

# Stat 120

Deepak Bastola

2023-04-02



# Contents

<b>Introduction to Statistics</b>	<b>5</b>
0.1 Learning Objectives . . . . .	5
<b>Basics R</b>	<b>9</b>
<b>1 What is R?</b>	<b>9</b>
1.1 What is RStudio? . . . . .	9
1.2 R Studio Server . . . . .	9
1.3 R/RStudio . . . . .	10
1.4 Installing R/RStudio (not needed if you are using the maize server) . . . . .	10
1.5 Install LaTeX (for knitting R Markdown documents to PDF): . .	11
1.6 Updating R/RStudio (not needed if you are using the maize2 server) . . . . .	11
1.7 Opening a new file . . . . .	12
1.8 Running codes and knitting .Rmd files: . . . . .	12
1.9 Few More Instructions . . . . .	12
1.10 VPN . . . . .	13
<b>2 R Markdown Basics</b>	<b>15</b>
2.1 R Markdown Syntax . . . . .	15
<b>3 Helpful R codes</b>	<b>21</b>
3.1 Residual Plots in ggplot2 . . . . .	21
3.2 Plotly codes . . . . .	23

<b>4 Homework Guidelines</b>	<b>25</b>
4.1 Format . . . . .	25
4.2 Content . . . . .	26
4.3 Problems using R . . . . .	26
<b>5 Graph Formatting</b>	<b>27</b>
5.1 Load the required packages and datasets . . . . .	27
5.2 Graph theme and colors . . . . .	27
5.3 Graph Sizing in R . . . . .	33
<b>6 Table Formatting</b>	<b>43</b>
<b>7 Report Guidelines</b>	<b>49</b>
<b>Class Activity</b>	<b>55</b>
<b>8 Class Activity 1</b>	<b>55</b>
8.1 Your Turn 1 . . . . .	55
8.2 Your Turn 2 . . . . .	56
<b>9 Class Activity 2</b>	<b>59</b>
9.1 Your Turn 1 . . . . .	59
9.2 Your Turn 2 . . . . .	59
<b>10 Class Activity 3</b>	<b>63</b>
10.1 Case Study 1 . . . . .	63
10.2 Case Study 2 . . . . .	64
<b>11 Class Activity 4</b>	<b>67</b>
11.1 Your Turn 1 . . . . .	67
11.2 Your Turn 2 . . . . .	74

# Introduction to Statistics

Welcome to the captivating world of statistics! This course will provide you with a solid foundation in statistical theory while taking you on a journey through the practical aspects of the subject. Along the way, you'll gain experience with statistical software, learn to interpret and effectively communicate statistical findings, and explore a range of topics in data analysis, statistical inference, and randomness. Some of the areas we'll delve into include linear regression, experimental design, normal distribution, sampling distributions, confidence intervals, and the bootstrap method.

Although statistics is a field that relies on mathematics, it stands apart as a distinct discipline. At the heart of this course is the ability to interpret results and grasp underlying concepts, rather than merely obtaining numerical outcomes. By engaging with a diverse set of problems, you'll become well-versed in statistical methodologies. However, it's crucial to remember that a deep understanding of the concepts is the key to drawing meaningful conclusions.

## 0.1 Learning Objectives

- Learn basic principles of data analysis, and how data is produced and used in studies and experiments.
- Understand role of variation and randomness. Understand principles of inference: confidence intervals and hypothesis tests.
- Develop ability to examine statistical arguments critically.
- Learn how to use software (R/RStudio) to analyze data, create graphs, perform basic statistical tests



# Basics R



# **Chapter 1**

## **What is R?**

R is a free and open source statistical programming language that facilitates statistical computation. There are a myriad of application that can be done in R, thanks to a huge online support community and dedicated packages. However, R has no graphical user interface and it has to be run by typing commands into a text interface.

### **1.1 What is RStudio?**

RStudio provides graphical interface to R! You can think of RStudio as a graphical front-end to R that provides extra functionality. The use of the R programming language with the RStudio interface is an essential component of this course.

### **1.2 R Studio Server**

The quickest way to get started is to go to <https://maize.mathcs.carleton.edu>, which opens an R Studio window in your web browser. Once logged in, I recommend that you do the following:

- Step 1: Create a folder for this course where you can save all of your work. In the Files window, click on New Folder.
- Step 2: Click on Tools -> Global Options -> R Markdown. Then uncheck the box that says “Show output inline...”

(It is also possible to download RStudio on your own laptop. Instructions may be found at the end of this document.)

## 1.3 R/RStudio

The use of the R programming language with the RStudio interface is an essential component of this course. You have two options for using RStudio:

- The **server version** of RStudio on the web at (<https://maize.mathcs.carleton.edu>). The advantage of using the server version is that all of your work will be stored in the cloud, where it is automatically saved and backed up. This means that you can access your work from any computer on campus using a web browser. This server may run slow during peak days/hours. I also recommend you to download a local version of R server in your computer in case of rare outages.
- A **local version** of RStudio installed on your machine. This option is highly recommended due to the computational resources this course demands. Using this version you can only store your files in your local machine. Additionally, we can save our work on GitHub. We will learn how to use GitHub in the beginning of the course. Both R and RStudio are free and open-source. Please make sure that you have recently updated both R and RStudio.

## 1.4 Installing R/RStudio (not needed if you are using the maize server)

Download the latest version of R: <https://cran.r-project.org/>

Download the free Rstudio desktop version: <https://www.rstudio.com/products/rstudio/download/>

Use the default download and install options for each. For R, download the “precompiled binary” distribution rather than the source code

### Updating R/RStudio (not needed if you are using the maize server)

If you have used a local version of R/RStudio before and it is still installed on your machine, then you should make sure that you have the most recent versions of each program.

- To check your version of R, run the command `getRversion()` and compare your version to the newest version posted on <https://cran.r-project.org/>. If you need an update, then install the newer version using the installation directions above.
- In RStudio, check for updates with the menu option `Help > Check for updates`. Follow directions if an update is needed.

#### Did it work? (A sanity check after your install/update)

Do whatever is appropriate for your operating system to launch RStudio. You should get a window similar to the screenshot you see here, but yours will be more boring because you haven't written any code or made any figures yet!

Put your cursor in the pane labeled *Console*, which is where you interact with the live R process. Create a simple object with code like `x <- 2 * 4` (followed by enter or return). Then inspect the `x` object by typing `x` followed by enter or return. You should see the value `8` printed. If this happened, you've succeeded in installing R and RStudio!

## 1.5 Install LaTeX (for knitting R Markdown documents to PDF):

You need a Latex compiler to create a pdf document from a R Markdown file. If you use the maize server, you don't need to install anything. If you are using a local RStudio, you should install a Latex compiler. Below are the recommended installers for Windows and Mac:

- MacTeX for Mac (3.2GB)
- MiKTeX for Windows (190MB)
- Alternatively, you can install the `tinytex` R package by running `install.packages("tinytex")` in the console.

## 1.6 Updating R/RStudio (not needed if you are using the maize2 server)

If you have used a local version of R/RStudio before and it is still installed on your machine, then you should make sure that you have the most recent versions of each program.

- To check your version of R, run the command `getRversion()` and compare your version to the newest version posted on <https://cran.r-project.org/>. If you need an update, then install the newer version using the installation directions above.
- In RStudio, check for updates with the menu option `Help > Check for updates`. Follow directions if an update is needed.

## 1.7 Opening a new file

If using Rstudio on your computer, using the **File>Open File** menu to find and open this .Rmd file.

If using Maize Rstudio from your browser:

- In the Files tab, select **Upload** and **Choose File** to find the .Rmd that you downloaded. Click *OK* to upload to your course folder/location in the maize server account.
- Click on the .Rmd file in the appropriate folder to open the file.

## 1.8 Running codes and knitting .Rmd files:

- You can run a line of code by placing your cursor in the line of code and clicking **Run Selected Line(s)**
- You can run an entire chunk by clicking the green triangle on the right side of the code chunk.
- After each small edit or code addition, **Knit** your Markdown. If you wait until the end to Knit, it will be harder to find errors in your work.
- Format output type: You can use any of pdf\_document, html\_document type, or word\_document type.
- **Maize users:** You may also need to allow for “pop-up” in your web browser when knitting documents.

## 1.9 Few More Instructions

The default setting in Rstudio when you are running chunks is that the “output” (numbers, graphs) are shown **inline** within the Markdown Rmd. If you prefer to have your plots appear on the right of the console and not below the chunk, then change the settings as follows:

1. Select Tools > Global Options.
2. Click the R Markdown section and uncheck (if needed) the option Show output inline for all R Markdown documents.
3. Click OK.

Now try running R chunks in the .Rmd file to see the difference. You can recheck this box if you prefer the default setting.

## 1.10 VPN

If you plan to do any work off campus this term, you need to install Carleton's VPN. This will allow you to access the **maize** server (if needed).

### Installing the GlobalProtect VPN

Follow the directions here to install VPN.



# Chapter 2

## R Markdown Basics

An R Markdown file (.Rmd file) combines R commands and written analyses, which are ‘knit’ together into an HTML, PDF, or Microsoft Word document.

An R Markdown file contains three essential elements:

- Header: The header (top) of the file contains information like the document title, author, date and your preferred output format (pdf\_document, word\_document, or html\_document).
- Written analysis: You write up your analysis after the header and embed R code where needed. The online help below shows ways to add formatting details like bold words, lists, section labels, etc to your final pdf/word/html document. For example, adding **\*\*** before and after a word will bold that word in your compiled document.
- R chunks: R chunks contain the R commands that you want evaluated. You embed these chunks within your written analysis and they are evaluated when you compile the document.

Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.

### 2.1 R Markdown Syntax

#### 2.1.1 Lists in R Markdown:

You can use asterisk mark to provide emphasis, such as **\*italics\*** or **\*\*bold\*\***. You can create lists with a dash:

```
- Item 1
- Item 2
- Item 3
  + Subitem 1
* Item 4
```

to produce

- Item 1
- Item 2
- Item 3
  - Subitem 1
- Item 4

You can embed Latex equations in-line,  $\frac{1}{n} \sum_{i=1}^n x_i$  or in a new line as  $\text{Var}(X) = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2$  to produce

$$\text{Var}(X) = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2$$

### 2.1.2 Embed an R code chunk:

Use the following

```
```r
Use back ticks to
create a block of code
```
```

to produce:

```
Use back ticks to
create a block of code
```

You can also evaluate and display the results of R code. Each tasks can be accomplished in a suitably labeled chunk like the following:

```
summary(cars)
```

```

speed          dist
Min.   : 4.0   Min.   : 2.00
1st Qu.:12.0  1st Qu.: 26.00
Median :15.0  Median : 36.00
Mean   :15.4  Mean   : 42.98
3rd Qu.:19.0  3rd Qu.: 56.00
Max.   :25.0  Max.   :120.00

fit <- lm(dist ~ speed, data = cars)
fit

```

```

Call:
lm(formula = dist ~ speed, data = cars)

Coefficients:
(Intercept)      speed
-17.579        3.932

```

### 2.1.3 Including Plots:

You can also embed plots. See Figure 2.1 for example:

```

par(mar = c(0, 1, 0, 1))
pie(
  c(280, 60, 20),
  c('Sky', 'Sunny side of pyramid', 'Shady side of pyramid'),
  col = c('#0292D8', '#F7EA39', '#C4B632'),
  init.angle = -50, border = NA
)

```

(Credit: Yihui Xie)

### 2.1.4 Read in data files:

```

simple_data <- read.csv("https://deepbas.io/data/simple-1.dat", )
summary(simple_data)

```

|                  | initials         | state        | age |
|------------------|------------------|--------------|-----|
| Length:3         | Length:3         | Min.   :45.0 |     |
| Class :character | Class :character | 1st Qu.:47.5 |     |

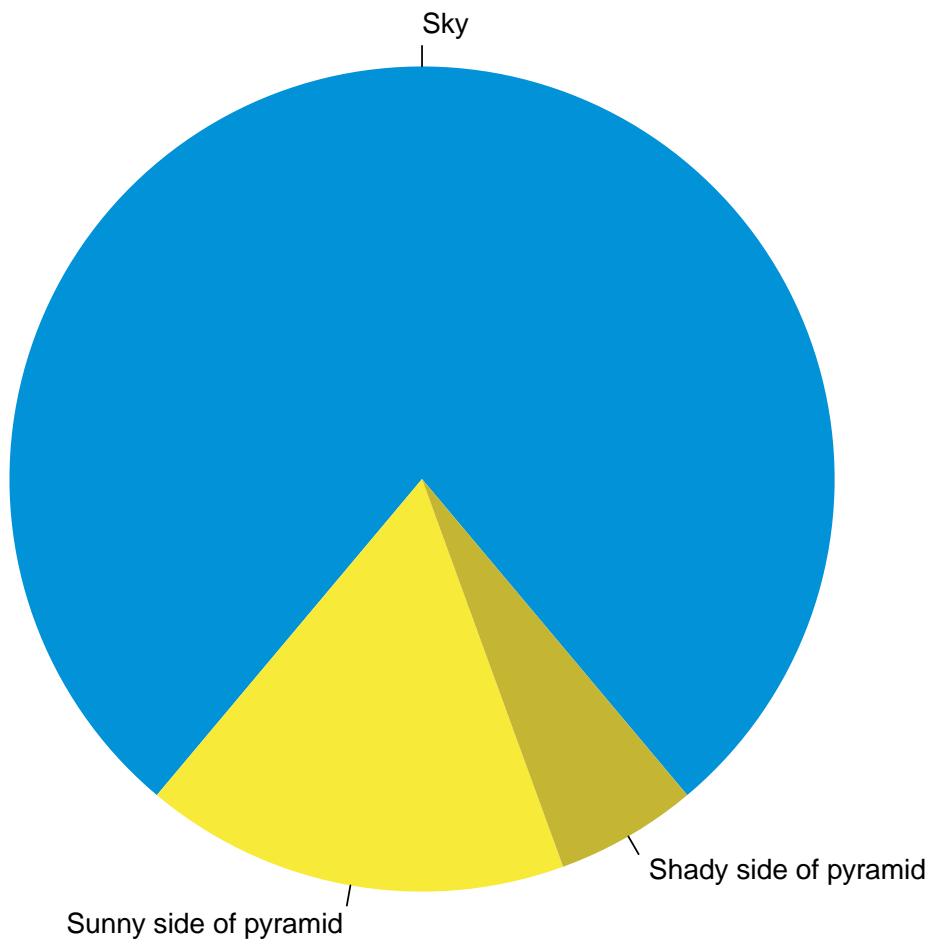


Figure 2.1: A fancy pie chart.

```
Mode :character Mode :character Median :50.0
       Mean :52.0
       3rd Qu.:55.5
       Max. :61.0
time
Length:3
Class :character
Mode :character
```

```
knitr::kable(simple_data)
```

| initials | state | age | time |
|----------|-------|-----|------|
| vib      | MA    | 61  | 6:01 |
| adc      | TX    | 45  | 5:45 |
| kme      | CT    | 50  | 4:19 |

