Analysis of STAT184 Topics

Eric Best

2023-11-29

Foreword

In this document, we will examine some code, datasets, and visualizations related to STAT184. There are three sections: a discussion on the Collaz Conjecture, an investigation on the physical properties of Diamonds, and a summary of what we have learned in STAT184 so far.

Part 1: The Collatz Conjecture

What is the Collatz Conjecture? The Collatz Conjecture was introduced by Lothar Collatz. The conjecture stipulates that these operations will eventually reduce any positive integer to 1. We can define this as a piece-wise function, with n as a positive integer:

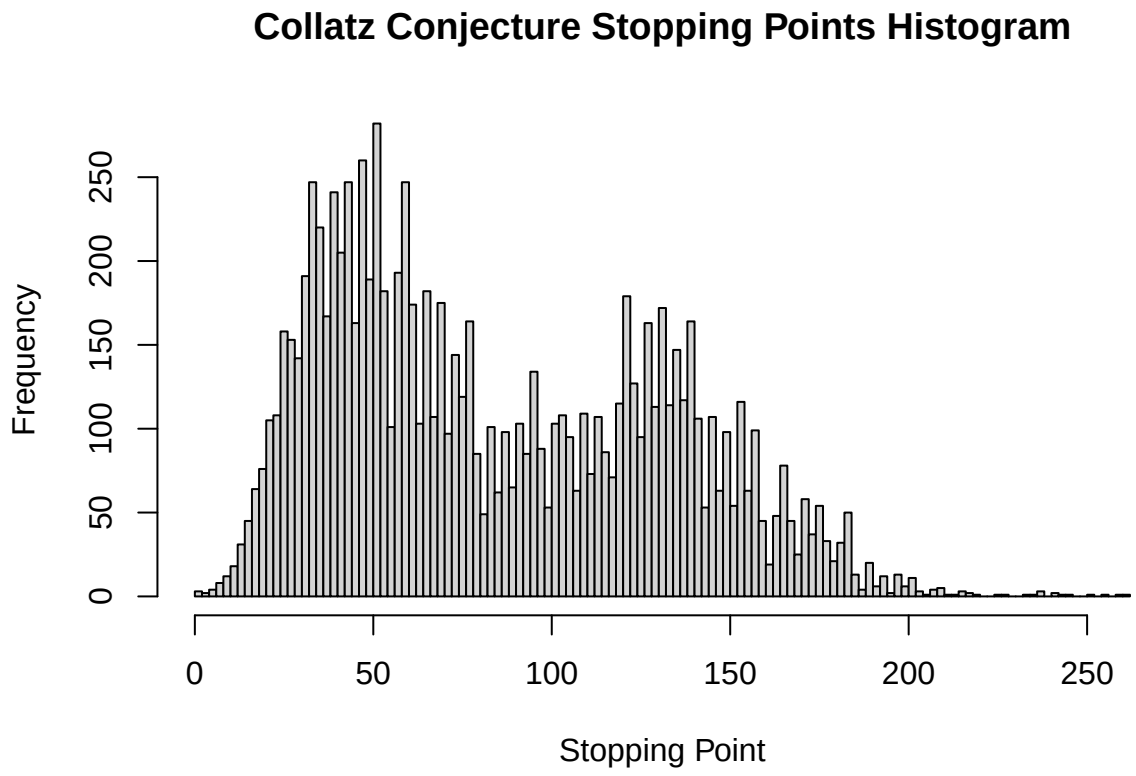
$$f(n) = \begin{cases} \frac{n}{2} & \text{if } n \text{ is even} \\ 3n+1 & \text{if } n \text{ is odd} \\ STOP & \text{if } n = 1 \end{cases}$$

Coding the Conjecture We've made R Code for a recursive implementation of the Conjecture. That is, the function will repeatedly call itself to eventually reduce n to 1. We will record the "Stopping Points", which are positive integers that state how many times the function is run to reduce n to 1.

```
## We are including this code to demonstrate how the Collatz Conjecture works.

## Function accepts a positive integer n
runCollatz <- function(integerN)
{
## Stop algorithm if argument is less than or equal to 0
  if (integerN <= 0)
  {
    stop("Must be a positive integer")
  }

## Create an integer to store Stopping Point
  integerStoppingPoint <- 0


## While n is not 1, go through the algorithm
  while (integerN != 1)
    {
## Case: n is even. Divide by two, resave into integerN
      if (integerN %% 2 == 0)
      {
        integerN <- integerN / 2
      }
```

```
## Case: n is odd. Multiply by three and add one, resave
      else
      {
        integerN <- 3 * integerN + 1
      }
## Accrue the stopping point counter
    integerStoppingPoint <- integerStoppingPoint + 1
    }
  return(integerStoppingPoint)
}
```

