Introductory Statistics with R tidyverse

Fan Wang

2020-05-02

Contents

Preface

This is a work-in-progress course website for Introductory Statistics for Undergraduate Students, produced by Fan. Course covers a limited subset of topics from *Statistics for Business and Economics* (Anderson Sweeney Williams Camm Cochran 12e).

R is used. Packages from Tidyverse (Wickham, 2019) are used, including tibble (Müller and Wickham, 2019) for framing data, tidyr (Wickham and Henry, 2019) and dplyr (Wickham et al., 2019b) for reshaping data and aggregating statistics, ggplot2 (Wickham et al., 2019a) for graphing, and readr (Wickham et al., 2018) for file reading and writing. Materials are presented as R, RMD, PDF and HTML files. To obtain all codes and raw files, see here for github set up. For HTML files, click on the links below.

From Fan's other repositories: For dynamic borrowing and savings problems, see Dynamic Asset Repository; For code examples, see also R Example Code, Matlab Example Code, and Stata Example Code; For intro econ with Matlab, see Intro Mathematics for Economists. See here for all of Fan's public repositories.

The site is built using Bookdown (Xie, 2020).

Please contact FanWangEcon for issues or problems.

Chapter 1

Survey

1.1 Generate A Dataset in R

> Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

R has a Variety of built-in small datasets for you to experiment with, for example, opening up R, you can just type in: mtcars, and that will show you some variables and observations. The command below shows the first 5 rows of dataset **mtcars**:

```
head(mtcars, 5)
```

There are 52 students in a class. I am interested in how many of you go to games at the University, and how many years each of you has lived in the city of Houston. But I do not have time to ask each of you questions.

- My population is all 52 students in the class.
- Given my limited time and resources, I will gather a sample of just 10 students.

Gathering information regarding game attendance for the 10 students, I can perhaps gain some insights about the population.

1.1.1 A Random Sample of Students in Class

I can point and pick ten random people, but how do I know I am choosing/selecting randomly? In general, we want to pick a random sample from the population. You don't want your sample of 10 to just be students in the first row or students who happen to be giving eye contacts. We want a random sample that hopefully gives us some representative information about the population of 52 students.

How do you pick random numbers?

Siri, Alexa, Google, …, you can get random numbers very easily now. You can download various apps also that allows you to draw a random integer from say between 1 to 4, with an equal chance of drawing any of the four numbers.

In our class: - we have 4 rows of seats - we have 13 columns of seats - there are 52 students in the class together. I will now randomly pick between 1 and 4, 10 times, and randomly pick between 1 and 13 also 10 times.

This gives me ten pairs of row and columns numbers, and these indicate which ten students will be a part of a randomly drawn sample of students.

```
# Setting seed means we will get the same set of random numbers each time
set.seed(12345)
# in R, draw integers between 1 and 4, 10 times
ROW.RAND.DRAWS = sample(1:4, 10, replace = TRUE)
# in R, draw integers between 1 and 13, 10 times
```

```
COL.RAND.DRAWS = sample(1:13, 10, replace = TRUE)
# Note that the way we are drawing randomly, we could draw the same individual twice.
rbind(ROW.RAND.DRAWS, COL.RAND.DRAWS)
```

```
##                [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
## ROW.RAND.DRAWS    2    3    4    2    4    4    2    1    3     4
## COL.RAND.DRAWS    8    2    6   11    6    7   10    1    8     7
```

1.1.1.1 Create the Dataset Row by Row

R has *base* packages. Many of R's best packages do not come with the base packages. You have to install separately. One of the best packages for data science is tidyverse. **tidyverse** has a great package called tibble, which is a great way to deal with what are called dataframes, where we store our data. **tidyverse** also has dplyr, which is a powerful set of tools for transforming data.

Sequentially, in class, we ask each of the 10 randomly drawn students a number of questions, and record the answers in the dataframe.

```
df <- tibble(ID = integer(), ROW = integer(), COL = integer(),
             gender = factor(),
             years.in.houston = double(),
             major = factor(),
             commute = factor(),
             games.attended = integer())


# Ask Students Questions and Record Answers
df <- add_row(df, ID=1 , ROW=3, COL=1,  gender='MALE',   years.in.houston=21.0,
              major='ECON', commute='YES', games.attended=0)
df <- add_row(df, ID=2 , ROW=4, COL=2,  gender='FEMALE', years.in.houston=21.0,
              major='HEALTH', commute='YES', games.attended=2)
df <- add_row(df, ID=3 , ROW=4, COL=10, gender='MALE',   years.in.houston=22.0,
              major='ECON', commute='YES', games.attended=0)
df <- add_row(df, ID=4 , ROW=4, COL=1,  gender='MALE',   years.in.houston=22.0,
              major='ECON', commute='YES', games.attended=14)
df <- add_row(df, ID=5 , ROW=2, COL=6,  gender='FEMALE', years.in.houston=20.0,
              major='ECON', commute='YES', games.attended=0)
df <- add_row(df, ID=6 , ROW=1, COL=7,  gender='MALE',   years.in.houston=3.0,
              major='PSYCH', commute='YES', games.attended=0)
df <- add_row(df, ID=7 , ROW=2, COL=6,  gender='MALE',   years.in.houston=25.0,
              major='ECON', commute='YES', games.attended=25)
df <- add_row(df, ID=8 , ROW=3, COL=6,  gender='MALE',   years.in.houston=20.0,
              major='CONSUMERSCIENCE', commute= 'YES', games.attended=2)
df <- add_row(df, ID=9 , ROW=3, COL=3,  gender='FEMALE', years.in.houston=5.0,
              major='HUMANRESOURCE', commute='YES', games.attended=0)
df <- add_row(df, ID=10, ROW=4, COL=13, gender='FEMALE', years.in.houston=20.0,
              major='ECON', commute='YES', games.attended=0)

# List All Variables in df
str(df)
```

```
## tibble [10 x 8] (S3: tbl_df/tbl/data.frame)
##  $ ID              : num [1:10] 1 2 3 4 5 6 7 8 9 10
##  $ ROW             : num [1:10] 3 4 4 4 2 1 2 3 3 4
##  $ COL             : num [1:10] 1 2 10 1 6 7 6 6 3 13
##  $ gender          : chr [1:10] "MALE" "FEMALE" "MALE" "MALE" ...
##  $ years.in.houston: num [1:10] 21 21 22 22 20 3 25 20 5 20
##  $ major           : chr [1:10] "ECON" "HEALTH" "ECON" "ECON" ...
##  $ commute         : chr [1:10] "YES" "YES" "YES" "YES" ...
##  $ games.attended  : num [1:10] 0 2 0 14 0 0 25 2 0 0
```

```
# Show full df Table
df
```

1.1.1.2 Analyze the Dataset and Create Additional Variables

Based on the variables we gathered, we can generate some additional variables. Specifically, we will generate dummy variables for games.attended, and major. Let's Generate Basic Sample Statistics. Summary will

```
# Generate Binary Variable of Having attended any games or not
df$games.any <- ifelse(df$games.attended>0, 1, 0)
df$games.any <- factor(df$games.any, levels = c(1,0), labels = c('Has.Attended', 'Never.Attended'))

# Generate Binary Variable Econ or Not, 1 if major is ECON, 0 otherwise
df$econ <- ifelse(df$major=='ECON', 1, 0)
df$econ <- factor(df$econ, levels = c(1, 0), labels = c('ECON', 'Not.Econ'))

# new data set with the two additional variables
df
```

We have a continuous/quantitative variable for games.attended, based on which we generated a binary/dummy variable of games.any to indicate if someone has attended any games or not.

- What fraction of students in the sample have attended games?
  - 60 percent of students never attended any games
- What is the average number of games students attended?
  - But on average students attended 4.3 games (because one student attended 25 games, and another student in the sample attended 10 games)

Suppose you are running the sports program, and each attendee of games pays the same ticket price, if you only care about total revenue, you only care about the average of 4.3 games attended per student. It does not matter if 1 student attended 10 games, or 10 students attended 1 game each, you get the same total revenue, and the same game per student attended.

But if you are trying to generate school spirit from the football program, you might care more about the 60 percent number, which you might think is too high. Too many students never going to any games.

If someone is writing an article about the popularity of football at the University, they can use the 4.3 number to say that it is really popular, or the 60 non-attendance ratio to say it is not that popular. So you see when we read news reports that have data, or when others provide us with statistics, we have to be careful and think if what they report are showing the full picture. The same simple attendance variable, calculated in two ways, shows two pictures of how popular football is.

```
# Using the dplyr package, we can find the fraction of individuals in factor variables
# Fraction of individuals who attended any games
df %>%
  group_by(games.any) %>%
  summarise (freq = n()) %>%
  mutate(fraction = freq / sum(freq))
```

```
# Average Game Attendance Overall, and for each sub-group, conditional on attending any or no attend
# This creates a column that has overall average for games.attended, showing in every row:
df %>% summarise(games.attended.avg =mean(games.attended))
# This creates a column that has overall average for games.attended, showing in every row:
df %>% group_by(games.any) %>% summarise(games.attended.avg = mean(games.attended))
```

```
# To show both overall average and the group specific average together
#   the first mutate adds to df a column that has the overall average, same value every row
#   then when we group by, we can calculate within group average for games.attended
#   we can also take the average of the avg var, which is the same for all rows, so grouped averages
df %>%
   mutate(avg=mean(games.attended)) %>%
   group_by(games.any) %>%
   summarise(avg.group=mean(games.attended),avg.overall=mean(avg))
```

If we drew a different set of 10 people from the class, we would likely get different statistics than what we have in the current random sample. We want to make inference based on the sample about the population knowing that each set of random draws could give different sample statistics. Sample statistics is not population statistics. If our samples are smaller than 10, our sample statistics are likely to be even further away from population statistics.

1.1.1.3 Store the data

A standard format for data storage is CSV (comma separated files). The file has texts and can be opened by any text editor. Each row of text corresponds to a row in the table above, which is an observation in the dataset. Commas separate values for each variable. The first row stores the list of column variable names, also separated by commas.

```
# This File is written in a subfolder survey of folder Stat4Econ
# In Stat4Econ, there is another folder called data
# Typing in ../ means go back to the master folder
# /data/ means go to the parallel subfolder data and store our csv file there
write_csv(df , path = 'data/classsurvey.csv')
```

Chapter 2

Dataset, Tables and Graphs

2.1 Opening a Dataset

> Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

We have a dataset on basketball teams. The dataset, *Basketball.csv*, can be downloaded here.

We will load in the dataset and do some analysis with it.

2.1.1 Paths to Data

Relative Path

The dataset is stored in a csv file. The folder structure for this file we are working inside and the data file is:

- main folder: Stat4Econ
  - subfolder: data
    - file: Basketball.csv
  - subfolder: descriptive
    - file: DataBasketball.ipynb (the jupyter notebook file)
    - file: DataBasetball.html (the html version of the jupyter notebook file

overall this means: - the csv file's location is: **'/Stat4Econ/descriptive/data/Basketball.csv'** - the working R code file's location is: **'/Stat4Econ/descriptive/data/DataBasketball.ipynb'**

Given this structure, to access the *Basketball.csv* dataset, we need to go one folder up from our current subfolder to the mainfolder, and then choose the data subfolder, and the Basketball.csv file in the subfolder.

Absolute Path

If these files are not in the same main folder but are in different locations on your computer, you can find the full path to the csv path and copy paste the path below in between the single quotes.

search on google to find out how to get the full path to file: - google search for find full path for file on mac + this might end up looking like: **'/Users/fan/Downloads/Basketball.csv'** - google search for find full path for file on PC + this might end up looking like: **'C:/Users/fan/Documents/Dropbox/Basketball.csv'**

Using Relative path to load in data

We will load in the data using base R read.csv function.

- For what the variables mean, see here
- For what NBA team names correspond to, see here.

```
# We can load the dataset in first by setting our directory, then loading in the dataset
basetball_data <- read.csv('data/Basketball.csv')
```

```r
# Alternatively, we can just use one line
basetball_data <- read.csv('data/Basketball.csv')
```

```r
# Summarize all variables in data frame
summary(basetball_data)
```

```
##        ilkid             year        firstname         lastname           team       leag
##  WILLIKE01 :   27   Min.   :1946   John    : 502   Williams:  441   TOT    : 1611   A: 1247   Min
##  EDWARJA01 :   25   1st Qu.:1974   Bob     : 404   Johnson :  400   NYK    :  947   N:20712   1st
##  CORBITY01 :   24   Median :1988   Mike    : 395   Smith   :  317   BOS    :  919             Med
##  JACKSJI01 :   24   Mean   :1986   Jim     : 344   Jones   :  298   DET    :  799             Mea
##  CASSESA01 :   23   3rd Qu.:1999   Chris   : 267   Davis   :  254   PHI    :  736             3rd
##  MALONMO01 :   23   Max.   :2009   Tom     : 260   Brown   :  209   LAL    :  712             Max
##  (Other)   :21813                  (Other):19787   (Other) :20040   (Other):16235
##       dreb             reb              asts             stl             blk           turnover
##  Min.   :   0.0   Min.   :   0.0   Min.   :   0.0   0      : 5939   0      : 6855   0      : 5696
##  1st Qu.:   1.0   1st Qu.:  44.0   1st Qu.:  20.0   1      :  607   1      : 1126   1      :  349
##  Median :  60.0   Median : 160.0   Median :  71.0   2      :  479   2      :  860   2      :  347
##  Mean   : 117.8   Mean   : 229.7   Mean   : 118.1   3      :  445   3      :  754   NULL   :  294
##  3rd Qu.: 180.0   3rd Qu.: 333.0   3rd Qu.: 167.0   5      :  345   4      :  647   3      :  287
##  Max.   :1538.0   Max.   :2149.0   Max.   :1164.0   4      :  343   5      :  608   5      :  246
##                                                     (Other):13801   (Other):11109   (Other):14740
##       ftm              tpa              tpm
##  Min.   :  0.0   Min.   :  0.00   Min.   :  0.0
##  1st Qu.: 20.0   1st Qu.:  0.00   1st Qu.:  0.0
##  Median : 70.0   Median :  2.00   Median :  0.0
##  Mean   :109.6   Mean   : 38.07   Mean   : 13.1
##  3rd Qu.:161.0   3rd Qu.: 27.00   3rd Qu.:  7.0
##  Max.   :840.0   Max.   :678.00   Max.   :269.0
##
```

2.2 One Variable Graphs and Tables

Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

```r
# For Data Structures
library(tibble)
# For Data Manipulations
library(dplyr)
# For Reading/Loading Data
library(readr)
# For plotting
library(ggplot2)
# For Additional table output
# install.packages("knitr")
library(knitr)
```

Let's load in the dataset we created from the in-class survey.

```r
# Load the dataset using readr's read_csv
df_survey <- read_csv('data/classsurvey.csv')
```

```r
# We have several factor variables, we can set them as factor one by one
df_survey[['gender']] <- as.factor(df_survey[['gender']])
# But that is a little cumbersome, we can using lapply, a core function in r to do this for all fact
factor_col_names <- c('gender', 'major', 'commute', 'games.any', 'econ')
df_survey[factor_col_names] <- lapply(df_survey[factor_col_names], as.factor)
# Check Variable Types
```

```
str(df_survey)
```

```
## tibble [10 x 10] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ ID              : num [1:10] 1 2 3 4 5 6 7 8 9 10
##  $ ROW             : num [1:10] 3 4 4 4 2 1 2 3 3 4
##  $ COL             : num [1:10] 1 2 10 1 6 7 6 6 3 13
##  $ gender          : Factor w/ 2 levels "FEMALE","MALE": 2 1 2 2 1 2 2 2 1 1
##  $ years.in.houston: num [1:10] 21 21 22 22 20 3 25 20 5 20
##  $ major           : Factor w/ 5 levels "CONSUMERSCIENCE",..: 2 3 2 2 2 5 2 1 4 2
##  $ commute         : Factor w/ 1 level "YES": 1 1 1 1 1 1 1 1 1 1
##  $ games.attended  : num [1:10] 0 2 0 14 0 0 25 2 0 0
##  $ games.any       : Factor w/ 2 levels "Has.Attended",..: 2 1 2 1 2 2 1 1 2 2
##  $ econ            : Factor w/ 2 levels "ECON","Not.Econ": 1 2 1 1 1 2 1 2 2 1
##  - attr(*, "spec")=
##   .. cols(
##   ..   ID = col_double(),
##   ..   ROW = col_double(),
##   ..   COL = col_double(),
##   ..   gender = col_character(),
##   ..   years.in.houston = col_double(),
##   ..   major = col_character(),
##   ..   commute = col_character(),
##   ..   games.attended = col_double(),
##   ..   games.any = col_character(),
##   ..   econ = col_character()
##   .. )
```

2.2.1 Categorical/Discrete

Using categorical/discrete variables, such as Major, Gender, Recent arrival or not, etc, we can generate frequency tables.

- Nominal Variables: These are categorical variables like name, or address, or major.
- Ordinal Variables: These are categorical variables like age, grade in school, that could be ordered sequentially.

The frequency table shows in separate columns (or rows) the name for all the categories for that categorical variable, and show next to these categories the number of times that category appears in the dataset. This is called a frequency table because frequencies are shown for each category. So if we show a one-way frequency for majors, there would be four rows for the four unique majors from our survey of 10 students, and we would write down the number of students in each of the majors out of the 10 students.

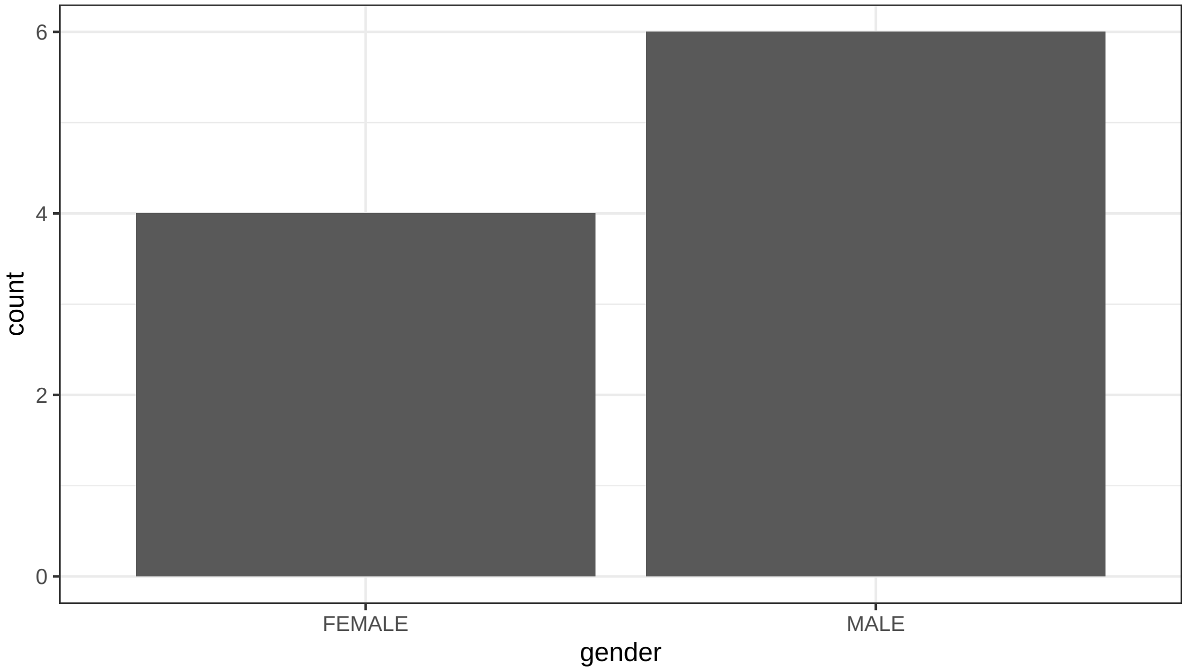
Rather than showing frequencies, we can also show ratios or percentages by dividing the number of observations in each category by the total number of observations in the survey.

