_r_y33 <int>, date <date>,
## #   year <dbl>, month <dbl>
```

Each variable is calculated from the survey as follows:

Code	Survey Question	Calculation
PEXP	"Now looking ahead – do you think that a year from now you (and your family living there) will be better off financially, worse off, or just about the same as now?"	Better - Worse + 100
PJOB	"During the next 5 years, what do you think the chances are that you (or your husband/wife) will lose a job you wanted to keep?"	Mean
UMEX	"How about people out of work during the coming 12 months -- do you think that there will be more unemployment than now, about the same, or less?"	Less - More + 100

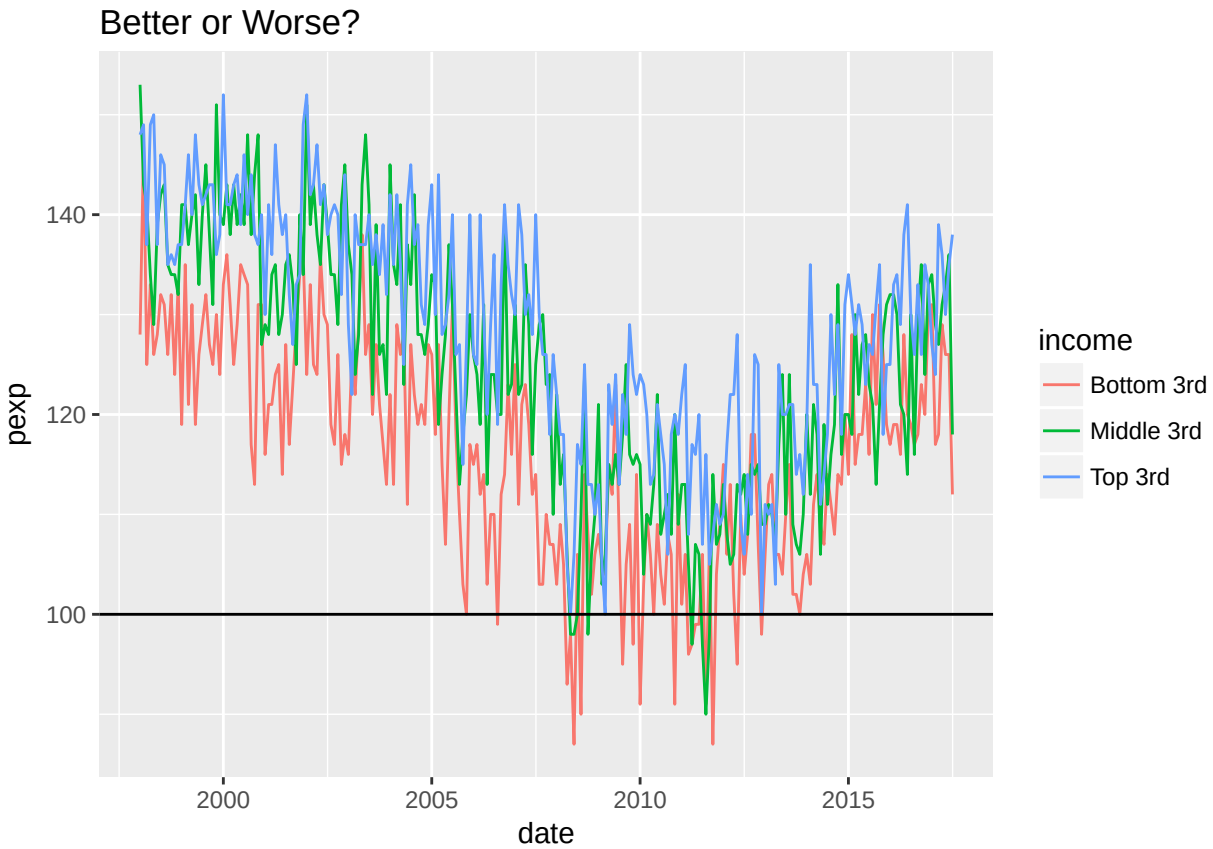
To make our dataset tidy, we want each variable to have it's own row. Since we added the income demographic option, we have multiple columns for each variable. Let's fix that with the `gather()` -> `separate()` -> `spread()` pattern.

```
library(forcats)
survey <- survey %>%
  gather(key = "key", value = "value", -year, -month, -date) %>%
  separate(key, into = c("variable", "type", "income")) %>%
  select(-type) %>%
  spread(key = "variable", value = "value") %>%
  mutate(income = fct_recode(as_factor(income), `Bottom 3rd` = "y13", `Middle 3rd` = "y23", `Top 3rd` =
survey
```

```
## # A tibble: 705 x 7
##          date  year month     income  pexp  pjob  umex
##        <date> <dbl> <dbl>     <fctr> <dbl> <dbl> <dbl>
##  1 1998-01-01  1998     1 Bottom 3rd   128  15.3    90
##  2 1998-01-01  1998     1 Middle 3rd   153  16.6    94
##  3 1998-01-01  1998     1    Top 3rd   148  15.8    99
##  4 1998-02-01  1998     2 Bottom 3rd   147  18.9   103
##  5 1998-02-01  1998     2 Middle 3rd   143  19.8   103
##  6 1998-02-01  1998     2    Top 3rd   149  14.7   100
##  7 1998-03-01  1998     3 Bottom 3rd   125  14.4    90
##  8 1998-03-01  1998     3 Middle 3rd   141  16.4   106
##  9 1998-03-01  1998     3    Top 3rd   137  17.8   104
## 10 1998-04-01  1998     4 Bottom 3rd   133  14.2   101
## # ... with 695 more rows
```

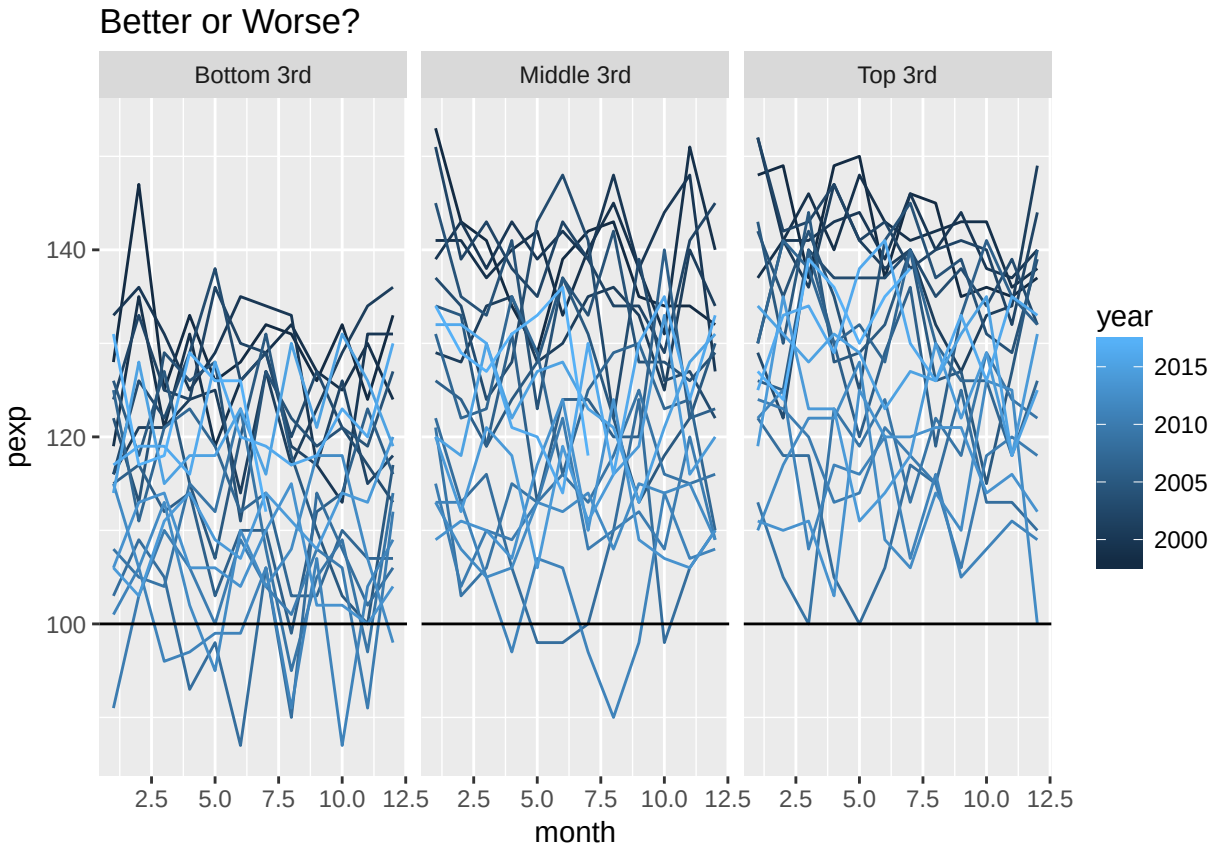
11.2 Simple Time Series

```
ggplot(survey, aes(date, pexp, color = income)) +
  geom_line() +
  geom_hline(yintercept = 100) +
  ggtitle("Better or Worse?")
```



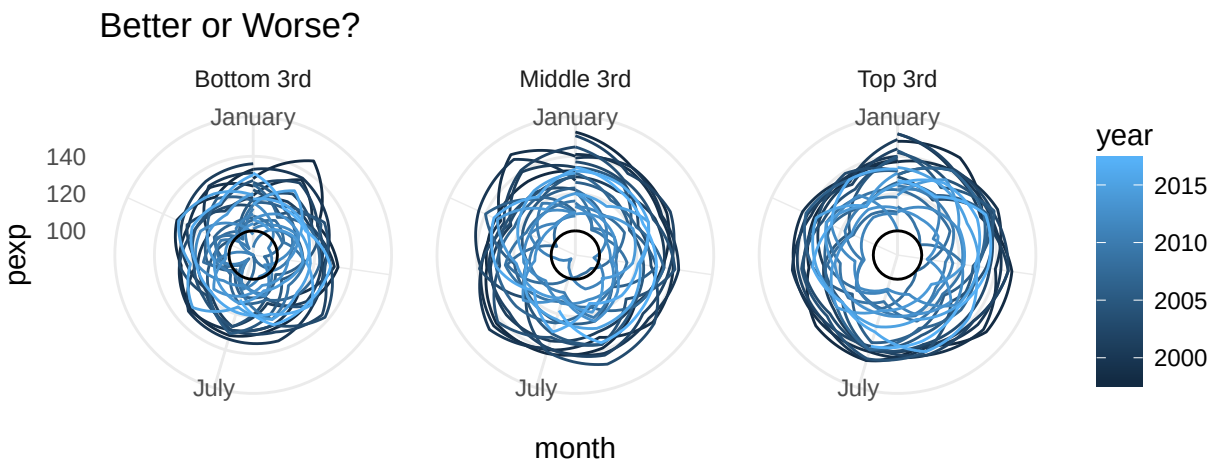
11.3 Stacked Periods

```
better <- ggplot(survey, aes(month, pexp, group = year, color = year)) +
  geom_line() +
  facet_wrap(~ income) +
  geom_hline(yintercept = 100) +
  ggtitle("Better or Worse?")
better
```



11.4 Polar Coordinates

```
better +
  coord_polar(theta = "x") +
  scale_x_continuous(breaks = c(1, 7), labels = month.name[c(1, 7)]) +
  theme_minimal()
```



11.5 Assignment

Plot the other two variables (`pjob` and `umex`) in polar coordinates giving each plot a title that helps communicate the meaning of their respective variable.

11.6 Data Attribution

> Source: Survey of Consumer Expectations, © 2013-2017 Federal Reserve Bank of New York (FRBNY). The SCE data are available without charge at /microeconomics/sce and may be used subject to license terms posted below. FRBNY disclaims any responsibility or legal liability for this analysis and interpretation of Survey of Consumer Expectations data.

Chapter 12

Text Analysis

This lecture uses the following packages:

`tidyverse`
`tidytext`
`twitteR`
`scales`

12.1 Data

12.1.1 Twitter

The content of recent Tweets can be downloaded using Twitter's Application Pragramming Interface (API). We will make use of a package, `twitteR`, that is designed to make working with this API easier.

In class I will provide you with an API Key and API Secret that you can use. Outside of class, you will need to set up your own Twitter application at https://apps.twitter.com.

12.1.2 Setting up twitteR

Install the `twitteR` package:

```
install.packages("twitteR")
```

Load the package:

```
library(twitteR)
```

Finally, authenticate with twitter using the `API Key`, `API Secret`, `Access Token`, and `Access Secret` from the application for this course or your own. `setup_twitter_oauth("API key", "API secret", "Access token", "Access secret")`

12.1.3 Grabbing a few tweets

We'll store 1000 tweets that contain `income` in `rstatTweets`:

```
income_tweets <- searchTwitter('income', n=10000)
```

Each tweet is stored as a `twitteR::status` object. To make it easy to gather the data we want to analyze let's create a function that will return all the columns in our soon to be created data frame.

```
simple_status <- function(status) {
  status$toDataFrame()
}
```

There are three ways we can use `map_df` to get the columns:

```
map_df(simple_status) # named function
map_df(function(x) x$toDataFrame()) # anonymous function
map_df(~ .$toDataFrame()) # formula
```

All of those options do the same thing. They all return a data frame with one row representing a tweet. Let's make use of the formula version to create our data frame:

```
library(tidyverse)
income_df <- income_tweets %>%
  map_df(~ .$toDataFrame())
head(income_df)
```

```
##
## 1    RT @theamwu: "I'm hugely nervous... I wouldn't be able to manage on a reduced income." Michelle
## 2      RT @DMR4USSenateCA: @RobertBentley76 @beinlibertarian @ToddHagopian @LarrySharpe @Liberty_Thur
## 3    Not excited about your #job? I'm looking for ppl who #dreambig. Join my #team &amp;amp; keep your j
## 4 RT @3yeAmHe: Income is not wealth.\nIncome is not wealth.\nIncome is not wealth.\nIncome is not wea
## 5                                                 6                                https://t.co/MlPZxr6
## 6      RT @MazMHussain: 2032: Millions of Americans rendered superfluous by automation are pacified
##   favorited favoriteCount replyToSN             created truncated
## 1     FALSE             0      <NA> 2017-10-30 03:00:42     FALSE
## 2     FALSE             0      <NA> 2017-10-30 03:00:41     FALSE
## 3     FALSE             0      <NA> 2017-10-30 03:00:37     FALSE
## 4     FALSE             0      <NA> 2017-10-30 03:00:31     FALSE
## 5     FALSE             0      <NA> 2017-10-30 03:00:29     FALSE
## 6     FALSE             0      <NA> 2017-10-30 03:00:25     FALSE
##   replyToSID                 id replyToUID
## 1       <NA> 924833413231546368       <NA>
## 2       <NA> 924833408924217345       <NA>
## 3       <NA> 924833388971872256       <NA>
## 4       <NA> 924833365521551366       <NA>
## 5       <NA> 924833358835802112       <NA>
## 6       <NA> 924833340221480960       <NA>
##                                                                                                statusSou
## 1                      <a href="http://twitter.com/#!/download/ipad" rel="nofollow">Twitter for iPad<
## 2 <a href="https://www.reddit.com/user/alllibertynews/m/libertynews" rel="nofollow">AllLibertyNews3<
## 3                                                <a href="https://ifttt.com" rel="nofollow">IFTTT<
## 4                    <a href="http://twitter.com/download/iphone" rel="nofollow">Twitter for iPhone<
## 5                                  <a href="http://publicize.wp.com/" rel="nofollow">WordPress.com<
## 6                  <a href="http://twitter.com/download/android" rel="nofollow">Twitter for Android<
##        screenName retweetCount isRetweet retweeted longitude latitude
## 1       bsadams25          142      TRUE     FALSE      <NA>     <NA>
## 2  alllibertynews            1      TRUE     FALSE      <NA>     <NA>
## 3 LifeLeadership4            0     FALSE     FALSE      <NA>     <NA>
## 4   pardonmeimrae           18      TRUE     FALSE      <NA>     <NA>
## 5       EthereumC            0     FALSE     FALSE      <NA>     <NA>
## 6   SnigdhChandra           52      TRUE     FALSE      <NA>     <NA>
```

12.2 Tidytext

The `tidytext` package helps us use all the tools in the tidyverse alongside text data. The key tool we'll use here is `unnest_tokens()`