### 2.1.5 Hide the code:

If we enter the `echo = FALSE` option in the R chunk (see the .Rmd file). This prevents the R code from being printed to your document; you just see the results.

| initials | state | age | time |
|----------|-------|-----|------|
| vib      | MA    | 61  | 6:01 |
| adc      | TX    | 45  | 5:45 |
| kme      | CT    | 50  | 4:19 |



# Chapter 3

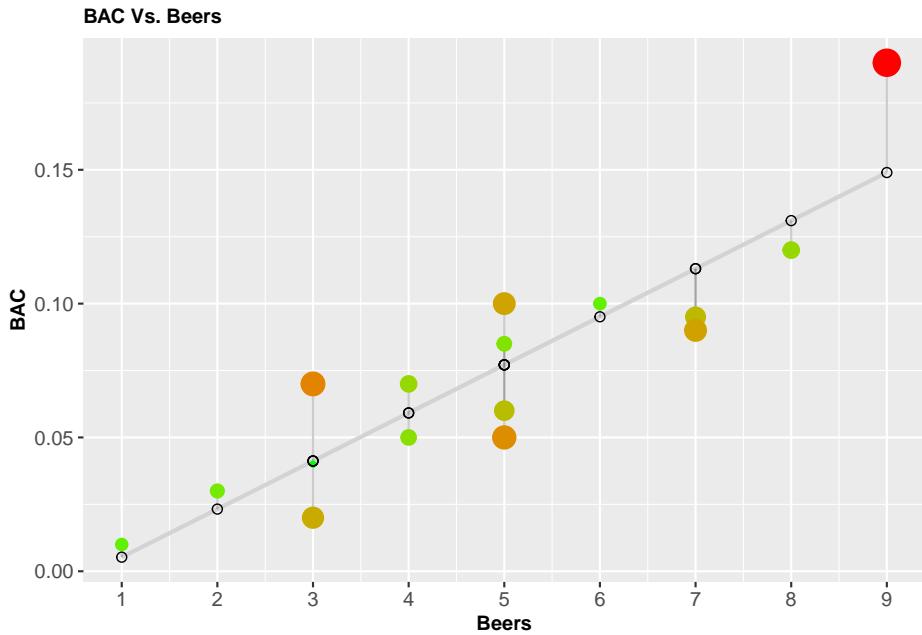
## Helpful R codes

### 3.1 Residual Plots in ggplot2

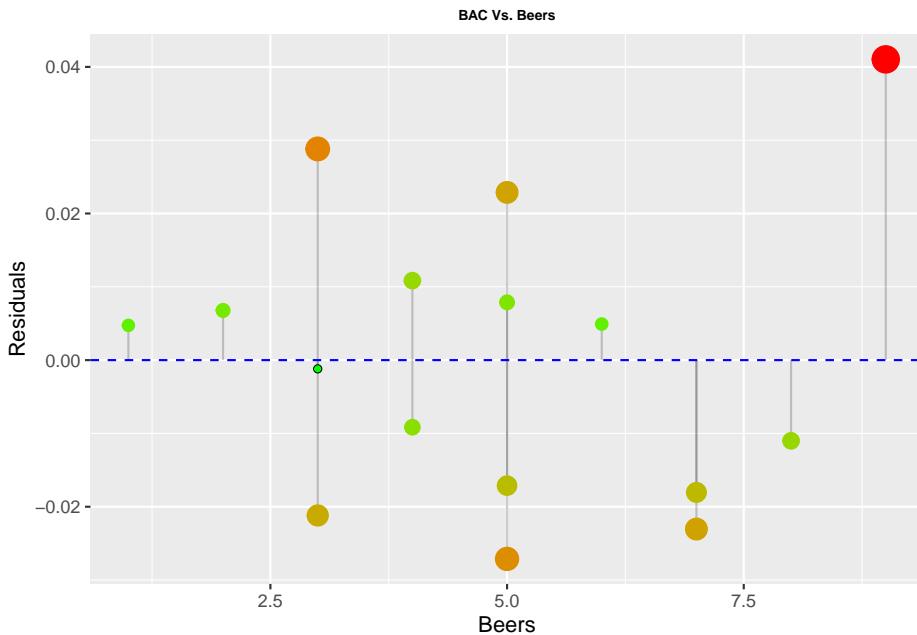
```
# residual size plot
library(ggplot2)
bac <- read.csv("https://raw.githubusercontent.com/deepbas/statdatasets/main/BAC.csv")

fit <- lm(BAC ~ Beers, data = bac) # fit the model
bac$predicted <- predict(fit)      # Save the predicted values
bac$residuals <- residuals(fit)    # Save the residual values

ggplot(bac, aes(x = Beers, y = BAC)) +
  geom_smooth(method = "lm", se = FALSE, color = "lightgrey") +      # regression line
  geom_segment(aes(xend = Beers, yend = predicted), alpha = .2) +      # draw line from point to
  geom_point(aes(color = abs(residuals), size = abs(residuals))) +    # size of the points
  scale_color_continuous(low = "green", high = "red") +
  labs(title = "BAC Vs. Beers") + # color of the points mapped to residual size - green smaller, r
  guides(color = FALSE, size = FALSE) +                                     # Size legend removed
  geom_point(aes(y = predicted), shape = 1, size = 2) +
  scale_x_continuous(breaks=1:9) +
  theme(axis.text=element_text(size=10),
        axis.title=element_text(size=10,face="bold"),
        plot.title = element_text(size = 10, face = "bold"))
```



```
ggplot(bac, aes(x = Beers, y = residuals)) +
  geom_point() +
  theme(legend.position = "none") +
  geom_segment(aes(xend = Beers, yend = 0), alpha = .2) +
  scale_color_continuous(low = "green", high = "red") +
  geom_point(aes(color = abs(residuals), size = abs(residuals))) + # size of the points
  geom_hline(yintercept = 0, col = "blue", size = 0.5, linetype = "dashed") +
  labs(title = "BAC Vs. Beers",
       x = "Beers",
       y = "Residuals") +
  theme(plot.title = element_text(hjust=0.5, size=7, face='bold'))
```



## 3.2 Plotly codes

```
library(plotly)

cell_phone_data <- data.frame(
  Type = c("Android", "iPhone", "Blackberry", "Non Smartphone", "No Cell Phone"),
  Frequency = c(458, 437, 141, 924, 293)
)

data <- data.frame(
  Gender = c("Female", "Male"),
  In_a_relationship = c(32, 10),
  Its_complicated = c(12, 7),
  Single = c(63, 45)
)

plot_ly(cell_phone_data, labels = ~Type, values = ~Frequency, type = 'pie',
        textposition = 'inside', hoverinfo = 'label+value+percent',
        textinfo = 'label', insidetextfont = list(color = '#FFFFFF')) %>%
layout(title = 'Cell Phone Usage',
       xaxis = list(showgrid = FALSE, zeroline = FALSE, showticklabels = FALSE),
       yaxis = list(showgrid = FALSE, zeroline = FALSE, showticklabels = FALSE))
```

```
plot_ly(data, x = ~Gender, y = ~In_a_relationship, type = 'bar', name = 'In a relationship')
  add_trace(y = ~Its_complicated, name = 'It\'s complicated') %>%
  add_trace(y = ~Single, name = 'Single') %>%
  layout(yaxis = list(title = 'Number of People'), barmode = 'group')
```

## Chapter 4

# Homework Guidelines

- You **can** discuss homework problems with classmates, but you must write up **your own** homework solutions and **do your own work in R (no sharing commands or output) unless explicitly told otherwise.**
- **Getting help:** You **can** use the following resources to complete your homework:
  - Carleton faculty (myself, other stat faculty, etc)
  - Discussions with classmates (see above) or knowledgeable friends
  - The math skills center
  - Lab assistants (in CMC 304)
  - Prefects
  - Student solutions provided in the back of your student textbook or in the student solution manual
  - It is okay to get coding help from prefects, tutors, classmates, online resources, but extra care should be done to write your own versions of the codes.
- You **cannot** use any resources other than the ones listed above to complete assignments (homework, reports, etc) for this class. E.g. you cannot use a friend's old assignments or reports, answers found on the internet, textbook (instructor) solutions manual, etc.

### 4.1 Format

- At the top of each assignment, provide the following details:
  - Class name (e.g., Stat xxx)
  - Homework number (e.g., "homework 1")
  - Your name

- The names of classmates that you worked with on all or part of the assignment
- Turn in a neat, **correctly ordered**, and **legible** assignment with no ragged edges. If it can't be read, it will not be graded.
- **Staple** - no folded corners accepted!

## 4.2 Content

- You must **show all work** and formulas used to answer any question which requires a numerical answer. Be sure to show the natural sequence of work needed to answer the problem.
- Use **complete sentences** when answering any problem that requires an explanation or overall problem summary.

## 4.3 Problems using R

- These problems must be written up as Word or pdf document using R Markdown.
- **Label** all output with the problem number.
- First **give your answer to a problem in written form**, never just give R output as your answer. Follow your written answer with your “work” which contains **all relevant R commands and output** (numeric output or graphs) that are needed to answer a homework problem. Do not include typos or unnecessary commands/output.

# Chapter 5

## Graph Formatting

This worksheet provides a comprehensive guide on graph formatting in R using the ggplot2 package. We will explore various aspects of formatting, such as adding figure numbers, captions, titles, axes labels, customizing themes, and using different color scales.

### 5.1 Load the required packages and datasets

```
Cereals <- read.csv("http://people.carleton.edu/~kstclair/data/Cereals.csv")
```

### 5.2 Graph theme and colors

#### 5.2.1 Adding figure numbers and captions

To automatically add figure numbers and captions, include the option `fig_caption: true` in the output options at the top of your markdown file. To add captions to the figures, use the `fig.cap` argument in the R code chunk that creates the figure.

```
ggplot(Cereals, aes(x = calgram)) +  
  geom_histogram(binwidth = 0.3, fill = "skyblue", color = "black") +  
  labs(title = "Histogram of Calorie Content in Cereals",  
       x = "Calorie Content (g)",  
       y = "Count") +  
  theme_minimal()
```

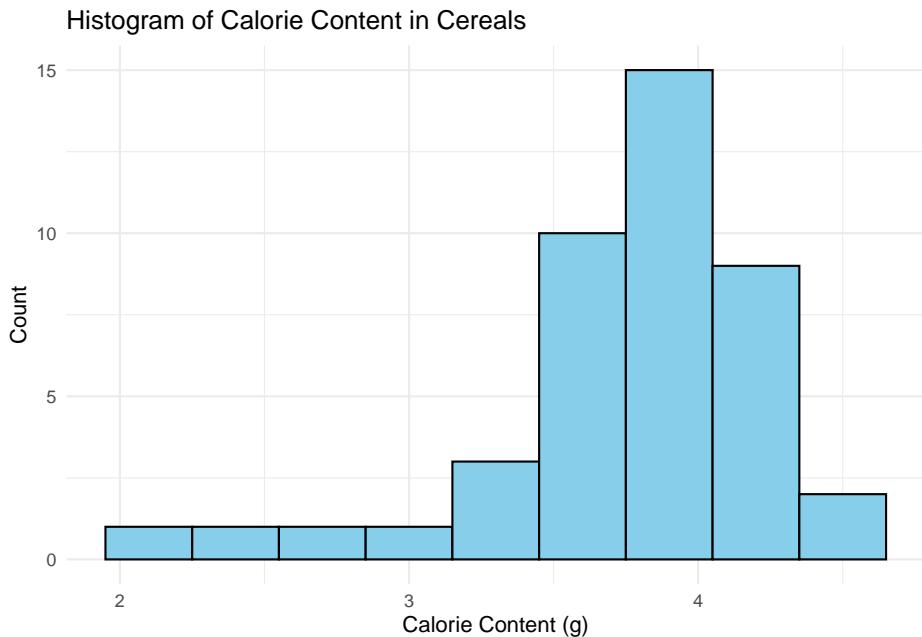
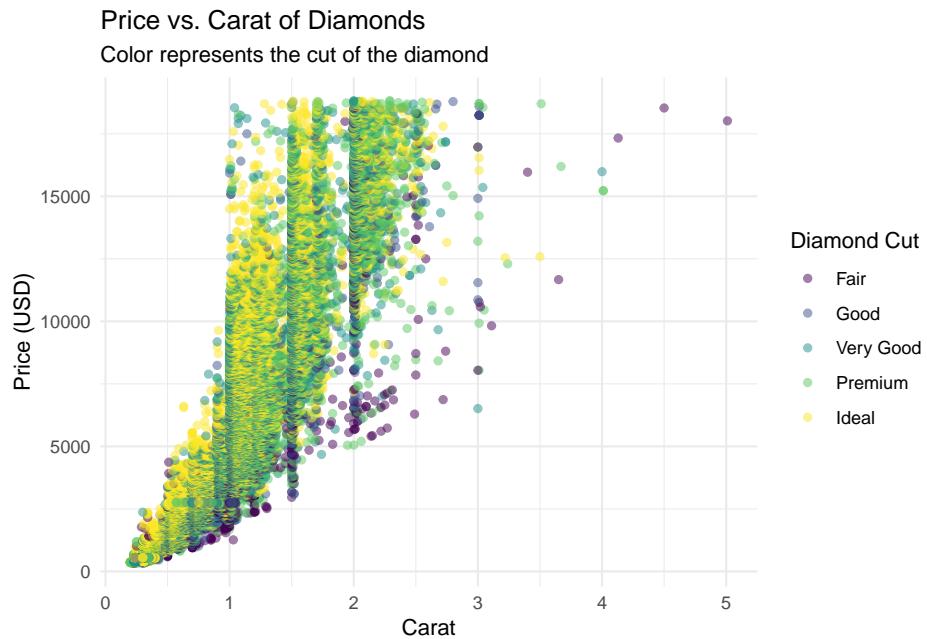


Figure 5.1: A nice figure

### 5.2.2 Customizing titles, axis labels, and legends

You can customize titles, axis labels, and legends using the `labs()` function.

