Stat 120

Deepak Bastola

2023-03-26

Contents

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 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, `$\frac{1}{n} \sum_{i=1}^{n} x_{i}$` to produce $\frac{1}{n}\sum_{i=1}^{n} x_i$ 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
```

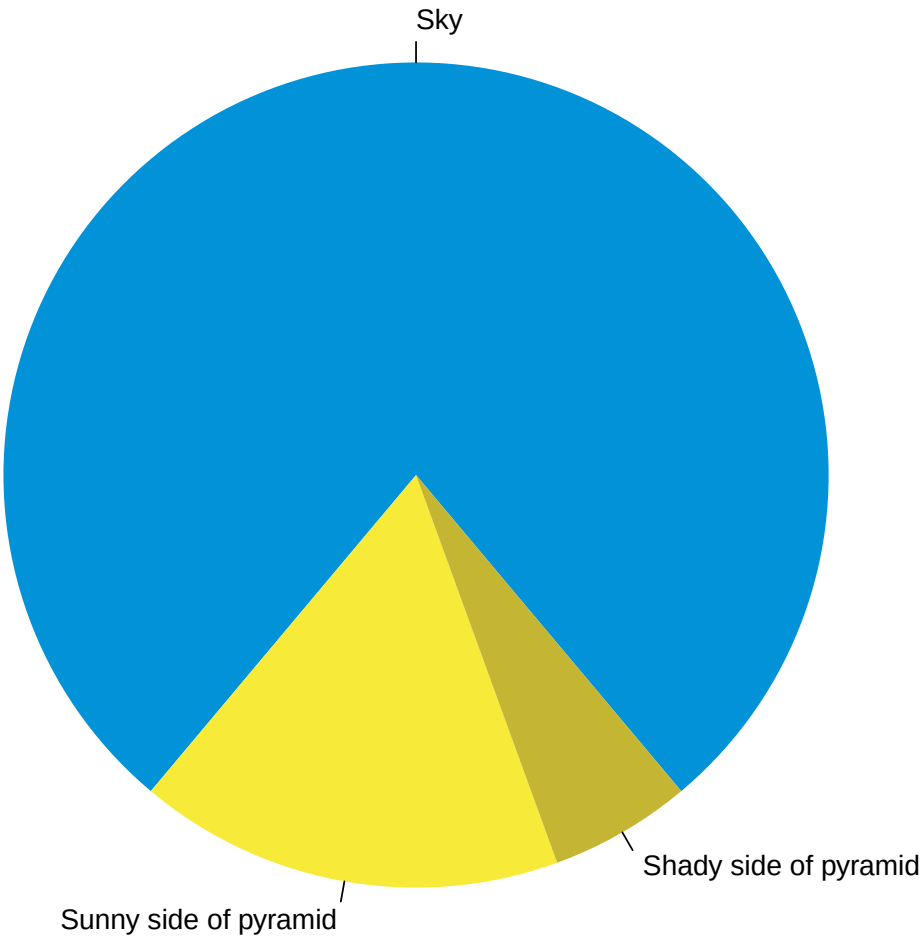


Figure 2.1: A fancy pie chart.

```
 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.

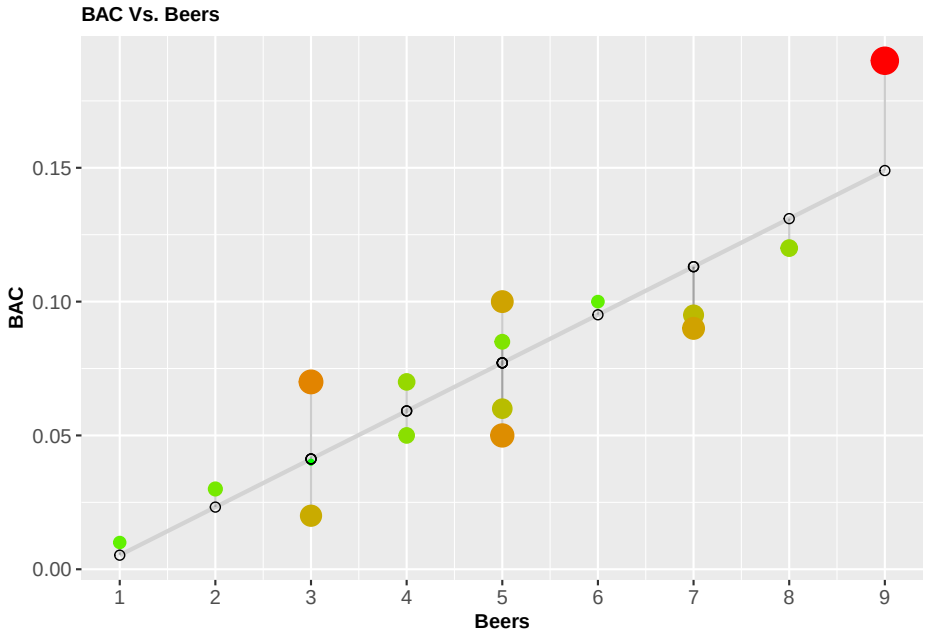
initials	state	age	time
vib	MA	61	6:01
adc	TX	45	5:45
kme	CT	50	4:19

