Module 2

Descriptive Statistics through Visualisation

James Baglin

Last updated: 13 July, 2020

Overview

##Summary

Learning Objectives

The learning objectives associated with this module are:

- Define, compute and interpret statistics used for summarising qualitative variables.
- Use technology to visualise summaries of qualitative data using common statistical plots, namely, bar charts and clustered bar charts.
- Interpret common visualisations of qualitative data.
- Define, compute and interpret statistics used for summarising quantitative variables.
- Use technology to visualise summaries of quantitative data using common statistical plots, namely, dot plots, histograms, and box plots.
- Use scatter plots to visualise the relationship between two quantitative variables.
- Understand and apply the basic principles of producing good plots.

Module Video

Why You Need to Study Statistics

Descriptive Statistics

Descriptive statistics are methods by which complex, noisy or vast amounts of data can be converted into insightful, bite-sized, pieces of usable information. Descriptive statistics summarise characteristics of data using numbers. These include things like the mean, range, mode or percentage. Statistical visualisations are visual displays of descriptive statistics or data, most commonly graphs or plots, that summarise important features or trends. Visualisations make use of our visual sense to assist with interpreting data beyond reading tables of numbers from a page. Visualisations offer a more exciting and pleasing way to summarise data. The following sections will introduce common visualisations used in statistics to summarise difference types of variables. Often these visualisations report descriptive statistics. Therefore, these visualisations will be used to introduce common descriptive statistics.

This module is for the romantics. We will consider a large dataset of around 54,000 diamond prices and characteristics, `Diamonds` (data/Diamonds.csv). Before popping the question to your significant other, ensure you know a little about diamonds… otherwise, you might be in for some heart ache. If you're to be on the receiving end of a proposal, you probably need to know more! We will revisit this awesome dataset a few times in the course.

Here's some information about the dataset, `Diamonds` . The dataset includes the prices and other attributes of almost 54,000 diamonds. The variables are as follows:

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

This module will also show you how to use R to summarise and plot your data. Open RStudio, load the Diamonds data (ensure you call the object `Diamonds`) and reproduce the R code and output below to get further practice with using R. Ensure you save your R script so you can come back to it later. Also ensure you define your factors correctly. The following code ensures that cut, colour and clarity variables are treated correctly as ordered factors.

```
Diamonds$cut<- factor(Diamonds$cut, levels=c('Fair','Good','Very Good','Premium','Ideal'),
                      ordered=TRUE)

Diamonds$color<- factor(Diamonds$color, levels=c('J','I','H','G','F','E','D'),
                        ordered=TRUE)


Diamonds$clarity<- factor(Diamonds$clarity,
                          levels=c('I1','SI2','SI1','VS2','VS1','VVS2','VVS1','IF'),
                           ordered=TRUE)
```

Here is a random sample of the full dataset:

Show 10 ▼ **entries** **Search:**

carat	cut	color	clarity	depth	table	price	x

1	1.23	Ideal	G	VVS2	62.2	56.1	9921	6.8
2	1.02	Premium	H	SI1	62	60	4796	6.37
3	0.71	Very Good	I	SI2	62.2	58	1902	5.61
4	0.54	Ideal	E	VS2	61.9	55	1754	5.2
5	0.73	Ideal	D	VVS2	62	53.6	4547	5.76
6	1.01	Ideal	E	SI1	62.9	56	5375	6.41
7	1.02	Premium	D	SI2	60	62	4381	6.59
8	1.01	Premium	G	VS1	60.3	58	6787	6.56
9	1.01	Very Good	I	VS1	59.7	58	4858	6.53
10	0.33	Ideal	D	VS2	61.8	55	692	4.43

Showing 1 to 10 of 100 entries

Previous 1 2 3 4 5 … 10 Next

54,000 is a lot of data. The only way we can get some useful information from a dataset of this size will be to use descriptive statistics. Let’s start with looking at summarising some qualitative variables.

Qualitative Variables

Frequency Distributions

Cut quality is a qualitative variable, measured on an ordinal scale, ranging from fair to ideal. A frequency table can be used to tally/count the number of times a particular cut quality appears in the dataset. We can use the `table()` function to generate a basic frequency distribution.

```
library(dplyr) # Required for data management and pipes
Diamonds$cut %>% table()
```

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

Reading the output, we find the “ideal”, 21,551, cut is the most frequently occurring value. The most frequently occurring qualitative value for a variable is called the **mode**. There, we just covered two types of descriptive statistics in no time.

Counts or tallies don't allow comparison between different datasets. For example, what if we wanted to compare the counts of the Diamonds variable to another smaller dataset? We need to use proportions, f/n, where f = the count or frequency or a value, and n = sample size. We can also readily convert proportions to percentages using $f/n * 100$. I'll assume we're all intimately familiar with percentages. You only have to walk into any shopping centre to be bombarded…

In R, we can report the proportions for cuts using:

```
Diamonds$cut %>% table() %>% prop.table()
```

```
## .
##       Fair       Good  Very Good    Premium      Ideal
## 0.02984798 0.09095291 0.22398962 0.25567297 0.39953652
```

or, percentages:

```
Diamonds$cut %>% table() %>% prop.table()*100
```

```
## .
##      Fair      Good Very Good   Premium     Ideal
##  2.984798  9.095291 22.398962 25.567297 39.953652
```

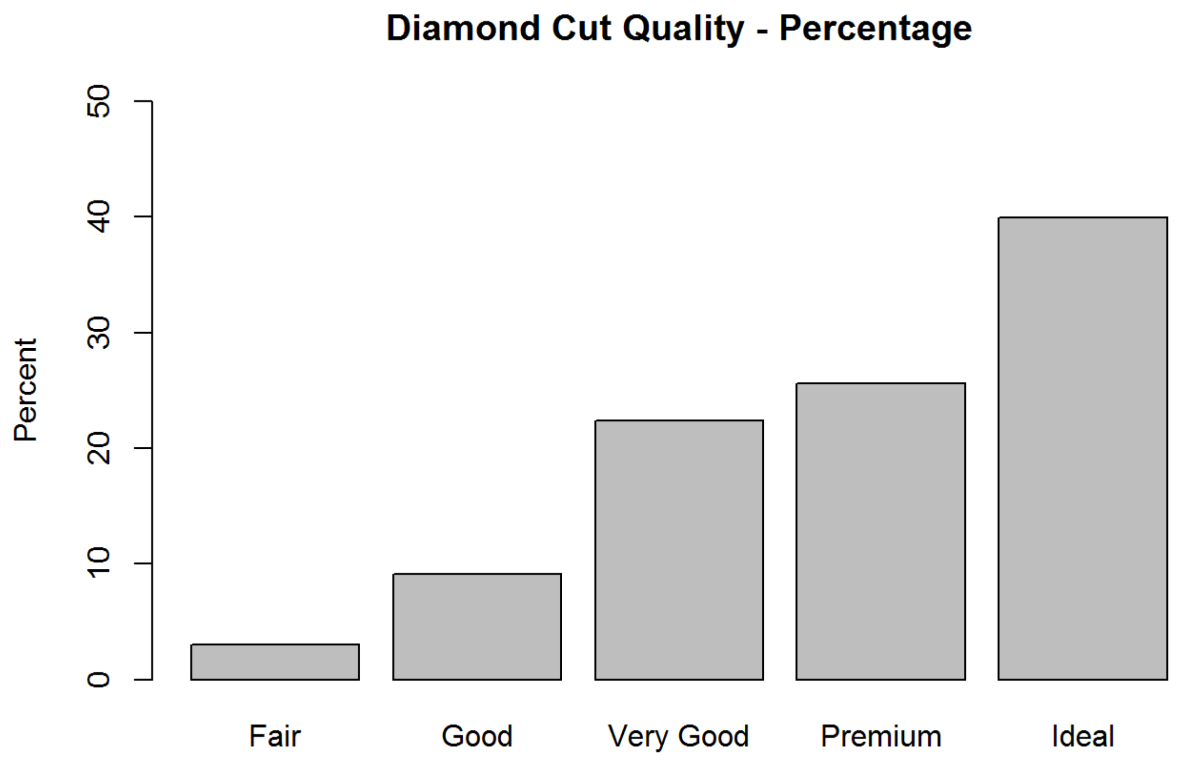
Bar Charts

A simple and effective visualisation of qualitative data is the humble bar chart. Here's how to get one in R. First we assign the table object name. Let' asign the percentages to an object named `perc`.

```
perc <- Diamonds$cut %>% table() %>% prop.table()*100
```

Now, call the bar plot using the following function. Note how the code defines a title, x axis label and sets the height of the y axis:

```
perc %>% barplot(main = "Diamond Cut Quality - Percentage",ylab="Percent"
  , ylim=c(0,50))
```



The height of each bar represents the percentage of cut quality in the data set. This is much quicker to interpret than written tallies. Bars can also represent raw counts/tallies or proportions. Ensure you label your axes correctly to help the reader interpret your scales. It's also a good idea to start your y-axis at 0, so as not to distort the scale and to allow relative comparisons across levels.