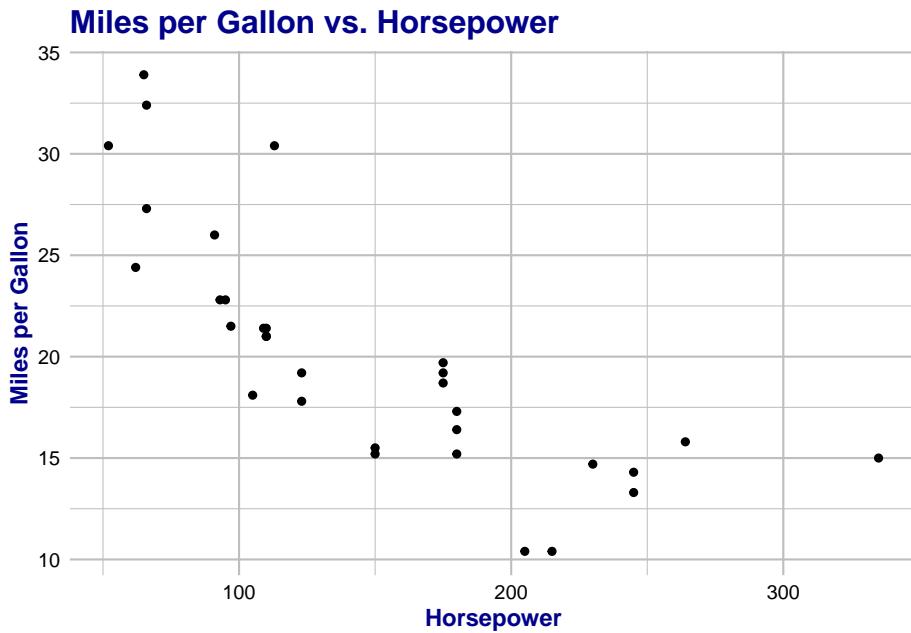
```
ggplot(data = diamonds, aes(x = carat, y = price, color = cut)) +
  geom_point(alpha = 0.5) +
  labs(title = "Price vs. Carat of Diamonds",
       subtitle = "Color represents the cut of the diamond",
       x = "Carat",
       y = "Price (USD)",
       color = "Diamond Cut") +
  theme_minimal()
```



### 5.2.3 Customizing themes

You can customize themes using the `theme()` function and various `element_*`() functions.

```
ggplot(mtcars, aes(x = hp, y = mpg)) +
  geom_point() +
  labs(title = "Miles per Gallon vs. Horsepower",
       x = "Horsepower",
       y = "Miles per Gallon") +
  theme_minimal() +
  theme(plot.title = element_text(size = 16, face = "bold", color = "darkblue"),
        axis.title = element_text(size = 12, face = "bold", color = "darkblue"),
        axis.text = element_text(size = 10, color = "black"),
        panel.grid.major = element_line(color = "gray", size = 0.5),
        panel.grid.minor = element_line(color = "gray", size = 0.25))
```



#### 5.2.4 Using different color scales

You can use different color scales for both continuous and discrete variables using the `scale_color_*`() and `scale_fill_*`() functions.

```
ggplot(data = diamonds, aes(x = carat, y = price, color = cut)) +
  geom_point(alpha = 0.5) +
  labs(title = "Price vs. Carat of Diamonds",
       subtitle = "Color represents the cut of the diamond",
       x = "Carat",
       y = "Price (USD)",
       color = "Diamond Cut") +
  scale_color_brewer(palette = "Set1") +
  theme_minimal()
```

#### 5.2.5 Customizing plot elements

You can customize plot elements such as points, lines, and bars using the corresponding `geom_*`() functions and their arguments.

```
ggplot(mtcars, aes(x = hp, y = mpg, shape = factor(gear), size = gear)) +
  geom_point(aes(color = factor(gear))) +
```

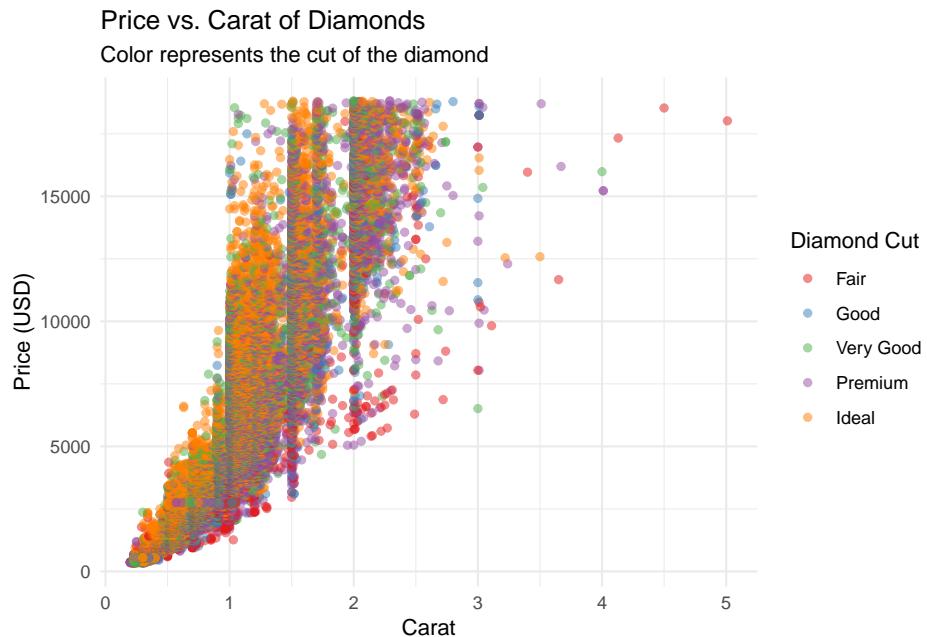
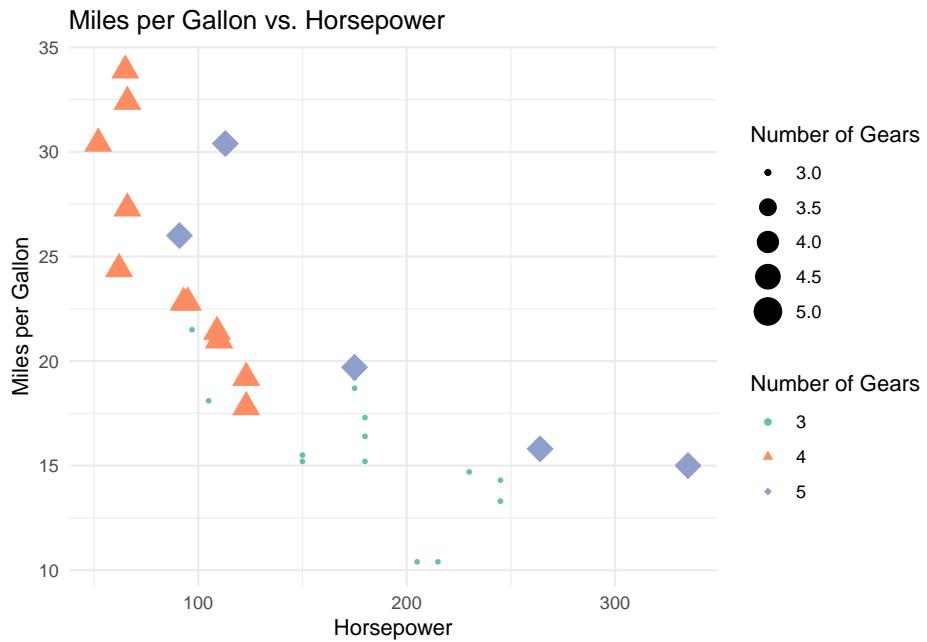


Figure 5.2: Figure 4: Scatterplot of price vs. carat of diamonds with color representing the cut and custom color scale.

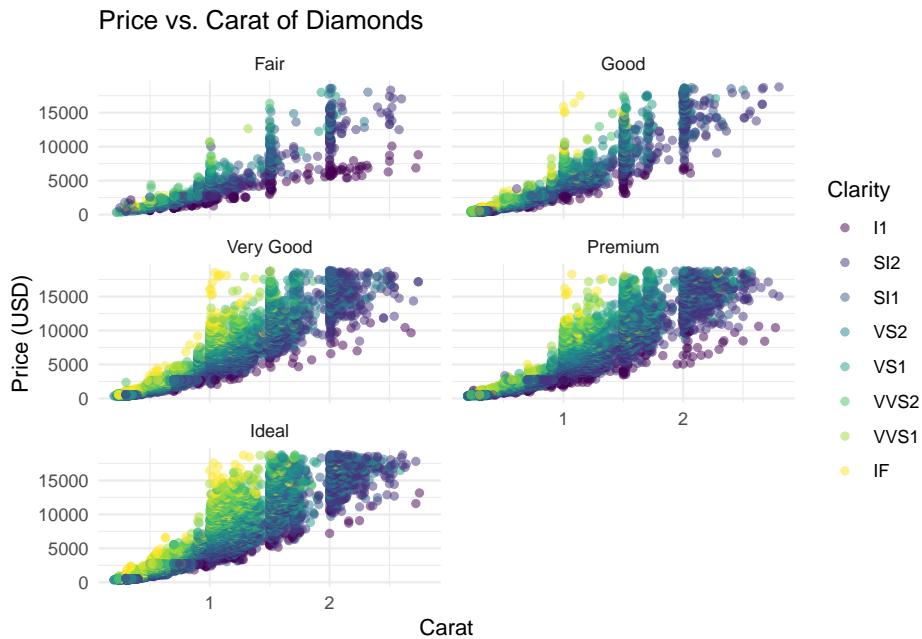
```
labs(title = "Miles per Gallon vs. Horsepower",
     x = "Horsepower",
     y = "Miles per Gallon",
     color = "Number of Gears",
     shape = "Number of Gears",
     size = "Number of Gears") +
theme_minimal() +
scale_shape_manual(values = c(16, 17, 18)) +
scale_color_brewer(palette = "Set2")
```



### 5.2.6 Faceting

You can create multiple plots based on a categorical variable using the `facet_wrap()` and `facet_grid()` functions.

```
ggplot(data = diamonds %>% filter(carat < 3), aes(x = carat, y = price)) +
  geom_point(aes(color = clarity), alpha = 0.5) +
  labs(title = "Price vs. Carat of Diamonds",
       x = "Carat",
       y = "Price (USD)",
       color = "Clarity") +
  facet_wrap(~ cut, ncol = 2) +
  theme_minimal()
```



## 5.3 Graph Sizing in R

This worksheet demonstrates how to adjust the size of various plots in R using ggplot2. We will explore different techniques to control the size of the plots and their elements.

### 5.3.1 Load the necessary libraries and data

```
library(ggplot2)
library(dplyr)

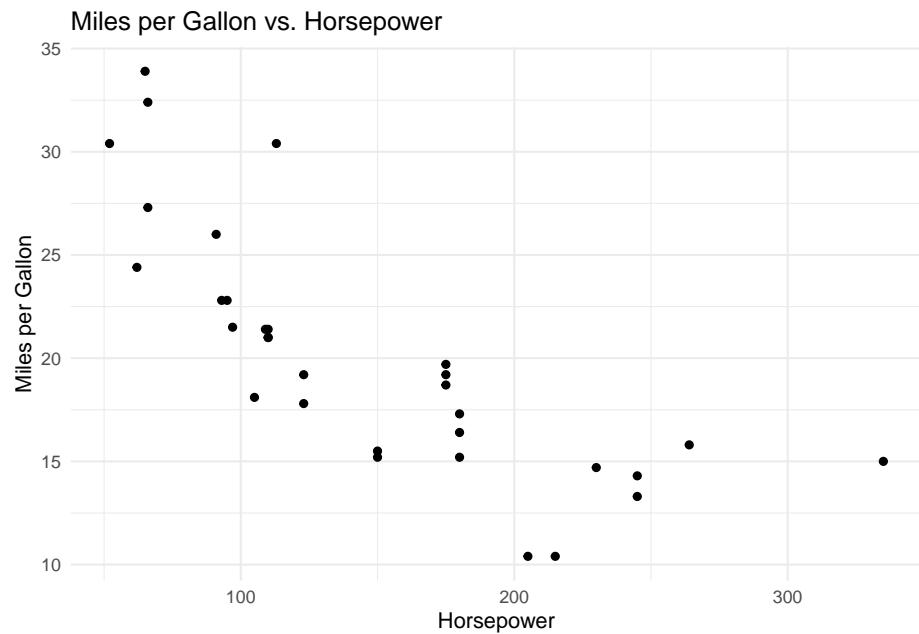
# Load the built-in datasets
data("mtcars")
data("diamonds")
data("iris")
```

### 5.3.2 Adjusting the overall size of the plot

You can control the overall size of the plot using the width and height options within the R Markdown output settings. Another way is to use the ggsave() function when saving the plot as an image file.

```
scatter_plot <- ggplot(mtcars, aes(x = hp, y = mpg)) +
  geom_point() +
  labs(title = "Miles per Gallon vs. Horsepower",
       x = "Horsepower",
       y = "Miles per Gallon") +
  theme_minimal()

scatter_plot
```

