Matteo's First RMD File !

Matteo Martone

2022-10-28

Collatz Conjecture

The collatz conjecture states that if you pick any number, it transform into one by a set of two arithmetic operations. If the number selected is odd, it will be multiplied by three, and then added by one. If the number selected is even, it will be divided by two. Once the number reaches one, the function stops.

What we want to know is a bit different. We want to solve for the amount of iterations the collatz conjecture processes until the number has reached one. For example, if we had started with five we would multiply by three and add one to get sixteen. Then we would divide by two to get eight, then again to get four, again to get two, and finally one. This had five total iterations. We can call this "stopping times".

```
# create a function "collatzcounter" that counts the amount of times the function takes to reach 0
collatzcounter <- function(n, w = 0){
  # if n = 1 then stop
  if (n == 1) {
    return (w)
  # if n is even then divide by 2
  } else if (n %% 2 == 0) {
    collatzcounter (n = n/2, w = w + 1)
  # if n is odd then multiple by 3 and add 1
  } else {
    collatzcounter (n = 3 * n + 1, w = w + 1)
  }

}
#invoke the function and select number you want to find numbers of times function executes to get to 1
collatzcounter(100)
```

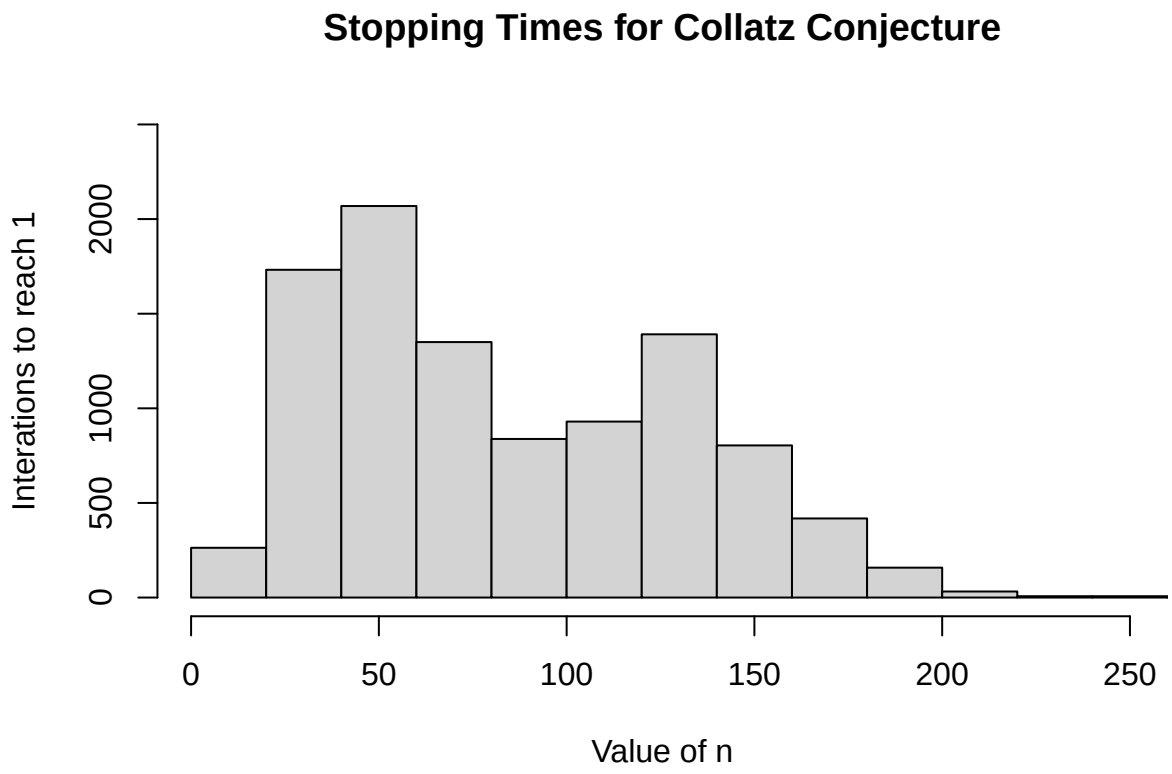
```
## [1] 25
```

```
# give example of n when the value is 100 - returns 25 stopping times til 1 is reached
```

