USING R/RSTUDIO IN THE TEACHING OF STATISTICS

by

Trey Brister

A thesis submitted to the Graduate Faculty of

Elizabeth City State University

in partial fulfillment of the

requirements for the Degree of

Master of Science in Mathematics

Elizabeth City, North Carolina

May 2022

APPROVED BY

Kenneth L. Jones, Ph.D.	Julian Allagan, Ph.D.
Committee Chair	Committee Member
Mohammad Talukder, Ph.D.	Jeffrey K. Ingram, M.S.
Committee Member	Committee Member

Table of Contents

Table of Figures

Table of Figures

Abstract

The growth of mathematics, statistics and computer science have continued in the Science, Technology, Engineering and Mathematics fields. A new field that has emerged from the combination of computer science, statistics and mathematics is the field of data science. Data science has emerged as a field of its own thanks to the significance of training people and developing tools that ae able to utilize the large amounts of data being generated in the modern world. This can be contributed to the development of various tools to be utilized by researchers, professionals, experts, and newcomers. The tools can be as simple as a one purpose application and as complex as a software package with a broad range of usage. One such tool that has emerged is R. R is a programming language that has found a large amount of popularity amongst statisticians. This is due to the capabilities of R being tuned towards statistics and its applications. As R continues to develop, we are able to see its history as well as the various tools R has for teaching statistics.

Dedication

To my family and friends, thank you for your continuous support along my educational journey up to this point. Thank you for understanding the sacrifices that I have had to make as a student in pursuit of my education.

To my thesis advisor and committee members, thank you for your support, assistance, and mentorship during my two years at Elizabeth City State University (ECSU). Thank you for the opportunity to attend ECSU in pursuit of my graduate degree.

Acknowledgments

Thanks to Dr. Kenneth L. Jones for being my thesis advisor. Many thanks to the Department of Mathematics, Computer Science, and Engineering Technology and its faculty for allowing me to pursue this degree. Thanks to the Sindhu Unnithan, Dr. Lester W. (Bill) Jones, and the rest of the Department of Mathematics at Xavier University of Louisiana for preparing me to be able to pursue a higher mathematics degree. Thanks to De La Salle High School for giving me the space to complete my degree as I taught. Thanks to Mon, Dad, and my sister Tyza for supporting me through my entire educational journey.

Chapter 1: Introduction

R has grown as a result of the growth of data science. Data science was founded from the interdisciplinary studies of the fields of mathematics, statistics and computer science. Data science is the field of acquiring, analyzing, and using data that is present in a variety of different states from structured to unstructured. Data science involves mathematical methods, statistical methods, and the usage of computer science principles too. Data science includes tools, principles, and philosophies held within the data science community. R is a programming language developed by statisticians and data scientists for the goals of statisticians and data scientists. This allows R to be well positioned as a tool for teaching statistics.

1.1 Basic Statistics Course Consists Of

1.1.1 Introduction to Statistics

The initial layout of statistics is explained. This portion covers basic concepts and terminology that will be used throughout and are part and partial to statistics itself. The initial definition of data and different statistical methods such as the mean, standard deviation, median, quartiles are introduced and defined. The definition of statistics is introduced and thoroughly explained as well as the importance of statistics to other academic disciplines. Some sub-fields or specific examples of the usage and applications of statistics are included. The concepts can be applied in or implemented in R thanks to its functionality.

1.1.2 Exploring Data with Tables and Graphs

Data can be neatly stored in the form of tables. Tables consist of rows and columns that are data points and variables respectively. Data in tabular form is rare to find data in in the large amounts of data being generated, stored and accessed. Tables allow data to be easily accessed and manipulated. The usage of graphs is to provide visualizations for the data. Using tables to generate graphs in R will provide more insight as to the relationships and nature of the data. There are some characteristics of the data that will be easier to observe in data visualizations. The graphs in R will provide additional details for statistical work in R allowing opportunities for further explanation or exploration in teaching.

1.1.3 Describing, Exploring and Comparing Data

1.1.3.1 Probability

Probability is the study of how likely a given event is to occur based on specified conditions. The field of probability can then be extended into four main subfields (What is Probability?, 2012). The first is classical probability (What is Probability?, 2012). The second is empirical probability (What is Probability?, 2012). The third is subjective probability (What is Probability?, 2012). The fourth is axiomatic probability, a hybrid of classical, empirical, and subjective probabilities (What is Probability?, 2012). The probability of an event cannot exceed one and cannot be less than 0. Additionally, the sum of the probabilities of all events in a sample space must equal 1. The probability of an event happening, such as event A, is P(A).

Probability distributions describe the likelihood of an event occurring. Probability distributions for discrete are different due to the different nature of discrete and continuous probabilities.

1.1.3.2 Correlation and Regression

Correlation describes the relationship between two variables, say x and y. Correlation is measured on the interval of -1 to 1, where -1 is a strong negative relationship between the variables x and y and 1 is a strong positive relationship between x and y. If the correlation is between -1 and 0, then there is a negative relationship that gets weaker the closer the correlation is to 0. If the correlation of x and y is between 0 and 1, then there is positive relationship between x and y that gets weaker as the correlation approaches 0. If the correlation is 0, then there is no relationship between the two variables. R has multiple methods for different situations of determining the correlation of variables in data. There are also methods to generate correlation matrices and graphs that can provide better visual aid as to the relationships between different variables.

Regression is one category of statistical modeling that can be done. Regression is using a set of independent variables in an attempt to predict the value of a continuous dependent variable. An example would be seeing how much time studying, how many credit hours a student is enrolled in, and if the student was employed or not to estimate the end of semester GPA of a student. For regression, linear models can be fit to data and then used to make predictions. R has multiple packages and functions for the purpose of fitting linear models to data. Using R to demonstrate regression and teach regression in a statistics course can be beneficial to instructors thanks to the large amount of statistical functions already available.

1.1.3.3 Discrete and Continuous Probability Distribution

Data in its simplest form is information. Data can be numerical, or it can be text. Data can be precise measurements or responses to a survey or social media post. Data can be gained from a potentially unlimited number of sources. Data is not always properly prepared or cleaned once acquired. Data preprocessing, steps to clean and organize the data for proper use, may have to be taken.

Discrete data is data that is countable. Examples are how many times a specific word appears in a text. Another is the number of blue, red, green and yellow beads that are in a bag of beads. Discrete data is often described using frequency or percentages. The most common ways to plot discrete data are bar plots. Discrete data can be used to group data as well to gain more information on the continuous data of specific groupings of data points.

The first type of data is continuous data. This is data that lies in an interval and or would be uncountable in the context of discrete data. Continuous data can include the distribution of heights in inches of the male population of a city, or the average heart rates of patients in a hospital. Continuous data is usually described with the minimum, maximum, median, first quartile, third quartile, mean and standard deviation. Other statistical measures may be used as well depending on the data itself. Continuous data is often plotted using histograms or density plots.

Discrete probability distributions show the probability of a specific event occurring. For example, if there were three events A, B, C, with the probabilities of 0.25, 0.50, and 0.25 respectively, we would get the resulting chart:

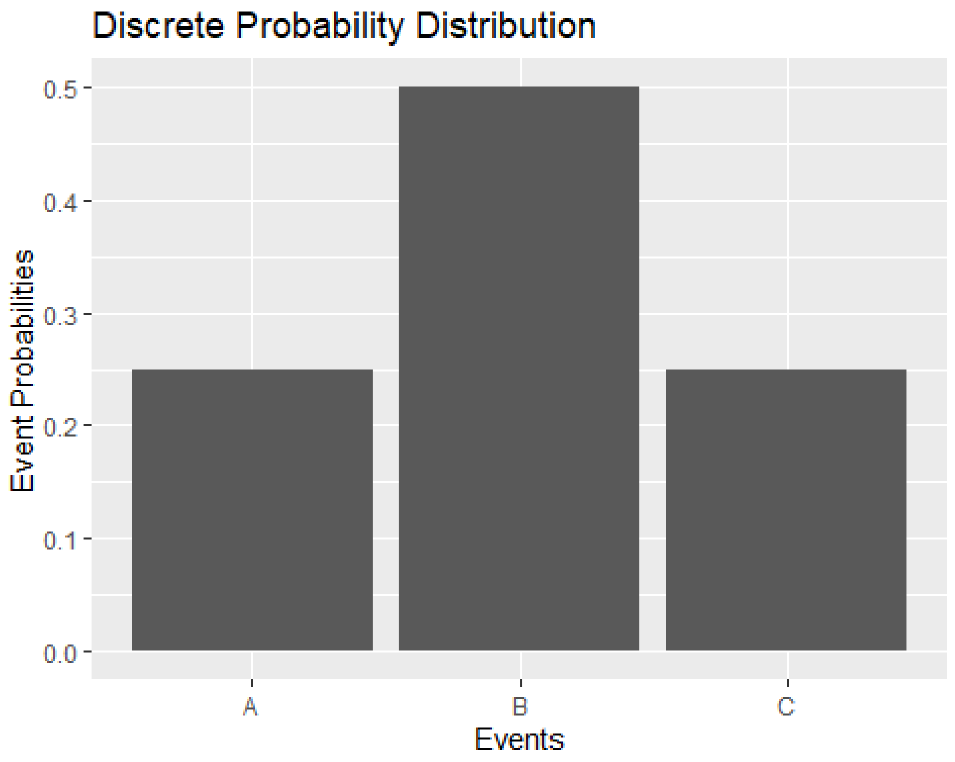


Figure 1: Example of discrete probability distribution

We can easily determine the probability of given event or set of events. Discrete probability distributions can be modeled using bar plots.

Continuous probability distributions can be used to determine the likelihood of a random variable taking a value within a specified interval.

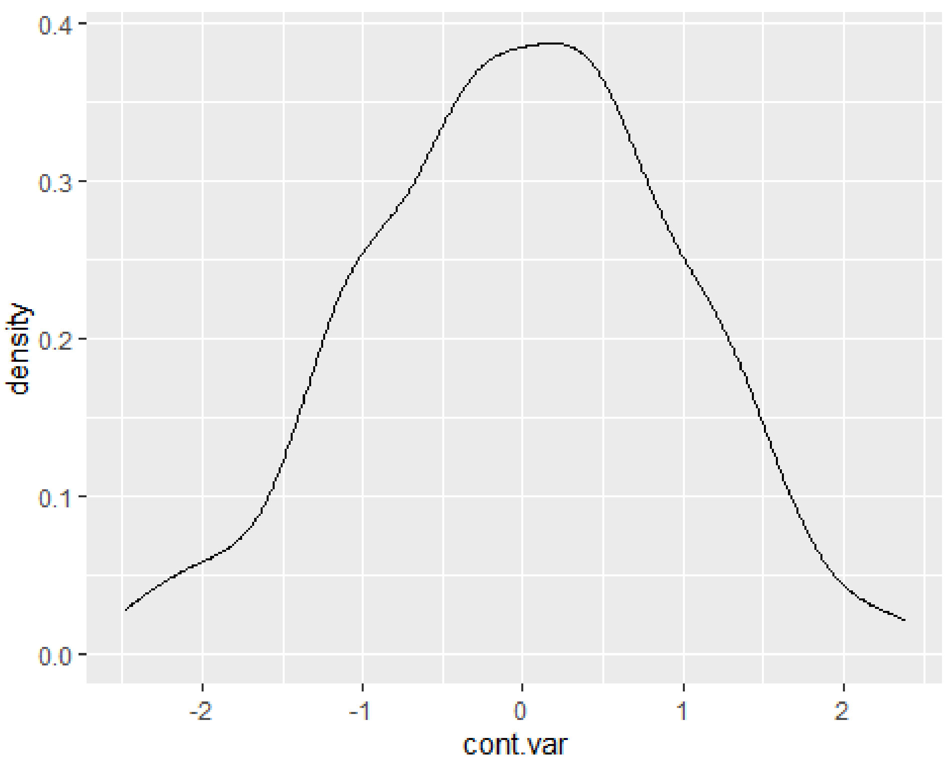


Figure 2: Example of continuous probability distribution

As observed above, continuous probability distributions can be plotted with density plots.

1.1.3.4 Normal Probability Distribution

The Normal Probability distribution is a bell-shaped curve that is symmetric, and the mean and median are equal to each other. The standard normal distribution is a normal distribution where the standard deviation is 1, and the mean is 0. The importance of the normal distribution is that it allows access to large of number of statistical tools. According to the Central Limit Theorem, when a sample size is large enough, the probability distribution will approach a normal distribution. In using normal distributions or distributions that satisfy the Central Limit Theorem to approach a normal distribution using R's functionality, more additional opportunities can be presented within the classroom.

1.1.3.5 Confidence Intervals and Sample Sizes

Sample data is data that is taken at random from a population to enable the estimation of parameters of the population. Sample sizes tend to be small as larger samples tend to cost more to obtain. Multiple samples from a population may also be taken.

Confidence intervals are a crucial part of the statistical process. Confidence intervals provide an interval that a true value for a population parameter may rest in given a significance level. The conventional significance level to use is the 95% significance level. Confidence levels are preferred over p-values due to the confidence interval providing situational information for the results from the sample data. The p-value on the other hand does not provide said results.

Using R to assist in teaching confidence intervals in the classroom can expand the options for instructors. Using built-in R datasets or external datasets students can be introduced to exercises and projects that reinforce the importance confidence intervals in statistical reporting and why they are favored over simply having the p-value.

1.1.4 Hypothesis Testing

Hypothesis testing is used to evaluate whether a presumption of an idea or a population parameter evaluated to determine the likelihood such is true from the sample data collected and used. There is a null and alternative hypothesis. The null hypothesis is the possibility the presumption is true, and the alternative hypothesis is the possibility the presumption is false. The significance level, typically set at 95%, is used to determine if the null hypothesis is kept or rejected.

Using R to reinforce hypothesis testing skills can be valuable to instructors. R Markdown has a lot of features such as the usage of Latex and R code that can enable students to write statistical reports to submit as assignments. This combined with knowledge and application of confidence intervals can allow students to be able to practice the coursework in a repeatable standardized structure.

1.2 R Compared to Other Software

R has been demonstrated to have considerable abilities for statistics. R has the basic properties that can be observed in a programming language to allow for usage. R can be used in its standalone software package, can be modified through loading additional packages, or modifying existing packages. This allows R users to tailor R on their systems to specific purposes or contexts. People can break down existing packages and functions in R to be able to understand how they work to be able to make modified forms that can fit more specific purposes. The open-source nature of R allows for people to be able to validate that the functions, objects and other components of R packages are working appropriately. If the package has issues or errors, then people can try to reach out to the managers of the package to get that package's issues resolved or attempt to resolve it themselves.

Graphing calculators can serve as convenient tools when R is not available for quick calculations. Laptops are less portable than graphing calculators, and graphing calculators have a good amount of functionality on their own. R, however, is more versatile than a graphing calculator when R is installed on a sufficient laptop or desktop. R can store and manipulate larger datasets and the full keyboard of a computer can be much quicker to use than a graphing calculators buttons. For instructors, this can be a massive advantage as they can increase the scale of data that homework and projects in statistics courses use without having to think of the limitations of a graphing calculator. The plots that are made in R can be put into a report or homework assignment instead of requiring the student to recreate the plots by hand or with another piece of software. R and RStudio can bring both the work and final answers to assignments to the same document without having to rely on software strictly outside the R environment.

Python is another programming language that is similarly an open-source project like R (Tarshizi, 2021). Python can be observed to be developed by Guido van Rossum and initially released in 1991 (Tarshizi, 2021). The organization that now manages Python is the Python Software Foundation (Tarshizi, 2021), similar to the R Core Group.

R's focus on statistics can be effective for the statistics community and problematic at least initially for other communities that do not have any established work in R. While there can be new packages developed, R users are left to develop what they want on their own if the packages needed do not exist. This makes R a programming language that can require motivation form the users to be able to learn R to be able to add whatever functions or packages that they desire. Python has a much broader and larger community behind it. This is evident when looking at data on the Stack Overflow inquiries that are posted per month from 2010 to 2020 (Tarshizi, 2021). The first month of 2020 displayed that Python was just a step below the most favored programming language (Tarshizi, 2021). Python is seen to be favored by approximately 30 percent of people involved in programming, whereas R was not in the top five programming languages at the same time (Tarshizi, 2021). With a larger community means more help is available and more resources within the programming language's community in general. A large reason for Python having a broader community can be attributed to Python being made for virtually anything (Tarshizi, 2021), not limited to statistics focused projects or work. This invites a lot more people to join the Python community and develop new software in Python as they do not need to have a set of goals that are statistics based or statistics related as can be observed in R (Tarshizi, 2021). More attention has been shifted to Python due to the continued advancement of data science (Tarshizi, 2021).

The focus on statistics can be noted to be more effective at preventing the R programming language from being diverted away in future versions from its support of statistical purposes and methods. R's community will be likely be easier to navigate for statistics purposes as the wide variety of packages to be developed for R are related in one way or another to statistics. In Python, there can be any number of different libraries for say astronomy, graphic design, cryptography, or highly unique interests not involved in statistics. The appeal of this specificity will likely sustain R has the statistics community continues to grow. The statistical focus of R, its community, and the continued advancement of functionality in R enable R to be an excellent software package for statistics coursework. It comes with a lot of functionality built-in or easily accessible to install by the user. Plus, once R is installed it can be used without an internet connection unlike some alternatives.

Excel lacks quite a bit when looking at the full scope of what R is capable of and will be capable of doing in the future as the R community grows. R code provided in coursework can be easily rerun by the students to be able to see how each section of the code works and functions. R encourages people to learn how to code for themselves and how to understand how the code works. With Excel, users will have to trust the proprietary software is correct and cannot necessarily edit the software as easily as with R. R grants more control over the data visualizations allowing the R plots to be considerably better than that of the Excel plots. When looking at the options available in R then the options available in Excel, it is observed that using Excel over R will result in noticeable restrictions of how much instructors and students will be able to use Excel for statistical purposes. Thanks to RStudio, the work done in R can be presented directly in an R Markdown document unlike Excel, where the tables have to be imported or copied from Excel to Word with varying levels of success. Excel can have issues performing calculations whereas the R has been validated to make sure that R is performing the correct calculations.

Stat Crunch is notably web-based and as a result requires a sufficient internet connection be accessible anytime a student wants to do work. Minitab can be used on the Windows operating system, but that does not provide equivalent support for potential users that MacOS or Linux. R is suitable for all of three of the dominant operating systems giving it an advantage over Minitab and Stat Crunch. R does not need a subscription to utilize any of its tools, packages, and associated software that is maintained by the R community. This makes R more accessible for institutions, students and instructors as there is no need to organize funding for buying R or RStudio.