Graphically, we can show the results from these one-way frequency tables using bar charts and pie charts. The bar charts would have separate bars for each category, and the heights of the bars would show the number of observations in that category, or the fraction of individuals in the categorical. The relative heights of bars are the same whether we show the frequencies or the fractions/ratios. A pie chart might give a more direct visual sense of the fraction of observations in each category.

2.2.1.1 A Frequency Table and a Bar Graph

```
# This Generates A Frequency table for the number of students of each gender
df_survey %>%
    group_by(gender) %>%
    summarise (frequency.count = n()) %>%
      mutate(proportions = frequency.count / sum(frequency.count))
```

```r
# We can make a bar graph for the Frequency Table
# graph size
options(repr.plot.width = 2, repr.plot.height = 2)
# Graph
bar.plot <- ggplot(df_survey) +
    geom_bar(aes(x=gender)) +
    theme_bw()
print(bar.plot)
```



2.2.1.2 Using R Functions to Generate Multiple Frequency Tables

```r
# There are Multiple Categorical Variables in the dataset
# Would be nice if we can generate frequency tables for all of them easily
# Let's create a list of  that are categorical
categorical.nomial.list <- c('gender', 'major', 'games.any', 'commute', 'econ')
```

```r
# Let's Write a Function, not that we need to, but let's do it
# We can give the function any name, here: dplyr.freq.table
dplyr.freq.table <- function(df, cate.var.str){

    # A print Statement
    print(sprintf("From Dataset: %s, Freq. Table for Variable: %s", deparse(substitute(df)), cate.va

    # Note below: !!sym(cate.var.str), because cate.var.str is string
    freq.table <- df %>%
        group_by(!!sym(cate.var.str)) %>%
        summarise (frequency.count = n()) %>%
        mutate(proportions = frequency.count / sum(frequency.count))

    # Function returns
    return(freq.table)
}

# Let's call test this function and generate our earlier table
```

```r
dplyr.freq.table(df = df_survey, cate.var.str = 'gender')
```

```
## [1] "From Dataset: df_survey, Freq. Table for Variable: gender"
```

```r
# Let's Now Use our function to generate Multiple Frequency Tables
# We will first use a explicit loop
for (ctr in seq_along(categorical.nomial.list)) {
    freq.table <- dplyr.freq.table(df = df_survey, cate.var.str = categorical.nomial.list[ctr])
    print(freq.table)
}
```

```
## [1] "From Dataset: df_survey, Freq. Table for Variable: gender"
## [1] "From Dataset: df_survey, Freq. Table for Variable: major"
## [1] "From Dataset: df_survey, Freq. Table for Variable: games.any"
## [1] "From Dataset: df_survey, Freq. Table for Variable: commute"
## [1] "From Dataset: df_survey, Freq. Table for Variable: econ"
```

Using Lapply to Generate Multiple Frequency Tables with a Single Line:

```r
# We will now use lapply, the single line loop tool in R
# Below, we are plugging each element of the list one by one into the function
# dplyr.freq.table, the first argument of the function is the dataset name
# which is fixed as df = df_survey
lapply(categorical.nomial.list,
       dplyr.freq.table,
       df = df_survey)
```

```
## [1] "From Dataset: df_survey, Freq. Table for Variable: gender"
## [1] "From Dataset: df_survey, Freq. Table for Variable: major"
## [1] "From Dataset: df_survey, Freq. Table for Variable: games.any"
## [1] "From Dataset: df_survey, Freq. Table for Variable: commute"
## [1] "From Dataset: df_survey, Freq. Table for Variable: econ"
```

```
## [[1]]
##
## [[2]]
##
## [[3]]
##
## [[4]]
##
## [[5]]
```

2.2.2 Continuous/Quantitative

Graphically, we can show a continuous variable using a histogram. This could be test scores, temperatures, years in Houston, etc. This involves first creating a categorical/discrete variable based on the continuous variable. Since the underlying continuous variable is ordered (low to high temperature unless major which is not ordered), the categorical/discrete variable we generate is an ordered categorical variable (majors could be called unordered categorical variable).

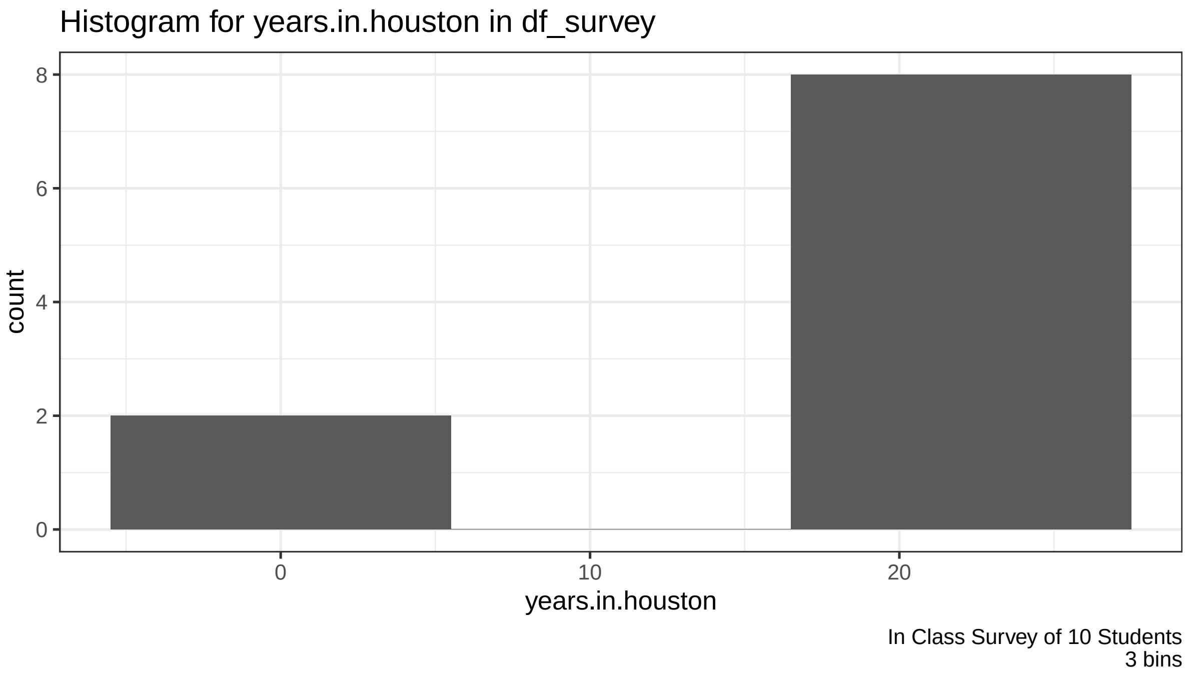
2.2.2.1 Histogram with Lapply

To generate the histogram, we: 1. Make sure that all observations belongs to one category + no observations belonging to no categories + each observation only belongs to one category 2. Each category is equidistance along the continuous variable's original scale. 3. Then we create a bar graph where each bar is a category, and the height of the bar represents the number of observations within that category.

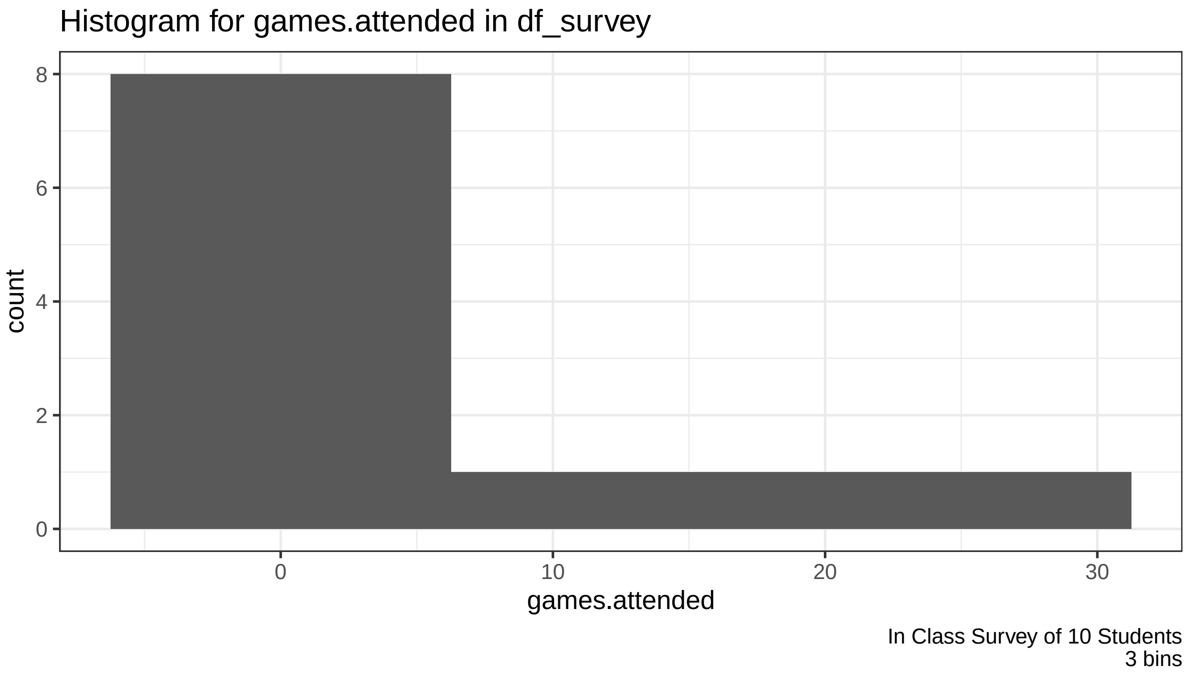
```r
# We will write a histogram function
ggplot.histogram <- function(df, cts.var.str){
    # Figure Size
    options(repr.plot.width = 4, repr.plot.height = 3)

    # Figure Title
    title <- sprintf("Histogram for %s in %s", cts.var.str, deparse(substitute(df)))

    # We have in our 10 student survey only 10 observations
    # We can still generate a histogram for our continuous variables
    # Will only use three bins
    histogram.3bins <- ggplot(df_survey, aes(x=!!sym(cts.var.str))) +
      geom_histogram(bins=3) +
      labs(title = paste0(title),
           caption = 'In Class Survey of 10 Students\n3 bins') +
      theme_bw()

    # obtain the data in the plot
    plot_data <- ggplot_build(histogram.3bins)
    # the dataframe below contains all the information for the histogram
    # bins and number of observations in each bins
    plot_dataframe <- plot_data$data[[1]]

    # return outputs
    return(list(gghist=histogram.3bins, hist.df=plot_dataframe))
}
```

```
# Now the list of continuous Variables and calling the function with lapply
cts.list <- c('years.in.houston', 'games.attended')

# lapply
results <- lapply(cts.list,
       ggplot.histogram,
       df = df_survey)

# Show results
for (ctr in seq_along(cts.list)){
    print(results[ctr])
}
```

```
## [[1]]
## [[1]]$gghist
```



```
##
## [[1]]$hist.df
##   y count  x xmin xmax    density ncount ndensity flipped_aes PANEL group ymin ymax colour   fill
## 1 2     2  0 -5.5  5.5 0.01818182   0.25     0.25       FALSE     1    -1    0    2     NA grey35
## 2 0     0 11  5.5 16.5 0.00000000   0.00     0.00       FALSE     1    -1    0    0     NA grey35
## 3 8     8 22 16.5 27.5 0.07272727   1.00     1.00       FALSE     1    -1    0    8     NA grey35
##
##
## [[1]]
## [[1]]$gghist
```



```
##
## [[1]]$hist.df
##   y count    x  xmin  xmax density ncount ndensity flipped_aes PANEL group ymin ymax colour   fil
## 1 8     8  0.0 -6.25  6.25   0.064  1.000    1.000       FALSE     1    -1    0    8     NA grey3
## 2 1     1 12.5  6.25 18.75   0.008  0.125    0.125       FALSE     1    -1    0    1     NA grey3
## 3 1     1 25.0 18.75 31.25   0.008  0.125    0.125       FALSE     1    -1    0    1     NA grey3
```

2.3 Multiple Variables Graphs and Tables

Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

```
# For Data Manipulations
library(tidyverse)
# For Additional table output
# install.packages("knitr")
library(knitr)
```

Let's load in the dataset we created from the in-class survey.

```
# Load the dataset using readr's read_csv
df_survey <- read_csv('data/classsurvey.csv')
```

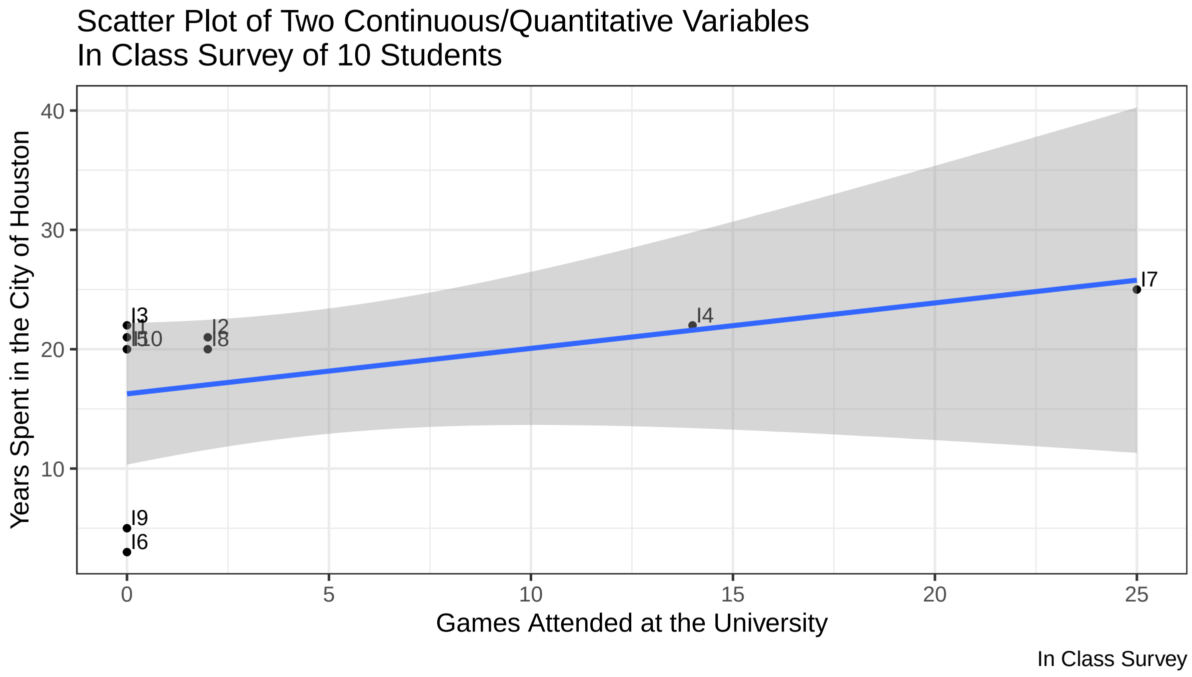
```
# We have several factor variables, we can set them as factor one by one
df_survey[['gender']] <- as.factor(df_survey[['gender']])
# But that is a little cumbersome, we can using lapply, a core function in r to do this for all fact
factor_col_names <- c('gender', 'major', 'commute', 'games.any', 'econ')
df_survey[factor_col_names] <- lapply(df_survey[factor_col_names], as.factor)
# Check Variable Types
str(df_survey)
```

```
## tibble [10 x 10] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ ID              : num [1:10] 1 2 3 4 5 6 7 8 9 10
##  $ ROW             : num [1:10] 3 4 4 4 2 1 2 3 3 4
##  $ COL             : num [1:10] 1 2 10 1 6 7 6 6 3 13
##  $ gender          : Factor w/ 2 levels "FEMALE","MALE": 2 1 2 2 1 2 2 2 1 1
##  $ years.in.houston: num [1:10] 21 21 22 22 20 3 25 20 5 20
##  $ major           : Factor w/ 5 levels "CONSUMERSCIENCE",..: 2 3 2 2 2 5 2 1 4 2
##  $ commute         : Factor w/ 1 level "YES": 1 1 1 1 1 1 1 1 1 1
##  $ games.attended  : num [1:10] 0 2 0 14 0 0 25 2 0 0
##  $ games.any       : Factor w/ 2 levels "Has.Attended",..: 2 1 2 1 2 2 1 1 2 2
##  $ econ            : Factor w/ 2 levels "ECON","Not.Econ": 1 2 1 1 1 2 1 2 2 1
##  - attr(*, "spec")=
##   .. cols(
##   ..   ID = col_double(),
##   ..   ROW = col_double(),
##   ..   COL = col_double(),
##   ..   gender = col_character(),
##   ..   years.in.houston = col_double(),
##   ..   major = col_character(),
##   ..   commute = col_character(),
##   ..   games.attended = col_double(),
##   ..   games.any = col_character(),
##   ..   econ = col_character()
##   .. )
```

2.3.1 Two Continuous Variables

With two continuous/quantitative variables, we can generate a scatter plot. Crucially, each point of the scatter plot represents one individual, the location of that point indicates the x and y values of that individual. The x and y values could be the individual's test score and hours studied for example.

```
# We can draw a scatter plot for two continuous variables
# Control Graph Size
options(repr.plot.width = 4, repr.plot.height = 4)
# Draw Scatter Plot
# 1. specify x and y
# 2. label each individual by their ID, add letter I in front of value
# 3. add in trend line
scatter <- ggplot(df_survey, aes(x=games.attended, y=years.in.houston)) +
      geom_point(size=1) +
      geom_text(aes(label=paste0('I', ID)), size=3, hjust=-.2, vjust=-.2) +
      geom_smooth(method=lm) + # Trend line
      labs(title = paste0('Scatter Plot of Two Continuous/Quantitative Variables'
                          ,'\nIn Class Survey of 10 Students'),
           x = 'Games Attended at the University',
           y = 'Years Spent in the City of Houston',
           caption = 'In Class Survey') +
      theme_bw()
print(scatter)
```



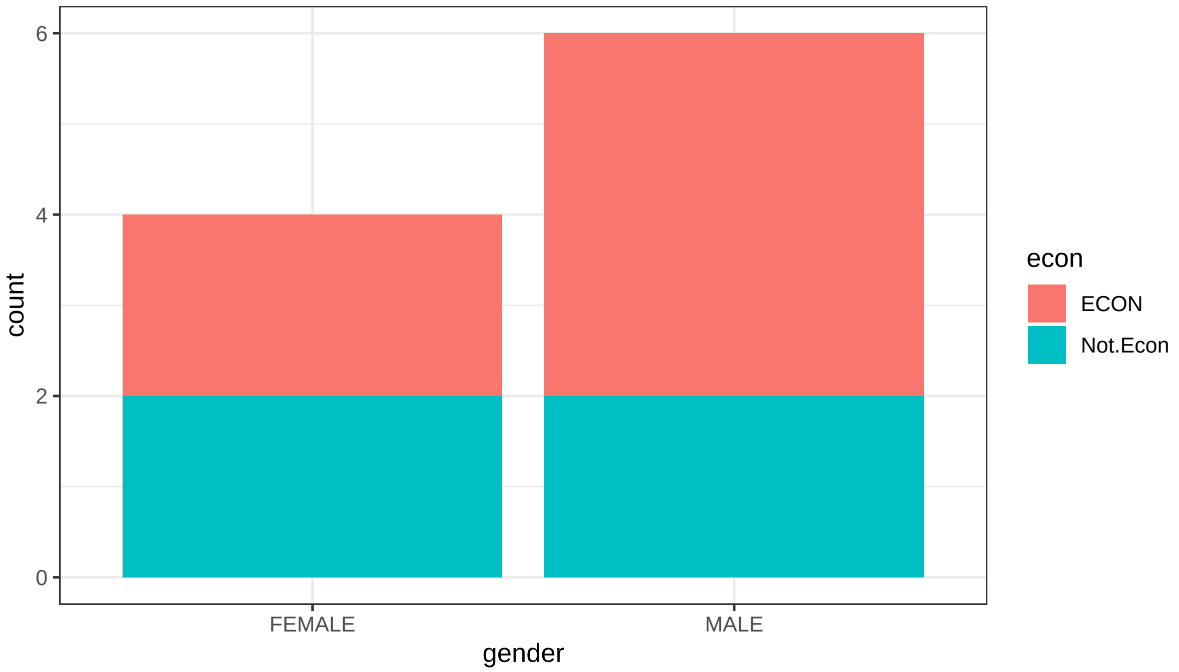
2.3.2 Two Categorical Variables

With two discrete/categorical variables, we can generate two-way frequency tables. This is very similar to what we did for one discrete variable, except now we have columns and rows, representing the categories of the two variables. The two variables could be gender and majors, we would write in each table cell the number of students who are male and econ majors, male and bio majors in for example the first column, and repeat this for girls in the second column.

```
# We can tabulate Frequencies based on two categorical variables
df_survey %>%
  group_by(gender, econ) %>%
  summarize(freq = n()) %>%
  spread(gender, freq)
```

```
# We can show the fraction of individuals in each of the four groups
df_survey %>%
  group_by(interaction(gender, econ)) %>%
  summarise(freq = n()) %>%
  mutate(fraction = freq / sum(freq))
```

```
# We can create stacked bar charts as well with the same data
# graph size
options(repr.plot.width = 3, repr.plot.height = 2)
# Graph
stacked.bar.plot <- ggplot(df_survey) +
    geom_bar(aes(x=gender, fill=econ)) +
    theme_bw()
print(stacked.bar.plot)
```



