V506 R Introductory Lab 3 Exercise – Fall 2024 - Solution

Luis Navarro

August 2024

1. Install DescTools library. Explore it. Read the help files for functions Freq and Mode.

```
# install DescTools
install.packages("DescTools")
?DescTools
```

2. Load "dplyr","ggplot2", "here" and "DescTools" libraries using "pacman" into your environment.

```
# run this code on your computer to install all the packages required for the code below
if(!require(dplyr)) {install.packages("dplyr")}
```

```
## Loading required package: 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
```

```
if(!require(pacman)) {install.packages("pacman")}
```

```
## Loading required package: pacman
```

```
if(!require(here)) {install.packages("here")}
```

```
## Loading required package: here
```

```
## here() starts at /Users/luisenriquenavarro/Library/CloudStorage/OneDrive-IndianaUniversity/V506/Fall
```

```
if(!require(rio)) {install.packages("rio")}
```

```
## Loading required package: rio
```

```
## Warning: package 'rio' was built under R version 4.4.1
```

```r
# Load the packages into your environment
library(pacman)
p_load(dplyr, ggplot2, here, DescTools)

# Clean the environment
rm(list=ls())
```

3. Ggplot includes the dataset diamonds which has 53940 rows and 10 variables. Load this data set and, using pipes, print out the summary statistics of this dataset.

```r
# Load diamonds data
diamonds_data <- ggplot2::diamonds

diamonds_data %>%
  summary()
```

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

4. Use the Freq function to build a frequency table of the variable `cut`. Look at the output and interpret it.

```r
# Get the frequency table using Freq from DescTools
Freq(diamonds_data$cut)
```

Note that Freq produces as output a data frame with character variables. This is convenient for displaying the results of the analysis. It might be less convenient if you want to manipulate the output from the frequency table.

Also note you can use pipes for this syntax. Hint: you can use the function `pull` from dplyr to extract the values from cut as a vector (instead of using the dollar sign syntax)

```
diamonds_data %>%
  # get the data from variable cut as a vector
  pull(cut) %>%
  # compute the frequency table
  Freq()
```

An alternative for that, is to use the `tabyl` function from the `janitor` library.

```
# Extra
# Install Janitor
install.packages("janitor")
```

```
library(janitor)
```

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

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

```
tabyl(diamonds_data$cut) %>% tibble()
```

```
# also with pipes
diamonds_data$cut %>%
  # do the frequency table
  tabyl() %>%
  # conver the frequency table to a tibble (data frame) object
  tibble()
```

5. Find the mode of variable cut. Interpret it using the output from the frequency table computed in the previous question.

```
Mode(diamonds_data$cut)
```

```
## [1] Ideal
## attr(,"freq")
## [1] 21551
## Levels: Fair < Good < Very Good < Premium < Ideal
```

6. Find the mean diamond price and the mean diamond price. Are they different?

```
mean_price = mean(diamonds_data$price)
median_price = median(diamonds_data$price)
```

```
# Show the results. Note you can use paste to create strings of text dependent on your analysis.
# Round justs limits the number of decimals of the number
paste("Average Price: ", round(mean_price,4), sep = "")
```

```
## [1] "Average Price: 3932.7997"
```