From this code in R we can see the amount of times a number takes to reach one using the collatz conjecture by placing any number in place of "n". To see this more visually we can create a histogram that displays all different values of n and how many iterations it will take from zero.

```
# create a vectorize form of collatzcounter using 'n'
collatzcounterVec <- Vectorize(
  FUN = collatzcounter,
  vectorize.args = 'n'
)
```

```r
# create of histogram of new vectorized form
# set n from 1 to 10,000 based off the stoppng time
hist(x = collatzcounterVec(n = 1:10000),
     # gives heading/title
    main = "Stopping Times for Collatz Conjecture",
    # defines x label
    xlab = "Value of n",
    # defines y label
    ylab = "Interations to reach 1",
    # setting of x to 250 and y to 2500
    xlim = c(1,250),
    ylim = c(1,2500),
)
```



What can we learn from this data visualization?

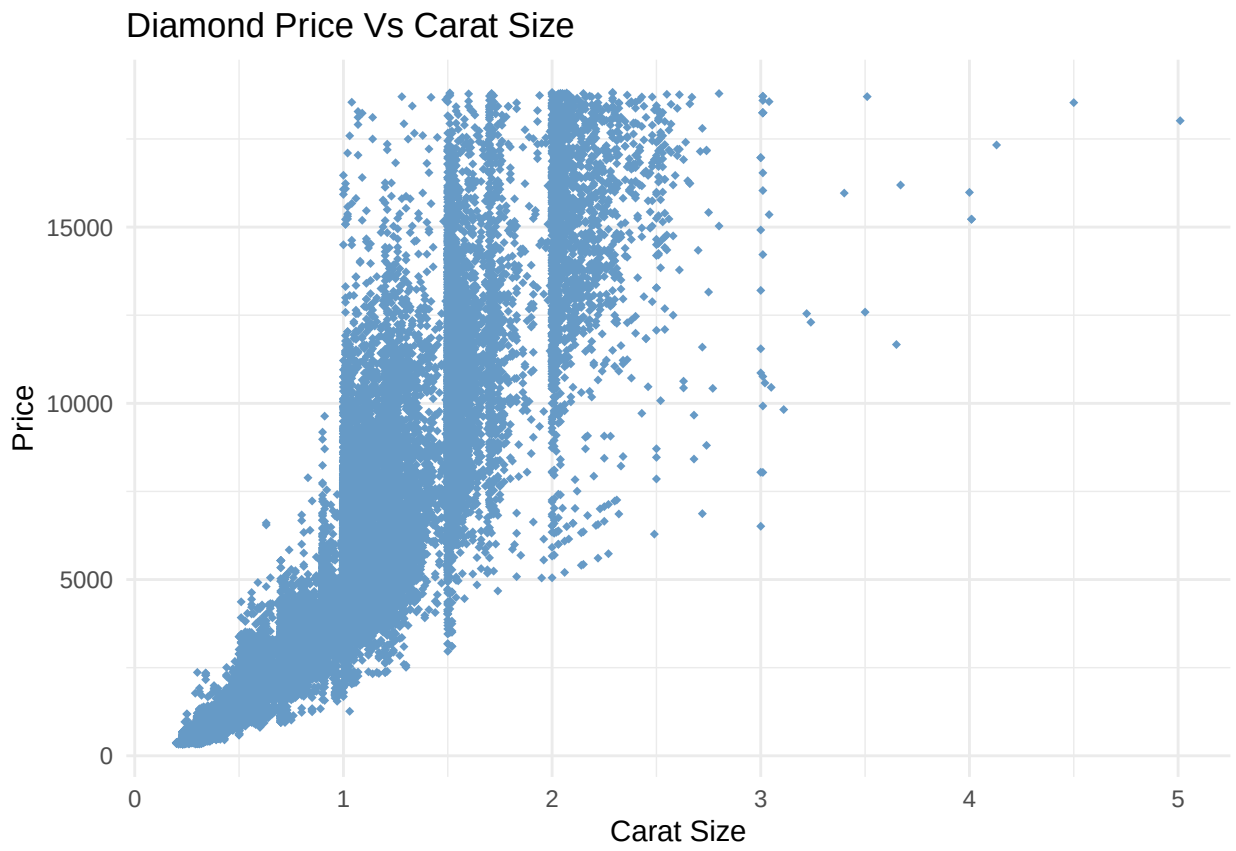
From this data visualization we can learn many things about the value of n and the number of iterations to reach one required. We can see that there is general downward trend of the total number of iterations required to reach one as the value of n increases. We can see that values of n ranging from zero to fifty are the largest, and anything after 200 is quite minimal.