Chapter 3

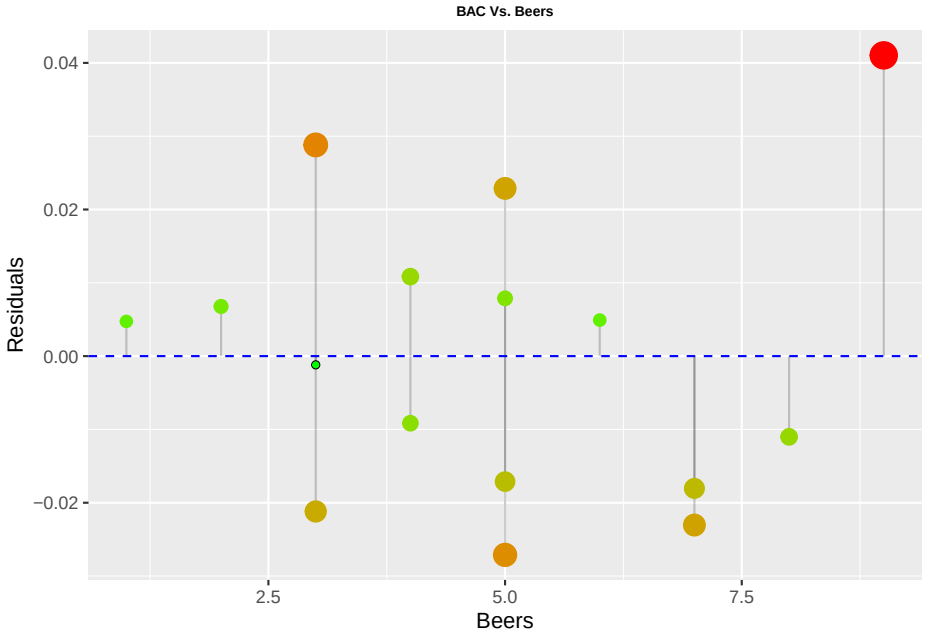
Helpful R codes

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 poin
  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 relations
  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()
```

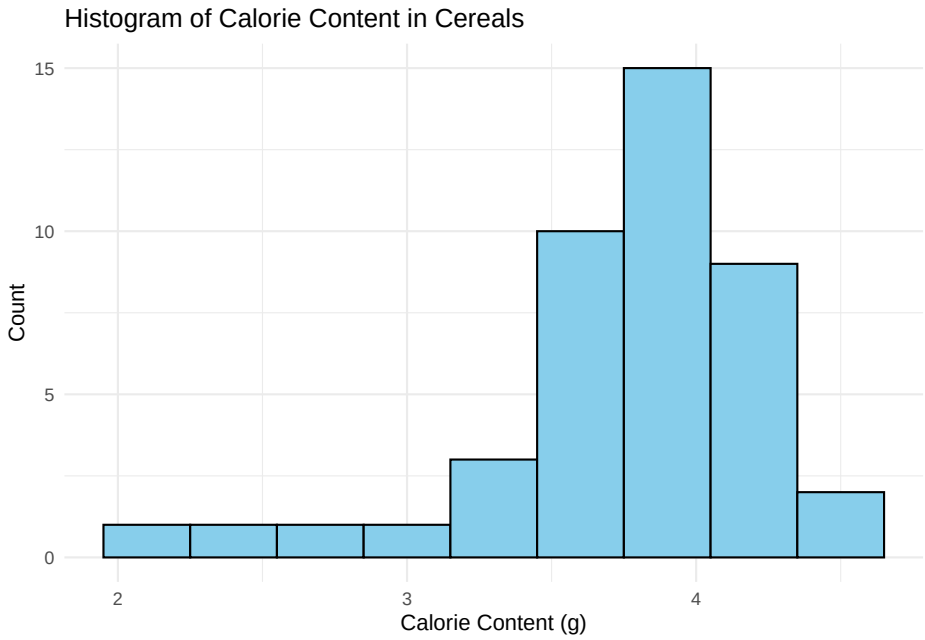
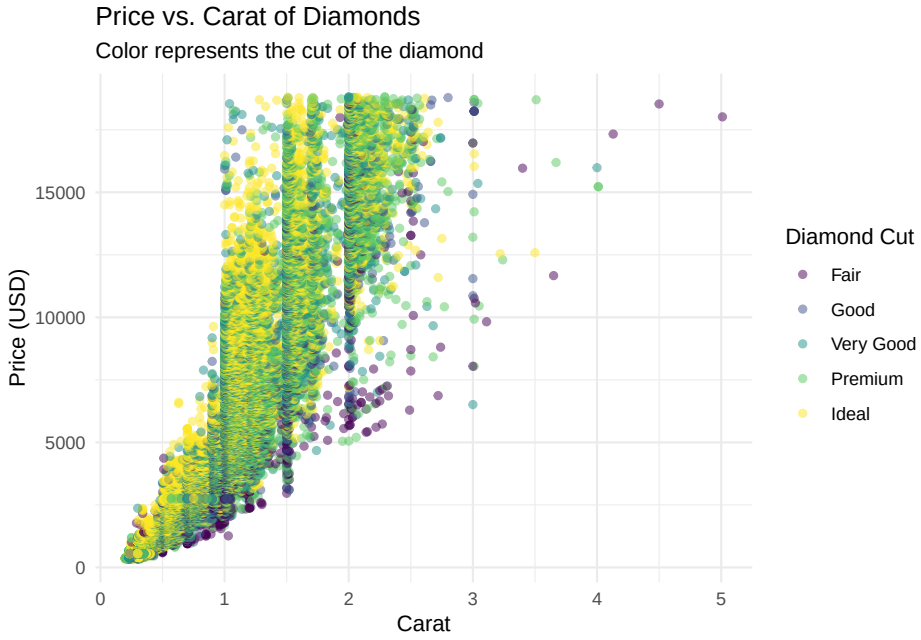


Figure 5.1: A nice figure

5.2.2 Customizing titles, axis labels, and legends