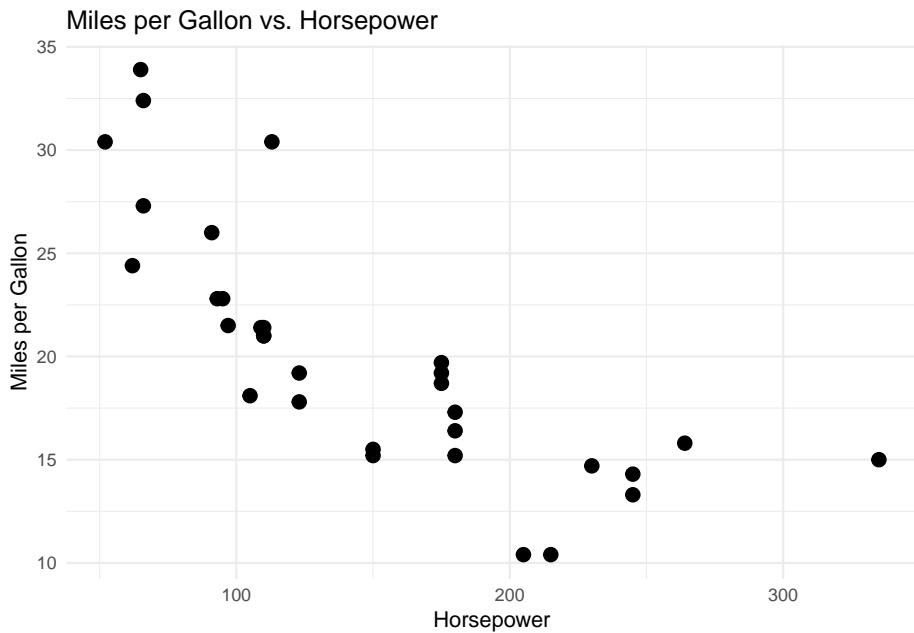


### 5.3.3 Adjusting the size of points, lines, and bars

Use the size parameter within the `geom_*`() functions to control the size of points, lines, and bars.

```
scatter_plot_large_points <- scatter_plot +
  geom_point(size = 3)

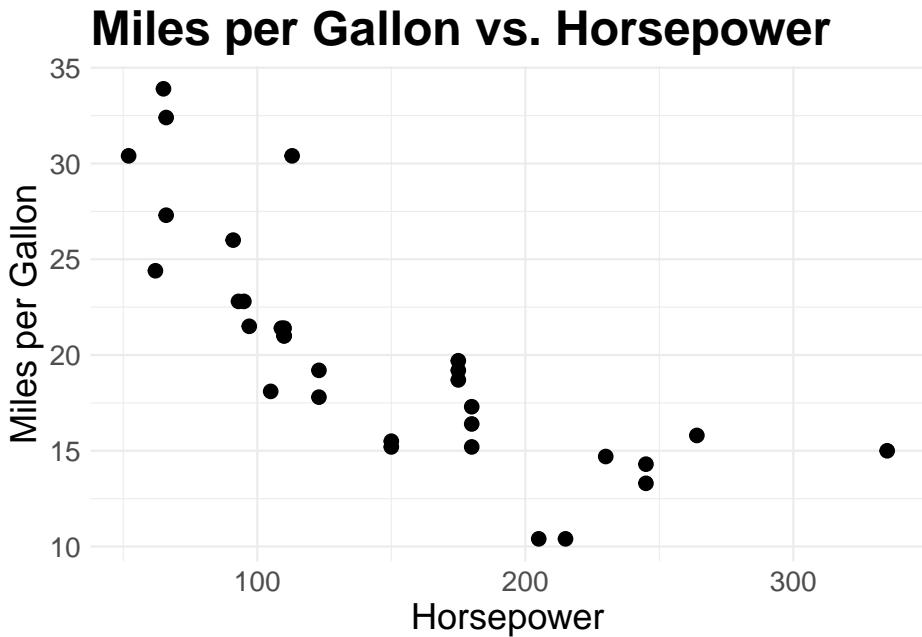
scatter_plot_large_points
```



### 5.3.4 Adjusting the size of text elements

You can change the size of text elements, such as axis labels and titles, using the `theme()` function.

```
scatter_plot_custom_text <- scatter_plot_large_points +  
  theme(plot.title = element_text(size = 24, face = "bold"),  
        axis.title.x = element_text(size = 18),  
        axis.title.y = element_text(size = 18),  
        axis.text.x = element_text(size = 14),  
        axis.text.y = element_text(size = 14))  
  
scatter_plot_custom_text
```

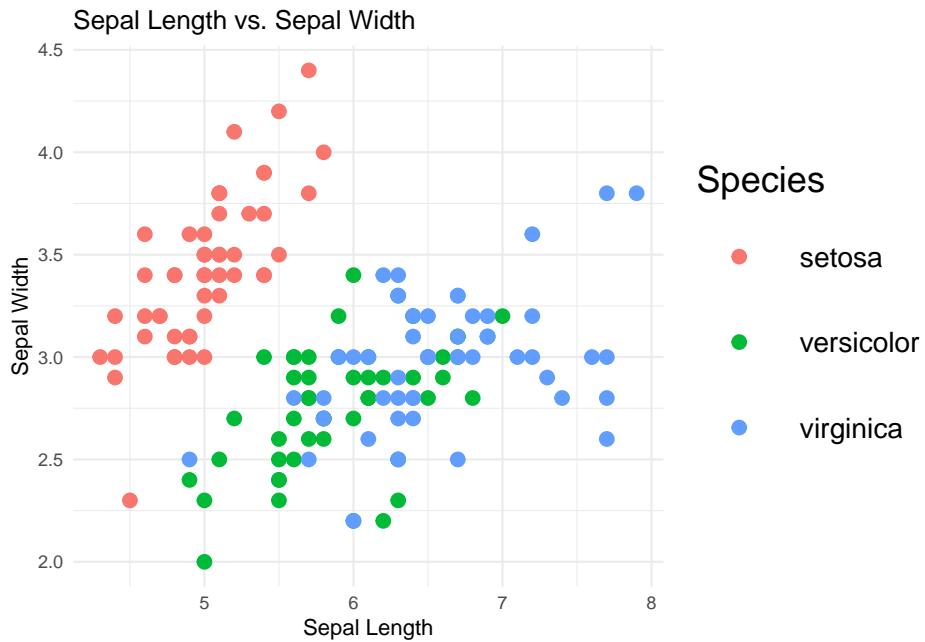


### 5.3.5 Adjusting the size of legend elements

You can modify the size of the legend elements using the `theme()` function along with `element_text()` and `element_rect()`.

```
iris_scatter_plot <- ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width, color = Species))
  geom_point(size = 3) +
  labs(title = "Sepal Length vs. Sepal Width",
       x = "Sepal Length",
       y = "Sepal Width",
       color = "Species") +
  theme_minimal() +
  theme(legend.title = element_text(size = 18),
        legend.text = element_text(size = 14),
        legend.key.size = unit(1.5, "cm"))

iris_scatter_plot
```

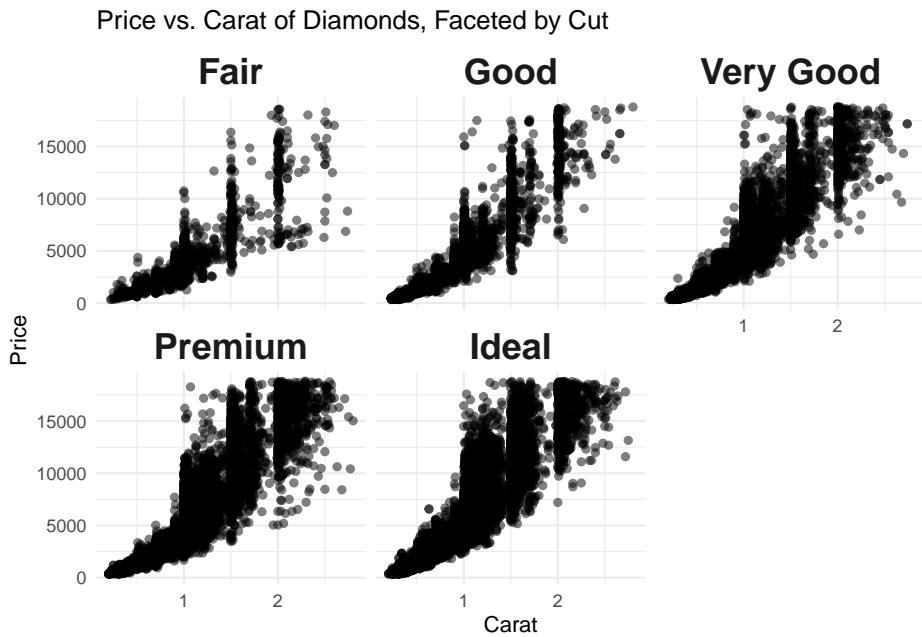


### 5.3.6 Adjusting the size of facet labels

You can control the size of facet labels using the `theme()` function along with `element_text()`.

```
diamonds_facet_plot <- ggplot(data = diamonds %>% filter(carat < 3), aes(x = carat, y = price)) +
  geom_point(alpha = 0.5) +
  facet_wrap(~cut) +
  labs(title = "Price vs. Carat of Diamonds, Faceted by Cut",
       x = "Carat",
       y = "Price") +
  theme_minimal() +
  theme(strip.text = element_text(size = 18, face = "bold"))

diamonds_facet_plot
```



### 5.3.7 Adjusting the size of axis ticks

You can modify the size of axis ticks using the `theme()` function along with `element_line()`.

```
scatter_plot_custom_ticks <- scatter_plot +
  theme(axis.ticks = element_line(size = 1.5),
        axis.ticks.length = unit(0.3, "cm"))

scatter_plot_custom_ticks
```

### 5.3.8 Adding text labels to points

Use the `geom_text()` or `geom_label()` functions to add text labels to points.

```
mtcars$car_name <- rownames(mtcars)

scatter_plot_labels <- ggplot(mtcars, aes(x = wt, y = mpg)) +
  geom_point(aes(color = gear)) +
  geom_text(aes(label = car_name), check_overlap = TRUE, vjust = 1.5) +
  labs(title = "Scatter plot of MPG vs Weight with Car Labels",
       x = "Weight",
```

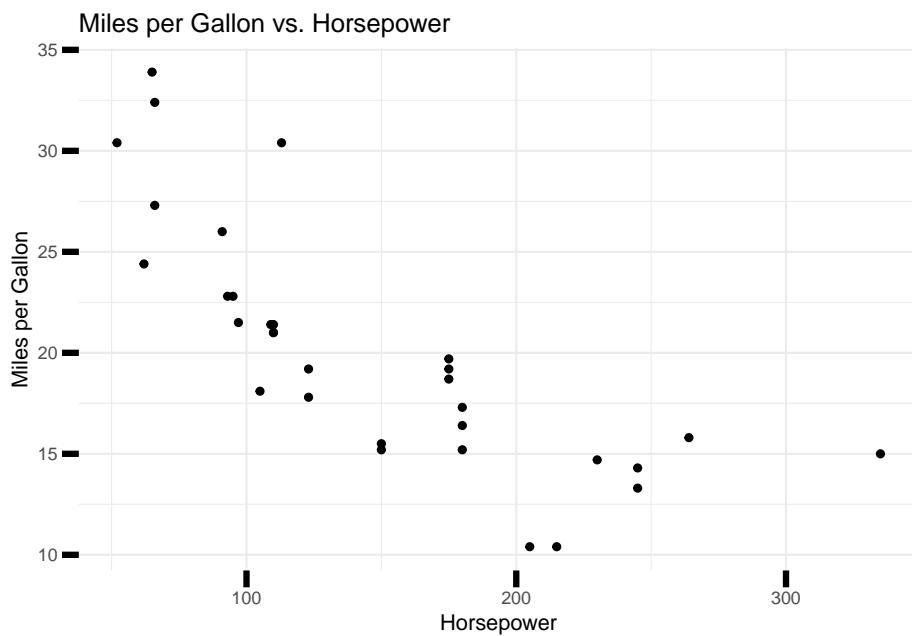
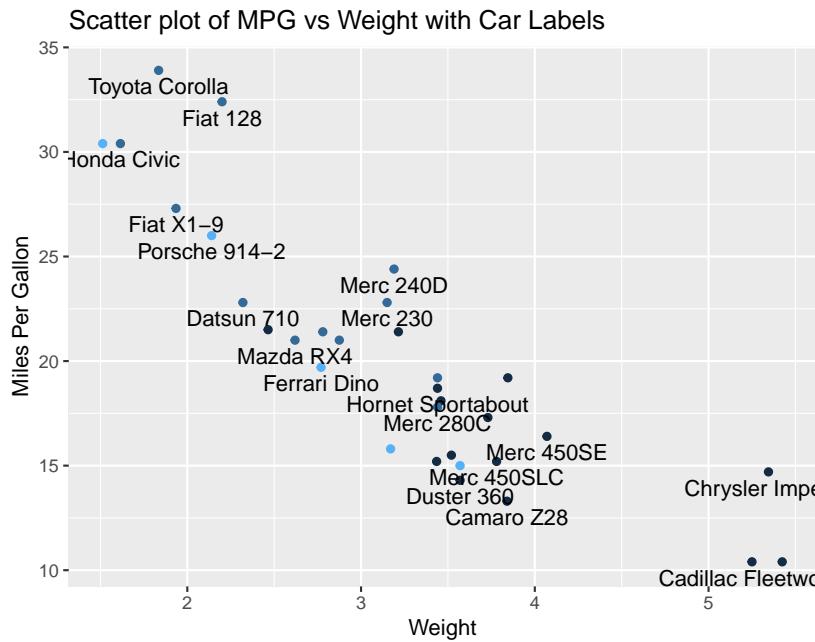


Figure 5.3: Figure 6: Scatterplot of mpg vs. hp with customized axis tick size.

```
y = "Miles Per Gallon",
color = "Gears")  
scatter_plot_labels
```

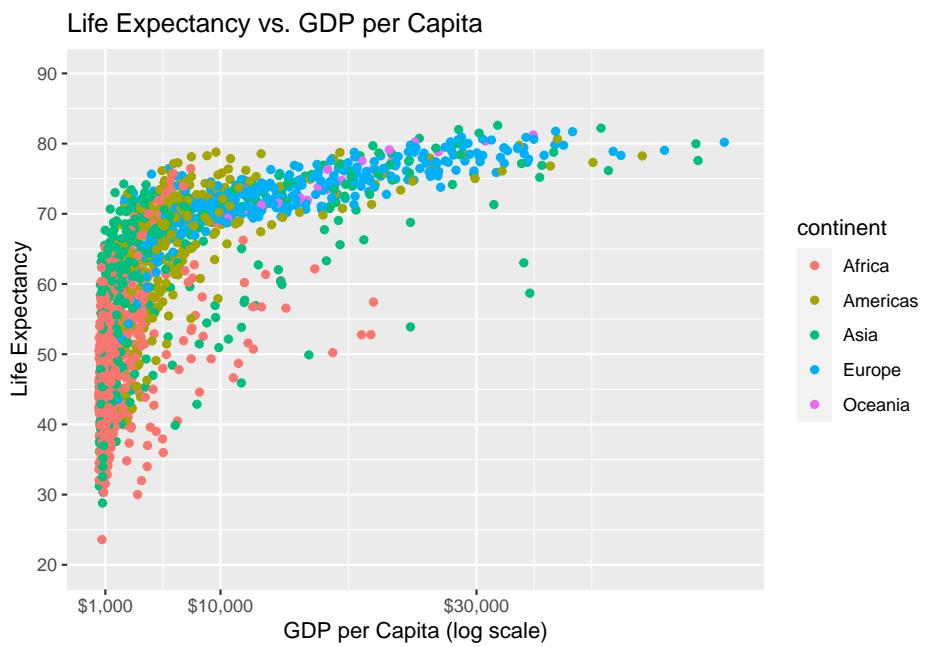


### 5.3.9 Modifying axis limits and scales

Use `scale_x_continuous()` and `scale_y_continuous()` to modify axis limits and scales.

```
data("gapminder", package = "gapminder")

ggplot(gapminder, aes(x = gdpPercap, y = lifeExp, color = continent)) +
  geom_point() +
  scale_x_log10() +
  scale_x_continuous(limits = c(500, 50000), breaks = c(1000, 10000, 30000), labels = s)
  scale_y_continuous(limits = c(20, 90), breaks = seq(20, 90, 10)) +
  labs(title = "Life Expectancy vs. GDP per Capita",
       x = "GDP per Capita (log scale)",
       y = "Life Expectancy")
```





# Chapter 6

## Table Formatting

In this worksheet, we will explore various options for outputting and formatting tables in R using the RMarkdown environment.