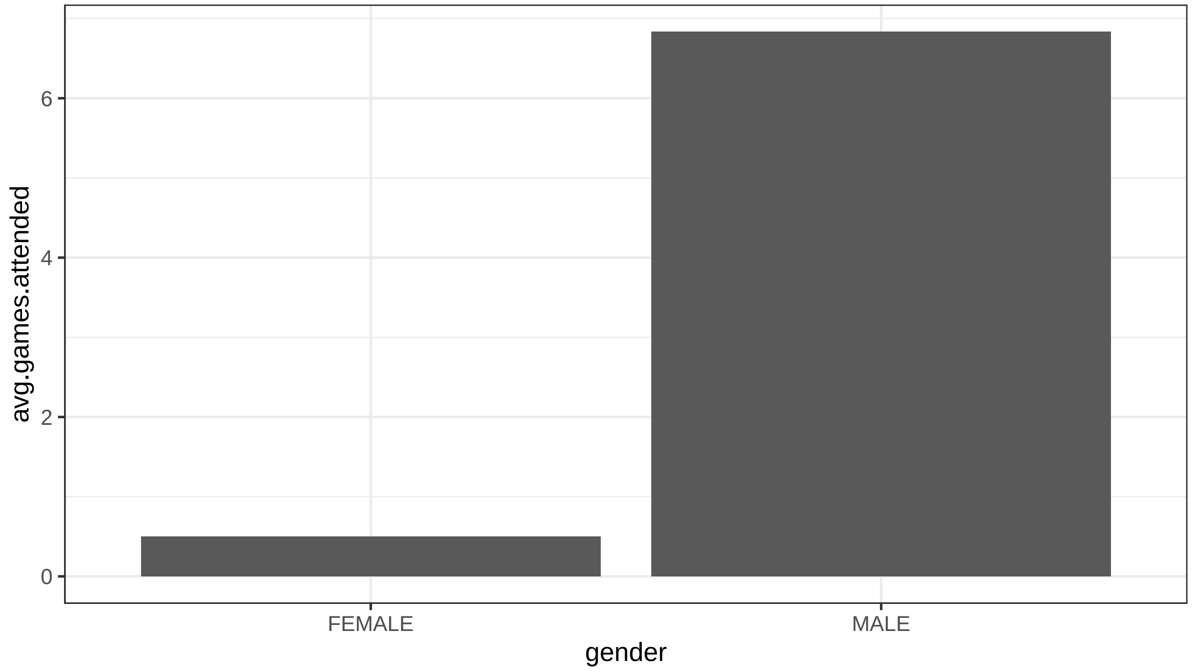
2.3.3 Continuous and Categorical Variable

2.3.3.1 Average Across Groups

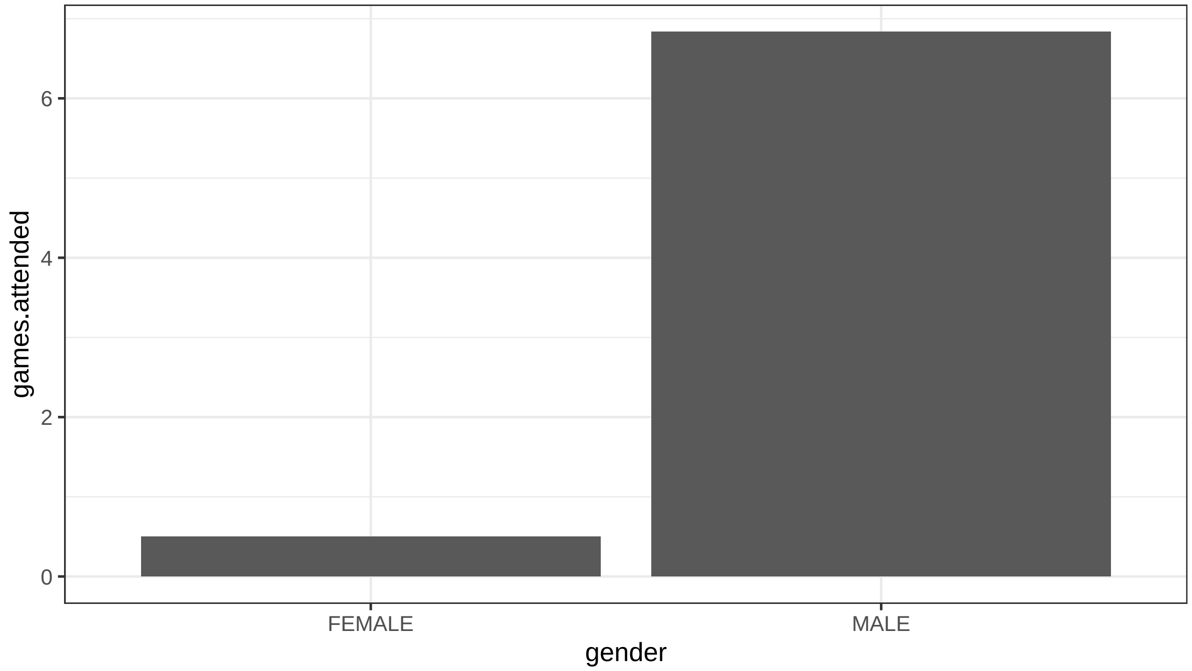
We can look at the average game attendance by female and male students in our sample, using a bar plot, where the height of the bars now represent the average of the *games.attended* variable for each group.

```
# We can first find the group averages
df_gender_avg_games <- df_survey %>%
                        group_by(gender) %>%
                        summarise (avg.games.attended = mean(games.attended))
df_gender_avg_games
```

```
# We can graph based on df_gender_avg_games
# Sizing the Figure Here
options(repr.plot.width = 2, repr.plot.height = 2)
# Plot, stat = identity means to plot the value in avg.games.attended for each gender
group.means <- ggplot(df_gender_avg_games) +
    geom_bar(aes(x=gender, y=avg.games.attended), stat = 'identity') +
    theme_bw()
print(group.means)
```



```
# But it is a little cumbersome to do this in two steps, we can do it in one step
# Sizing the Figure Here
options(repr.plot.width = 2, repr.plot.height = 2)
# Plot directly from df_survey, summary over x for y
# The result looks the same
group.means.joint <- ggplot(df_survey) +
    geom_bar(aes(x=gender, y=games.attended), stat = "summary", fun.y = "mean") +
    theme_bw()
print(group.means.joint)
```

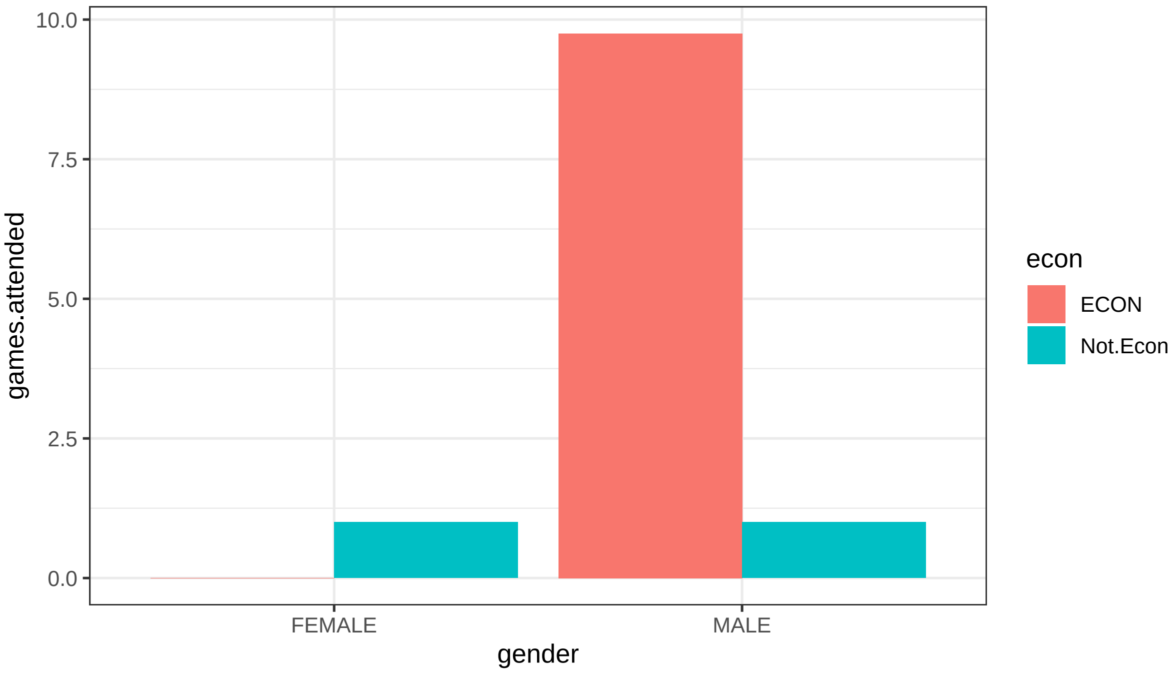


2.3.3.2 Average Across Two Groups: Gender and Majors

What is the average game attendance for male and female, econ and non-econ majors? We have 2 female econ majors, 2 female non-econ majors, 4 male econ majors and 2 male non-econ majors, and their average game attendances are: 0, 1, 9.75 and 1 games.

```
# We can calculate the statistics as a table, and also show obs in each group
df_survey %>%
    group_by(gender, econ ) %>%
    summarise (avg.games.attended = mean(games.attended), N.count = n())
```

```
# Let's Show these Visually
options(repr.plot.width = 4, repr.plot.height = 2)
# Plot directly from df_survey
# Using fill for econ, this means econ or not will fill up with different colors
# Still caculate average
# Postion "dodge" means that econ and non-econ wil be shown next to each other
# By default position is to stack different fill groups on top of each other.
two.group.means <- ggplot(df_survey) +
    geom_bar(aes(x=gender, y=games.attended, fill=econ),
             stat = "summary", fun.y = "mean", position = "dodge") +
    theme_bw()
print(two.group.means)
```



Chapter 3

Summarizing Data

3.1 Mean and Standard Deviation

> Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

3.1.1 Temperature Across Locations over Time

Why do we need the standard deviation? We will demonstrate its usefulness by studying temperature dataset. This dataset covers a variety of cities in the United States across all States and Territories. For each city, we have the average temperature in each month. The unit of observation is at the city/month level. We have variables for the state, the city, the month and the average temperature.

The dataset, *TempCitiesUSA.csv*, can be downloaded here.

```
# Load in Data Tools
# For Reading/Loading Data
library(tidyverse)
# Load in Data
df_temp <- read_csv('data/TempCitiesUSA.csv')
```

Listing Unique Levels for Categorical Variables in the Dataset

We can see that the state and city variables are string variables. We can show unique states and cities by months. In the program below, I append the number of observations for each category.

From the tables below, we can see that each city has 12 observations (for the 12 months), and each state has multiple cities.

```
# A function that shows Unique Values for Categorical Variables in a Table format
show.unique.values <- function(df, cate.var.str, lvl_str_max_len=15){

    # Unique Categories
    unique.cates <- df %>%
        group_by(!!sym(cate.var.str)) %>%
        summarise(freq = n()) %>%
        mutate(distinct_N = paste0(!!sym(cate.var.str), ' (n=', freq, ')')) %>%
        select(distinct_N)

    # At most 10 columns
    unique.count <- dim(unique.cates)[1]
    col.count <- min(ceiling(sqrt(unique.count)), 8)
    row.count <- ceiling(unique.count/col.count)

    # Generate Table to Fill in
    expand.length = row.count*col.count
    unique.cates.expand <- vector(mode = "character", length = expand.length)

    # Unique Categories and Counts
    unique.cates.shorter <- substring(t(unique.cates), first = 1, last = lvl_str_max_len)
    unique.cates.expand[0:unique.count] <- unique.cates.shorter

    # Reshape
    dim(unique.cates.expand) <- c(row.count, col.count)

    # Show
    title <- sprintf("From Dataset: %s, %d unique Levels for: %s",
                     deparse(substitute(df)), unique.count, cate.var.str)
    return(list(title=title,
           levels=unique.cates.expand))
}
```

```
# List of categorical Variables
cate.vars.list <- c('month', 'state', 'city')
lapply(cate.vars.list, show.unique.values, df = df_temp, lvl_str_max_len = 30)
```

```
## [[1]]
## [[1]]$title
## [1] "From Dataset: df_temp, 12 unique Levels for: month"
##
## [[1]]$levels
##      [,1]        [,2]        [,3]        [,4]
## [1,] "1 (n=261)" "4 (n=261)" "7 (n=261)" "10 (n=261)"
## [2,] "2 (n=261)" "5 (n=261)" "8 (n=261)" "11 (n=261)"
## [3,] "3 (n=261)" "6 (n=261)" "9 (n=261)" "12 (n=261)"
##
##
## [[2]]
## [[2]]$title
## [1] "From Dataset: df_temp, 59 unique Levels for: state"
##
## [[2]]$levels
##      [,1]                  [,2]          [,3]                 [,4]                  [,5]
## [1,] "AK (n=204)"          "CO (n=60)"   "ID (n=36)"          "MARSHALL IS. (n=12)" "NC (n=60)"
## [2,] "AL (n=36)"           "CT (n=24)"   "IL (n=48)"          "MD (n=12)"           "ND (n=48)"
## [3,] "AMER SAMOA (n=12)"   "D.C. (n=24)" "IN (n=48)"          "ME (n=24)"           "NE (n=96)"
## [4,] "AR (n=36)"           "DE (n=12)"   "KS (n=60)"          "MI (n=108)"          "NH (n=24)"
## [5,] "AZ (n=60)"           "FL (n=156)"  "KY (n=36)"          "MN (n=60)"           "NJ (n=24)"
## [6,] "CA (n=180)"          "GA (n=60)"   "LA (n=48)"          "MO (n=48)"           "NM (n=36)"
## [7,] "CA. (n=12)"          "HI (n=36)"   "MA (n=36)"          "MS (n=36)"           "NV (n=60)"
## [8,] "CAROLINE IS. (n=12)" "IA (n=36)"   "MARSHALL IS (n=12)" "MT (n=72)"           "NY (n=108)"
##
##
## [[3]]
## [[3]]$title
## [1] "From Dataset: df_temp, 254 unique Levels for: city"
##
## [[3]]$levels
##       [,1]                      [,2]                       [,3]                             [,4]
##  [1,] "ABERDEEN (n=12)"         "BOSTON (n=12)"            "ELKO (n=12)"                    "HOUG
##  [2,] "ABILENE (n=12)"          "BRIDGEPORT (n=12)"        "ELY (n=12)"                     "HOUS
##  [3,] "AKRON (n=12)"            "BROWNSVILLE (n=12)"       "ERIE (n=12)"                    "HUNT
##  [4,] "ALAMOSA (n=12)"          "BUFFALO (n=12)"           "EUGENE (n=12)"                  "HUNT
##  [5,] "ALBANY (n=12)"           "BURLINGTON (n=12)"         "EUREKA (n=12)"                  "HURO
##  [6,] "ALBUQUERQUE (n=12)"      "CAPE HATTERAS (n=12)"      "EVANSVILLE (n=12)"              "INDI
##  [7,] "ALLENTOWN (n=12)"        "CARIBOU (n=12)"            "FAIRBANKS (n=12)"               "INTE
##  [8,] "ALPENA (n=12)"           "CASPER (n=12)"             "FARGO (n=12)"                   "ISLI
##  [9,] "AMARILLO (n=12)"         "CHARLESTON (n=12)"         "FLAGSTAFF (n=12)"               "JACK
## [10,] "ANCHORAGE (n=12)"        "CHARLOTTE (n=12)"          "FLINT (n=12)"                   "JACK
## [11,] "ANNETTE (n=12)"          "CHATTANOOGA (n=12)"        "FORT MYERS (n=12)"              "JOHN
## [12,] "APALACHICOLA (n=12)"     "CHEYENNE (n=12)"           "FORT SMITH (n=12)"              "JUNE
## [13,] "ASHEVILLE (n=12)"        "CLAYTON (n=12)"            "FORT WAYNE (n=12)"              "KAHU
## [14,] "ASTORIA (n=12)"          "CLEVELAND (n=12)"          "FRESNO (n=12)"                  "KALI
## [15,] "ATHENS (n=12)"           "COLORADO SPRINGS (n=12)"   "GAINESVILLE (n=12)"             "KANS
## [16,] "ATLANTA (n=12)"          "COLUMBIA (n=24)"           "GALVESTON (n=12)"               "KEY 
## [17,] "ATLANTIC CITY AP (n=12)" "COLUMBUS (n=24)"           "GLASGOW (n=12)"                 "KING
## [18,] "AUSTIN/BERGSTROM (n=12)" "CONCORD (n=12)"            "GOODLAND (n=12)"                "KNOX
## [19,] "AUSTIN/CITY (n=12)"      "CONCORDIA (n=12)"          "GRAND FORKS (n=12)"             "KODI
## [20,] "AVOCA (n=12)"            "CORPUS CHRISTI (n=12)"     "GRAND ISLAND (n=12)"            "KORO
## [21,] "BAKERSFIELD (n=12)"      "DALLAS-FORT WORTH (n=12)" "GRAND JUNCTION (n=12)"          "KOTZ
## [22,] "BALTIMORE (n=12)"        "DALLAS-LOVE FIELD (n=12)" "GRAND RAPIDS (n=12)"            "KWAJ
## [23,] "BARROW (n=12)"           "DAYTON (n=12)"             "GREAT FALLS (n=12)"             "LA C
## [24,] "BATON ROUGE (n=12)"      "DAYTONA BEACH (n=12)"      "GREEN BAY (n=12)"               "LAKE
## [25,] "BECKLEY (n=12)"          "DEL RIO (n=12)"            "GREENVILLE-SPARTANBURG AP (n=1" "LAND
## [26,] "BETHEL (n=12)"           "DENVER (n=12)"             "GUAM (n=12)"                    "LANS
## [27,] "BILLINGS (n=12)"         "DES MOINES (n=12)"         "HARRISBURG (n=12)"              "LAS 
## [28,] "BINGHAMTON (n=12)"       "DETROIT (n=12)"            "HARTFORD (n=12)"                "LEWI
## [29,] "BISHOP (n=12)"           "DODGE CITY (n=12)"         "HAVRE (n=12)"                   "LEXI
## [30,] "BISMARCK (n=12)"         "DULUTH (n=12)"             "HELENA (n=12)"                  "LIHU
## [31,] "BLUE HILL (n=12)"        "EL PASO (n=12)"            "HILO (n=12)"                    "LINC
## [32,] "BOISE (n=12)"            "ELKINS (n=12)"             "HOMER (n=12)"                   "LITT
##       [,6]                       [,7]                        [,8]
##  [1,] "NEWARK (n=12)"            "ROSWELL (n=12)"            "TAMPA (n=12)"
##  [2,] "NOME (n=12)"              "SACRAMENTO (n=12)"         "TOLEDO (n=12)"
##  [3,] "NORFOLK (n=24)"           "SAINT CLOUD (n=12)"        "TOPEKA (n=12)"
##  [4,] "NORTH LITTLE ROCK (n=12)" "SALEM (n=12)"              "TUCSON (n=12)"
##  [5,] "NORTH PLATTE (n=12)"      "SALT LAKE CITY (n=12)"     "TULSA (n=12)"
##  [6,] "OKLAHOMA CITY (n=12)"     "SAN ANGELO (n=12)"         "TUPELO (n=12)"
##  [7,] "OLYMPIA (n=12)"           "SAN ANTONIO (n=12)"        "UNALAKLEET (n=12)"
##  [8,] "OMAHA (NORTH) (n=12)"     "SAN DIEGO (n=12)"          "VALDEZ (n=12)"
##  [9,] "OMAHA EPPLEY AP (n=12)"   "SAN FRANCISCO AP (n=12)"   "VALENTINE (n=12)"
## [10,] "ORLANDO (n=12)"           "SAN FRANCISCO C.O. (n=12)" "VERO BEACH (n=12)"
## [11,] "PAGO PAGO (n=12)"         "SAN JUAN (n=12)"           "VICTORIA (n=12)"
## [12,] "PENDLETON (n=12)"         "SANTA BARBARA (n=12)"      "WACO (n=12)"
## [13,] "PENSACOLA (n=12)"         "SANTA MARIA (n=12)"        "WAKE ISLAN+B1214D (n=1)"
## [14,] "PEORIA (n=12)"            "SAULT STE. MARIE (n=12)"   "WAKE ISLAND (n=11)"
## [15,] "PHILADELPHIA (n=12)"      "SAVANNAH (n=12)"           "WASHINGTON DULLES AP (n=12)"
## [16,] "PHOENIX (n=12)"           "SCOTTSBLUFF (n=12)"        "WASHINGTON NAT'L AP (n=12)"
## [17,] "PITTSBURGH (n=12)"        "SEATTLE C.O. (n=12)"       "WATERLOO (n=12)"
## [18,] "POCATELLO (n=12)"         "SEATTLE SEA-TAC AP (n=12)" "WEST PALM BEACH (n=12)"
## [19,] "POHNPEI (n=12)"           "SEXTON SUMMIT (n=12)"      "WICHITA (n=12)"
## [20,] "PORT ARTHUR (n=12)"       "SHERIDAN (n=12)"           "WICHITA FALLS (n=12)"
## [21,] "PORTLAND (n=24)"          "SHREVEPORT (n=12)"         "WILLIAMSPORT (n=12)"
## [22,] "PROVIDENCE (n=12)"        "SIOUX CITY (n=12)"         "WILLISTON (n=12)"
## [23,] "PUEBLO (n=12)"            "SIOUX FALLS (n=12)"        "WILMINGTON (n=24)"
## [24,] "QUILLAYUTE (n=12)"        "SOUTH BEND (n=12)"         "WINNEMUCCA (n=12)"
## [25,] "RALEIGH (n=12)"           "SPOKANE (n=12)"            "WINSLOW (n=12)"
## [26,] "RAPID CITY (n=12)"        "SPRINGFIELD (n=24)"        "WORCESTER (n=12)"
## [27,] "REDDING (n=12)"           "ST. LOUIS (n=12)"          "YAKIMA (n=12)"
## [28,] "RENO (n=12)"              "ST. PAUL ISLAND (n=12)"    "YAKUTAT (n=12)"
## [29,] "RICHMOND (n=12)"          "STOCKTON (n=12)"           "YOUNGSTOWN (n=12)"
## [30,] "ROANOKE (n=12)"          "SYRACUSE (n=12)"          "YUMA (n=12)"
## [31,] "ROCHESTER (n=24)"        "TALKEETNA (n=12)"         ""
## [32,] "ROCKFORD (n=12)"         "TALLAHASSEE (n=12)"       ""
```