Histogram for Stopping Points What kind of behavior do stopping points exhibit? Let's use a histogram to see the distribution of this part of the Conjecture. We'll analyze positive integers starting with 1 and going up to 10,000, recording their respective stopping points.



Some Conclusions on the Histogram Before we discuss this histogram, it's important to note that the Collatz Conjecture remains unproven. Meaning, mathematicians haven't found an integer that is not eventually reduced to 1 by the algorithm. With that said, we can make some generalizations about stopping points, answering Dr. Hatfield's question:

- It seems that stopping points above $n = 150$ start to become less common, with stopping points above $n = 200$ being very uncommon.
- $n = 50$ is the most common stopping point. This is a right-skewed distribution, so we know that the stopping point mode should be around that value. Further, that mode should be greater than the stopping point mean and median.

- Knowing the above, we can conclude that for larger and more immense input vectors, we will get the same distribution according to the Law of Large Numbers.

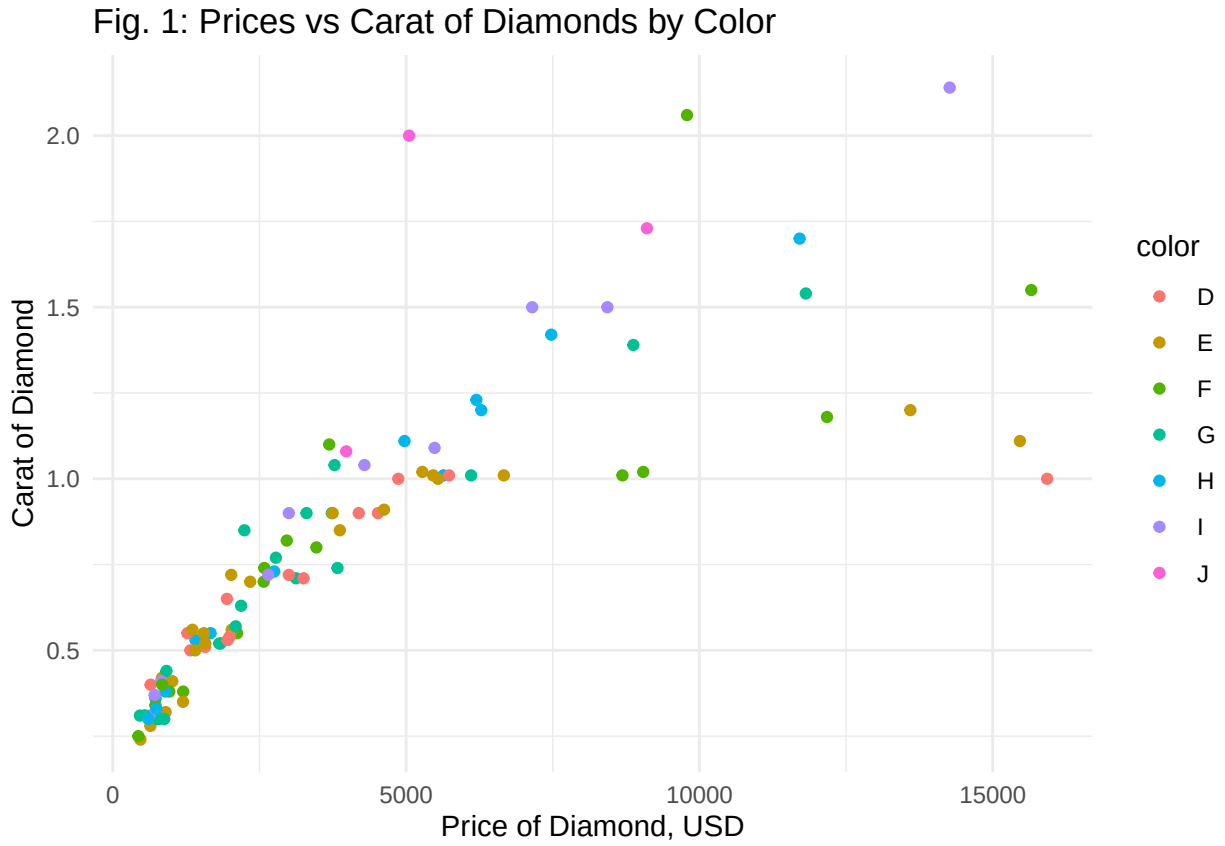
Part 2: Investigating Diamonds

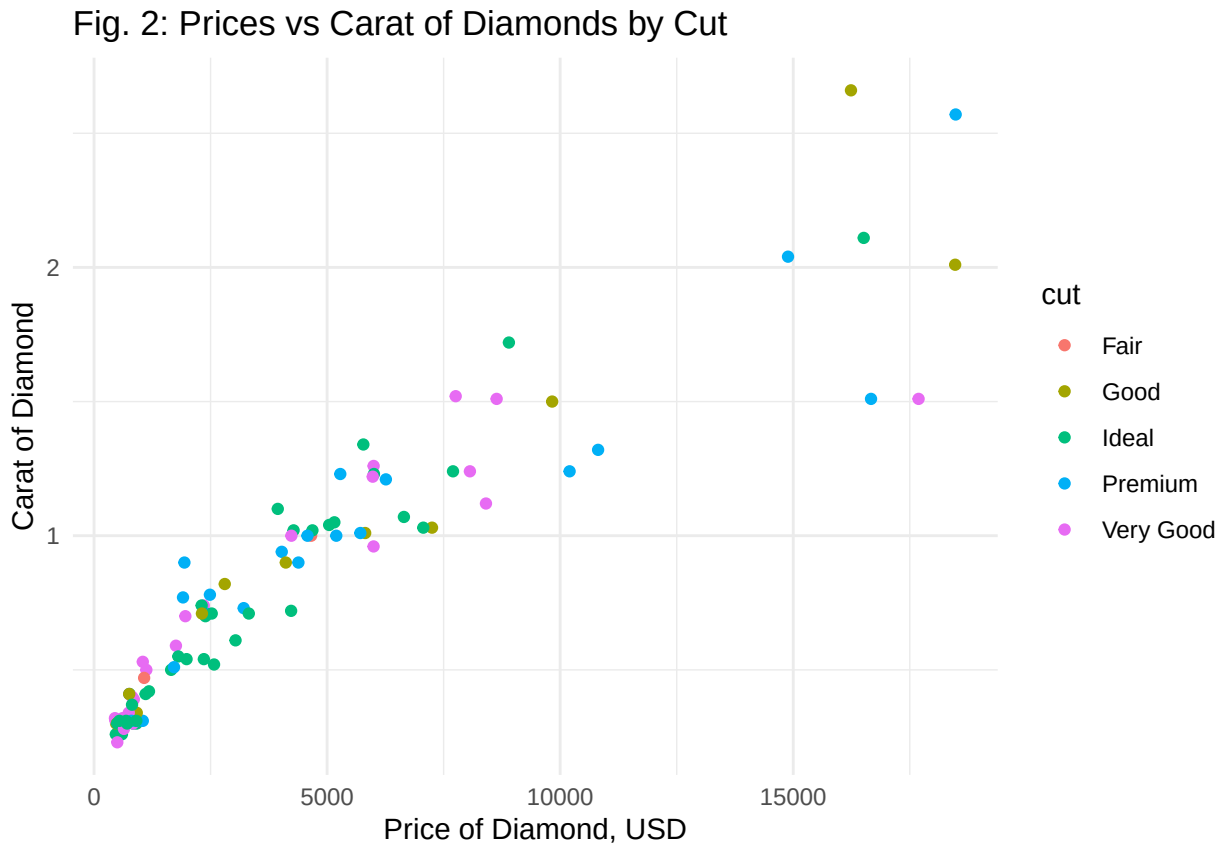
Discussion on Diamonds and their Physical Properties There is a data set included with ggplot2 and it is called Diamonds. It's a table of 53,940 diamond cuts, prices included, and contains information on various physical properties and classifications for each diamond. Here is a summary of those properties:

- **Carat**: a measurement of weight. We also have **x**, **y**, and **z** dimensions for each diamond in millimeters.
- **Cut**: the quality of the diamond's cut, ranging from Fair to Ideal.
- **Clarity**: the measurement of a diamond's purity and rarity.
- **Depth**: a percentage-based measurement from top to bottom of the stone.
- **Table**: the facet of the diamond that appears when viewed face-up.

