Introduction to Data Science Assignment 1 Class ID: 52

Class ID: 52

```
# HW 1 Due Monday Sept 18, 2017. Upload R file or notebook to Moodle with
# filename: HW1_490ID_YOURUNI.R
# Do Not remove any of the comments. These are marked by #

###Your Unique Class ID: 52
```

1.

```
# 1.
# Load the data for this assignment into your R session
# with the following command:
library(ggplot2)
data(diamonds)

#(1). Check to see that the data were loaded by running:
objects(diamonds)
```

```
##  [1] "carat"   "clarity" "color"   "cut"     "depth"   "price"   "table"
##  [8] "x"       "y"       "z"
```

```
# This should show ten variables:
# carat, clarity, color, cut, depth,
# price, table, x, y, z
```

2

```
#(2). Use the nrow() function to find out how many observations there are.
nrow(diamonds) # 53940 rows
```

```
## [1] 53940
```

```
# For the following questions, use one of: head(), summary(),
# class(), min(), max(), hist(), quantile(), table()
# to answer the questions.
```

Ans: There are 53940 observations

3

```
#(3). Show the summary or the structure of this dataset.
summary(diamonds)
```

```
##      carat               cut        color        clarity     
##  Min.   :0.2000   Fair      : 1610   D: 6775   SI1    :13065  
##  1st Qu.:0.4000   Good      : 4906   E: 9797   VS2    :12258  
##  Median :0.7000   Very Good:12082   F: 9542   SI2    : 9194  
##  Mean   :0.7979   Premium   :13791   G:11292   VS1    : 8171  
##  3rd Qu.:1.0400   Ideal     :21551   H: 8304   VVS2   : 5066  
##  Max.   :5.0100                     I: 5422   VVS1   : 3655  
##                                     J: 2808   (Other): 2531  
##      depth           table           price             x         
##  Min.   :43.00   Min.   :43.00   Min.   :  326   Min.   : 0.000  
##  1st Qu.:61.00   1st Qu.:56.00   1st Qu.:  950   1st Qu.: 4.710  
##  Median :61.80   Median :57.00   Median : 2401   Median : 5.700  
##  Mean   :61.75   Mean   :57.46   Mean   : 3933   Mean   : 5.731  
##  3rd Qu.:62.50   3rd Qu.:59.00   3rd Qu.: 5324   3rd Qu.: 6.540  
##  Max.   :79.00   Max.   :95.00   Max.   :18823   Max.   :10.740  
##                                                                  
##        y                z        
##  Min.   : 0.000   Min.   : 0.000  
##  1st Qu.: 4.720   1st Qu.: 2.910  
##  Median : 5.710   Median : 3.530  
##  Mean   : 5.735   Mean   : 3.539  
##  3rd Qu.: 6.540   3rd Qu.: 4.040  
##  Max.   :58.900   Max.   :31.800  
##
```

```
# summary :
#      carat               cut        color        clarity
#  Min.   :0.2000   Fair      : 1610   D: 6775   SI1     :13065
#  1st Qu.:0.4000   Good      : 4906   E: 9797   VS2     :12258
#  Median :0.7000   Very Good:12082   F: 9542   SI2     : 9194
#  Mean   :0.7979   Premium   :13791   G:11292   VS1     : 8171
#  3rd Qu.:1.0400   Ideal     :21551   H: 8304   VVS2    : 5066
#  Max.   :5.0100                     I: 5422   VVS1    : 3655
#                                     J: 2808   (Other): 2531
#      depth           table           price             x
#  Min.   :43.00   Min.   :43.00   Min.   :  326   Min.   : 0.000
#  1st Qu.:61.00   1st Qu.:56.00   1st Qu.:  950   1st Qu.: 4.710
#  Median :61.80   Median :57.00   Median : 2401   Median : 5.700
#  Mean   :61.75   Mean   :57.46   Mean   : 3933   Mean   : 5.731
#  3rd Qu.:62.50   3rd Qu.:59.00   3rd Qu.: 5324   3rd Qu.: 6.540
#  Max.   :79.00   Max.   :95.00   Max.   :18823   Max.   :10.740
#
#        y                z
#  Min.   : 0.000   Min.   : 0.000
#  1st Qu.: 4.720   1st Qu.: 2.910
#  Median : 5.710   Median : 3.530
#  Mean   : 5.735   Mean   : 3.539
#  3rd Qu.: 6.540   3rd Qu.: 4.040
#  Max.   :58.900   Max.   :31.800
```

```
dim(diamonds)
```

```
## [1] 53940    10
```

```
# Rows: 53940, Columns: 10
```