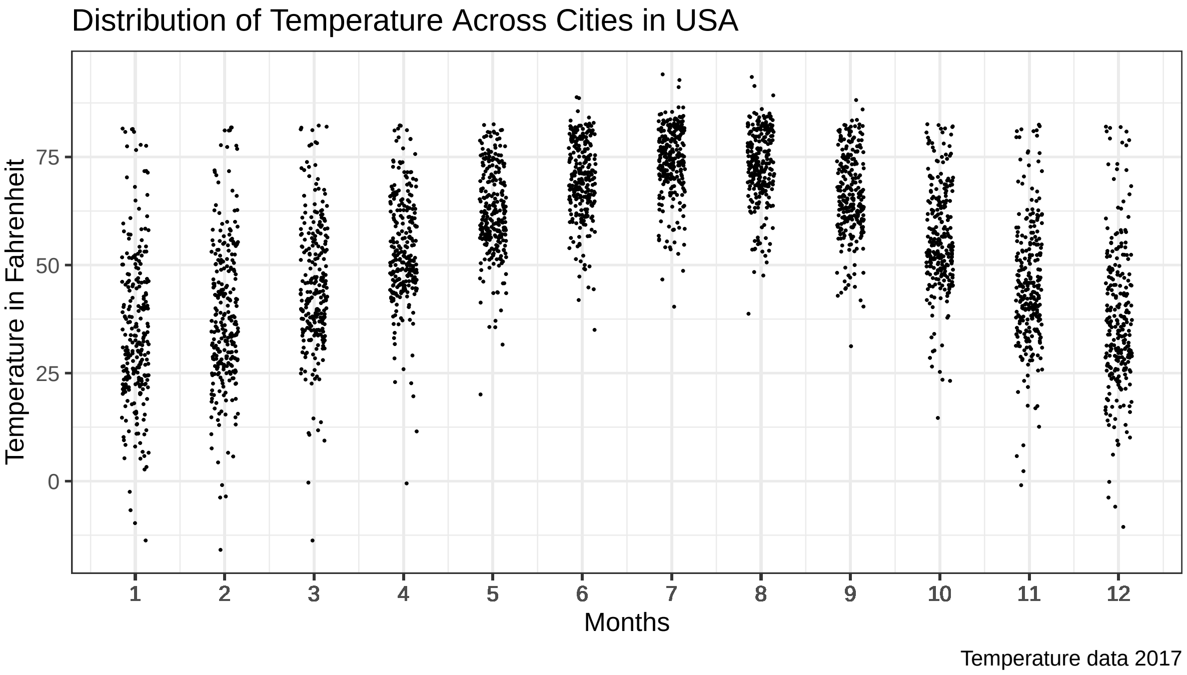
3.1.1.1 Scatter Plot of Temperature and Months

We can do a scatter plot where the x-axis is a month and the y-axis is the temperature in each city, to get a sense of the distribution of temperatures. What does this chart show us? Is this the pattern you would have expected?

- the overall temperature is higher during summer months
- the temperature is more tightly distributed during summer months than January or December

The United State is pretty big, during the winter months some places are frigid, and other areas are very hot. During the summer months, however, most places are warmer.

```
# Control Graph Size
options(repr.plot.width = 5, repr.plot.height = 5)
# Draw Scatter Plot
# 1. specify x and y
# 2. label each state
# 3. add in trend line
scatter <- ggplot(df_temp, aes(x=month, y=temp.f)) +
      geom_jitter(size=0.1, width = 0.15) +
      labs(title = 'Distribution of Temperature Across Cities in USA',
           x = 'Months',
           y = 'Temperature in Fahrenheit',
           caption = 'Temperature data 2017') +
      scale_x_continuous(labels = as.character(df_temp$month),
                         breaks = df_temp$month) +
      theme_bw()
print(scatter)
```



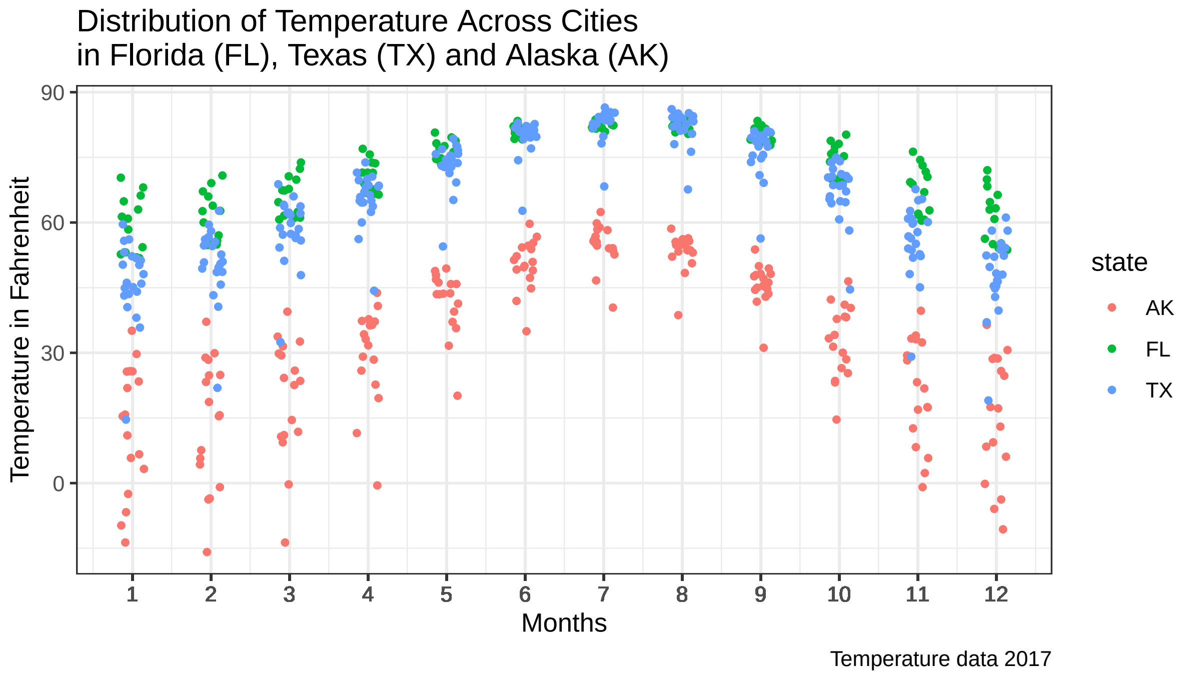
3.1.1.2 Scatter Plot of Temperature and Months for 3 States

Now, we will generate a similar chart as above, but let's select three states, and use different colors for each of the three states.

We can see that there are differences in average temperature across cities in each state in each month, but the different states also have different levels of variations in city temperatures within months.

We want to calculate both mean and standard deviations to capture both differences in averages over the year, as well as differences in how temperature varies within a month over the year.

```
# Control Graph Size
options(repr.plot.width = 5, repr.plot.height = 5)
# First Filter Data
df_temp_txflak <- df_temp %>% filter(state %in% c('AK', 'TX', 'FL'))

# Draw Scatter Plot
# 1. specify x and y
# 2. label each state
# 3. add in trend line
scatter <- ggplot(df_temp_txflak, aes(x=month, y=temp.f,
                                      colour=state)) +
      geom_jitter(size=1, width = 0.15) +
      labs(title = 'Distribution of Temperature Across Cities\nin Florida (FL), Texas (TX) and Alask
           x = 'Months',
           y = 'Temperature in Fahrenheit',
           caption = 'Temperature data 2017') +
      scale_x_continuous(labels = as.character(df_temp$month),
                         breaks = df_temp$month) +
      theme_bw()
print(scatter)
```



3.1.1.3 Mean and Standard Deviation Within Month Acorss USA

We can calculate the average temperature, as well as the standard deviation of temperature, in each month across cities in the USA. Let's show what these are using dplyr, and let's graph them out.

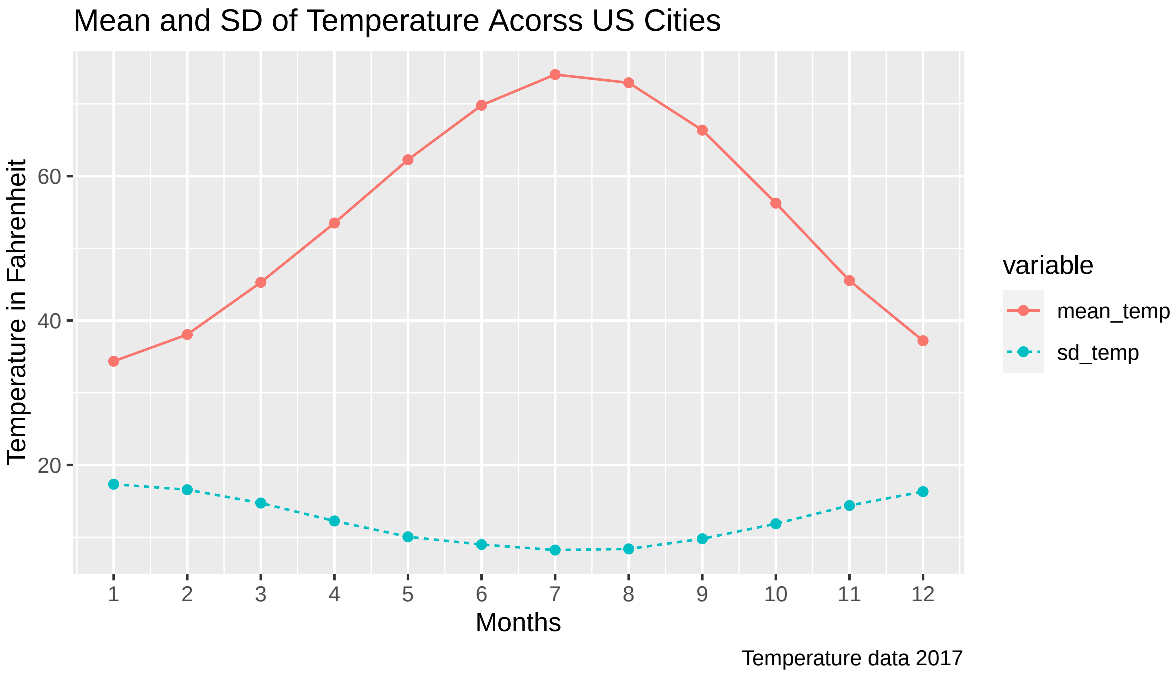
It's pretty amazing what mean, and standard deviation can do for us. We started with a dataset with many many observations, many many temperatures. Now with just 24 numbers below, we have created a way to summarize the large set of observations concisely. Twelve numbers for means for the 12 months, and 12 numbers for the standard deviations in 12 months.

This is like flying in the sky and taking a snapshot of the ground below from thousands of miles up.

The exciting thing here is, which statistics should we generate to adequately summarize what is going on on the ground within all the data observations? In this case here, if we show the mean, it informatively indicates that temperature is hotter during the summer, but it does not show the tightening of the temperature distribution during the summer months that we see in the scatter plot above. Adding standard deviation to our summary statistics, however, allows us also to see that as well.

```
# Show mean and standard deviation in tabular form
df_temp_mth_summ <- df_temp %>%
    group_by(month) %>%
    summarise(mean_temp = mean(temp.f), sd_temp = sd(temp.f))
```

```
# Control Graph Size
options(repr.plot.width = 5, repr.plot.height = 4)
# Show mean and standard deviation in graphical form
# We will gather the data first, it is an essential reshaping command
lineplot <- df_temp_mth_summ %>%
    gather(variable, value, -month) %>%
    ggplot(aes(x=month, y=value, colour=variable, linetype=variable)) +
        geom_line() +
        geom_point() +
        labs(title = 'Mean and SD of Temperature Acorss US Cities',
             x = 'Months',
             y = 'Temperature in Fahrenheit',
             caption = 'Temperature data 2017') +
        scale_x_continuous(labels = as.character(df_temp_mth_summ$month),
                           breaks = df_temp_mth_summ$month)
print(lineplot)
```



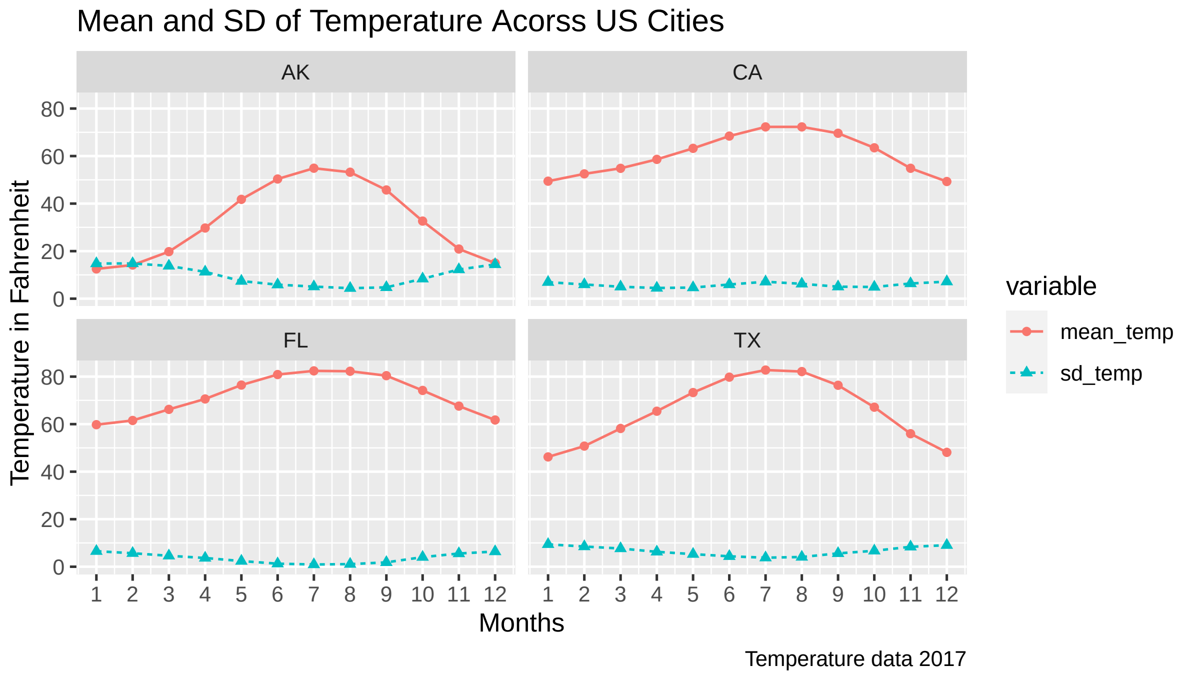
3.1.1.4 Mean and Standard Deviation Within Month Acorss States in USA

We have various states, how do these mean and sd charts vary across the big states that we have, where there are numerous cities in each state?

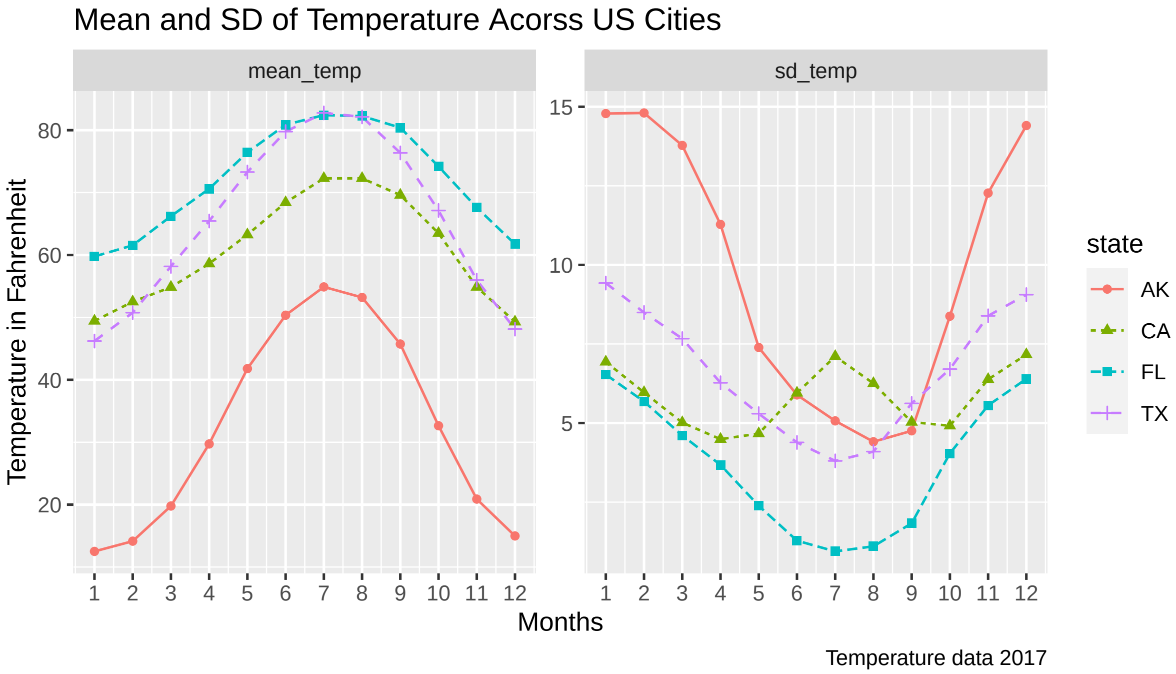
Let's generate some state-specific charts, using very simple commands below, and see how fascinating the United States is.