### 6.0.1 Basic Table Formatting with `kable`

The `kable()` function from the `knitr` package provides a simple way to output tables in RMarkdown.

```
library(knitr)
kable(mtcars[1:5, 1:5], caption = "A basic table using kable")
```

We will also use the `Gapminder` dataset for our examples. This dataset contains information about life expectancy, GDP per capita, and population size for various countries and years. Here's an example of how to display the first 10 rows of the `Gapminder` dataset.

```
data("gapminder", package = "gapminder")
knitr::kable(head(gapminder, 10), caption = "Table 1: First 10 rows of the Gapminder dataset.")
```

Table 6.1: A basic table using `kable`

|                   | mpg  | cyl | disp | hp  | drat |
|-------------------|------|-----|------|-----|------|
| Mazda RX4         | 21.0 | 6   | 160  | 110 | 3.90 |
| Mazda RX4 Wag     | 21.0 | 6   | 160  | 110 | 3.90 |
| Datsun 710        | 22.8 | 4   | 108  | 93  | 3.85 |
| Hornet 4 Drive    | 21.4 | 6   | 258  | 110 | 3.08 |
| Hornet Sportabout | 18.7 | 8   | 360  | 175 | 3.15 |

Table 6.2: Table 1: First 10 rows of the Gapminder dataset.

| country     | continent | year | lifeExp | pop      | gdpPercap |
|-------------|-----------|------|---------|----------|-----------|
| Afghanistan | Asia      | 1952 | 28.801  | 8425333  | 779.4453  |
| Afghanistan | Asia      | 1957 | 30.332  | 9240934  | 820.8530  |
| Afghanistan | Asia      | 1962 | 31.997  | 10267083 | 853.1007  |
| Afghanistan | Asia      | 1967 | 34.020  | 11537966 | 836.1971  |
| Afghanistan | Asia      | 1972 | 36.088  | 13079460 | 739.9811  |
| Afghanistan | Asia      | 1977 | 38.438  | 14880372 | 786.1134  |
| Afghanistan | Asia      | 1982 | 39.854  | 12881816 | 978.0114  |
| Afghanistan | Asia      | 1987 | 40.822  | 13867957 | 852.3959  |
| Afghanistan | Asia      | 1992 | 41.674  | 16317921 | 649.3414  |
| Afghanistan | Asia      | 1997 | 41.763  | 22227415 | 635.3414  |

Table 6.3: A formatted table with kableExtra

|                   | mpg  | cyl | disp | hp  | drat |
|-------------------|------|-----|------|-----|------|
| Mazda RX4         | 21.0 | 6   | 160  | 110 | 3.90 |
| Mazda RX4 Wag     | 21.0 | 6   | 160  | 110 | 3.90 |
| Datsun 710        | 22.8 | 4   | 108  | 93  | 3.85 |
| Hornet 4 Drive    | 21.4 | 6   | 258  | 110 | 3.08 |
| Hornet Sportabout | 18.7 | 8   | 360  | 175 | 3.15 |

### 6.0.2 Formatting Tables with kableExtra

To further customize the table appearance, we can use the `kableExtra` package.

```
#install.packages("kableExtra")
library(kableExtra)
kable(mtcars[1:5, 1:5], caption = "A formatted table with kableExtra") %>%
  kable_styling("striped", full_width = F)
```

### 6.0.3 Customizing column formats

Use the `column_spec()` function from the `kableExtra` package to customize the appearance of individual columns.

```
gapminder %>%
  head(10) %>%
  knitr::kable(caption = "Table 3: First 10 rows of the Gapminder dataset with custom column styling")
  kableExtra::kable_styling("striped", full_width = F) %>%
  kableExtra::column_spec(2, bold = TRUE, color = "red") %>%
  kableExtra::column_spec(4, monospace = TRUE)
```

Table 6.4: Table 3: First 10 rows of the Gapminder dataset with custom column formatting.

| country     | continent | year | lifeExp | pop      | gdpPerCap |
|-------------|-----------|------|---------|----------|-----------|
| Afghanistan | Asia      | 1952 | 28.801  | 8425333  | 779.4453  |
| Afghanistan | Asia      | 1957 | 30.332  | 9240934  | 820.8530  |
| Afghanistan | Asia      | 1962 | 31.997  | 10267083 | 853.1007  |
| Afghanistan | Asia      | 1967 | 34.020  | 11537966 | 836.1971  |
| Afghanistan | Asia      | 1972 | 36.088  | 13079460 | 739.9811  |
| Afghanistan | Asia      | 1977 | 38.438  | 14880372 | 786.1134  |
| Afghanistan | Asia      | 1982 | 39.854  | 12881816 | 978.0114  |
| Afghanistan | Asia      | 1987 | 40.822  | 13867957 | 852.3959  |
| Afghanistan | Asia      | 1992 | 41.674  | 16317921 | 649.3414  |
| Afghanistan | Asia      | 1997 | 41.763  | 22227415 | 635.3414  |

#### 6.0.4 Formatting Tables with flextable

Another option for table formatting is the `flextable` package.

```
#install.packages("flextable")
library(flextable)
ft <- flextable(mtcars[1:5, 1:5])
ft <- set_caption(ft, caption = "A table using flextable")
ft
```

Table 6.5: A table using `flextable`

| mpg  | cyl | disp | hp  | drat |
|------|-----|------|-----|------|
| 21.0 | 6   | 160  | 110 | 3.90 |
| 21.0 | 6   | 160  | 110 | 3.90 |
| 22.8 | 4   | 108  | 93  | 3.85 |
| 21.4 | 6   | 258  | 110 | 3.08 |
| 18.7 | 8   | 360  | 175 | 3.15 |

#### 6.0.5 Formatting Tables with gt

The `gt` package provides another way to create formatted tables in R.

```
#install.packages("gt")
library(gt)
gt(mtcars[1:5, 1:5]) %>%
  tab_header(title = "A table using gt")
```

A table using gt

| mpg  | cyl | disp | hp  | drat |
|------|-----|------|-----|------|
| 21.0 | 6   | 160  | 110 | 3.90 |
| 21.0 | 6   | 160  | 110 | 3.90 |
| 22.8 | 4   | 108  | 93  | 3.85 |
| 21.4 | 6   | 258  | 110 | 3.08 |
| 18.7 | 8   | 360  | 175 | 3.15 |

### 6.0.6 Hiding R commands and R output

As mentioned in the graph formatting handout, adding the chunk option echo=FALSE will display output (like graphs) produced by a chunk but not show the commands used in the chunk. You can stop both R commands and output from being displayed in a document by adding the chunk option include=FALSE.

As you work through a report analysis, you may initially want to see all of your R results as you are writing your report. But after you've summarized results in paragraphs or in tables, you can then use the include=FALSE argument to hide your R commands and output in your final document. If you ever need to rerun or reevaluate your R work for a report, you can easily recreate and edit your analysis since the R chunks used in your original report are still in your R Markdown .Rmd file.

### 6.0.7 Summary statistics with pander

We can use the pander package to create summary tables.

```
#install.packages("pander")
library(pander)
pander(summary(mtcars$mpg), caption = "Summary statistics for miles per gallon")
```

Table 6.7: Summary statistics for miles per gallon

| Min. | 1st Qu. | Median | Mean  | 3rd Qu. | Max. |
|------|---------|--------|-------|---------|------|
| 10.4 | 15.43   | 19.2   | 20.09 | 22.8    | 33.9 |

### 6.0.8 t-test results with pander

Let's perform a t-test comparing the miles per gallon (mpg) for cars with 4 and 6 cylinders.

```
t_test_result <- t.test(mpg ~ as.factor(cyl), data = mtcars, subset = cyl %in% c(4, 6))
pander(t_test_result, caption = "Comparing MPG for 4 and 6 cylinder cars")
```

Table 6.8: Comparing MPG for 4 and 6 cylinder cars (continued below)

| Test statistic  | df    | P value         | Alternative hypothesis |
|-----------------|-------|-----------------|------------------------|
| 4.719           | 12.96 | 0.0004048 * * * | two.sided              |
| mean in group 4 |       | mean in group 6 |                        |
| 26.66           |       | 19.74           |                        |

### 6.0.9 Chi-square test results with pander

Now let's perform a chi-square test to check for an association between the number of cylinders and the type of transmission (automatic or manual).

```
my_table <- table(mtcars$cyl, mtcars$am)
chisq_test_result <- chisq.test(my_table)
pander(chisq_test_result, caption = "Chi-square test for cylinders and transmission type")
```

Table 6.10: Chi-square test for cylinders and transmission type

| Test statistic | df | P value   |
|----------------|----|-----------|
| 8.741          | 2  | 0.01265 * |

### 6.0.10 Linear regression results with pandoc

Finally, let's fit a linear regression model of miles per gallon (mpg) as a function of weight (wt) and display the results.

```
lm_result <- lm(mpg ~ wt, data = mtcars)
pander(lm_result, caption = "Linear regression of MPG on weight")
```

Table 6.11: Linear regression of MPG on weight

|             | Estimate | Std. Error | t value | Pr(> t )  |
|-------------|----------|------------|---------|-----------|
| (Intercept) | 37.29    | 1.878      | 19.86   | 8.242e-19 |
| wt          | -5.344   | 0.5591     | -9.559  | 1.294e-10 |

## Chapter 7

# Report Guidelines

The data analysis report that you will be writing for this class will ask two things of you:

1. use appropriate statistical methods to answer research questions and
2. clearly and concisely communicate the meaning of your statistics and graphs to a **reader** who has a basic knowledge of statistics.

Your report should be organized, well-written with proper use of grammar, and contain sound reasoning and correct interpretations of statistical evidence. Also include *at least one* graphical display of your data on the body of your report. Be sure to hide the code used to produce it!

1. Your lab reports should be organized into the following sections:
  - **Introduction:** describe the data and your research questions
  - **Results:** describe your statistical analysis and interpret your graphs and numbers
  - **Discussion:** summarize your findings and answer your research questions, describe any limitations of your analysis
  - **Technical Appendix:** Staple all relevant R commands and output to the end of your written report. Only including commands without output is not enough. You must appropriately comment your code (telling me what each section of code is doing), and edit your code and output (no typos or errors allowed).
2. Type your report in Word/Google doc (or similar software) and use R (not Excel, Statkey or other software) for your analysis. You can also use R Markdown to write up your reports but you need to take care to only include labeled and numbered graphical output from R in the main

document. Commands and numerical output should be placed in the Technical Appendix. If you are interested in using Markdown talk to me for hints on its use for reports.

3. Carefully decide **appropriate graphs and numbers** to include. There is no need to show the same data in different forms (e.g. no need to show both a histogram and boxplot for the same variable). You also don't need to include all numbers given in R output if you only use a few in your analysis. You also don't need to "show" (or prove) skewness or outliers using numbers, just use your graphs to display skewness or outliers.
4. **Interpret** and give meaning to **all** graphs and numbers that you choose to include in your report. Do not include algebraic calculations or too much technical detail.
5. **Including Numbers:** Never include R numerical output or commands. Summarize needed output in a nicely formatted Word table or just integrate numbers into your writing. In you include tables, label them numerically (Table 1, Table 2, etc) and give each a title. Number in order of how they appear in the paper and refer to these tables by their number ("Table 1 displays summary statistics for income.").
6. **Including Graphs:** Resize all graphs appropriately so they fit nicely into your written report. Large graphs that take up most of a page with no, or very little, writing on the page impede the flow of the report and reduce its readability. Label all graphs numerically (Figure 1, etc.) as they occur in the paper, give each a title and refer to by number. See the stats lab manual chapter 1 if you need help copying plots into a Word/google doc.
7. **Do not explain every step taken in your study.** For example, there is no need to include a statement such as "I used R to create a histogram of income and observed that the distribution was right skewed". Instead just say "The distribution of income is skewed to the right (Figure 1)".
8. Avoid using weak phrases like "The average height of men is higher than the average for women." **Use numbers to bolster your explanation:** "The average height of men is three inches more than the average for women (68.5 vs. 65.5 inches)."
9. The **precision** of your data should dictate the precision of your statistics. In general, your statistics can have one to two more significant digits than your data. For example, if height is recorded to the nearest inch then the mean height should be reported as 65.5 (or 65.49) rather than the R value of 65.49268.
10. Sometimes a question posed in a study is **ambiguous** and there may be more than one way to correctly answer to the question. In grading your reports and paper, **I am most concerned with the logic of your**

conclusions and how you support your claim using data and statistical evidence.