```
library(tidytext)
income_words <- income_df %>%
  mutate(text = gsub("\n|[[:digit:][:punct:]]+", "", text)) %>%
  unnest_tokens(word, text) %>%
  anti_join(stop_words)
income_words %>%
  count(word, sort = TRUE)
```

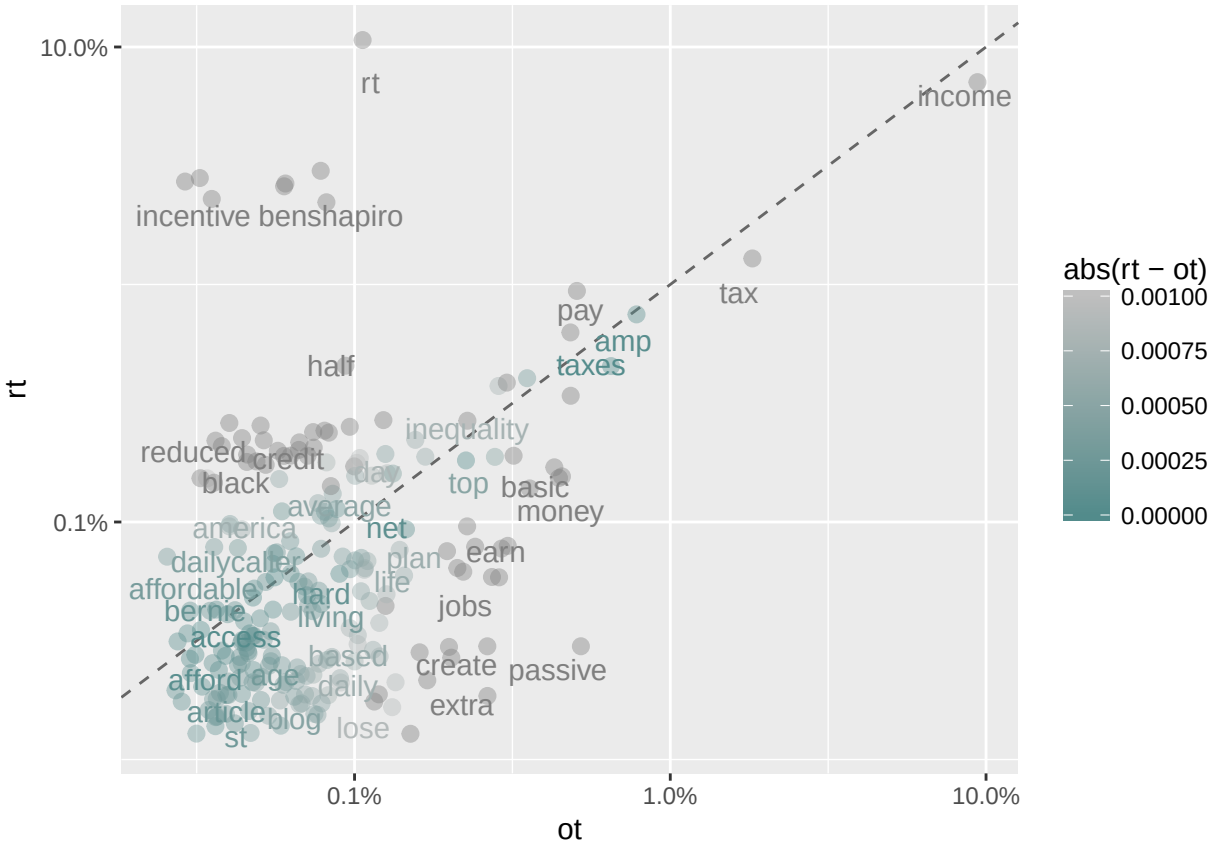
```
## # A tibble: 15,299 x 2
##          word     n
##         <chr> <int>
##  1     income  8523
##  2         rt  6237
##  3    httpstc  1751
##  4    service  1746
##  5       true  1746
##  6 benshapiro  1741
##  7    freedom  1738
##  8    seizing  1727
##  9    product  1726
## 10  incentive  1722
## # ... with 15,289 more rows
```

Let's compare the words in original tweets (ot) to those found in retweets (rt).

```
library(scales)
type_proportions <- income_words %>%
  mutate(is_retweet = ifelse(isRetweet, "rt", "ot")) %>%
  group_by(is_retweet) %>%
  count(word, sort = TRUE) %>%
  mutate(proportion = n / sum(n)) %>%
  filter(n > 10) %>%
  select(is_retweet, word, proportion) %>%
  spread(is_retweet, proportion)
type_proportions
```

```
## # A tibble: 865 x 3
##              word           ot           rt
##  *          <chr>        <dbl>        <dbl>
##  1        aboogle           NA 0.0004901068
##  2            aca           NA 0.0003080672
##  3         access 0.0004209227 0.0004060885
##  4     accessible           NA 0.0001960427
##  5       accident           NA 0.0001820397
##  6         actual           NA 0.0002660580
##  7             ad 0.0003086766           NA
##  8          added 0.0004489842           NA
##  9         adding           NA 0.0002380519
## 10 addmefastwwwim           NA 0.0002240488
## # ... with 855 more rows
```

```
type_proportions %>%
  ggplot(aes(ot, rt, color = abs(rt - ot))) +
  geom_abline(color = "gray40", lty = 2) +
  geom_jitter(alpha = 0.4, size = 2.5, height = 0.1, width = 0.1) +
  geom_text(aes(label = word), check_overlap = TRUE, vjust = 1.5) +
  scale_x_log10(labels = percent_format()) +
  scale_y_log10(labels = percent_format()) +
  scale_color_gradient(limits = c(0, 0.001), low = "darkslategray4", high = "gray75")
```



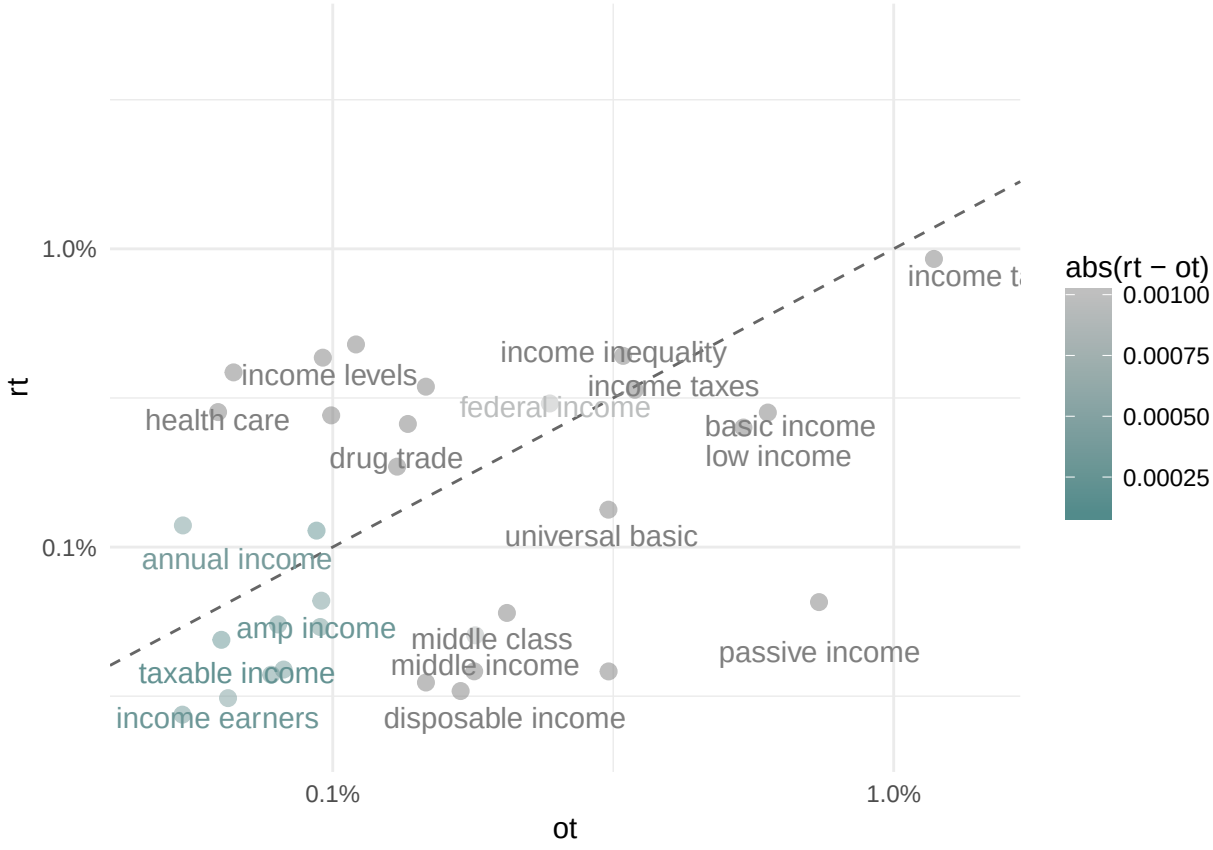
12.3 N-grams

An n-gram is a sequence of n tokens. For fun with n-grams, check out Google's Ngram Viewer.

Let's compare bigrams (two-word n-grams) in our tweets across retweets (rt) and original tweets (ot).

```
bigrams <- income_df %>%
  mutate(text = gsub("\n|[[:digit:][:punct:]]+", "", text)) %>%
  unnest_tokens(word, text, token = "ngrams", n = 2) %>%
  separate(word, c("word1", "word2"), sep = " ") %>%
  filter(!word1 %in% stop_words$word & !word2 %in% stop_words$word) %>%
  unite(word, word1, word2, sep = " ")

bigram_proportions <- bigrams %>%
  mutate(is_retweet = ifelse(isRetweet, "rt", "ot")) %>%
  group_by(is_retweet) %>%
  count(word, sort = TRUE) %>%
```

```
  mutate(proportion = n / sum(n)) %>%
  filter(n > 10) %>%
  select(is_retweet, word, proportion) %>%
  filter(is_retweet != '') %>%
  spread(is_retweet, proportion)

bigram_proportions %>%
  ggplot(aes(ot, rt, color = abs(rt - ot))) +
  geom_abline(color = "gray40", lty = 2) +
  geom_jitter(alpha = 0.5, size = 2.5, height = 0.1, width = 0.1) +
  geom_text(aes(label = word), check_overlap = TRUE, vjust = 1.5) +
  scale_x_log10(labels = percent_format()) +
  scale_y_log10(labels = percent_format()) +
  scale_color_gradient(limits = c(0.0001, 0.001), low = "darkslategray4", high = "gray75") +
  theme_minimal()
```



12.3.1 Skip N-grams

Skip n-grams are phrases of n kept words with at most k words that are skipped between each word that is kept. Suppose $n = 3$ and $k = 2$, with the input phrase "the rain in Spain falls mainly in the plain," the output will be.

```
tokenizers::tokenize_skip_ngrams("the rain in Spain falls mainly in the plain", n = 3, k = 2)
```

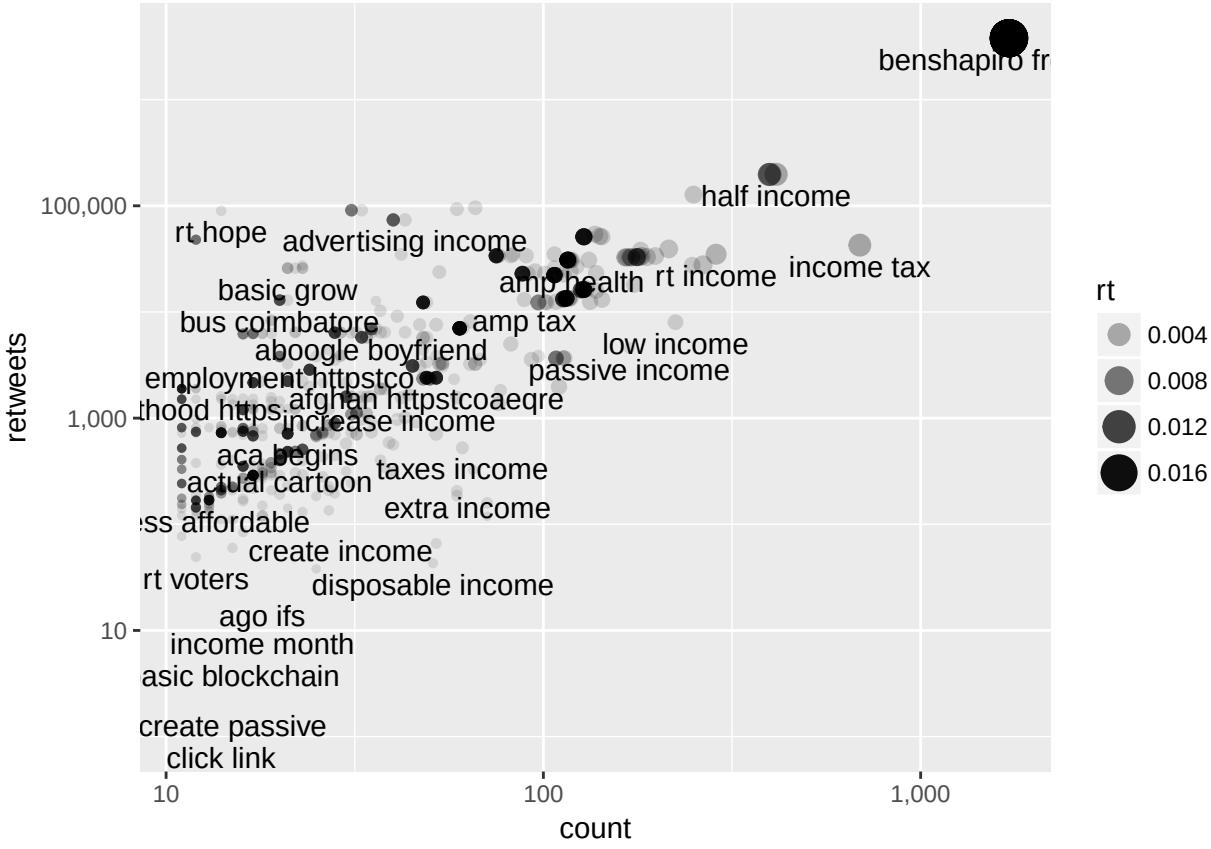
```
## [[1]]
##  [1] "the spain in"       "rain falls the"     "in mainly plain"
```

```
##  [4] "the in falls"       "rain spain mainly"  "in falls in"
##  [7] "spain mainly the"   "falls in plain"     "the rain in"
## [10] "rain in spain"      "in spain falls"     "spain falls mainly"
## [13] "falls mainly in"    "mainly in the"      "in the plain"
```

Let's gather simple two-word skip ngrams with up to two skipped words between each kept word.

```
skipgrams <- income_df %>%
  mutate(text = gsub("\n|[[:digit:][:punct:]]+", "", text)) %>%
  unnest_tokens(word, text, token = "skip_ngrams", n = 2, k = 2) %>%
  separate(word, c("word1", "word2"), sep = " ") %>%
  filter(!word1 %in% stop_words$word & !word2 %in% stop_words$word) %>%
  unite(word, word1, word2, sep = " ")
```

```
retweet_counts <- skipgrams %>%
  group_by(word) %>%
  summarise(retweets = sum(retweetCount), count = n())

type_proportions <- skipgrams %>%
  mutate(is_retweet = ifelse(isRetweet, "rt", "ot")) %>%
  group_by(is_retweet) %>%
  count(word) %>%
  mutate(proportion = n / sum(n)) %>%
  filter(n > 10) %>%
  select(is_retweet, word, proportion) %>%
  spread(is_retweet, proportion) %>%
  merge(retweet_counts)

type_proportions %>%
  ggplot(aes(count, retweets)) +
  geom_point(aes(size = rt, alpha = rt)) +
  geom_text(aes(label = word), check_overlap = TRUE, vjust = 1.5) +
  scale_x_log10(labels = comma_format()) +
  scale_y_log10(labels = comma_format())
```



12.4 Assignment

(1) Use searchTwitter to download tweets about another topic. (2) Create a plot that compares word choice across android and iPhone devices using the following `mutate()` expression.