Chapter 2: History of R

2.1 General

In this chapter we examine the history of R that has been tracked well since the original ideas of S. Since R is open source, this will be more effective in keeping track of R's development in the present and future. Since the main R code can be accessed by anyone, we can see the code and details regarding the code overall. This includes how the code has been changed and if any portions of the code are outdated. Different versions of R and its packages can be accessed, allowing for users to be able to look at more specific details of the development history of R and its packages.

2.2 S

R stems from the programming language S made by employees of Bell Telephone Laboratories (Peng, 2020). The creation of S began on the year 1976 (Peng, 2020). S then underwent a number of changes and different design paths seen in the fourth iteration of S followed by S-PLUS (Peng, 2020). S has recently rested in the possession of TIBCO (Peng, 2020). The value of S cannot be underestimated as S received the Association for Computing Machinery's Software System Award of 1998 (Peng, 2020).

S originated from the data scientist community instead of the typical software communities of the other programming languages (Peng, 2020). The result of this is that S was made to appeal to the data science community as the first priority (Peng, 2020). John Chambers, one of the creators of S, has mentioned that S was to give people a start in S without having to learn the concepts of programming languages until the complexity is involved (Peng, 2020). S was meant to appeal to the newcomers and the experts (Peng, 2020). R is founded on S, so many similarities are shared between the two (Peng, 2020).

2.3 R

R was created in part to allow everyone to access as S was proprietary technology (Peng, 2020). 1991 was when R was created, but the notification to the public was in 1993 (Peng, 2020). The license that R has to allow it to be completely available to all is the GNU General Public License (Peng, 2020). R could then be edited by any means deemed appropriate by anyone (Peng, 2020). This has allowed R to be developed for any computer from mobile devices to standard computers (Peng, 2020). R will also undergo regular updates that can resolve small issues in R or provide new applications (Peng, 2020). Virtually anyone could propose an update for R to be approved by the R Core Group (Peng, 2020). The R Core Group serves as the governing body for the R programming language (Peng, 2020). Any edits that are to happen to the main branch for R is managed by R Core Group (Peng, 2020). The year 2000 marks when R was made available for everyone (Peng, 2020). R can be observed to be a programming language that was not serve a profit. The R language was to serve the desires and goals of the data science community and people who were interested in data science.

R started off being similar to the previous S language to allow for simple conversion from S to the newer R at that time (Peng, 2020). The details of R beyond the surface are more akin to Scheme (Peng, 2020).

R is well known for enhanced graphical applications that have been revered in various works (Peng, 2020). R typically outcompetes the alternatives in regard to this (Peng, 2020). The development of R’s graphical packages has continued and its lead amongst the alternatives (Peng, 2020).

Concerning the licensing for R, the license itself is possessed by the R Foundation (Peng, 2020). R falls into the category of “free software” (Peng, 2020) and provides people with a lot of freedom when using R. There are four main principles for software packages within this category (Peng, 2020). The first is the software can be utilized for virtually anything (Peng, 2020). The second is that you can reverse engineer the software then modify said software to the most basic parts (Peng, 2020). The third is that you can clone your versions for others to have (Peng, 2020). The final principle is that anyone can better the software package to be shared with the broader space (Peng, 2020), in the case of R being the data science community.

2.3.1 R and its Repositories

R can be broken down into two components (Peng, 2020). The first and most crucial part is the R as a standalone software package (Peng, 2020). The second component is the other R packages that can be downloaded and included within R (Peng, 2020). The standalone R has supplemental packages that can assist in regular tasks or functionality that are expected of R (Peng, 2020). There are supplemental packages that are advised to be included alongside the standalone R for their value in regular data science tasks (Peng, 2020). Beyond that, there are supplemental packages that may be installed and loaded in R (Peng, 2020). The number of supplemental packages is 4,000+ (Peng, 2020). The R packages can be expected to have creators from different backgrounds and interests given the broad availability of R (Peng, 2020). The two main sources of packages are CRAN and Biconductor (Peng, 2020). The number of packages that are actually viable cannot be feasibly measured as packages can technically be stored and distributed form possibly anywhere (Peng, 2020). A clear example of such can be seen in code repositories as exemplified by Bitbucket (Peng, 2020).

These different repositories present different opportunities for students to learn how to use different statistical operations and methods. R being a programming language presents more opportunities for students as they can use R to practice different statistical methods for coursework. They can take what is learned form the coursework material and look into applying that material in practical ways through projects and homework assignments. This reinforces students to understand the material so that they are able to apply statistical methods in R appropriately.

2.4 Examples of the Usage of R

Some basic uses of R will be examined in later chapters. This started on simpler tasks of arithmetic operations and assigning values to variables to R’s plotting abilities. R is straightforward in its syntax and allowing users to develop their own functions. The plotting abilities in R are especially worth noting. The plotting allows for flexibility for the user to be able to most effectively portray information. It can be observed that with the ggplot2 library or standalone R, a number of plots can be constructed to look at the relationships of different types of variables.

2.5 Working In R

When working in R, people will typically download and install the interactive development environment known as RStudio (Tarshizi, 2021). This puts all the crucial portions for coding in R live all in one application (Tarshizi, 2021). Two parts of RStudio that are crucial are the text editor for typing the code and console for running the typed code to produce output (Tarshizi, 2021). An additional benefit of RStudio is R Markdown that simplifies the production of papers or documents with R code included (Tarshizi, 2021). R Markdown is not limited to R though (Tarshizi, 2021). Other possibilities are SQL, Python, CSS, JavaScript, Bash (Tarshizi, 2021) and potentially more based on the motivation and interests of R users. Some similar software for Python is PyCharm, IPython, and Project Jupyter (Tarshizi, 2021). It is observable that an exact or similar replica with all the similar functions of RStudio is not present in Python (Tarshizi, 2021). Python coding has broader support than R code for different software (Tarshizi, 2021).

2.6 Costs Running Code

Depending on the code that is being written for the statistics work being done, R does not seem to have any notable problems in comparison with Python (Tarshizi, 2021). This does not mean that they will execute similarly in all possible situations (Tarshizi, 2021). Python has been seen to have an edge with for loops where the number of loops is below 100 when compared to R simply due to how Python and R are built differently (Tarshizi, 2021). Overall, current technology makes this difference barely noticeable if at all (Tarshizi, 2021). Even if there are statistics goals that require a lot of resources that are many ways to get those resources without jeopardizing the current project (Tarshizi, 2021). Both R an Python have demonstrated the ability to be ran on mobile devices, which have much less to offer than the modern laptop, desktop or server computer (Tarshizi, 2021).

2.7 Libraries and Packages

As previously pointed out, having a programming language that concentrated to a particular sheet of goals makes the community code simpler to manage and navigate. This is shown by the management of the code in CRAN, where a large set of R packages can be simply accessed (Tarshizi, 2021). For Python, this is not the case as seen in the Python Package Index (Tarshizi, 2021).

2.8 Graphics

As demonstrated earlier in the Methodology, R is highly capable of making suitable graphics and is primarily limited by the user. Python has the alternatives of matplotlib and its derivative seaborn (Tarshizi, 2021). R, however, can use the graphics in the R Markdown document that can simplify the entire process overall when writing papers. There are options to remove the visibility of code or the code output too. R, through RSudio, can be flexible to the demands of the user. This makes things a lot simpler considering that there is more efficiency in writing. There are added benefits for R Markdown being able to recognize Latex syntax as well.

2.8.1 R Cannot Do Everything

R is most notably held back in the way it is built (Peng, 2020). R was built using S as a reference, while S was created over five decades earlier (Peng, 2020). R has had slow development to remove the problems that have emerged from such (Peng, 2020). Objects created in R are placed in the memory, which makes R problematic if too much memory is required (Peng, 2020). This is notably more than alternative software packages with similar goals (Peng, 2020). This has been eased thanks to the continued advancement of hardware and software that is available, this is not a permanent solution (Peng, 2020). There is data being collected that will still be far greater than the memory that can be found on a given computer (Peng, 2020). Another more notable drawback of R is that the R community has not developed a package for every scenario (Peng, 2020). A large amount of the labor relies heavily upon the productivity, creativity and dedication of people that have R and want some application to build the application if the application is nonexistent at the time (Peng, 2020). R has been pushed by the involvement of those that use R (Peng, 2020).

Chapter 3: Data Manipulation

In this chapter we will be examining data manipulation in R.

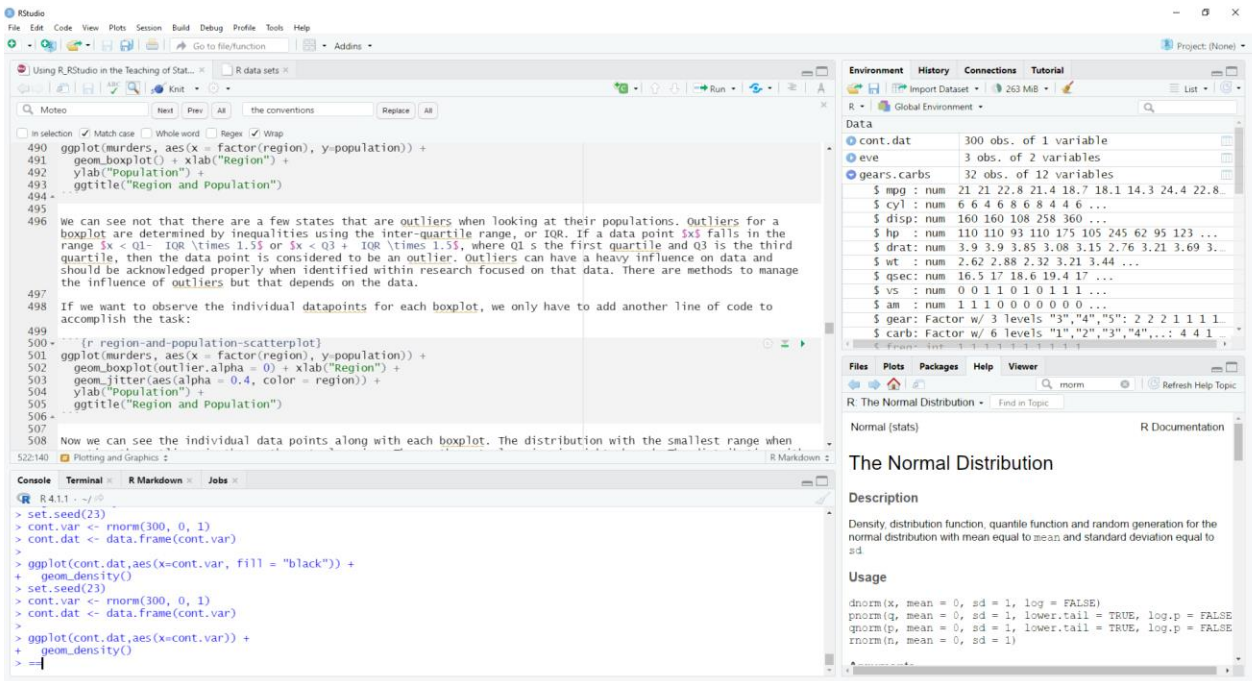
3.1 R

Installing R is as simple as going to https://www.r-project.org/, then following the specified link to download the R programming language from a chosen source such as the cloud mirror of RStudio. There will be different software download links for different operating systems. After the download is complete, proceed to open and run the install program and proceed to install R onto your computer. For some versions of Linux, there may already be an R version present.

3.2 RStudio

For RStudio installs, you go to https://www.rstudio.com/ and click on the link for downloading RStudio at the top. On the following webpage, you scroll down and click the download link for the free version. After that, open the downloaded RStudio install file and run it. Following this, both R and RStudio should both be installed on the computer.

3.3 RStudio Panes



```
ggplot(murders, aes(x = factor(region), y=population)) +
  geom_boxplot() + xlab("Region") +
  ylab("Population") +
  ggtitle("Region and Population")
```

We can see not that there are a few states that are outliers when looking at their populations. Outliers for a
boxplot are determined by inequalities using the inter-quartile range, or IQR. If a data point x falls in the
range $x < Q1- IQR \times 1.5$ or $x < Q3 + IQR \times 1.5$, where Q1 s the first quartile and Q3 is the third
quartile, then the data point is considered to be an outlier. Outliers can have a heavy influence on data and
should be acknowledged properly when identified within research focused on that data. There are methods to manage
the influence of outliers but that depends on the data.

If we want to observe the individual datapoints for each boxplot, we only have to add another line of code to
accomplish the task:

```{r region-and-population-scatterplot}
ggplot(murders, aes(x = factor(region), y=population)) +
  geom_boxplot(outlier.alpha = 0) + xlab("Region") +
  geom_jitter(aes(alpha = 0.4, color = region)) +
  ylab("Population") +
  ggtitle("Region and Population")
```

Now we can see the individual data points along with each boxplot. The distribution with the smallest range when
```

```
> set.seed(23)
> cont.var <- rnorm(300, 0, 1)
> cont.dat <- data.frame(cont.var)
>
> ggplot(cont.dat,aes(x=cont.var, fill = "black")) +
+   geom_density()
> set.seed(23)
> cont.var <- rnorm(300, 0, 1)
> cont.dat <- data.frame(cont.var)
>
> ggplot(cont.dat,aes(x=cont.var)) +
+   geom_density()
> ==
```

```
Data
cont.dat       300 obs. of 1 variable
eve            3 obs. of 2 variables
gears.carbs    32 obs. of 12 variables
    $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8...
    $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
    $ disp: num  160 160 108 258 360 ...
    $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
    $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3...
    $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
    $ qsec: num  16.5 17 18.6 19.4 17 ...
    $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
    $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
    $ gear: Factor w/ 3 levels "3","4","5": 2 2 2 1 1 1 1...
    $ carb: Factor w/ 6 levels "1","2","3","4",..: 4 4 1 ...
```

Normal {stats} R Documentation

The Normal Distribution

Description

Density, distribution function, quantile function and random generation for the normal distribution with mean equal to `mean` and standard deviation equal to `sd`.

Usage

```
dnorm(x, mean = 0, sd = 1, log = FALSE)
pnorm(q, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE
qnorm(p, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE
rnorm(n, mean = 0, sd = 1)
```

Figure 3: Screenshot of RStudio on Windows

RStudio has four main panes in the top left, top right, lower left and lower right. The top left pane is display opened files for coding and markdown. This pane also offers a view of data or datasets that can be larger than the top right pane depending on how the panes are adjusted. The top right pane can be used to look at specific objects and data in the R session. The history tab of the top right pane tracks the code that has been previously run and the connections tab tracks any web connections that have made in R. The bottom left pane starts with Console tab which allows for users to input code to run directly and will display previously ran code from the top left pane. The Terminal tab in the bottom left is for working with bash coding for projects. The R Markdown tab displays the output from compiling the R Markdown file. The bottom right pane starts with the Files tab. The Files tab shows the home or current working directory, allows for navigation to files through directories, and allows users to set a new working directory. It also
```

has some basic file manager options. The Plots tab is used to display plots that are specified to be plotted there or plots R code from the Console. The Packages tab enables users to see which packages are installed, which packages are loaded into R and to explore the functionality of packages. The Packages tab allows users to install and load packages without using code. The Help tab is used to view the available help files in R for objects, datasets, functions and packages.

3.4 Creating Objects in R

Creating objects in R can be as simple as using <- and = for such. The object for R is determined based on the type of data supplied. For example, a string will make the object a character. A number will make the object numeric. Examples can be seen below:

```r
a <- 1

b <- "Math is fun"
```

Now the objects a and b have be created. We can check the types of said objects below:

```r
class(a)

## [1] "numeric"

class(b)

## [1] "character"
```

We can see that the object a is numeric. The object b is a character. We can see the values stored in these objects if we print them. This can be observed below:

```r
a
```

```
## [1] 1

b

## [1] "Math is fun"
```

3.5 Operations in R

We can see the values that were stored remain unaltered. For other objects, similar can be expected. The objects will remain in the memory for the R session until the session is terminated or the object itself is deleted. If the user wants to actively remove objects in the code, usch as objects that will no longer be used, then the remove or rm commands of R may be used. On the other hand, if we want to reuse a or b for other objects, all we have to do is reassign them.

Given R's feature you can perform mathematical operations in R. This includes addition, subtraction, division, modulus and multiplication to calculation of specific mathematical and statistical formulas. Let us demonstrate operations without the use of variables. First we will setup the operations to perform:

```
3 + 5 - 11

4 * 8 / 6

19^2 / 3

2 ^ 8 + 64 - 144 / 1.5
```

We can see the results for the operations below:

```
## [1] -3

## [1] 5.333333
## [1] 120.3333

## [1] 224
```

Next is demonstrating another set of operations but we will store the results in variables for reuse:

```
c <- 7 + 8 * 2

d <- 1 / c

e <- 3 ^ d + 33.33

f <- e - 1.089 * 5

g <- 8 %% 2
```

To see the results we must print the variables:

```
c

## [1] 23

d

## [1] 0.04347826

e

## [1] 34.37892

f

## [1] 28.93392

g
```

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

R has no issues with calculating the results of the previous operations. Another helpful tool in R is to use vectors to store multiple objects together in an easily accessible form. A vector is referred to as a data structure, and is one of many ways to organize data in R. What about if we want add the elements of two vectors of the same size together? To get the two vectors we will use the seq function to generate the numbers for each vector. Each vector will be stored in its own separate variable, then we can add the two vectors using the two variables the vectors are assigned to. We will reassign a and b to be our variables for storing the two vectors:

```
a <- seq(1,100)

b <- seq(0,by = .5,length.out =100)
```

We will check the length of the vectors with the length function. This is to make sure that both vectors are the same length.

```
length(a)

## [1] 100

length(b)

## [1] 100
```

Now we will add the two vectors together into a new object g.

```
g <- a + b

g
```

```
##   [1]   1.0   2.5   4.0   5.5   7.0   8.5  10.0  11.5  13.0  14.5  16.0  1
7.5
##  [13]  19.0  20.5  22.0  23.5  25.0  26.5  28.0  29.5  31.0  32.5  34.0  3
5.5
##  [25]  37.0  38.5  40.0  41.5  43.0  44.5  46.0  47.5  49.0  50.5  52.0  5
3.5
##  [37]  55.0  56.5  58.0  59.5  61.0  62.5  64.0  65.5  67.0  68.5  70.0  7
1.5
##  [49]  73.0  74.5  76.0  77.5  79.0  80.5  82.0  83.5  85.0  86.5  88.0  8
9.5
##  [61]  91.0  92.5  94.0  95.5  97.0  98.5 100.0 101.5 103.0 104.5 106.0 10
7.5
##  [73] 109.0 110.5 112.0 113.5 115.0 116.5 118.0 119.5 121.0 122.5 124.0 12
5.5
##  [85] 127.0 128.5 130.0 131.5 133.0 134.5 136.0 137.5 139.0 140.5 142.0 14
3.5
##  [97] 145.0 146.5 148.0 149.5
```

We can see We can see in the output that the calculations for g are the result of adding vectors a and b. We can see that for the output, before the first element of each line there is a number indicating which index the first element can be found in. For example, we can look at 5th, 30th, and 77th indices to see which numbers are stored there without printing out the entire vector:

```
g[5]
```

```
## [1] 7
```

```
g[30]

## [1] 44.5

g[77]

## [1] 115
```

If we look back at the vector g we can see it has the same numbers in the same indices.

