Title: “Knowledge Mining EPPS 6323 Dr. Ho Lab04”
Author: Glen Cooper
Date: 2/20/2022

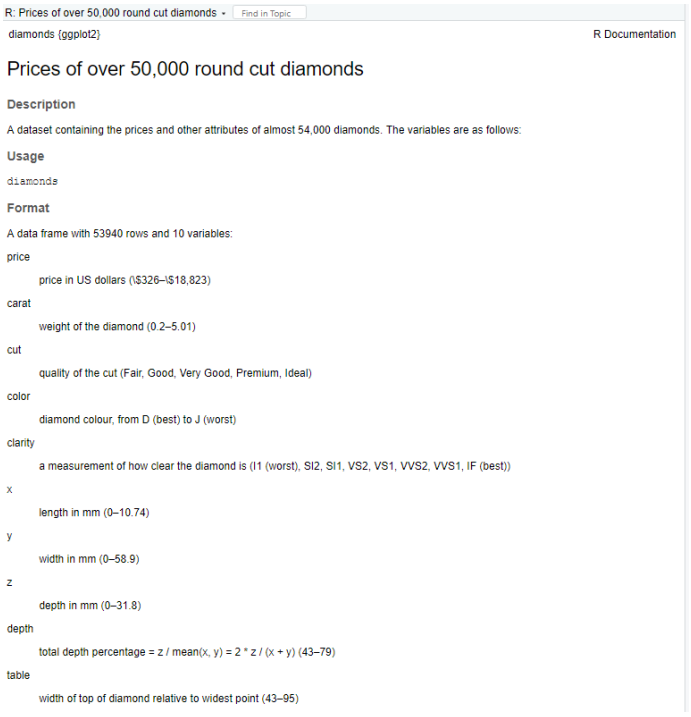
CODE

```
# R Programming (EDA 2)
# Adapted from R4DS Chapter 7: Exploratory Data Analysis
# STUDENT NOTATION: All responses to question, added coding, or other code fixes are indicated
#                   by comments enclosed in "[Brackets]"

# Prerequisite: use preload function to get tidyverse and descr load them
install.packages("descr")                    # [Glen add to Lab04 on 2/19/22]
install.packages("RColorBrewer")             # [Install RColorBrewer package]
library("RColorBrewer")                      # [Load RColorBrewer]
library("rmarkdown")
library(tidyverse)
library(descr) # Describe attributes of objects/variables
library(ggplot2) #[Added library]
setwd("C:/Users/glenc/Downloads") #[set working directory]

# Explore the diamonds dataset
?diamonds
```

OUTPUT



R: Prices of over 50,000 round cut diamonds — Find in Topic

diamonds {ggplot2} — R Documentation

Prices of over 50,000 round cut diamonds

Description

A dataset containing the prices and other attributes of almost 54,000 diamonds. The variables are as follows:

Usage

```
diamonds
```

Format

A data frame with 53940 rows and 10 variables:

price
: price in US dollars (\$326–\$18,823)

carat
: weight of the diamond (0.2–5.01)

cut
: quality of the cut (Fair, Good, Very Good, Premium, Ideal)

color
: diamond colour, from D (best) to J (worst)

clarity
: a measurement of how clear the diamond is (I1 (worst), SI2, SI1, VS2, VS1, VVS2, VVS1, IF (best))

x
: length in mm (0–10.74)

y
: width in mm (0–58.9)

z
: depth in mm (0–31.8)

depth
: total depth percentage = z / mean(x, y) = 2 * z / (x + y) (43–79)

table
: width of top of diamond relative to widest point (43–95)

CODE

```
# Examine the variables using the class function
class(cut)
sapply(diamonds,class)
```

OUTPUT

```
> # Examine the variables using the class function
> class(cut)
[1] "ordered" "factor"
> sapply(diamonds,class)
$carat
[1] "numeric"

$cut
[1] "ordered" "factor"

$color
[1] "ordered" "factor"

$clarity
[1] "ordered" "factor"

$depth
[1] "numeric"

$table
[1] "numeric"

$price
[1] "integer"

$x
[1] "numeric"

$y
[1] "numeric"

$z
[1] "numeric"
```