Specifically, we will have two charts: 1. the first chart has 4 subplots for each state showing the mean and sd for each state across months 2. the second chart has 2 subplots, showing inside each four lines for the four states.

```
# Control Graph Size
options(repr.plot.width = 6, repr.plot.height = 6)
# Show mean and standard deviation in graphical form
# We start from the dataset:
# 1. select a subset of states we want
# 2. group by state and month to generate mean and sd
# 3. reshape data with gather
# 4. generate line plots, state by state

lineplot <- df_temp %>%
    filter(state %in% c('AK', 'CA', 'FL', 'TX')) %>%
    group_by(state, month) %>%
    summarise(mean_temp = mean(temp.f), sd_temp = sd(temp.f)) %>%
    gather(variable, value, -month, -state) %>%
    ggplot(aes(x=month, y=value,
               colour=variable, linetype=variable, shape=variable)) +
        facet_wrap( ~ state) +
        geom_line() +
        geom_point() +
        labs(title = 'Mean and SD of Temperature Acorss US Cities',
             x = 'Months',
             y = 'Temperature in Fahrenheit',
             caption = 'Temperature data 2017') +
        scale_x_continuous(labels = as.character(df_temp_mth_summ$month),
                           breaks = df_temp_mth_summ$month)
print(lineplot)
```



```
# Control Graph Size
options(repr.plot.width = 6, repr.plot.height = 4)
# Show mean and standard deviation in graphical form
# We start from the dataset:
# 1. select a subset of states we want
# 2. group by state and month to generate mean and sd
# 3. reshape data with gather
# 4. generate line plots, state by state

lineplot <- df_temp %>%
    filter(state %in% c('AK', 'CA', 'FL', 'TX')) %>%
    group_by(state, month) %>%
    summarise(mean_temp = mean(temp.f), sd_temp = sd(temp.f)) %>%
    gather(variable, value, -month, -state) %>%
    ggplot(aes(x=month, y=value,
               colour=state, linetype=state, shape=state)) +
        facet_wrap( ~ variable, scales="free_y") +
        geom_line() +
        geom_point() +
        labs(title = 'Mean and SD of Temperature Acorss US Cities',
             x = 'Months',
             y = 'Temperature in Fahrenheit',
             caption = 'Temperature data 2017') +
        scale_x_continuous(labels = as.character(df_temp_mth_summ$month),
                           breaks = df_temp_mth_summ$month)
print(lineplot)
```



3.2 Coefficient of Variation and Correlation

> Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

We have various tools at our disposal to summarize variables and the relationship between variables. Imagine that we have multiple toolboxes. This is the first one. There are two levels to this toolbox. Three Basic Tools:

1. (sample) Mean of X (or Y)
2. (sample) Standard Deviation of X (or Y)
3. (sample) Covariance of X and Y

Additionally, we have two tools that combine the tools from the first level:

1. Coefficient of Variation = (Standard Deviation)/(Mean)
2. Correlation = (Covariance of X and Y)/((Standard Deviation of X)*(Standard Deviation of Y))

The tools on the second level rescale the standard deviation and covariance statistics.

3.2.1 Education and Wage

The dataset, *EPIStateEduWage2017.csv*, can be downloaded here.

Two variables:

1. Fraction of individual with college degree in a state
   - this is in Fraction units, the minimum is 0.00, the maximum is 100 percent, which is 1.00
2. Average hourly salary in the state
   - this is in Dollar units

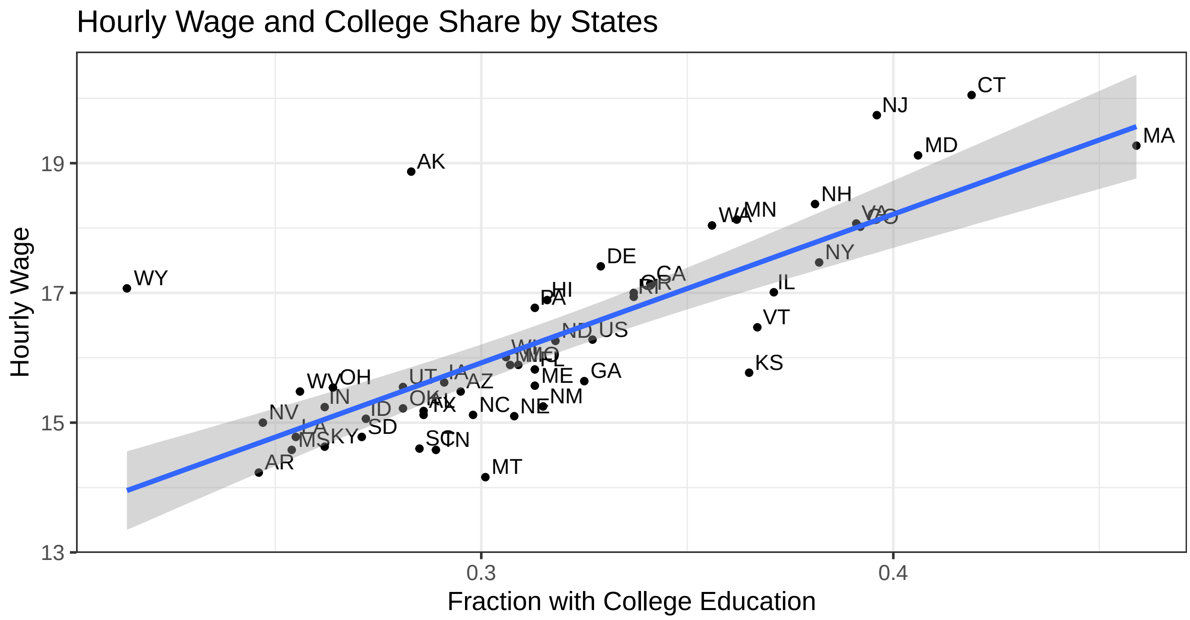
```
# Load in Data Tools
# For Reading/Loading Data
library(readr)
library(tibble)
library(dplyr)
library(ggplot2)
# Load in Data
df_wgedu <- read_csv('data/EPIStateEduWage2017.csv')
```

3.2.1.1 A Scatter Plot

We can Visualize the Data with a Scatter Plot. There seems to be a positive relationship between the share of individuals in a state with a college education, and the average hourly salary in that state.

While most states are along the trend line, we have some states, like WY, that are outliers. WY has a high hourly salary but low share with college education.

```
# Control Graph Size
options(repr.plot.width = 5, repr.plot.height = 5)
# Draw Scatter Plot
# 1. specify x and y
# 2. label each state
# 3. add in trend line
scatter <- ggplot(df_wgedu, aes(x=Share.College.Edu, y=Hourly.Salary)) +
      geom_point(size=1) +
      geom_text(aes(label=State), size=3, hjust=-.2, vjust=-.2) +
      geom_smooth(method=lm) +
      labs(title = 'Hourly Wage and College Share by States',
           x = 'Fraction with College Education',
           y = 'Hourly Wage',
           caption = 'Economic Policy Institute\n www.epi.org/data/') +
      theme_bw()
print(scatter)
```



Economic Policy Institute
www.epi.org/data/

3.2.1.2 Standard Deviations and Coefficient of Variation

The two variables above are in different units. We first calculate the mean, standard deviation, and covariance. With just these, it is hard to compare the standard deviation of the two variables, which are on different scales.

The sample standard deviations for the two variables are: 0.051 and 1.51, in fraction and dollar units. Can we say the hourly salary has a larger standard deviation? But it is just a different scale. 1.51 is a large number, but that does not mean that variable has greater variation than the fraction with college education variable.

Converting the Statistics to Coefficient of Variations, now we have: 0.16 and 0.09. Because of the division, these are both in fraction units–standard deviations as a fraction of the mean. Now these are more comparable.

```r
# We can compute the three basic statistics
stats.msdv <- list(
              # Mean, SD and Var for the College Share variable
              Shr.Coll.Mean = mean(df_wgedu$Share.College.Edu),
              Shr.Coll.Std = sd(df_wgedu$Share.College.Edu),
              Shr.Coll.Var = var(df_wgedu$Share.College.Edu),

              # Mean, SD and Var for the Hourly Wage Variable
              Hr.Wage.Mean = mean(df_wgedu$Hourly.Salary),
              Hr.Wage.Std = sd(df_wgedu$Hourly.Salary),
              Hr.Wage.Var = var(df_wgedu$Hourly.Salary)
              )

# We can compute the three basic statistics
stats.coefvari <- list(
              # Coefficient of Variation
              Shr.Coll.Coef.Variation = (stats.msdv$Shr.Coll.Std)/(stats.msdv$Shr.Coll.Mean),
              Hr.Wage.Coef.Variation = (stats.msdv$Hr.Wage.Std)/(stats.msdv$Hr.Wage.Mean)
              )

# Let's Print the Statistics we Computed
as_tibble(stats.msdv)
as_tibble(stats.coefvari)
```

3.2.1.3 Covariance and Correlation

For covariance, hard to tell whether it is large or small. To make comparisons possible, we calculate the coefficient of variations and correlation statistics.

The covariance we get is positive: 0.06, but is this actually large positive relationship? 0.06 seems like a small number.

Rescaling covariance to correlation, the correlation between the two variables is: 0.78. Since the correlation of two variable is below -1 and $+1$, we can now say actually the two variables are very positively related. A higher share of individuals with a college education is strongly positively correlated with a higher hourly salary.

```
# We can compute the three basic statistics
states.covcor <- list(
              # Covariance between the two variables
              Shr.Wage.Cov = cov(df_wgedu$Hourly.Salary,
                                 df_wgedu$Share.College.Edu),
              # Correlation
              Shr.Wage.Cor = cor(df_wgedu$Hourly.Salary, df_wgedu$Share.College.Edu),
              Shr.Wage.Cor.Formula = (cov(df_wgedu$Hourly.Salary, df_wgedu$Share.College.Edu)
                                      /(stats.msdv$Shr.Coll.Std*stats.msdv$Hr.Wage.Std))
              )

# Let's Print the Statistics we Computed
as_tibble(states.covcor)
```

Chapter 4

Basics of Probability

4.1 Experimental Outcomes

Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

1. Experiment
   - "We define an experiment as a process that generates well-defined outcomes" (ASWCC P171)
2. Sample Space
   - "The sample space for an experiment is the set of all experimental outcomes" (ASWCC P172)
3. Experimental outcomes
   - "An experimental outcome is also called a sample point to identify it as an element of the sample space." (ASWCC P172)
4. Events
   - "An Event is a collection of Sample Points" (AWSCC P181), could be just one sample point (one experimental outcome)
5. Probability
   - "Probability is a numerical measure of the likelihood that an event will occur." (AWSCC P171)

4.1.1 Sample Space and Probabilities

Sample Space and Experimental Outcomes

We can use the letter S or Ω to denote sample space. Supose we have a set of n experimental outcomes:

$$S = \{E_1, E_2, ..., E_{n-1}, E_n\}$$

We can call S a sample space if:

1. E_i are mutually exclusive:
   - E_i for all i are separate outcomes that do not overlap.
   - Suppose there are multiple people running for the Presidency, some of these candidates are women, and some of these women are from Texas. If one of the E_i is that a woman becomes the President, and another E_i is that someone from Texas becomes the President, S would not be a sample space, because we could have a woman who is also from Texas win the Presidency.
2. E_i are jointly exhaustive:
   - For the experiment with well-defined outcomes, the E_i in S need to cover all possible outcomes.
   - If you are throwing a six sided dice, there are six possible experimental outcomes, if S only has five of them, it would not be a sample space.

Thinking about the world in terms of sample space is pretty amazing.

Assigning Probability to Experimental Outcomes

We can assign probabilities to events of a sample space. Since experimental outcomes are themselves events as well, we can assign probabilities to each experimental outcome.

For the mutually exclusive and jointly exhaustive experimental outcomes of the sample space, there are two equirements for assigning probabilities: - Each element of the sample space can not have negative probability of happening, and also can not have more than 1 probability of happening, with P denotes probability, we have:

$$0 \leq P(E_i) \leq 1$$

- The probabilities of all the mutually exclusive and jointly exhaustive experimental outcomes in the sample space sum up to 1. For an experimental with n experimental outcomes:

$$\Sigma_{i=1}^{n} P(E_i) = 1$$

```r
# Load Library
library(tidyverse)

# Define a List of Experimental Outcomes
experimental.outcomes.list <- c('Heavy Rain', 'Light Rain', 'No Rain')

# Probabilities on experimental outcomes
experimental.outcome.prob <- c(0.1, 0.2, 0.7)

# Show these in a Tibble
tibble(tomorrow.experimental.outcomes = experimental.outcomes.list,
       experimental.outcome.prob = experimental.outcome.prob)
```

```r
# What could happen tomorrow?
# We live in a probabilistic world, from today's perspective, tomorrow is uncertain
# If we draw tomorrow from a hat, given our possible outcomes
# and the probabilities associated with the outcomes
# what are the possible tomorrows?
number.of.tomorrow.to.draw = 20
tomorrow.weather.draws <- sample(experimental.outcomes.list,
                                 size = number.of.tomorrow.to.draw,
                                 replace = TRUE,
                                 prob = experimental.outcome.prob)

# A little tibble to show results
# There are only three unique tomorrows, each of three weather outcomes
# could happen, but the chance of each happening differs by the probabilities
# we set in experimental.outcome.prob
tibble(which.tomorrow = paste0('tomorrow:', 1:number.of.tomorrow.to.draw),
       tomorrow.weather = tomorrow.weather.draws)
```

4.1.2 Union and Intersection and Complements

Definitions:

1. Complement of Event A:
   - "Given an event A, the complement of A is defined to be the event consisting of all sample points that are not in A. The complement of A is denoted by A^c." (AWSCC P185)
2. The Union of Events A and B:
   - "The union of A and B is the event containing all sample points belonging to A or B or both. The union is denoted by $A \cup B$." (AWSCC P186)
3. The Intersection of Events A and B:
   - "Given two events A and B, the intersection of A and B is the event containing the sample points belonging to both A and B. The intersection is denoted by $A \cap B$." (AWSCC P187)

Probabilities for Complements and Union

The Probabilities of Complements add up to 1:

$$P(A) + P(A^c) = 1$$

The **Addition Law**:

$$P(A \cup B) = P(A) + P(B) - P(A \cap B)$$

If two events A and B are mutually exclusive, which means they do not share any experimental outcomes (sample points), then: $P(A \cap B) = 0$, and $P(A \cup B) = P(A) + P(B)$.

The **Multiplication Law for Indepedent Events**:

$$P(A \cap B) = P(A) \cdot P(B)$$

If the probability of event A happening does not change the probability of event B happening, and vice-versa. The two events are independent. Below we arrive this formulation from conditional probability.

4.1.3 Conditional Probability

We use a straight line | to denote conditional probability. Given A happens, what is the probability of B happening?

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)}$$

This says the probability of A happening given that B happens is equal to the ratio of the probability that both A and B happen divided by the probability of B happening.

The formula also means that the probability that both A and B happens is equal to the probability that B happens times the probability that A happens conditional on B happening:

$$P(A \cap B) = P(A \mid B) \cdot P(B)$$

If A and B are independent, that means the probability of A happening does not change whether B happens or not, then, $P(A \mid B) = P(A)$, and:

$$\text{If A and B are independent: } P(A \cap B) = P(A) \cdot P(B)$$

This is what we wrote down earlier as well.

4.2 Sample Space and Probability Examples

> Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

In *Sample Space, Experimental Outcomes, Events, Probabilities*, we discussed various definitions. Here are some examples.

- Experiment: Throwing a Quarter
- Experimental outcomes: lands on Heads or lands on Tails
- Sample Space: $S = \{H, T\}$
- Event: There are only two experimental outcomes.
- Probability:
  - $1 \geq P(H) \geq 0$
  - $1 \geq P(T) \geq 0$
  - $P(H) + P(T) = 1$

4.2.1 Presidential Election

After various primaries there are four major candidates who could become president. Voting takes place in half a year. Over this year, many events could happen that would increase or decrease the support for candidates. After voting, the outcome is certain. But from today's perspective, when voting is one year from now, outcomes are uncertain.

- Experiment: Presidential Election in One Year
- Experimental outcomes: candidates DT, HC, JS, GJ winning.
- Sample Space: {DT, HC, JS, GJ}
- Event:
  - A woman wins is an event: W $= \{HC, JS\}$
  - A man wins: M $= \{DT, GJ\}$
  - Someone from a small party wins: S $= \{JS, GJ\}$
- Probability:
  - The probability that a woman does not win:

    $$P(\mathrm{W}^c) = 1 - P(\mathrm{W}) = P(\mathrm{M})$$

  - There are no experimental outcomes, given the four that we have, where both woman and man wins, the two events are mutually exclusive:

    $$P(\mathrm{W} \cap \mathrm{M}) = 0$$

  - The probability that a woman wins conditional on a man winnning is zero:

    $$P(\mathrm{W} \mid \mathrm{M}) = \frac{P(\mathrm{W} \cap \mathrm{M})}{P(\mathrm{M})} = \frac{0}{P(DT) + P(GJ)} = 0$$

  - The probability that a woman wins and someone from a small party wins:

    $$P(\mathrm{W} \cap \mathrm{S}) = P(JS)$$

  - The probability that a woman wins conditional on someone from a small party winning:

    $$P(\mathrm{W} \mid \mathrm{S}) = \frac{P(\mathrm{W} \cap \mathrm{S})}{P(\mathrm{S})} = \frac{P(JS)}{P(JS) + P(GJ)}$$

  - Either a women wins or someone from a small party wins:

    $$P(\mathrm{W} \cup \mathrm{S}) = P(\mathrm{W}) + P(\mathrm{S}) - P(\mathrm{W} \cap \mathrm{S}) = P(HC) + P(JS) + P(JS) + P(GJ) - P(JS) = P(HC) + P(JS) + P(GJ)$$

4.2.2 Throwing a Dice

- Experiment: Throwing a Dice
- Experimental outcomes: lands on one of the six sides
- Sample Space: $S = \{1, 2, 3, 4, 5, 6\}$
- Event: One could define many possible events. The experimental outcome is less than 3 is an event, greater than 2 is an event, been odd is an event, etc.

In Throw an Unfair Four Sided Dice, we show what happens when we throw an unfair four sided dice many times. In the example below, we will throw an unfair dice, where the probability of landing on the side with 1 is 50 percent, and the chance of landing on each successive side is half of the chance of landing on the previous side. This is a dice weighted towards the smaller numbers.