Diamonds

The Diamonds data set is one in which that has many different attributes of a diamond like price, carat size, cut type, clarity and many other sizing dimensions. From this data set there are many comparisons we can draw up. Something interesting we can compare is the size of diamond (measured in carats), and the price of diamond (dollars).

```
#load packages and data
library(ggplot2)
data(diamonds)

# use ggplot package to create a data visualization
ggplot(diamonds) +
  aes(x = carat, y = price) +
  geom_point(
    # define shape and size of points
    shape = "diamond",
    #shrink size
    size = 1.25,
    #change color to diamond like
    colour = "#669AC6"
  ) +
  labs(
    # add labels and title
    x = "Carat Size",
    y = "Price",
    title = "Diamond Price Vs Carat Size"
  ) +
  theme_minimal()
```



What we can learn from this data visualization?

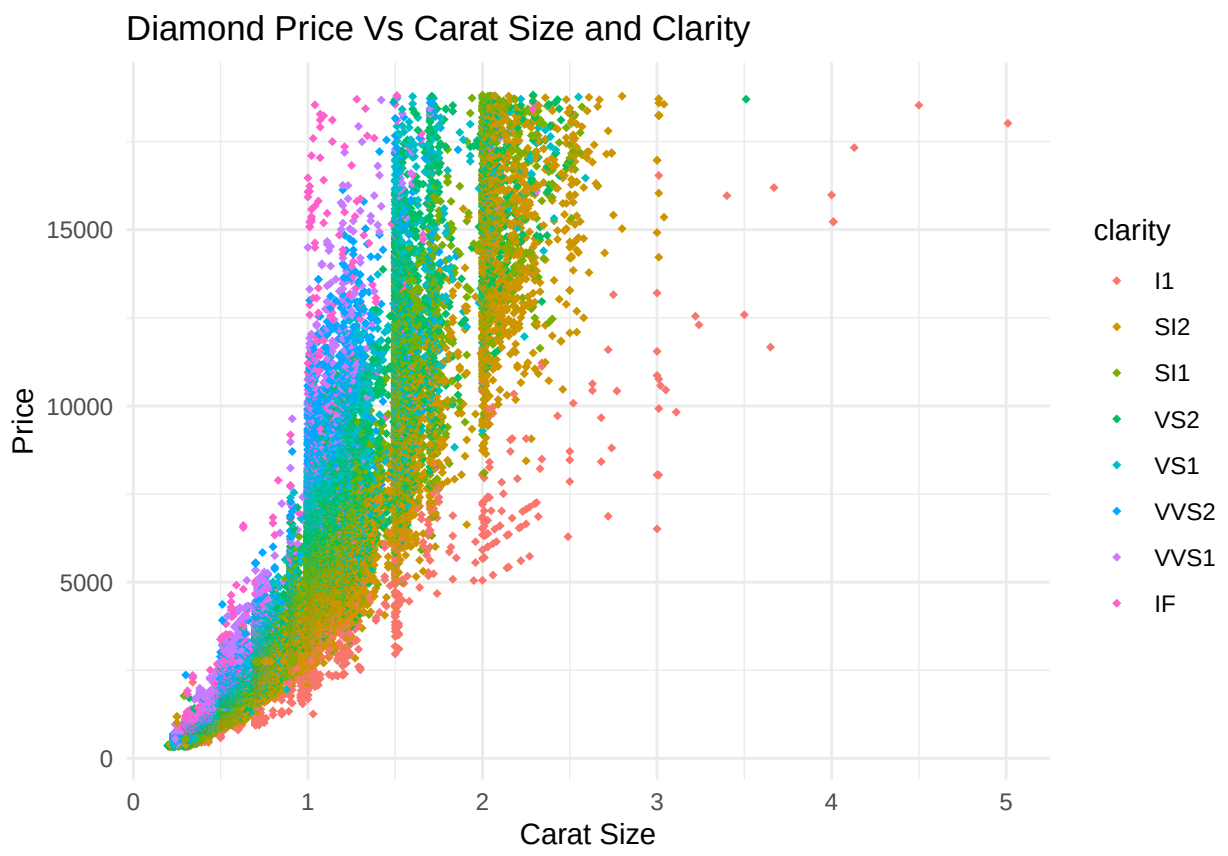
From this data visualization we can immediately see a positive correlation between carat size and the price of a diamond. We know this because as the x-value is increasing, the y-value is as well. We can infer that as carat size increases, price generally does as well. However, there are many other attributes of a diamond that can have a direct correlation to the price. The next we will be adding is the clarity of the diamond.