CODE

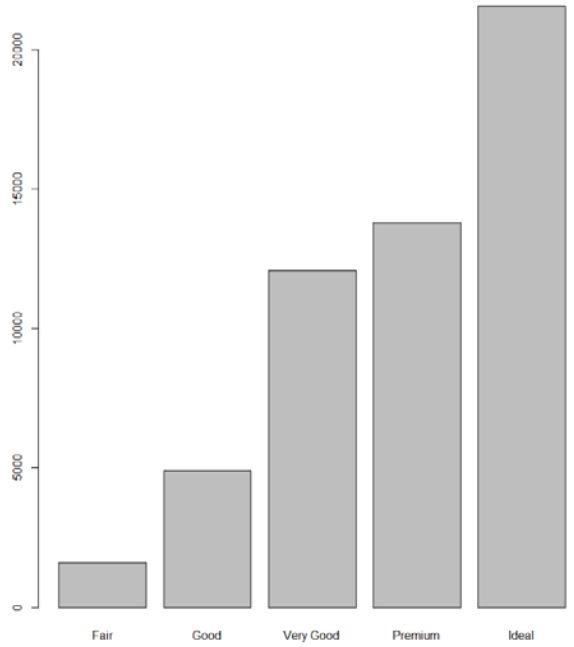
```
# Frequency table from descr package
attach(diamonds)
freq(cut,plot=F) # can add a plot by default [Added "q" to "fre(cut,plot=F)"]
freq(cut)       # [Added plot per Dr. Ho's request]
```

OUTPUT

```
> # Frequency table from descr package
> attach(diamonds)
The following objects are masked from diamonds (pos = 4):

    carat, clarity, color, cut, depth, price, table, x, y, z

> freq(cut,plot=F) # can add a plot by default [Added "q" to "fre(cut,plot=F)"]
cut
          Frequency Percent Cum Percent
Fair           1610   2.985       2.985
Good           4906   9.095      12.080
Very Good     12082  22.399      34.479
Premium       13791  25.567      60.046
Ideal         21551  39.954     100.000
Total         53940 100.000
> freq(cut)        # [Added plot per Dr. Ho's request]
cut
          Frequency Percent Cum Percent
Fair           1610   2.985       2.985
Good           4906   9.095      12.080
Very Good     12082  22.399      34.479
Premium       13791  25.567      60.046
Ideal         21551  39.954     100.000
Total         53940 100.000
>
```



CODE

```
# Frequency table, alternative method

with(diamonds, {freq(cut, plot=T)})
```

OUTPUT

```
> # Frequency table, alternative method
> with(diamonds, {freq(cut, plot=T)})
cut
          Frequency Percent Cum Percent
Fair           1610   2.985       2.985
Good           4906   9.095      12.080
Very Good     12082  22.399      34.479
Premium       13791  25.567      60.046
Ideal         21551  39.954     100.000
Total         53940 100.000
> |
```

CODE

```
# Create tibble object out of diamonds dataset

diam_tb=as_tibble(diamonds)

class(diam_tb) # How is this different from a data frame?

        #[Tibbles have a refined print method that shows only the first 10 rows, and all the columns that fit on screen]
```

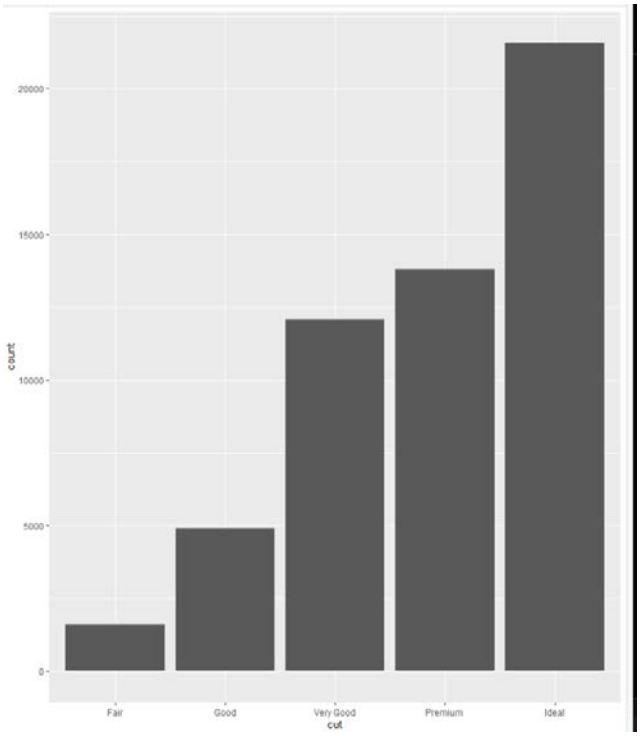
OUTPUT

```
> # Create tibble object out of diamonds dataset
> diam_tb=as_tibble(diamonds)
> class(diam_tb) # How is this different from a data frame?
[1] "tbl_df"     "tbl"        "data.frame"
>               #[Tibbles have a refined print method that shows only the first 10 rows, and all the columns that fit on screen]
> |
```