# **Class Activity**



# Chapter 8

## Class Activity 1

### 8.1 Your Turn 1

---

- a. Run the following chunk. Comment on the output.

```
example_data = data.frame(ID = c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10),
                           Greeting = c(rep("Hello", 5), rep("Goodbye", 5)),
                           Male = rep(c(TRUE, FALSE), 5),
                           Weight = runif(n=10, 50, 300))
```

Click for answer

```
example_data
```

|    | ID | Greeting | Male  | Weight    |
|----|----|----------|-------|-----------|
| 1  | 1  | Hello    | TRUE  | 160.19413 |
| 2  | 2  | Hello    | FALSE | 172.02620 |
| 3  | 3  | Hello    | TRUE  | 56.10925  |
| 4  | 4  | Hello    | FALSE | 127.48601 |
| 5  | 5  | Hello    | TRUE  | 116.91639 |
| 6  | 6  | Goodbye  | FALSE | 66.74723  |
| 7  | 7  | Goodbye  | TRUE  | 93.67351  |
| 8  | 8  | Goodbye  | FALSE | 187.10819 |
| 9  | 9  | Goodbye  | TRUE  | 209.62473 |
| 10 | 10 | Goodbye  | FALSE | 175.89333 |

*Answer:* We see a data frame with four columns, where the first column is an **identifier** for the cases. We have information on the greeting types, whether male or not, and weight on these cases in the remaining columns.

- b. What is the dimension of the dataset called ‘example\_data’?

Click for answer

```
dim(example_data)
[1] 10  4
nrow(example_data)
[1] 10
ncol(example_data)
[1] 4
```

*Answer:* There are 10 rows and 4 columns.

---

## 8.2 Your Turn 2

- a. Read the dataset `EducationLiteracy` from the Lock5 second edition book.

Click for answer

```
# read in the data
library(readr)
education_lock5 <- read_csv("https://www.lock5stat.com/datasets2e/EducationLiteracy.csv")
```

- b. Print the header (i.e. first 6 cases by default) of the dataset in part a.

Click for answer

```
head(education_lock5)
```

```
# A tibble: 6 x 3
  Country          EducationExpenditure Literacy
  <chr>            <dbl>        <dbl>
1 Afghanistan      3.1         31.7
2 Albania          3.2         96.8
```

|                       |     |      |
|-----------------------|-----|------|
| 3 Algeria             | 4.3 | NA   |
| 4 Andorra             | 3.2 | NA   |
| 5 Angola              | 3.5 | 70.6 |
| 6 Antigua and Barbuda | 2.6 | 99   |

- c. What is the dimension of the dataset in a?

Click for answer

```
dim(education_lock5)
```

```
[1] 188    3
```

*Answer:* There are 188 rows and 3 columns.

- d. What type of variables are `Country`, `EducationExpenditure`, and `Literacy`?

Click for answer

*Answer:* `Country` is a categorical variable. `EducationExpenditure` and `Literacy` are both quantitative variables.

- e. If we would like to use education expenditure to predict the literacy rate of each countries, which variable is the explanatory variable and which one is the response?

Click for answer

*Answer:* The education expenditure is the explanatory variable, and the literacy rate is the response.



# Chapter 9

## Class Activity 2

### 9.1 Your Turn 1

#### 9.1.1 Summary of article on It depends on how you ask!

Click for answer

*Answer:*

This study aimed to measure the effects of psychological biases on estimates of compliance with public health guidance regarding COVID-19. Results showed that compliance estimates were reduced when questions were framed negatively and anonymity was increased. Effect sizes were large, with compliance estimates diminishing by up to 17% points and 10% points, respectively. These findings suggest that standard tracking surveys pose questions in ways that lead to higher compliance estimates than alternative approaches.

---

### 9.2 Your Turn 2

#### 9.2.1 Gettysburg random sample

Let's take a simple random sample (SRS) of Gettysburg words. The "population" is contained in the spreadsheet `GettysburgPopulationCounts.csv`. Carefully load this data into R:

```
pop <- read.csv("https://raw.githubusercontent.com/deepbas/statdatasets/main/GettysbergAddress.csv")
head(pop)
```

|   | position | size | word  |
|---|----------|------|-------|
| 1 | 1        | 4    | Four  |
| 2 | 2        | 5    | score |
| 3 | 3        | 3    | and   |
| 4 | 4        | 5    | seven |
| 5 | 5        | 5    | years |
| 6 | 6        | 3    | ago,  |

The `position` variable enumerates the list of words in the population (address).

(a). Sample

Run the following command to obtain a SRS of 10 words from the 268 that are in the population:

```
samp <- sample(1:268, size=10)
samp
```

```
[1] 50 213 170 71 60 197 236 196 115 217
```

This tells you the position (row number) of your sampled words. What are your sampled positions? Why are your sampled positions different from other folks in class?

(b). Get words and lengths

We will *subset* the data set `pop` to obtain only the sampled rows listed in `samp`. We do this using **square bracket notation** ‘dataset[row number, column number/name]’. Run the following command to find your sampled words and sizes:

```
pop[samp,]
```

|     | position | size | word      |
|-----|----------|------|-----------|
| 50  | 50       | 2    | so        |
| 213 | 213      | 4    | that      |
| 170 | 170      | 9    | dedicated |
| 71  | 71       | 7    | portion   |
| 60  | 60       | 5    | great     |
| 197 | 197      | 3    | the       |
| 236 | 236      | 4    | died      |
| 196 | 196      | 2    | to        |
| 115 | 115      | 6    | cannot    |
| 217 | 217      | 4    | they      |

- c. Compute your sample mean

The word lengths in part (b) are the data for your sample. You can compute your sample mean using a calculator, or using R. Let's try R (you will find it faster!). First save the quantitative variable `size` in a new variable called `mysize`:

```
mysize <- pop[samp, "size"]  
mysize
```

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

Then find the mean of these values:

```
mean(mysize)
```

```
[1] 4.6
```

How does this sample mean (from a truly random sample) compare to your sample mean from the non-random sample?

Click for answer

*Answer:* The true mean is 4.29. Your two means will likely vary. Since the many non-random samples generally overestimated the population mean length, it is possible (but not guaranteed) that *your* one non-random sample gave a mean length that is greater than the random sample's mean length.

### 9.2.2 Driving with a Pet on your Lap

Over 30,000 people participated in an online poll on `cnn.com` conducted in April 2012 asking: “Have you ever driven with a pet on your lap”? We see that 34% of the participants answered yes and 66% answered no.

- a. Can you conclude that a random sample was used from the description given? Explain.

Click for answer

*Answer:* No you can't make this conclusion from the info given. In fact, an online poll at a website like `cnn.com` is almost always reporting results from a non-random sample. The people who respond are individuals who visit `cnn.com`, then see the online poll and decide to respond.

- b. Explain why it is not appropriate to generalize these results to all drivers, or even to all drivers who visit cnn.com.

Click for answer

*Answer:* This is a volunteer sample, and volunteer samples are often biased and can't be generalized to *all drivers* (the population). It is likely that people who have driven with a pet on their lap are more likely to respond to the poll.

- c. How might we select a sample of people that would give us results that we can generalize to a broader population?

Click for answer

*Answer:* A random sample of individuals from all U.S. drivers would need to be selected and given the poll question. (There are many ways to do this, the most common being a variation of random digit dialing where phone numbers are randomly selected from known area codes.)

- d. Is the variable measured in this study quantitative or categorical?

Click for answer

*Answer:* Categorical (yes or no answer to the question).

# Chapter 10

## Class Activity 3

### 10.1 Case Study 1

Consider the following case study:

“Swimming with dolphins can certainly be fun, but is it also therapeutic for patients suffering from clinical depression? To investigate this possibility, researchers recruited 30 subjects aged 18-65 with a clinical diagnosis of mild to moderate depression. Subjects were required to discontinue use of any antidepressant drugs or psychotherapy four weeks prior to the experiment, and throughout the experiment. These 30 subjects went to an island off the coast of Honduras, where they were randomly assigned to one of two treatment groups. Both groups engaged in the same amount of swimming and snorkeling each day, but one group (the animal care program) did so in the presence of bottlenose dolphins and the other group (outdoor nature program) did not. At the end of two weeks, each subjects’ level of depression was evaluated, as it had been at the beginning of the study, and it was determined whether they showed substantial improvement (reducing their level of depression) by the end of the study (Antonioli and Reveley, 2005).”

Observed data:

The researchers found that 10 of 15 subjects in the dolphin therapy group showed substantial improvement, compared to 3 of 15 subjects in the control group.

- (a). Identify the observational units in this study.

Click for answer

*Answer:* The observational units in this study are the 30 subjects.

- (b). Classify each variable as categorical or quantitative.

Click for answer

*Answer:* The variables in this study can be classified as follows: Categorical: Treatment Group (Dolphin and Control),

Quantitative: Age, Depression Score (Beginning and End of Study)

(c). Which variable would you regard as explanatory and which as response?

Click for answer

*Answer:* The explanatory variable would be the Treatment Group and the response variable would be the Level of Depression.

(d). Is this an observational study or an experiment? Justify your answer.

Click for answer

*Answer:* This is an experiment because the researchers randomly assigned the subjects to the two treatment groups, and then observed the effect of the treatment (presence of dolphins) on the response variable (level of depression).

(e). Construct a two-way table based on the results of the experiment.

Click for answer

Two-way table:

| Treatment     | Improved | Not Improved | Total |
|---------------|----------|--------------|-------|
| Dolphin Group | 10       | 5            | 15    |
| Control Group | 3        | 12           | 15    |
| Total         | 13       | 17           | 30    |

---

## 10.2 Case Study 2

Consider the following case study:

“Researchers want to find out how a new diet affects weight gain among underweight subjects. This experiment only has two treatment conditions, the new diet and the standard diet. For this study, the researchers recruited 200 subjects which will be grouped into 100 pairs based on shared characteristics such as age, gender, weight, height, lifestyle, and so on. A 20-year-old female within the weight range of 90-110 pounds and the height range of 60-63 inches will be paired with another 20-year-old female that falls into the same weight and height categories. Once all 100 pairs are made, a subject from each pair will be randomly assigned into the treatment group (will be administered the new diet for 2 months) while the other subject from the pair will be assigned to the control group (will be assigned to follow the standard diet for two months).

At the end of the time period of 2 months, researchers will measure the total weight gain for each subject.”

Observed data:

The researchers found that 60 of 100 subjects in the new diet group showed substantial improvement, compared to 43 of 100 subjects in the standard diet group.

- (a). Identify the observational units in this study.

Click for answer

*Answer:* The observational units in this study are the 200 subjects.

- (b). Classify each variable as categorical or quantitative.

Click for answer

*Answer:* The variables are: age (quantitative), gender (categorical), weight (quantitative), height (quantitative), lifestyle (categorical), and total weight gain (quantitative).

- (c). Which variable would you regard as explanatory and which as response?

Click for answer

*Answer:* The explanatory variable is the type of diet (new or standard) and the response variable is the total weight gain.

- (d). Is this an observational study or an experiment? Justify your answer.

Click for answer

*Answer:* This is an experiment because the researchers are manipulating the explanatory variables (type of diet) to observe the effects on the response variables (total weight gain).

- (e). If it is an experiment, is it randomized comparative experiment or a matched pairs experiment?

Click for answer

*Answer:* This is a matched pairs experiment because each subject is paired with another subject who has similar characteristics and one subject from each pair is randomly assigned to the treatment group and the other to the control group. More specifically, this is a *pretest – posttest* matched pairs design.

- (f). Construct a two-way table based on the results of the experiment.

Click for answer

Two-way table:

| Outcome        | New Diet | Standard Diet | Total |
|----------------|----------|---------------|-------|
| Improvement    | 60       | 43            | 103   |
| No Improvement | 40       | 57            | 97    |
| Total          | 100      | 100           | 200   |

# Chapter 11

## Class Activity 4

### 11.1 Your Turn 1

#### 11.1.1 Flowers v. Mississippi

The data set `APM_DougEvansCases.csv` contains data from 1517 potential black and white jurors for 66 cases that Doug Evans was primary prosecutor for between 1992 and 2017. These jurors were available for Doug Evans to strike using his “peremptory strikes” during the jury selection phase.

(a). Inspect data

Read in the data

```
jurors <- read.csv("https://raw.githubusercontent.com/deepbas/statdatasets/main/APM_DougEvansCase
```

```
# dimension of dataset  
dim(jurors)
```

```
[1] 1517      6
```

Look at the first **three rows** of the data set

```
jurors[c(1,2,3), ]
```

|   | trial_id | race  | struck_state        | defendant_race |
|---|----------|-------|---------------------|----------------|
| 1 | 4        | Black | Not struck by State | White          |
| 2 | 4        | Black | Struck by State     | White          |
| 3 | 4        | White | Not struck by State | White          |

```

      same_race           struck_by
1 different race Juror chosen to serve on jury
2 different race           Struck by the state
3     same race Juror chosen to serve on jury

```

To get the data from one variable, we use the command `dataset$variable`. For example, `jurors$struck_state` gives us the data values from the `struck_state` variable, which tells us if a juror was struck by the state from the jury pool. Here we can see the first 10 entries in this variable:

```
jurors$struck_state[1:10]
```

```

[1] "Not struck by State" "Struck by State"
[3] "Not struck by State" "Not struck by State"
[5] "Struck by State"     "Not struck by State"
[7] "Struck by State"     "Not struck by State"
[9] "Not struck by State" "Not struck by State"

```

(b). Table of counts and proportions

The `summary` command used with a data frame gives summaries of each variable

```
summary(jurors)
```

```

trial_id          race        struck_state
Min.   : 4.0    Length:1517      Length:1517
1st Qu.: 52.0   Class :character Class :character
Median  : 82.0   Mode   :character Mode   :character
Mean    :112.6
3rd Qu.:170.0
Max.   :301.0
defendant_race   same_race      struck_by
Length:1517      Length:1517      Length:1517
Class :character Class :character Class :character
Mode  :character Mode  :character Mode  :character

```

The `table` command gives the distribution of counts for a single categorical variable. To obtain the count table for `struck_state` you need to

```
counts <- table(jurors$struck_state)
counts
```

|                     |                 |
|---------------------|-----------------|
| Not struck by State | Struck by State |
| 1084                | 433             |

We can add the `prop.table` command to turn these counts into proportions:

```
prop.table(counts)
```

|                     |                 |
|---------------------|-----------------|
| Not struck by State | Struck by State |
| 0.7145682           | 0.2854318       |

- What proportion of eligible jurors were struck by the state from the jury pool?

Click for answer

*Answer:* about 28.5% of eligible jurors were struck by the state.