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

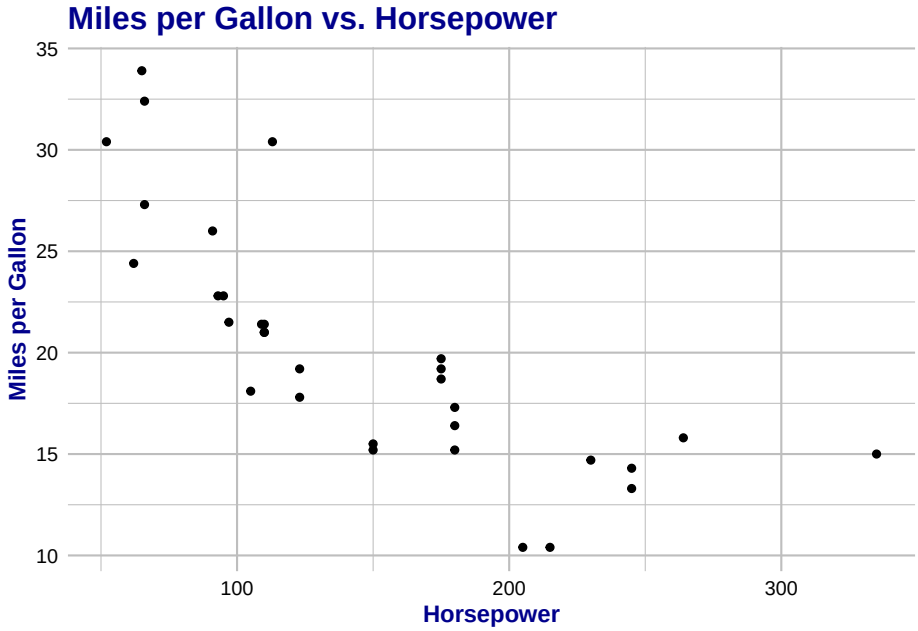
```
ggplot(data = diamonds, aes(x = carat, y = price, color = cut)) +
  geom_point(alpha = 0.5) +
  labs(title = "Price vs. Carat of Diamonds",
       subtitle = "Color represents the cut of the diamond",
       x = "Carat",
       y = "Price (USD)",
       color = "Diamond Cut") +
  theme_minimal()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