```
mutate(type = case_when(
  grepl("android", statusSource) ~ "android",
  grepl("iPhone", statusSource) ~ "iPhone",
  TRUE ~ "other"
))
```

Part II

Topics

Data Sources Overview

While you can find many data sources by typing `public data sources` in your favorite search engine, the following lists should help you get started.

Macro Data

US

- Federal Reserve Economic Data (FRED)
- Bureau of Labor Statistics
- Bureau of Economic Analysis
- National Bureau of Economic Research
- Congressional Budget Office: Budget and Economic Data
- American FactFinder (American Community Survey, Census Summary Files, etc.)

Other US

- The Conference Board (includes consumer confidence index)
- Survey of Consumers
- Historical Exchange Rates
- Center for Medicare and Medicaid Services

Micro Data

- Panel Study on Income Dynamics
- IPUMS (census and survey data)
- US Census Public Use Microdata Sample (PUMS)
- Center for Medicare and Medicaid Services

Hawaii Data

- State of Hawaii Department of Business, Economic Development and Tourism (DBEDT)

Collections of data lists

- American Economic Association (AEA) list of data sources

Anscombe's Quartet

Anscombe quartet emphasizes the need to move beyond basic numerical summaries of your data. The `anscombe` dataset has four sets of `x` and `y` variables with very similar summaries, but distinct visual patterns

Prep the data

```
anscombe
```

```
##    x1 x2 x3 x4    y1   y2    y3    y4
## 1  10 10 10  8  8.04 9.14  7.46  6.58
## 2   8  8  8  8  6.95 8.14  6.77  5.76
## 3  13 13 13  8  7.58 8.74 12.74  7.71
## 4   9  9  9  8  8.81 8.77  7.11  8.84
## 5  11 11 11  8  8.33 9.26  7.81  8.47
## 6  14 14 14  8  9.96 8.10  8.84  7.04
## 7   6  6  6  8  7.24 6.13  6.08  5.25
## 8   4  4  4 19  4.26 3.10  5.39 12.50
## 9  12 12 12  8 10.84 9.13  8.15  5.56
## 10  7  7  7  8  4.82 7.26  6.42  7.91
## 11  5  5  5  8  5.68 4.74  5.73  6.89
```

First we'll use `tidyr` to reshape the anscombe dataset to make it easier to work with. We want a column to identify each point, `id`, a column for the series (`x1` is the `x` value in series `1`), and columns for `x` and `y`. In the case of the `anscombe` dataset, rows group `x` and `y` vaules, but are not important across series.

```
library(tidyverse)
tidy_anscombe <- anscombe %>%
  mutate(id = row_number()) %>%
  gather(key = key, value = value, everything(), -id)
tidy_anscombe %>% as.tbl
```

```
## # A tibble: 88 x 3
##       id   key value
##    <int> <chr> <dbl>
##  1     1    x1    10
##  2     2    x1     8
##  3     3    x1    13
##  4     4    x1     9
##  5     5    x1    11
##  6     6    x1    14
##  7     7    x1     6
##  8     8    x1     4
##  9     9    x1    12
```

```
## 10    10    x1     7
## # ... with 78 more rows
```

Now we want can split the `key` column into an `x_or_y` column and a `series` column.

```
tidy_anscombe <- tidy_anscombe %>%
  separate(key, c("x_or_y", "series"), 1)
tidy_anscombe %>% as.tbl
```

```
## # A tibble: 88 x 4
##       id x_or_y series value
##  * <int>  <chr>  <chr> <dbl>
##  1     1      x      1    10
##  2     2      x      1     8
##  3     3      x      1    13
##  4     4      x      1     9
##  5     5      x      1    11
##  6     6      x      1    14
##  7     7      x      1     6
##  8     8      x      1     4
##  9     9      x      1    12
## 10    10      x      1     7
## # ... with 78 more rows
```

Now we can use `spread()` to create the final form of our table, regrouping the associated x and y values. We could have done something simpler since we knew there were only 4 series, but the code we used will work for an arbitrary number of series.

```
tidy_anscombe <- tidy_anscombe %>%
  spread(x_or_y, value)
tidy_anscombe %>% as.tbl
```

```
## # A tibble: 44 x 4
##       id series     x     y
##  * <int>  <chr> <dbl> <dbl>
##  1     1      1    10  8.04
##  2     1      2    10  9.14
##  3     1      3    10  7.46
##  4     1      4     8  6.58
##  5     2      1     8  6.95
##  6     2      2     8  8.14
##  7     2      3     8  6.77
##  8     2      4     8  5.76
##  9     3      1    13  7.58
## 10     3      2    13  8.74
## # ... with 34 more rows
```

Numeric summary

```
tidy_anscombe %>%
  group_by(series) %>%
  summarise(
    mean_x = mean(x),
    mean_y = mean(y),
```

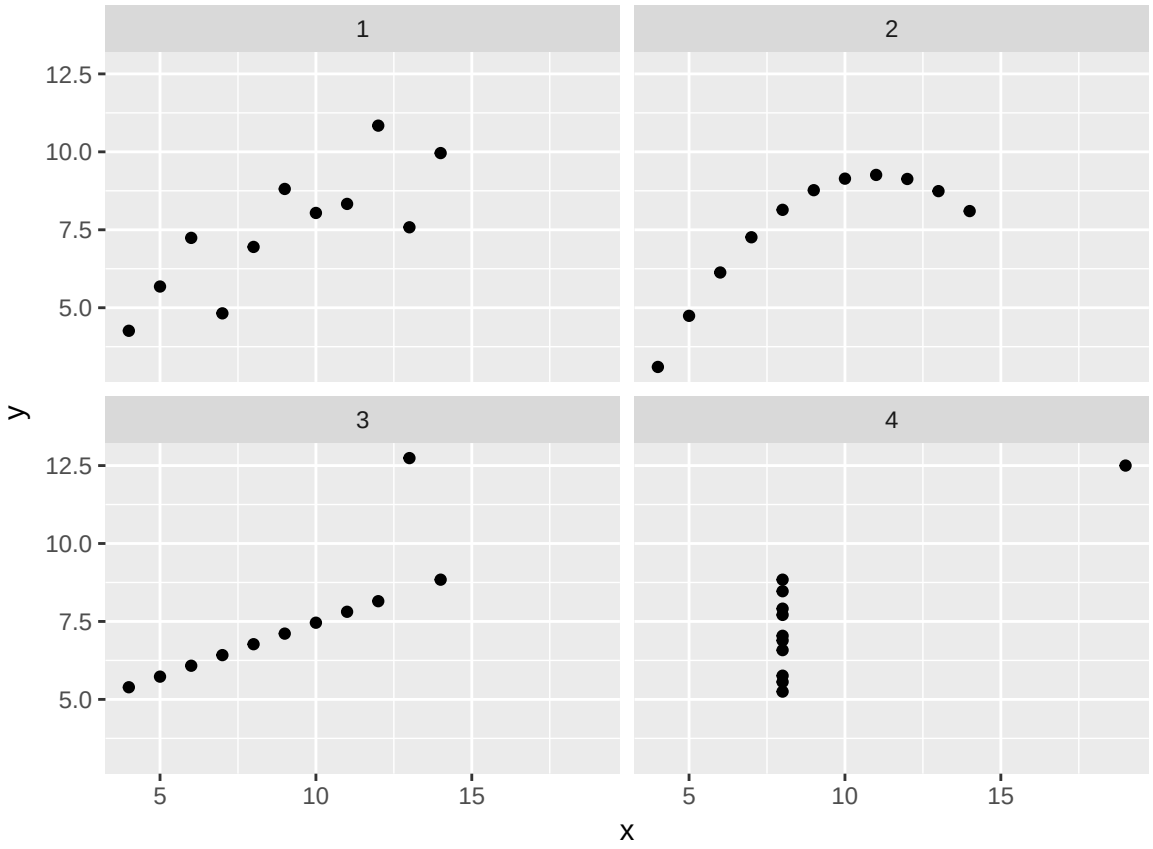
```
    sd_x = sd(x),
    sd_y = sd(y),
    cor = cor(x, y)
  )
```

```
## # A tibble: 4 x 6
##   series mean_x   mean_y     sd_x     sd_y       cor
##    <chr>  <dbl>    <dbl>    <dbl>    <dbl>     <dbl>
## 1      1      9 7.500909 3.316625 2.031568 0.8164205
## 2      2      9 7.500909 3.316625 2.031657 0.8162365
## 3      3      9 7.500000 3.316625 2.030424 0.8162867
## 4      4      9 7.500909 3.316625 2.030579 0.8165214
```

Visual summary

While the numeric summaries suggest very similar datasets, the visual summaries help identify the differences:

```
library(ggplot2)
tidy_anscombe %>%
  ggplot(aes(x, y)) +
  geom_point() +
  facet_wrap(~ series) +
  coord_fixed()
```



The Datasaurus Dozen

The Datasaurus Dozen is a set of series, like Anscombe's quartet, with similar numerical summaries and radically different visual summaries. See a great discussion of this dataset by the creators, Justin Matejka and George Fitzmaurice here

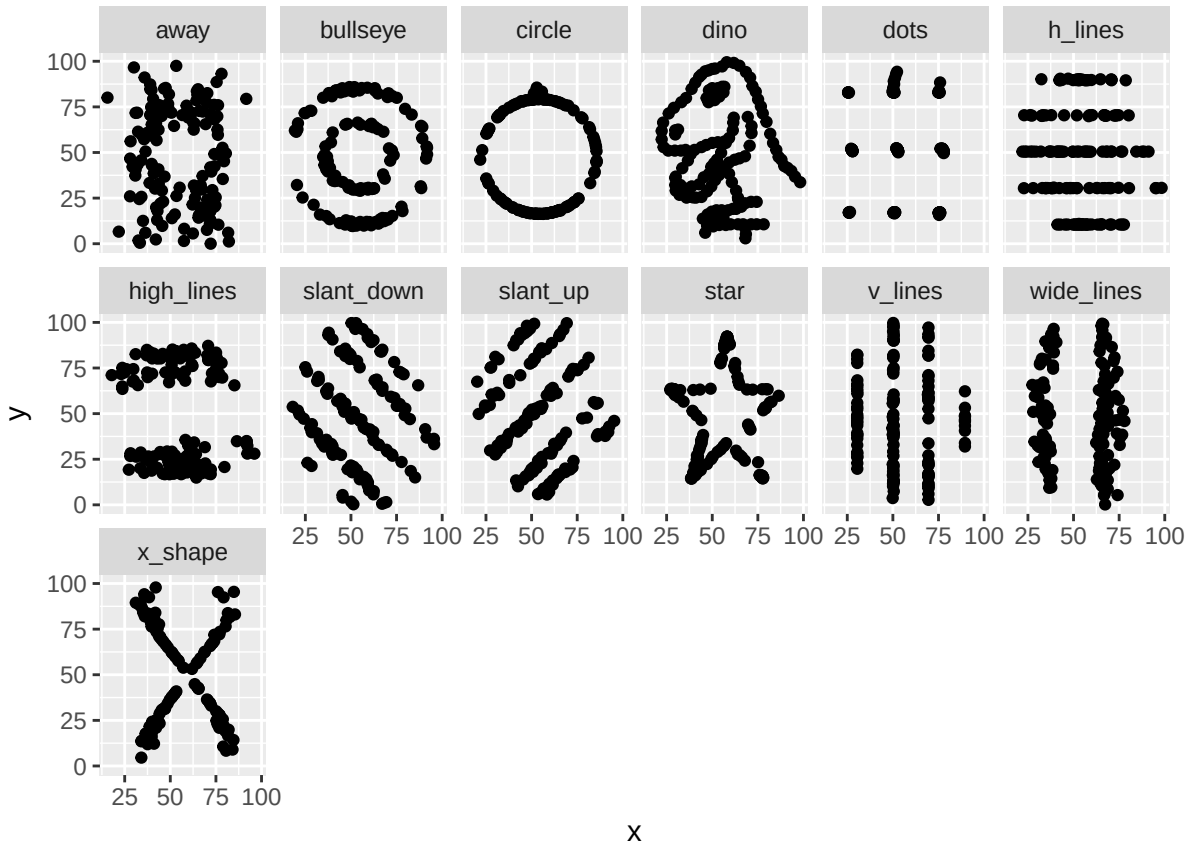
Download the data here and move the DatasaurusDozen.tsv file into your data folder.

```
datasaurus <- read_tsv("data/DatasaurusDozen.tsv")
datasaurus %>%
  group_by(dataset) %>%
  summarise(
    mean_x = mean(x),
    mean_y = mean(y),
    sd_x = sd(x),
    sd_y = sd(y),
    cor = cor(x, y)
  )
```

```
## # A tibble: 13 x 6
##       dataset   mean_x   mean_y     sd_x     sd_y         cor
##         <chr>    <dbl>    <dbl>    <dbl>    <dbl>       <dbl>
##  1       away 54.26610 47.83472 16.76982 26.93974 -0.06412835
##  2   bullseye 54.26873 47.83082 16.76924 26.93573 -0.06858639
##  3     circle 54.26732 47.83772 16.76001 26.93004 -0.06834336
##  4       dino 54.26327 47.83225 16.76514 26.93540 -0.06447185
##  5       dots 54.26030 47.83983 16.76774 26.93019 -0.06034144
##  6    h_lines 54.26144 47.83025 16.76590 26.93988 -0.06171484
##  7 high_lines 54.26881 47.83545 16.76670 26.94000 -0.06850422
##  8 slant_down 54.26785 47.83590 16.76676 26.93610 -0.06897974
##  9   slant_up 54.26588 47.83150 16.76885 26.93861 -0.06860921
## 10       star 54.26734 47.83955 16.76896 26.93027 -0.06296110
## 11    v_lines 54.26993 47.83699 16.76996 26.93768 -0.06944557
## 12 wide_lines 54.26692 47.83160 16.77000 26.93790 -0.06657523
## 13    x_shape 54.26015 47.83972 16.76996 26.93000 -0.06558334
```

Visual summaries

```
datasaurus %>%
  ggplot(aes(x, y)) +
  geom_point() +
  facet_wrap(~ dataset, ncol = 6) +
  coord_fixed()
```


away
bullseye
circle
dino
dots
h_lines
high_lines
slant_down
slant_up
star
v_lines
wide_lines
x_shape
y
x
0
25
50
75
100

Probability

The source for this topic is the Open Intro labs https://www.openintro.org/stat/labs.php

Hot Hands

Basketball players who make several baskets in succession are described as having a *hot hand*. Fans and players have long believed in the hot hand phenomenon, which refutes the assumption that each shot is independent of the next. However, a 1985 paper by Gilovich, Vallone, and Tversky collected evidence that contradicted this belief and showed that successive shots are independent events (http://www.cs.colorado.edu/~mozer/Teaching/syllabi/7782/readings/gilovich%20vallone%20tversky.pdf). This paper started a great controversy that continues to this day, as you can see by Googling *hot hand basketball*.

We do not expect to resolve this controversy today. However, in this lab we'll apply one approach to answering questions like this. The goals for this lab are to (1) think about the effects of independent and dependent events, (2) learn how to simulate shooting streaks in R, and (3) to compare a simulation to actual data in order to determine if the hot hand phenomenon appears to be real.

Saving your code

Click on File -> New -> R Script. This will open a blank document above the console. As you go along you can copy and paste your code here and save it. This is a good way to keep track of your code and be able to reuse it later. To run your code from this document you can either copy and paste it into the console, highlight the code and hit the Run button, or highlight the code and hit command+enter on a mac or control+enter on a PC.

You'll also want to save this script (code document). To do so click on the disk icon. The first time you hit save, RStudio will ask for a file name; you can name it anything you like. Once you hit save you'll see the file appear under the Files tab in the lower right panel. You can reopen this file anytime by simply clicking on it.

Getting Started

Our investigation will focus on the performance of one player: Kobe Bryant of the Los Angeles Lakers. His performance against the Orlando Magic in the 2009 NBA finals earned him the title *Most Valuable Player* and many spectators commented on how he appeared to show a hot hand. Let's load some data from those games and look at the first several rows.

```
load("data/kobe.RData")
head(kobe)
```

In this data frame, every row records a shot taken by Kobe Bryant. If he hit the shot (made a basket), a hit, `H`, is recorded in the column named `basket`, otherwise a miss, `M`, is recorded.

Just looking at the string of hits and misses, it can be difficult to gauge whether or not it seems like Kobe was shooting with a hot hand. One way we can approach this is by considering the belief that hot hand shooters tend to go on shooting streaks. For this lab, we define the length of a shooting streak to be the *number of consecutive baskets made until a miss occurs.*

For example, in Game 1 Kobe had the following sequence of hits and misses from his nine shot attempts in the first quarter:

H M | M | H H M | M | M | M

To verify this use the following command:

```
kobe$basket[1:9]
```

Within the nine shot attempts, there are six streaks, which are separated by a "|" above. Their lengths are one, zero, two, zero, zero, zero (in order of occurrence).

1. What does a streak length of 1 mean, i.e. how many hits and misses are in a streak of 1? What about a streak length of 0?

The custom function `calc_streak`, which was loaded in with the data, may be used to calculate the lengths of all shooting streaks and then look at the distribution.