Contingency Tables

When we need to explore the relationship between two categorical variables, we can create contingency tables, also known as cross-tabulations. These tables present one categorical variable as the rows and the other categorical variable as the columns. These tables make it easy for us to calculate conditional probabilities or percentages, which makes it easier for us to explore potential associations between variables. In the following example, we will consider the relationship between the cut of a diamond and its clarity. To get the contingency table for raw counts, we use:

```
table(Diamonds$cut,Diamonds$clarity)
```

```
##
##               I1  SI2  SI1  VS2  VS1 VVS2 VVS1   IF
##   Fair       210  466  408  261  170   69   17    9
##   Good        96 1081 1560  978  648  286  186   71
##   Very Good   84 2100 3240 2591 1775 1235  789  268
##   Premium    205 2949 3575 3357 1989  870  616  230
##   Ideal      146 2598 4282 5071 3589 2606 2047 1212
```

However, because there are differences between the row and column totals for each category, it makes looking at trends difficult. We can calculate the conditional column percentages using the following code. Notice how the `round()` function has been included to reduce the size of the probabilities in the table cells. Otherwise, R will round to six decimal places and the table will be too large.

```
table(Diamonds$cut,Diamonds$clarity) %>% prop.table(margin = 2) %>% round
  (3)
```

```
##
##                I1   SI2   SI1   VS2   VS1  VVS2  VVS1    IF
##   Fair      0.283 0.051 0.031 0.021 0.021 0.014 0.005 0.005
##   Good      0.130 0.118 0.119 0.080 0.079 0.056 0.051 0.040
##   Very Good 0.113 0.228 0.248 0.211 0.217 0.244 0.216 0.150
##   Premium   0.277 0.321 0.274 0.274 0.243 0.172 0.169 0.128
##   Ideal     0.197 0.283 0.328 0.414 0.439 0.514 0.560 0.677
```

These are conditional column probabilities, determined by `margin = 2`, because if we add all the probabilities for a column together, they will equal 1, e.g. sum(0.283, 0.130, 0.113, 0.227, 0.197) = 1.00. Let me prove it:

```
tbl <- table(Diamonds$cut,Diamonds$clarity) %>% prop.table(margin = 2) %
  >% round(3)
tbl[,1]
```

```
##      Fair      Good Very Good   Premium     Ideal
##     0.283     0.130     0.113     0.277     0.197
```

```
tbl[,1] %>% sum()
```

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

If we had set `margin = 1` we would get row proportions:

```
table(Diamonds$cut,Diamonds$clarity) %>% prop.table(margin = 1) %>% round
  (3)
```

```
##
##                I1   SI2   SI1   VS2   VS1  VVS2  VVS1    IF
##   Fair      0.130 0.289 0.253 0.162 0.106 0.043 0.011 0.006
##   Good      0.020 0.220 0.318 0.199 0.132 0.058 0.038 0.014
##   Very Good 0.007 0.174 0.268 0.214 0.147 0.102 0.065 0.022
##   Premium   0.015 0.214 0.259 0.243 0.144 0.063 0.045 0.017
##   Ideal     0.007 0.121 0.199 0.235 0.167 0.121 0.095 0.056
```

If we had left `margin` blank, we would get cell proportions:

```
table(Diamonds$cut,Diamonds$clarity) %>% prop.table() %>% round(3)
```

```
##
##                I1   SI2   SI1   VS2   VS1  VVS2  VVS1    IF
##   Fair      0.004 0.009 0.008 0.005 0.003 0.001 0.000 0.000
##   Good      0.002 0.020 0.029 0.018 0.012 0.005 0.003 0.001
##   Very Good 0.002 0.039 0.060 0.048 0.033 0.023 0.015 0.005
##   Premium   0.004 0.055 0.066 0.062 0.037 0.016 0.011 0.004
##   Ideal     0.003 0.048 0.079 0.094 0.067 0.048 0.038 0.022
```

Summing all the cells together would now equal 1. We will spend more time in Module 3 looking at the interpretation of large contingency tables. Now, let's interpret the column probabilities. Recall:

```
table(Diamonds$cut,Diamonds$clarity) %>% prop.table(margin = 2) %>% round
  (3)
```

```
##
##                I1   SI2   SI1   VS2   VS1  VVS2  VVS1    IF
##   Fair      0.283 0.051 0.031 0.021 0.021 0.014 0.005 0.005
##   Good      0.130 0.118 0.119 0.080 0.079 0.056 0.051 0.040
##   Very Good 0.113 0.228 0.248 0.211 0.217 0.244 0.216 0.150
##   Premium   0.277 0.321 0.274 0.274 0.243 0.172 0.169 0.128
##   Ideal     0.197 0.283 0.328 0.414 0.439 0.514 0.560 0.677
```

Let's contrast the worst clarity I1 with the best, IF (Flawless). For I1, we find that the probability for a fair, good, very good, premium and ideal cut are 0.283, 0.130, 0.113, 0.227, and 0.197 respectively. However, for IF, the same probabilities are 0.005, 0.040, 0.150, 0.128 and 0.677 respectively. We can see that IF diamonds are far more likely to have superior cuts, which makes perfect sense. You don't give the best diamonds to the cutting apprentice!

Interpreting associations and trends from large contingency tables such as this can take a considerable amount of brain power. We can make this process easier and more enjoyable using a visualisation. We can use an extension of the bar chart for this purpose.