CODE

```
# Plot the bar chart using ggplot2
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut))
```

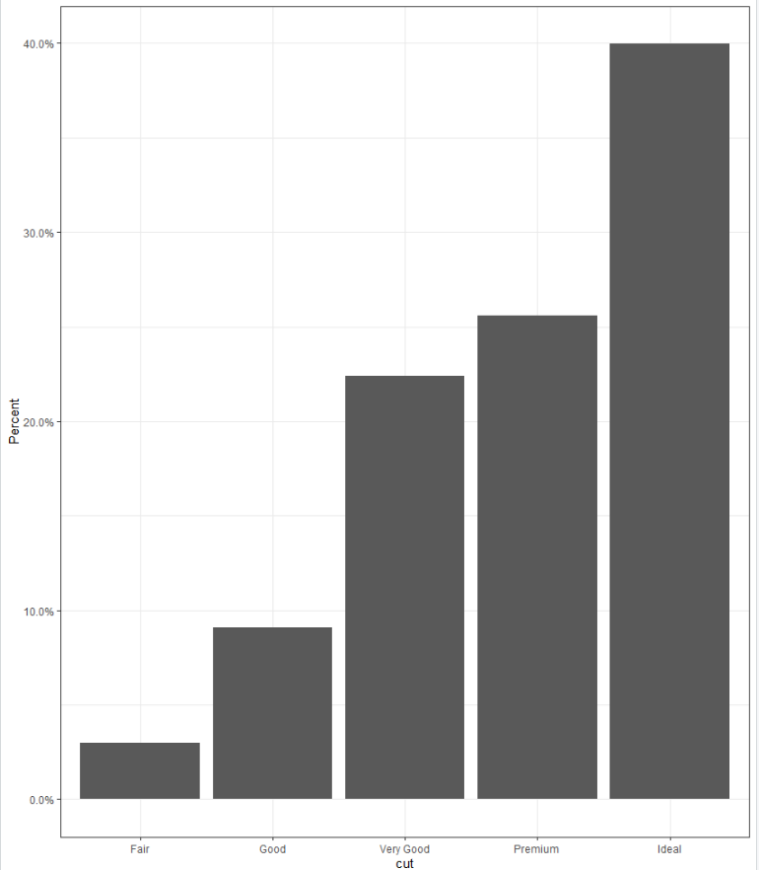
OUTPUT



CODE

```
# Better plot with percentage on y axis
ggplot(data = diamonds,aes(cut)) +
  geom_bar(mapping = aes(y = (..count..)/sum(..count..))) + theme_bw() +
  scale_y_continuous(labels=scales::percent) +
  ylab("Percent")
```

OUTPUT



CODE

```
# Exercise 1

# Save the chart into PNG, PDF and SVG formats

# What are the differences among all these formats?

# [Responses:

#  PNG is a picture file good for editing pixels

#  PDF is an Adobe Acrobat file good for sending text and images to others

#  SVG or a Scalable Vector Graphic (SVG) is a type of image format.

#  Unlike other varieties, SVGs don’t rely on unique pixels Instead, they use ‘vector’ data.

#  This makes them scalable for web design]
```