Ans: Dimension of diamonds: Rows: 53940, Columns: 10

4

```
#(4). List the categorical variables in this dataset. !!!
names(diamonds)
```

```
##  [1] "carat"   "cut"     "color"   "clarity" "depth"   "table"   "price"
##  [8] "x"       "y"       "z"
```

```
class(diamonds$carat)
```

```
## [1] "numeric"
```

```
class(diamonds$cut)
```

```
## [1] "ordered" "factor"
```

```
class(diamonds$color)
```

```
## [1] "ordered" "factor"
```

```
class(diamonds$clarity)
```

```
## [1] "ordered" "factor"
```

```
class(diamonds$depth)
```

```
## [1] "numeric"
```

```
class(diamonds$table)
```

```
## [1] "numeric"
```

```
class(diamonds$price)
```

```
## [1] "integer"
```

```
class(diamonds$x)
```

```
## [1] "numeric"
```

```
class(diamonds$y)
```

```
## [1] "numeric"
```

```
class(diamonds$z)
```

```
## [1] "numeric"
```

```
# Ans: cut, color, clarity
```

Ans: The categorical variables are : cut, color, clarity

```
#(5). What was the highest price of the diamonds ?
max(diamonds$price) # 18823
```

```
## [1] 18823
```

Ans: The highest price of the diamonds are : 18823

6

```
#(6). What was the average price of the diamonds ?
mean(diamonds$price) #3932.8
```

```
## [1] 3932.8
```

Ans: The average price of diamonds: 3932.8

7

```
#(7). What is the number of the Ideal cut ?
table(diamonds$cut == "Ideal") # TRUE == Ideal : 21551
```

```
##
## FALSE  TRUE
## 32389 21551
```

```
summary(diamonds$cut) # Ideal : 21551
```

```
##      Fair      Good Very Good   Premium     Ideal
##      1610      4906     12082     13791     21551
```

Ans: The number of Ideal cut = 21551

8

```
#(8). What is the number of the diamonds which are Premium and have a clarity leve
l
#     of IF?
summary(diamonds$clarity == "IF" & diamonds$cut == "Premium") # True: 230
```

```
##    Mode   FALSE    TRUE
## logical   53710     230
```

Ans : 230

9

```
#(9). What is the averge price difference between the clarity level SI2 and IF?
#     Hint(Use aggregate())
# 1. A way of writing the code without aggregate
SI2.price = diamonds$price[diamonds$clarity == "SI2" ]
IF.price = diamonds$price[diamonds$clarity == "IF"]
SI2.price.avg = sum(SI2.price) / length(SI2.price)
IF.price.avg = sum(IF.price) / length(IF.price)
abs(SI2.price.avg - IF.price.avg)  # 2198.189
```

```
## [1] 2198.189
```

```
# 2.  Using the aggregate function
aggr = aggregate(diamonds$price, by = list(diamonds$clarity), FUN=mean)
class(aggr) # data.frame
```

```
## [1] "data.frame"
```

```
abs(aggr$x[aggr$Group.1 == "IF"] - aggr$x[aggr$Group.1 == "SI2"]) # 2198.189
```

```
## [1] 2198.189
```