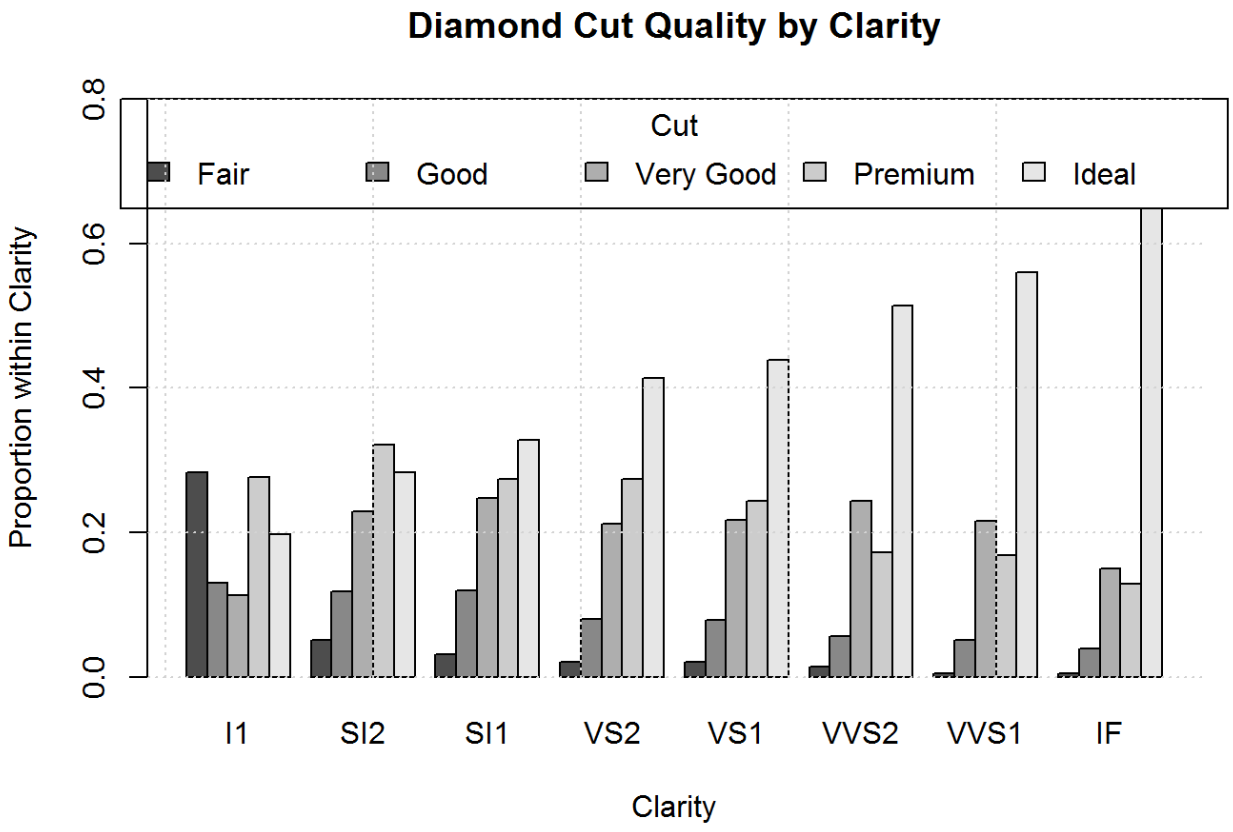
Clustered Bar Charts

Clustered bar charts are a great way to visualise two qualitative variables. Let's plot the conditional column proportions of the cut by clarity contingency table using R. First, we need to save the conditional proportions in an object called `table_1` . This will make it easy for us to create the clustered bar chart.

```
table_1 <- table(Diamonds$cut,Diamonds$clarity) %>% prop.table(margin = 2
  )
```

Next we can plot the probabilities. Note the various options used to change the plot title, y axis label, y axis range, legend and x axis label.

```
table_1 %>% barplot(main = "Diamond Cut Quality by Clarity", ylab="Propor
  tion within Clarity",
                        ylim=c(0,.8), legend=rownames(table_1), beside=TRUE,
                        args.legend=c(x = "top", horiz=TRUE, title="Cut"),
                        xlab="Clarity")
grid()
```



Notice how `grid()` was added after the plot was produced in R. Grid lines can help the viewer quickly read off and compare values on the plot axes.

Looking at this clustered bar chart, it becomes quickly apparent that as the clarity of a diamond increases, the quality of the cut also tends to increase. Perfect! See how simple statistics can be. You just interpreted your first categorical association. We will dig deeper into categorical associations again in Module 3 and 8.

Quantitative Variables

Quantitative variables require different types of statistical summaries and visualisations. Let's start with a small sample of the diamond data to make the calculations manageable, and then we will scale-up to the full 54,000 for some awesome plotting later in the module.

The `Diamonds_sample.csv` (\data\Diamonds_sample.csv) dataset contains a small random sample of 30 diamonds' mass measured in carats.

```
Diamonds_sample <- read.csv("data/Diamonds_sample.csv")
```

Here are the data:

Show 30 ▼ **entries** **Search:**

	carat	cut	color	clarity	depth	table	price	x
1	0.34	Very Good	H	VS2	61.9	56	537	4.46
2	1	Very Good	D	SI1	59.3	61	5987	6.45
3	1.58	Very Good	G	VS2	59.6	58	13037	7.59
4	1.1	Premium	F	SI1	59.5	59	5821	6.77
5	0.54	Premium	D	VS2	61.6	55	1913	5.28
6	0.4	Premium	E	SI1	62.2	62	882	4.69
7	1	Good	E	SI1	61.6	62	5784	6.35
8	1.04	Very Good	I	SI1	62.5	62	4175	6.39
9	0.41	Very Good	G	SI2	63.4	58	784	4.73
10	0.31	Very Good	E	SI1	63.4	57	698	4.33
11	0.7	Good	G	VS1	60.6	61	2591	5.71
12	0.5	Very Good	H	VS2	63.9	57	1387	5.04

13	0.32	Ideal	F	VS1	61.2	56.2	695	4.41
14	0.43	Very Good	F	SI1	62.5	56	792	4.82
15	0.31	Ideal	D	VS2	62.2	56	942	4.33
16	2.06	Premium	G	SI2	59.8	61	15949	8.26
17	1.04	Ideal	I	VS1	61.8	56	5229	6.49
18	2.38	Premium	H	SI1	58.2	60	16643	8.81
19	0.7	Good	F	SI1	63.7	58	2242	5.62
20	1.01	Premium	G	VS1	62.7	58	6618	6.41
21	0.33	Ideal	G	VS2	61.2	55	579	4.46
22	1	Good	H	SI1	63.8	53	4596	6.39
23	0.91	Premium	H	VS2	62.5	59	4018	6.19
24	2.08	Ideal	J	IF	61	55	17986	8.32
25	0.44	Very Good	D	VS2	62.8	58	850	4.79
26	1.5	Ideal	H	VS1	61	56.8	11557	7.36
27	0.56	Ideal	D	SI1	62.3	56	1723	5.26
28	0.32	Ideal	D	VS2	62.6	55	972	4.39
29	1.11	Ideal	H	SI2	61.2	56	5496	6.7
30	0.9	Good	I	VS1	63.3	59	3644	6.06

Showing 1 to 30 of 30 entries

Previous 1 Next

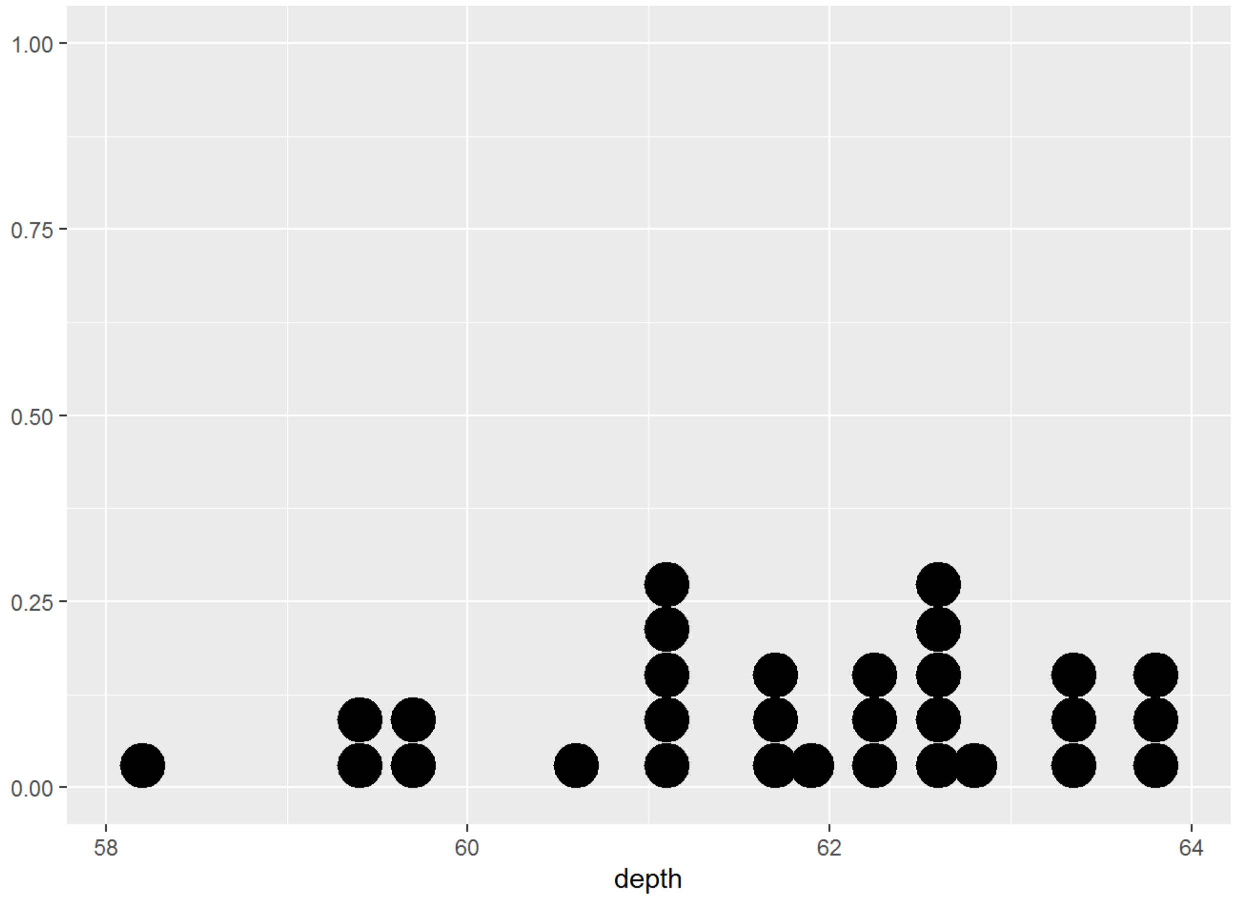
Dot Plots

Dot plots are a nice visual representation of small quantitative datasets. Each dot is arranged into bins on the x-axis and stacked on top of each other to report the frequency. Dot plots represent the distribution of a quantitative variable, and granularity of the small sample. We can use dot plots to quickly see where values are clustering and tending towards, as well as unusual cases known as outliers. To get a dot plot in R, you first must install and load the `ggplot2` package.

```
install.packages("ggplot2")
library(ggplot2)
```

We can use a `ggplot2` quick plot function to produce the dot plot.