We can do other mathematical operations as well such as subtraction, division and multiplication. To demonstrate, two new vectors of length five will be created. This is to minimize the output:

```
c <- c(4, 3, 2, 5, 10)
d <- c(12.5, 23.67, 87.56, 90.2134, -50.2)
c - d

## [1]  -8.5000 -20.6700 -85.5600 -85.2134  60.2000

d * c

## [1]   50.000   71.010  175.120  451.067 -502.000

d / c

## [1]  3.12500  7.89000 43.78000 18.04268 -5.02000
```

Each operation was still performed for each element. Similar operations can be done with vectors of different sizes but a warning will be thrown. Additionally, elements can be repeated in operations.

Vectors are not the only data structure that can be used in R. There are also matrices, lists and dataframes. Each data structure has its advantages and disadvantages. Additionally, each data structure can be used frequently within the packages of R providing a common means for each person to being understanding the function and capabilities of different packages in R.

3.6 Logic Operators

The logic operators of R are $>, <, ==, !=, \leq, \geq$ or greater than, less than, equal to, not equal to, less than or equal to (`<=`) and greater than or equal to (`>=`) respectively. Logic operators are used to compare values or objects within R and the result of the operation is either `TRUE` for the condition being satisfied or `FALSE` for the condition not being satisfied. We can view a few examples below:

```
3 > 4

## [1] FALSE

4 < 3

## [1] FALSE

"where" == "where"

## [1] TRUE

"data" != "datum"

## [1] TRUE

3 <= 100

## [1] TRUE
```

```
456 >= 400

## [1] TRUE
```

We can see that for each of the operations that there was a respective `TRUE` or `FALSE` outcome depending on whether the condition of the logic operator was satisfied. We can see for the first two, 3 is not greater than 4 and 4 is not less than 3 so those are both `FALSE`. For the third operation the string “where” is equal to the string “where” so the outcome is `TRUE`. For the fourth operation, “data” is not equal to “datum” so the outcome is `TRUE`. For the fifth operation, 3 is less than or equal to 100, so the outcome is `TRUE`. For the fifth operations, 456 is greater than or equal to 400 so the outcome is `TRUE`.

3.7 If, Else, Ifelse

Sometimes the actions to be performed need a certain condition to occur. This is to prevent unnecessary steps from carrying out or to prevent error from using the wrong type of object as the wrong input. For this, we have if, else, and ifelse. All three accomplish the similar task that they only allow certain actions to occur if a condition is met. The ifelse statement can work for vectors, unlike if and else.

```
e <- 4

if(e %% 3 == 0)

{

  print("e is divisible by 3")

} else

{
  print("e is not divisible by 3")
}

## [1] "e is not divisible by 3"
```

We can observe that the above code chunk checks if the value assigned to the variable e will have a remainder of 0 when divided by 3. The value of e was 4, which will not have a remainder of 0 when divided by 3. In the next code chunk we will observe what happens when the variable e is assigned the value of 24.

```
e <- 24
if(e %% 3 == 0)
{
  print("e is divisible by 3")
} else
{
  print("e is not divisible by 3")
}

## [1] "e is divisible by 3"
```

We can observe that the condition was satisfied, and the printed statement changed as well. If the condition is true for the if statement, then the actions in the if statement are carried out. If the if statement condition is false, the if statement actions do not run but when an else statement is present the else statement’s actions will be run when the if statement condition is not true. The ifelse statement can obtain the same results, but it works with vectors as well. This is because the ifelse statement is vectorized.

```
f <- c(1, 3, 16, 24, 84, 96, 333, 444, 505)
ifelse(f %% 3 == 0, TRUE, FALSE)

## [1] FALSE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE
```

We can see from the output above that the elements of the vector f were given a TRUE or FALSE value. TRUE was for when the element would have a remainder of 0 when divided by 3. If that condition was not satisfied, then the value of FALSE was given.

3.8 Functions

Another option in R is using functions. Functions are algorithms designed to perform repeat tasks over and over. R and its packages have built-in functions that can be accessed to perform routine tasks. For example, the mean function can be used the compute the mean of a vector:

$$\frac{1}{n}\sum_{i=1}^{n} x_i$$

```
h <- c(9, 11, -12, -456, 78, 90, 300, 150, 152, -56)
mean(h)

## [1] 26.6
```

When supplied the vector h it can be observed that the computed mean is 26.6. If we want additional information, we can use ?mean get the help file. We can see that there are additional arguments to use as well. One is na.rm that makes sure that only values that are not NA are used to calculate the mean. Otherwise, an error is thrown. We can demonstrate below:

```
i <- c(NA, 4, 5, 100, 10000.3, 4.67, 90.34, NA, 45, 67, -100.34, -500, -7000)
mean(i)

## [1] NA

mean(i, na.rm = TRUE)

## [1] 246.9064
```

The if `na.rm = FALSE` then the result is `NA` as the vector `i` has `NA` values. When `na.rm = TRUE` the mean can be computed.

3.8.1 User-defined Functions

Users of R can build their own functions as well. This can cut down on the amount of code that is typed and save time. For example, say we wanted a function that square a given number, add 2, then subtract 36. That can be written into a function. We can also add an if statment to make sure that NA values do not throw an error.

```
new.example.fun <- function(x){
  if(is.na(x)){
    return("Can not compute")
  }else{
    x <- x^ 2 + 2 - 36
    return(x)
  }
}
```

Now we can use the function to perform the operations. First, we use the number 12 for the argument x.

```
new.example.fun(12)
```

Through user-defined functions, people can develop functions, packages, and applications in R that may not exist or modify existing ones. This allows for people to be able to adapt R for their specific needs.

Chapter 4: Data Visualizations Using ggplot2

In this chapter we look at using data visualizations in R using the package ggplot2.

4.1 Packages

Packages provide additional options beyond what is available in R without having to come up with your own functions. Libraries can be loaded into R using the `library()` function but if the package is not installed it must be installed first. The way to install packages in the R code is the `install.packages()` function. It should be noted that function names can overlap. When this occurs, R will retain the most recently defined function as the default for use when functions share the same name. If a user defined functions shares the same name as a function currently available in R, the user-defined function will be saved as the default. When multiple packages are loaded, some packages may mask functions in other packages. When this happens, using the package name when calling the function can provide access to specific functions from specific packages.

The ggplot2, dplyr, plyr and dslabs packages will be loaded for the next section on plotting and graphics:

```
library(dslabs)

library(plyr)

library(dplyr)

library(ggplot2)
```

4.2 Plotting and Graphics

A crucial part of R's data analysis capabilities are its graphical abilities. This can be observed through both the built-in R abilities and those provided through R packages. Being able to read in data and process it are important steps that require data visualizations. Some characteristics of data can be easily observed by using visualizations. Additionally, visualizations can be far easier to explain and interpret for a broad range of audiences than raw data or tables of lengthy values.

We start by loading the heights dataset:

```
data("murders")
```

Next we can look at the population data of the murders dataset in the dslabs package. The data was taken in the year 2010 by the FBI from what can be read in the help file. The population variable is continuous. An appropriate visualization is a histogram and that cna be done using the hist() function. The result can be seen below:

```
hist(murders$population, breaks = 10, xlab = "Population",
     title = "Histogram of Populations in US States")
```

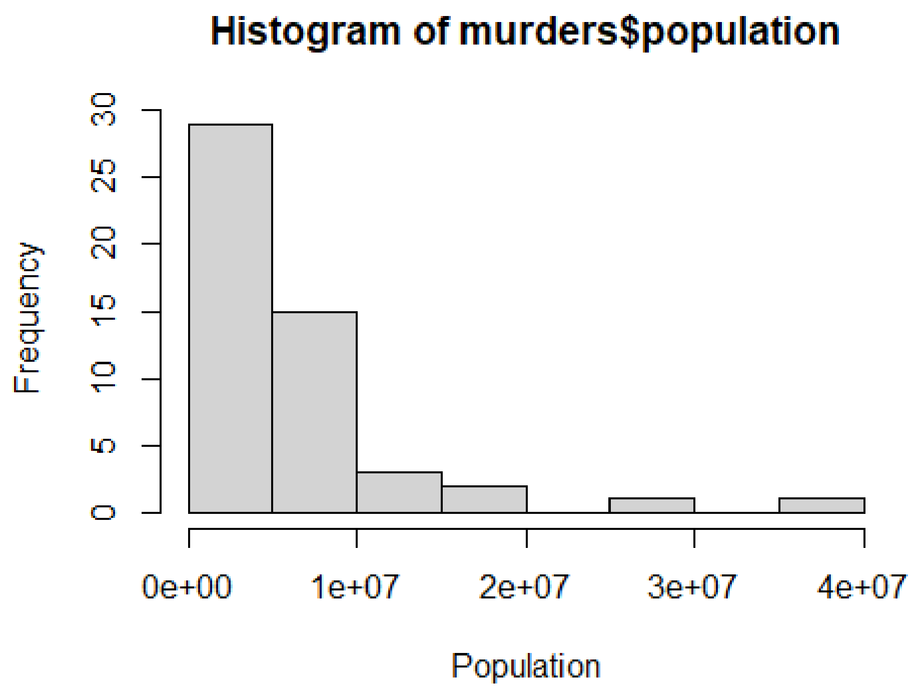


Figure 4: Histogram of populations of states from the murders dataset using base R

The histogram displays that distribution of populations in the United States. The distribution is left skewed. There may be a few states that are outliers as well. The histogram was done using the `plot()` function that is found in the standalone R software. Using ggplot, more options can be found for creating histograms.

```r
ggplot(murders, aes(x = population)) +
  geom_histogram(binwidths = 10) +
  xlab("Population") +
  ggtitle("Histogram of Populations")
```

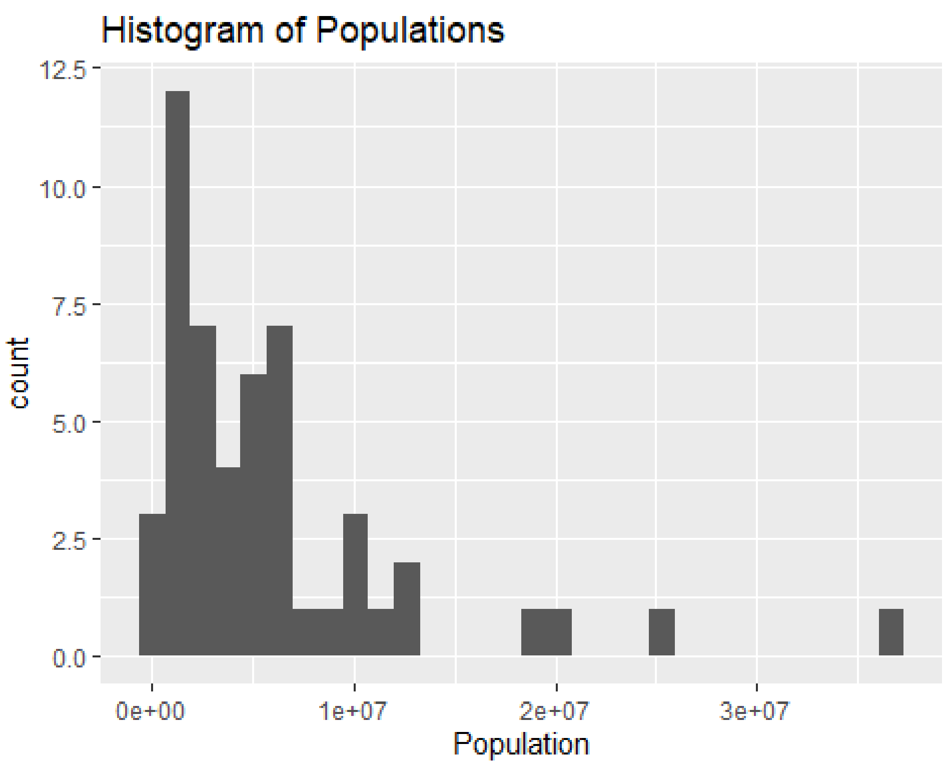


Figure 5: Histogram of populations of states from the murders dataset using ggplot2

The ggplot package can do not only the same things as the built-in `plot()` function, but more. The ggplot2 package can offer a lot more control over the visualizations than the built-in R functions and can allow for stacking visualizations on top of one another. For an additional option, we have demonstrated what happens if we change the color of the histogram bars to reflect the number of states in each region:

```
ggplot(murders, aes(x = population, fill = region)) +
  geom_histogram(binwidths = 10) +
  xlab("Population") +
  ggtitle("Histogram of Populations")
```

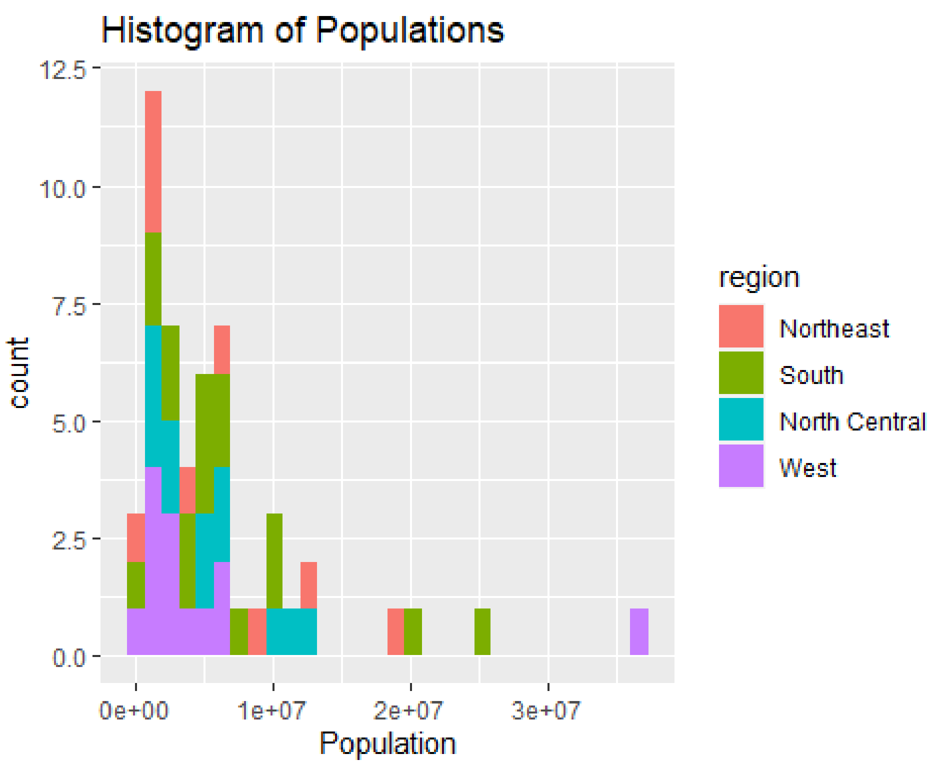


Figure 6: Histogram of populations of states from the murders dataset using ggplot2 and different colors for different regions

Another demonstration of the power of plot is plotting two continuous variables. For this we will use the variables population and total.

```
plot(murders$population, murders$total, xlab = "Population",
     ylab = "Total", main="Population and Total")
```

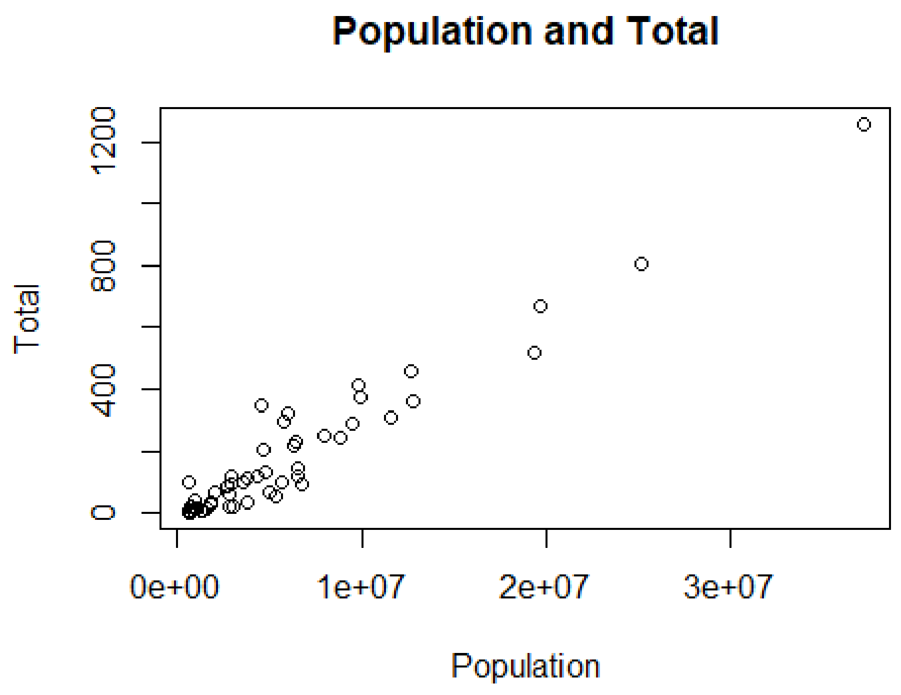


Figure 7: Scatterplot of population and total gun murders from the murders dataset using base R

We can see above that the result is a scatterplot. We can see that the there is a positive relationship between the population and total from what is seen in the data. Doing similar with ggplot provides plenty more options to use. In using these options, we can get a deeper analysis of the data.

```
ggplot(murders, aes(x = population, y=total, color = region)) +
  geom_point() + xlab("Population") +
  ylab("Total") +
  ggtitle("Population and Total")
```