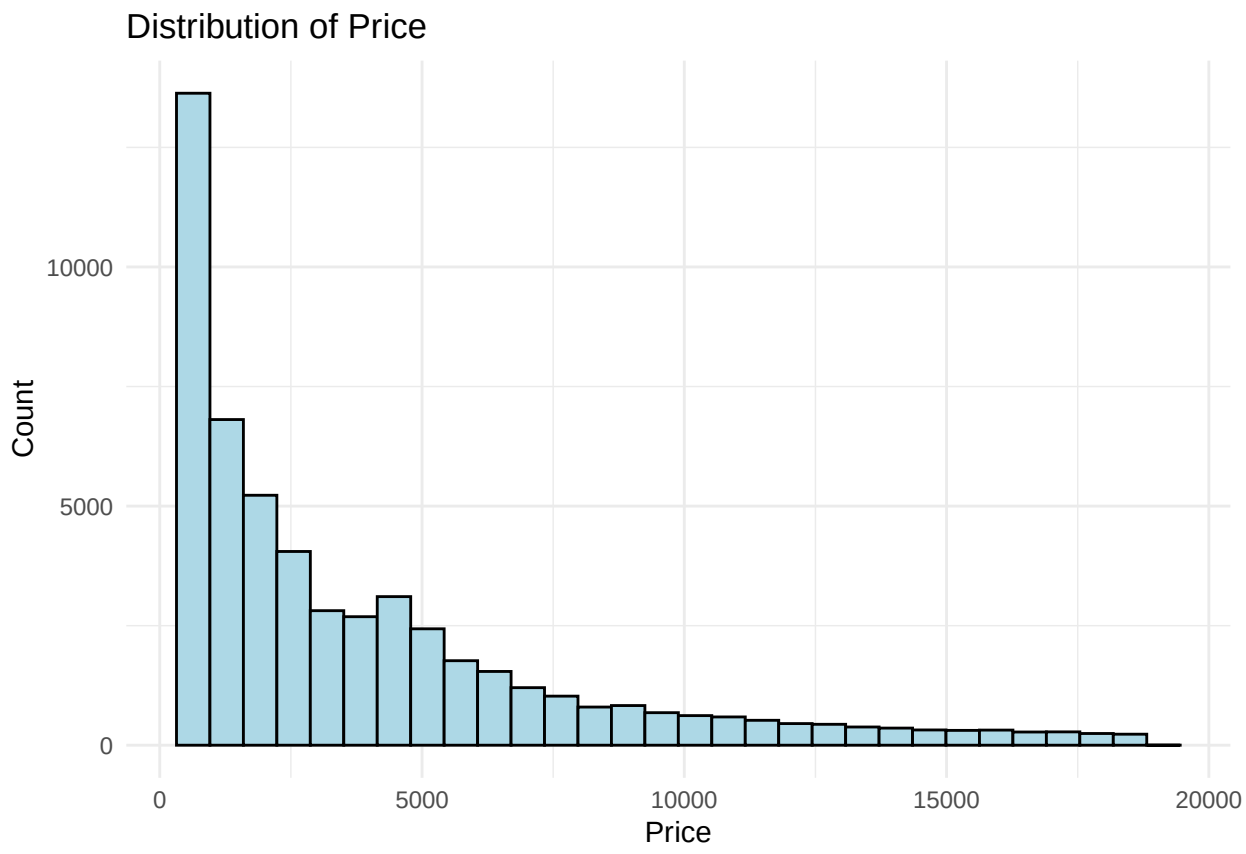
```
paste("Median Price: ", round(median_price,4), sep = "")
```

```
## [1] "Median Price: 2401"
```

7. Using ggplot(), create a histogram of the price distribution of the diamonds. Does the shape of the distribution make sense given the mean and median price? This chart should follow chart best practices—labels on the axes, a title, an appropriate binwidth, etc.

```
# note you can use pipes with ggplot.
# just be careful with the distinction between pipes and + signs
# pipes are for dplyr code
# + signs are for ggplot objects and layers
price_histogram <- diamonds_data %>%
  ggplot(mapping = aes(x = price)) +
  geom_histogram(color = "black", fill = "lightblue") +
  labs(x = "Price", y = "Count", title = "Distribution of Price") +
  theme_minimal()

price_histogram
```