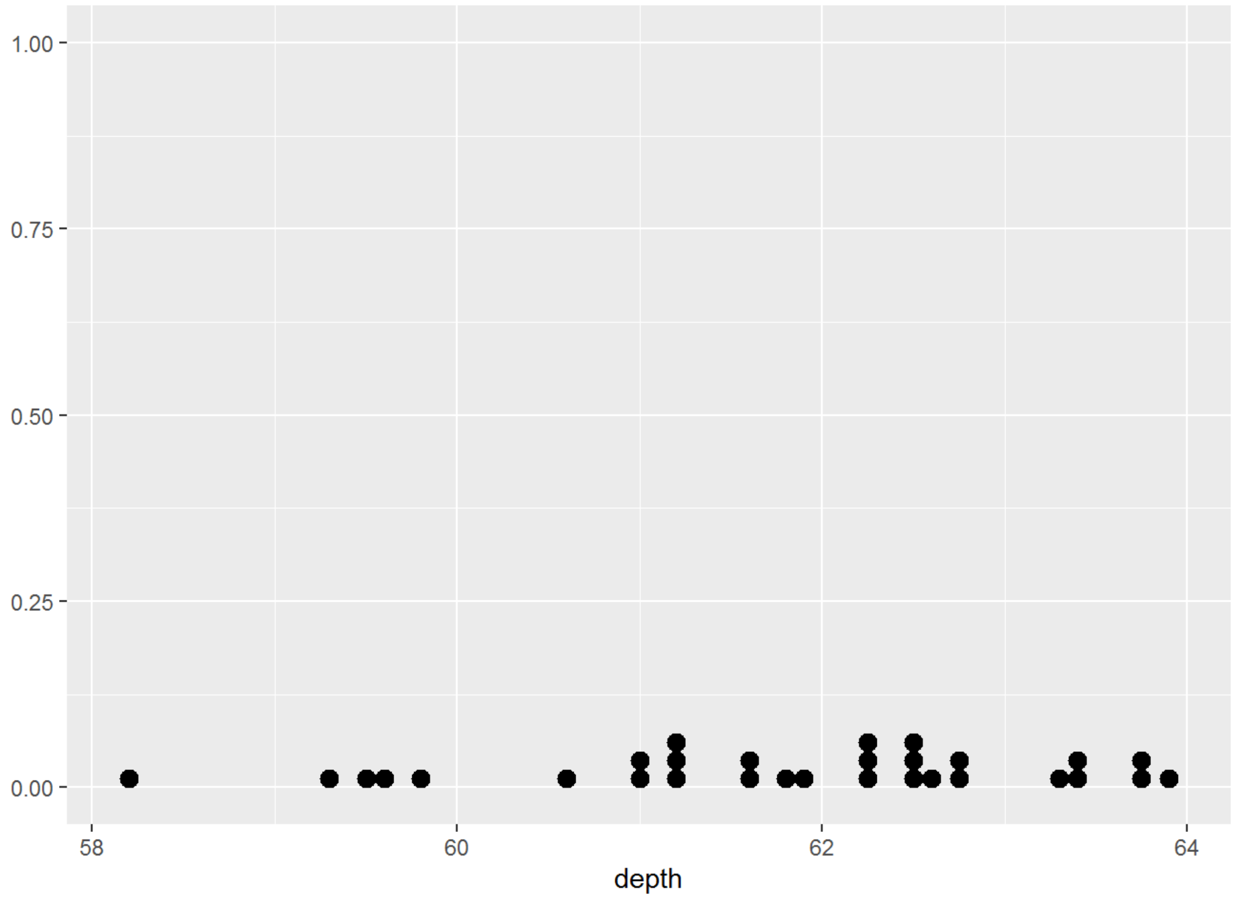
```
Diamonds_sample %>% qplot(data = ., x = depth, geom = "dotplot", binwidth
  = .25)
```



The dot plot of 30 random diamond depths are arranged into bins with widths of .25 mm. For example, three values were between 63.75 mm and 64mm, five values between 61 and 61.25, etc. It’s a little hard to tell, which is where we encounter some of the issues with dot plots. The main drawback of dot plots is that they don’t show the actual values. So, we can’t tell exactly what they are. They are also sensitive to the number of bins or internals used in the plot. Setting the right bin widths is tricky.

Let’s set a larger number of intervals, or bins, in the plot, by making the intervals smaller.

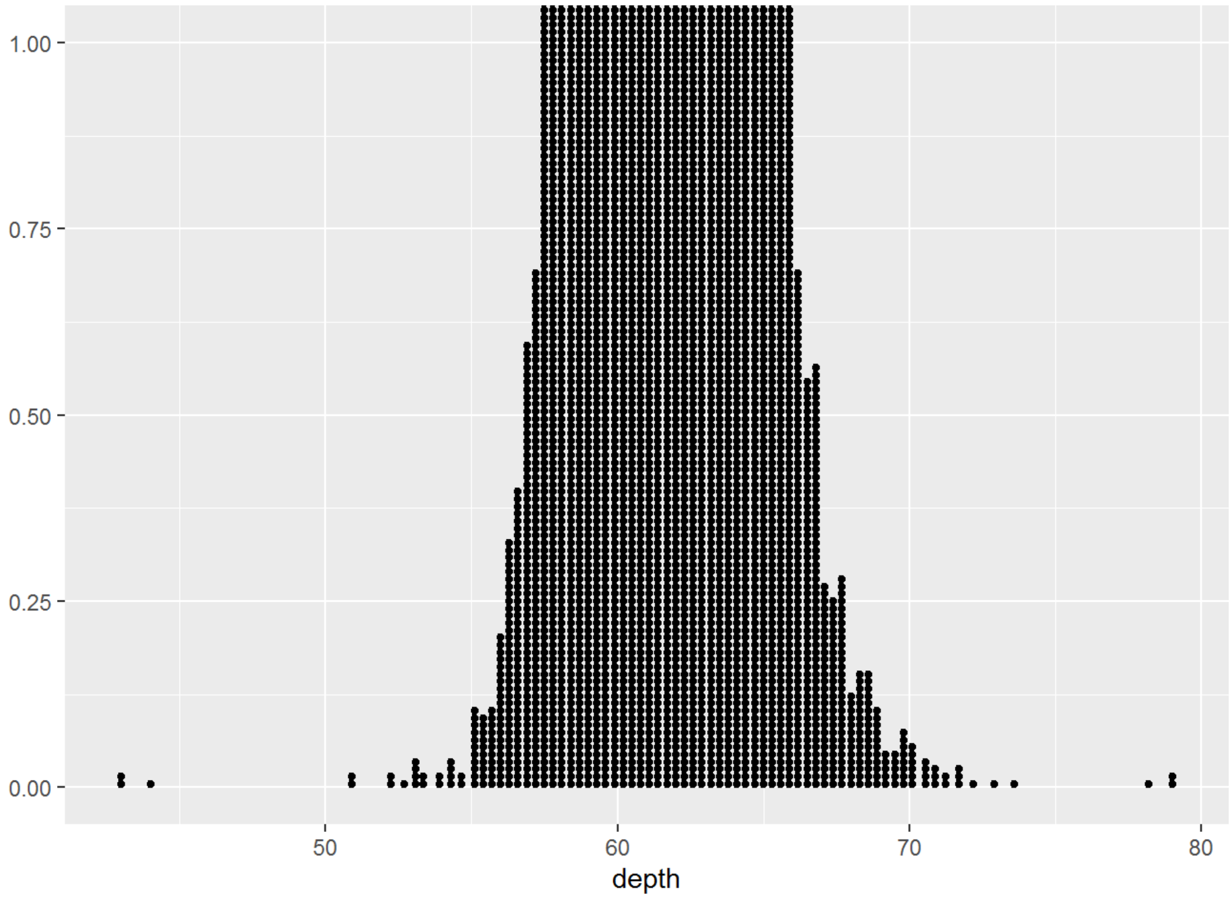
```
Diamonds_sample %>% qplot(data = ., x = depth, geom = "dotplot", binwidth
  = .1)
```



By increasing the number of internals or bins to `nint=10`, the dot plot drastically changes appearance. This is a drawback to visualising small samples. Often the choice of parameters used in the plot can change interpretations. Be mindful.