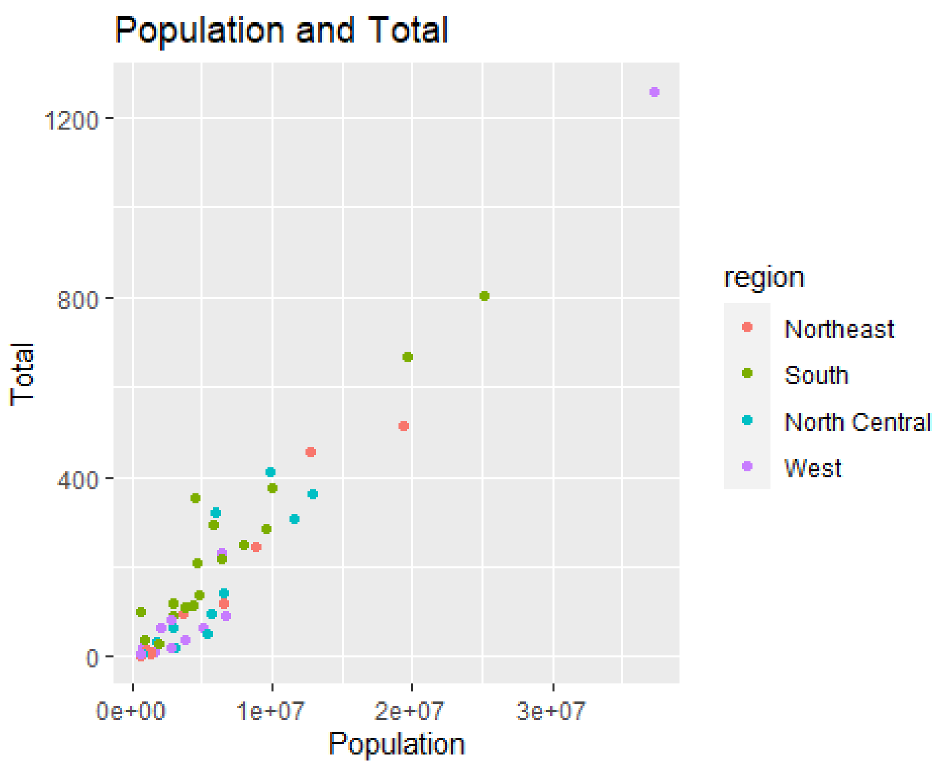


Figure 8: Scatterplot of population and total gun murders from the murders dataset using ggplot2

Now we can see how the population and total murders differ based on the region. The western states are centered near the bottom left. Again, we can confirm that there is a positive relationship between the variables of population and total murders. We can also add a linear model to confirm this.

```
ggplot(murders, aes(x = population, y=total)) +
  geom_smooth(se = FALSE, method="lm") +
  geom_point(aes(color = region)) + xlab("Population") +
  ylab("Total") +
  ggtitle("Population and Total")
```

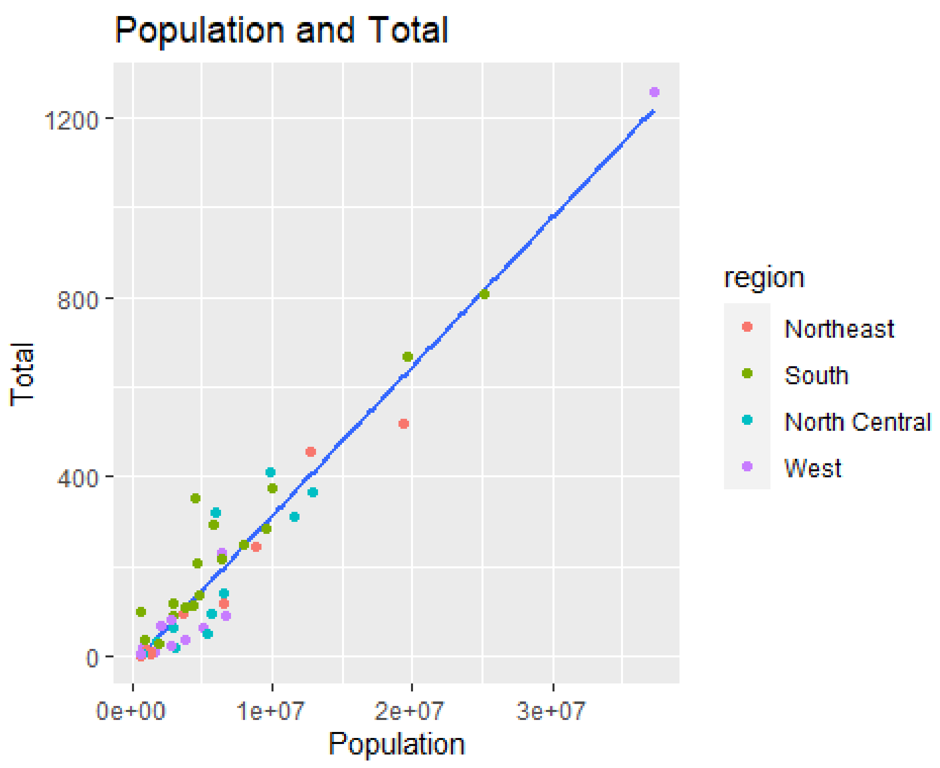


Figure 9: Scatterplot of population and total gun murders from the murders dataset using ggplot2 with a linear model added

The linear model reflects a positive correlation between the population and total murders variables. We can see there is a positive relationship in the data overall. What about looking at the distribution of populations by region? What about the distribution of total murders by region? We can accomplish both with the built-in R tools, or ggplot2, but we will continue forward with ggplot2 thanks to the increased flexibility. This will allow for more options for using the data visualizations.

First, we look at the distribution of populations by region. To do so, we use a boxplot:

```
ggplot(murders, aes(x = factor(region), y=population)) +
  geom_boxplot() + xlab("Region") +
  ylab("Population") +

  ggtitle("Region and Population")
```

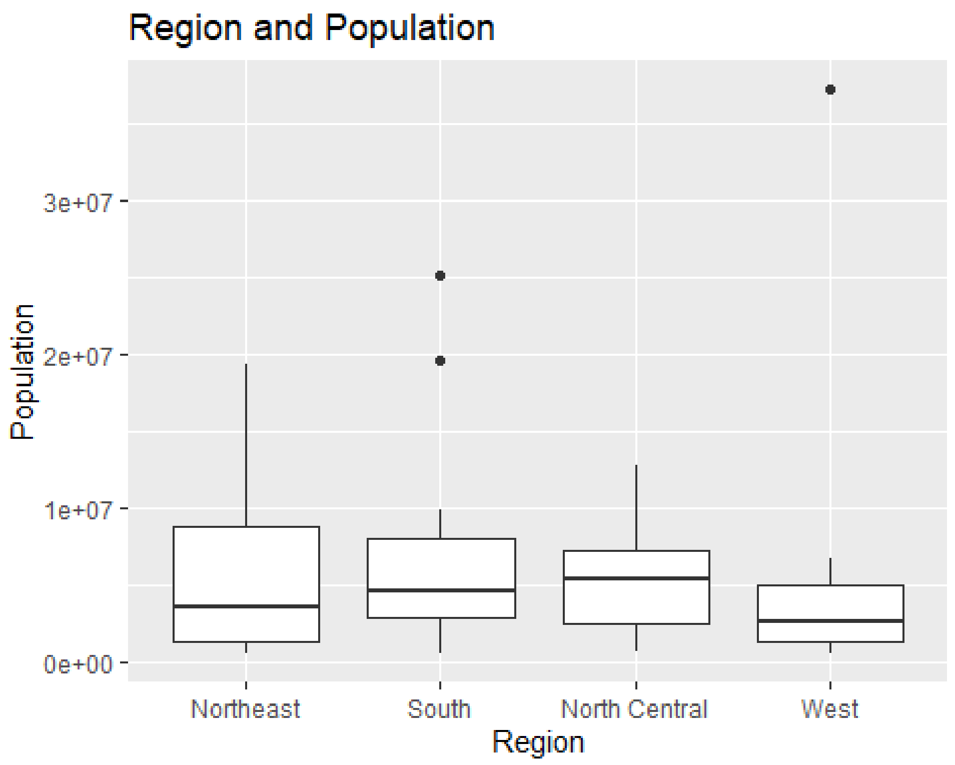


Figure 10: Box plots for the population for each region of the US from 2010 Murders data

We can see not that there are a few states that are outliers when looking at their populations. Outliers for a boxplot are determined by inequalities using the inter-quartile range, or IQR. If a data point x falls in the range $x < Q1 - IQR \times 1.5$ or $x < Q3 + IQR \times 1.5$, where Q1 s the first quartile and Q3 is the third quartile, then the data point is considered to be an outlier. Outliers can have a heavy influence on data and should be acknowledged properly when identified within research focused on that data. There are methods to manage the influence of outliers but that depends on the data.

If we want to observe the individual datapoints for each boxplot, we only have to add another line of code to accomplish the task:

```
ggplot(murders, aes(x = factor(region), y=population)) +
  geom_boxplot(outlier.alpha = 0) + xlab("Region") +
  geom_jitter(aes(alpha = 0.4, color = region)) +
  ylab("Population") +
  ggtitle("Region and Population")
```

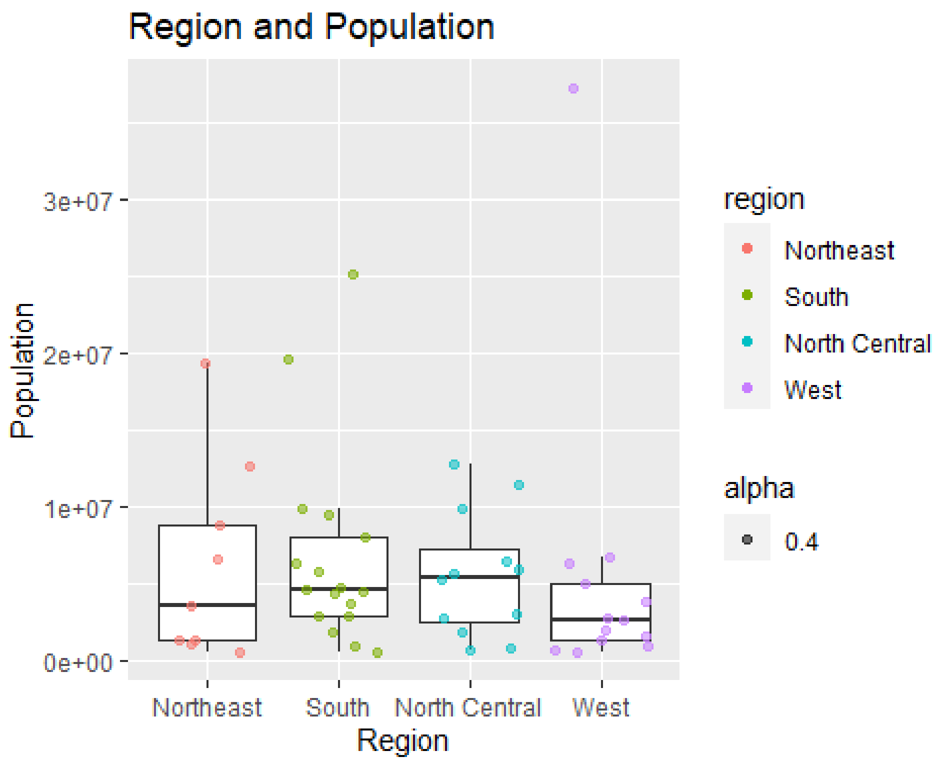


Figure 11: Box plots for the population for each region while displaying individual data points of the US from 2010 Murders data

Now we can see the individual data points along with each boxplot. The distribution with the smallest range when counting the outliers is the north central region. The north central region is right skewed. The distribution with the largest range counting the outliers is the western population distribution. The western region is left skewed and many of the western states have lower populations compared to the rest. The remaining regions are left skewed too.

Another idea that can be used is to use `facet_wrap()` or `facet_grid()` to create multiple plots involving the same variables based on some categorical variable to product multiple plots. Perhaps when generating the scatterplot earlier of total murders versus population, we wanted to have different plots for each region of the United States. This provides more control over what is displayed to audiences. A demonstration of such can be seen below:

```r
ggplot(murders, aes(x = population, y=total)) +
  geom_point() + xlab("Population") +
  ylab("Total") +
  facet_wrap(~region, nrow = 2, ncol = 2, scales = "fixed") +
  ggtitle("Population and Total")
```

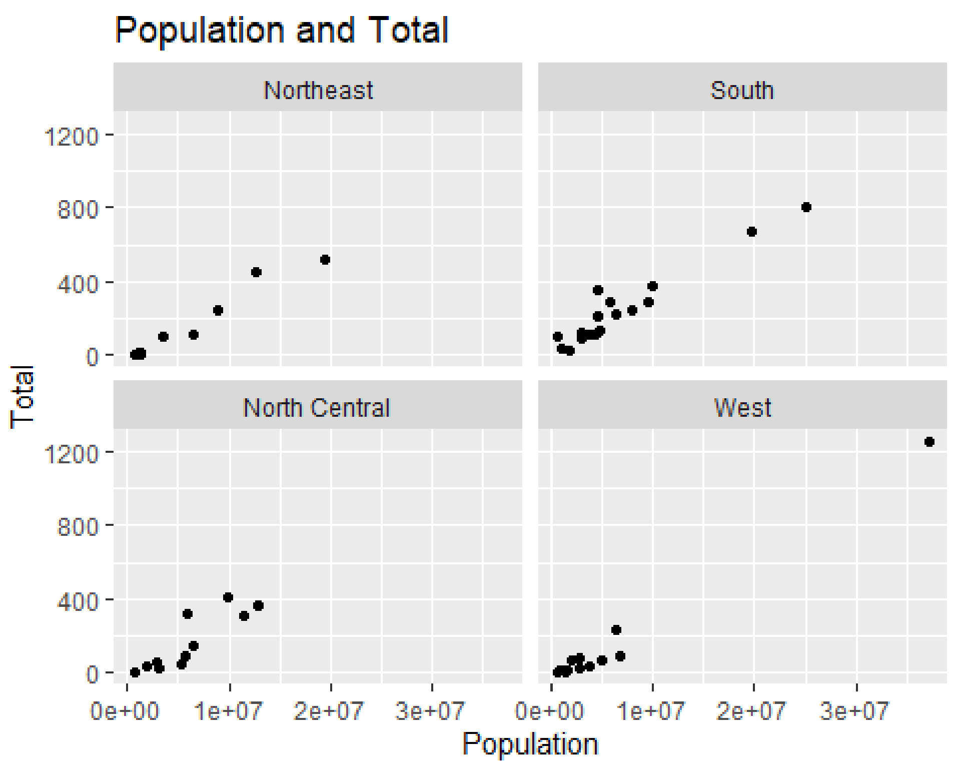


Figure 12: Scatterplots for the populations versus total gun murders for each state, with a scatterplot for each region for 2010 US Gun Murders data

The above plots demonstrate the total murders versus the population for a state, with a different plot for each region. Doing this, we do not need to provide different colors to differentiate the data points for each region in a single scatterplot. While there can be a larger amount of plots, whether that is the best means to display the data is to be evaluated by the user. Using the `facet_wrap()` or `facet_grid` options can assist with providing more options for how to display data visualizations.

Another set of demonstrations can be seen after loading the mtcars dataset. The data was obtained from a 1974 edition magazine *Motor Trend*. There are 32 rows and 11 columns in this dataset to use. The first plot will be of miles per gallon to displacement.

```
data("mtcars")

ggplot(mtcars, aes(x = disp, y = mpg)) +

  geom_point() +

  xlab("Displacement (Cubic Inches)") +

  ylab("Miles Per Gallon (MPG)") +

  ggtitle("Displacement and Miles Per Gallon")
```

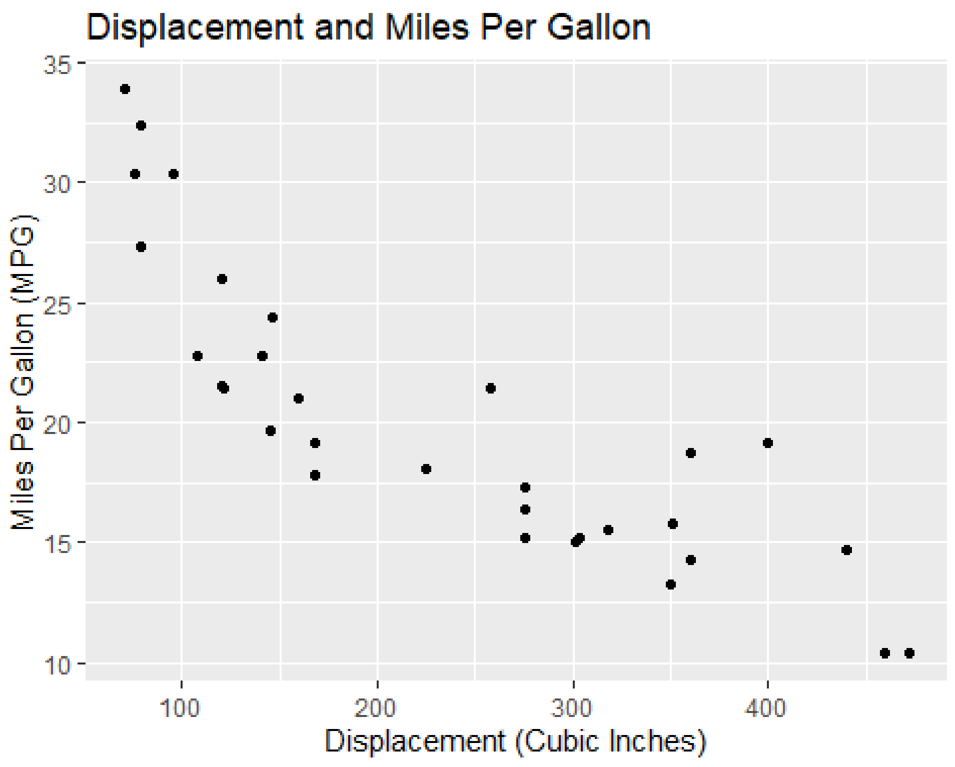


Figure 13: Displacement versus miles per gallon scatterplot for Motor Trend data

The scatterplot above shows a general trend that as the displacement increases, the miles per gallon for the vehicle will decrease. The lowest miles per gallon vehicles in the dataset have a displacement greater than 450 cubic inches. This can be further demonstrated using the `geom_smooth()` option plot a linear regression model.

```
ggplot(mtcars, aes(x = disp, y = mpg)) +
  geom_point() +
  geom_smooth(se=FALSE,method = "lm") +
  xlab("Displacement (Cubic Inches)") +
  ylab("Miles Per Gallon (MPG)") +
  ggtitle("Displacement and Miles Per Gallon")
```