OUTPUT

None

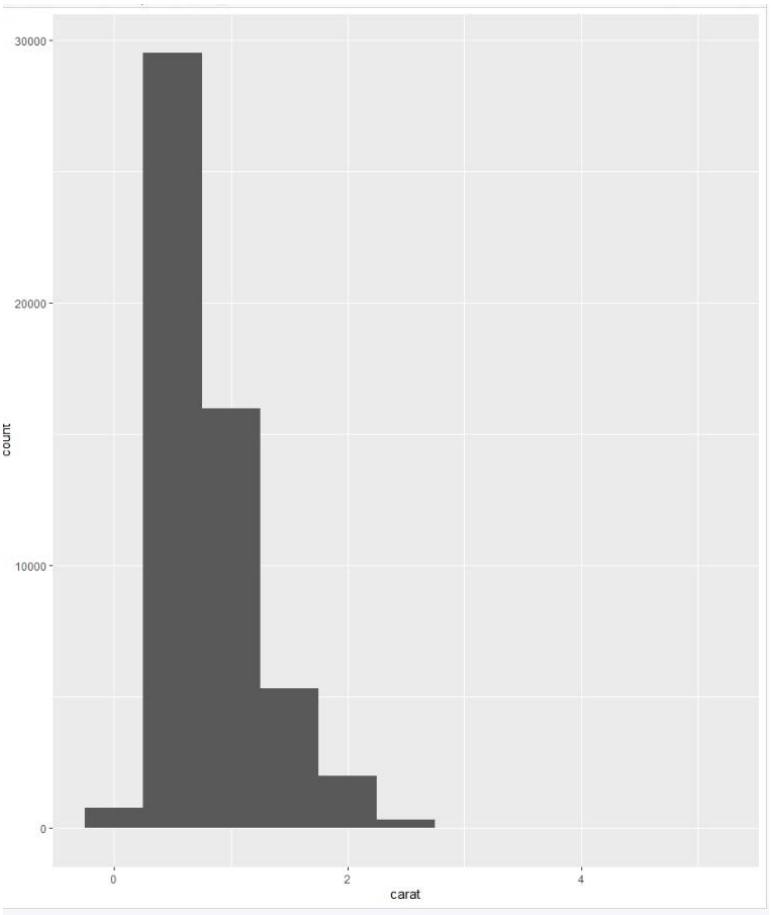
CODE

```
# Plot another variable carat

ggplot(data = diam_tb) +

  geom_histogram(mapping = aes(x = carat), binwidth = 0.5)
```

OUTPUT



CODE

```
# Exercise 2

# Can you improve the visualization of this chart?

# [Response: Yes this might be better using percentages]

# What is the difference between barchart and histogram?

# [Response: barcharts should distinct bars for each grouping while histograms are continuous unseparated bars.]
```

OUTPUT

None

CODE

```
# Subsetting the dataset
smaller <- diamonds %>%
  filter(carat < 3)
```

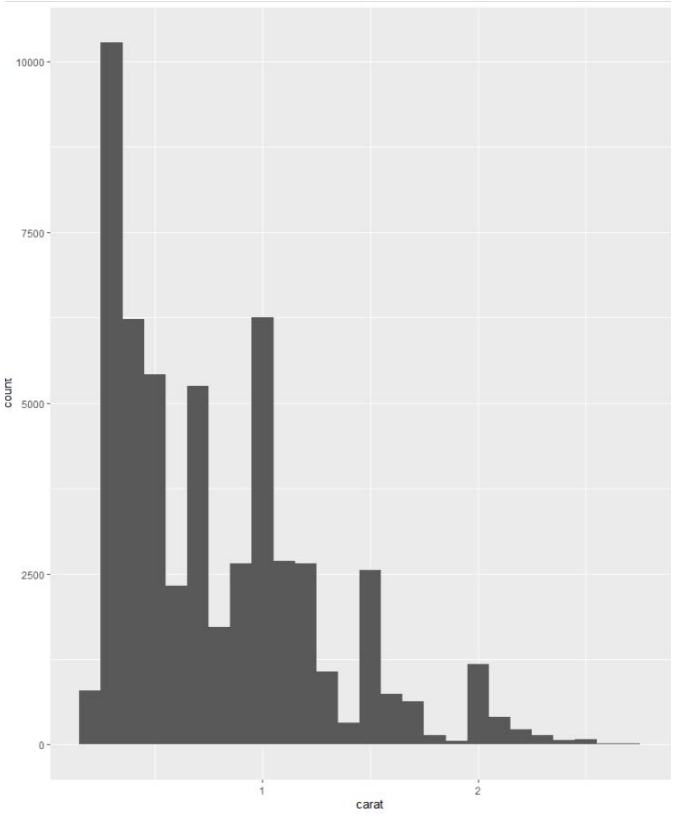
OUTPUT

None

CODE

```
# Histogram of smaller dataset
ggplot(data = smaller, mapping = aes(x = carat)) +
  geom_histogram(binwidth = 0.1)
```