They are also problematic for large datasets…

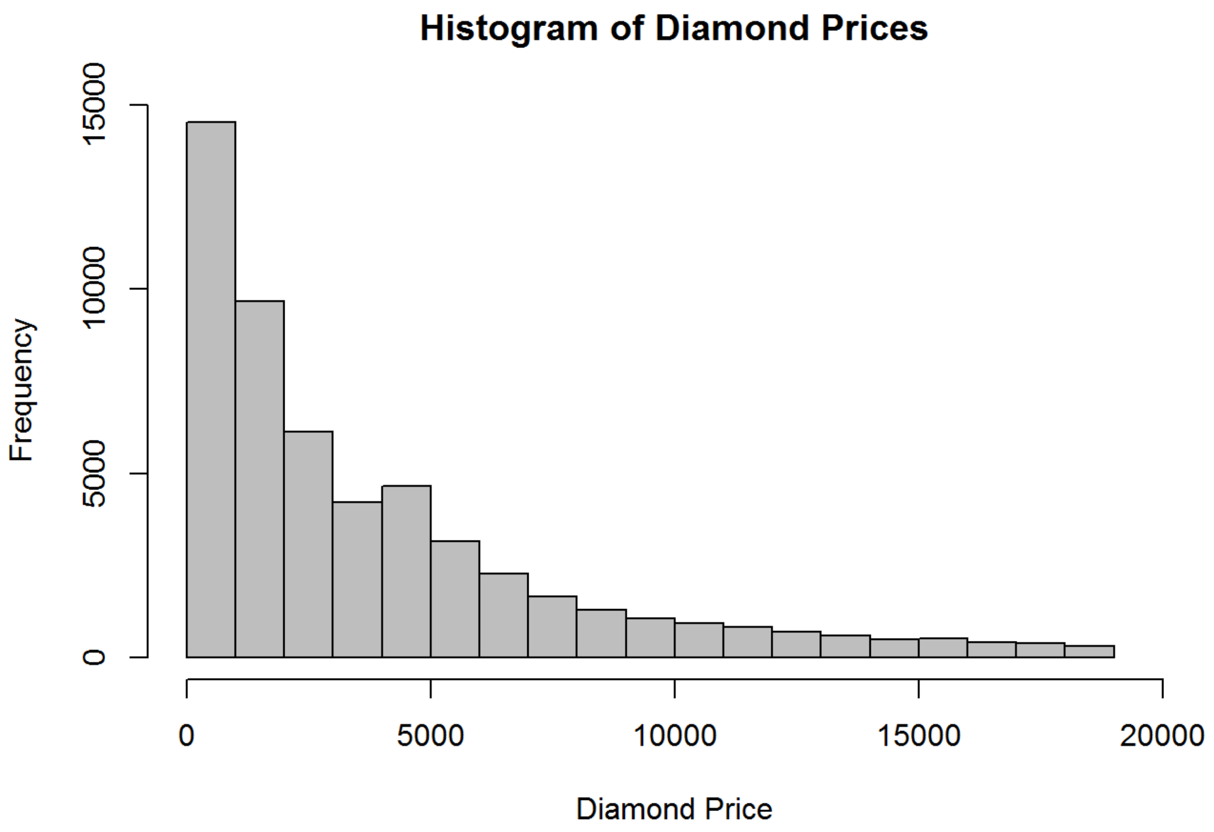
```
Diamonds %>% qplot(data = ., x = depth, geom = "dotplot", binwidth = .25)
```



Histograms

When dealing with large amounts of quantitative data, we can use histograms to explore the distributions of interval and ratio data. Let’s use R to create some histograms of diamond prices:

```
Diamonds$price %>%  hist(col="grey",xlim=c(0,20000),
                         xlab="Diamond Price",
                         main="Histogram of Diamond Prices")
```



Take note of how the code limits the x axis to values between $0 and $20,000, we have also set the colour of the bars to grey and a plot title.

Histograms break the data into bins, or intervals, depicted using bars, where the height of the bar typically refers to the frequency. Due to the large amount of data, the bars are joined together to form a continuous wall. This is the drawback of using histograms for small samples. The continue wall of bars can give the viewer a false sense of the data's density. Dot plots are much better for small samples as each dot depicts the granularity of the data. The histogram of prices is interesting. We can quickly glean the following:

- Prices range goes all the way up to $19,000 U.S.
- Most diamond prices are below $5,000
- The most common diamond prices are between 0 to $1,000.

The price distribution is an example of what we would call a positively, or right skewed, distribution. You will see examples of other distributions shortly.

We can use R to find the bin intervals used to construct the plot:

```
bins <- Diamonds$price %>% hist()
```

```
bins$breaks
```

```
##  [1]     0  1000  2000  3000  4000  5000  6000  7000  8000  9000 10000
11000
## [13] 12000 13000 14000 15000 16000 17000 18000 19000
```

```
bins$counts
```

```
##  [1] 14524  9683  6129  4225  4665  3163  2278  1668  1307  1076   934
825
## [13]   701   603   504   513   425   405   312
```

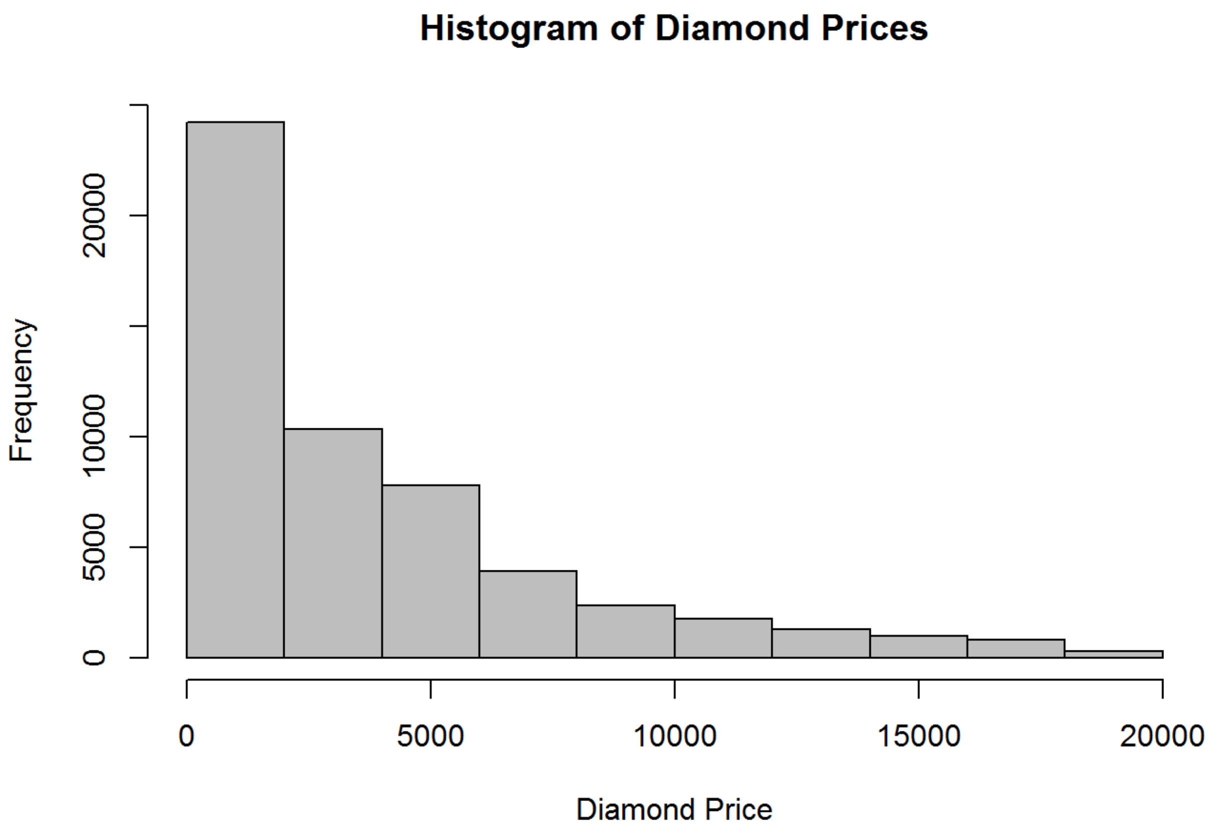
Using this information, we can create the table to the below which explains how the data were plotted in the histogram above.

```
binstable <- data.frame(Breaks = bins$breaks, Counts = c(0,bins$counts))
binstable
```