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))) +
```

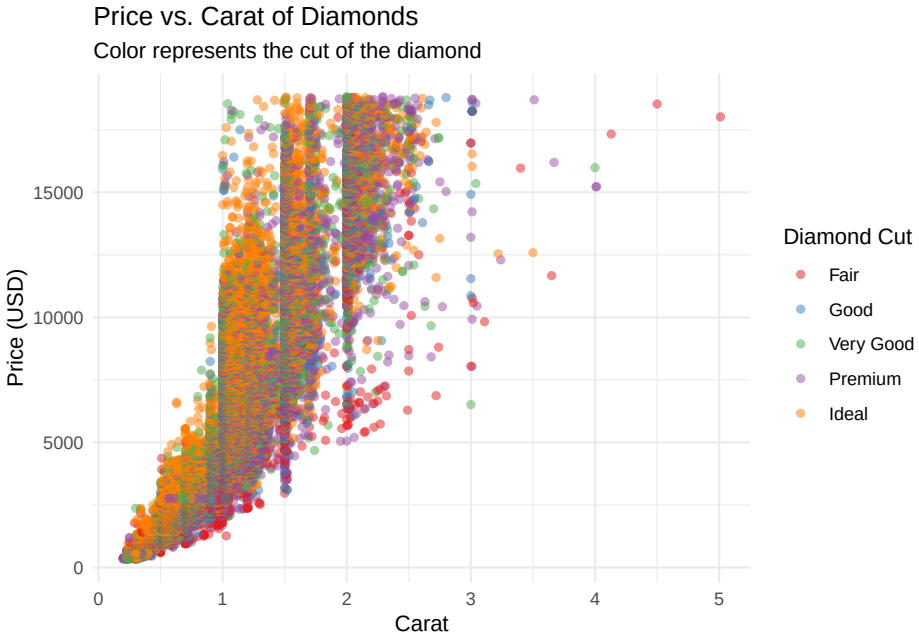
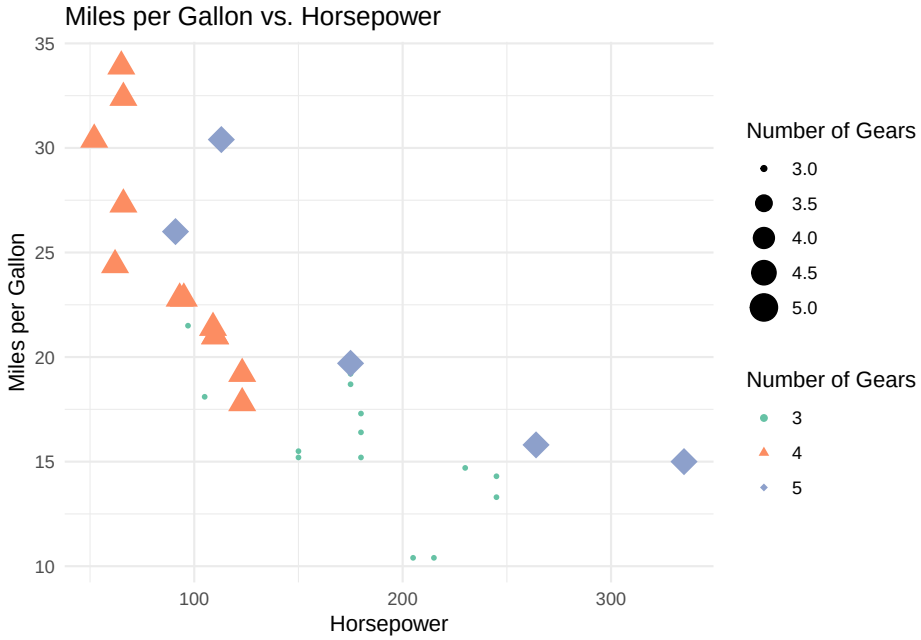


Figure 5.2: Figure 4: Scatterplot of price vs. carat of diamonds with color representing the cut and custom color scale.

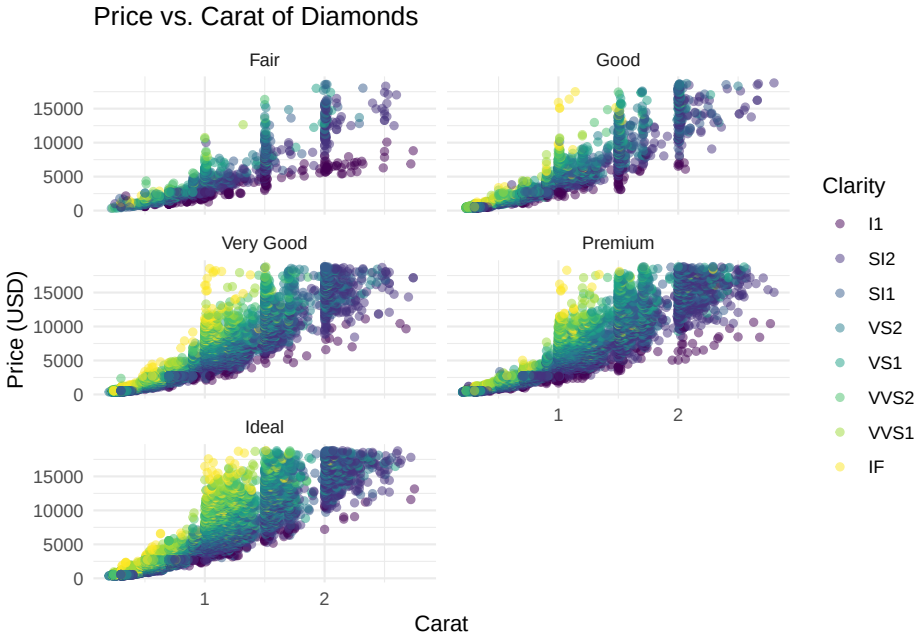
```
labs(title = "Miles per Gallon vs. Horsepower",
     x = "Horsepower",
     y = "Miles per Gallon",
     color = "Number of Gears",
     shape = "Number of Gears",
     size = "Number of Gears") +
theme_minimal() +
scale_shape_manual(values = c(16, 17, 18)) +
scale_color_brewer(palette = "Set2")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