Ans: 2198.189

10

```
#(10). Total depth percentage is represented as the depth divided by
#      the mean of the length and width of the diamond,
```

11

```
# (11).
# Try running each expression in R.
# Record the error message in a comment
# Explain what it means.
# Be sure to directly relate the wording of the error message with the problem you
find in the expression.
```

z/mean(x, y)

```
### Error message here
#         Error: object 'z' not found
### Explanation here
```

Explanation: “Because the z variable was not declared to be anything before running the expression, so running this expression would result in R unable to find the z object, thus causing an error.”

```
#diamonds$z/(mean(diamonds$x, diamonds$y))
```

diamonds$z/(mean(diamonds$x, diamonds$y))

```
### Error message here
        #Error in mean.default(diamonds$x, diamonds$y) :
        #'trim' must be numeric of length one
### Explanation here
```

Explanation: “Because the mean function has several parameters, and only 1 default parameter. The expression using the mean(diamonds$x, diamonds$y) has the diamonds$y object match the ‘trim’ parameter, which takes a fraction (0 to 0.5) of observations to be trimmed form each end of the first argument in the mean function before the mean is computed. yet the parameter ‘diamonds$y’ that we passed in the mean function has a length of 53940 numeric values, thus causing an error that the ‘trim’ parameter must be numberic of length one.”

```
#diamonds$z/(rowMeans(diamonds$x, diamonds$y))
```

diamonds$z/(rowMeans(diamonds$x, diamonds$y))

```
### Error message here
        #Error in rowMeans(diamonds\$x, diamonds$y) :
        #  'x' must be an array of at least two dimensions
### Explanation here
```

Explanation: “The ‘rowMeans’ function has to take in an array of two or more dimensions or a numeric data frame. And the expression gave it diamonds$x as the first ‘x’ argument, which is only a 1 dimension numeric vector, thus and error occurs because the first argument must have more than 1 dimension.”

12

```
#(12).Study the following code about how to do the compution
# calculation that we want for the previous question.
Depth <- (diamonds$z)/rowMeans(cbind(diamonds$x, diamonds$y))
```

13

```
#(13). Can you get the same result without using function?
#         Given an expression to it.  Name the values Depth1
Depth1 = diamonds$z / ((diamonds$y + diamonds$x) / 2)
```

Ans: diamonds$z$/((diamonds$y + diamonds$x) / 2)