See the table below for the true probabilities of the unfiar dice.

```r
# Load Library
library(tidyverse)

# Define a List of Experimental Outcomes
experimental.outcomes.list <- c(1,2,3,4,5,6)
```

```
# Probabilities on experimental outcomes
# Suppose dice is weighted towards 1
fracbase <- 0.50
experimental.outcome.prob <- c((1-fracbase)^0*fracbase,
                               (1-fracbase)^1*fracbase,
                               (1-fracbase)^2*fracbase,
                               (1-fracbase)^3*fracbase,
                               (1-fracbase)^4*fracbase,
                               (1-fracbase)^5)

# Show these in a Tibble
dice.true.prob <- tibble(dice.outcomes.list = experimental.outcomes.list,
                         dice.true.prob = experimental.outcome.prob)
dice.true.prob
```

4.2.2.1 Throw the Dice 10 Times

We throw the dice 10 times, each time, get one of the six possible experimental outcomes, the chance of getting these outcomes are determined by the true probabilities of the unfair dice. If we throw 10 times, that does not mean we will get exact side 1 five times. Side 1 having proability of 50 percent means for each throw there is a 50 percent chance of getting side 1. But each throw is independent of other throws, so we could just by chance end up with no side 1 after ten throws, although the chance of that happening is low. How many times did we get each side in the ten random tosses below?

```
# What could happen tomorrow?
# We live in a probabilistic world, drawing future from a hat
# If we draw 20 times, what happens in the future?
# It's pretty amazing, we get to see the future!
number.of.futures.to.draw = 10
future.dice.draws <- sample(experimental.outcomes.list,
                            size = number.of.futures.to.draw,
                            replace = TRUE,
                            prob = experimental.outcome.prob)

# A little tibble to show results
tibble(which.future.dice = paste0('dice draws:', 1:number.of.futures.to.draw),
       dice.draws = future.dice.draws)
```

4.2.2.2 Redefining Sample Space for Throwing a Dice

Following our definition for sample space above, we can actually define an alternative sample spaces, for example:

- Experiment: Throwing a Dice
- Experimental outcomes: lands on greater or equal to 4, lands on less or equal to 3
- Sample Space: $S = \{\text{'less equal to 3'}, \text{'greater equal to 4'}\}$

This is also a sample space with two sample points (or experimental outcomes). Following our definitions earlier, these two experimental outcomes are mutually exclusive and jointly exhaustive. We could also call these two experimental outcomes here two different events of the sample space with the six sample points.

4.2.3 Two Basketball Games

- Experiment: One team plays two games
- Experimental outcomes:
  1. E_1: win first, win second
  2. E_2: win first, lose second
  3. E_3: lose first, win second
  4. E_4: lose first, lose second

- Sample Space: $S = \{E_1, E_2, E_3, E_4\}$
- Events:
  - no wins: $\text{W0} = \{E_4\}$
  - win twice: $\text{W2} = \{E_1\}$
  - win first game: $\text{WF} = \{E_1, E_2\}$
  - win second game: $\text{WS} = \{E_1, E_3\}$
  - etc.
- Probability:
  - What is the probability of winning the second game and the first game? Using the conditional probability formula, it is the probability of winning the first game times the probability of winning the second game conditional on winning the first game:

$$P(\text{WS} \cap \text{WF}) = P(\text{WS} \mid \text{WF}) \cdot P(\text{WF}) = P(E_1)$$

    - If winning the first game does not change the probability of winning the second game, then we have:
$$P(E_1) = P(\text{WS} \mid \text{WF}) \cdot P(\text{WF}) = P(\text{WS}) \cdot P(\text{WF})$$
    - If the chance of winning the first and second game are the same, $P(\text{WS}) = P(\text{WF}) = p$, then we have:
$$P(E_1) = P(\text{WS}) \cdot P(\text{WF}) = p \cdot p = p^2$$
  - The probability of winning at least one game:

$$P(\text{WS} \cup \text{WF}) = P(\text{WS}) + P(\text{WF}) - P(\text{WS} \cap \text{WF}) = P(E_1) + P(E_2) + P(E_3)$$

4.2.3.1 Redefining Sample Space for two Basketball Game

Following our definition for sample space above, we can actually define an alternative sample spaces using these events:

- Experiment: One team plays two games
- Experimental outcomes: Win 0, 1, or 2 times
- Sample Space: $S = \{\text{'0 Win'}, \text{'1 Win'}, \text{'2 Wins'}\}$

This is also a sample space with three sample points (or experimental outcomes). Following our definitions earlier, these three experimental outcomes are mutually exclusive and jointly exhaustive. We could also call these three experimental outcomes here three different events of the sample space with the four paths of game play experimental outcomes (sample points).

4.3 Sample Average

> Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

In Sample Space, Experimental Outcomes, Events, Probabilities, we discussed various definitions. We went over various examples in Examples of Sample Space and Probabilities. Here we look at what happens if we throw a four sided dice many times.

4.3.1 An Unfair Dice

Throwing a Dice:

- Experiment: Throwing a Four Sided Dice
- Experimental outcomes: lands on one of the four sides
- Sample Space: $S = \{1, 2, 3, 4\}$

In the example below, we will throw an unfair dice, where the probability of landing on the side with 1 is 60 percent, and the chance of landing on each successive side is 60 percent of the chance of landing on the previous side. This is a dice weighted towards the smaller numbers.

See the table below for the true probabilities of the unfair dice.

```r
# Load Library
library(tidyverse)

# Define a List of Experimental Outcomes
experimental.outcomes.list <- c(1,2,3,4)

# Probabilities on experimental outcomes
# Suppose dice is weighted towards 1
fracbase <- 0.6
experimental.outcome.prob <- c((1-fracbase)^0*fracbase,
                               (1-fracbase)^1*fracbase,
                               (1-fracbase)^2*fracbase,
                               (1-fracbase)^3)

# Show these in a Tibble
dice.true.prob <- tibble(dice.outcomes.list = experimental.outcomes.list,
                         dice.true.prob = experimental.outcome.prob)
dice.true.prob
```

4.3.1.1 Throw the Dice 5 Times

We throw the dice 5 times, each time, get one of the four possible experimental outcomes, the chance of getting these outcomes are determined by the true probabilities of the unfair dice.

```r
# What could happen tomorrow?
# We live in a probabilistic world, drawing future from a hat
# If we draw 5 times, what happens in the future?
# It's pretty amazing, we get to see the future!
number.of.futures.to.draw = 5
future.dice.draws <- sample(experimental.outcomes.list,
                                 size = number.of.futures.to.draw,
                                 replace = TRUE,
                                 prob = experimental.outcome.prob)

# A little tibble to show results
tibble(which.future.dice = paste0('dice draws:', 1:number.of.futures.to.draw),
       dice.draws = future.dice.draws)
```

4.3.1.2 Throw the dice 50, 5000 and 500,000 times

If we throw the dice 50 times, 5000 times, 500,000 times, what will happen?

For each group of experiments, we can aggregate the *empirical distribution* of the four sides. The more times we throw the dice, the closer our empirical distribution gets to the true distribution. We can see the result from the table below.

To do this, we first write a function, then we lapply to invoke the function multiple times.

```r
# Function to Make Many Draws
future.draws <- function(number.of.futures.to.draw, select.dice.draws=FALSE) {
    # Number.of.futures.to.draw = 500
    # Future Draws

    dice.draws <- sample(experimental.outcomes.list,
                         size = number.of.futures.to.draw,
                         replace = TRUE,
                         prob = experimental.outcome.prob)

    # Empirical Distribution Name
    sample.frac.var <- paste0('sample.frac.n', sprintf("%d", number.of.futures.to.draw))
    # Group Futures
    group.fracs <- tibble(dice.draw = dice.draws) %>%
                    group_by(dice.draw) %>%
                    summarise(freq = n()) %>%
                    mutate(!!sample.frac.var :=
                           as.numeric(sprintf("%0.5f", freq / sum(freq)))) %>%
                    arrange(dice.draw) %>%
                    select(dice.draw, !!sample.frac.var)

    # Whether to includ dice.draws categorical
    if (select.dice.draws){
#         group.fracs <- group.fracs
    } else {
        group.fracs <- group.fracs %>% select(!!sample.frac.var)
    }

    # Return
    return(group.fracs)
}
```

```
# Draw future 10, 100, 1000, 10000, 100000 times
# How many times we get 1,2,3,4?
number.of.futures.to.draw.list = c(1000, 5000, 500000)

# Apply function
bind_cols(dice.true.prob, lapply(number.of.futures.to.draw.list, future.draws))
```

4.3.1.3 Throw Four Sided Dice Different Number of Times, and Melt Data

Using the function we created above, we can draw a graph that shows what happens to the empirical distribution of four dice sides as we increase the number of draws.

```
# Generate Data
# Log Space Draws of Outcomes
number.future.logspace <- floor(exp(log(10)*seq(log10(100),log10(1000000), length.out=500)))

# lapply, generating a list of dataframes, then merge together
draw.outcomes <- lapply(number.future.logspace,
                        future.draws, select.dice.draws=TRUE) %>%
                        reduce(full_join, by = 'dice.draw') %>%
                        mutate_all(funs(replace_na(.,0)))

# Melt Data
draw.outcomes.long <- draw.outcomes %>%
                        gather(variable, value, -dice.draw) %>%
                        dplyr::mutate(draw.count =
                                      as.numeric(str_extract(variable, "[^.n]+$"))) %>%
                        select(dice.draw, draw.count, value)

# 1 to 6 are categorical factors
draw.outcomes.long$dice.draw <- paste0('dice side = ', draw.outcomes.long$dice.draw)
draw.outcomes.long$dice.draw <- factor(draw.outcomes.long$dice.draw)

# Show Melt Table
head(draw.outcomes.long, n=10)
tail(draw.outcomes.long, n=10)
```

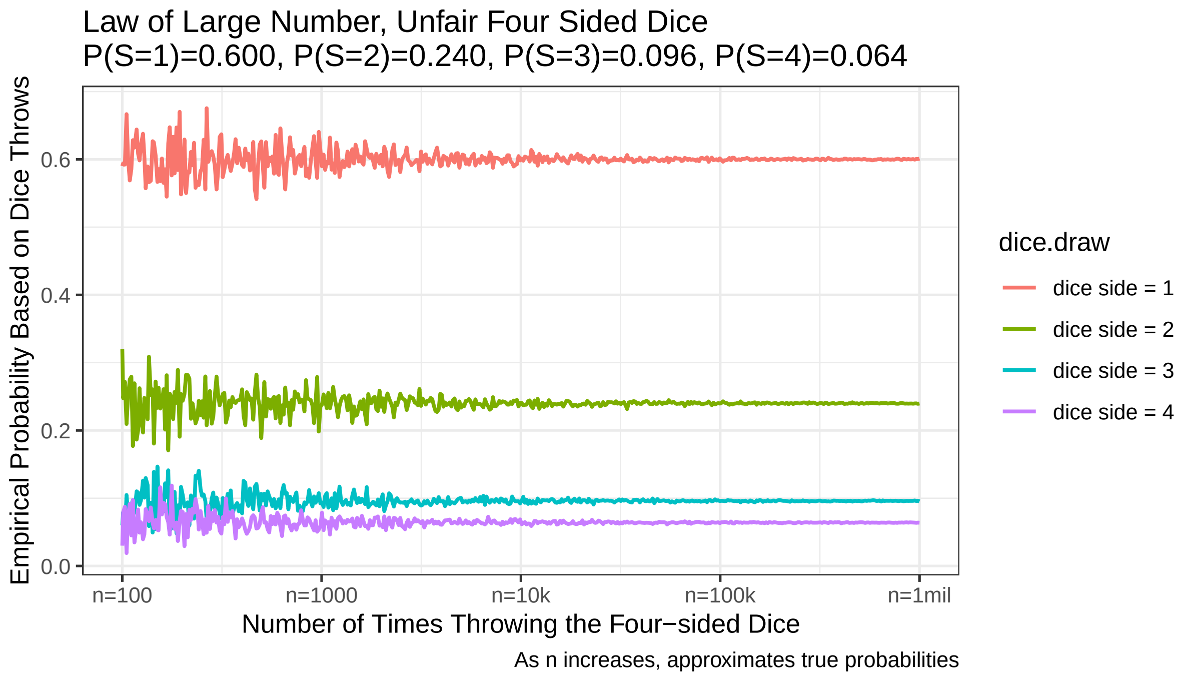
Graphically, What happens when the number of dice throws increase? A crucial thing to understand about probability is that we are not saying if you throw 10 dice, there will be exactly 6 dice out of the 10 that will land on side=1 (given 60 percent probability of landing on side 1).

What we are saying is that, given that each dice throw is independent, if we throw the dice many many times, the empirical distribution of dice throws will approximate the actual true probability of landing on each of the four sides of the dice.

The graph between demonstrates this. The x-axis is in log-scale. We start with 10 throws, and end with 1 million throws. The Y-axis is the empirical probability, with 0.1=10 percent. We have four colors for each of the four sides.

We can see that the empirical probability based on actual dice throws converges to the true probability as we increase the number of dice throws.

```r
# Graph
# Control Graph Size
options(repr.plot.width = 6, repr.plot.height = 4)

# x-labels
x.labels <- c('n=100', 'n=1000', 'n=10k', 'n=100k', 'n=1mil')
x.breaks <- c(100, 1000, 10000, 100000, 1000000)

# title line 2
title_line2 <- sprintf("P(S=1)=%0.3f, P(S=2)=%0.3f, P(S=3)=%0.3f, P(S=4)=%0.3f",
                   experimental.outcome.prob[1], experimental.outcome.prob[2],
                   experimental.outcome.prob[3], experimental.outcome.prob[4])

# Graph Results--Draw
line.plot <- draw.outcomes.long %>%
  ggplot(aes(x=draw.count, y=value,
             group=dice.draw,
             colour=dice.draw)) +
  geom_line(size=0.75) +
  labs(title = paste0('Law of Large Number, Unfair Four Sided Dice\n', title_line2),
       x = 'Number of Times Throwing the Four-sided Dice',
       y = 'Empirical Probability Based on Dice Throws',
       caption = 'As n increases, approximates true probabilities') +
  scale_x_continuous(trans='log10', labels = x.labels, breaks = x.breaks) +
  theme_bw()

print(line.plot)
```



4.4 Multiple-Step Experiment

Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

4.4.1 Playing the Lottery Three times

In *Sample Space, Experimental Outcomes, Events, Probabilities*, we discussed various definitions. We studied various examples in *Examples of Sample Space and Probabilities*.

In the example below, we have a multiple-step experiment: "If an experiment can be describe as a sequence of n steps with m possible outcomes on each step, then the total number of experimental outcomes is m^n." (ASWCC P172) You can use letters other than n and m as well.

Key questions:

- What are the possible experimental outcomes?
- What are the probabilities associated with each of the experimental outcomes?

We are thinking about the problem from day zero perspective. We could play the lottery three times, on day one, day two and day three.

Suppose you can either win or lose when you play the lottery. Each lottery ticket costs the same. You will buy 1 lottery ticket on each day for three days straight. The chance of winning is the same for each lottery ticket (perhaps the number of people buying lottery is the same every day). You see at the end of each day whether you won or lost.

The question is, what are the possible outcomes after three days? You are standing from the perspective of today, looking forward to what could happen tomorrow, two days from now, three days from now.

4.4.1.1 Assumptions and Symbols

Assumptions

Based on the descriptions above, we can write down the following assumptions we are making, which are also the assumptions for the binomial distribution:

1. There are n numbers of identical lottery plays
2. The outcomes of each lottery are always win or lose, just these two possible outcomes.

3. The chance of winning each time, p_{win}, is the same during each play. The chance of losing is the same as well, $1 - p_{\text{win}}$.
4. Whether you won or not before does not impact whether you will win in the future.

Symbols we will use

1. n: denotes the number of lottery plays (number of trials)
2. x: denotes the number of times you win, at most $x = n$, at least $x = 0$, the number of losses is $n - x$
3. p: the probability of winning, the probability of losing is therefore $1 - p$.

The Tree Structure with Probability Written on it, *Notations*:

- W = Win, L = Lose
- D0 = Day 0, D1 = Day 1, D2 = Day 2

4.4.1.2 Play the Lottery Just once

From day zero perspective, by the end of day one, there are two possible experimental outcomes, with different probabilities associated with each outcome.

- D0:
  - D1 W: $P(E_1) = P(W) = p$
  - D1 L: $P(E_2) = P(L) = 1 - p$

There are two experimental outcomes in the sample space:

$$\Sigma_{i=1}^{2}\left(P\left(E_i\right)\right) = p + (1 - p) = 1$$

4.4.1.3 Play the Lottery Twice

From day zero perspective, by the end of day two, there are four possible experimental outcomes, with different probabilities associated with each outcome.

We are assuming that each lottery play is independent, we learned before that:

$$P(A \cap B) = P(A \mid B) \cdot P(B)$$

and when A and B are independent:

$$P(A \cap B) = P(A) \cdot P(B)$$

- D0:
  - D1 W:
    - D2 W: $P(E_1) = P(\text{First Game Win} \cap \text{Second Game Win}) = P(WW) = P(W) \cdot P(W) = p \cdot p$
    - D2 L: $P(E_2) = P(\text{First Game Win} \cap \text{Second Game Lose}) = P(WL) = P(W) \cdot (1 - P(W)) = p \cdot (1 - p)$
  - D1 L:
    - D2 W: $P(E_3) = P(LW) = (1 - p) \cdot p$
    - D2 L: $P(E_4) = P(LL) = (1 - p) \cdot (1 - p)$

There are four experimental outcomes in the sample space:

$$\begin{aligned}
\Sigma_{i=1}^{4}\left(P\left(E_i\right)\right) =& p \cdot p + p \cdot (1 - p) + (1 - p) \cdot p + (1 - p) \cdot (1 - p) && (4.1)\\
=& p \cdot (p + 1 - p) + (1 - p) \cdot (p + 1 - p) && (4.2)\\
=& p + (1 - p) && (4.3)\\
=& 1 && (4.4)\\
& && (4.5)
\end{aligned}$$

4.4.1.4 Play the Lottery Three Times

From day zero perspective, by the end of day three, there are eight possible experimental outcomes, with different probabilities associated with each outcome.

- D0:
  - D1 W:
    - D2 W
      - D3 W: $P(E_1) = P(WWW) = p^3 \cdot (1-p)^0$
      - D3 L: $P(E_2) = P(WWL) = p^2 \cdot (1-p)^1$
    - D2 L:
      - D3 W: $P(E_3) = P(WLW) = p^2 \cdot (1-p)^1$
      - D3 L: $P(E_4) = P(WLL) = p^1 \cdot (1-p)^2$
  - D1 L:
    - D2 W
      - D3 W: $P(E_5) = P(LWW) = p^2 \cdot (1-p)^1$
      - D3 L: $P(E_6) = P(LWL) = p^1 \cdot (1-p)^2$
    - D2 L:
      - D3 W: $P(E_7) = P(LLW) = p^1 \cdot (1-p)^2$
      - D3 L: $P(E_8) = P(LLL) = p^0 \cdot (1-p)^3$

There are eight experimental outcomes in the sample space, and they sum up to one as before:

$$\Sigma_{i=1}^{8} \left(P\left(E_i\right)\right) = 1 \tag{4.6}$$

$$\tag{4.7}$$

4.4.1.5 Probabilities of Events

Generally, after n trials/lotteries, if a particular experimental outcome had x number of wins/successes (which means $n-x$ number of losses/failures), the probability that this particular experimental outcome happen is:

$$p^x \cdot p^{n-x}$$

The probabilities we wrote in the previous examples all follow this rule. Remember that if someone won every time, $n = x$, and we will have $p^x \cdot p^{n-x} = p^x \cdot p^0 = p^x \cdot 1 = p^x$.

So far we have discussed different experimental outcomes with unique sequences of wins and losses. In the 3 lottery example above, we have 8 experimental outcomes. If we are mainly interested in the total number of wins and losses, we can group these 8 experimental outcomes into 4 events.