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.

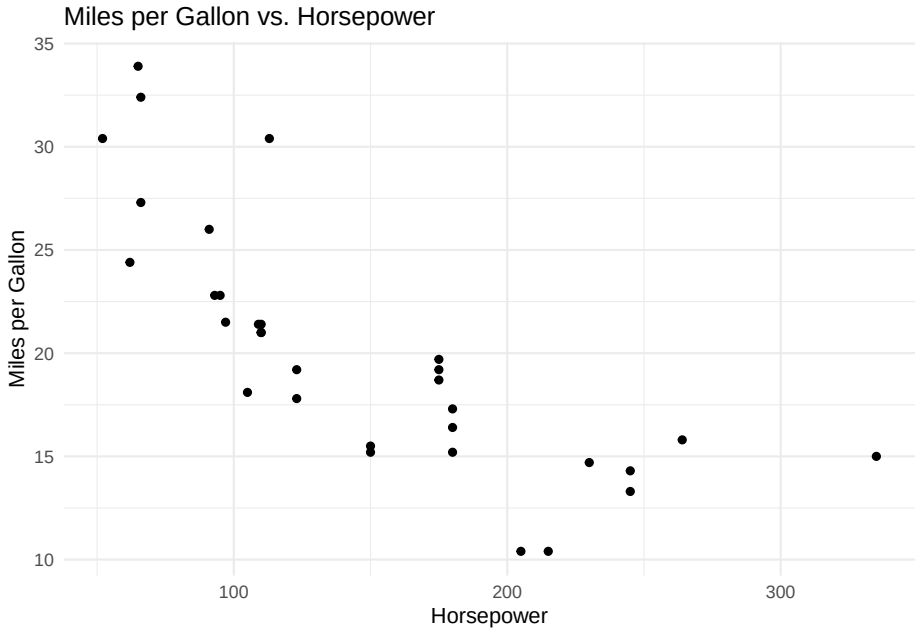
5.3.1 Load the necessary libraries and data

```
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.

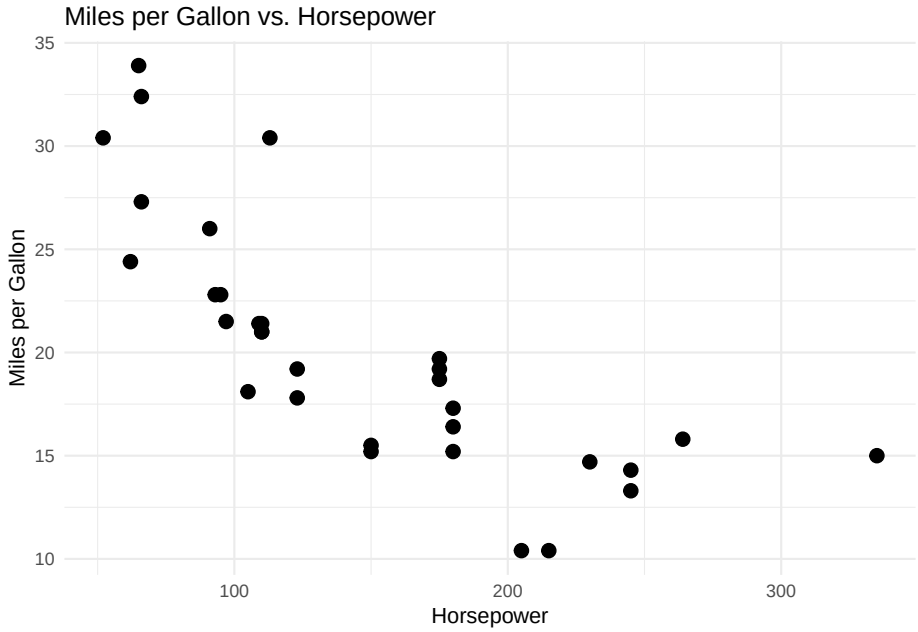
```
scatter_plot <- ggplot(mtcars, aes(x = hp, y = mpg)) +
  geom_point() +
  labs(title = "Miles per Gallon vs. Horsepower",
       x = "Horsepower",
       y = "Miles per Gallon") +
  theme_minimal()