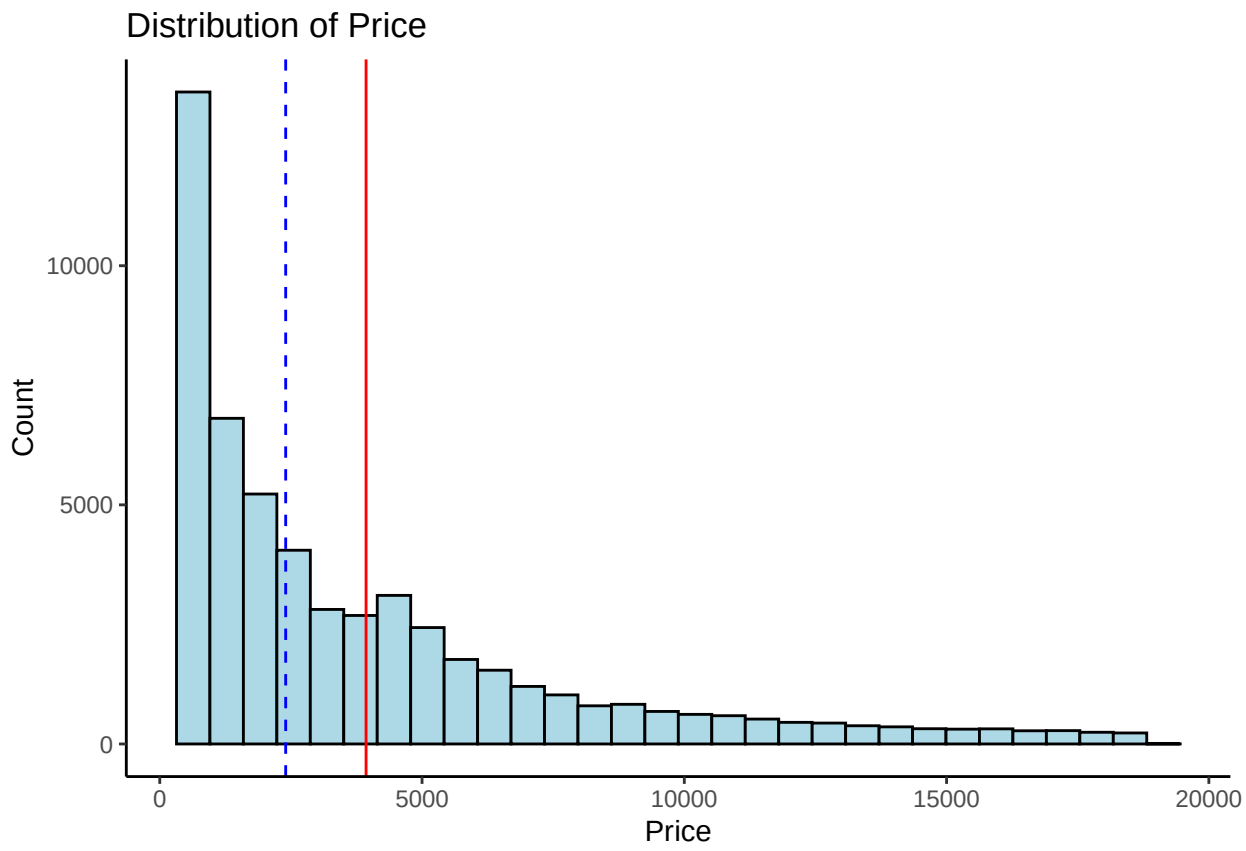
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