```
kobe_streak <- calc_streak(kobe$basket)
barplot(table(kobe_streak))
```

Note that instead of making a histogram, we chose to make a bar plot from a table of the streak data. A bar plot is preferable here since our variable is discrete – counts – instead of continuous.

2. Describe the distribution of Kobe's streak lengths from the 2009 NBA finals. What was his typical streak length? How long was his longest streak of baskets?

Compared to What?

We've shown that Kobe had some long shooting streaks, but are they long enough to support the belief that he had hot hands? What can we compare them to?

To answer these questions, let's return to the idea of *independence*. Two processes are independent if the outcome of one process doesn't effect the outcome of the second. If each shot that a player takes is an independent process, having made or missed your first shot will not affect the probability that you will make or miss your second shot.

A shooter with a hot hand will have shots that are *not* independent of one another. Specifically, if the shooter makes his first shot, the hot hand model says he will have a *higher* probability of making his second shot.

Let's suppose for a moment that the hot hand model is valid for Kobe. During his career, the percentage of time Kobe makes a basket (i.e. his shooting percentage) is about 45%, or in probability notation,

$$P(\text{shot } 1 = \text{H}) = 0.45$$

If he makes the first shot and has a hot hand (*not* independent shots), then the probability that he makes his second shot would go up to, let's say, 60%,

$$P(\text{shot } 2 = \text{H} \,|\, \text{shot } 1 = \text{H}) = 0.60$$

As a result of these increased probabilites, you'd expect Kobe to have longer streaks. Compare this to the skeptical perspective where Kobe does *not* have a hot hand, where each shot is independent of the next. If he hit his first shot, the probability that he makes the second is still 0.45.

$$P(\text{shot } 2 = \text{H} \,|\, \text{shot } 1 = \text{H}) = 0.45$$

In other words, making the first shot did nothing to effect the probability that he'd make his second shot. If Kobe's shots are independent, then he'd have the same probability of hitting every shot regardless of his past shots: 45%.

Now that we've phrased the situation in terms of independent shots, let's return to the question: how do we tell if Kobe's shooting streaks are long enough to indicate that he has hot hands? We can compare his streak lengths to someone without hot hands: an independent shooter.

Simulations in R

While we don't have any data from a shooter we know to have independent shots, that sort of data is very easy to simulate in R. In a simulation, you set the ground rules of a random process and then the computer uses random numbers to generate an outcome that adheres to those rules. As a simple example, you can simulate flipping a fair coin with the following.

```
outcomes <- c("heads", "tails")
sample(outcomes, size = 1, replace = TRUE)
```

The vector `outcomes` can be thought of as a hat with two slips of paper in it: one slip says `heads` and the other says `tails`. The function `sample` draws one slip from the hat and tells us if it was a head or a tail.

Run the second command listed above several times. Just like when flipping a coin, sometimes you'll get a heads, sometimes you'll get a tails, but in the long run, you'd expect to get roughly equal numbers of each.

If you wanted to simulate flipping a fair coin 100 times, you could either run the function 100 times or, more simply, adjust the `size` argument, which governs how many samples to draw (the `replace = TRUE` argument indicates we put the slip of paper back in the hat before drawing again). Save the resulting vector of heads and tails in a new object called `sim_fair_coin`.

```
sim_fair_coin <- sample(outcomes, size = 100, replace = TRUE)
```

To view the results of this simulation, type the name of the object and then use `table` to count up the number of heads and tails.

```
sim_fair_coin
table(sim_fair_coin)
```

Since there are only two elements in `outcomes`, the probability that we "flip" a coin and it lands heads is 0.5. Say we're trying to simulate an unfair coin that we know only lands heads 20% of the time. We can adjust for this by adding an argument called `prob`, which provides a vector of two probability weights.

```
sim_unfair_coin <- sample(outcomes, size = 100, replace = TRUE, prob = c(0.2, 0.8))
```

`prob=c(0.2, 0.8)` indicates that for the two elements in the `outcomes` vector, we want to select the first one, `heads`, with probability 0.2 and the second one, `tails` with probability 0.8. Another way of thinking about this is to think of the outcome space as a bag of 10 chips, where 2 chips are labeled "head" and 8 chips "tail". Therefore at each draw, the probability of drawing a chip that says "head"" is 20%, and "tail" is 80%.

3. In your simulation of flipping the unfair coin 100 times, how many flips came up heads?

In a sense, we've shrunken the size of the slip of paper that says "heads", making it less likely to be drawn and we've increased the size of the slip of paper saying "tails", making it more likely to be drawn. When we simulated the fair coin, both slips of paper were the same size. This happens by default if you don't provide a `prob` argument; all elements in the `outcomes` vector have an equal probability of being drawn.

If you want to learn more about `sample` or any other function, recall that you can always check out its help file.

```
?sample
```

Simulating the Independent Shooter

Simulating a basketball player who has independent shots uses the same mechanism that we use to simulate a coin flip. To simulate a single shot from an independent shooter with a shooting percentage of 50% we type,

```
outcomes <- c("H", "M")
sim_basket <- sample(outcomes, size = 1, replace = TRUE)
```

To make a valid comparison between Kobe and our simulated independent shooter, we need to align both their shooting percentage and the number of attempted shots.

4. What change needs to be made to the `sample` function so that it reflects a shooting percentage of 45%? Make this adjustment, then run a simulation to sample 133 shots. Assign the output of this simulation to a new object called `sim_basket`.

Note that we've named the new vector `sim_basket`, the same name that we gave to the previous vector reflecting a shooting percentage of 50%. In this situation, R overwrites the old object with the new one, so always make sure that you don't need the information in an old vector before reassigning its name.

With the results of the simulation saved as `sim_basket`, we have the data necessary to compare Kobe to our independent shooter. We can look at Kobe's data alongside our simulated data.

```
kobe$basket
sim_basket
```

Both data sets represent the results of 133 shot attempts, each with the same shooting percentage of 45%. We know that our simulated data is from a shooter that has independent shots. That is, we know the simulated shooter does not have a hot hand.

On your own

Comparing Kobe Bryant to the Independent Shooter

Using `calc_streak`, compute the streak lengths of `sim_basket`.

- Describe the distribution of streak lengths. What is the typical streak length for this simulated independent shooter with a 45% shooting percentage? How long is the player's longest streak of baskets in 133 shots?
- If you were to run the simulation of the independent shooter a second time, how would you expect its streak distribution to compare to the distribution from the question above? Exactly the same? Somewhat similar? Totally different? Explain your reasoning.

- How does Kobe Bryant's distribution of streak lengths compare to the distribution of streak lengths for the simulated shooter? Using this comparison, do you have evidence that the hot hand model fits Kobe's shooting patterns? Explain.

Distributions

The source for this topic is the Open Intro labs https://www.openintro.org/stat/labs.php

In this lab we'll investigate the probability distribution that is most central to statistics: the normal distribution. If we are confident that our data are nearly normal, that opens the door to many powerful statistical methods. Here we'll use the graphical tools of R to assess the normality of our data and also learn how to generate random numbers from a normal distribution.

The Data

This week we'll be working with measurements of body dimensions. This data set contains measurements from 247 men and 260 women, most of whom were considered healthy young adults.

```
load("data/bdims.RData")
```

Let's take a quick peek at the first few rows of the data.

```
head(bdims)
```

You'll see that for every observation we have 25 measurements, many of which are either diameters or girths. A key to the variable names can be found at http://www.openintro.org/stat/data/bdims.php, but we'll be focusing on just three columns to get started: weight in kg (`wgt`), height in cm (`hgt`), and `sex` (`1` indicates male, `0` indicates female).

Since males and females tend to have different body dimensions, it will be useful to create two additional data sets: one with only men and another with only women.

```
mdims <- subset(bdims, sex == 1)
fdims <- subset(bdims, sex == 0)
```

1. Make a histogram of men's heights and a histogram of women's heights. How would you compare the various aspects of the two distributions?

The normal distribution

In your description of the distributions, did you use words like *bell-shaped* or *normal*? It's tempting to say so when faced with a unimodal symmetric distribution.

To see how accurate that description is, we can plot a normal distribution curve on top of a histogram to see how closely the data follow a normal distribution. This normal curve should have the same mean and standard deviation as the data. We'll be working with women's heights, so let's store them as a separate object and then calculate some statistics that will be referenced later.

```
fhgtmean <- mean(fdims$hgt)
fhgtsd   <- sd(fdims$hgt)
```

Next we make a density histogram to use as the backdrop and use the `lines` function to overlay a normal probability curve. The difference between a frequency histogram and a density histogram is that while in a frequency histogram the *heights* of the bars add up to the total number of observations, in a density histogram the *areas* of the bars add up to 1. The area of each bar can be calculated as simply the height *times* the width of the bar. Using a density histogram allows us to properly overlay a normal distribution curve over the histogram since the curve is a normal probability density function. Frequency and density histograms both display the same exact shape; they only differ in their y-axis. You can verify this by comparing the frequency histogram you constructed earlier and the density histogram created by the commands below.

```
hist(fdims$hgt, probability = TRUE)
x <- 140:190
y <- dnorm(x = x, mean = fhgtmean, sd = fhgtsd)
lines(x = x, y = y, col = "blue")
```

After plotting the density histogram with the first command, we create the x- and y-coordinates for the normal curve. We chose the `x` range as 140 to 190 in order to span the entire range of `fheight`. To create `y`, we use `dnorm` to calculate the density of each of those x-values in a distribution that is normal with mean `fhgtmean` and standard deviation `fhgtsd`. The final command draws a curve on the existing plot (the density histogram) by connecting each of the points specified by `x` and `y`. The argument `col` simply sets the color for the line to be drawn. If we left it out, the line would be drawn in black.

The top of the curve is cut off because the limits of the x- and y-axes are set to best fit the histogram. To adjust the y-axis you can add a third argument to the histogram function: `ylim = c(0, 0.06)`.

2. Based on the this plot, does it appear that the data follow a nearly normal distribution?

Evaluating the normal distribution

Eyeballing the shape of the histogram is one way to determine if the data appear to be nearly normally distributed, but it can be frustrating to decide just how close the histogram is to the curve. An alternative approach involves constructing a normal probability plot, also called a normal Q-Q plot for "quantile-quantile".

```
qqnorm(fdims$hgt)
qqline(fdims$hgt)
```

A data set that is nearly normal will result in a probability plot where the points closely follow the line. Any deviations from normality leads to deviations of these points from the line. The plot for female heights shows points that tend to follow the line but with some errant points towards the tails. We're left with the same problem that we encountered with the histogram above: how close is close enough?

A useful way to address this question is to rephrase it as: what do probability plots look like for data that I *know* came from a normal distribution? We can answer this by simulating data from a normal distribution using `rnorm`.

```
sim_norm <- rnorm(n = length(fdims$hgt), mean = fhgtmean, sd = fhgtsd)
```

The first argument indicates how many numbers you'd like to generate, which we specify to be the same number of heights in the `fdims` data set using the `length` function. The last two arguments determine the mean and standard deviation of the normal distribution from which the simulated sample will be generated. We can take a look at the shape of our simulated data set, `sim_norm`, as well as its normal probability plot.

3. Make a normal probability plot of `sim_norm`. Do all of the points fall on the line? How does this plot compare to the probability plot for the real data?

Even better than comparing the original plot to a single plot generated from a normal distribution is to compare it to many more plots using the following function. It may be helpful to click the zoom button in the plot window.

```
qqnormsim(fdims$hgt)
```

4. Does the normal probability plot for `fdims$hgt` look similar to the plots created for the simulated data? That is, do plots provide evidence that the female heights are nearly normal?

5. Using the same technique, determine whether or not female weights appear to come from a normal distribution.

Normal probabilities

Okay, so now you have a slew of tools to judge whether or not a variable is normally distributed. Why should we care?

It turns out that statisticians know a lot about the normal distribution. Once we decide that a random variable is approximately normal, we can answer all sorts of questions about that variable related to probability. Take, for example, the question of, "What is the probability that a randomly chosen young adult female is taller than 6 feet (about 182 cm)?" (The study that published this data set is clear to point out that the sample was not random and therefore inference to a general population is not suggested. We do so here only as an exercise.)

If we assume that female heights are normally distributed (a very close approximation is also okay), we can find this probability by calculating a Z score and consulting a Z table (also called a normal probability table). In R, this is done in one step with the function `pnorm`.

```
1 - pnorm(q = 182, mean = fhgtmean, sd = fhgtsd)
```

Note that the function `pnorm` gives the area under the normal curve below a given value, `q`, with a given mean and standard deviation. Since we're interested in the probability that someone is taller than 182 cm, we have to take one minus that probability.

Assuming a normal distribution has allowed us to calculate a theoretical probability. If we want to calculate the probability empirically, we simply need to determine how many observations fall above 182 then divide this number by the total sample size.

```
sum(fdims$hgt > 182) / length(fdims$hgt)
```

Although the probabilities are not exactly the same, they are reasonably close. The closer that your distribution is to being normal, the more accurate the theoretical probabilities will be.

6. Write out two probability questions that you would like to answer; one regarding female heights and one regarding female weights. Calculate the those probabilities using both the theoretical normal distribution as well as the empirical distribution (four probabilities in all). Which variable, height or weight, had a closer agreement between the two methods?

On Your Own

- Now let's consider some of the other variables in the body dimensions data set. Using the figures at the end of the exercises, match the histogram to its normal probability plot. All of the variables have been standardized (first subtract the mean, then divide by the standard deviation), so the units won't be of any help. If you are uncertain based on these figures, generate the plots in R to check.

a. The histogram for female biiliac (pelvic) diameter (`bii.di`) belongs to normal probability plot letter _____.

b. The histogram for female elbow diameter (`elb.di`) belongs to normal probability plot letter _____.

c. The histogram for general age (`age`) belongs to normal probability plot letter _____.

d. The histogram for female chest depth (`che.de`) belongs to normal probability plot letter _____.

- Note that normal probability plots C and D have a slight stepwise pattern. Why do you think this is the case?
- As you can see, normal probability plots can be used both to assess normality and visualize skewness. Make a normal probability plot for female knee diameter (`kne.di`). Based on this normal probability plot, is this variable left skewed, symmetric, or right skewed? Use a histogram to confirm your findings.

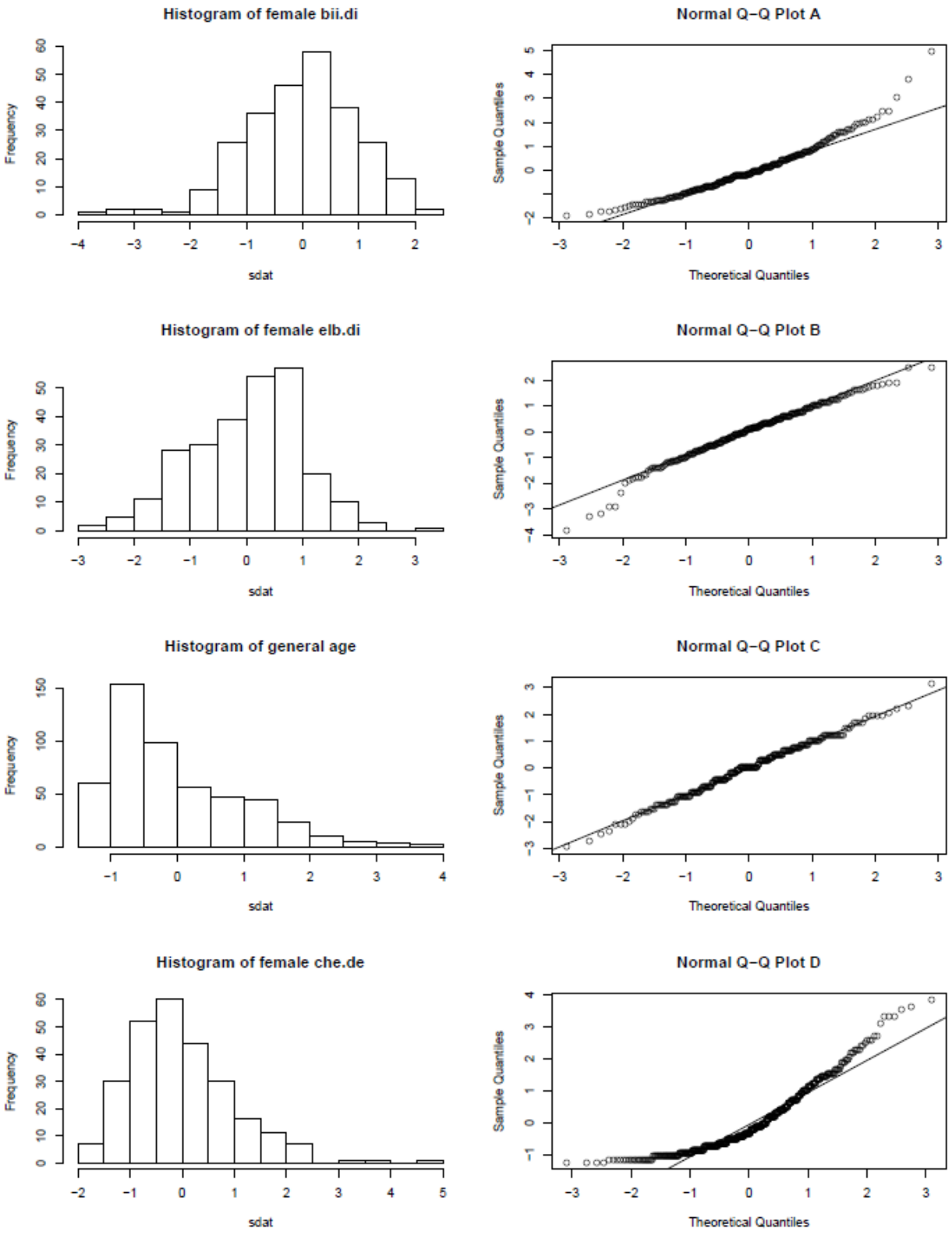


Figure 12.1: histQQmatch

How to Judge Visualizations

Read Junk Charts Trifecta Checkup

Kaiser Fung has a useful framework for judging the quality of data visualizations. He calls his framework the Junk Charts Trifecta Checkup. To make that shorter, we'll just call this framework the *Trifecta Checkup*. This framework boils down to the following 3 questions:

1. What is the **question**?
2. What does the **data** say?
3. What does the **visual** say?

Each visualization can be labeled according to which question(s) it does not answer well. For example, a chart with a meaningful question and relevant data, but ineffective visuals, would be labeled **v**. A chart that gets the data and question wrong, but has a nice visual, would be labeled **qd**.

Intro to Inference

The source for this topic is the Open Intro labs https://www.openintro.org/stat/labs.php

In this lab, we investigate the ways in which the statistics from a random sample of data can serve as point estimates for population parameters. We're interested in formulating a *sampling distribution* of our estimate in order to learn about the properties of the estimate, such as its distribution.

The data

We consider real estate data from the city of Ames, Iowa. The details of every real estate transaction in Ames is recorded by the City Assessor's office. Our particular focus for this lab will be all residential home sales in Ames between 2006 and 2010. This collection represents our population of interest. In this lab we would like to learn about these home sales by taking smaller samples from the full population. Let's load the data.