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.

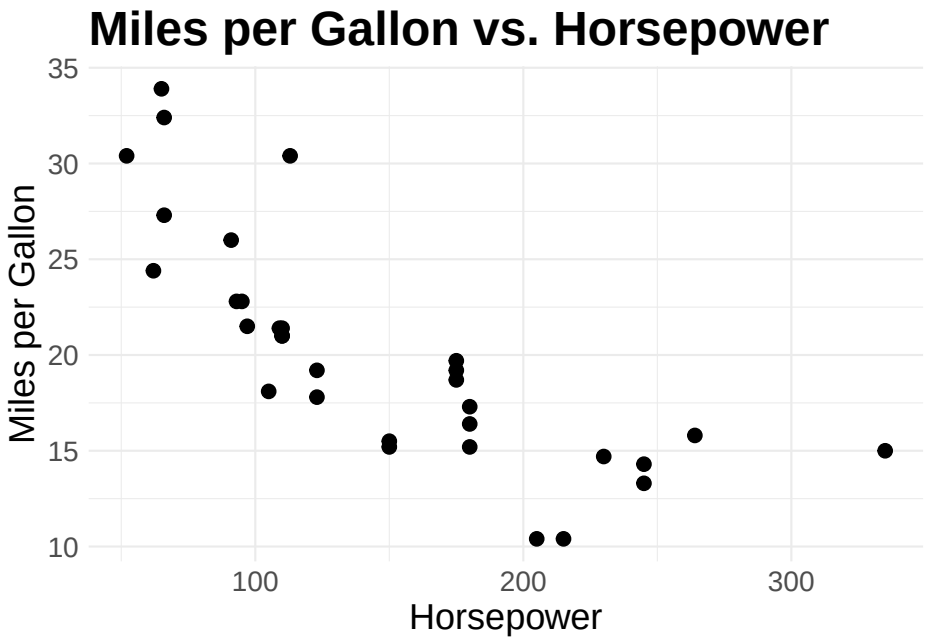
```
scatter_plot_large_points <- scatter_plot +
  geom_point(size = 3)

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.

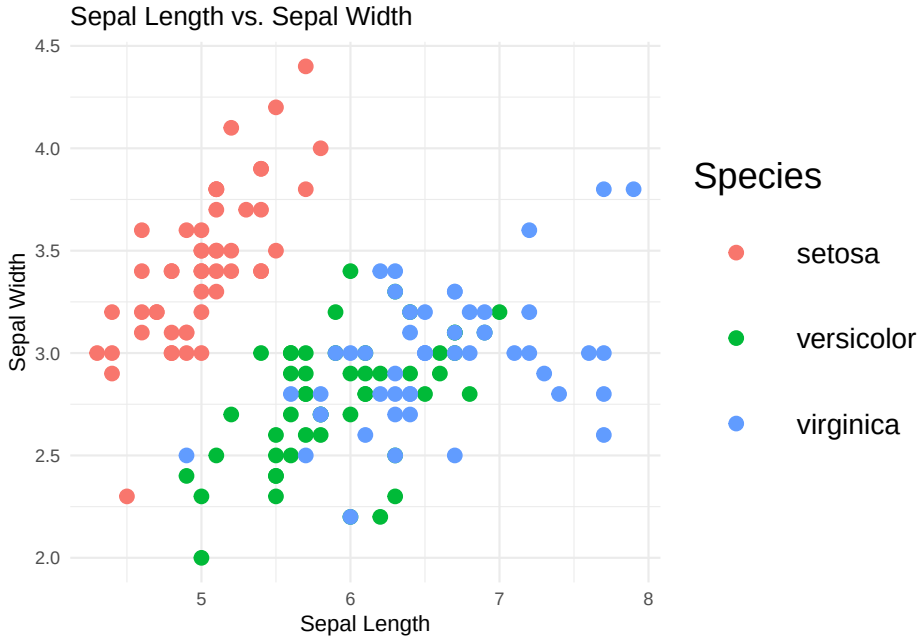
```
scatter_plot_custom_text <- scatter_plot_large_points +
  theme(plot.title = element_text(size = 24, face = "bold"),
        axis.title.x = element_text(size = 18),
        axis.title.y = element_text(size = 18),
        axis.text.x = element_text(size = 14),
        axis.text.y = element_text(size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