(c). Bar graph for one variable

We can create a data frame `count_data` containing the counts and their corresponding categories.

```
count_data <- data.frame(counts)
count_data
```

|   | Var1                | Freq |
|---|---------------------|------|
| 1 | Not struck by State | 1084 |
| 2 | Struck by State     | 433  |

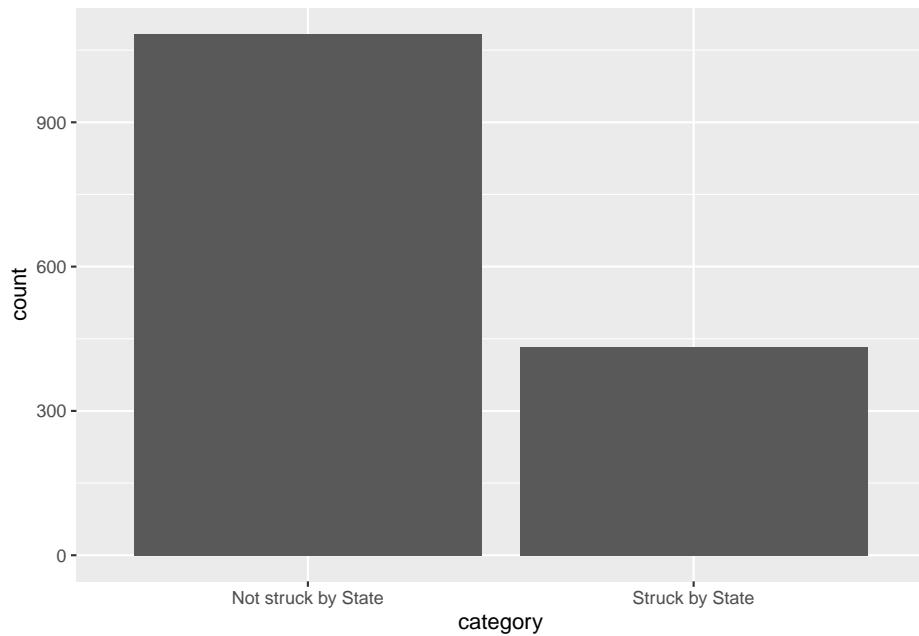
Then, we use `ggplot2` to create a bar plot with the categories on the x-axis and the counts on the y-axis. The column names of `count_data` are automatically assigned to be `Var1` and `Freq`. We can change the column names to `category` and `count`, for example, as:

```
colnames(count_data) = c("category", "count")
count_data
```

|   | category            | count |
|---|---------------------|-------|
| 1 | Not struck by State | 1084  |
| 2 | Struck by State     | 433   |

The `geom_bar(stat = "identity")` function is used to create the bars, and we set the y-axis label using `labs()`.

```
# Create a bar plot using ggplot2
library(ggplot2) # load the package
ggplot(count_data, aes(x = category, y = count)) +
  geom_bar(stat = "identity") +
  labs(y = "count")
```



(d). Two-way tables

First 10 entries of `race` and `struck_state` variable is

```
jurors[(1:10),(2:3)]
```

|    | race  | struck_state        |
|----|-------|---------------------|
| 1  | Black | Not struck by State |
| 2  | Black | Struck by State     |
| 3  | White | Not struck by State |
| 4  | White | Not struck by State |
| 5  | Black | Struck by State     |
| 6  | White | Not struck by State |
| 7  | Black | Struck by State     |
| 8  | White | Not struck by State |
| 9  | White | Not struck by State |
| 10 | White | Not struck by State |

The `table` command also gives two-way tables when two variables are included. Here is the two-way table for juror race and state struck status:

```
mytable <- table(jurors$race, jurors$struck_state)
mytable
```

|       | Not struck by State | Struck by State |
|-------|---------------------|-----------------|
| Black | 225                 | 310             |
| White | 859                 | 123             |

- How many jurors were white and were not struck by the state?

Click for answer

*answer:* 859

(e). Conditional proportions: state strike status by juror race

The `prop.table` command gives conditional proportions for a two-way table. We plug our two-way table into `prop.table` with a `margin=1` to get proportions grouped by the `row` variable:

```
prop.table(mytable, margin = 1)
```

|       | Not struck by State | Struck by State |
|-------|---------------------|-----------------|
| Black | 0.4205607           | 0.5794393       |
| White | 0.8747454           | 0.1252546       |

Of all eligible black jurors, about 57.9% were struck by the state.

- What proportion of eligible white jurors were struck by the state?

Click for answer

*answer:* about 12.5%

- Is there evidence of an association between juror race and state strikes?

Click for answer

*answer:* Yes, there is an association because the rate of state strikes varies greatly by juror race with about 60% of black jurors were struck compared to only 13% of white jurors

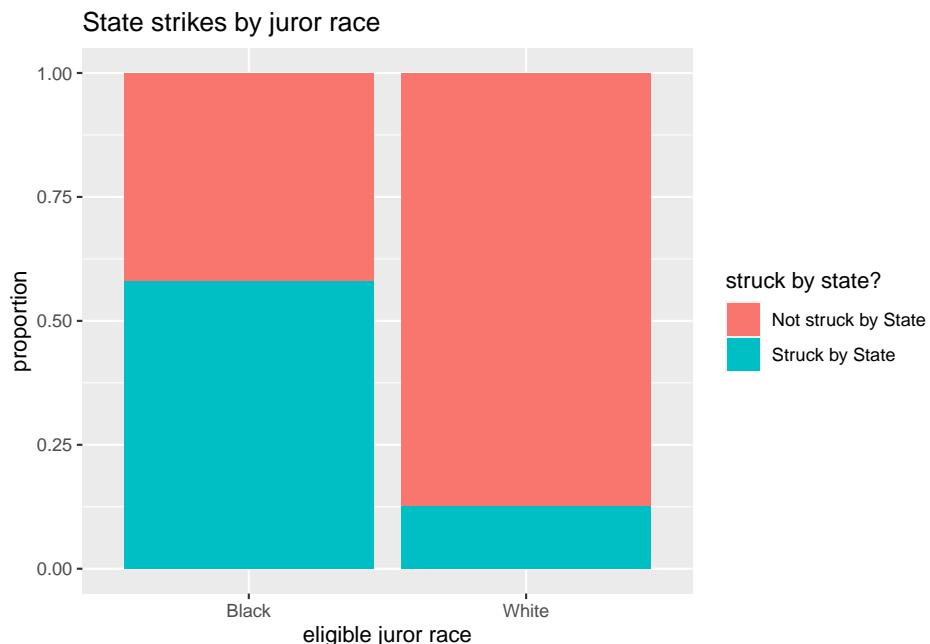
(f). Stacked bar graph for two variables

We can visualize the conditional distribution from part (e) with a stacked bar graph created using the `ggplot2` graphing package. First, load this package's functions with the `library` command:

```
library(ggplot2)
```

Now we can use the `geom_bar` command in this package. Here we get the conditional distribution of `struck_state` given `race`:

```
ggplot(jurors, aes(x = race, fill = struck_state)) +
  geom_bar(position = "fill") +
  labs(title = "State strikes by juror race", y = "proportion",
       x = "eligible juror race", fill = "struck by state?")
```



The basic syntax for this function is to let `ggplot` know your data set name (`jurors`), then specify the grouping or conditional variable on the x-axis (`race`) in the `aes` (aesthetic) argument. The `fill` variable is the response variable (`struck_state`). We add (+) the `geom_bar` geometry to get a bar graph with the `fill` position specified. Adding an informative label and title complete the graph.

(g). Conditional distribution of race grouped by strike status

We can “flip” our response and grouping variables easily (if we think it makes sense to do so). Here we specify the `margin=2` to get proportions grouped by the **column** variable:

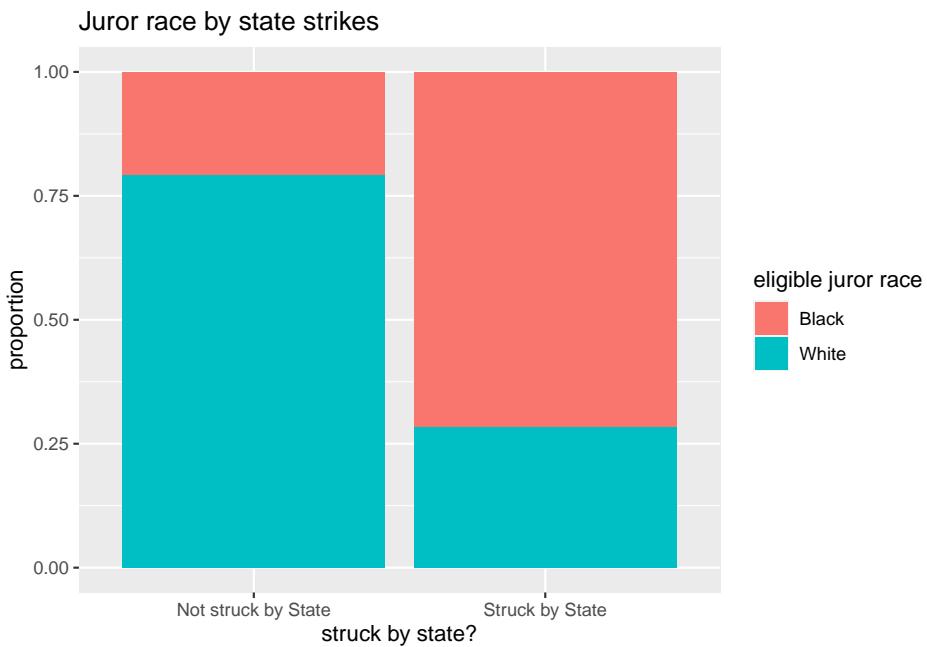
```
prop.table(mytable, margin = 2)
```

|       | Not struck by State | Struck by State |
|-------|---------------------|-----------------|
| Black | 0.2075646           | 0.7159353       |
| White | 0.7924354           | 0.2840647       |

Notice that the proportions add to one **down** each column. Of all eligible jurors struck by the state, about 71.6% were black.

The stacked bar graph for this distribution is

```
ggplot(jurors, aes(x = struck_state, fill = race)) +
  geom_bar(position = "fill") +
  labs(title = "Juror race by state strikes", y = "proportion",
       fill = "eligible juror race", x = "struck by state?")
```



- What proportion of eligible jurors who were not struck by the state were black? were white?

Click for answer

*Answer:* Of all jurors not struck by the state, about 20.8% were black

---

## 11.2 Your Turn 2

### 11.2.1 Graduate programs acceptance and sex

How are grad school program acceptance rates associated with sex? We will look at a classic data set from Berkeley grad school applications from 1973 (*Science*, 1975). The data cases are applicants to four graduate programs at Berkeley during 1973. The variable `result` tells us if the applicant was accepted to the graduate program, `sex` tells us the sex of the applicant (male or female), and `program` tells us program type (programs 1,2,3 or 4).

```
grad <- read.csv("https://raw.githubusercontent.com/deepbas/statdatasets/main/Berkeley
```

```
# dimension of the dataset
dim(grad)
```

```
[1] 3014     3
```

```
# first 6 rows
head(grad)
```

```
      program  sex result
1 program1 male accept
2 program1 male accept
3 program1 male accept
4 program1 male accept
5 program1 male accept
6 program1 male accept
```

(a). Table of counts and proportions

```
prop.table(table(grad$result))
```

|           | accept    | reject |
|-----------|-----------|--------|
| 0.4260119 | 0.5739881 |        |

- What proportion of applicants were accepted?

Click for answer

*Answer:* About 43% (1284/3014) of applicants were accepted.

(b). Two-way tables

The `table` command also gives two-way tables when two variables are included. Here is the two-way table for result and sex:

```
table(grad$sex, grad$result)
```

|        | accept | reject |
|--------|--------|--------|
| female | 262    | 587    |
| male   | 1022   | 1143   |

- How many applicants involved females who were accepted?

Click for answer

*Answer:* : 262 applicants involved females who were accepted.

(c). Conditional proportions: acceptance given sex

The `prop.table` command gives conditional proportions for a two-way table. First let's save the two-way table in an object named `mytable`:

```
mytable <- table(grad$sex, grad$result)
```

Then use `prop.table` to get the distribution of result conditioned (grouped) on applicant's sex:

```
prop.table(mytable, 1)
```

|        | accept    | reject    |
|--------|-----------|-----------|
| female | 0.3085984 | 0.6914016 |
| male   | 0.4720554 | 0.5279446 |

The value of 1 in this command tell's R that you want *row* proportions (the denominator of the proportion is each row total).

- What proportion of female were accepted?

Click for answer

*Answer:* about 31% ( $262/(262+587)$ )

- What proportion of males were accepted?

Click for answer

*Answer:* about 47% ( $1022/(1022+1143)$ )

(d). Bar graph for one variable

We can create a data frame `count_data1` containing the counts and their corresponding categories.

```
counts1 <- table(grad$result)
count_data1 <- data.frame(counts1)
```

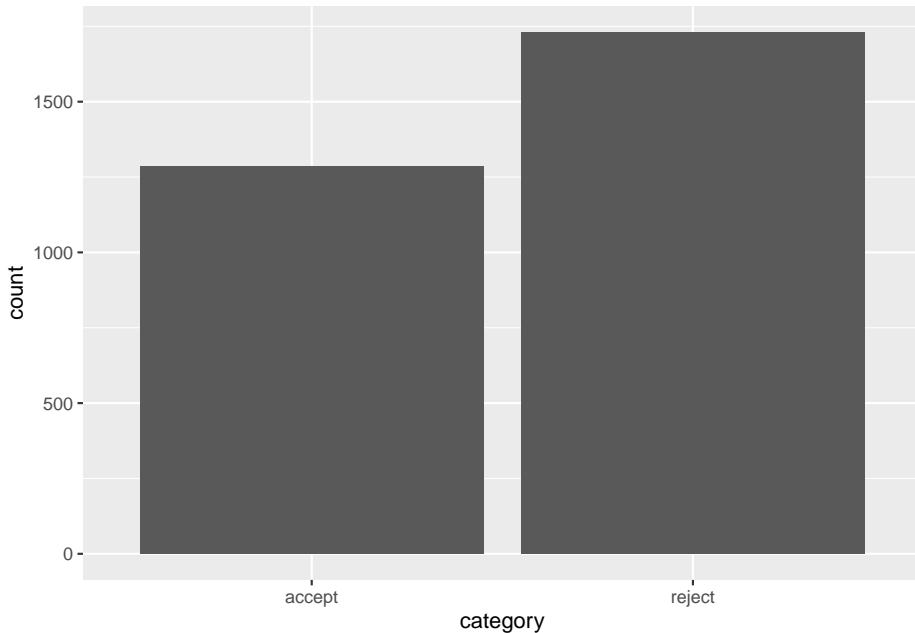
Then, we use `ggplot2` to create a bar plot with the categories on the x-axis and the counts on the y-axis. The column names of `count_data` are automatically assigned to be `Var1` and `Freq`. We can change the column names to `category` and `count`, for example, as:

```
colnames(count_data1) = c("category", "count")
count_data1
```

|   | category | count |
|---|----------|-------|
| 1 | accept   | 1284  |
| 2 | reject   | 1730  |

The `geom_bar(stat = "identity")` function is used to create the bars, and we set the y-axis label using `labs()`.