14

```
#(14) What did you get from
all.equal(Depth, Depth1)
```

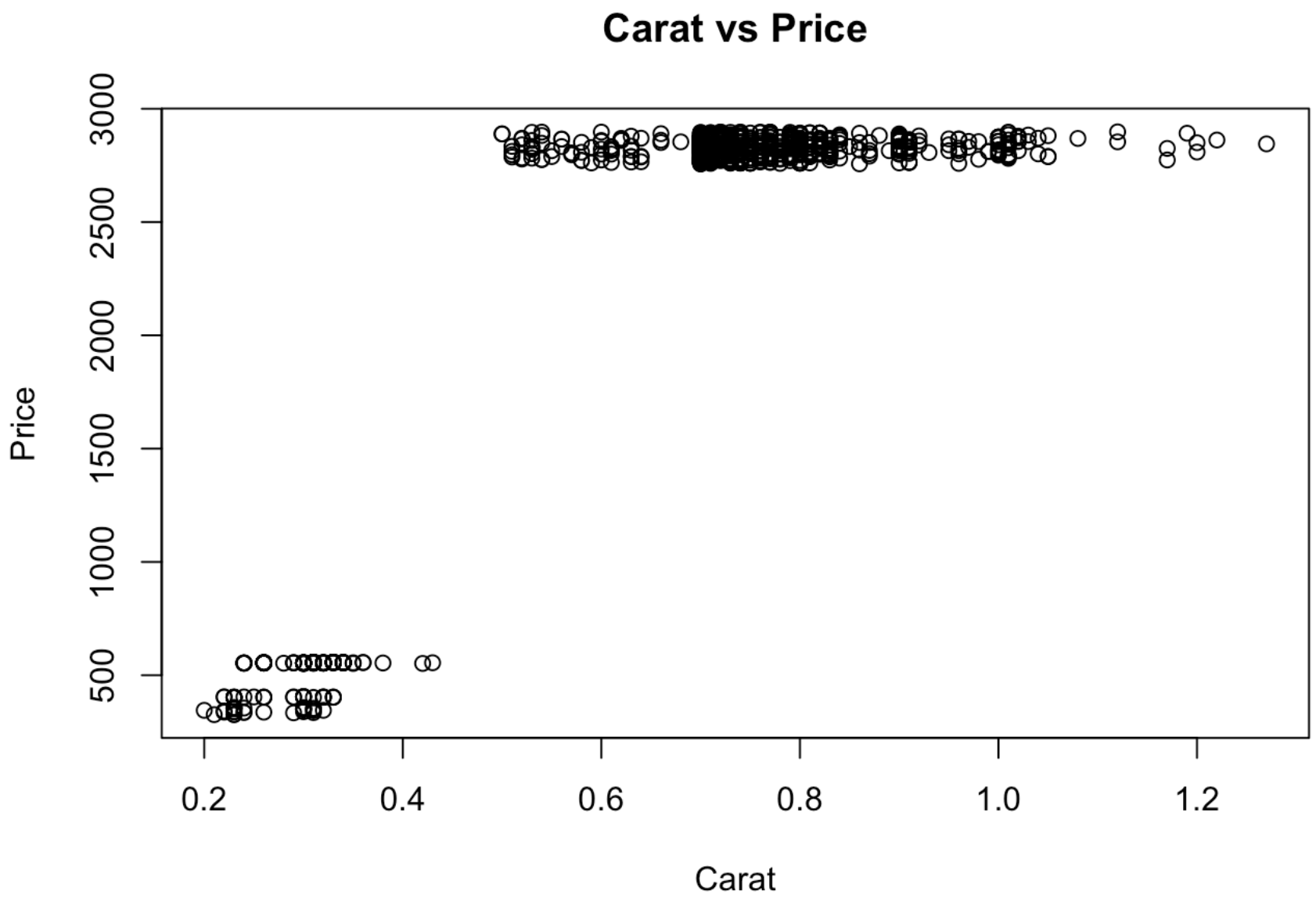
```
## [1] TRUE
```

```
# True
```

Ans: True

2.

```
# 2. Run the following code to make a plot.
# (don't worry right now about what this code is doing)
plot(diamonds[1:1000,]$carat, diamonds[1:1000,]$price, xlab = "Carat", ylab = "Pri
ce", main = "Carat vs Price")
```




```
# (1) Use the Zoom button in the Plots window to enlarge the plot.
# Resize the plot so that it is long and short, so it is easier to read.
# Include this plot in the homework your turn in.
```

2

```
# (2) Make an interesting observation about the relation between
#     Carat and Price based on this plot
# (something that you couldn't see with the calculations so far.)

### Your answer goes here
```

Ans: “The prices of diamonds with carat weight below around the weight of 0.5 carat prices range about from 200 to 600. As for the carat weight above 0.5 carat, the price ranges around 2700 to about 2950 which has a smaller range than diamonds that weigh less than 0.5 carat. Also, the relation between price

and carat of the diamond doesn't seem to have any correlation, as the price of the diamonds don't really correspond to the carat weight of the diamond for those below and above 0.5 carat weight respectively."

3

```
# (3) What interesting question about the diamonds
# would you like to answer with these data, but don't yet know
# how to do it?

### Your answer goes here
```

Ans: "Common intuition would suggest that a diamond with a higher weight in carat would have a higher pricing than those that have less weight in carat, does that hold true?"