8. Use the geom_vline layer of ggplot to add vertical lines to depict the mean and median price. Interpret the graph with these new features.

```
price_histogram +
  geom_vline(xintercept = mean_price, color = "red", linetype = "solid") +
  geom_vline(xintercept = median_price, color = "blue", linetype = "dashed") +
  # note you can overwrite the theme and other layers
  theme_classic()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

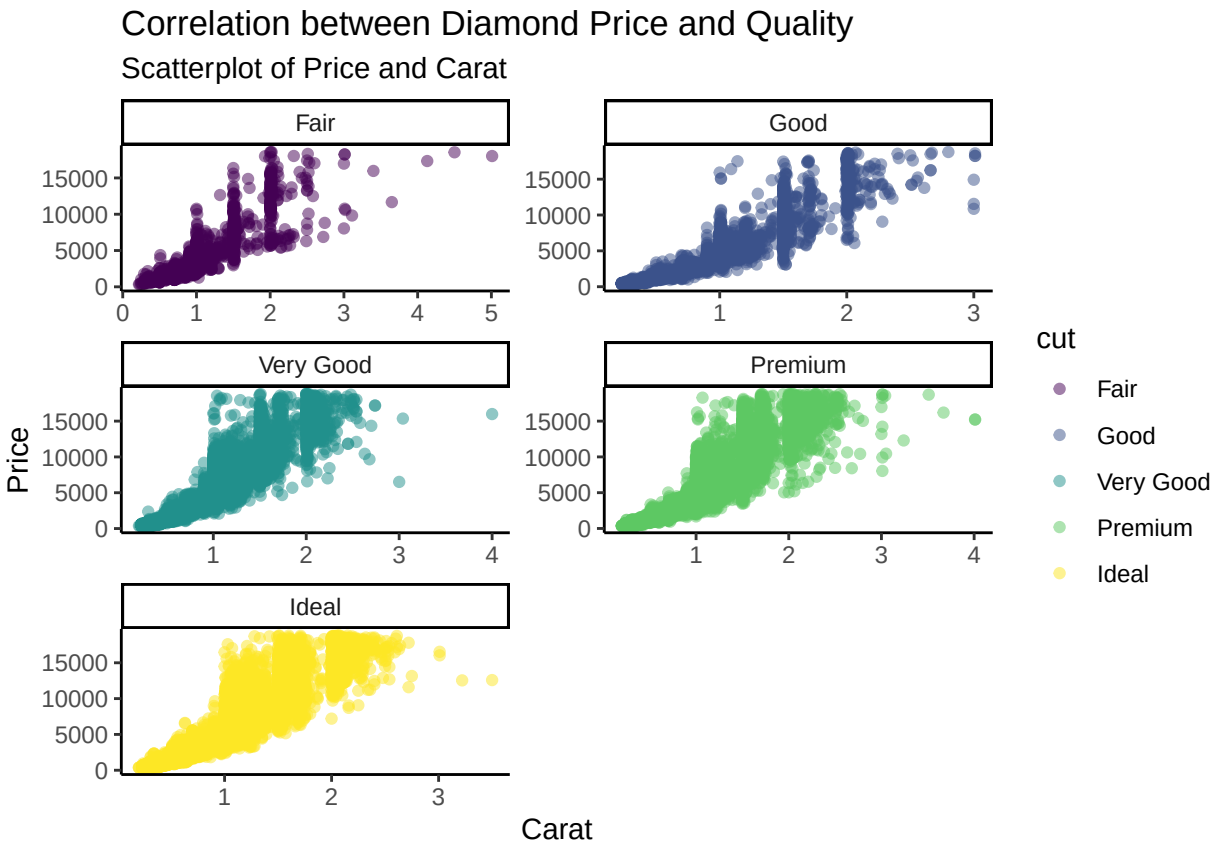


9. Save a new dataset called diamonds2 where you have added a variable that is equal to 1 if the price of the diamond greater than or equal to the median price. This variable will be equal to 0 if the price of the diamond is below the median price. Call this new variable high_price. (You should use ifelse(), mutate, and pipes to create this variable).

```
library(here)
v506_path <- here("/Users/luisenriquenavarro/Library/CloudStorage/OneDrive-IndianaUniversity/V506/Fall24

diamonds2 <- diamonds %>%
  mutate(high_price = ifelse(price >= median_price, 1, 0))