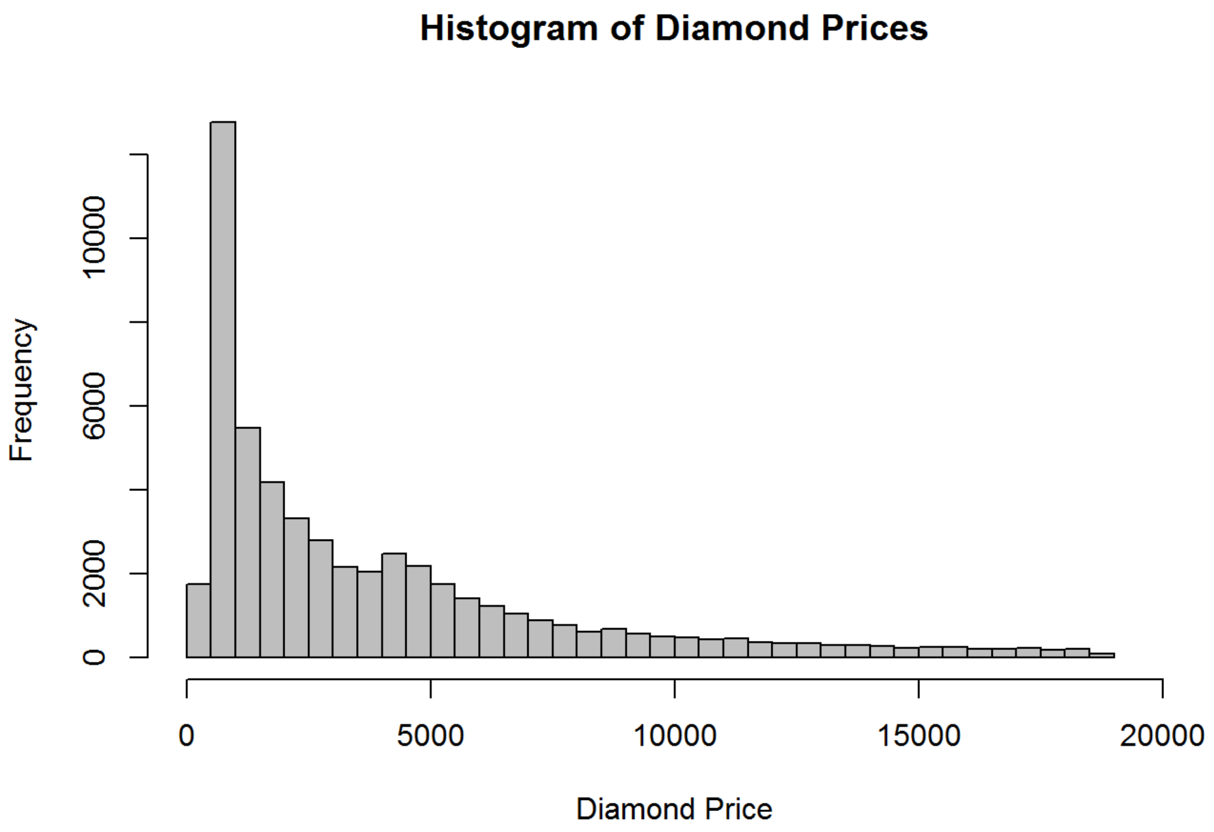
```
##    Breaks Counts
## 1       0      0
## 2    1000  14524
## 3    2000   9683
## 4    3000   6129
## 5    4000   4225
## 6    5000   4665
## 7    6000   3163
## 8    7000   2278
## 9    8000   1668
## 10   9000   1307
## 11  10000   1076
## 12  11000    934
## 13  12000    825
## 14  13000    701
## 15  14000    603
## 16  15000    504
## 17  16000    513
## 18  17000    425
## 19  18000    405
## 20  19000    312
```

Even with histograms and large data, we must be careful with how the choice of bins can change the appearance of a histogram. Consider the following histogram to see how much change can occur when reducing or increasing the number of bins from the default values.

```
Diamonds$price %>% hist(col="grey", xlim=c(0,20000), xlab="Diamond Price"
  ,
                        main="Histogram of Diamond Prices", breaks=10)
```



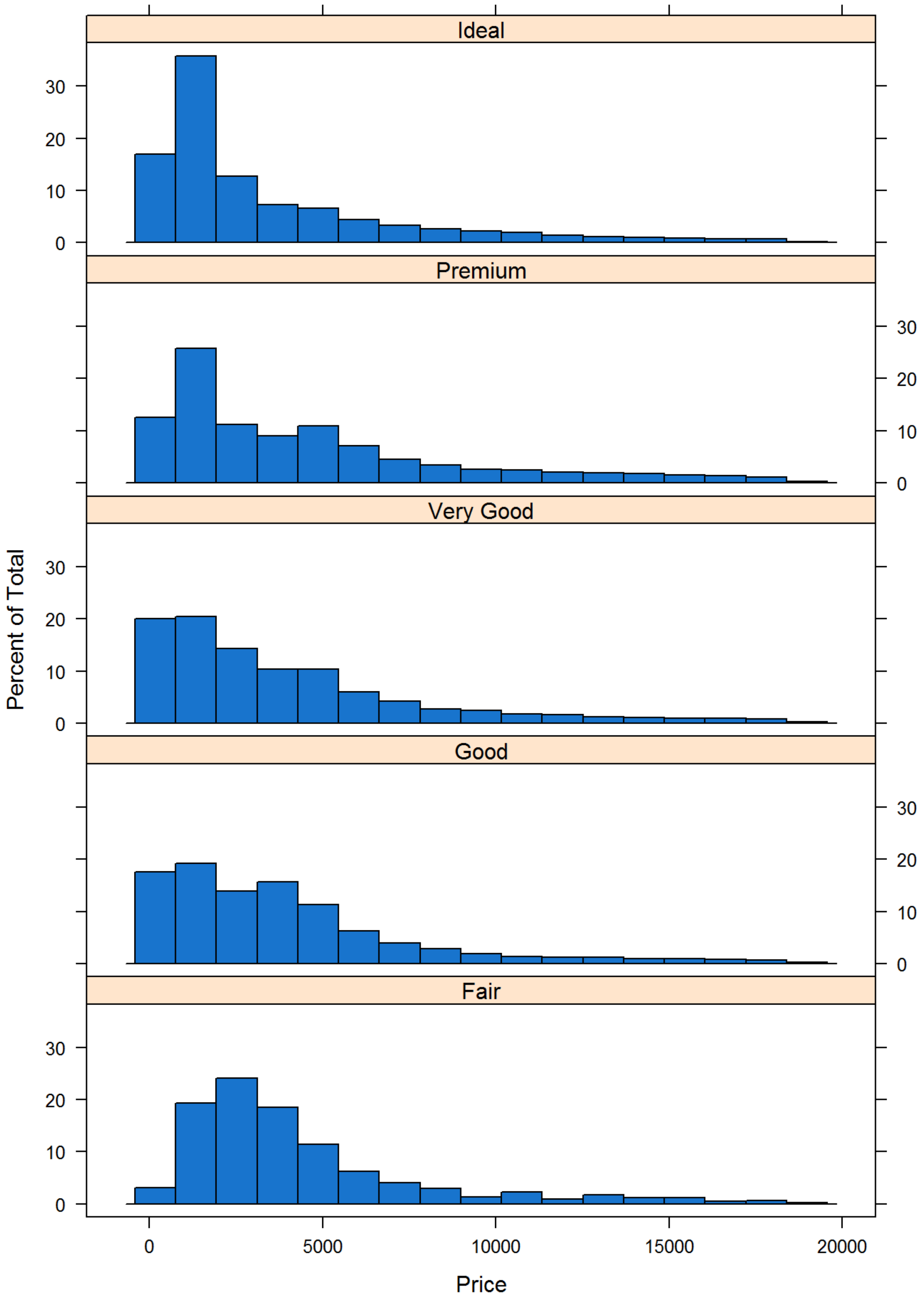
```
Diamonds$price %>% hist(col="grey", xlim=c(0,20000), xlab="Diamond Price"
  ,
                        main="Histogram of Diamond Prices", breaks=50)
```



Histograms are a great way to compare different distributions. We can panel a series of histograms together on a common scale to help comparisons. The following code panels together histograms of diamond prices across different cuts.