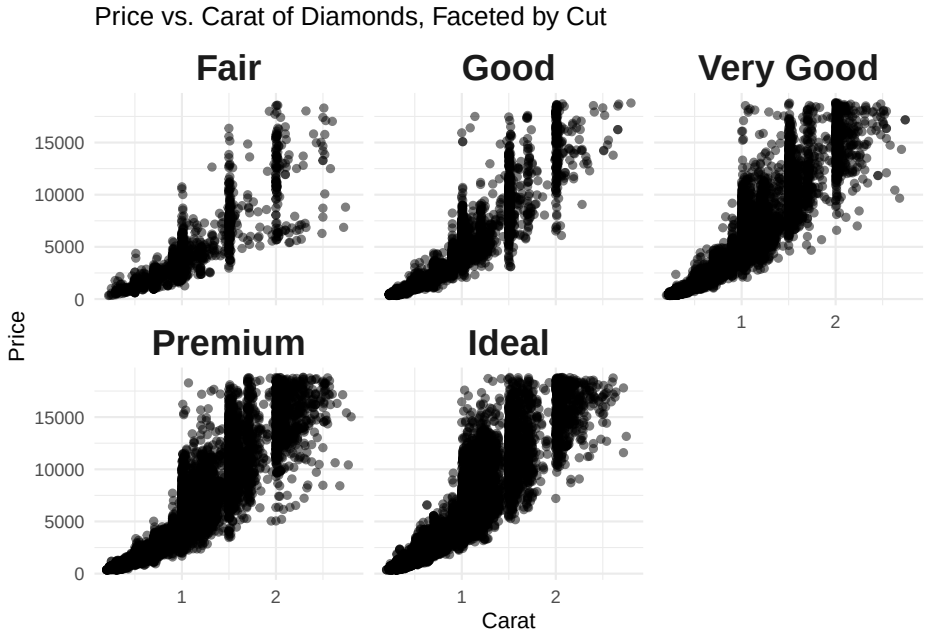
```
iris_scatter_plot <- ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width, color = Specie
  geom_point(size = 3) +
  labs(title = "Sepal Length vs. Sepal Width",
       x = "Sepal Length",
       y = "Sepal Width",
       color = "Species") +
  theme_minimal() +
  theme(legend.title = element_text(size = 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",
```

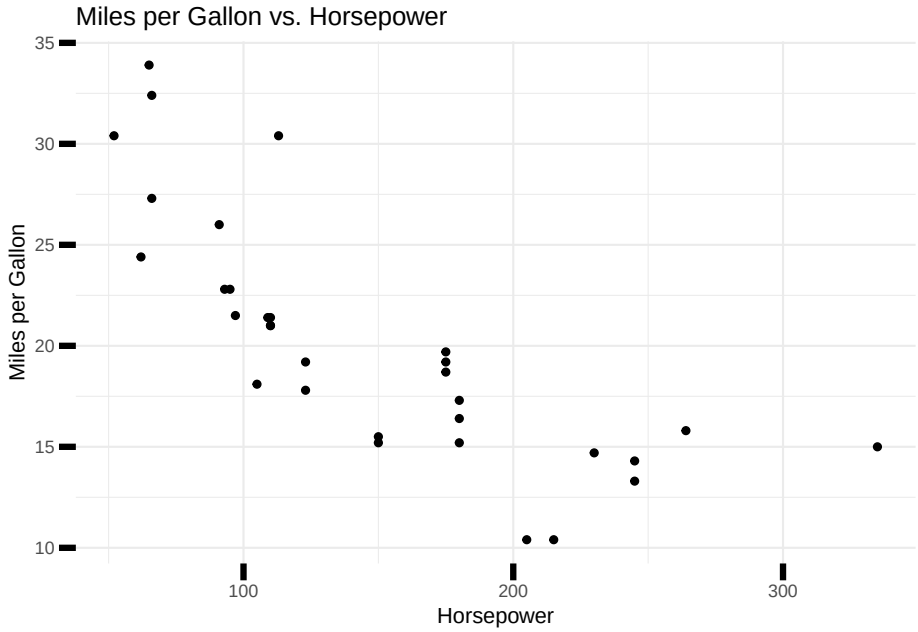
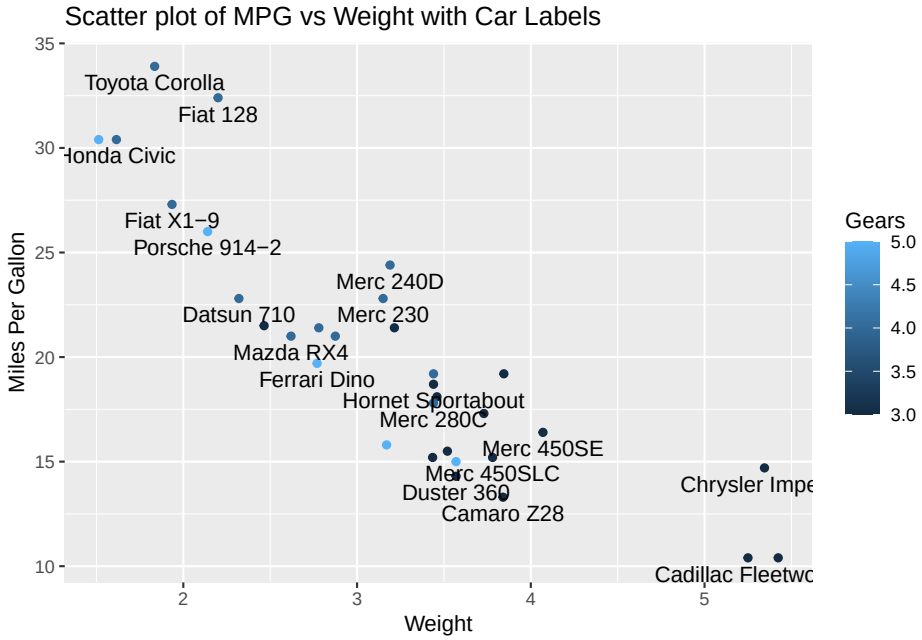