```
load("data/ames.RData")
```

We see that there are quite a few variables in the data set, enough to do a very in-depth analysis. For this lab, we'll restrict our attention to just two of the variables: the above ground living area of the house in square feet (`Gr.Liv.Area`) and the sale price (`SalePrice`). To save some effort throughout the lab, create two variables with short names that represent these two variables.

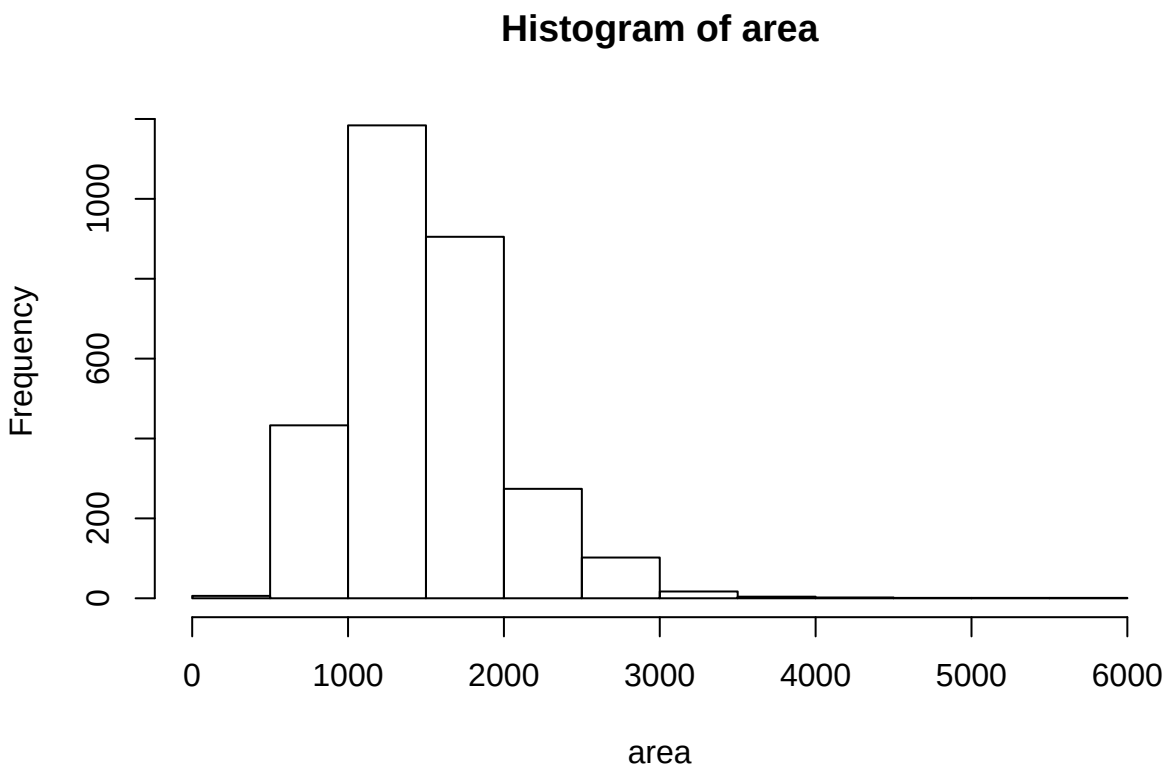
```
area <- ames$Gr.Liv.Area
price <- ames$SalePrice
```

Let's look at the distribution of area in our population of home sales by calculating a few summary statistics and making a histogram.

```
summary(area)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     334    1126    1442    1500    1743    5642
```

```
hist(area)
```



1. Describe this population distribution.

The unknown sampling distribution

In this lab we have access to the entire population, but this is rarely the case in real life. Gathering information on an entire population is often extremely costly or impossible. Because of this, we often take a sample of the population and use that to understand the properties of the population.

If we were interested in estimating the mean living area in Ames based on a sample, we can use the following command to survey the population.

```
samp1 <- sample(area, 50)
```

This command collects a simple random sample of size 50 from the vector `area`, which is assigned to `samp1`. This is like going into the City Assessor's database and pulling up the files on 50 random home sales. Working with these 50 files would be considerably simpler than working with all 2930 home sales.

2. Describe the distribution of this sample. How does it compare to the distribution of the population?

If we're interested in estimating the average living area in homes in Ames using the sample, our best single guess is the sample mean.

```
mean(samp1)
```

```
## [1] 1457.86
```

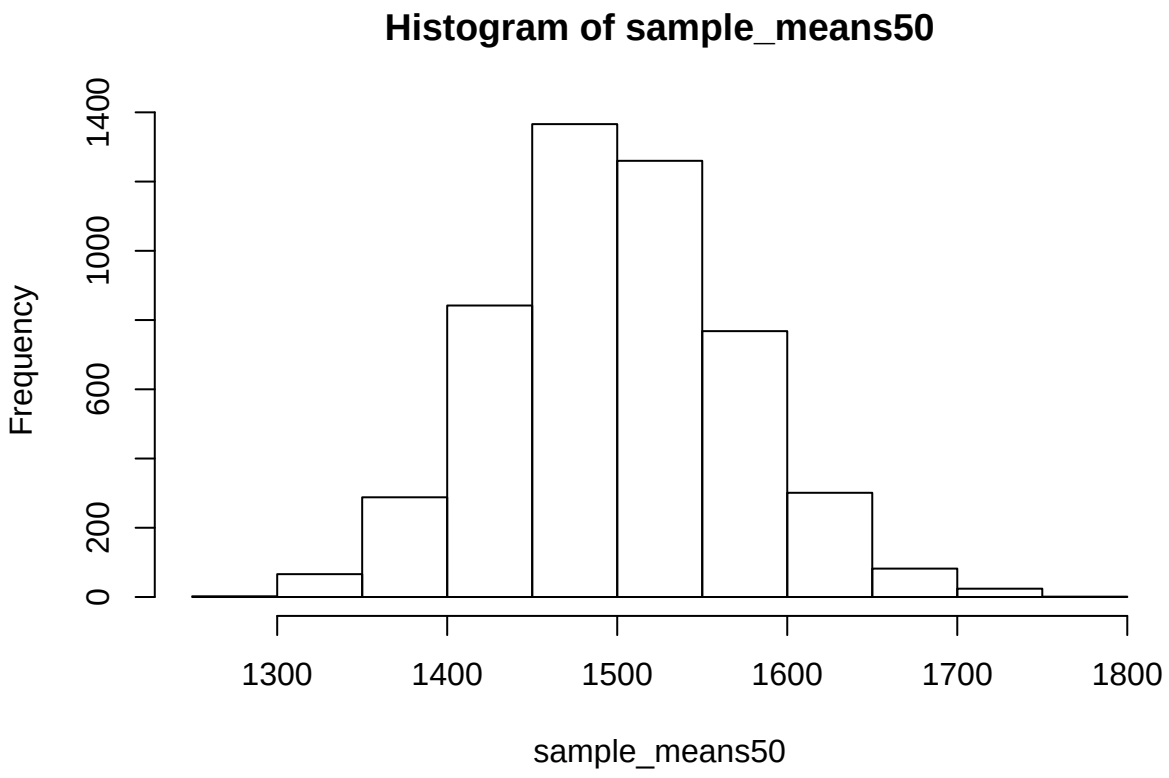
Depending on which 50 homes you selected, your estimate could be a bit above or a bit below the true population mean of 1499.69 square feet. In general, though, the sample mean turns out to be a pretty good estimate of the average living area, and we were able to get it by sampling less than 3% of the population.

3. Take a second sample, also of size 50, and call it `samp2`. How does the mean of `samp2` compare with the mean of `samp1`? Suppose we took two more samples, one of size 100 and one of size 1000. Which

would you think would provide a more accurate estimate of the population mean?

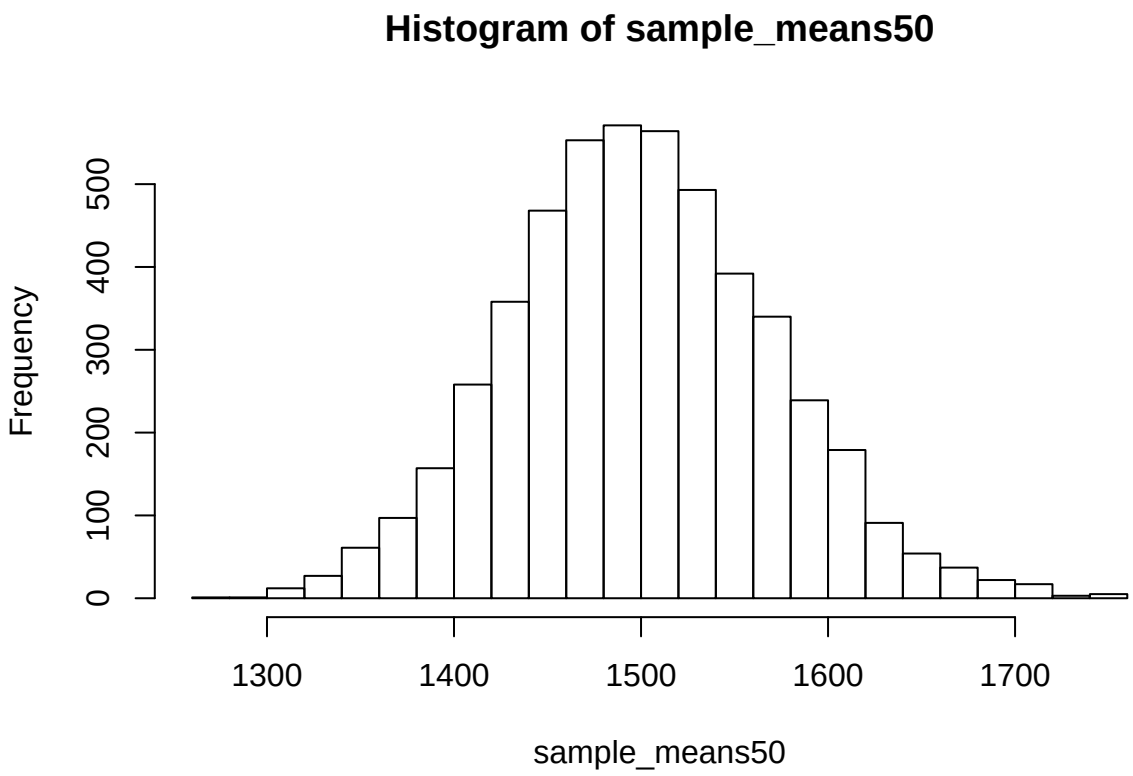
Not surprisingly, every time we take another random sample, we get a different sample mean. It's useful to get a sense of just how much variability we should expect when estimating the population mean this way. The distribution of sample means, called the *sampling distribution*, can help us understand this variability. In this lab, because we have access to the population, we can build up the sampling distribution for the sample mean by repeating the above steps many times. Here we will generate 5000 samples and compute the sample mean of each.

```
sample_means50 <- rep(NA, 5000)

for(i in 1:5000){
   samp <- sample(area, 50)
   sample_means50[i] <- mean(samp)
   }

hist(sample_means50)
```



If you would like to adjust the bin width of your histogram to show a little more detail, you can do so by changing the `breaks` argument.

```
hist(sample_means50, breaks = 25)
```



Here we use R to take 5000 samples of size 50 from the population, calculate the mean of each sample, and store each result in a vector called `sample_means50`. On the next page, we'll review how this set of code works.

4. How many elements are there in `sample_means50`? Describe the sampling distribution, and be sure to specifically note its center. Would you expect the distribution to change if we instead collected 50,000 sample means?

Interlude: The `for` loop

Let's take a break from the statistics for a moment to let that last block of code sink in. You have just run your first `for` loop, a cornerstone of computer programming. The idea behind the for loop is *iteration*: it allows you to execute code as many times as you want without having to type out every iteration. In the case above, we wanted to iterate the two lines of code inside the curly braces that take a random sample of size 50 from `area` then save the mean of that sample into the `sample_means50` vector. Without the `for` loop, this would be painful:

```
sample_means50 <- rep(NA, 5000)

samp <- sample(area, 50)
sample_means50[1] <- mean(samp)

samp <- sample(area, 50)
sample_means50[2] <- mean(samp)

samp <- sample(area, 50)
sample_means50[3] <- mean(samp)

samp <- sample(area, 50)
```

```
sample_means50[4] <- mean(samp)
```

and so on…

With the for loop, these thousands of lines of code are compressed into a handful of lines. We've added one extra line to the code below, which prints the variable `i` during each iteration of the `for` loop. Run this code.