We have to use a slightly different `histogram()` function from the `lattice` package:

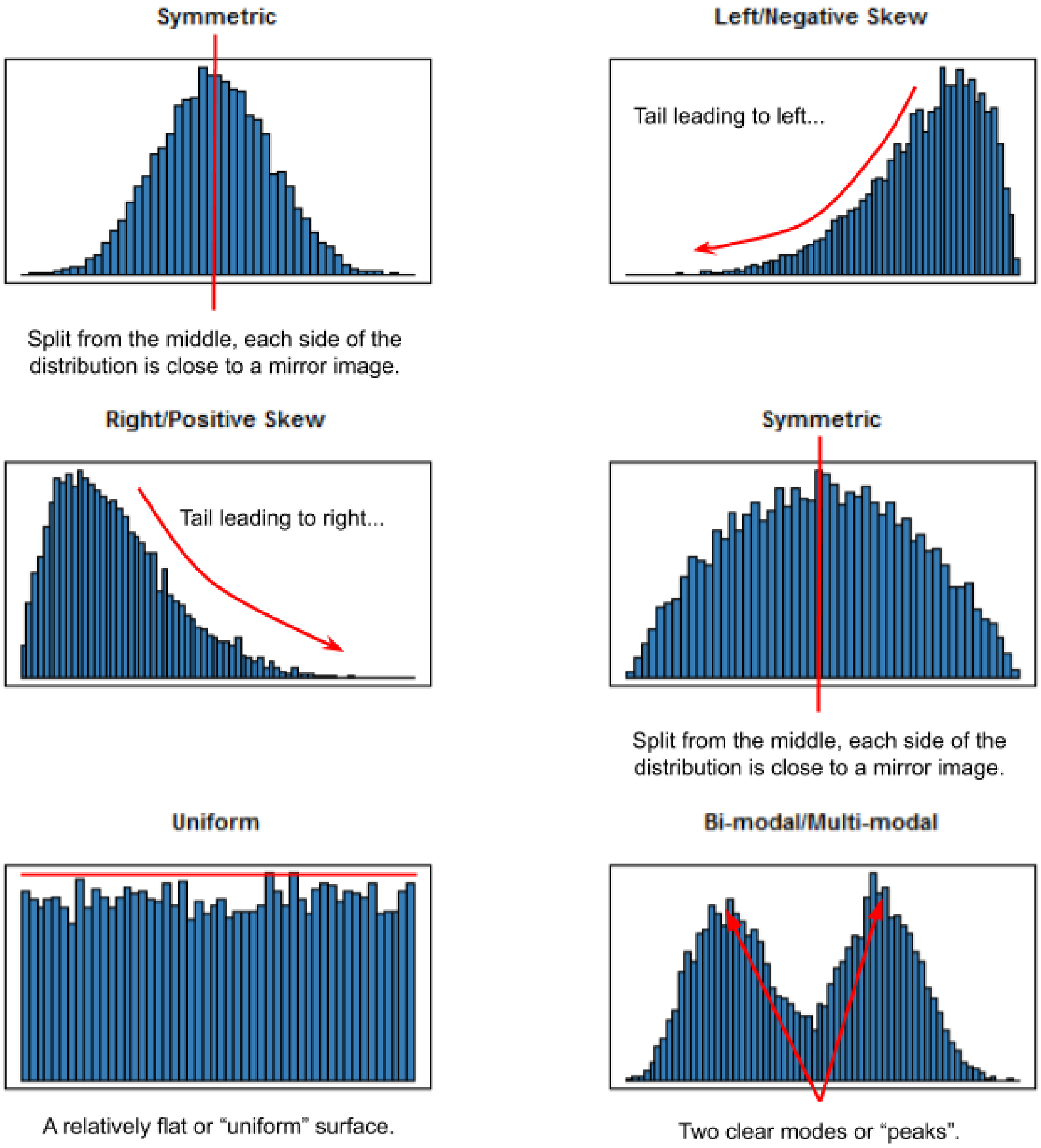
```
library(lattice)
Diamonds %>% histogram(~ price|cut, col="dodgerblue3",
                       layout=c(1,5), data=., xlab="Price")
```



As you can see, the histograms look largely the same and it's difficult to see a distinct relationship between cut and price. Cut, alone, is probably not a good indicator of price.

Distribution Shapes

Statisticians have an entire language dedicated to articulating the shape of quantitative variables' distributions. The following info-graphic introduces some of the common terms. These are important terms to understand, as the shape of a distribution often determines our choice of descriptive statistics and statistical tests. You will learn more as the course goes on.



Symmetric

Split from the middle, each side of the distribution is close to a mirror image.

Left/Negative Skew

Right/Positive Skew

Symmetric

Split from the middle, each side of the distribution is close to a mirror image.

Uniform

A relatively flat or "uniform" surface.

Bi-modal/Multi-modal

Two clear modes or "peaks".

Measures of Central Tendency

We've looked at dot plots and histograms for visualising quantitative variables. Now we will consider some basic descriptive statistics used to summarise the essential features of data distributions, including both measures of central tendency and variation. We will focus our attention on the mean and standard deviation.

Mean and Standard Deviation

Let's step back to the small sample of diamond data, `Diamonds_sample`, to calculate the mean depth of the data depicted in the dot plot. The mean of a dataset, $\bar{x}$ is calculated as:

$$\bar{x} = \frac{\sum_{n}^{i=1} x_i}{n}$$

where x_i is a sample value and n is the sample size. The formula is simply the sum of all the data points divided by the sample size. In R, we could code:

```
sum(Diamonds_sample$depth)/length(Diamonds_sample$depth)
```

```
## [1] 61.77667
```

Fortunately, R has statistical functions that make this easier.

```
Diamonds_sample$depth %>% mean()
```

```
## [1] 61.77667
```

The **mean** is a measure of central tendency that can be used to describe the average or typical value for a dataset. It takes all values into account and is therefore sensitive to all values in a dataset. This can create problems in skewed distributions or distributions with unusual or outlying values.

The mean is also a single measure of centre. It tells us nothing about spread or variation. Variation is the essence of statistics, so we need summary statistics that can convey variation. A crude measure of variation includes the range:

$$\text{Range} = \text{Max} - \text{Min}$$

The range is simply the maximum value of a dataset, minus the minimum value. Using R:

```
max(Diamonds_sample$depth)-min(Diamonds_sample$depth)
```

```
## [1] 5.7
```

```
range(Diamonds_sample$depth)
```

```
## [1] 58.2 63.9
```

The range is not very useful. It is based on only two data points and misses all the interesting stuff happening in between. A better indicator of variation is the sample variance, s^2 and sample standard deviation, s. The sample variance is calculated using the following formula:

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

You may also wonder why the equation divides by $n - 1$ and not just n. This is known as Bessel's correction (https://en.wikipedia.org/wiki/Bessel%27s_correction). A sample's variance is known to underestimate the population variance, σ^2. Dividing by $n - 1$ corrects for this bias.

The standard deviation is simply the square-root of the variance:

$$s = \sqrt{s^2}$$

To calculate this, we can use the `var()` and `sd()` functions:

```
Diamonds_sample$depth %>% var()
```

```
## [1] 2.106678
```

```
Diamonds_sample$depth %>% sd()
```

```
## [1] 1.45144
```

The following code walks through the computational steps of these two formulae. First we compute the variance, `var()`.