1. Event win three times: W3 $= \{E_1\}$

$$P(W3) = p^3 \cdot (1-p)^0$$

2. Event win two times: W2 $= \{E_2, E_3, E_5\}$
   - There are three experimental outcomes out of the eight where you win two times. The probability of each of these experimental outcomes happening is: $p^2 \cdot (1-p)^1$, hence:

$$P(W2) = 3 \cdot p^2 \cdot (1-p)^1$$

3. Event win one time: W1 $= \{E_4, E_6, E_7\}$
   - There are three experimental outcomes out of the eight where you win one time. The probability of each of these experimental outcomes happening is: $p^1 \cdot (1-p)^2$, hence:

$$P(W1) = 3 \cdot p^1 \cdot (1-p)^2$$

4. Event win zero time: W0 $= \{E_8\}$

$$P(W0) = p^0 \cdot (1-p)^3$$

These four events are mutually exclusive and jointly exhaustive, we can define a new sample space that includes four experimental outcomes, win zero times, win once, win twice, win three times.

Chapter 5

Discrete Probability Distribution

5.1 Discrete Random Variable and Binomial

Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

Back to Fan's Intro Stat Table of Content

We have been looking at various examples of discrete Random Variables in Sample Space, Experimental Outcomes, Events, Probabilities, Examples of Sample Space and Probabilities, Multiple-Step Experiment: Playing the Lottery Three times, and Throw an Unfair Four Sided Dice.

Now we state this a little bit more formally.

5.1.1 Discrete Random Variable

1. Random Variable
   - "A random variable is a numerical description of the outcome of an experiment." (ASWCC P217)

   $$\text{we could use letter } x \text{ to represent a random variable}$$

2. Discrete Random Variables
   - "A random variable that may assume either a finite number of values or an infinite sequence of values such as $0, 1, 2, ...$ is referred to as a discrete random variable." (ASWCC P217)
3. Discrete Probability Mass Function
   - "For a discrete random variable x, a probability function, denoted by $f(x)$, provides the probability for each value of the random variable." (ASWCC P220)
   - We can think of different values of x as been mapped from an experimental outcome in the sample space. Probability can not be negative, and the sum of the probability of different possible x outcomes sum to 1 (ASWCC P221):

   $$f(x) \geq 0$$

   $$\Sigma f(x) = 1$$

4. Expected Value of a Discrete Random Variable (ASWCC P225)

   $$E(x) = \mu_x = \Sigma x \cdot f(x)$$

5. Variance of a Discrete Random Variable (ASWCC P225)

   $$Var(x) = \sigma_x^2 = \Sigma \left((x - \mu_x)^2 \cdot f(x) \right)$$

Well Known Discrete Probability Distributions

There is a variety of different Discrete Random Variable Probability Mass Functions that are appropriate for analyzing different situations. Think of these as tools, you don't need to memorize the formulas, but you can learn about under what conditions these distributions are useful. Learn about the parameters of these distributions, and use the appropriate Excel or R function.

The simplest distribution is the Discrete Uniform Distribution. The uniform probability mass function is: $f(x) = \frac{1}{n}$, where n is the *the number of values the random variable may assume*, corresponding to experimental outcomes. (ASWCC P222)

We focus on the Binomial Distribution here, a variety of other distributions are related to the binomial distribution:

- Binomial Distribution
- Bernoulli Distribution
- Hypergeometric Distribution
- Beta-Binomial Distribution

5.2 Binomial Experiment

We have already examined Multiple-Step Experiment: Playing the Lottery Three times. Here we re-state the four properties of the Binomial Experiment (ASWCC P240):

1. "The experiment consists of a sequence of n identical trials."
2. "Two outcomes are possible on each trial. We refer to one outcome as a success and the other outcome as a failure."
3. "The probability of a success, denoted by p, does not change from trial to trial. Consequently, the probability of a failure, denoted by $1-p$, does not change from trial to trial."
4. "The trials are independent."

Binomial Sample Space

Note that given that there are n trials, the possible x go from $x=0$ to $x=n$. In another word, if there are three trials, $n=3$, there are four possible experimental outcomes for x.

- Experiment: The binomial experiment with n trials
- Experimental outcomes: $x = 0, x = 1, ..., x = (n-1), x = n$
- Sample Space: $S = \{0, 1, ..., n-1, n\}$

Binomial Probability Mass Funcion

What is the probability of having x success out of n trials, given that there is p chance of success for each trial and the assumptions above? The answer is given by this formula, which is the binomial probability mass function:

$$f(x; n, p) = C_x^n \cdot p^x \cdot (1-p)^{n-x} = \frac{n!}{(n-x)! \cdot x!} \cdot p^x \cdot (1-p)^{n-x}$$

With the binomial experiment, we can now use a formula to assign probabilities. A formula is a piece of machinery that takes inputs and spits out outputs, Thie binomial probability mass function has three inputs, x, n and p. You need to tell R, Excel, or alternative programs what these three numbers are, and the program will spit a probability out for you. n and p are the parameters.

Binomial Expected Value and Variance

For the binomial discrete random variable, it turns out the expected value and variance are:

$$E(x) = n \cdot p$$

$$Var(x) = n \cdot p \cdot (1-p)$$

We can find the expected value and variance by summing over all terms following: $E(x) = \mu_x = \Sigma x \cdot f(x)$ and $Var(x) = \sigma_x^2 = \Sigma\left((x-\mu_x)^2 \cdot f(x)\right)$, and we will always end up with these two results. It is intuitive. The average number of wins given that you play n trials and given that the chance of winning each game is p is $n \cdot p$.

5.2.1 Binomial Example: Larceny

In 2004, in the United State, 18.3 percent of Larceny-Theft were cleared. "Clearance rates are used by various groups as a measure of crimes solved by the police."

5.2.1.1 Chance of x out n Arrested

10 people commit 10 larceny-theft crimes, suppose the situation follows the conditions of the binomial experiment, what is the chance that nobody is arrested? The chance that 1, 2, 3, or all 10 are arrested?

- $n = 10$: 10 crimes, 10 trials
- $p = 0.183$: what is p here? For the police success is clearing a crime.
- x: if $x = 10$, that means all larceny-thefts were cleared, if $x = 0$, this means the police failed to clear all 10 thefts.

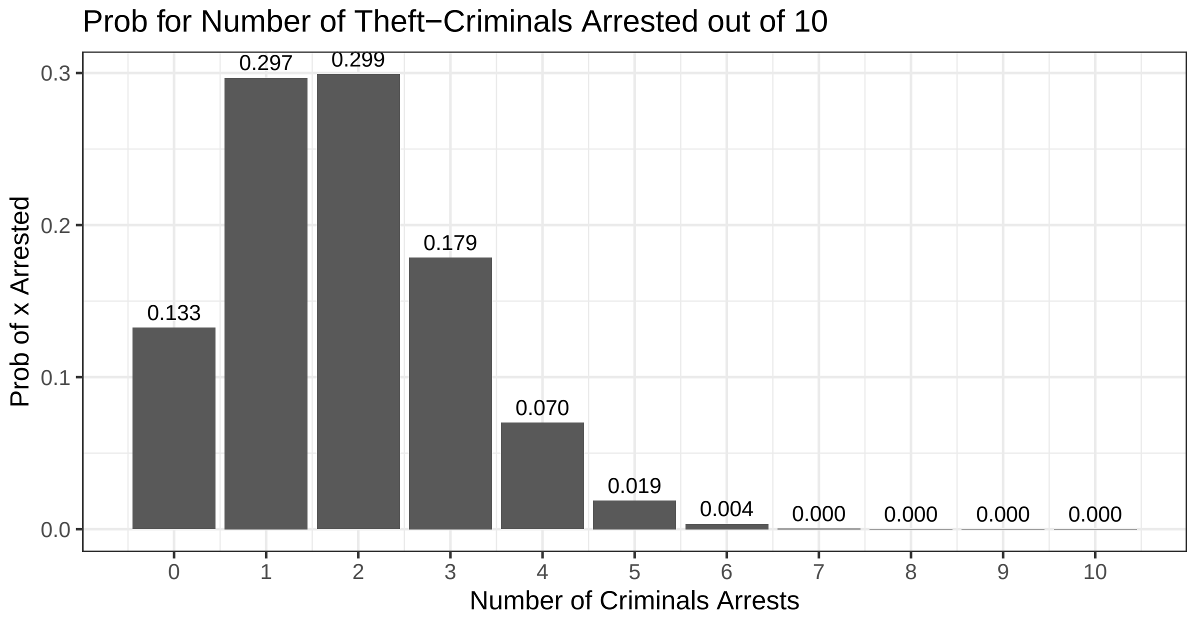
For example, the chance that 2 out of 10 is arrested is:

$$f\left(x=2; n=10, p=0.183\right) = \frac{10!}{(10-2)! \cdot 2!} \cdot 0.183^2 \cdot (1-0.183)^{10-2} = 0.299$$

We can use the r function, *dbinom*, to calculate these probabilites: *dbinom(2,10,0.183)*. Additionally, *pbinom(2,10,0.183)* tells us the cumulative probability that at most 2 out of 10 are arrested.

We do this in the code below. From the graph below, we can see that there is a 13.3% chance that none of the 10 thieves would be arrested, and alomst 0 percent chance that all 10 of then would be arrested. The chance that 6 out of the 10 thiefs would be arrested is less than 1 percent.

```r
# library
library(tidyverse)

# Parameters
n <- 10
p <- 0.183

# A vector of different arrest counts
zero2max <- 0:n

# Probability for different arrest counts
prob_of_arrests <- dbinom(zero2max, n, p)

# Control Graph Size
options(repr.plot.width = 5, repr.plot.height = 4)

# Number of Arrests Probabilities
arrest.prob.mass <- tibble(arrests=zero2max, prob=prob_of_arrests)

# Titles Strings etc
graph.title <- sprintf('Prob for Number of Theft-Criminals Arrested out of %s' ,n)
graph.caption <- sprintf(
    paste0('Assuming the binomial properties apply\n',
           '2004 Larceny/Left USA Clearance Rate = %s'), p)

# Create a Table and Graph it
arrest.graph <- arrest.prob.mass %>%
    ggplot(aes(x=arrests, y=prob, label = sprintf('%0.3f', prob))) +
    geom_bar(stat = 'identity') +
    geom_text(vjust = -0.5, size = 3) +
    labs(title = graph.title,
         x = 'Number of Criminals Arrests',
         y = 'Prob of x Arrested',
         caption = graph.caption) +
    scale_x_continuous(labels = zero2max, breaks = zero2max) +
    theme_bw()

print(arrest.graph)
```



Assuming the binomial properties apply
2004 Larceny/Left USA Clearance Rate = 0.183

5.2.1.2 What is the Chance of Arrest at Least X out of N People

What is the chance that at most 3 out of the 10 thiefs are arrested? That includes the chance that no one is arrested, 1, 2, or 3 people are arrested:

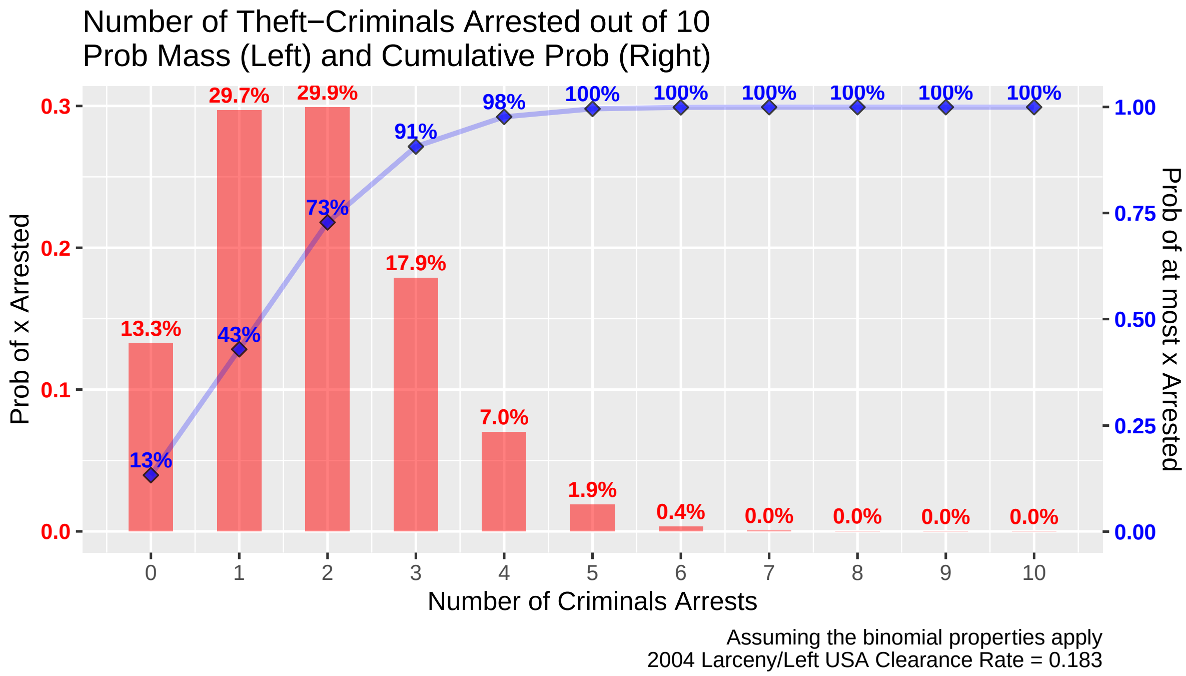
$$p\left(x \le 3; n=10, p=0.183\right) \; = \Sigma_{x=0}^{3}\left(f\left(x; n=10, p=0.183\right)\right) \; = \Sigma_{x=0}^{3}\left(\frac{10!}{(10-x)!\cdot x!}\cdot 0.183^{x}\cdot\left(1-0.183\right)^{10-x}\right) \; = 0.133+$$

Given the low clearance rate, there is a 90 percent chance that at most 3 out of 10 criminals are arrested.

We can graph this out. We will graph the previous graph again but overlay the cumulative probability on top.

```
# Cumulative Probability for different arrest counts, before dbinom, now pbinom
cumulative_prob_of_arrests <- pbinom(zero2max, n, p)

# Data File that Includes Cumulative Probability and Mass
arrest.prob <- tibble(arrests=(zero2max), prob=prob_of_arrests, cum.prob=cumulative_prob_of_arrests)

# Control Graph Size
options(repr.plot.width = 5, repr.plot.height = 4)

# Create a Table and Graph it
#     geom_text(aes(y=prob, label = sprintf('%0.3f', prob)), vjust = -0.5, size = 3) +

axis.sec.ratio <- max(cumulative_prob_of_arrests)/max(prob_of_arrests)
right.axis.color <- 'blue'
left.axis.color <- 'red'

# Probabilities
arrest.graph <- arrest.prob %>%
    ggplot(aes(x=arrests)) +
    geom_bar(aes(y=prob),
             stat='identity', alpha=0.5, width=0.5, fill=left.axis.color) +
    geom_text(aes(y=prob,
```

```r
                  label = paste0(sprintf('%2.1f', prob*100), '%')),
              vjust = -0.5, size = 3, color=left.axis.color, fontface='bold')

# Cumulative Probabilities
arrest.graph <- arrest.graph +
    geom_line(aes(y=cum.prob/axis.sec.ratio),
              alpha=0.25, size=1, color=right.axis.color) +
    geom_point(aes(y=cum.prob/axis.sec.ratio),
               alpha=0.75, size=2, shape=23, fill=right.axis.color) +
    geom_text(aes(y=cum.prob/axis.sec.ratio,
                  label = paste0(sprintf('%2.0f', cum.prob*100), '%')),
              vjust = -0.5, size = 3, color=right.axis.color, fontface='bold')

# Titles Strings etc
graph.title <- sprintf(
    paste0('Number of Theft-Criminals Arrested out of %s\n',
           'Prob Mass (Left) and Cumulative Prob (Right)'), n)
graph.caption <- sprintf(
    paste0('Assuming the binomial properties apply\n',
           '2004 Larceny/Left USA Clearance Rate = %s'), p)
graph.title.x <- 'Number of Criminals Arrests'
graph.title.y.axisleft <- 'Prob of x Arrested'
graph.title.y.axisright <- 'Prob of at most x Arrested'

# Title
arrest.graph <- arrest.graph +
    labs(title = graph.title,
         x = graph.title.x, y = graph.title.y.axisleft,
         caption = graph.caption) +
    scale_y_continuous(sec.axis =
                       sec_axis(~.*axis.sec.ratio, name = graph.title.y.axisright)) +
    scale_x_continuous(labels = zero2max, breaks = zero2max) +
    theme(axis.text.y = element_text(face='bold'),
          axis.text.y.right = element_text(color = right.axis.color),
          axis.text.y.left = element_text(color = left.axis.color))

# Print
print(arrest.graph)
```



Number of Theft−Criminals Arrested out of 10
Prob Mass (Left) and Cumulative Prob (Right)

Assuming the binomial properties apply
2004 Larceny/Left USA Clearance Rate = 0.183

5.2.2 Binomial Example: WWII German Soldier

During WWII, 13.6 million Germans served in the German army, and 4.2 million were killed or missing. This is a death rate of 30.9 percent.

Suppose there are many German towns that each sent 100 soldiers to the front, suppose the binomial properties apply, what is the fraction of solidiers who will return to their towns after the war?

- $n = 100$: suppose 100 solidiers from each German town went to join the German Army during WWII.
- $p = 1 - 0.309 = 0.691$: p is the chance of survival.
- x: if $x = 1$, that means 1 out of 100 survived.

```
# Parameters
n <- 100
p <- 1-0.309

# Generate Data
# A vector of different survival counts
zero2max <- 0:n
# Probability for different survival counts
prob_of_survives <- dbinom(zero2max, n, p)
# Cumulative Probability for different survival counts, before dbinom, now pbinom
cumulative_prob_of_survives <- pbinom(zero2max, n, p)
# Data File that Includes Cumulative Probability and Mass
survive.prob <- tibble(survives=(zero2max), prob=prob_of_survives, cum.prob=cumulative_prob_of_survi
```

5.2.2.1 WWII German Soldier–Graph

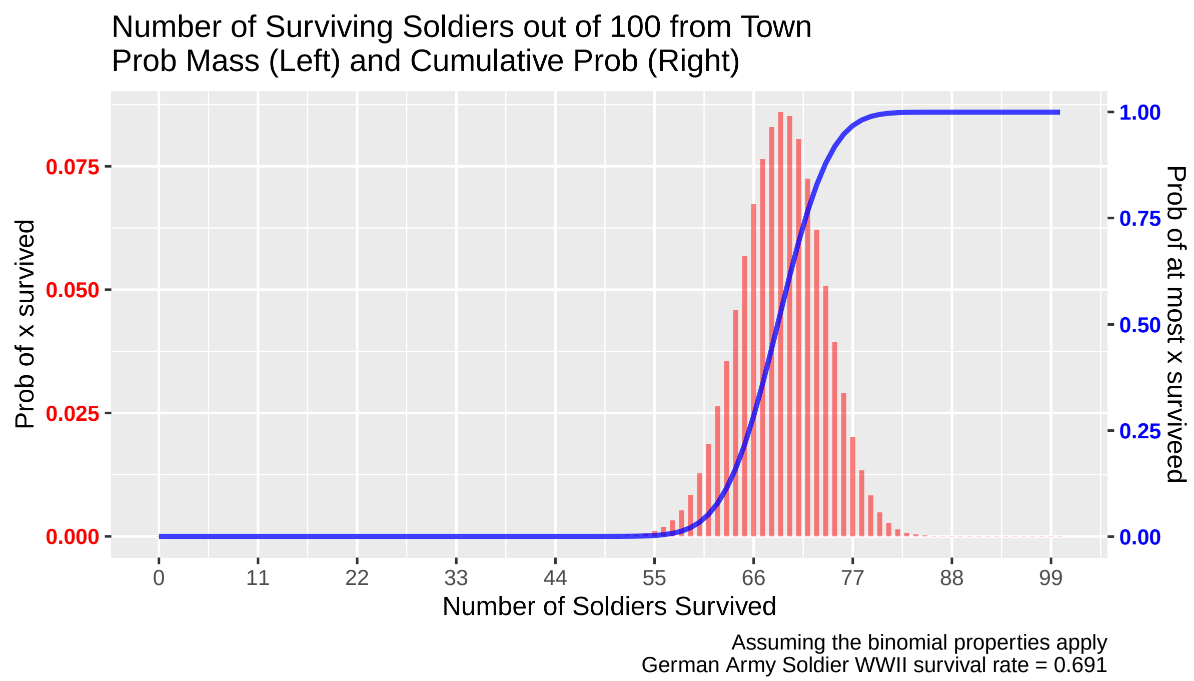
We can see the results graphically as well below. Note that the graph looks normal. This is indeed the case, when n gets larger, the bionomial distribution approximates the normal distribution, with mean $n \cdot p$ and variance $n \cdot p \cdot (1 - p)$.