```
sample_means50 <- rep(NA, 5000)

for(i in 1:5000) {
   samp <- sample(area, 50)
   sample_means50[i] <- mean(samp)
   #print(i)
}
```

Let's consider this code line by line to figure out what it does. In the first line we *initialized a vector*. In this case, we created a vector of 5000 zeros called `sample_means50`. This vector will will store values generated within the `for` loop.

The second line calls the `for` loop itself. The syntax can be loosely read as, "for every element `i` from 1 to 5000, run the following lines of code". You can think of `i` as the counter that keeps track of which loop you're on. Therefore, more precisely, the loop will run once when `i = 1`, then once when `i = 2`, and so on up to `i = 5000`.

The body of the `for` loop is the part inside the curly braces, and this set of code is run for each value of `i`. Here, on every loop, we take a random sample of size 50 from `area`, take its mean, and store it as the ith element of `sample_means50`.

In order to display that this is really happening, we asked R to print `i` at each iteration. This line of code is optional and is only used for displaying what's going on while the `for` loop is running.

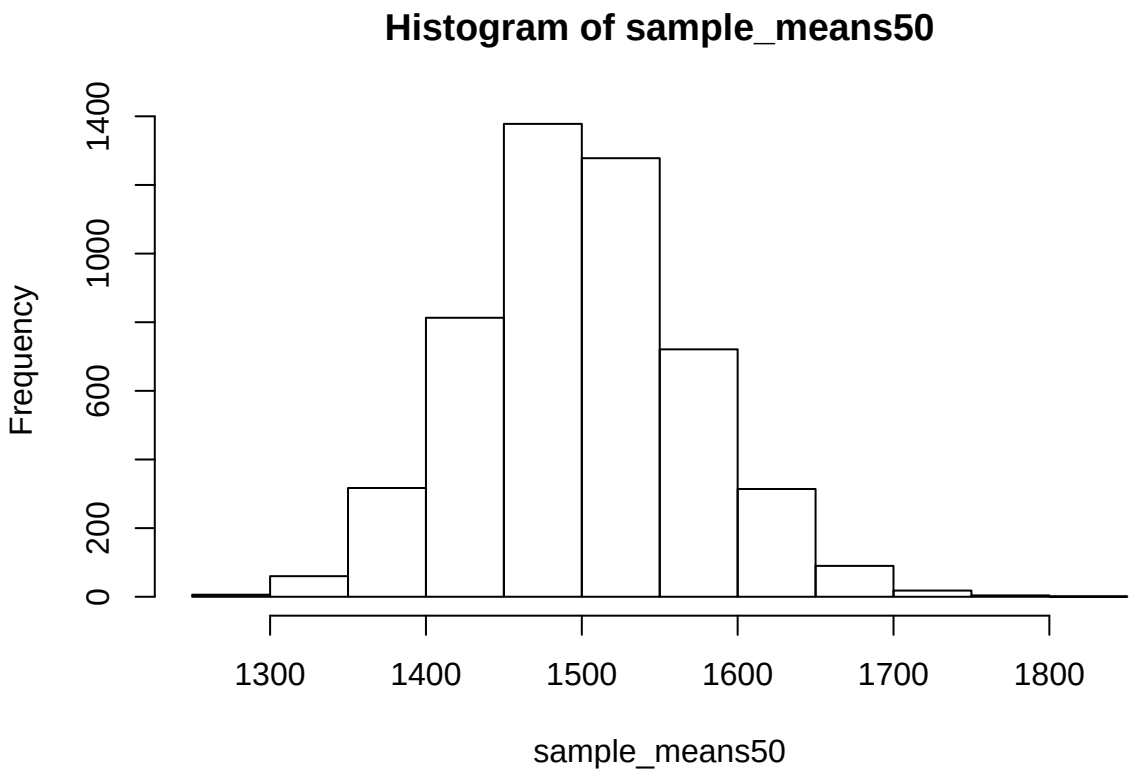
The `for` loop allows us to not just run the code 5000 times, but to neatly package the results, element by element, into the empty vector that we initialized at the outset.

5. To make sure you understand what you've done in this loop, try running a smaller version. Initialize a vector of 100 zeros called `sample_means_small`. Run a loop that takes a sample of size 50 from `area` and stores the sample mean in `sample_means_small`, but only iterate from 1 to 100. Print the output to your screen (type `sample_means_small` into the console and press enter). How many elements are there in this object called `sample_means_small`? What does each element represent?

Sample size and the sampling distribution

Mechanics aside, let's return to the reason we used a `for` loop: to compute a sampling distribution, specifically, this one.

```
hist(sample_means50)
```



The sampling distribution that we computed tells us much about estimating the average living area in homes in Ames. Because the sample mean is an unbiased estimator, the sampling distribution is centered at the true average living area of the the population, and the spread of the distribution indicates how much variability is induced by sampling only 50 home sales.

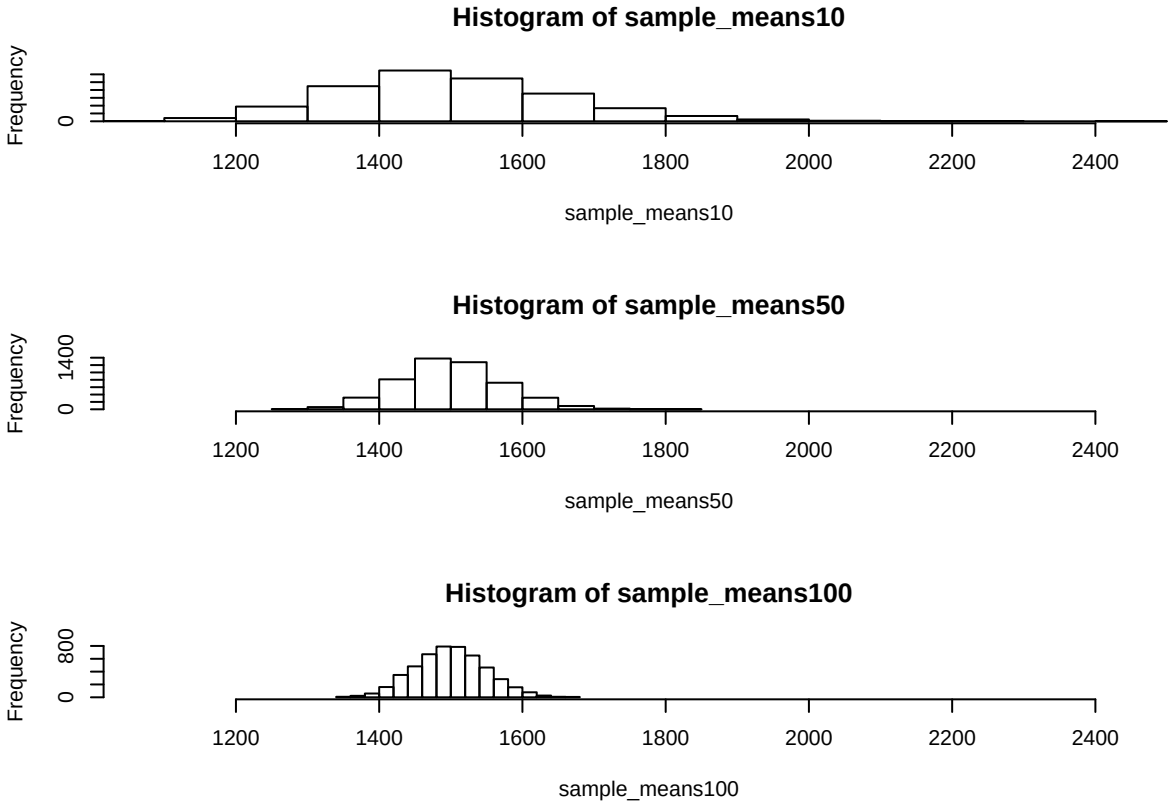
To get a sense of the effect that sample size has on our distribution, let's build up two more sampling distributions: one based on a sample size of 10 and another based on a sample size of 100.

```
sample_means10 <- rep(NA, 5000)
sample_means100 <- rep(NA, 5000)

for(i in 1:5000){
  samp <- sample(area, 10)
  sample_means10[i] <- mean(samp)
  samp <- sample(area, 100)
  sample_means100[i] <- mean(samp)
}
```

Here we're able to use a single `for` loop to build two distributions by adding additional lines inside the curly braces. Don't worry about the fact that `samp` is used for the name of two different objects. In the second command of the `for` loop, the mean of `samp` is saved to the relevant place in the vector `sample_means10`. With the mean saved, we're now free to overwrite the object `samp` with a new sample, this time of size 100. In general, anytime you create an object using a name that is already in use, the old object will get replaced with the new one.

To see the effect that different sample sizes have on the sampling distribution, plot the three distributions on top of one another.

```
par(mfrow = c(3, 1))

xlimits <- range(sample_means10)

hist(sample_means10, breaks = 20, xlim = xlimits)
hist(sample_means50, breaks = 20, xlim = xlimits)
hist(sample_means100, breaks = 20, xlim = xlimits)
```



The first command specifies that you'd like to divide the plotting area into 3 rows and 1 column of plots (to return to the default setting of plotting one at a time, use `par(mfrow = c(1, 1))`). The `breaks` argument specifies the number of bins used in constructing the histogram. The `xlim` argument specifies the range of the x-axis of the histogram, and by setting it equal to `xlimits` for each histogram, we ensure that all three histograms will be plotted with the same limits on the x-axis.

6. When the sample size is larger, what happens to the center? What about the spread?

On your own

So far, we have only focused on estimating the mean living area in homes in Ames. Now you'll try to estimate the mean home price.

- Take a random sample of size 50 from `price`. Using this sample, what is your best point estimate of the population mean?
- Since you have access to the population, simulate the sampling distribution for $\bar{x}_{price}$ by taking 5000 samples from the population of size 50 and computing 5000 sample means. Store these means in a

vector called `sample_means50`. Plot the data, then describe the shape of this sampling distribution. Based on this sampling distribution, what would you guess the mean home price of the population to be? Finally, calculate and report the population mean.

- Change your sample size from 50 to 150, then compute the sampling distribution using the same method as above, and store these means in a new vector called `sample_means150`. Describe the shape of this sampling distribution, and compare it to the sampling distribution for a sample size of 50. Based on this sampling distribution, what would you guess to be the mean sale price of homes in Ames?
- Of the sampling distributions from 2 and 3, which has a smaller spread? If we're concerned with making estimates that are more often close to the true value, would we prefer a distribution with a large or small spread?

Confidence Intervals

The source for this topic is the Open Intro labs https://www.openintro.org/stat/labs.php

Sampling from Ames, Iowa

If you have access to data on an entire population, say the size of every house in Ames, Iowa, it's straight forward to answer questions like, "How big is the typical house in Ames?" and "How much variation is there in sizes of houses?". If you have access to only a sample of the population, as is often the case, the task becomes more complicated. What is your best guess for the typical size if you only know the sizes of several dozen houses? This sort of situation requires that you use your sample to make inference on what your population looks like.

The data

In the previous lab, "Sampling Distributions", we looked at the population data of houses from Ames, Iowa. Let's start by loading that data set.

```
load("data/ames.RData")
```

In this lab we'll start with a simple random sample of size 60 from the population. Specifically, this is a simple random sample of size 60. Note that the data set has information on many housing variables, but for the first portion of the lab we'll focus on the size of the house, represented by the variable `Gr.Liv.Area`.

```
population <- ames$Gr.Liv.Area
head(population)
```

```
## [1] 1656  896 1329 2110 1629 1604
```

```
samp <- sample(population, 60)
samp
```

```
##  [1] 1306  492 1687 1624 1034 1176  894 1561 1192 1352 1224 1884  894 1229
## [15] 1222 1728 2787 1603 1732 1229 1640 1287  765 1151 1652 2110 1288 1725
## [29] 2267  816 1403 1273  914 1652 1025 3627 1176 1866 1484  987 2452 1298
## [43] 1802 1322 1400 1911 1347 2514 1159 1236 1594 1025 1698  912  864 1987
## [57] 1690 1671  958 1950
```

1. Describe the distribution of your sample. What would you say is the "typical" size within your sample? Also state precisely what you interpreted "typical" to mean.

2. Would you expect another student's distribution to be identical to yours? Would you expect it to be similar? Why or why not?

Confidence intervals

One of the most common ways to describe the typical or central value of a distribution is to use the mean. In this case we can calculate the mean of the sample using,

```
sample_mean <- mean(samp)
sample_mean
```

```
## [1] 1479.133
```

Return for a moment to the question that first motivated this lab: based on this sample, what can we infer about the population? Based only on this single sample, the best estimate of the average living area of houses sold in Ames would be the sample mean, usually denoted as $\bar{x}$ (here we're calling it `sample_mean`). That serves as a good *point estimate* but it would be useful to also communicate how uncertain we are of that estimate. This can be captured by using a *confidence interval*.

We can calculate a 95% confidence interval for a sample mean by adding and subtracting 1.96 standard errors to the point estimate (See Section 4.2.3 if you are unfamiliar with this formula).

```
se <- sd(samp) / sqrt(60)
se
```

```
## [1] 68.76275
```

```
lower <- sample_mean - 1.96 * se
upper <- sample_mean + 1.96 * se
c(lower, upper)
```

```
## [1] 1344.358 1613.908
```

This is an important inference that we've just made: even though we don't know what the full population looks like, we're 95% confident that the true average size of houses in Ames lies between the values *lower* and *upper*. There are a few conditions that must be met for this interval to be valid.

3. For the confidence interval to be valid, the sample mean must be normally distributed and have standard error $s/\sqrt{n}$. What conditions must be met for this to be true?

Confidence levels

4. What does "95% confidence" mean? If you're not sure, see Section 4.2.2.

In this case we have the luxury of knowing the true population mean since we have data on the entire population. This value can be calculated using the following command:

```
mean(population)
```

```
## [1] 1499.69
```

5. Does your confidence interval capture the true average size of houses in Ames? If you are working on this lab in a classroom, does your neighbor's interval capture this value?

6. Each student in your class should have gotten a slightly different confidence interval. What proportion of those intervals would you expect to capture the true population mean? Why? If you are working in this lab in a classroom, collect data on the intervals created by other students in the class and calculate the proportion of intervals that capture the true population mean.

Using R, we're going to recreate many samples to learn more about how sample means and confidence intervals vary from one sample to another. *Loops* come in handy here (If you are unfamiliar with loops, review the Sampling Distribution Lab).

Here is the rough outline:

- Obtain a random sample.
- Calculate and store the sample's mean and standard deviation.
- Repeat steps (1) and (2) 50 times.
- Use these stored statistics to calculate many confidence intervals.

But before we do all of this, we need to first create empty vectors where we can save the means and standard deviations that will be calculated from each sample. And while we're at it, let's also store the desired sample size as `n`.

```
samp_mean <- rep(NA, 50)
samp_sd <- rep(NA, 50)
n <- 60
```

Now we're ready for the loop where we calculate the means and standard deviations of 50 random samples.

```
for(i in 1:50){
  samp <- sample(population, n) # obtain a sample of size n = 60 from the population
  samp_mean[i] <- mean(samp)    # save sample mean in ith element of samp_mean
  samp_sd[i] <- sd(samp)        # save sample sd in ith element of samp_sd
}
```

Lastly, we construct the confidence intervals.

```
lower_vector <- samp_mean - 1.96 * samp_sd / sqrt(n)
upper_vector <- samp_mean + 1.96 * samp_sd / sqrt(n)
```

Lower bounds of these 50 confidence intervals are stored in `lower_vector`, and the upper bounds are in `upper_vector`. Let's view the first interval.

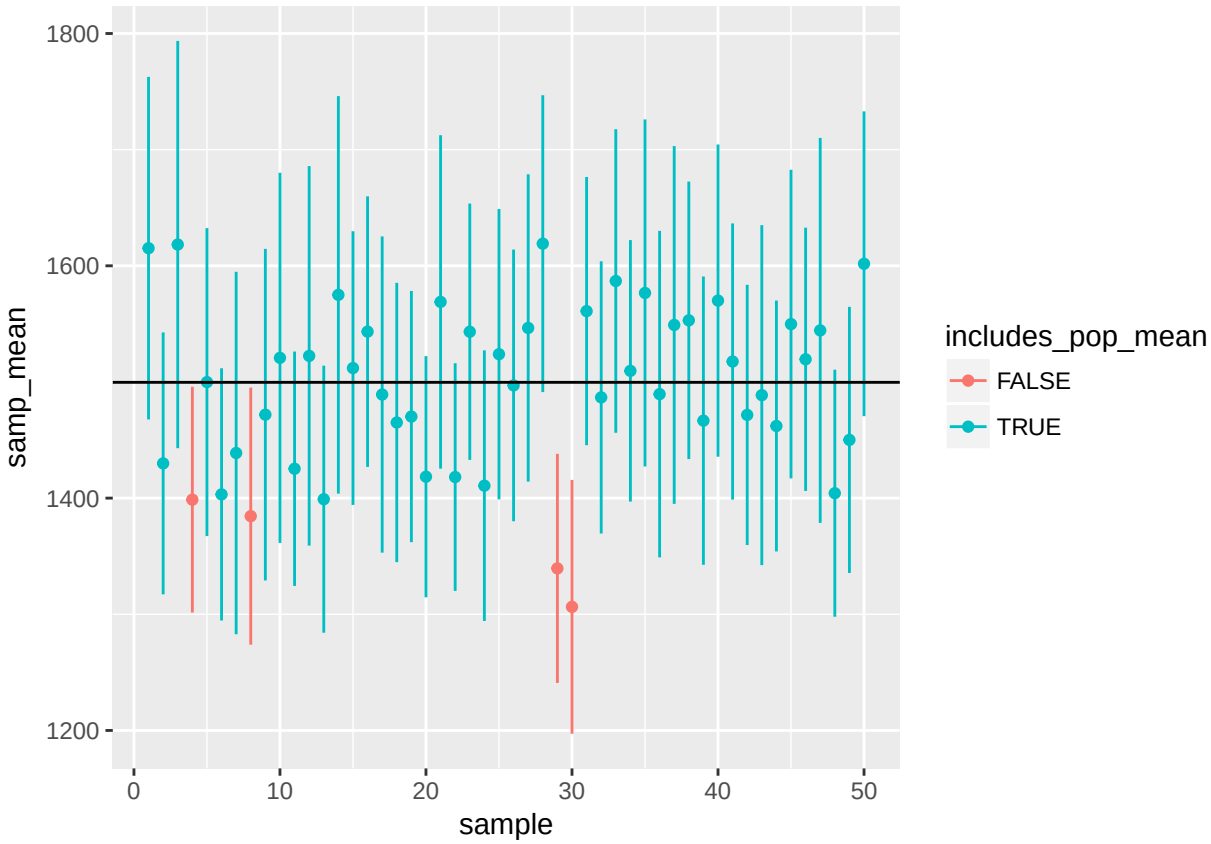
```
c(lower_vector[1], upper_vector[1])
```

```
## [1] 1467.735 1762.631
```

On your own

Let's visualize these confidence intervals.