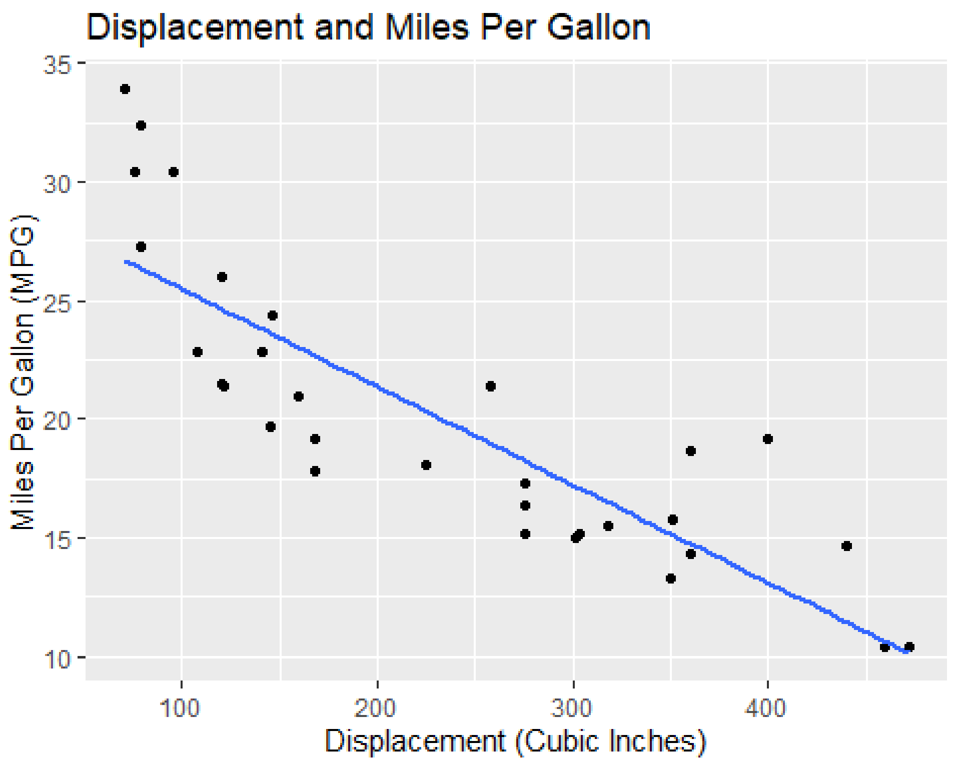


Figure 14: Displacement versus miles per gallon scatterplot for Motor Trend data with a linear model added

After seeing the linear model being plotted with the data points, we can observe that a negative relationship exists between the displacement and miles per gallon of a vehicle. The linear model also be used to estimate the miles per gallon for a vehicle given the displacement. If additional data was added, those points would be used in generating the linear model as well. If a more flexible model is needed, the other methods for `geom_smooth` can be used. The next method to be demonstrated will be `loess`.

```
ggplot(mtcars, aes(x = disp, y = mpg)) +
  geom_point() +
  geom_smooth(se=FALSE,method = "loess") +
  xlab("Displacement (Cubic Inches)") +
  ylab("Miles Per Gallon (MPG)") +
  ggtitle("Displacement and Miles Per Gallon")

## `geom_smooth()` using formula 'y ~ x'
```

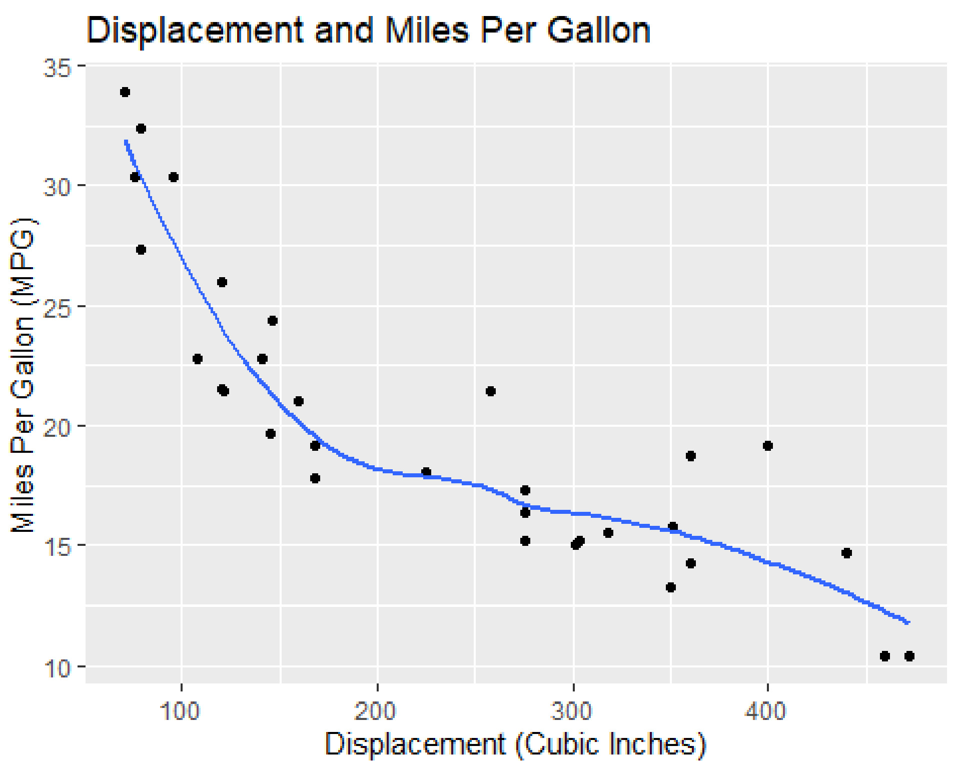


Figure 15: Displacement versus miles per gallon scatterplot for Motor Trend data with a loess model added

The `loess` method gives a model that is influenced much easier by the data points available. We can display additional data as well through the use of colors on the scatterplot without the models. This way observations can be made about the shape of the engine or the number of cylinders in the vehicle.

```
ggplot(mtcars, aes(x = disp, y = mpg, color=factor(cyl))) +
  geom_point() +
  xlab("Displacement (Cubic Inches)") +
  ylab("Miles Per Gallon (MPG)") +
  ggtitle("Displacement and Miles Per Gallon") +

  labs(color = "Cylinders")
```

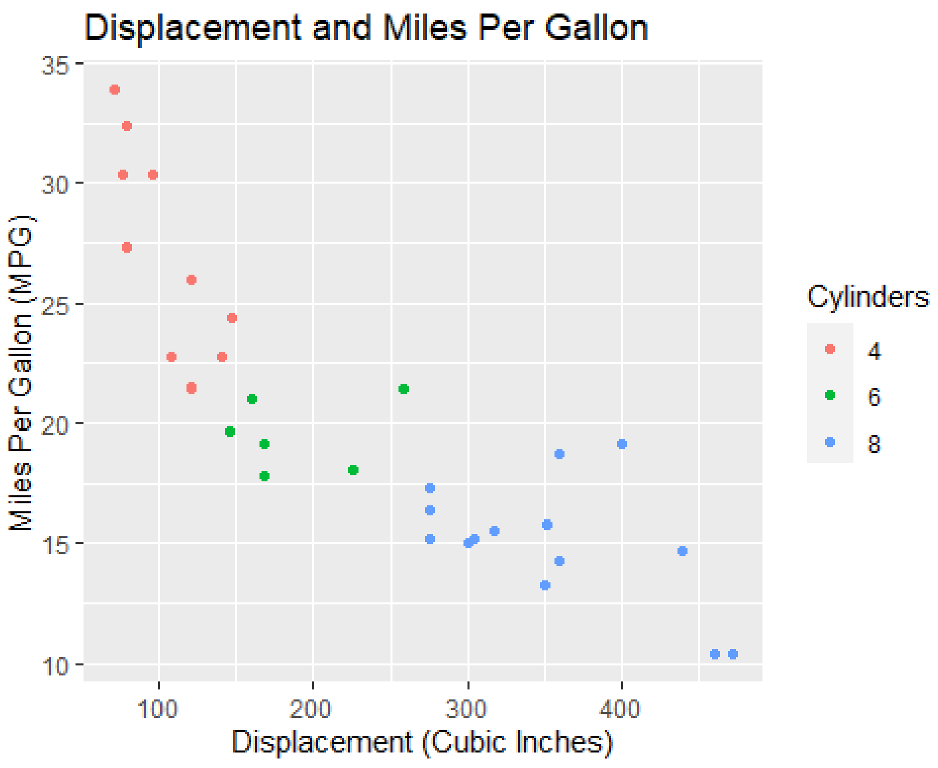


Figure 16: Displacement versus miles per gallon scatterplot for Motor Trend data with different colors for each grouping of cylinders for vehicles in the Motor Trend data

It is observable that there are clear divisions in the groups of cylinders in the plot. The 4 cylinders froup have the highest miles per gallon cars and the least displacement out of the three different groupings for cylinders. The 6 cylinders group is midway between the 4 cylinders group and the 8 cylinders group. The 8 cylinders group seem have the largest amount of displacement and the least miles per gallon. Thanks to the colors that are now separating the groupings we can observe this difference of cylinders in the plot.

We make a separate plot grouping the engine shapes:

```
ggplot(mtcars, aes(x = disp, y = mpg, color=factor(vs))) +
  geom_point() +
  xlab("Displacement (Cubic Inches)") +
  ylab("Miles Per Gallon (MPG)") +
  ggtitle("Displacement and Miles Per Gallon") +
  labs(color = "Engine Shape")
```

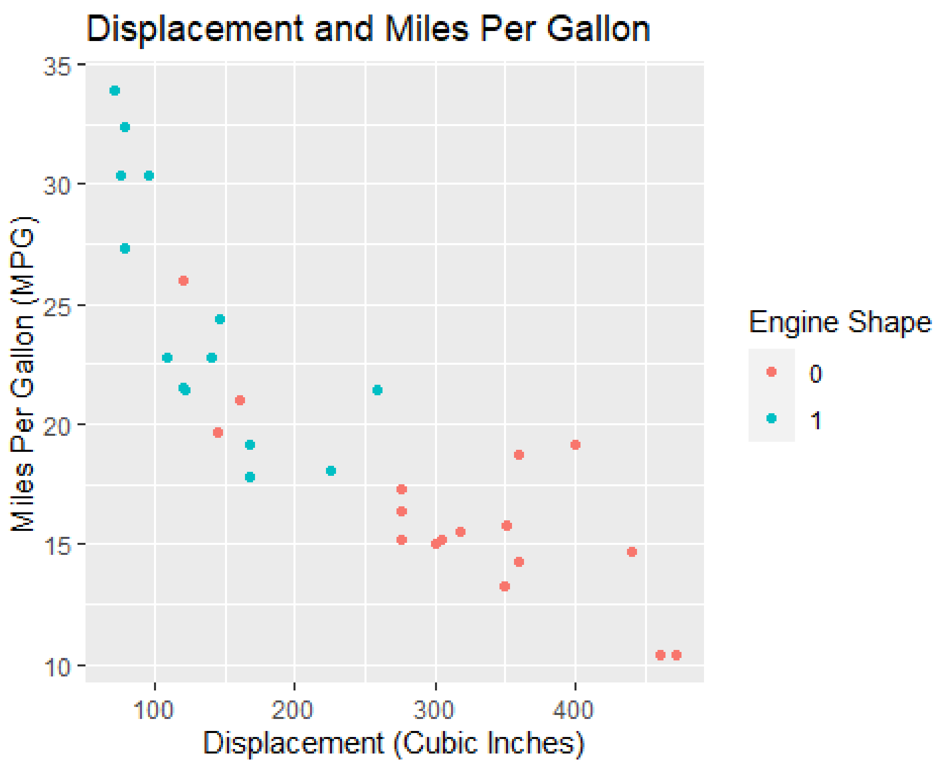


Figure 17: Displacement versus miles per gallon scatterplot for Motor Trend data with different colors for each grouping of engine shapes for vehicles in the Motor Trend data

Looking at the engine shape, the separation of the groups is not as straightforward as with the number of cylinders. It might be better to display a plot for each engine shape:

```
ggplot(mtcars, aes(x = disp, y = mpg)) +
  geom_point() +
  xlab("Displacement (Cubic Inches)") +
  ylab("Miles Per Gallon (MPG)") +
  ggtitle("Displacement and Miles Per Gallon") +
  facet_wrap(~factor(vs), scales = "fixed")
```

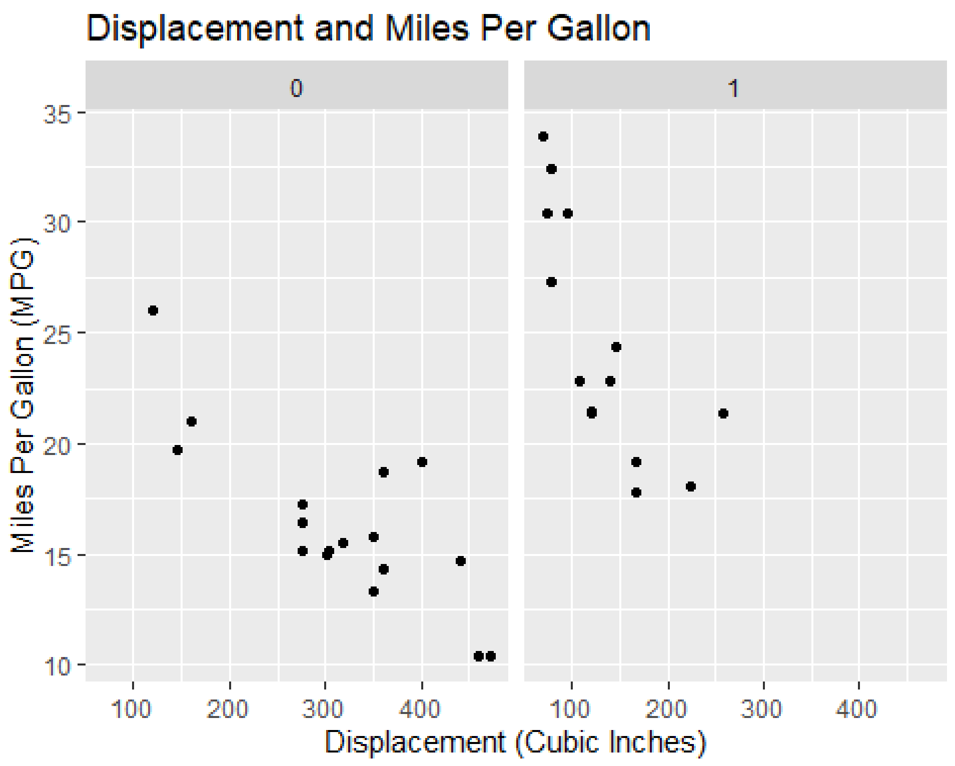


Figure 18: Displacement versus miles per gallon scatterplot for Motor Trend data with different colors for each grouping of engine shapes for vehicles in the Motor Trend data

Thanks to the flexibility of ggplot2, depending on our goal we will be able to use different graphics to be able to display the data. If we want to simply show the groupings of the engine shapes, coloring the data points based on engine shapes may be a good starting point. If we want to separate the data based on engine shapes then compare the two, that is also possible. The `facet_wrap()` and `facet_grid()` are options for organizing plots that may be burdened with too much data or to cut down on the amount lines of R code that need to be written to perform similar tasks.

If we wanted to see if there was a relationship between the number of cylinders and displacement, boxplots can be drawn:

```
ggplot(mtcars, aes(x = factor(cyl), y = disp)) +
  geom_boxplot() +
  xlab("Number of Cylinders") +
  ylab("Displacement (Cubic Inches)") +
  ggtitle("Displacement and Number of Cylinders")
```

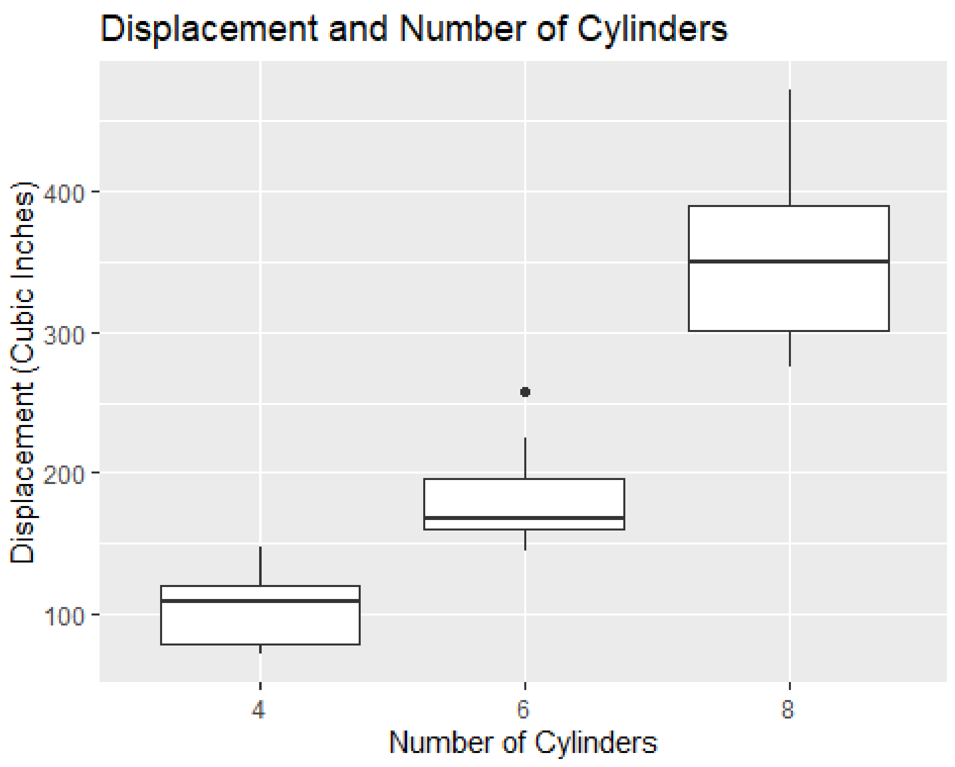


Figure 19: Box plots of the distributions of displacement of vehicles by each grouping of the number of cylinders for the vehicles in the data

We again see that as the number of cylinders increases, the distributions move to higher values. We can see that for four cylinders, the distribution is skewed to the right. For six cylinders, the distribution is skewed left with one possible outlier, and for eight cylinders the distribution is slightly skewed right. Thanks to the boxplots we can see the skewness of the distributions, the minimums, first quartiles, medians, third quartiles and the maximums of the distributions of displacement for each group of cylinders.