How might these properties influence the price of a diamond? We will investigate this using data visualizations and statistical summaries.

Visualizations of the Diamonds Dataset Shown below are dot plots that include 100 cases from the Diamonds dataset, with the first (Fig. 1) concerning color and the second (Fig. 2) regarding cut:





Some Conclusions on the Diamonds Visualizations First of all, let's discuss the Color and Cut properties. The less color, the higher the value of the diamond. A colorless diamond is white in appearance, while a near-colorless diamond will have a small yellow tint. The Diamonds dataset only has diamonds with colors ranging from D to J, which is what most jewelry stores sell. The D rating is thus the most desirable for our data. For Cut, the most desirable diamonds radiate light; put simply, they sparkle the most. Therefore, Ideal is the most valuable Cut label. Here are some conclusions we can make about this dataset:

- The majority of stones will fall under $5000 USD in price and hover around and below 1.5 carats in weight. High carat stones tend to be valuable even if they are of mid-range colors. This is reflected in both figures. This might indicate that high-carat stones are more rare in nature.
- There are few expensive Fair diamonds and J Color diamonds, because they are not as desirable. There are few D Color diamonds that are high in price, perhaps due to their rarity.
- Even an Ideal or D Color Diamond can have a small price if is low in carats. It appears as if diamond buyers find Very Good to be a minimum standard for Cut quality.s
- For both figures, we can see that the distribution appears roughly quadratic in appearance. If more samples are taken, this shape becomes obvious.

We will now use a table to display summary statistics about the Diamonds dataset. This will allow us to examine many more cases compared to the above visualizations. We'll consider the x values for the diamonds in these calculations.

Looking at the Diamond Summary Table We've made a rather large table to look at each combination of Color and Cut in the dataset, with the most desirable combinations in bold. We can see immediately that most diamonds aren't below 3.73 mm for their x-value, and the maximums for each category fall at around

