---

output: Basic data exploration: Diamonds

html\_document: default

pdf\_document: default

---

```{r setup, include=FALSE}

knitr::opts\_chunk$set(echo = TRUE)

```

### Free memory Functions

# Clear environment

rm(list = ls())

# Clear packages

# pacman::p\_unload(rgl)

# Clear plots

# dev.off() # But only if there IS a plot

# Clear console

cat("\014") # ctrl+L

## Load libraries

```{r}

library(tidyverse) #packages you may need dplyr, magrittr, ggplot2, stringr

library(lessR)

library(forcats)

library(DataExplorer)

library(ggthemes)

```

## Check the structure

```{r}

```

# Check details on the data frame, diamonds

```{r}

lessR::details(diamonds)

```

####################################################################################################

## Graphical Methods

####################################################################################################

Now that we’ve checked for missing values and typos and made corrections, we can graphically examine the sample data distribution of our data. Frequency distributions are useful because they can help us visualize the center (e.g., RV) and spread or dispersion (e.g., low and high) of our data. Typically, in introductory statistics, the normal (i.e., Gaussian) distribution is emphasized.

#Short Description of Graphical Methods

#Plot Types Description

#Bar a plot where each bar represents the frequency of observations for a ‘group’

#Histogram a plot where each bar represents the frequency of observations for a ‘given range of values’

#Density estimation of the frequency distribution based on the sample data

#Box-Whisker is a visual representation of median, quartiles, symmetry, skewness, and outliers

#Scatter & Line a graphical display of one variable plotted on the x-axis and another on the y axis

#Quantile-Quantile a plot of the actual data values against a normal distribution

## Histogram

A histogram is like a bar plot, except that instead of summarizing categorical data, it categorizes a continuous variable like clay content into non-overlapping intervals for the sake of display. The number of intervals can be specified by the user or can be automatically determined using an algorithm, such as nclass. Since histograms are dependent on the number of bins, for small data sets they are not the best method of determining the shape of a distribution.

# Histogram -

# Create a histogram of "carat" using base graphics

```{r}

```

# Histogram - Create a histogram of "carat" using ggplot2 and binwidth=15. Try different binwidths.

```{r}

```

# We can create a frequency count by creating bins of the "carat" variable by 0.1. Explore using the functions count and cut\_width()

```{r binning}

diamonds %>% count(cut\_width(carat,.1))

```

Explore the distribution of price. Do you discover anything unusual or surprising? (Hint: carefully think about the bin width and make sure you try a wide range of values.)

## Create a histogram of the variable, "price", using ggplot2. Try binwidth =10, binwidth=100, binwidth=200

```{r}

```

# When experimenting with different binwidths, we see there is a break in the plot happens because there is no price=$1500 in the data.

## Find the frequency count of each level in the variable "cut".

```{r}

```

## Count by cut and clarity

```{r}

diamonds %>% count(cut, clarity)

```

## The variable "cut" is an ordinal factor. Create a stacked histogram of the variable "price" color by the variable "cut" using ggplot2.

```{r}

```

## Use a violin plot to explore the relationship of price and cut.

```{r}

ggplot(diamonds,aes(y=price, x=cut)) +

geom\_point() +

geom\_violin()

```

## Create an object, smaller, filtering diamonds with a size of fewer than three carats. Create a histogram of price. Choose a smaller bandwidth than 30.

```{r}

```

## Exploring the relationship between variables a. price and cut and b. price and clarity for diamonds less than 3 carats.

```{r}

ggplot(filter(smaller), aes(x = price, fill=cut)) +

geom\_histogram(binwidth = 50)

ggplot(filter(smaller), aes(x = price, fill=clarity)) +

geom\_histogram(binwidth = 100)

```

# Density Curve

A density estimation, also known as a Kernel density plot, generally provides better visualization of the shape of the distribution in comparison to the histogram. Compared to the histogram where the y-axis represents the number or percent (i.e., frequency) of observations, the y-axis for the density plot represents the probability of observing any given value, such that the area under the curve equals one. One curious feature of the density curve is the hint of two peaks (i.e., bimodal distribution?).