4.3 Handling Data in R

If we want to look at two categorical variables, like engine shape and the number fo cylinders, we can use a stacked bar plot. To do this, we use the loaded R libraries to count the number of data points that satisfy each possible group for the variables vs and cyl in the mtcars dataset. For example, one possible group is a V-Shaped engine with four cylinders.

```r
library(magrittr)


engine.cyl <- mtcars %>%
  mutate(vs = ifelse(vs ==0, "V-shaped", "straight"),
         cyl = factor(cyl)) %>%
  group_by(vs, cyl) %>%
  count()


ggplot(engine.cyl, aes(x=vs, y = n, fill = cyl)) +
  geom_bar(position="stack", stat="identity") +
  labs(fill = "Cylinders") +
  xlab("Engine Shape") +
  ylab("Count") +

  ggtitle("Engine Shape and Number of Cylinders")
```

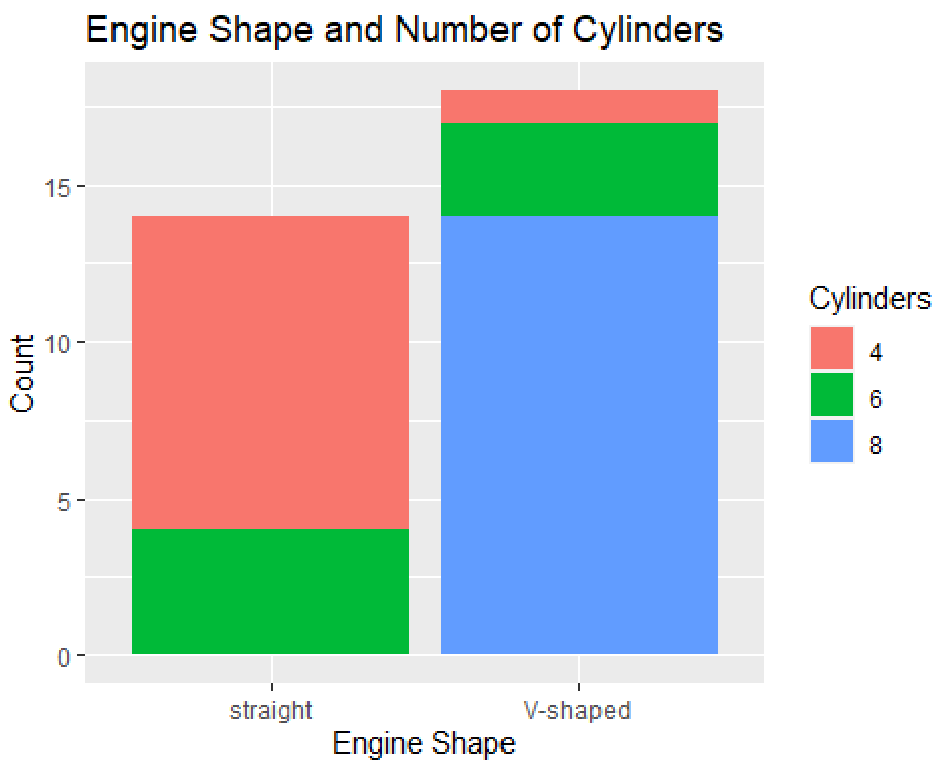


Figure 20: Stacked bar graph engine shapes with the bars colored to measure the number of each cylinder grouping for each engine shape

The bar plot shows that there for a straight engine, there are mostly 4 cylinders with a few 6 cylinders vehicles. For the V-shaped engines, there are mostly 8 cylinders, far fewer six cylinders, and only a few four cylinders vehicles. This would not have been as clear since the original data did not have counts for these groups. In the plot we can see the counts for these different groups based on engine shape and number of cylinders. One important detail that can be pointed out here is that the mtcars dataset has no 8 cylinder vehicle with a straight engine.

Another implementation would be to use dodge instead of stacked for the purpose of looking at the individual cylinder group counts with respect to the engine shape:

```
ggplot(engine.cyl, aes(x=vs, y = n, fill = cyl)) +
  geom_bar(position="dodge", stat="identity") +
  labs(fill = "Cylinders") +
  xlab("Engine Shape") +
  ylab("Count") +
  ggtitle("Engine Shape and Number of Cylinders")
```

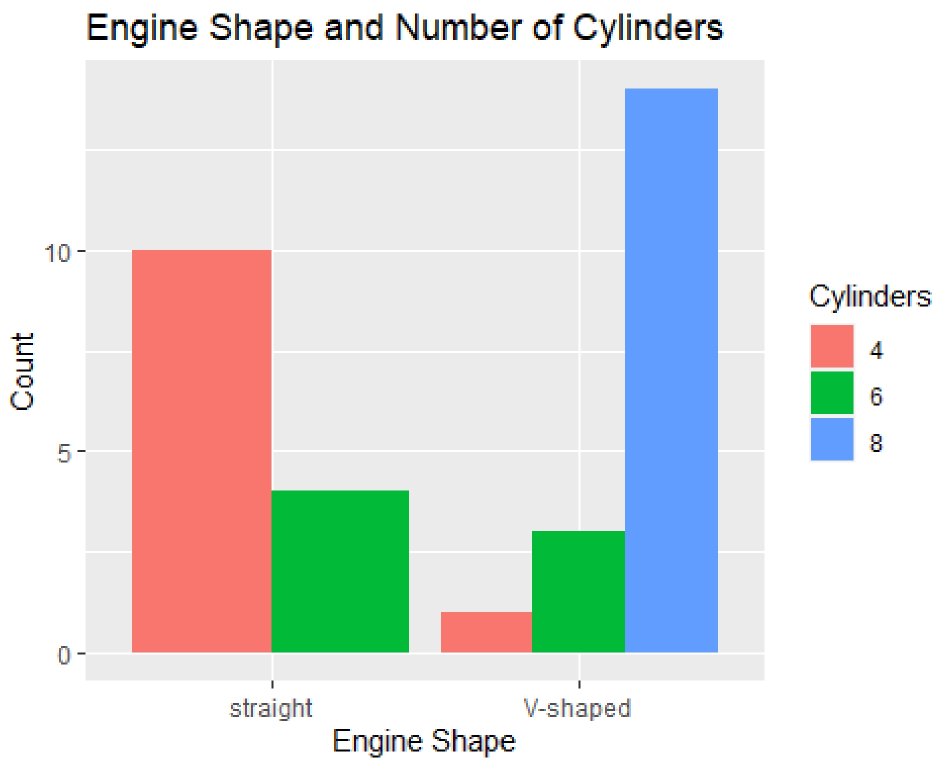


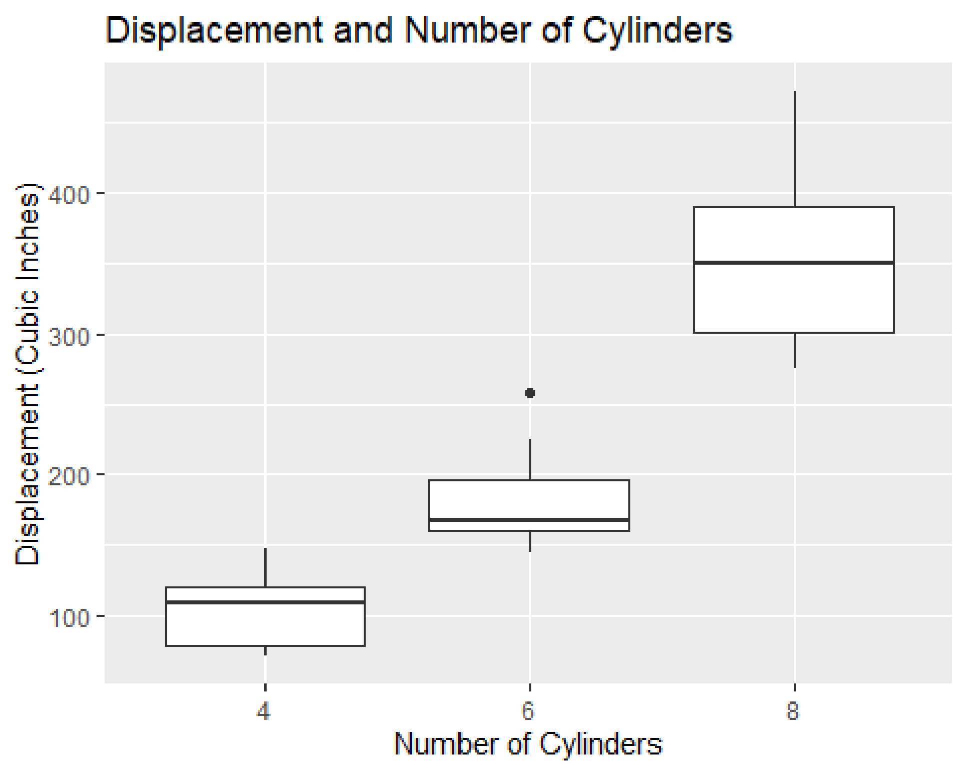
Figure 21: Stacked bar graph of engine shapes with the bars colored and spread for each grouping of cylinders in the grouping of engine shapes

Looking at the edited plot, most of the data points are V-shaped engines with eight cylinders.

Demonstrated previously is the ability to get the data needed for new plots or operations form existing data. R's packages can simplify the process of getting or calculating data from datasets. This includes the data that is present but not explicitly given a single distinct variable. Data that has a lot of variables can be troublesome to work with. If a few variables explain the data well enough, that should be sufficient. For example, a set of columns for counting the groupings of the engine shape and the number of cylinders can add additional unneeded columns to the dataset. This is because the same information can be calculated from the vs and cyl variables that are already present within the dataset. Simply adding more columns to the existing dataset would do little to provide additional information in a simple and efficient manner.

We can go back to the previous box plot earlier and get the values of the minimum, first quartile, median, third quartile, and maximum:

```
ggplot(mtcars, aes(x = factor(cyl), y = disp)) +
  geom_boxplot() +
  xlab("Number of Cylinders") +
  ylab("Displacement (Cubic Inches)") +
  ggtitle("Displacement and Number of Cylinders")
```



```
mtcars %>%

  mutate(cyl = factor(cyl)) %>%

  group_by(cyl) %>%

  summarize(minimum = min(disp), Q1 = quantile(disp,probs=.25),

            median = median(disp), Q3 = quantile(disp,probs=.75),

            maximum =max(disp))

## # A tibble: 3 x 6

##   cyl   minimum    Q1 median    Q3 maximum

##   <fct>   <dbl> <dbl>  <dbl> <dbl>   <dbl>

## 1 4        71.1  78.8   108   121.    147.
## 2 6         145    160      168.   196.     258
## 3 8         276.   302.     350.   390      472
```

Table 1: The minimum, first quartile, median, third quartile, and maximum for all three groups of cylinders in the Motor Trend data

Now we can see the five number summary for the cylinders and displacement box plot. Additional operations can be done using the values of the five number summary for each grouping the of the number of cylinders. These values also give a bit more context overall to the box plot. More values like the mean and standard deviation can be calculated as well to view more concerning the distributions of the data. This is demonstrated in the dataframe below:

```
mtcars %>%
  mutate(cyl = factor(cyl)) %>%
  group_by(cyl) %>%
  summarize(minimum = min(disp), Q1 = quantile(disp,probs=.25),
            median = median(disp), Q3 = quantile(disp,probs=.75),
            maximum =max(disp), avg = mean(disp), stdev = sd(disp))

## # A tibble: 3 x 8
##   cyl   minimum    Q1 median    Q3 maximum   avg stdev
##   <fct>   <dbl> <dbl>  <dbl> <dbl>   <dbl> <dbl> <dbl>
## 1 4        71.1  78.8   108   121.    147.  105.  26.9
## 2 6        145    160      168.   196.     258    183.   41.6

## 3 8        276.   302.     350.   390      472    353.   67.8
```

Table 2: The minimum, first quartile, median, third quartile, maximum, average, and standard deviation for all three groups of cylinders in the Motor Trend data

The addition of the mean and standard deviation provides additional details relevant to the distribution. Using the summarize function, we were able to create the dataframe above. Each grouping of the number of cylinders is a row and each statistic is a column. The function `summarize` has other variants as well to increase the flexibility of its usage. This allows for simpler means of calculating statistics of data than might be observed solely from using the `summarize` function.

Dataframes are a data structure that are consist of rows and columns, where each column is named. The dataframe is often used when accessing data in R and has a number of packages that can already support data frames and operations an R user may want to use. Dataframes can allow for data to be cleanly viewed as well. Due to the conventions concerning dataframes, they are popular amongst the R community and the statistics community.

We can observe another instance of stacked bar graph usage by looking at the groupings of the number of forward gears and the number of carburetors in vehicles. With the proposed stacked bar graph, we look at the different groupings between the number of forward gears and the number of carburetors for the vehicles in the dataset.

```
gears.carbs <- mtcars %>%

  mutate(gear = factor(gear),

         carb = factor(carb)) %>%
  group_by(gear, carb) %>%
  count()


ggplot(gears.carbs, aes(x=gear, y = n, fill = carb)) +
  geom_bar(position="stack", stat="identity") +
  labs(fill = "Number of Carburetors") +
  xlab("Number of Forward Gears") +
  ylab("Count") +
  ggtitle("Number of Forward Gears and Number of Carburetors")
```

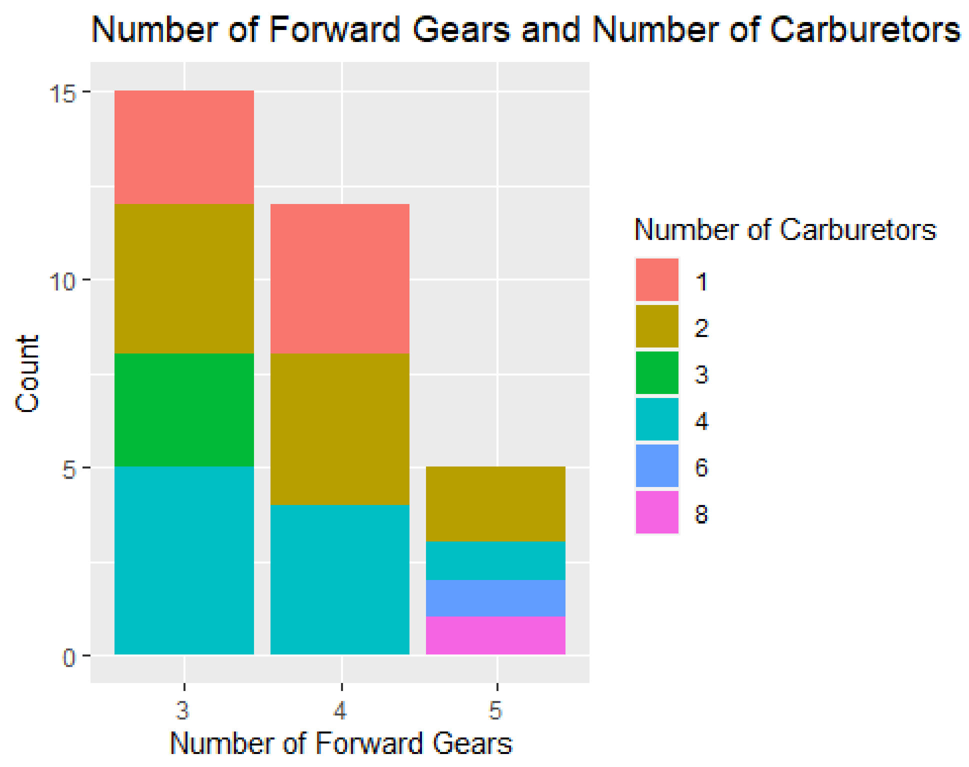


Figure 22: Stacked bar graph of the number of forward gears for the vehicles, with different colors for the count of the number of carburetors for the vehicles

It is easily observable that for the four forward gears group there are only vehicles with 1, 2, or four carburetors. For the three forward gears group, there are vehicles with 1, 2, 3, and 4 carburetors in the mtcars dataset. For the five forward gears group there are only vehicles with 2, 4, 6, and 8 carburetors. Groupings for 2 and four carburetors exist for each grouping of forward gears in the mtcars dataset. The 1 carburetor group only exists for the groupings of 3 and 4 forward gears in the mtcars dataset. The 6 and 8 carburetors groups only exist for for the five forward gears group in the mtcars dataset.

The flexibility that ggplot provides cannot be understated. Its presence extends the overall abilities of R users to be able to make attention-getting visualizations that can accurately display the relationships that exist within data. Multiple visualizations can be produced allowing for multiple ways to be able to interpret the data. Likely the best part is that ggplot2 provides a broad amount of options to be able to suit the needs of the newcomers to R and the expert users or R. Like the other packages, ggplot2 can be expected to be maintained and updated to continue to appeal to and serve the R community. For students, this can allow for students to enhance presentations within the classroom and for instructors to allow for students to experiment with possible data visualizations on their own or within designated groups for group projects. It can provide students with more tools to be able to use for semester projects, theses, final reports and the like that can contribute both to the statistics courses and courses beyond statistics.

Chapter 5: Homework Assignments

In this Chapter, we go over three homework assignments and solutions for a statistics course using R.

Assignment 1

Questions

Question 1: Provide a definition of statistics.

Question 2: Write out an equation for calculating the mean of a population.

Question 3: Write out an equation for calculating the standard deviation of a population.

Question 4: Calculate the mean of the following numbers with R: 1, 23, -44, 55, 67, 87, 99, -77, 15, 99, 15, 27, 15.

Question 5: Calculate the standard deviation of the following numbers with R: 1, 23, -44, 55, 67, 87, 99, -77, 15, 99, 15, 27, 15.

Question 6: Calculate the minimum and maximum of the following numbers with R: 1, 23, -44, 55, 67, 87, 99, -77, 15, 99, 15, 27, 15.

Question 7: Calculate the first quartile and the third quartile of the following numbers with R: 1, 23, -44, 55, 67, 87, 99, -77, 15, 99, 15, 27, 15.

Question 8: Calculate the median of the following numbers with R: 1, 23, -44, 55, 67, 87, 99, -77, 15, 99, 15, 27, 15.

Question 9: Calculate the median and mean of the following numbers: 0, 23, -44, 55, 65, 87, 1000, -77, 15, 99, 15, 27, 15. Did the median change? Did the mean change? If the median or mean changed, why?

Solutions

Question 1: Provide a definition of statistics.

Students should be able to provide an answer similar or equivalent to: statistics is the field of being able to collect, manage, manipulate, process, analyze, interpret and visualize data from a population. Students can use a definition that captures the same meaning of the provided definition. Students may also use a definition that is taken directly from the course notes or texts. The purpose of this question is to make sure students have a basic understanding of what statistics is before continuing forward with the course work.