Again this is given binomial assumptions. Which means in this case that soldiers from each town has equal probability of dying. And the chance of one soldier from a town dying does not change the chance of other soldiers from the same town dying. These are unlikely to be correct assumptions, but maybe they are approximately right.

We can see from the figure below that if the survival distribution follows binomial, less than 1 percent of towns should expect more than 80 out of 100 soldiers to return. And less than 1 percent of towns should expect less than 57 soldiers to return. Hence:

- Given a 30.9 percent death rate, nearly all German towns will have between 57 to 80 soldiers returning from the war out the 100 they sent to join the German army.

```r
# Control Graph Size
options(repr.plot.width = 5, repr.plot.height = 4)
# Two axis colors
axis.sec.ratio <- max(cumulative_prob_of_survives)/max(prob_of_survives)
right.axis.color <- 'blue'
left.axis.color <- 'red'

# Probabilities
survive.graph <- survive.prob %>%
    ggplot(aes(x=survives)) +
    geom_bar(aes(y=prob),
             stat='identity', alpha=0.5, width=0.5, fill=left.axis.color)

# Cumulative Probabilities
survive.graph <- survive.graph +
    geom_line(aes(y=cum.prob/axis.sec.ratio),
              alpha=0.75, size=1, color=right.axis.color)

# Titles Strings etc
graph.title <- sprintf(
    paste0('Number of Surviving Soldiers out of %s from Town\n',
           'Prob Mass (Left) and Cumulative Prob (Right)') ,n)
graph.caption <- sprintf(
    paste0('Assuming the binomial properties apply\n',
           'German Army Soldier WWII survival rate = %s'), p)
graph.title.x <- 'Number of Soldiers Survived'
graph.title.y.axisleft <- 'Prob of x survived'
graph.title.y.axisright <- 'Prob of at most x surviveed'

# Titles etc
survive.graph <- survive.graph +
    labs(title = graph.title,
         x = graph.title.x,
         y = graph.title.y.axisleft,
         caption = graph.caption) +
    scale_y_continuous(sec.axis =
                           sec_axis(~.*axis.sec.ratio, name = graph.title.y.axisright)) +
    scale_x_continuous(labels = zero2max[floor(seq(1,n,length.out=10))],
                           breaks = zero2max[floor(seq(1,n,length.out=10))]) +
    theme(axis.text.y = element_text(face='bold'),
          axis.text.y.right = element_text(color = right.axis.color),
          axis.text.y.left = element_text(color = left.axis.color))

# Print
print(survive.graph)
```



5.2.2.2 WWII German Soldier–Table

We can see from the table of results what is the distribution of the number of soldiers returning to each village in greater detail.

```
# print(survive.prob)
# start_idx <- 81
f_table <- function(start_idx) {

    survive.subset <- round(survive.prob[seq(start_idx, start_idx+20),], 4)
    survive.subset$prob <- paste0(survive.subset$prob*100,
                                  '% chance EXACTLY (n=', survive.subset$survives, ') survived')
    survive.subset$one.minus.cum.prob <- paste0((1-survive.subset$cum.prob)*100,
                                                '% chance AT LEAST (n=', survive.subset$survives, ')
    survive.subset$cum.prob <- paste0(round((survive.subset$cum.prob)*100, 4),
                                      '% chance AT MOST (n=', survive.subset$survives, ') survived')

    return(survive.subset[,2:4])
}
lapply(c(1,21,41,61,81), f_table)
```

```
## [[1]]
##
## [[2]]
##
## [[3]]
##
## [[4]]
##
## [[5]]
```

5.3 Poisson Distribution

Go back to fan’s REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

We looked at the binomial probability distribution Discrete Random Variable and Binomial Experiment. Now we look at the poisson distribution.

Suppose you run a restaurant, maybe you know on average how many guests arrive on a weekday night between 6 and 9, but every night the exact number might be different, what is the distribution of guest arrivals? By chance, there could be 0 guests, there could also be a lot more guest than average potentially. We use the poisson distribution to think about these arrival probabilities.

For us to use the Poisson distribution, the experiment we study need to have these two properties (ASWCC P250):

1. "The probability of an occurrence is the same for any two intervals of equal length."
2. "The occurrence or nonoccurrence in any interval is independent of the occurrence or nonoccurrence in any other interval."

Poisson Sample Space

Guests arrive at your restaurant between 6 and 9 on a weekday night, 0 guest could arrive, 1, 2, 3, ..., 10, 11, 13, ... , 20, 21, 22, ... Unlike in the Binomial case, where we have a maximum number of games that we can win out of n games played, here we don't impose a maximum number of guests that can arrive.

- Experiment: arrivals
- Experimental outcomes: $x = 0, x = 1, ..., ...$, unlike the bionomial, there is no limit here
- Sample Space: $S = \{0, 1, ..., ...\}$ (the support is infinite)

Poisson Probability Mass Funcion

What is the probability of having x arrivals during an interval of time given that the expected (mean) number of arrival in that interval of time is μ? The answer is given by this formula, which is the poisson probability mass function:

$$f(x;\mu) = \frac{\mu^x \cdot e^{-\mu}}{x!} \tag{5.1}$$

$$\tag{5.2}$$

With the poisson experiment, we have this formula to assign probabilities. The formula has two inputs, x and μ. $e = 2.71828$ is not a parameter, it is a fixed number like π, it is a mathematical constant. You need to tell R, Excel, or alternative programs what these two numbers x and μ are, and the program will spit a probability out for you. μ is the parameter.

Poisson Expected Value and Variance

For the Poisson discrete random variable, it turns out the expected value and variance are:

$$E(x) = \mu$$

$$Var(x) = \mu$$

We can check by summing over : $E(x) = \mu_x = \Sigma x \cdot f(x)$ and $Var(x) = \sigma_x^2 = \Sigma\left((x - \mu_x)^2 \cdot f(x)\right)$, and we will always end up with these two results. Note that if we are actually adding up terms, since there is no maximum arrival limit, we have to approximate the summation up to a large number of arrivals.

5.3.1 Poisson Example: Horse-Kicking

Using data from the Prussian Army on "number of soldiers killed by being kicked by a horse each year in each of 14 cavalry corps over a 20-year period", Ladislaus Bortkiewicz showed in 1898 in Gesetz der kleinen Zahlen (The Rule of Small Numbers) that this followed the Poisson distribution. This is one of the most famous examples of the Poisson Distribution.

He found that there was 0.70 deaths per one corps per one year.

Chance of 2 death per year per corp

- $x = 2$: 2 death in a corp in a year by horse-kick

- $\mu = 0.70$

For example, the chance that 2 soldier from a corp in a year die of horse kick:

$$f\left(x=2;\mu=0.7\right)=\frac{0.7^{2}\cdot e^{-0.7}}{2!}=0.122$$

We can use the r function, *dpois*, to calculate these probabilites: *dpois(2, 0.7)*. Additionally, *ppois(2,0.7)* calculates the cumulative probability that at most 2 die from horse-kick in a corp in a year.

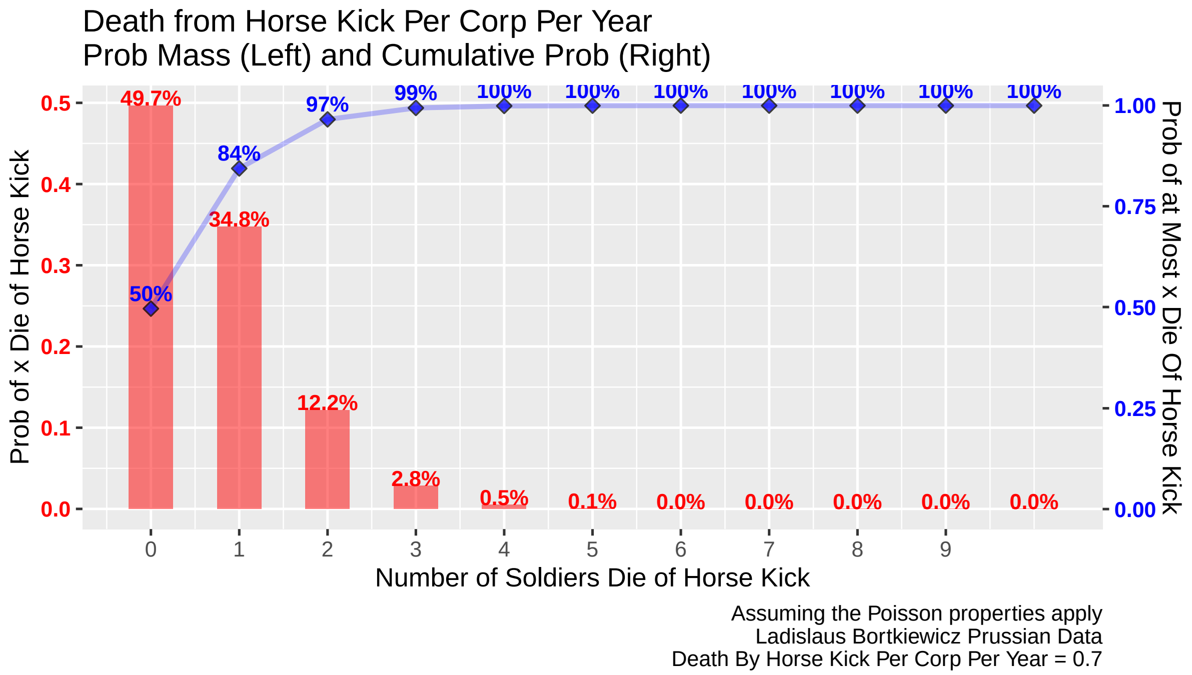
Distributional Graphs Horse Kicking Death Per Corp Per Year

We can see from the results below that given our parameter $\mu = 0.7$, there is almost a 50 percent chance that a corp has no death from horse kick in a year. And the chances that 1, 2, 3 and 4 Prussian soldiers die from horse-kick are 35, 12, 3, and 0.5 percent.

The chance that at least 2 soldiers die from each corp each year is 16 percent.

If you are running the Prussian Army, you would want to know these statistics. You would want to track these statistics overtime and try to improve training etc.

```r
library(tidyverse)

# Parameters
n <- 10
mu <- 0.70

# Generate Data
# A vector of different survival counts
zero2large <- 0:n
# Probability for different survival counts
prob_of_diekicks <- dpois(zero2large, mu)
# Cumulative Probability for different survival counts, before dbinom, now pbinom
cumulative_prob_of_diekicks <- ppois(zero2large, mu)
# Data File that Includes Cumulative Probability and Mass
diekick.prob <- tibble(diekicks=(zero2large), prob=prob_of_diekicks, cum.prob=cumulative_prob_of_die

# Control Graph Size
options(repr.plot.width = 5, repr.plot.height = 4)
# Two axis colors
axis.sec.ratio <- max(cumulative_prob_of_diekicks)/max(prob_of_diekicks)
right.axis.color <- 'blue'
left.axis.color <- 'red'

# Probabilities
diekick.graph <- diekick.prob %>%
    ggplot(aes(x=diekicks)) +
    geom_bar(aes(y=prob),
             stat='identity', alpha=0.5, width=0.5, fill=left.axis.color) +
    geom_text(aes(y=prob,
                  label = paste0(sprintf('%2.1f', prob*100), '%')),
              vjust = -0., size = 3, color=left.axis.color, fontface='bold')

# Cumulative Probabilities
diekick.graph <- diekick.graph +
    geom_line(aes(y=cum.prob/axis.sec.ratio),
              alpha=0.25, size=1, color=right.axis.color) +
    geom_point(aes(y=cum.prob/axis.sec.ratio),
               alpha=0.75, size=2, shape=23, fill=right.axis.color) +
    geom_text(aes(y=cum.prob/axis.sec.ratio,
                  label = paste0(sprintf('%2.0f', cum.prob*100), '%')),
              vjust = -0.5, size = 3, color=right.axis.color, fontface='bold')


# Graph Strings
graph.title <- sprintf(
    paste0('Death from Horse Kick Per Corp Per Year\n',
           'Prob Mass (Left) and Cumulative Prob (Right)'))
graph.caption <- sprintf(
    paste0('Assuming the Poisson properties apply\n',
           'Ladislaus Bortkiewicz Prussian Data\n',
           'Death By Horse Kick Per Corp Per Year = %s'), mu)
graph.title.x <- 'Number of Soldiers Die of Horse Kick'
graph.title.y.axisleft <- 'Prob of x Die of Horse Kick'
graph.title.y.axisright <- 'Prob of at Most x Die Of Horse Kick'

# Graphing
diekick.graph <- diekick.graph +
    labs(title = graph.title,
         x = graph.title.x,
         y = graph.title.y.axisleft,
         caption = graph.caption) +
    scale_y_continuous(sec.axis =
                       sec_axis(~.*axis.sec.ratio,
                                name = graph.title.y.axisright)) +
    scale_x_continuous(labels = zero2large[floor(seq(1,n,length.out=10))],
                       breaks = zero2large[floor(seq(1,n,length.out=10))]) +
    theme(axis.text.y = element_text(face='bold'),
          axis.text.y.right = element_text(color = right.axis.color),
          axis.text.y.left = element_text(color = left.axis.color))

# Print
print(diekick.graph)
```



```
# Tabular
round(as.tibble(diekick.prob), 3)
```

Appendix A

Index and Code Links

A.1 Survey links

1. An In-class Survey: **rmd** | **r** | **pdf** | **html**
   - create a tibble dataset
   - draw 10 random students from 50 and build a survey
   - **r**: *factor() + ifelse()*
   - **dplyr**: *group_by() + mutate() + summarise()*
   - **tibble**: *add_row()*
   - **readr**: *write_csv()*

A.2 Dataset, Tables and Graphs links

1. Opening a Dataset: **rmd** | **r** | **pdf** | **html**
   - Opening a Dataset.
   - **r**: *setwd()*
   - **readr**: *write_csv()*
2. One Variable Graphs and Tables: **rmd** | **r** | **pdf** | **html**
   - Frequency table, bar chart and histogram.
   - R function and lapply to generate graphs/tables for different variables.
   - **r**: *c('word1','word2') + function() + for (ctr in c(1,2)) {} + lapply()*
   - **dplyr**: *group_by() + summarize() + n()*
   - **ggplot**: *geom_bar() + geom_histogram() + labs(title = 'title', caption = 'caption')*
3. Multiple Variables Graphs and Tables: **rmd** | **r** | **pdf** | **html**
   - Two-way frequency table, stacked bar chart annd scatter-plot
   - **r**: *interaction()*
   - **dplyr**: *group_by(var) + summarize(freq = n()) + spread(gender, freq)*
   - **ggplot**: *aes(x,y,fill) + geom_bar(stat='identity', fun.y='mean', position='dodge') + geom_point(size) + geom_text(size,hjust,vjust) + geom_smooth(method=lm) + labs(title,x,y,caption)*

A.3 Summarizing Data links

1. Mean and Standard Deviation: **rmd** | **r** | **pdf** | **html**
   - Mean and standard deviation from a dataset with city-month temperatures.
   - **r**: *dim() + min() + ceiling() + lapply() + vector(mode="character",length) + substring(var, first, last) + func <- function(return(list))*
   - **dplyr**: *mutate() + select() + filter()*
   - **tidyr**: *gather(vara, val, -varb)*
   - **rlang**: *!!sym(str_var_name)*

- **ggplot**: *aes(x, y, colour, linetype, shape) + facet_wrap(~var, scales=‘free_y’) + geom_line() + geom_point() + geom_jitter(size, width) + scale_x_continuous(labels, breaks)*

2. Rescaling Standard Deviation and Covariance: **rmd** | **r** | **pdf** | **html**
   - Scatter-plot of a dataset with state-level wage and education data.
   - Coefficient of variation and standard deviation, correlation and covariance.
   - **r**: *mean() + sd() + var() + cov() + cor()*
   - **ggplot**: *geom_point(size) + geom_text() + geom_smooth()*

A.4 Basics of Probability links

1. Sample Space, Experimental Outcomes, Events, Probabilities: **rmd** | **r** | **pdf** | **html**
   - Sample Space, Experimental Outcomes, Events and Probability.
   - Union, intersection and complements
   - conditional probability
2. Examples of Sample Space and Probabilities: **rmd** | **r** | **pdf** | **html**
   - Throwing a quarter, four candidates for election, six-sided unfair dice, two basketball games
   - **r**: *sample(size, replace, prob)*
3. Throw an Unfair Four Sided Dice: **rmd** | **r** | **pdf** | **html**
   - Throw an unfair dice many times, law of large number.
   - **r**: *head() + tail() + factor() + sample() + as.numeric() + paste0(‘dice=’, var) + sprintf(‘%0.3f’, 1.1234) + sprintf(“P(S=1)=%0.3f, P(S=2)=%0.3f”, 1.1, 1.2345)*
   - **stringr**: *str_extract() + as.numeric(str_extract(variable, “[^.n]+$”))*
   - **dplyr**: *mutate(!!str_mean_var := as.numeric(sprintf(‘%0.5f’, freq / sum(freq))))*
   - **ggplot**: *geom_line() + scale_x_continuous(trans=‘log10’, labels=c(‘n=100’, ‘n=1000’), breaks=c(100, 1000))*
4. Multiple-Step Experiment: Playing the Lottery Three times: **rmd** | **r** | **pdf** | **html**
   - Paths after 1, 2 and 3 plays.

A.5 Discrete Probability Distribution links

1. Discrete Random Variable and Binomial Experiment: **rmd** | **r** | **pdf** | **html**
   - Discrete Random Variable, expected value and variance.
   - Binomial Properties, examples using USA larceny clearance rate, WWII German soldier survival rate
   - **r**: *dbinom() + pbinom() + sprintf(paste0(‘abc\n’, ‘efg = %s’), ‘opq’) + round(1.123, 2) + lapply()*
   - **ggplot**: *df %>% ggplot(aes(x)) + geom_bar(aes(y=prob), stat=‘identity’, alpha=0.5, width=0.5, fill) + geom_text(aes(y=prob, label=paste0(sprintf(‘%2.1f’, p), ‘%’)), vjust, size, color, fontface) + labs(title, x, y, caption) + scale_y_continuous(sec.axis, name) + + scale_x_continuous(labels, breaks) + theme(axis.text.y, axis.text.y.right, axis.text.y.left)*
2. Poisson Probability Distribution: **rmd** | **r** | **pdf** | **html**
   - Poisson Properties, Ladislaus Bortkiewicz and Prussian army horse-kick deaths.
   - **r**: *dpois() + ppois()*
   - **ggplot**: *geom_bar() + geom_text() + gome_line() + geom_point() + geom_text() + labs() + scale_y_continuous() + scale_x_continuous() + theme()*

Bibliography

Müller, K. and Wickham, H. (2019). *tibble: Simple Data Frames.* R package version 2.1.3.

Wickham, H. (2019). *tidyverse: Easily Install and Load the 'Tidyverse'.* R package version 1.3.0.

Wickham, H., Chang, W., Henry, L., Pedersen, T. L., Takahashi, K., Wilke, C., Woo, K., and Yutani, H. (2019a). *ggplot2: Create Elegant Data Visualisations Using the Grammar of Graphics.* R package version 3.2.1.

Wickham, H., François, R., Henry, L., and Müller, K. (2019b). *dplyr: A Grammar of Data Manipulation.* R package version 0.8.3.

Wickham, H. and Henry, L. (2019). *tidyr: Tidy Messy Data.* R package version 1.0.0.

Wickham, H., Hester, J., and Francois, R. (2018). *readr: Read Rectangular Text Data.* R package version 1.3.1.

Xie, Y. (2020). *bookdown: Authoring Books and Technical Documents with R Markdown.* R package version 0.18.