Diamonds Data Visualization 2

```
library(ggplot2)
data(diamonds)

# add new variable 'clarity' which is defined by the color and legend
ggplot(diamonds) +
  aes(x = carat, y = price, color = clarity) +
  geom_point(
    shape = "diamond",
    size = 1.25,
  ) +
  # add color scaling for legend
  scale_color_hue(direction = 1) +
  labs(
    x = "Carat Size",
```

```
    y = "Price",
    # add new title reflecting variables
    title = "Diamond Price Vs Carat Size and Clarity"
  ) +
  theme_minimal()
```



In the second diamonds data visualization we have a lot more to look at. From the original we have the comparison of the carat size to price. The next attribute we need to look at is located at the key on right side labeled "clarity". Each level of clarity from I1 to IF (included to internally flawless) is labeled by a color. When looking at the new data visualization we can see price is heavily influence by clarity. In fact, there is a direct correlation. When the level of clarity increases, so does price. IF diamonds are always at the top of the pricing.

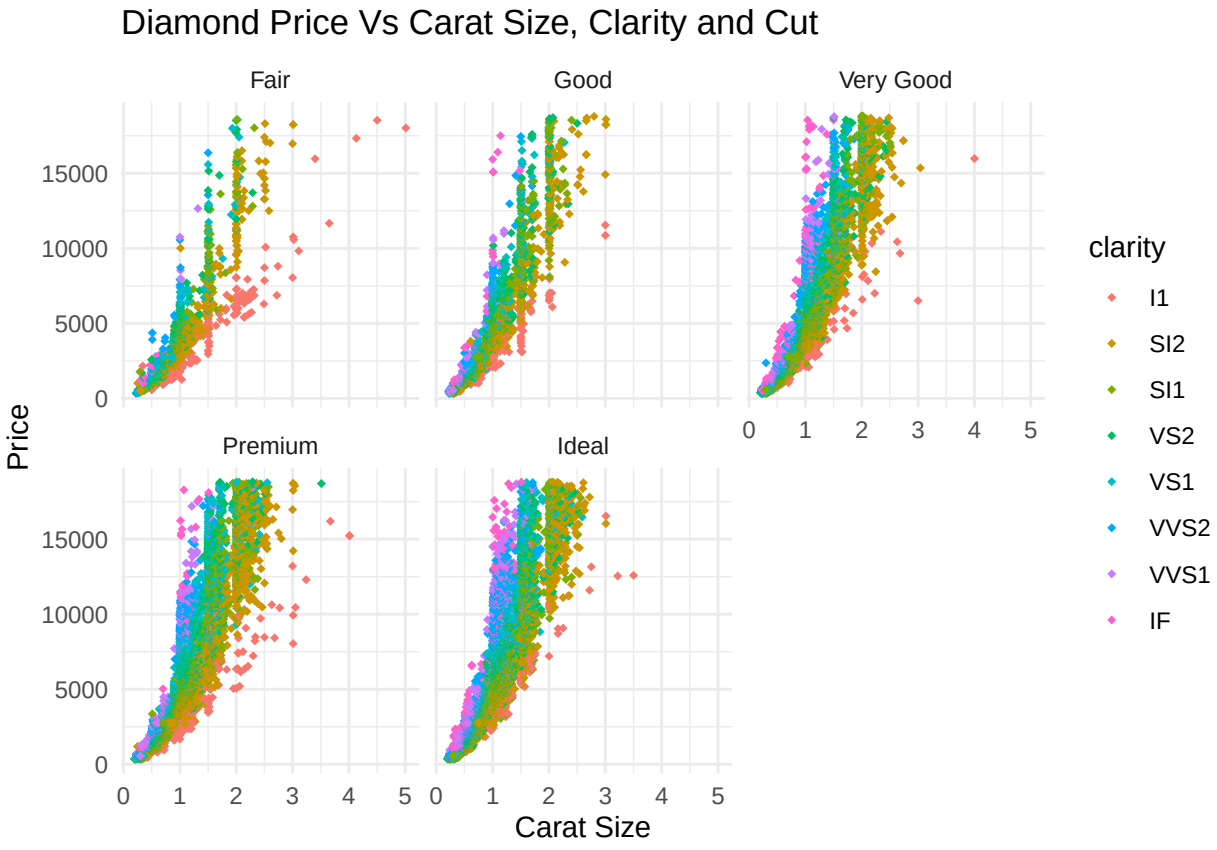
Something we can also notice is that once carat size goes beyond 2, there seems to be a drastic decrease in IF and VVS (very very slightly included) diamonds. This most likely has to do with the rarity. This prove trues for other clarity as well, as carat size increases, quality of clarity decreases in quantity.