```
library(ggplot2)
df <- data.frame(sample = 1:50, samp_mean = samp_mean, lower = lower_vector, upper = upper_vector,
                 includes_pop_mean = mean(population) < upper_vector & mean(population) > lower_vector)
ggplot(df, aes(sample, samp_mean, color = includes_pop_mean)) +
  geom_segment(aes(x = sample, y = lower, xend = sample, yend = upper)) +
  geom_point() +
  geom_hline(yintercept = mean(population))
```



- Pick a confidence level of your choosing, provided it is not 95%. What is the appropriate critical value?
- Calculate 50 confidence intervals at the confidence level you chose in the previous question. You do not need to obtain new samples, simply calculate new intervals based on the sample means and standard deviations you have already collected. Using the `ggplot()` code above, plot all intervals and calculate the proportion of intervals that include the true population mean. How does this percentage compare to the confidence level selected for the intervals?

Inference for Numerical Data

The source for this topic is the Open Intro labs https://www.openintro.org/stat/labs.php

North Carolina births

In 2004, the state of North Carolina released a large data set containing information on births recorded in this state. This data set is useful to researchers studying the relation between habits and practices of expectant mothers and the birth of their children. We will work with a random sample of observations from this data set.

Exploratory analysis

Load the `nc` data set into our workspace.

```
load("data/nc.RData")
```

We have observations on 13 different variables, some categorical and some numerical. The meaning of each variable is as follows.

variable	description
`fage`	father's age in years.
`mage`	mother's age in years.
`mature`	maturity status of mother.
`weeks`	length of pregnancy in weeks.
`premie`	whether the birth was classified as premature (premie) or full-term.
`visits`	number of hospital visits during pregnancy.

variable	description
`marital`	whether mother is `married` or `not married` at birth.
`gained`	weight gained by mother during pregnancy in pounds.
`weight`	weight of the baby at birth in pounds.
`lowbirthweight`	whether baby was classified as low birthweight (`low`) or not (`not low`).
`gender`	gender of the baby, `female` or `male`.
`habit`	status of the mother as a `nonsmoker` or a `smoker`.
`whitemom`	whether mom is `white` or `not white`.

1. What are the cases in this data set? How many cases are there in our sample?

As a first step in the analysis, we should consider summaries of the data. This can be done using the `summary` command:

```
summary(nc)
```

As you review the variable summaries, consider which variables are categorical and which are numerical. For numerical variables, are there outliers? If you aren't sure or want to take a closer look at the data, make a graph.

Consider the possible relationship between a mother's smoking habit and the weight of her baby. Plotting the data is a useful first step because it helps us quickly visualize trends, identify strong associations, and develop research questions.

2. Make a side-by-side boxplot of `habit` and `weight`. What does the plot highlight about the relationship between these two variables?

```
library(tidyverse)
ggplot(nc, aes(habit, weight)) + geom_boxplot()
```

The box plots show how the medians of the two distributions compare, but we can also compare the means of the distributions using the following function to split the `weight` variable into the `habit` groups, then take the mean of each using the `mean` function.

```
by(nc$weight, nc$habit, mean)
```

We can do the same thing with `group_by` and `summarize` (or `summarise`).

```
habits <- nc %>%
  group_by(habit) %>%
  summarize(mean_weight = mean(weight), n = n(), sd_weight = sd(weight))
habits
```

For more examples see `?dplyr::summarise`.

There is an observed difference, but is this difference statistically significant? In order to answer this question we will conduct a hypothesis test.

Inference

3. Check if the conditions necessary for inference are satisfied. Note that you will need to obtain sample sizes to check the conditions. You can compute the group size using the same `by` command above but replacing `mean` with `length`.
4. Write the hypotheses for testing if the average weights of babies born to smoking and non-smoking mothers are different.

$$H_0 : \mu_{nonsmoker} = \mu_{smoker}$$

$$H_A : \mu_{nonsmoker} \neq \mu_{smoker}$$

Test statistic:

$$z = \frac{\bar{x_1} - \bar{x_2}}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}}$$

```
std_err <- sqrt(
    habits[1,]$sd_weight^2 / habits[1,]$n +
    habits[2,]$sd_weight^2 / habits[2,]$n
  )
std_err
difference <- habits[1,]$mean_weight - habits[2,]$mean_weight
difference
z <- difference / std_err
z
```

Two-tailed test: Here the `p_value` is the probability of drawing a difference as large as the one we had.

```
p_value <- 2 * (1 - pnorm(z))
p_value
```

5. Now let's construct a confidence interval for the difference between the weights of babies born to smoking and non-smoking mothers.

```
a <- 0.05
difference + (qnorm(a/2) * std_err)
difference + (qnorm(1 - a/2) * std_err)
```

stats package

The stats library is loaded by default in R and includes functions for performing the above calculations.

```
t.test(nc[nc$habit == "smoker",]$weight, nc[nc$habit == "nonsmoker",]$weight)
```

On your own

- Calculate a 95% confidence interval for the average length of pregnancies (weeks) and interpret it in context. Note that since you're doing inference on a single population parameter, there is no explanatory variable, so you can omit the x variable from the function.
- Calculate a new confidence interval for the same parameter at the 90% confidence level. You can change the confidence level by adding a new argument to the function: conflevel = 0.90.
- Conduct a hypothesis test evaluating whether the average weight gained by younger mothers is different than the average weight gained by mature mothers.
- Now, a non-inference task: Determine the age cutoff for younger and mature mothers. Use a method of your choice, and explain how your method works.
- Pick a pair of numerical and categorical variables and come up with a research question evaluating the relationship between these variables. Formulate the question in a way that it can be answered using a hypothesis test and/or a confidence interval. Answer your question using the inference function, report the statistical results, and also provide an explanation in plain language.

Inference for Categorical Data

The source for this topic is the Open Intro labs https://www.openintro.org/stat/labs.php

In August of 2012, news outlets ranging from the Washington Post to the Huffington Post ran a story about the rise of atheism in America. The source for the story was a poll that asked people, "Irrespective of whether you attend a place of worship or not, would you say you are a religious person, not a religious person or a convinced atheist?" This type of question, which asks people to classify themselves in one way or another, is common in polling and generates categorical data. In this lab we take a look at the atheism survey and explore what's at play when making inference about population proportions using categorical data.

The survey

To access the press release for the poll, conducted by WIN-Gallup International, click on the following link:

http://www.wingia.com/web/files/richeditor/filemanager/Global_INDEX_of_Religiosity_and_Atheism_PR___6.pdf

Take a moment to review the report then address the following questions.

1. In the first paragraph, several key findings are reported. Do these percentages appear to be *sample statistics* (derived from the data sample) or *population parameters*?

2. The title of the report is "Global Index of Religiosity and Atheism". To generalize the report's findings to the global human population, what must we assume about the sampling method? Does that seem like a reasonable assumption?

The data

Turn your attention to Table 6 (pages 15 and 16), which reports the sample size and response percentages for all 57 countries. While this is a useful format to summarize the data, we will base our analysis on the original data set of individual responses to the survey. Load this data set into R with the following command.

```
load("data/atheism.RData")
```

3. What does each row of Table 6 correspond to? What does each row of `atheism` correspond to?

To investigate the link between these two ways of organizing this data, take a look at the estimated proportion of atheists in the United States. Towards the bottom of Table 6, we see that this is 5%. We should be able to come to the same number using the `atheism` data.

4. Using the command below, create a new dataframe called `us12` that contains only the rows in `atheism` associated with respondents to the 2012 survey from the United States. Next, calculate the proportion of atheist responses. Does it agree with the percentage in Table 6? If not, why?

```
us12 <- subset(atheism, nationality == "United States" & year == "2012")
```

Inference on proportions

As was hinted at in Exercise 1, Table 6 provides *statistics*, that is, calculations made from the sample of 51,927 people. What we'd like, though, is insight into the population *parameters*. You answer the question, "What proportion of people in your sample reported being atheists?" with a statistic; while the question "What proportion of people on earth would report being atheists" is answered with an estimate of the parameter.

The inferential tools for estimating population proportion are analogous to those used for means in the last chapter: the confidence interval and the hypothesis test.

5. Write out the conditions for inference to construct a 95% confidence interval for the proportion of atheists in the United States in 2012. Are you confident all conditions are met?

If the conditions for inference are reasonable, we can either calculate the standard error and construct the interval by hand, or allow the `prop.test` function to do it for us.

```
table(us12$response)
prop.test(table(us12$response), alternative = "two.sided")
```

We can plot the counts of each `response`.

```
library(ggplot2)
ggplot(us12, aes(response)) + geom_histogram(stat = "count")
```

Note that since the goal is to construct an interval estimate for a proportion, it's necessary to specify what constitutes a "success", which here is a response of `"atheist"`.

Although formal confidence intervals and hypothesis tests don't show up in the report, suggestions of inference appear at the bottom of page 7: "In general, the error margin for surveys of this kind is ± 3-5% at 95% confidence".

6. Based on the R output, what is the margin of error for the estimate of the proportion of the proportion of atheists in US in 2012?

7. Using the `prop.test` function, calculate confidence intervals for the proportion of atheists in 2012 in two other countries of your choice, and report the associated margins of error. Be sure to note whether the conditions for inference are met. It may be helpful to create new data sets for each of the two countries first, and then use these data sets in the `prop.test` function to construct the confidence intervals.

How does the proportion affect the margin of error?

Imagine you've set out to survey 1000 people on two questions: are you female? and are you left-handed? Since both of these sample proportions were calculated from the same sample size, they should have the same margin of error, right? Wrong! While the margin of error does change with sample size, it is also affected by the proportion.

Think back to the formula for the standard error: $SE = \sqrt{p(1-p)/n}$. This is then used in the formula for the margin of error for a 95% confidence interval: $ME = 1.96 \times SE = 1.96 \times \sqrt{p(1-p)/n}$. Since the population proportion p is in this ME formula, it should make sense that the margin of error is in some way dependent on the population proportion. We can visualize this relationship by creating a plot of ME vs. p.

The first step is to make a vector `p` that is a sequence from 0 to 1 with each number separated by 0.01. We can then create a vector of the margin of error (`me`) associated with each of these values of `p` using the familiar approximate formula ($ME = 2 \times SE$). Lastly, we plot the two vectors against each other to reveal their relationship.

```
n <- 1000
p <- seq(0, 1, 0.01)
me <- 2 * sqrt(p * (1 - p)/n)
plot(me ~ p, ylab = "Margin of Error", xlab = "Population Proportion")
```

8. Describe the relationship between `p` and `me`.

Success-failure condition

The textbook emphasizes that you must always check conditions before making inference. For inference on proportions, the sample proportion can be assumed to be nearly normal if it is based upon a random sample of independent observations and if both $np \geq 10$ and $n(1-p) \geq 10$. This rule of thumb is easy enough to follow, but it makes one wonder: what's so special about the number 10?

The short answer is: nothing. You could argue that we would be fine with 9 or that we really should be using 11. What is the "best" value for such a rule of thumb is, at least to some degree, arbitrary. However, when np and $n(1-p)$ reaches 10 the sampling distribution is sufficiently normal to use confidence intervals and hypothesis tests that are based on that approximation.

We can investigate the interplay between n and p and the shape of the sampling distribution by using simulations. To start off, we simulate the process of drawing 5000 samples of size 1040 from a population with a true atheist proportion of 0.1. For each of the 5000 samples we compute $\hat{p}$ and then plot a histogram to visualize their distribution.

```
p <- 0.1
n <- 1040
p_hats <- rep(0, 5000)

for(i in 1:5000){
  samp <- sample(c("atheist", "non_atheist"), n, replace = TRUE, prob = c(p, 1-p))
  p_hats[i] <- sum(samp == "atheist")/n
}

hist(p_hats, main = "p = 0.1, n = 1040", xlim = c(0, 0.18))
```

These commands build up the sampling distribution of $\hat{p}$ using the familiar `for` loop. You can read the sampling procedure for the first line of code inside the `for` loop as, "take a sample of size n with replacement from the choices of atheist and non-atheist with probabilities p and $1-p$, respectively." The second line in the loop says, "calculate the proportion of atheists in this sample and record this value." The loop allows us to repeat this process 5,000 times to build a good representation of the sampling distribution.

9. Describe the sampling distribution of sample proportions at $n = 1040$ and $p = 0.1$. Be sure to note the center, spread, and shape.
   Hint: Remember that R has functions such as `mean` to calculate summary statistics.

10. Repeat the above simulation three more times but with modified sample sizes and proportions: for $n = 400$ and $p = 0.1$, $n = 1040$ and $p = 0.02$, and $n = 400$ and $p = 0.02$. Plot all four histograms together by running the `par(mfrow = c(2, 2))` command before creating the histograms. You may need to expand the plot window to accommodate the larger two-by-two plot. Describe the three new sampling distributions. Based on these limited plots, how does n appear to affect the distribution of $\hat{p}$? How does p affect the sampling distribution?

Once you're done, you can reset the layout of the plotting window by using the command `par(mfrow = c(1, 1))` command or clicking on "Clear All" above the plotting window (if using RStudio). Note that the latter will get rid of all your previous plots.

11. If you refer to Table 6, you'll find that Australia has a sample proportion of 0.1 on a sample size of 1040, and that Ecuador has a sample proportion of 0.02 on 400 subjects. Let's suppose for this exercise that these point estimates are actually the truth. Then given the shape of their respective sampling distributions, do you think it is sensible to proceed with inference and report margin of errors, as the reports does?

On your own

The question of atheism was asked by WIN-Gallup International in a similar survey that was conducted in 2005. (We assume here that sample sizes have remained the same.) Table 4 on page 13 of the report summarizes survey results from 2005 and 2012 for 39 countries.

- Answer the following two questions using the `prop.test` function. As always, write out the hypotheses for any tests you conduct and outline the status of the conditions for inference.

  a. Is there convincing evidence that Spain has seen a change in its atheism index between 2005 and 2012?
  Hint: Create a new data set for respondents from Spain. Form confidence intervals for the true proportion of athiests in both years, and determine whether they overlap.

  b. Is there convincing evidence that the United States has seen a change in its atheism index between 2005 and 2012?

- If in fact there has been no change in the atheism index in the countries listed in Table 4, in how many of those countries would you expect to detect a change (at a significance level of 0.05) simply by chance?
  Hint: Look in the textbook index under Type 1 error.

- Suppose you're hired by the local government to estimate the proportion of residents that attend a religious service on a weekly basis. According to the guidelines, the estimate must have a margin of error no greater than 1% with 95% confidence. You have no idea what to expect for p. How many people would you have to sample to ensure that you are within the guidelines?
  Hint: Refer to your plot of the relationship between p and margin of error. Do not use the data set to answer this question.

Introduction to Linear Regression

The source for this topic is the Open Intro labs https://www.openintro.org/stat/labs.php

Batter up

The movie Moneyball focuses on the "quest for the secret of success in baseball". It follows a low-budget team, the Oakland Athletics, who believed that underused statistics, such as a player's ability to get on base, betterpredict the ability to score runs than typical statistics like home runs, RBIs (runs batted in), and batting average. Obtaining players who excelled in these underused statistics turned out to be much more affordable for the team.

In this lab we'll be looking at data from all 30 Major League Baseball teams and examining the linear relationship between runs scored in a season and a number of other player statistics. Our aim will be to summarize these relationships both graphically and numerically in order to find which variable, if any, helps us best predict a team's runs scored in a season.

The data

Let's load up the data for the 2011 season.