OUTPUT



CODE

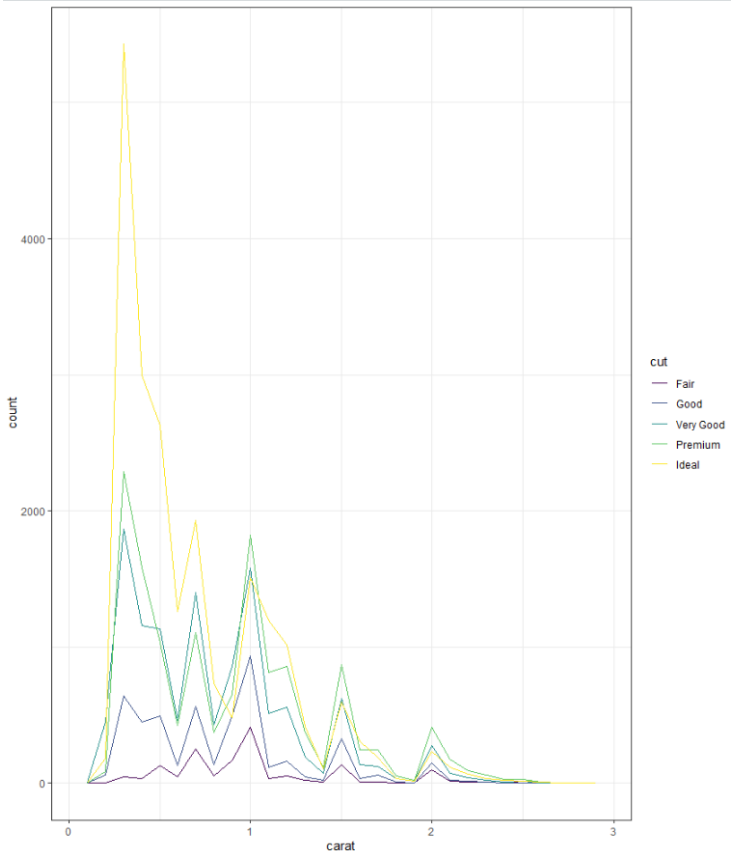
```
# Polygon

ggpolygon <- ggplot(data = smaller, mapping = aes(x = carat, colour = cut)) +

  geom_freqpoly(binwidth = 0.1) + theme_bw()       #[Added graph assignment to a data frame ggpoloygon]

ggpolygon                                  #[Added a display of the data frame ggpoloygon]
```

OUTPUT



CODE

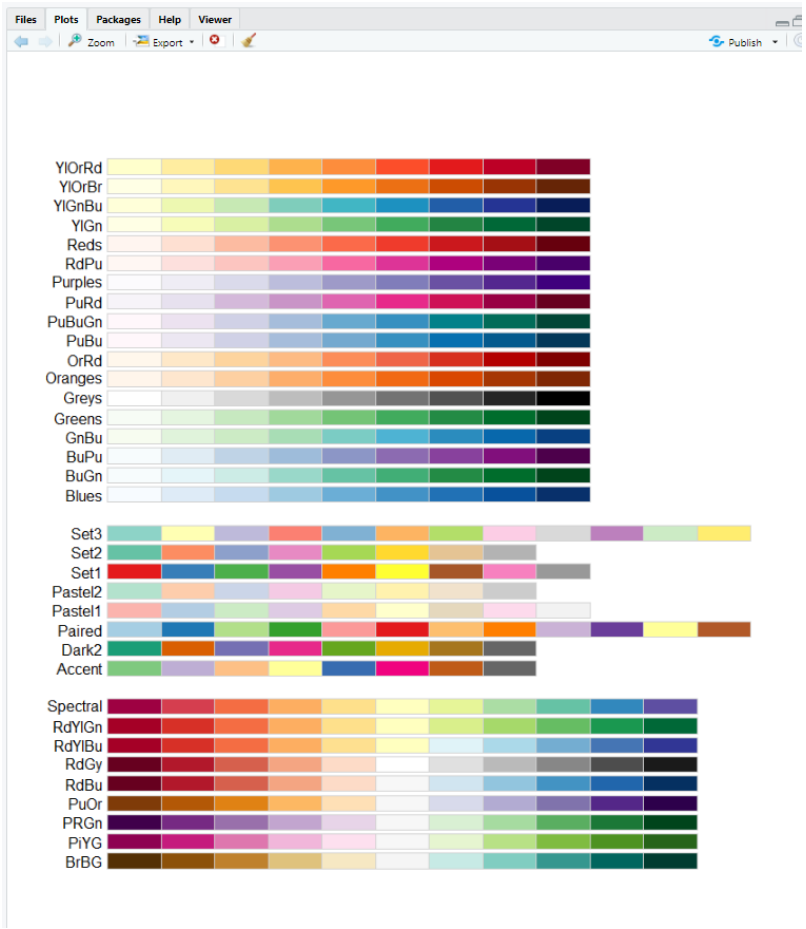
```
# Exercise 3

# Can you change colors?

# Hint: use the RcolorBrewer package

display.brewer.all(colorblindFriendly = FALSE)     # [Show all color palettes]
```