Figure 5.3: Figure 6: Scatterplot of mpg vs. hp with customized axis tick size.

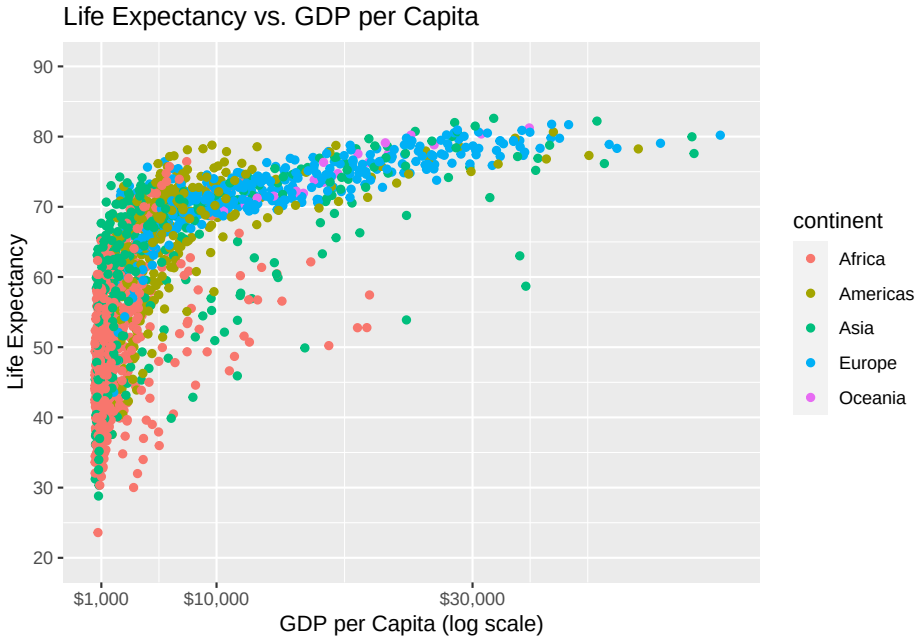
```
       y = "Miles Per Gallon",
       color = "Gears")

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")
```


Life Expectancy vs. GDP per Capita
Life Expectancy
90
80
70
60
50
40
30
20
$1,000
$10,000
$30,000
GDP per Capita (log scale)
continent
Africa
Americas
Asia
Europe
Oceania

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 c
  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 pander

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 178.91529
2   2    Hello FALSE 228.12963
3   3    Hello  TRUE 285.52978
4   4    Hello FALSE 293.59579
5   5    Hello  TRUE 150.03505
6   6  Goodbye FALSE  50.97638
7   7  Goodbye  TRUE 263.58988
8   8  Goodbye FALSE  52.73071
9   9  Goodbye  TRUE 210.22994
10 10  Goodbye FALSE  51.55478
```

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.