```
load("data/mlb11.RData")
```

In addition to runs scored, there are seven traditionally used variables in the data set: at-bats, hits, home runs, batting average, strikeouts, stolen bases, and wins. There are also three newer variables: on-base percentage, slugging percentage, and on-base plus slugging. For the first portion of the analysis we'll consider the seven traditional variables. At the end of the lab, you'll work with the newer variables on your own.

1. What type of plot would you use to display the relationship between `runs` and one of the other numerical variables? Plot this relationship using the variable `at_bats` as the predictor. Does the relationship look linear? If you knew a team's `at_bats`, would you be comfortable using a linear model to predict the number of runs?

If the relationship looks linear, we can quantify the strength of the relationship with the correlation coefficient.

```
cor(mlb11$runs, mlb11$at_bats)
```

Sum of squared residuals

Think back to the way that we described the distribution of a single variable. Recall that we discussed characteristics such as center, spread, and shape. It's also useful to be able to describe the relationship of two numerical variables, such as `runs` and `at_bats` above.

2. Looking at your plot from the previous exercise, describe the relationship between these two variables. Make sure to discuss the form, direction, and strength of the relationship as well as any unusual observations.

Just as we used the mean and standard deviation to summarize a single variable, we can summarize the relationship between these two variables by finding the line that best follows their association. Use the following interactive function to select the line that you think does the best job of going through the cloud of points.

```
plot_ss(x = mlb11$at_bats, y = mlb11$runs)
```

```
library(ggplot2)
ggplot(mlb11, aes(at_bats, runs)) + geom_point() + geom_smooth(method = "lm")
```

After running this command, you'll be prompted to click two points on the plot to define a line. Once you've done that, the line you specified will be shown in black and the residuals in blue. Note that there are 30 residuals, one for each of the 30 observations. Recall that the residuals are the difference between the observed values and the values predicted by the line:

$$e_i = y_i - \hat{y}_i$$

The most common way to do linear regression is to select the line that minimizes the sum of squared residuals. To visualize the squared residuals, you can rerun the plot command and add the argument `showSquares = TRUE`.

```
plot_ss(x = mlb11$at_bats, y = mlb11$runs, showSquares = TRUE)
```

Note that the output from the `plot_ss` function provides you with the slope and intercept of your line as well as the sum of squares.

3. Using `plot_ss`, choose a line that does a good job of minimizing the sum of squares. Run the function several times. What was the smallest sum of squares that you got? How does it compare to your neighbors?

The linear model

It is rather cumbersome to try to get the correct least squares line, i.e. the line that minimizes the sum of squared residuals, through trial and error. Instead we can use the `lm` function in R to fit the linear model (a.k.a. regression line).

```
m1 <- lm(runs ~ at_bats, data = mlb11)
```

The first argument in the function `lm` is a formula that takes the form `y ~ x`. Here it can be read that we want to make a linear model of `runs` as a function of `at_bats`. The second argument specifies that R should look in the `mlb11` data frame to find the `runs` and `at_bats` variables.

The output of `lm` is an object that contains all of the information we need about the linear model that was just fit. We can access this information using the summary function.

```
summary(m1)
```

Let's consider this output piece by piece. First, the formula used to describe the model is shown at the top. After the formula you find the five-number summary of the residuals. The "Coefficients" table shown next is key; its first column displays the linear model's y-intercept and the coefficient of `at_bats`. With this table, we can write down the least squares regression line for the linear model:

$$\hat{y} = -2789.2429 + 0.6305 * atbats$$

One last piece of information we will discuss from the summary output is the Multiple R-squared, or more simply, R^2. The R^2 value represents the proportion of variability in the response variable that is explained by the explanatory variable. For this model, 37.3% of the variability in runs is explained by at-bats.

4. Fit a new model that uses `homeruns` to predict `runs`. Using the estimates from the R output, write the equation of the regression line. What does the slope tell us in the context of the relationship between success of a team and its home runs?

Prediction and prediction errors

Let's create a scatterplot with the least squares line laid on top.

```
plot(mlb11$runs ~ mlb11$at_bats)
abline(m1)
```

The function `abline` plots a line based on its slope and intercept. Here, we used a shortcut by providing the model `m1`, which contains both parameter estimates. This line can be used to predict y at any value of x. When predictions are made for values of x that are beyond the range of the observed data, it is referred to as *extrapolation* and is not usually recommended. However, predictions made within the range of the data are more reliable. They're also used to compute the residuals.

5. If a team manager saw the least squares regression line and not the actual data, how many runs would he or she predict for a team with 5,578 at-bats? Is this an overestimate or an underestimate, and by how much? In other words, what is the residual for this prediction?

Model diagnostics

To assess whether the linear model is reliable, we need to check for (1) linearity, (2) nearly normal residuals, and (3) constant variability.

Linearity: You already checked if the relationship between runs and at-bats is linear using a scatterplot. We should also verify this condition with a plot of the residuals vs. at-bats. Recall that any code following a # is intended to be a comment that helps understand the code but is ignored by R.

```
plot(m1$residuals ~ mlb11$at_bats)
abline(h = 0, lty = 3)  # adds a horizontal dashed line at y = 0
```

6. Is there any apparent pattern in the residuals plot? What does this indicate about the linearity of the relationship between runs and at-bats?

Nearly normal residuals: To check this condition, we can look at a histogram

```
hist(m1$residuals)
```

or a normal probability plot of the residuals.

```
qqnorm(m1$residuals)
qqline(m1$residuals)  # adds diagonal line to the normal prob plot
```

7. Based on the histogram and the normal probability plot, does the nearly normal residuals condition appear to be met?

Constant variability:

8. Based on the plot in (1), does the constant variability condition appear to be met?

On Your Own

- Choose another traditional variable from `mlb11` that you think might be a good predictor of `runs`. Produce a scatterplot of the two variables and fit a linear model. At a glance, does there seem to be a linear relationship?
- How does this relationship compare to the relationship between `runs` and `at_bats`? Use the R^2 values from the two model summaries to compare. Does your variable seem to predict `runs` better than `at_bats`? How can you tell?
- Now that you can summarize the linear relationship between two variables, investigate the relationships between `runs` and each of the other five traditional variables. Which variable best predicts `runs`? Support your conclusion using the graphical and numerical methods we've discussed (for the sake of conciseness, only include output for the best variable, not all five).
- Now examine the three newer variables. These are the statistics used by the author of *Moneyball* to predict a teams success. In general, are they more or less effective at predicting runs that the old variables? Explain using appropriate graphical and numerical evidence. Of all ten variables we've analyzed, which seems to be the best predictor of `runs`? Using the limited (or not so limited) information you know about these baseball statistics, does your result make sense?
- Check the model diagnostics for the regression model with the variable you decided was the best predictor for runs.

Multiple Linear Regression

The source for this topic is the Open Intro labs https://www.openintro.org/stat/labs.php

Grading the professor

Many college courses conclude by giving students the opportunity to evaluate the course and the instructor anonymously. However, the use of these student evaluations as an indicator of course quality and teaching effectiveness is often criticized because these measures may reflect the influence of non-teaching related characteristics, such as the physical appearance of the instructor. The article titled, "Beauty in the classroom: instructors' pulchritude and putative pedagogical productivity" (Hamermesh and Parker, 2005) found that instructors who are viewed to be better looking receive higher instructional ratings. (Daniel S. Hamermesh, Amy Parker, Beauty in the classroom: instructors pulchritude and putative pedagogical productivity, *Economics of Education Review*, Volume 24, Issue 4, August 2005, Pages 369-376, ISSN 0272-7757, 10.1016/j.econedurev.2004.07.013. http://www.sciencedirect.com/science/article/pii/S0272775704001165.)

In this lab we will analyze the data from this study in order to learn what goes into a positive professor evaluation.

The data

The data were gathered from end of semester student evaluations for a large sample of professors from the University of Texas at Austin. In addition, six students rated the professors' physical appearance. (This is aslightly modified version of the original data set that was released as part of the replication data for *Data Analysis Using Regression and Multilevel/Hierarchical Models* (Gelman and Hill, 2007).) The result is a data frame where each row contains a different course and columns represent variables about the courses and professors.

```
load("data/evals.RData")
```

variable	description
`score`	average professor evaluation score: (1) very unsatisfactory - (5) excellent.
`rank`	rank of professor: teaching, tenure track, tenured.

variable	description
ethnicity	ethnicity of professor: not minority, minority.
gender	gender of professor: female, male.
language	language of school where professor received education: english or non-english.
age	age of professor.
cls_perc_eval	percent of students in class who completed evaluation.
cls_did_eval	number of students in class who completed evaluation.
cls_students	total number of students in class.
cls_level	class level: lower, upper.
cls_profs	number of professors teaching sections in course in sample: single, multiple.
cls_credits	number of credits of class: one credit (lab, PE, etc.), multi credit.
bty_f1lower	beauty rating of professor from lower level female: (1) lowest - (10) highest.
bty_f1upper	beauty rating of professor from upper level female: (1) lowest - (10) highest.

variable	description
`bty_f2upper`	beauty rating of professor from second upper level female: (1) lowest - (10) highest.
`bty_m1lower`	beauty rating of professor from lower level male: (1) lowest - (10) highest.
`bty_m1upper`	beauty rating of professor from upper level male: (1) lowest - (10) highest.
`bty_m2upper`	beauty rating of professor from second upper level male: (1) lowest - (10) highest.
`bty_avg`	average beauty rating of professor.
`pic_outfit`	outfit of professor in picture: not formal, formal.
`pic_color`	color of professor's picture: color, black & white.

Exploring the data

1. Is this an observational study or an experiment? The original research question posed in the paper is whether beauty leads directly to the differences in course evaluations. Given the study design, is it possible to answer this question as it is phrased? If not, rephrase the question.
2. Describe the distribution of `score`. Is the distribution skewed? What does that tell you about how students rate courses? Is this what you expected to see? Why, or why not?
3. Excluding `score`, select two other variables and describe their relationship using an appropriate visualization (scatterplot, side-by-side boxplots, or mosaic plot).

Simple linear regression

The fundamental phenomenon suggested by the study is that better looking teachers are evaluated more favorably. Let's create a scatterplot to see if this appears to be the case:

```
plot(evals$score ~ evals$bty_avg)
```

Before we draw conclusions about the trend, compare the number of observations in the data frame with the approximate number of points on the scatterplot. Is anything awry?

4. Replot the scatterplot, but this time use the function `jitter()` on the y- or the x-coordinate. (Use `?jitter` to learn more.) What was misleading about the initial scatterplot?

5. Let's see if the apparent trend in the plot is something more than natural variation. Fit a linear model called `m_bty` to predict average professor score by average beauty rating and add the line to your plot using `abline(m_bty)`. Write out the equation for the linear model and interpret the slope. Is average beauty score a statistically significant predictor? Does it appear to be a practically significant predictor?

6. Use residual plots to evaluate whether the conditions of least squares regression are reasonable. Provide plots and comments for each one (see the Simple Regression Lab for a reminder of how to make these).

Multiple linear regression

The data set contains several variables on the beauty score of the professor: individual ratings from each of the six students who were asked to score the physical appearance of the professors and the average of these six scores. Let's take a look at the relationship between one of these scores and the average beauty score.

```
plot(evals$bty_avg ~ evals$bty_f1lower)
cor(evals$bty_avg, evals$bty_f1lower)
```

As expected the relationship is quite strong - after all, the average score is calculated using the individual scores. We can actually take a look at the relationships between all beauty variables (columns 13 through 19) using the following command:

```
plot(evals[,13:19])
```

These variables are collinear (correlated), and adding more than one of these variables to the model would not add much value to the model. In this application and with these highly-correlated predictors, it is reasonable to use the average beauty score as the single representative of these variables.

In order to see if beauty is still a significant predictor of professor score after we've accounted for the gender of the professor, we can add the gender term into the model.

```
m_bty_gen <- lm(score ~ bty_avg + gender, data = evals)
summary(m_bty_gen)
```

7. P-values and parameter estimates should only be trusted if the conditions for the regression are reasonable. Verify that the conditions for this model are reasonable using diagnostic plots.

8. Is `bty_avg` still a significant predictor of `score`? Has the addition of `gender` to the model changed the parameter estimate for `bty_avg`?

Note that the estimate for `gender` is now called `gendermale`. You'll see this name change whenever you introduce a categorical variable. The reason is that R recodes `gender` from having the values of `female` and `male` to being an indicator variable called `gendermale` that takes a value of 0 for females and a value of 1 for males. (Such variables are often referred to as "dummy" variables.)

As a result, for females, the parameter estimate is multiplied by zero, leaving the intercept and slope form familiar from simple regression.

$$\begin{aligned}\widehat{score} &= \hat{\beta}_0 + \hat{\beta}_1 \times bty\_avg + \hat{\beta}_2 \times (0) \\ &= \hat{\beta}_0 + \hat{\beta}_1 \times bty\_avg\end{aligned}$$

We can plot this line and the line corresponding to males with the following custom function.

```
multiLines(m_bty_gen)
```

9. What is the equation of the line corresponding to males? (*Hint:* For males, the parameter estimate is multiplied by 1.) For two professors who received the same beauty rating, which gender tends to have the higher course evaluation score?

The decision to call the indicator variable `gendermale` instead of`genderfemale` has no deeper meaning. R simply codes the category that comes first alphabetically as a 0. (You can change the reference level of a categorical variable, which is the level that is coded as a 0, using the`relevel` function. Use `?relevel` to learn more.)

10. Create a new model called `m_bty_rank` with `gender` removed and `rank` added in. How does R appear to handle categorical variables that have more than two levels? Note that the rank variable has three levels: `teaching`, `tenure track`, `tenured`.

The interpretation of the coefficients in multiple regression is slightly different from that of simple regression. The estimate for `bty_avg` reflects how much higher a group of professors is expected to score if they have a beauty rating that is one point higher *while holding all other variables constant.* In this case, that translates into considering only professors of the same rank with `bty_avg` scores that are one point apart.

The search for the best model

We will start with a full model that predicts professor score based on rank, ethnicity, gender, language of the university where they got their degree, age, proportion of students that filled out evaluations, class size, course level, number of professors, number of credits, average beauty rating, outfit, and picture color.

11. Which variable would you expect to have the highest p-value in this model? Why? *Hint:* Think about which variable would you expect to not have any association with the professor score.

Let's run the model…

```
m_full <- lm(score ~ rank + ethnicity + gender + language + age + cls_perc_eval
             + cls_students + cls_level + cls_profs + cls_credits + bty_avg
             + pic_outfit + pic_color, data = evals)
summary(m_full)
```

12. Check your suspicions from the previous exercise. Include the model output in your response.

13. Interpret the coefficient associated with the ethnicity variable.

14. Drop the variable with the highest p-value and re-fit the model. Did the coefficients and significance of the other explanatory variables change? (One of the things that makes multiple regression interesting is that coefficient estimates depend on the other variables that are included in the model.) If not, what does this say about whether or not the dropped variable was collinear with the other explanatory variables?

15. Using backward-selection and p-value as the selection criterion, determine the best model. You do not need to show all steps in your answer, just the output for the final model. Also, write out the linear model for predicting score based on the final model you settle on.

16. Verify that the conditions for this model are reasonable using diagnostic plots.

17. The original paper describes how these data were gathered by taking a sample of professors from the University of Texas at Austin and including all courses that they have taught. Considering that each row represents a course, could this new information have an impact on any of the conditions of linear regression?

18. Based on your final model, describe the characteristics of a professor and course at University of Texas at Austin that would be associated with a high evaluation score.

19. Would you be comfortable generalizing your conclusions to apply to professors generally (at any university)? Why or why not?

Bibliography