Table 1: Fig. 3: Statistical Facts for Diamond Length Values, in Millimeters

Cut	Color	Minimum	1st Quartile	2nd Quartile	3rd Quartile	Maximum	Median	Mean	Standard Deviation	Count
Fair	**D**	**4.09**	**5.59**	**6.08**	**6.30**	**9.42**	**6.08**	**6.02**	**0.83**	**163**
Fair	E	3.87	5.28	6.07	6.35	8.16	6.07	5.91	0.83	224
Fair	F	4.19	5.45	6.06	6.37	8.58	6.06	5.99	0.88	312
Fair	G	3.87	5.59	6.13	6.52	8.75	6.13	6.19	0.93	313
Fair	H	4.40	6.01	6.33	7.16	10.00	6.33	6.58	0.94	303
Fair	I	4.62	5.92	6.34	7.14	9.11	6.34	6.56	0.90	175
Fair	J	4.24	6.06	6.49	7.38	10.74	6.49	6.75	1.09	119
Good	**D**	**3.83**	**4.78**	**5.69**	**6.31**	**8.15**	**5.69**	**5.62**	**0.93**	**662**
Good	E	3.83	4.78	5.68	6.31	9.08	5.68	5.62	0.95	933
Good	F	3.83	5.02	5.76	6.32	8.69	5.76	5.71	0.92	907
Good	G	3.94	5.05	6.05	6.38	8.90	6.05	5.85	1.02	871
Good	H	4.04	5.07	6.11	6.62	9.44	6.11	5.97	1.13	702
Good	I	4.19	5.56	6.28	7.21	9.38	6.28	6.25	1.22	522
Good	J	4.22	5.65	6.44	7.23	9.32	6.44	6.38	1.12	307
Ideal	**D**	**3.81**	**4.47**	**5.11**	**5.72**	**9.04**	**5.11**	**5.19**	**0.82**	**2834**
Ideal	E	3.76	4.47	5.10	5.76	8.52	5.10	5.22	0.85	3903
Ideal	F	3.90	4.53	5.23	6.18	8.67	5.23	5.41	0.98	3825
Ideal	G	3.92	4.52	5.24	6.49	8.75	5.24	5.51	1.04	4883
Ideal	H	3.94	4.57	5.68	6.68	9.65	5.68	5.73	1.17	3115
Ideal	I	3.94	4.77	5.84	6.87	9.49	5.84	5.98	1.24	2093
Ideal	J	3.93	5.23	6.50	7.20	9.25	6.50	6.32	1.22	896
Premium	**D**	**3.73**	**4.71**	**5.38**	**6.44**	**8.99**	**5.38**	**5.60**	**1.02**	**1602**
Premium	E	3.79	4.70	5.41	6.41	9.26	5.41	5.59	1.03	2337
Premium	F	3.73	4.87	5.89	6.55	9.24	5.89	5.88	1.02	2331
Premium	G	3.95	4.75	5.90	6.71	9.44	5.90	5.86	1.15	2924
Premium	H	3.96	5.17	6.48	7.04	9.44	6.48	6.25	1.22	2359
Premium	I	3.97	5.40	6.74	7.46	10.14	6.74	6.49	1.32	1428
Premium	J	4.22	6.04	6.92	7.62	10.02	6.92	6.81	1.22	808
Very Good	**D**	**3.86**	**4.69**	**5.46**	**6.30**	**9.08**	**5.46**	**5.50**	**0.97**	**1513**
Very Good	E	3.74	4.60	5.33	6.27	8.65	5.33	5.43	1.00	2400
Very Good	F	3.84	4.73	5.69	6.38	8.64	5.69	5.61	1.00	2164
Very Good	G	3.88	4.69	5.68	6.44	8.65	5.68	5.66	1.04	2299
Very Good	H	3.89	4.96	6.13	6.76	9.23	6.13	5.99	1.15	1823
Very Good	I	3.95	5.58	6.35	7.19	10.01	6.35	6.27	1.17	1204
Very Good	J	3.94	5.67	6.56	7.29	8.89	6.56	6.46	1.14	678

Note:
GGplot2 Diamonds Dataset had cases with non-positive Lengths. Above data excludes those.
[1] Data displayed above rounded to two decimal places.

$x = 8.0$. Indeed, the maximum of all categories is $x = 10.02$. This could mean that there is some sort of physical bounds for the sizes of diamonds found in nature.

Considering the Plots and Table Together Before, we said that certain combinations of Cut and Color might make a diamond more valuable. First, we can say the dataset has far fewer Fair and Good Diamonds compared to other categories. We might conclude that there is little consumer demand for these stones compared to the supply. It just seems that consumers prefer higher-quality cuts. The standard deviations are not especially high for all these label combinations, which supports the "clustering" of stones observed with higher random samples.

Part 3: Reflecting on STAT184 Thus Far

I'm pleased with this class so far and highly value my time spent learning R/RStudio and coding and design techniques.

Learning R and RStudio With regards to learning R, I had heard of the language before, but I thought it was similar to MATLAB or Maple. Prior to this course, I've taken classes in C++ and Java and learned basic C on my own. It turns out that R has a long history of improvements and feature additions over the years, similar to the development of C in the 1970s and 1980s. This makes R a very robust and adaptable language on its own, but there are many projects being worked on for R that expand its usefulness, as discussed in Activity 01.

RStudio is significantly easier to use compared to Eclipse or Visual Studio. I have never used an Integrated Development Environment (IDE) that was so helpful with debugging and correcting code. The included documentation is good, although some of the examples the authors use are rather simple. I have to admit that some of the operations in R were confusing at first, like using the piping operator `%>%`. I generalized the concept of piping as just passing the argument of one function to another. Once I understood this operation, it became a lot easier to read and interpret R code. We also used a textbook called 'Data Computing.' I found this textbook to be dry, but at least it's open source.