" To answer this question, I would have to count the portions of diamonds with a higher price per carat than those that are heavier than itself in the dataset. If the portion is large, then it means that for diamonds, carat and price isn't highly correlated, and that heavier diamonds do not necessarily mean they have a higer price."

```
# For the remainder of this assignment we will work with
# one of the random number generators in R.
```

4.

```
# 4.
# Use the following information about you to generate some random values:
#a. Use you UIN number to set the seed in set.seed() function.
set.seed(665363548)
#b. Use your birthday month for the mean of the normal.
mean = 3
#c. Use your birthday day for the standard deviation (sd) of the normal curve.
sd = 19
#d. Generate 10 random values using the parameters from b and c.
rnorm(10, mean = 3, sd = 19)
```

```
##  [1] -27.338705  24.293371 -15.265145   1.857929  11.691374  24.648747
##  [7]   4.265548 -19.929143   1.843425  -8.853153
```

```
#e. Assign the values to a variable named with your first name.
Roger = rnorm(10, mean = 3, sd = 19)
#f. Provide/Show the values generated.
Roger
```

```
##  [1]  18.467346 -15.717807  30.216830   4.938865 -23.435029  -5.829286
##  [7] -23.721310 -30.204336  11.832295 -12.448093
```

```
# [1]  16.0435819  -8.7419770   1.4455514   1.6544967  12.3591525 -50.8375381
# [7]  -0.3500531 -15.4846238  15.7315860  -6.9326284
```

Ans: (a) set.seed(665363548) (b) mean = 3 (c) sd = 19 (d) rnorm(10, mean = 3, sd = 19) (e) Roger = rnorm(10, mean = 3, sd = 19) (f) [1] 16.0435819 -8.7419770 1.4455514 1.6544967 12.3591525 -50.8375381 [7] -0.3500531 -15.4846238 15.7315860 -6.9326284

5.

1

```
# 5.
#(1). Generate a vector called "normsamps" containing
# 100 random samples from a normal distribution with
# mean 5 and SD 2.
normsamps = rnorm(100, mean = 5, sd = 2)
```

Ans: normsamps = rnorm(100, mean = 5, sd = 2)

2

```
#(2). Calculate the mean and sd of the 100 values.
normsamps.mean = mean(normsamps) # [1] 4.956662
normsamps.sd = sd(normsamps) # [1] 1.949206
### The return values from your computation go here
# Mean = 4.956662
# SD = 1.949206
```

Ans: Mean : mean(normsamps) = 4.956662 SD : sd(normsamps) = 1.949206

3

```
# (3). Use implicit coercion of logical to numeric to calculate
# the fraction of the values in normsamps that are more than 8.
length(normsamps[normsamps > 8]) / length(normsamps) # 0.07
```

```
## [1] 0.07
```

Ans: length(normsamps[normsamps > 8]) / length(normsamps) 0.07

4

```
# (4). Look up the help for rnorm.
# You will see a few other functions listed.
# Use one of them to figure out about what answer you
# should expect for the previous problem.
# That is, find the area under the normal(5, 2) curve
# to the right of 8.  This should be the chance of getting
# a random value more than 8.


# What value do you expect?
 # expected value from the previous question : 0.07
# What value did you get?
1 - pnorm(8, mean = 5, sd = 2)
```

```
## [1] 0.0668072
```

```
#     0.0668072
```