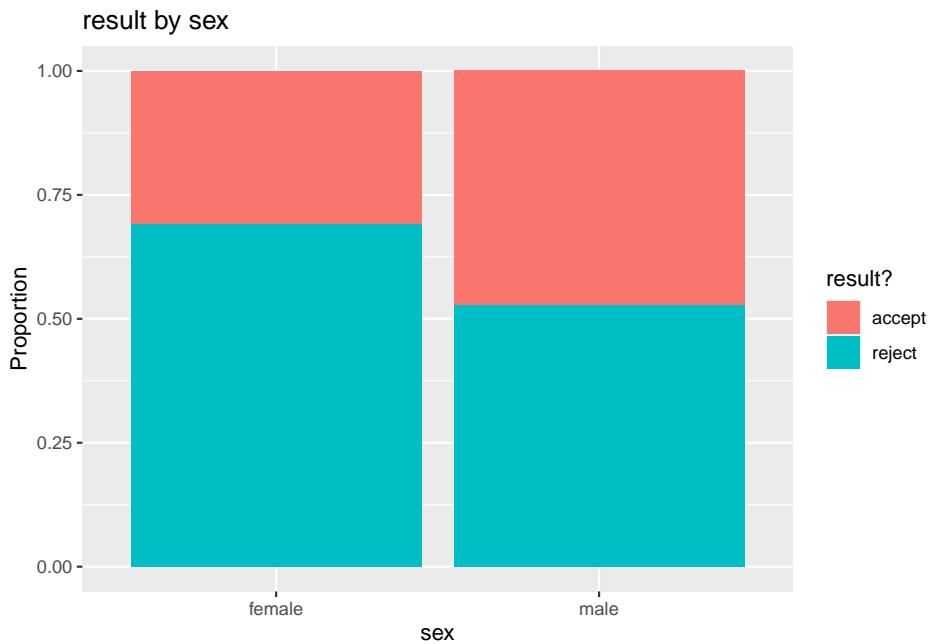
```
# Create a bar plot using ggplot2
library(ggplot2) # load the package
ggplot(count_data1, aes(x = category, y = count)) +
  geom_bar(stat = "identity") +
  labs(y = "count")
```



(e). Stacked bar graph for two variables

Now we can use the `geom_bar` command in this package. Here we get the conditional distribution of `result` given `sex`:

```
library(ggplot2) # don't need if you already entered it for example 1
ggplot(grad, aes(x = sex, fill = result)) +
  geom_bar(position = "fill") +
  labs(y="Proportion", title = "result by sex", fill = "result?", x = "sex")
```



The basic syntax for this function is to let `ggplot` know your data set name (`grad`), then specify the grouping or conditional variable on the x-axis (`sex`) in the `aes` (aesthetic) argument. The `fill` variable is the response variable (`result`). We add (+) the `geom_bar` geometry to get a bar graph with the `fill` position specified. Adding an informative label and title complete the graph.

- Verify that this graph is plotting the conditional proportions from part (c)

#### (f). Subsetting by program type

Finally, we will repeat the previous analysis of result and sex, but this time we will divide (or subset) the data set by program type. To do this we need to know how the values of `program` are coded:

```
table(grad$program)
```

```
program1 program2 program3 program4
      933       585       782       714
```

Here we use the `filter` command available from the `dplyr` package to get only the applicants to program 1:

```
library(dplyr)
grad.p1 <- filter(grad, program == "program1") # gets rows where program equal program1
head(grad.p1)
```

```
program sex result
1 program1 male accept
2 program1 male accept
3 program1 male accept
4 program1 male accept
5 program1 male accept
6 program1 male accept
```

```
dim(grad.p1)
```

```
[1] 933    3
```

Verify that the number of rows in the subsetted program 1 data set matches the number of program 1 applicants shown in the `table` of counts above.

- Repeat the `filter` command to get a data set for program 2 and call the new data set `grad.p2`. Verify that the number of rows in this dataset matches the number of program 2 applicants in the original data set.

```
# enter R code for (f) here
grad.p2 <- filter(grad, program == "program2") # gets rows where program equal program1
head(grad.p2)
```

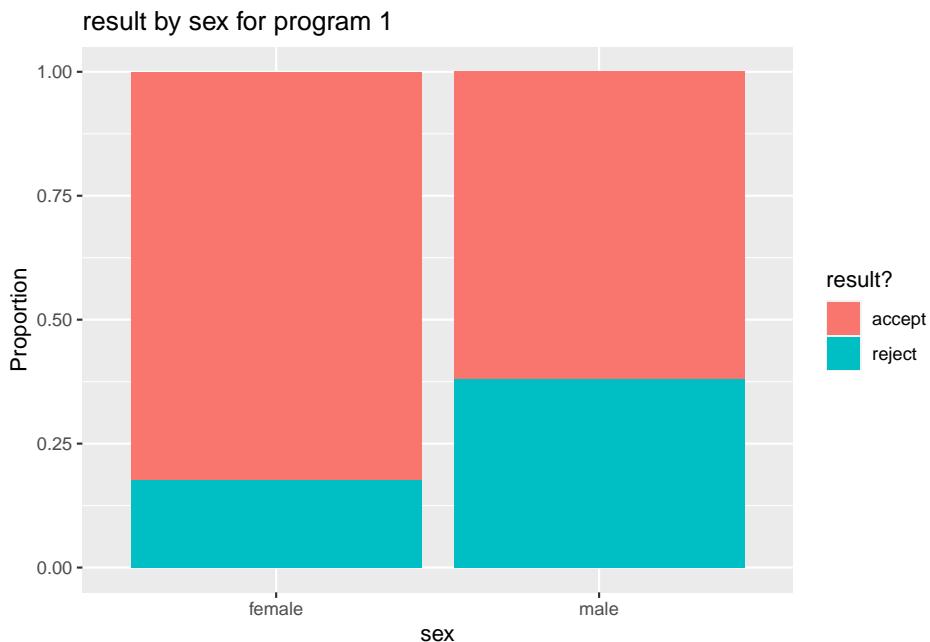
```
program sex result
1 program2 male accept
2 program2 male accept
3 program2 male accept
4 program2 male accept
5 program2 male accept
6 program2 male accept
```

(g). Result by sex for program 1.

- Show the distribution of result conditioned on applicant's sex for the program 1 data set. Get both a table of conditional proportions (or percentages) and a stacked bar graph.

Click for answer

```
# enter R code for (g) here
ggplot(grad.p1, aes(x = sex, fill = result)) +
  geom_bar(position = "fill") +
  labs(y="Proportion", title = "result by sex for program 1",
       fill = "result?", x = "sex")
```



```
prop.table(table(grad.p1$sex, grad.p1$result), 1)
```

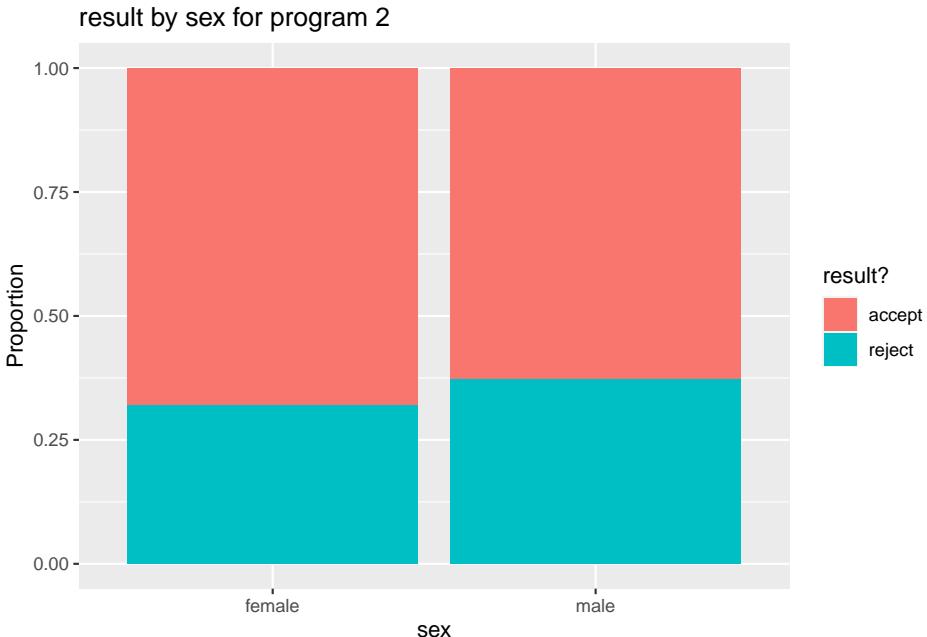
|        | accept    | reject    |
|--------|-----------|-----------|
| female | 0.8240741 | 0.1759259 |
| male   | 0.6193939 | 0.3806061 |

(h). Result by sex for program 2.

- Repeat part (g) but this time use the program 2 data set. Compare the two bar graphs for (g) and (h) and explain how they show that females have a higher acceptance rate after accounting for program type (1 or 2).

Click for answer

```
# enter R code for (h) here
ggplot(grad.p2, aes(x = sex, fill = result)) +
  geom_bar(position = "fill") +
  labs(y="Proportion", title = "result by sex for program 2",
       fill = "result?", x = "sex")
```



```
prop.table(table(grad.p2$sex, grad.p2$result), 1)
```

|        | accept    | reject    |
|--------|-----------|-----------|
| female | 0.6800000 | 0.3200000 |
| male   | 0.6285714 | 0.3714286 |

*Answer:* For both programs 1 and 2, we see that female applicants have a slightly higher rate of acceptance than male applicants. After accounting for program type, we now see that black defendants have a higher rate of death penalty than white defendants. Without accounting for program type, the opposite was true (see parts (c) and (e)).

Why? the confounding affect of program type which is associated with both result and sex:

Click for answer

- females prefer to apply to programs 3 and 4 while males prefer programs 1 and 2 (more than 3 and 4).
  - 44% of females applied to program 3 and 40% to program 4
  - 38% of males applied to program 1 and 26% to program 2

```
prop.table(table(grad$sex, grad$program), 1)
```

|        | program1   | program2   | program3   | program4   |
|--------|------------|------------|------------|------------|
| female | 0.12720848 | 0.02944641 | 0.44169611 | 0.40164900 |
| male   | 0.38106236 | 0.25866051 | 0.18799076 | 0.17228637 |

-Programs 3 and 4 were much harder to get into than programs 1 and 2 - 64% of applicants to program 1 were accepted and 63% of applicants to program 2 were accepted - 6% of applicants to program 4 were accepted and 34% of applicants to program 3 were accepted

```
prop.table(table(grad$program, grad$result), 1)
```

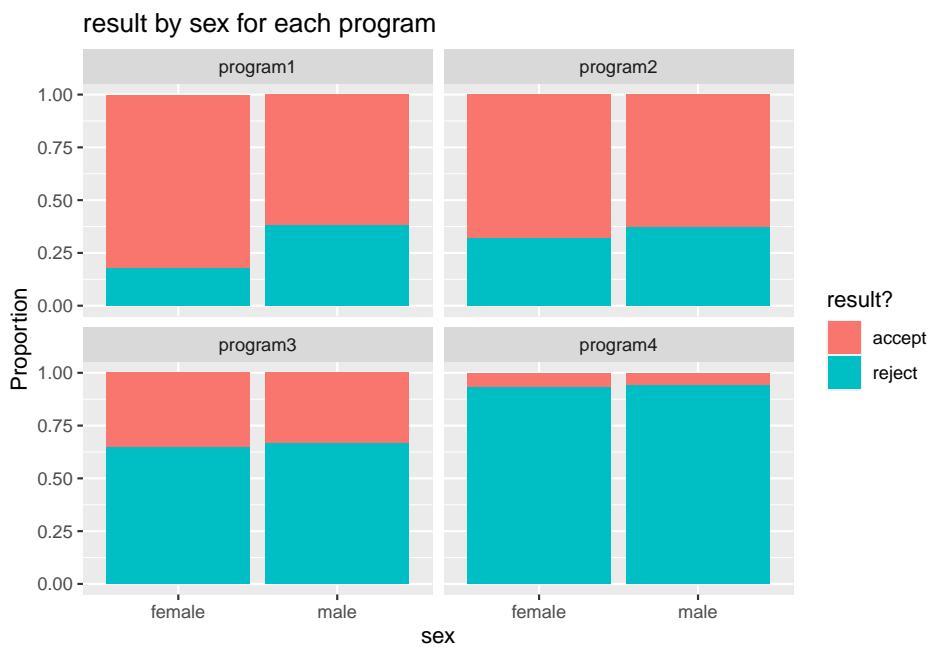
|          | accept     | reject     |
|----------|------------|------------|
| program1 | 0.64308682 | 0.35691318 |
| program2 | 0.63076923 | 0.36923077 |
| program3 | 0.34398977 | 0.65601023 |
| program4 | 0.06442577 | 0.93557423 |

So since the majority of females applied to the toughest programs (as measured by acceptance rates), there overall rate of acceptance was lower for females compared to males. But when we break down these rates by program type, we see that females have higher acceptance rates than males (see the visual in part (i)).

(i). A bar graph with three variables

If we simply want to graph the relationship between result and sex for each type of program, we can avoid subsetting the data by using the `facet_wrap` command in `ggplot2`. It is one simple addition to the stacked bar graph in part (e):

```
ggplot(grad, aes(x = sex, fill = result)) +
  geom_bar(position = "fill") +
  labs(y="Proportion",
       title = "result by sex for each program",
       fill = "result?",
       x = "sex") +
  facet_wrap(~program)
```



- Verify that this command creates side-by-side stacked bar graphs that match your graphs in parts (g) and (h) for programs 1 and 2.

Click for answer

*Answer:* The graphs match.