While these two are both great comparisons to diamond price, there is another I was to draw attention to, the cut of the diamond.

Diamonds data visualization 3

```
#load packages and data
library(ggplot2)
data(diamonds)
```

```
# add new variable 'clarity' which is defined by the color and legend
ggplot(diamonds) +
  aes(x = carat, y = price, color = clarity) +
  geom_point(
    shape = "diamond",
    size = 1.25,
  ) +
  # add color scaling for legend
  scale_color_hue(direction = 1) +
  labs(
    x = "Carat Size",
    y = "Price",
    # add new title reflecting variables
    title = "Diamond Price Vs Carat Size, Clarity and Cut "
  ) +
  theme_minimal() +
  # add facet -- new variable cut to compare each
  facet_wrap(vars(cut))
```



From this we can see the original two data visualizations. What we have added this time is a facet to draw a greater comparison on the influence of price. This facet creates five different visualizations with each having respect to their own cuts (fair, good, very good, premium, and ideal). What we can see immediately is that the cut of the diamond also reflects to price. As cut quality increase, so does price.

We know this because of the density of the scatter plots. In the "fair" cut scatter plot, the correlation is positive but not super strong. While in the "ideal" cut scatter plot, the correlation is extremely positive and

almost vertical. A diamond of "ideal" cut has much more worth than a "fair" cut.