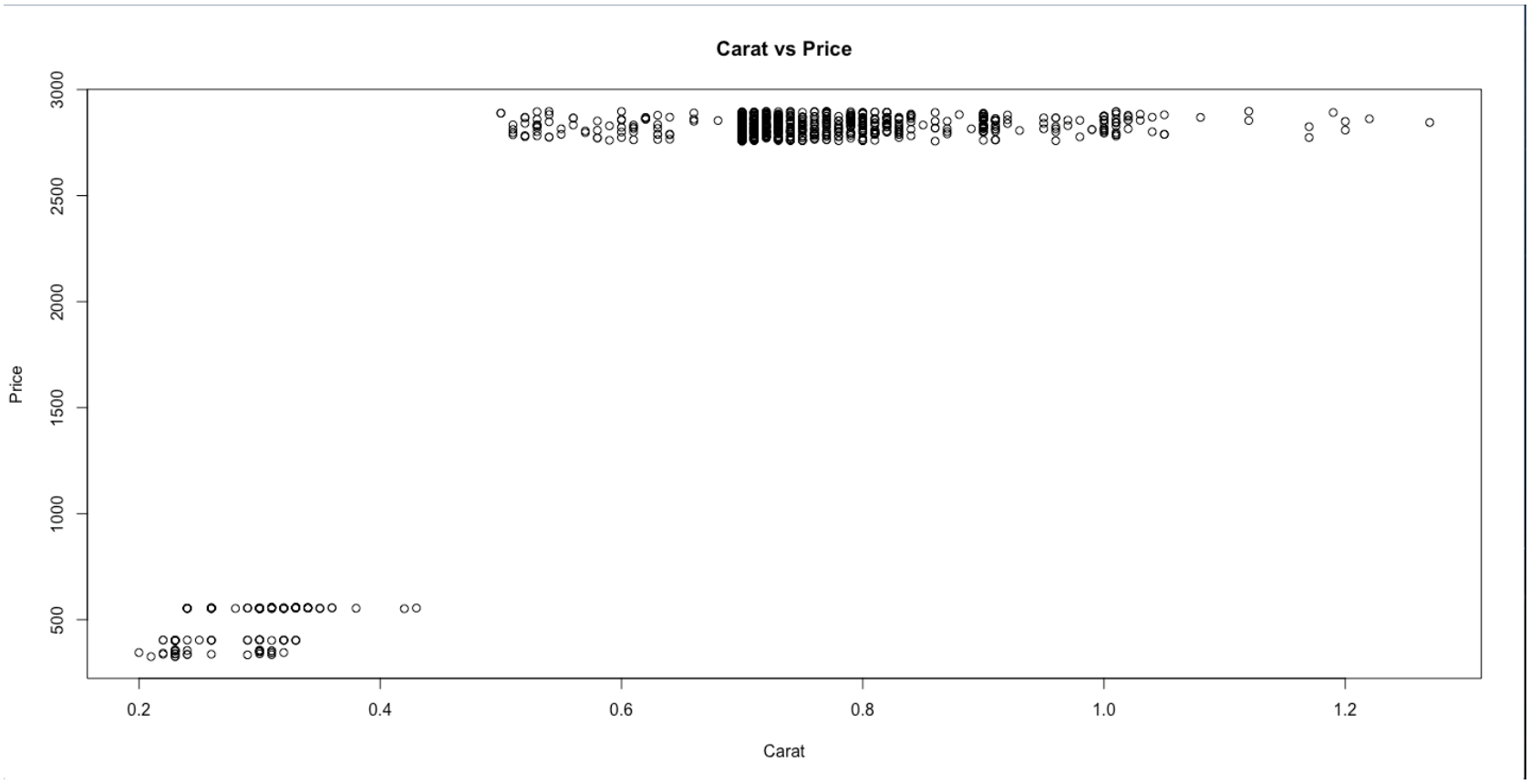
Ans: 0.0668072

```
# Why might they be different?
```

Ans: “Because the first one only had a sample of 100 which is a really small sample size, unable to precisely calculate what it may actually be. As for the R program, it uses a huge sample size to make sure the computation precise as it can be.”"

2

```
# (1) Use the Zoom button in the Plots window to enlarge the plot.
# Resize the plot so that it is long and short, so it is easier to read.
# Include this plot in the homework your turn in.
```


Carat vs Price
Price
3000
2500
2000
1500
1000
500
0.2
0.4
0.6
0.8
1.0
1.2
Carat