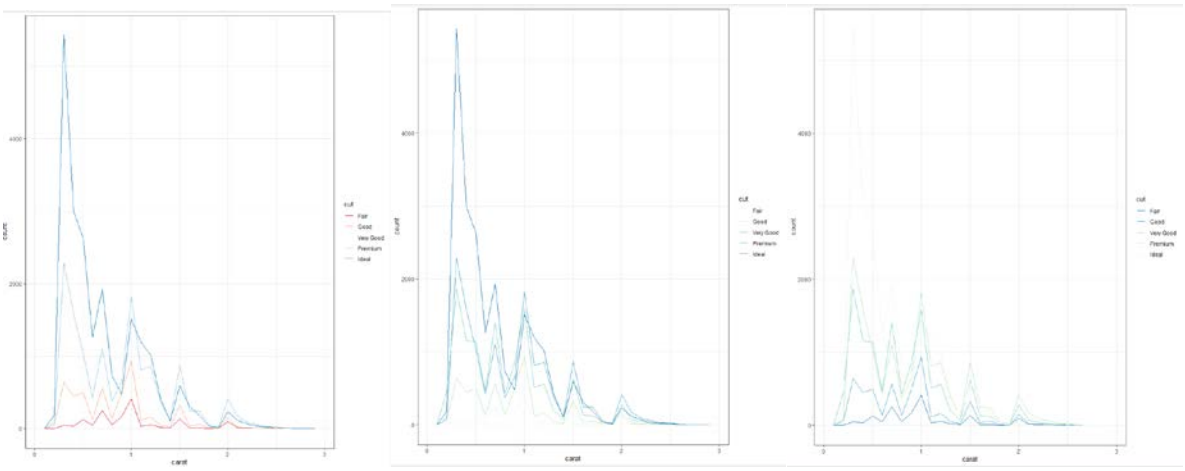
OUTPUT



CODE

```
ggpolygon +                          # [Apply scale_colour_brewer function]
  scale_colour_brewer(palette = "RdBu")
ggpolygon +                          # [Change color brewer palette to Red Blue]
  scale_colour_brewer(palette = "GnBu")          # [Change color brewer palette to Green Blue]
ggpolygon +                          # [Reverse color direction of Green Blue plot]
  scale_colour_brewer(palette = "GnBu", direction = - 1)
```

OUTPUT



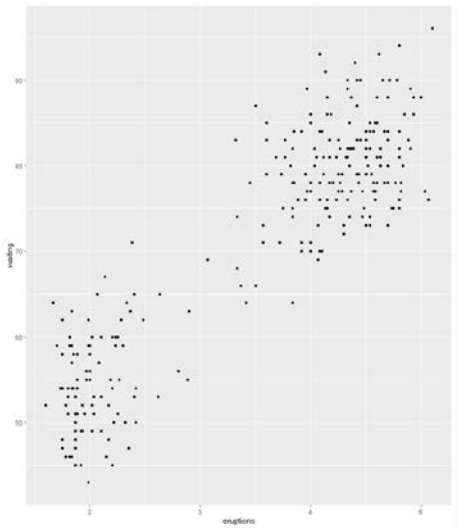
CODE

```
# Continuous variables

ggplot(data = faithful) +

  geom_point(mapping = aes(x = eruptions, y = waiting))
```

OUTPUT



CODE

```
# Exercise 4

# What method you will use to analyze: [Regression]

# Dependent variable: continuous[carat], Independent variable: discrete [price]

# Dependent variable: discrete [price], Independent variable: continuous [carat]

# Dependent variable: continuous [carat], Independent variable: continuous [depth]
```

OUTPUT

None