General Design Practices Tidy data is the first important design principle of this class, which means reducing a dataset to a table form that follows these three rules: every column is a variable, every row is an observation, and all values must have their own cell. Not every dataframe comes in a form that is easily digestible for both a person reading the data and the RStudio environment. These concepts were introduced in Activity #04, and we mastered them in Activity #08 by tidying the messy Military Marital Data table.

A related but equally important concept is Data Wrangling, which we developed in Activity #06 and Activity #08 while finding pertinent Diamonds data. Wrangling means finding and obtaining the data we need to achieve a goal. Now, we know how to create a solid foundation of data for creating good tables and visualizations.

Regarding actual graphic and element design, we have read excerpts from Tufte and Kosslyn. To put it simply, they have provided us with advice on creating powerful yet approachable graphics and tables. I've learned how to create simple visualizations that still convey a lot of information. I can now sometimes identify where R and R packages are used in making visualizations for everyday media. The PCIP System can also be utilized for creating visualizations and tables.

One of the most interesting topics discussed was data-scraping. We have learned to extract data from webpages that can be tidied and wrangled into appreciable datasets. Immediately I could see applications to Ethical Hacking and Penetration Testing using other scraping tools developed for R.

Coding Practices I learned a lot about coding guidelines and practices in this course. Activity #02 helped me become much more specific with nouns and verbs for R in order to achieve a specific goal. I've sometimes had issues deciding exactly what I want to code, and the PCIP System helps with this. Using this framework, I have a good set of guidelines for starting, improving, and finalizing a code project.

Activity #05 was the most important for learning to code visualizations, specifically using ggplot. I enjoyed investigating additional ways to make these visualizations more engaging, like adding footnotes. Activity #08 was important for learning to code a data wrangling operation. One of my favorite parts of that activity was hand-coding dataset-specific wrangling instructions. It immediately brought to mind the idea of creating generalized data wrangling operations that will work on any dataset. Perhaps that objective has already been fulfilled by independent R package developers.

Tying it All Together In the past few weeks of class, we've examined R Markdown. RMD allows us to create reports and presentations that include code, text, mathematical symbols, tables, and visualizations. I can see how this has become the standard for creating scholarly documents. It is a very clean system that helps us organize our work much better and expedites the creation of document files. I appreciate that code chunks can have localized properties, such as specifying a table name or instructing R not to print error and warning messages. Suffice to say, I will be using RMD for report management over word processors.