Question 2: Write out an equation for calculating the mean of a population.

mean $= \frac{1}{n}\sum_{i=1}^{n} x$

Question 2 is to get students to practice writing equations in R through the use of Latex. This allows for statistical formulas to be able to be written by students or mathematical proofs to be derived within the same report that the R code is present in. This does not require full knowledge of using Latex. If there are questions as to Latex syntax, Latex documentation can be referred to provide students with adequate information to complete assignments. The solution presented is one version of the solutions that may be presented, but the important part is that the mathematical formula is correct. Instructors can specify syntax for the class.

Question 3: Write out an equation for calculating the standard deviation of a population.

standard deviation $= \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n}}$

Question 3 is once again to allow students to practice writing equations in the R Markdown document that all work can be typed and saved in. The students can get credit if there is some similar form of population standard deviation formula typed. If students type the standard deviation of a sample, they get no credit as that is not what was asked in the question.

Question 4: Calculate the mean of the following numbers with R: 1, 23, -44, 55, 67, 87, 99, -77, 15, 99, 15, 27, 15.

```
x <- c(1, 23, -44, 55, 67, 87, 99, -77, 15, 99, 15, 27, 15)
mean(x)

## [1] 29.38462
```

For calculating the mean, students should use the mean function provided in R unless otherwise specified by the instructor. The answer given should be 29.385, rounded up to the third decimal place. The R code and its output should be on full display for evaluation by the instructor. Students should store the numbers in an object for future reference.

Question 5: Calculate the standard deviation of the following numbers with R: 1, 23, -44, 55, 67, 87, 99, -77, 15, 99, 15, 27, 15.

```
sd(x)

## [1] 52.64589
```

The standard deviation is approximately 52.646, rounded to the third decimal place. For calculating the standard deviation, students should use the sd function in R unless specified by the instructor. The R code and output should be on full display for the instructor to evaluate.

Question 6: Calculate the minimum and maximum of the following numbers with R: 1, 23, -44, 55, 67, 87, 99, -77, 15, 99, 15, 27, 15.

```
min(x)

## [1] -77

max(x)

## [1] 99
```

The minimum is -77 and the maximum is 99. For calculating the minimum and maximum, students should use the min and max functions respectively in R unless specified by the instructor. The R code and output should be on full display for the instructor to evaluate.

Question 7: Calculate the first quartile and the third quartile of the following numbers with R: 1, 23, -44, 55, 67, 87, 99, -77, 15, 99, 15, 27, 15.

```
quantile(x, probs = c(.25,.75))

## 25% 75%
##  15  67

quantile(x, probs = c(.25))

## 25%
##  15

quantile(x, probs = c(.75))
```

```
## 75%
##   67
```

The minimum is 15 and the maximum is 67. For calculating the first quartile and third quartile, students should use the quantile function in R unless specified by the instructor. The R code and output should be on full display for the instructor to evaluate.

Question 8: Calculate the median of the following numbers with R: 1, 23, -44, 55, 67, 87, 99, -77, 15, 99, 15, 27, 15.

```
median(x)

## [1] 23
```

The median is calculated to be 23. The median should be calculated using the median function in R, unless specified by the instructor. The R code and output should be on full display for the instructor to evaluate.

Question 9: Calculate the median and mean of the new set of numbers: 0, 23, -44, 55, 65, 87, 1000, -77, 15, 99, 15, 27, 15. Did the median change? Did the mean change? If the median or mean changed, why?

```
y <- c(0, 23, -44, 55, 65, 87, 1000, -77, 15, 99, 15, 27, 15)
median(y)

## [1] 23

mean(y)

## [1] 98.46154
```

The resulting median and mean are respectively 23 and 98.462, rounded to three decimal places, for the new set of numbers. The median remained the same but mean decreased as it is not resistant to outliers. This is a good exercise for teaching students an early introduction to the differences of the median and mean. The median being resistant to outliers whereas the mean is not.

Assignment 2

Questions

Load the murders dataset and tidyverse using:

```
library(tidyverse)

library(dslabs)

data(murders)
```

Question 1: Print the first five rows of the murders dataframe with the `head()` function. What are the names of the first five states that appear and their respective populations?

Question 2: Using the `summarize` function in R, get the five number summary (minimum, first quartile, median, third quartile, maximum) for the population data in the murders dataset.

Question 3: Using the mutate function, add a variable called rate that states the gun murder rate per 100000 people. Store the new dataframe in an object called `murders2`. Print the first three rows showing that they include the rate variable.

Question 4: Make a scatter plot of the population versus the rate for each state.

Question 5: Make a box plot on the distribution of the murder rates by region. How many possible outliers are there per region?

Question 6: Using the plot from Question 5 which region has the highest median rate? Which region has the lowest median rate?

Question 7: Are the distributions of gun murder rates for the southern and western regions skewed, and if so, in which direction? Justify using the plot from Question 5.

Question 8: For the rate variable, would a bar plot or density plot be more appropriate for plotting the probability distribution? Explain your reasoning.

Solutions

Question 1: Print the first five rows of the murders dataframe with the `head()` function. What are the names of the first five states that appear and their respective populations?

```
head(murders,5)

##         state abb region population total

## 1    Alabama  AL  South    4779736   135

## 2     Alaska  AK   West     710231    19

## 3    Arizona  AZ   West    6392017   232

## 4   Arkansas  AR  South    2915918    93

## 5 California  CA   West   37253956  1257
```

Table 3: Printing first five states of the murders data

The first five states are Alabama, Alaska, Arizona, Arkansas, and California. Their respective populations are 4779736, 710231, 6392017, 2915918, and 37253956. The importance in this exercise is to get students to look into data before working with it. This gives students an opportunity to understand the data and if any preprocessing needs to be done beforehand. It also allows students to see what they want to pay attention to in the data.

Question 2: Using the `summarize` function in R, get the five number summary (minimum, first quartile, median, third quartile, maximum) for the population data in the murders dataset.

```
murders %>%

  summarize(minimum = min(population), Q1 = quantile(population, probs
= .25),
            median = median(population), Q3 = quantile(population, pro
bs = .75),
            maximum = max(population))

##   minimum      Q1  median      Q3  maximum
## 1  563626 1696962 4339367 6636085 37253956
```

Table 4: Five number summary of the murders data

The minimum is 563626, the first quartile is 1696962, the median is 4339367, the thrid quartile is 6636085, and the maximum is 37253956. The population variable should be used in the calculations, not the total variable. This exercise reinforces the use of the five number summary and the usage of summarize to simplify the output into one table.

Question 3: Using the mutate function, add a variable called rate that states the gun murder rate per 100000 people. Store the new dataframe in an object called murders2. Print the first three rows showing that they include the rate variable.

```
murders2 <- murders %>%
  mutate(rate = total/population * 100000)
head(murders2,3)

##     state abb region population total     rate
## 1 Alabama  AL  South    4779736   135 2.824424
## 2  Alaska  AK   West     710231    19 2.675186
## 3 Arizona  AZ   West    6392017   232 3.629527
```

The code the students submit should show the that the rate variable has been added in the `murders2` object. The formula used should be (total / population * 100000) or a similar formula. The resulting dataframe printed should be similar to the solution presented above.

Question 4: Make a scatterplot of the population versus the rate for each state.

```
q <- ggplot(murders2,aes(x=population, y= rate)) +
  geom_point() +
  ggtitle("Population Versus Gun Murder Rate") +
  xlab("Population") + ylab("Rate")
q
```

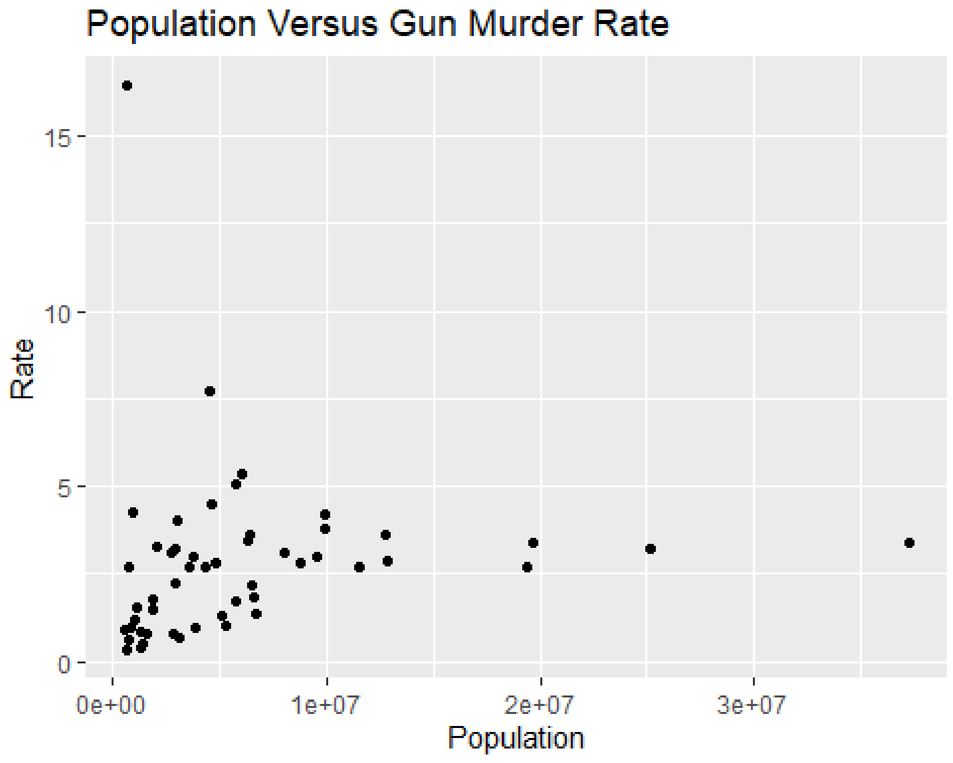


Figure 23: Population Versus Gun Murder Rate

The scatterplot should be created using ggplot2, and specifically the `geom_scatter()` option. The code can be written multiple ways, but the resulting scatterplot should be similar. The plot should have a title with the population assigned to the x-axis and rate to the y-axis. This exercise is to start building students familiarity with ggplot. Students can relabel the x-axis and y-axis, but that should not be a necessity. Instructors are given the opportunity to see if students can accurately select the correct graphing option in ggplot2 to use and see if students can provide the correct inputs.

Question 5: Make a boxplot on the distribution of the murder rates. Are there any outliers?

```
box <- ggplot(murders2, aes(x = region, y=rate)) +
  geom_boxplot() +
  ggtitle("Distribution of Murder Rates") +
  ylab("Rate")
box
```

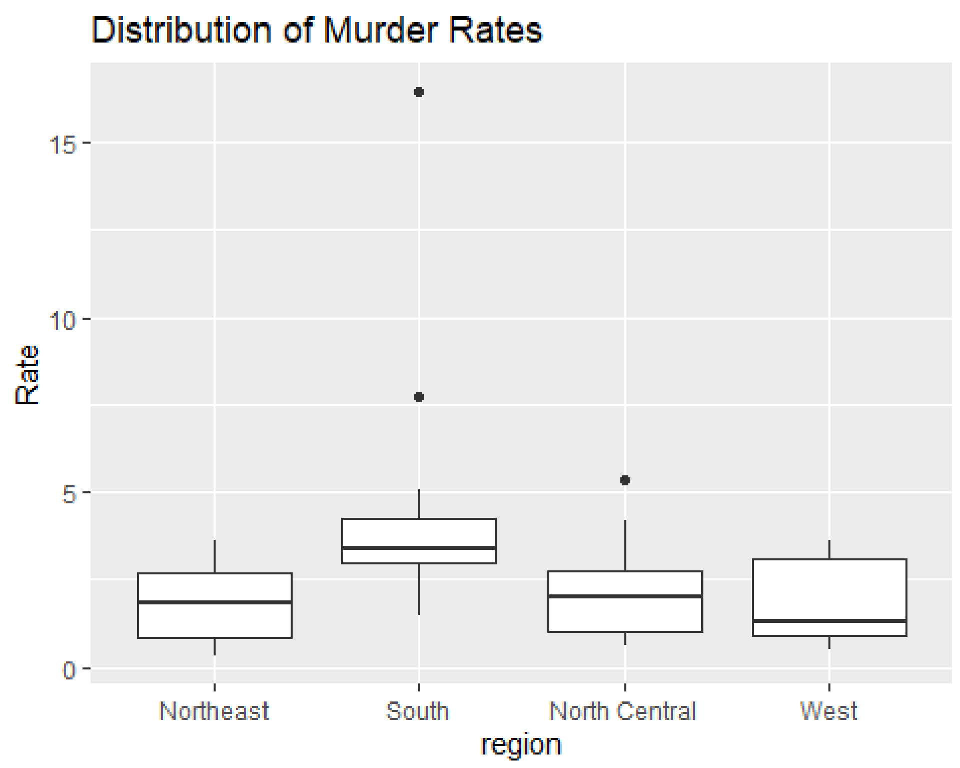


Figure 24: Box plots of the distribution of murder rates by region

Students are able to practice the usage of ggplot and familiarizing themselves with the right graph to use for different use cases. There appear to be two outliers in the southern region and one in the north central region. Students should be able to demonstrate the usage of `geom_boxplot()`. A plot similar to the one above should be observed in the students' code output.

Question 6: Using the plot from Question 5 which region has the highest median rate? Which region has the lowest median rate?

Students should not just reprint the plot for Question 5 as the answer of Question 6. Students should be able to identify from the plot that the region with the highest median rate is the south.

The region with the lowest median rate is the west. In this exercise, students are practicing analysis of their own work, specifically the use of data visualizations to understand the data.

Question 7: Are the distributions of gun murder rates for the southern and western regions skewed, and if so, in which direction? Justify using the plot from Question 5.

Students should be able to state that the gun murder rates for the southern and western regions are left skewed. Looking at the median's position in comparison to the rest of the box plot for both the southern and western regions, the median is at the lower end of the box plot. The resulting answers will inform instructors of students comprehension of the material and the ability to apply knowledge of said material.

Question 8: For the rate variable, would a bar plot or density plot be more appropriate for plotting the probability distribution? Explain your reasoning.

The density plot should be used as rate is a continuous variable. Bar plots are used for discrete probability distributions and bar plots utilize discrete variables. The student's answer should be a response similar what was stated in the previous two sentences. In identifying whether the bar plot or the density plot would be more effective, students are demonstrating application of knowledge of probability distributions and the best way to visualize such. This allows instructors to be able to see if students understand which plots can be used for discrete or continuous probability distributions.

Assignment 3

Load the murders dataset and tidyverse using:

```
library(tidyverse)

library(dslabs)

data(murders)
```

Questions

Question 1: Define correlation and how do we interpret it?

Question 2: Write the formula for the Pearson Correlation Coefficient.

Question 3: Let there be two vectors `x` and `y`. Let `x` = [1, 10, 42, 55, 17, 33, 55, 57, 58, 64]. Let `y` = [-1, 16, 10, 2, 8, 32, 28, 19, 40, 57]. Find the correlation using the `cor` function.

Question 4: Using the mutate function, add a variable called rate that states the gun murder rate per 100000 people. Store the new dataframe in an object called `murders2`. Make a scatterplot of population versus gun murders rate. Does there seem to be a positive of negative correlation between population and gun murders rate? Verify this with the cor function.

Question 5: Construct a box plot for the rate variable by region. Use `geom_jitter()` display the individual data points.

Question 6: Construct a density plot for the rate variable by region. Use the `fill` argument of ggplot aesthetics.

Question 7: What is a weakness of box plots?

Solutions

Question 1: Define correlation and how do we interpret it?

Students should be able to state something along these lines: Correlation is a statistical relationship that exists between two variables. If the correlation is greater than or equal to -1 but less than 0, the variables have a negative relationship. If the correlation is less than or equal to 1 but greater than 0, the variables have a positive relationship. If the correlation is 0, the variables have no relationship.

The answers will provide insight to instructors if the students are reading the course notes and understanding the terminology. It can also provide information as to whether the students have enough comprehension to to elaborate on the topic of correlation.

Question 2: Write the formula for the Peearson Correlation Coefficient.

$$r = \frac{n(\sum xy) - (\sum x)(\sum y)}{\sqrt{(n\sum x^2 - (\sum x^2))(n\sum y^2 - (\sum y)^2)}}$$

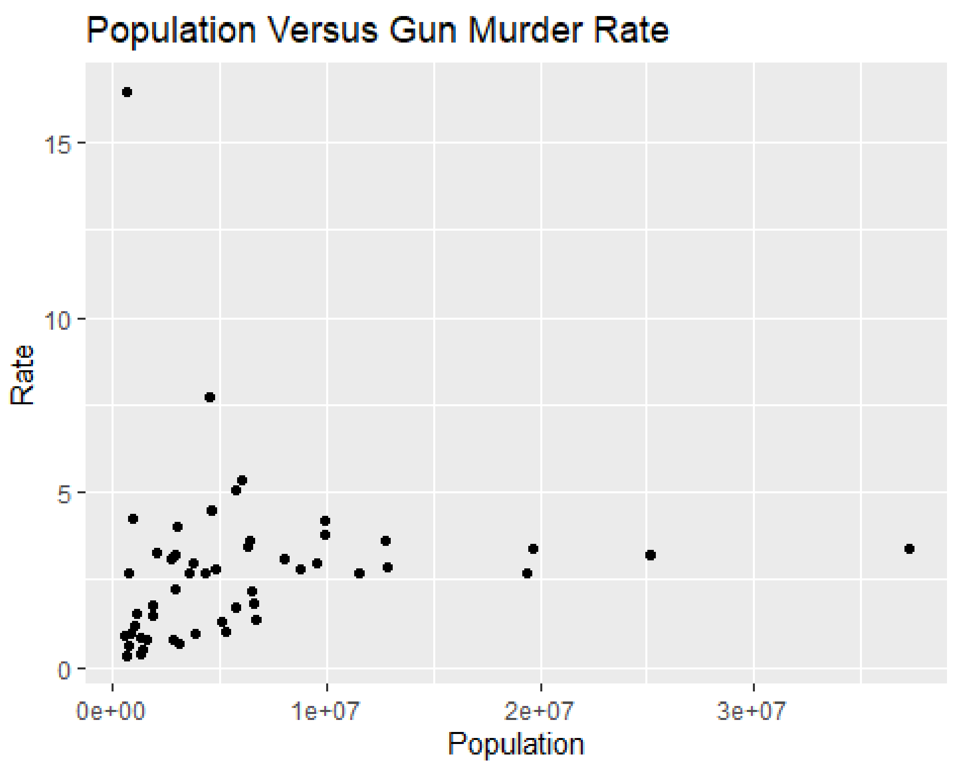
Question 3: Let there be two vectors x and y. Let x = [1, 10, 42, 55, 17, 33, 55, 57, 58, 64]. Let y = [-1, 16, 10, 2, 8, 32, 28, 19, 40, 57]. Find the correlation using the `cor` function.

```
x <- c(1, 10, 42, 55, 17, 33, 55, 57, 58, 64)
y <- c(-1, 16, 10, 2, 8, 32, 28, 19, 40, 57)
cor(x,y)

## [1] 0.5910533
```

Students should be able to easily apply the `cor` function to find the answer to this question. The answer is 0.591, rounded to three decimal places. The code and the code output should be fully displayed for evaluation. Instructors will be able to see if students can properly use R's built-in `cor` function.