Diamonds Summary

Below is a summary table of the diamonds data set with respect to the cut and width of the diamond.

```
#load packages and data
library(janitor)
```

```
##
## Attaching package: 'janitor'

## The following objects are masked from 'package:stats':
##
##     chisq.test, fisher.test
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(kableExtra)
```

```
## Warning in !is.null(rmarkdown::metadata$output) && rmarkdown::metadata$output
## %in% : 'length(x) = 3 > 1' in coercion to 'logical(1)'

##
## Attaching package: 'kableExtra'

## The following object is masked from 'package:dplyr':
##
##     group_rows
```

```
data(diamonds)

diamondsummary_y <- diamonds %>%
  #sorts all by the cut
  group_by(cut) %>%
  # select y which is the width
  select(cut, y ) %>%
  # summarize function for wanted info below
```

Table 1: Diamonds Cut Summary For width

Cut	min	Q1	Q2	Median	Q3	Q4	Max	Arithmetic Mean	Arithr
Fair	0	5.50	5.98	6.10	6.23	7.00	10.54	6.18	
Good	0	4.74	5.63	5.99	6.23	6.60	9.38	5.85	
Very Good	0	4.62	5.40	5.77	6.16	6.68	9.94	5.77	
Premium	0	4.66	5.62	6.06	6.40	6.94	58.90	5.94	
Ideal	0	4.46	4.96	5.26	5.72	6.57	31.80	5.52	

```
  summarize(
    across(
      .cols = where(is.numeric),
      .fns = list(
        min = ~min(.x, na.rm = TRUE),
        # even though this says quantile, they are quintile broken up by there prob - .2.4.6.8
        Q1 = ~quantile(.x, probs = 0.2, na.rm = TRUE),
        Q2 = ~quantile(.x, probs = 0.4, na.rm = TRUE),
        median = ~median(.x, na.rm = TRUE),
        Q3 = ~quantile(.x, probs = 0.6, na.rm = TRUE),
        Q4 = ~quantile(.x, probs = 0.8, na.rm = TRUE),
        max = ~max(.x, na.rm = TRUE),
        mean = ~mean(.x, na.rm = TRUE),
        sasd = ~sd(.x, na.rm = TRUE)
      )
    ),
    # this adds commas inbetween large numbers like 1000 -> 1,000
    count = format(n(), big.mark = ','),
  )
# gives columns names
colnames(diamondsummary_y) <- c('Cut','min','Q1','Q2','Median','Q3','Q4','Max','Arithmetic Mean','Arithr

#polish using kable
diamondsummary_y %>%
  kable(
    caption = "Diamonds Cut Summary For width",
    booktabs = TRUE,
    align = c("l", rep("c", 6)),
    digits = 2
  ) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("striped", "condensed"),
    font_size = 16
  )
```

From this table, we can immediately see it is about a diamond cut summary for the width. We know this because of the title. We can also see it is based on the cut of the diamond and the summary with respect to it. There are five types of cuts given in rows, and ten summary statistics are given by the column. You can easily read the min through the max broken up by quintiles and then the mean, standard deviation, and

count. From this, you can compare each type of cut and whichever statistic you want.

Something I noticed was that the premium and ideal cut max widths of 58.9 and 31.8, were much larger than the other cuts all falling around nine through eleven. This caught my eye because they all have relatively similar means and medians.

Reflections

I have learned many, many things in this course. This course was actually the first time I used R, and the first time I have ever coded. Everything I have learned has been from scratch. I learned about the PCIP. I learned about base r and how to write code. I learned about data visualizations and how they can enhance a data set. I learned about packages in r and how to use them. I learned how to make an RMD file like this one.

There are truthfully an infinite amount of things I have learned because of this class but what I feel that I have learned the most in this course is how to create a data visualization.

Something I aspire to do in the future is sports analytics. A very large portion of that is finding a data set, wrangling through the data, then create an efficient data visualization to greater understand the data set. I have seen this be used for almost every sport.

I was always beyond interested, but know that I know to do it and really do enjoy doing it, it makes me excited for my career and future.