Code Appendix

We have reproduced our code here as a reference.

```
## This chunk's code was included to provide portability to different systems.
## We are using groundhog to load packages from a specific date.
## We are then using 'here' to load the ggplot2 dataset called diamonds.csv
## Diamonds was obtained from: https://github.com/tidyverse/ggplot2/blob/main/data-raw/diamonds.csv

library(groundhog)
groundhog.day="2023-11-20"
pkgs=c('janitor','dplyr','kableExtra', 'ggplot2', 'here')
groundhog.library(pkgs, groundhog.day)
knitr::opts_chunk$set(warning = FALSE, message = FALSE)
##tinytex::install_tinytex(force = TRUE)
csv_path <- here("diamonds.csv")
diamonds <- read.csv(csv_path)
##devtools::install_github("kupietz/kableExtra")

## We are including this code to demonstrate how the Collatz Conjecture works.

## Function accepts a positive integer n
runCollatz <- function(integerN)
{
## Stop algorithm if argument is less than or equal to 0
  if (integerN <= 0)
  {
    stop("Must be a positive integer")
  }

## Create an integer to store Stopping Point
  integerStoppingPoint <- 0


## While n is not 1, go through the algorithm
  while (integerN != 1)
    {
## Case: n is even. Divide by two, resave into integerN
      if (integerN %% 2 == 0)
      {
        integerN <- integerN / 2
```

```
      }

## Case: n is odd. Multiply by three and add one, resave
      else
      {
        integerN <- 3 * integerN + 1
      }
## Accrue the stopping point counter
    integerStoppingPoint <- integerStoppingPoint + 1
    }
  return(integerStoppingPoint)
}
## This code is included to show a histogram of the Collatz Stopping Points, in order to answer Dr. Hati

intervalValues <- 1:10000
##Using sapply to call runCollatz on our desired list
collatzVector <- sapply(intervalValues, runCollatz)

hist(collatzVector, breaks = 100, main = "Collatz Conjecture Stopping Points Histogram", xlab = "Stoppi
```

```
## This code was included to show a dot plot of 100 diamonds and the relationship between prices, carat

library(ggplot2)
diamondsSampleDot <- diamonds[sample(1:nrow(diamonds), 100, replace=FALSE),]

## Displaying dot plots to investigate price, carat, and diamond physical properties
ggplot(data = diamondsSampleDot, mapping = aes(x = price, y = carat, colour = color) ) +
geom_point(size = 1.5) +
labs(x = "Price of Diamond, USD", y = "Carat of Diamond", title = "Fig. 1: Prices vs Carat of Diamonds b
theme_minimal()
```

```
## This code was included to show a dot plot of 100 diamonds and the relationship between prices, carat

library(ggplot2)
diamondsSampleDot <- diamonds[sample(1:nrow(diamonds), 100, replace=FALSE),]

## Displaying dot plots to investigate price, carat, and diamond physical properties
ggplot(data = diamondsSampleDot, mapping = aes(x = price, y = carat, colour = cut) ) +
geom_point(size = 1.5) +
labs(x = "Price of Diamond, USD", y = "Carat of Diamond", title = "Fig. 2: Prices vs Carat of Diamonds b
theme_minimal()
```

```
## This code was included to tidy up and display a table regarding the diamonds' physical properties

###################
##Data Wrangling##
###################

dimensionStats <- diamonds %>%
  filter(x > 0) %>%                                    ## Get the positive nonzero x-value
  group_by(cut, color) %>%                                   ## Group by diamond cut
  select(cut, color, x) %>%                                  ## Select for cut and x-value
  summarize(                                           ## Use summarize and across like in cl
    across(
```

```
      .cols = where(is.numeric),
      .fns = list(
        minimum = ~min(.x, na.rm = TRUE),                          ## Specified statistics
        quartile1 = ~quantile(.x, probs = 0.25, na.rm = TRUE),
        quartile2 = ~quantile(.x, probs = 0.50, na.rm = TRUE),
        quartile3 = ~quantile(.x, probs = 0.75, na.rm = TRUE),
        max = ~max(.x, na.rm = TRUE),
        median = ~median(.x, na.rm = TRUE),
        meanArithmetic = ~mean(.x, na.rm = TRUE),
        meanStanDev = ~sd(.x, na.rm = TRUE)
      )
    ),
    count = n()                                                    ## Display the count for each Cut clas
  )
#Diamonds Dataset Lengths contains some values equal to zero. These were not included in the data shown
##############
##Polishing##
##############

## Clean up the column names because they look rough otherwise
colnames(dimensionStats) <- c("Cut", "Color", "Minimum", "1st Quartile", "2nd Quartile", "3rd Quartile"
## Round the table with mutate, across and where
dimensionStatsRounded <- dimensionStats %>%
  mutate(across(where(is.numeric), ~round(., 2)))

## Making the table look good
dimensionStatsFormatted <- dimensionStatsRounded %>%
  kable(
    caption = "Fig. 3: Statistical Facts for Diamond Length Values, in Millimeters",
    booktabs = TRUE,
    align = c("l", rep("c", 6)),
  ) %>%
  kableExtra::kable_styling(
    full_width = F,
    bootstrap_options = c("striped", "condensed", "responsive"),
    font_size = 16,
    latex_options = "scale_down",
    position = "center",

  )%>% ## Coloring the rows and font
  kableExtra::row_spec(0, bold = TRUE, color = "black") %>%
  ## Highlighting the most desirable combinations of Color and Cut
  kableExtra::row_spec(1, bold = TRUE, color = "black") %>%
  kableExtra::row_spec(8, bold = TRUE, color = "black") %>%
  kableExtra::row_spec(15, bold = TRUE, color = "black") %>%
  kableExtra::row_spec(22, bold = TRUE, color = "black") %>%
  kableExtra::row_spec(29, bold = TRUE, color = "black")


## Adding footnotes
dimensionStatsFormatted <- dimensionStatsFormatted %>%
  footnote(
    c("GGplot2 Diamonds Dataset had cases with non-positive Lengths. Above data excludes those."),
```

```
    c("Data displayed above rounded to two decimal places."),
  )
dimensionStatsFormatted
```