Question 4: Using the mutate function, add a variable called rate that states the gun murder rate per 100000 people. Store the new dataframe in an object called murders2. Make a scatterplot of population versus gun murders rate. Does there seem to be a positive of negative correlation between population and gun murders rate? Verify this with the `cor` function.

```
murders2 <- murders %>%
  mutate(rate = total/population * 100000)


q <- ggplot(murders2,aes(x=population, y= rate)) +
  geom_point() +
  ggtitle("Population Versus Gun Murder Rate") +
  xlab("Population") + ylab("Rate")
q
```



```
cor(murders2$population, murders2$rate)

## [1] 0.0939413
```

There is a weak positive correlation between the population and rate variables. This is verified by the result of the `cor` function, which is 0.094 rounded to three decimal places. Students will demonstrate the ability to be able to pull specific data from tables/dataframes. This is crucial for being able to work with datasets as not all functions will be designed to process the entire dataset. Another learning point for students is that for some tasks or assignments, only a few variables will be needed. If students have errors caused by inputting the dataset, likely students were not looking at the help file on the `cor` file, as the `cor` function accepts vectors not whole datasets.

Question 5: Construct a boxplot for the rate variable by region. Use `geom_jitter()` display the individual data points.

```
ggplot(murders2, aes(region, rate)) +
  geom_boxplot() +
  ggtitle("Gun Murder Rate Boxplot by Region") +
  xlab("Region") +
  ylab("Rate") +
  geom_jitter(width =0.25)
```

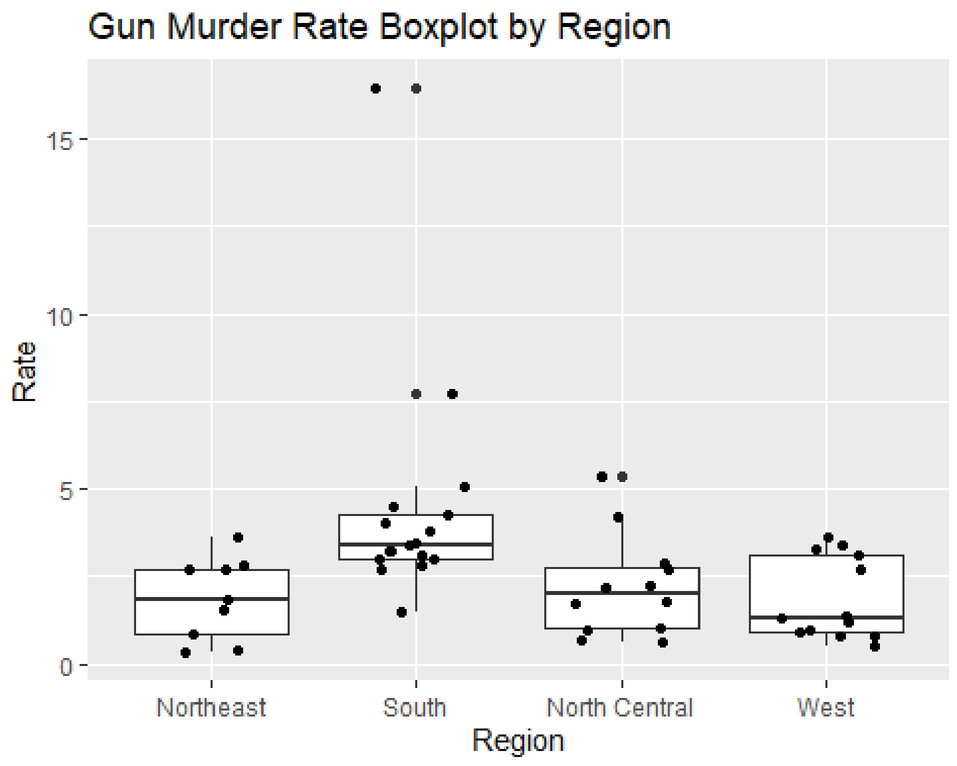


Figure 25: Boxplot of the distributions of murder rates by region from the murders data

The resulting plot should be similar to the one above with the code utilizing the `geom_jitter()` function of ggplot2. The plot should also have a title. Students will be able to demonstrate to instructors the usage ggplot2 to create more informative plots from available data.

Question 6: Construct a density plot for the rate variable by region. Use the `fill` argument of ggplot aesthetics.

```
ggplot(murders2, aes(rate, color = region)) +
  geom_density() +
  ggtitle("Gun Murder Rate Density by Region") +
  xlab("Rate") +
  ylab("Density")
```

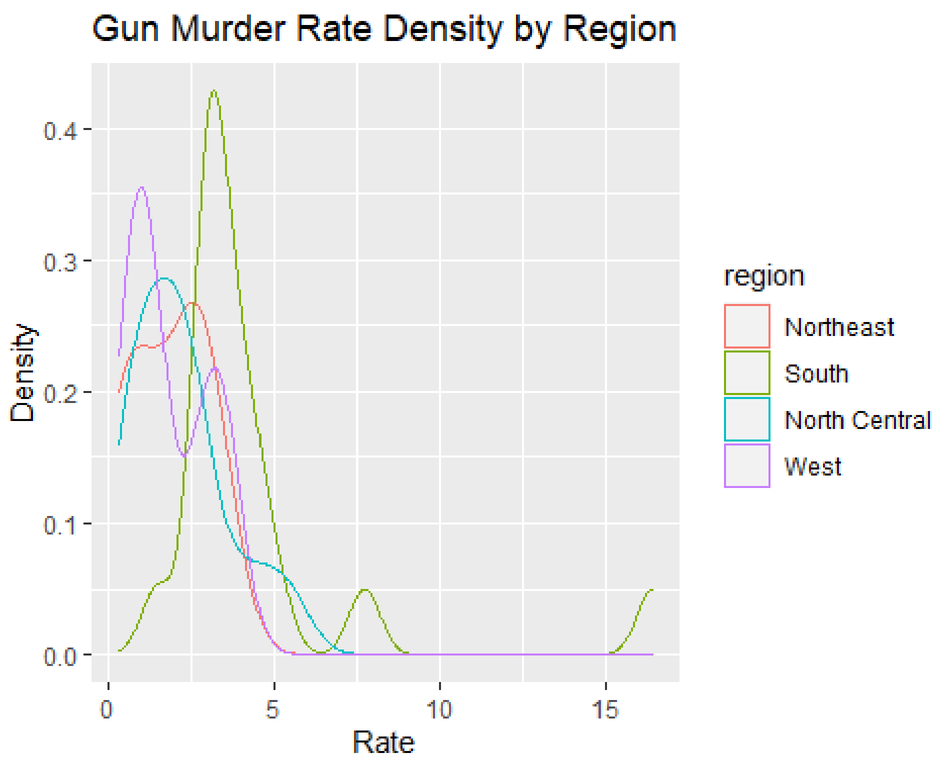


Figure 26: Density plot of murder rates by region

Question 7: What is a weakness of box plots?

The answer here should address that box plots do not show multimodality in data. This question is to measure students' ability to think critically and evaluate the strengths and weaknesses of different plotting methods. This allows instructors to gauge whether students are considering the implications of choosing one plotting method over another.

Chapter 6: Future Work

In this Chapter we will discuss the future work and conclusions.

Future work will include logistic regression and linear models. There are two main statistical modeling tasks: classification and regression. Logistic regression is a classification model that can be easy to interpret. Linear models are a regression model that is easy to interpret. Using logistic regression models and linear models can present students with a good introduction to statistical modeling in the appropriate level of statistical coursework.

Another set of possibilities to look into to advance this work using R for teaching and practicing the Chi-Square Goodness of Fit, McNemar Test, and the Fisher Exact Test. This can reinforce students' ability to learn R, reinforce their knowledge from the coursework, and present opportunities for students to practice appropriate writing for statistical work.

Python presents another possibility in using a programming language for the teaching of statistics. Python has libraries that grant access to statistical methods, although these are not pre-installed as can be seen in R. Additionally, statistical libraries can be more difficult to find in the large collection of Python libraries that are available. If Python can be included in statistical reports as smoothly as R has demonstrated will need to be examined as well.

Conclusion

R is a useful programming language that has become highly valued in the statistics community. It can be straightforward to use one the syntax has been studied sufficiently. There are many existing functions that can be used in the standalone R and more can be added through additional packages that can be downloaded and loaded into R. R can be traced back to the S programming language, and can be straightforward in using in statistics projects. The plotting capabilities of R are not to be underestimated. To aid in statistics projects, R has data structures and packages such as dplyr to make accessing and managing data a smoother experience. One alternative to R is Python, but it should be mentioned that Python is not centered on statistics like R is. The result being that Python has a lot of libraries that are not statistics related which can make the management of Python libraries less focused than R. R on the other hand has packages that appeal to statistics or some application of data science. This does not mean R has a package or function for every possible purpose or project, but R provides the tools for users to be able to modify existing function or packages and create new ones counter such. All of the previously mentioned features can make R a valuable and accessible component in statistics coursework. The knowledge and skills that reinforced by teaching statistics with R can then be carried into other courses in academia or into professional careers. Using R will enable more flexibility for instructors to teach statistics courses and evaluate the progress of the class. Overall, R provides a useful, valuable toolset for teaching statistics.

References

A Modern History of Data Science. University of Wisconsin Data Science Degree. (2021). Retrieved 17 December 2021, from https://datasciencedegree.wisconsin.edu/blog/history-of-data-science/.

Peng, R. (2020). R Programming for Data Science. Bookdown.org. Retrieved 16 December 2021, from https://bookdown.org/rdpeng/rprogdatascience/.

Tarshizi, E. (2021). Tools of the Trade: R vs. Python. University of San Diego. Retrieved 23 December 2021, from https://onlinedegrees.sandiego.edu/data-science-tools-of-the-trade-r-vs-python-2/.

What Is Big Data? | University of Wisconsin. University of Wisconsin Data Science Degree. (2021). Retrieved 30 December 2021, from https://datasciencedegree.wisconsin.edu/data-science/what-is-big-data/.

What is Probability?. Web.ma.utexas.edu. (2012). Retrieved 2 January 2022, from https://web.ma.utexas.edu/users/mks/statmistakes/probability.html.

Appendix

```r
knitr::opts_chunk$set(echo = TRUE)
a <- 1
b <- "Math is fun"
class(a)
class(b)
a
b
3 + 5 - 11
4 * 8 / 6
19^2 / 3
2 ^ 8 + 64 - 144 / 1.5
3 + 5 - 11
4 * 8 / 6
19^2 / 3
2 ^ 8 + 64 - 144 / 1.5
c <- 7 + 8 * 2
d <- 1 / c
e <- 3 ^ d + 33.33
f <- e - 1.089 * 5
g <- 8 %% 2
c
d
e
f
g
a <- seq(1,100)
b <- seq(0,by = .5,length.out =100)
length(a)
length(b)
g <- a + b
g
g[5]
g[30]
g[77]
c <- c(4, 3, 2, 5, 10)
d <- c(12.5, 23.67, 87.56, 90.2134, -50.2)
c - d
d * c
d / c
e <- 4
if(e %% 3 == 0)
{
  print("e is divisible by 3")
} else
{
  print("e is not divisible by 3")
}
```

```r
e <- 24
if(e %% 3 == 0)
{
  print("e is divisible by 3")
} else
{
  print("e is not divisible by 3")
}
f <- c(1, 3, 16, 24, 84, 96, 333, 444, 505)
ifelse(f %% 3 == 0, TRUE, FALSE)
h <- c(9, 11, -12, -456, 78, 90, 300, 150, 152, -56)
mean(h)
i <- c(NA, 4, 5, 100, 10000.3, 4.67, 90.34, NA, 45, 67, -100.34, -500, -7000)
mean(i)
mean(i, na.rm = TRUE)
new.example.fun <- function(x){
  if(is.na(x)){
    return("Can not compute")
  }else{
    x <- x^ 2 + 2 - 36
    return(x)
  }
}
library(dslabs)
library(plyr)
library(dplyr)
library(ggplot2)
data("murders")
hist(murders$population, breaks = 10, xlab = "Population",
     title = "Histogram of Populations in US States")
ggplot(murders, aes(x = population)) +
  geom_histogram(binwidths = 10) +
  xlab("Population") +
  ggtitle("Histogram of Populations")
ggplot(murders, aes(x = population, fill = region)) +
  geom_histogram(binwidths = 10) +
  xlab("Population") +
  ggtitle("Histogram of Populations")
plot(murders$population, murders$total, xlab = "Population",
     ylab = "Total", main="Population and Total")
ggplot(murders, aes(x = population, y=total, color = region)) +
  geom_point() + xlab("Population") +
  ylab("Total") +
  ggtitle("Population and Total")
ggplot(murders, aes(x = population, y=total)) +
  geom_smooth(se = FALSE, method="lm") +
  geom_point(aes(color = region)) + xlab("Population") +
  ylab("Total") +
  ggtitle("Population and Total")
ggplot(murders, aes(x = factor(region), y=population)) +
  geom_boxplot() + xlab("Region") +
  ylab("Population") +
  ggtitle("Region and Population")
ggplot(murders, aes(x = factor(region), y=population)) +
  geom_boxplot(outlier.alpha = 0) + xlab("Region") +
  geom_jitter(aes(alpha = 0.4, color = region)) +
  ylab("Population") +
  ggtitle("Region and Population")
ggplot(murders, aes(x = population, y=total)) +
  geom_point() + xlab("Population") +
  ylab("Total") +
  facet_wrap(~region, nrow = 2, ncol = 2, scales = "fixed") +
  ggtitle("Population and Total")
data("mtcars")
ggplot(mtcars, aes(x = disp, y = mpg)) +
  geom_point() +
  xlab("Displacement (Cubic Inches)") +
  ylab("Miles Per Gallon (MPG)") +
  ggtitle("Displacement and Miles Per Gallon")
ggplot(mtcars, aes(x = disp, y = mpg)) +
  geom_point() +
  geom_smooth(se=FALSE,method = "lm") +
  xlab("Displacement (Cubic Inches)") +
  ylab("Miles Per Gallon (MPG)") +
  ggtitle("Displacement and Miles Per Gallon")
ggplot(mtcars, aes(x = disp, y = mpg)) +
  geom_point() +
  geom_smooth(se=FALSE,method = "loess") +
  xlab("Displacement (Cubic Inches)") +
  ylab("Miles Per Gallon (MPG)") +
  ggtitle("Displacement and Miles Per Gallon")
ggplot(mtcars, aes(x = disp, y = mpg, color=factor(cyl))) +
  geom_point() +
  xlab("Displacement (Cubic Inches)") +
  ylab("Miles Per Gallon (MPG)") +
  ggtitle("Displacement and Miles Per Gallon") +
  labs(color = "Cylinders")
ggplot(mtcars, aes(x = disp, y = mpg, color=factor(vs))) +
  geom_point() +
  xlab("Displacement (Cubic Inches)") +
  ylab("Miles Per Gallon (MPG)") +
  ggtitle("Displacement and Miles Per Gallon") +
  labs(color = "Engine Shape")
ggplot(mtcars, aes(x = disp, y = mpg)) +
  geom_point() +
  xlab("Displacement (Cubic Inches)") +
  ylab("Miles Per Gallon (MPG)") +
  ggtitle("Displacement and Miles Per Gallon") +
  facet_wrap(~factor(vs), scales = "fixed")
ggplot(mtcars, aes(x = factor(cyl), y = disp)) +
  geom_boxplot() +
  xlab("Number of Cylinders") +
  ylab("Displacement (Cubic Inches)") +
  ggtitle("Displacement and Number of Cylinders")
library(magrittr)


engine.cyl <- mtcars %>%
  mutate(vs = ifelse(vs ==0, "V-shaped", "straight"),
         cyl = factor(cyl)) %>%
  group_by(vs, cyl) %>%
  count()

ggplot(engine.cyl, aes(x=vs, y = n, fill = cyl)) +
  geom_bar(position="stack", stat="identity") +
  labs(fill = "Cylinders") +
  xlab("Engine Shape") +
  ylab("Count") +
  ggtitle("Engine Shape and Number of Cylinders")
ggplot(engine.cyl, aes(x=vs, y = n, fill = cyl)) +
  geom_bar(position="dodge", stat="identity") +
  labs(fill = "Cylinders") +
  xlab("Engine Shape") +
  ylab("Count") +
  ggtitle("Engine Shape and Number of Cylinders")
ggplot(mtcars, aes(x = factor(cyl), y = disp)) +
  geom_boxplot() +
  xlab("Number of Cylinders") +
  ylab("Displacement (Cubic Inches)") +
  ggtitle("Displacement and Number of Cylinders")
mtcars %>%
  mutate(cyl = factor(cyl)) %>%
  group_by(cyl) %>%
  summarize(minimum = min(disp), Q1 = quantile(disp,probs=.25),
            median = median(disp), Q3 = quantile(disp,probs=.75),
            maximum =max(disp))
mtcars %>%
  mutate(cyl = factor(cyl)) %>%
  group_by(cyl) %>%
  summarize(minimum = min(disp), Q1 = quantile(disp,probs=.25),
            median = median(disp), Q3 = quantile(disp,probs=.75),
            maximum =max(disp), avg = mean(disp), stdev = sd(disp))
gears.carbs <- mtcars %>%
  mutate(gear = factor(gear),
         carb = factor(carb)) %>%
  group_by(gear, carb) %>%
  count()

ggplot(gears.carbs, aes(x=gear, y = n, fill = carb)) +
  geom_bar(position="stack", stat="identity") +
  labs(fill = "Number of Carburetors") +
  xlab("Number of Forward Gears") +
  ylab("Count") +
  ggtitle("Number of Forward Gears and Number of Carburetors")
```