saveRDS(diamonds2, here(v506_path, 'Lab3', 'diamonds2.Rds'))
```

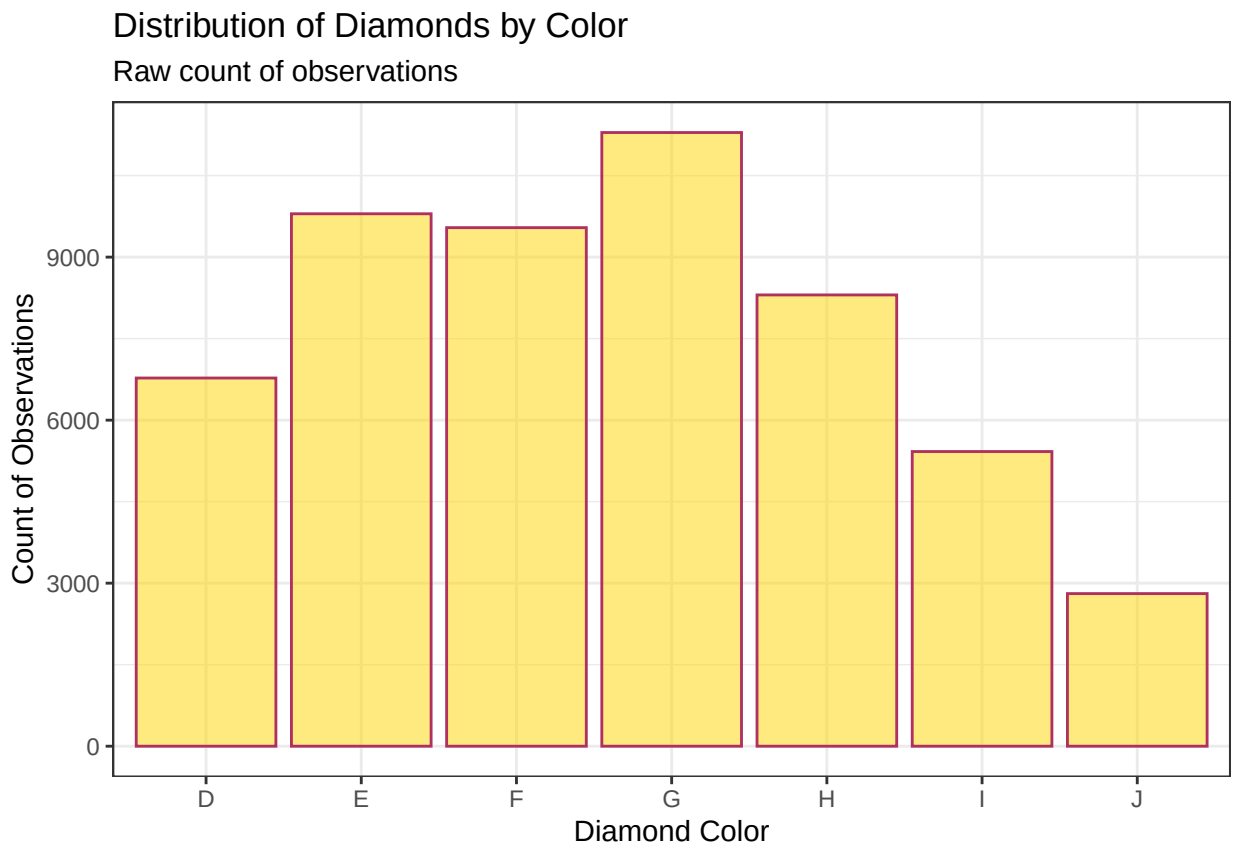
10. Using ggplot(), create a scatterplot with carat on the x-axis, price on the y-axis. Use the facet_wrap() to create 5 separate plots for each "cut". Again, include the axis titles and a main title.

```
# note you can use ggplot without pipes as well
scatter_diamonds <- ggplot(data = diamonds_data,
         mapping = aes(x = carat, y = price,
                       # you can specify individual colors by categories on variable cut
                       color = cut)) +
  geom_point(alpha = 0.5) +
  labs(x = "Carat", y = "Price", title = "Correlation between Diamond Price and Quality", subtitle = "S
  theme_classic() +
  facet_wrap(~cut, scales = "free", ncol = 2)

scatter_diamonds
```



11. Using ggplot(), create a bar chart where the x-axis is the "color" of the diamond. Again, include the axis titles and a main title.

```
diamonds_color_bar <- diamonds_data %>%
  ggplot(mapping = aes(x = color)) +
  # parameters inside are for formatting
  geom_bar(color = "maroon", fill = "gold", alpha = 0.5) +
  labs(x = "Diamond Color", y = "Count of Observations",
       title = "Distribution of Diamonds by Color",
       subtitle = "Raw count of observations") +
  theme_bw()

diamonds_color_bar
```



Extra: save your graphs

```
# Where to store the graph
file <- here(v506_path, 'Lab3','scatter_diamonds.jpg')
# Save the graph
ggsave(filename = file, plot = scatter_diamonds)
```

```
## Saving 6.5 x 4.5 in image
```