## Create a density plot using the variable, carat, using ggplot2

```{r}

```

## Create a density plot using the variable, carat, and color by clarity. Use ggplot2.

```{r}

```

## Box plots

Box plots are a graphical representation of the five-number summary, depicting quartiles (i.e., the 25%, 50%, and 75% quantiles), minimum, maximum, and outliers (if present). Boxplots convey the shape of the data distribution, the presence of extreme values, and the ability to compare with other variables using the same scale, providing an excellent tool for screening data, determining thresholds for variables, and developing working hypotheses.

The parts of the boxplot are shown in the figure below. The “box” of the boxplot is defined as the 1st quartile, (Q1 in the figure) and the 3rd quartile, (Q3 in the figure). The median, or 2nd quartile, is the dark line in the box. The whiskers (typically) show data that is 1.5 \* IQR above and below the 3rd and 1st quartile. Any data point that is beyond a whisker is considered an outlier.

That is not to say the outlier points are in error, just that they are extreme compared to the rest of the dataset. However, you may want to evaluate these points to ensure that they are correct.

## Create boxplot of variable, carat, with base graphics

```{r}

```

## Create a boxplot of variable, carat, using ggplot2. Mark the plot at x=1.

```{r}

```

## Create boxplots of variable, cut and carat, using ggplot2.

```{r}

```

```{r}

#reorder by median

diamonds %>%

mutate(cut = factor(cut),

cut = forcats::fct\_reorder(cut, carat)) %>%

ggplot(aes(cut,carat)) +

geom\_boxplot()

```

## Violin plot and boxplot using ggplot2. Use the package gridExtra to place both plots in the same display.

```{r}

require(gridExtra)

p1 <- ggplot(diamonds,aes(y=carat,x=cut)) + geom\_point() +

geom\_violin()

p2 <- ggplot(diamonds,aes(y=carat,x=cut)) + geom\_boxplot()

grid.arrange(p1, p2, ncol=2)

```

## Scatter plot

Plotting points of one ratio or interval variable against another is a scatter plot. Plots can be produced for a single or multiple pairs of variables. The purpose of a scatterplot is to see how one variable relates to another. With modeling in general the goal is parsimony (i.e., simple). The goal is to determine the fewest number of variables required to explain or describe a relationship. If two variables explain the same thing, i.e., they are highly correlated, only one variable is needed. The scatterplot provides a perfect visual reference for this.

## Create a scatterplot of carat and price with base graphics

```{r}

```

## Scatterplot without using formula

```{r}

```

## How many diamonds are 0.99 carats?

## How many are 1 carat?

## What do you think is the cause of the difference?

```{r}

```

# The difference could be an error in measurements. This could also possibly be because there is a premium for a 1-carat diamond, thus maybe there is a rounding up of the measurement.

## Scatterplot using ggplot2

```{r}

g <- ggplot(diamonds, aes(x=carat,y=price))

g + geom\_point(aes(color=color))

```

## Facet\_wrap and facet\_grid in ggplot2

```{r}

g + geom\_point(aes(color=color)) +

facet\_wrap(~color)

```

```{r}

g + geom\_point(aes(color=color)) +

facet\_grid(cut~clarity)

```

## Theme in ggplot2

There are seven other themes built in to ggplot2 1.1.0:

theme\_bw(): a variation on theme\_grey() that uses a white background and thin grey grid lines.

theme\_linedraw(): A theme with only black lines of various widths on white backgrounds, reminiscent of a line drawing.

theme\_light(): similar to theme\_linedraw() but with light grey lines and axes, to direct more attention towards the data.

theme\_dark(): the dark cousin of theme\_light(), with similar line sizes but a dark background. Useful to make thin coloured lines pop out.

theme\_minimal(): A minimalistic theme with no background annotations.

theme\_classic(): A classic-looking theme, with x and y axis lines and no gridlines.

theme\_void(): A completely empty theme.

```{r}

g2 <- ggplot(diamonds, aes(x=carat,y=price)) + geom\_point(aes(color=color))

## Lets apply few themes

p1 <- g2 + theme\_economist() + scale\_color\_economist()

p2 <- g2 + theme\_excel() + scale\_color\_excel()

p3 <- g2 + theme\_tufte()

p4 <- g2 + theme\_wsj()

grid.arrange(p1, p2, nrow=1, ncol=2)

grid.arrange(p3, p4, nrow=1, ncol=2)

#Notice how the different themes compare

```