Calculate the deviations $x_i - \bar{x}$

```
dev <- Diamonds_sample$depth - mean(Diamonds_sample$depth)
sd.table <- data.frame(Depth = Diamonds_sample$depth,
                       Mean = mean(Diamonds_sample$depth), Deviation = de
  v)
sd.table
```

```
##    Depth     Mean   Deviation
## 1   61.9 61.77667  0.12333333
## 2   59.3 61.77667 -2.47666667
## 3   59.6 61.77667 -2.17666667
## 4   59.5 61.77667 -2.27666667
## 5   61.6 61.77667 -0.17666667
## 6   62.2 61.77667  0.42333333
## 7   61.6 61.77667 -0.17666667
## 8   62.5 61.77667  0.72333333
## 9   63.4 61.77667  1.62333333
## 10  63.4 61.77667  1.62333333
## 11  60.6 61.77667 -1.17666667
## 12  63.9 61.77667  2.12333333
## 13  61.2 61.77667 -0.57666667
## 14  62.5 61.77667  0.72333333
## 15  62.2 61.77667  0.42333333
## 16  59.8 61.77667 -1.97666667
## 17  61.8 61.77667  0.02333333
## 18  58.2 61.77667 -3.57666667
## 19  63.7 61.77667  1.92333333
## 20  62.7 61.77667  0.92333333
## 21  61.2 61.77667 -0.57666667
## 22  63.8 61.77667  2.02333333
## 23  62.5 61.77667  0.72333333
## 24  61.0 61.77667 -0.77666667
## 25  62.8 61.77667  1.02333333
## 26  61.0 61.77667 -0.77666667
## 27  62.3 61.77667  0.52333333
## 28  62.6 61.77667  0.82333333
## 29  61.2 61.77667 -0.57666667
## 30  63.3 61.77667  1.52333333
```

Now square the deviations, $(x_i - \bar{x})^2$:

```
dev2 <- dev^2
sd.table$DevSq <- dev2
sd.table
```

```
##    Depth     Mean   Deviation        DevSq
## 1   61.9 61.77667  0.12333333 1.521111e-02
## 2   59.3 61.77667 -2.47666667 6.133878e+00
## 3   59.6 61.77667 -2.17666667 4.737878e+00
## 4   59.5 61.77667 -2.27666667 5.183211e+00
## 5   61.6 61.77667 -0.17666667 3.121111e-02
## 6   62.2 61.77667  0.42333333 1.792111e-01
## 7   61.6 61.77667 -0.17666667 3.121111e-02
## 8   62.5 61.77667  0.72333333 5.232111e-01
## 9   63.4 61.77667  1.62333333 2.635211e+00
## 10  63.4 61.77667  1.62333333 2.635211e+00
## 11  60.6 61.77667 -1.17666667 1.384544e+00
## 12  63.9 61.77667  2.12333333 4.508544e+00
## 13  61.2 61.77667 -0.57666667 3.325444e-01
## 14  62.5 61.77667  0.72333333 5.232111e-01
## 15  62.2 61.77667  0.42333333 1.792111e-01
## 16  59.8 61.77667 -1.97666667 3.907211e+00
## 17  61.8 61.77667  0.02333333 5.444444e-04
## 18  58.2 61.77667 -3.57666667 1.279254e+01
## 19  63.7 61.77667  1.92333333 3.699211e+00
## 20  62.7 61.77667  0.92333333 8.525444e-01
## 21  61.2 61.77667 -0.57666667 3.325444e-01
## 22  63.8 61.77667  2.02333333 4.093878e+00
## 23  62.5 61.77667  0.72333333 5.232111e-01
## 24  61.0 61.77667 -0.77666667 6.032111e-01
## 25  62.8 61.77667  1.02333333 1.047211e+00
## 26  61.0 61.77667 -0.77666667 6.032111e-01
## 27  62.3 61.77667  0.52333333 2.738778e-01
## 28  62.6 61.77667  0.82333333 6.778778e-01
## 29  61.2 61.77667 -0.57666667 3.325444e-01
## 30  63.3 61.77667  1.52333333 2.320544e+00
```

Now sum the squared deviations, $\sum_{i=1}^{n}(x_i - \bar{x})^2$

```
sumdev2 <- sd.table$DevSq %>% sum()
sumdev2
```

```
## [1] 61.09367
```

Now divide by $n - 1$ (Bessel's correction):

```
variance <- sumdev2/(length(Diamonds_sample$depth)-1)
variance
```

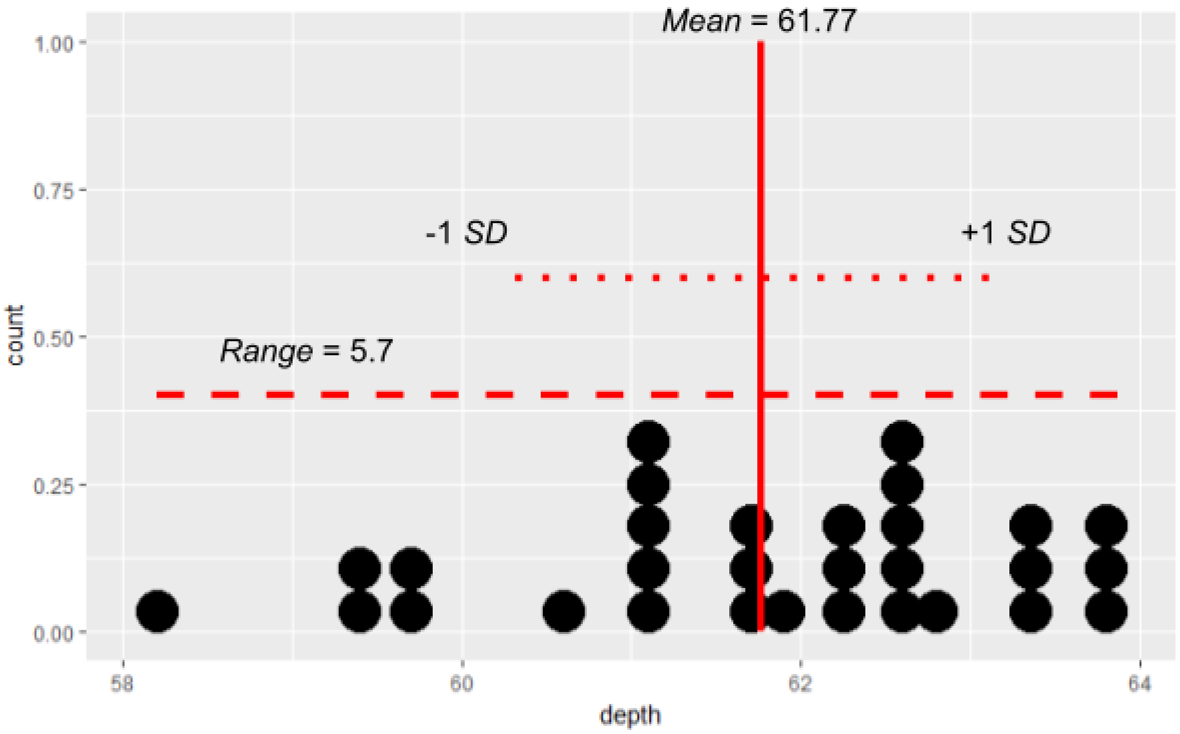
```
## [1] 2.106678
```

Correct! Square root the variance, $\sqrt{s^2}$ to get the sample standard deviation, s:

```
variance %>% sqrt()
```

```
## [1] 1.45144
```

Correct again! The variance for the depth data was 1.45. As the variance represents an averaged squared value, it is unit-less. The standard deviation corrects this by taking the square root of the variance in order to convert it back to its original scale. The standard deviation uses all the values in a dataset and is therefore a much more useful indicator of variability. The standard deviation represents the average deviation from the mean. For example, we could state that diamond depth percentages variaed by 1.45% on average. In other words, very littl variation. The mean of the depth data was calculated to be 61.78. The following info-graphic depicts the mean, range and standard deviation for diamond depth data according to a dot plot.



Now let’s calculate the mean and standard deviation for `price` in the complete `Diamond` dataset. We will short cut this using the `summary()` function.

```
Diamonds$price %>% summary()
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     326     950    2401    3933    5324   18823
```

This is missing the standard deviation. We can use the `summarise()` function from the `dplyr` package to produce a similar summary. There are many other packages that make this easier, however, I believe learning to use `dplyr` to build these tables gives you far more flexibility and prevents the need to install another package. I'll prove this shortly. Let's have a look at a basic summarised table.

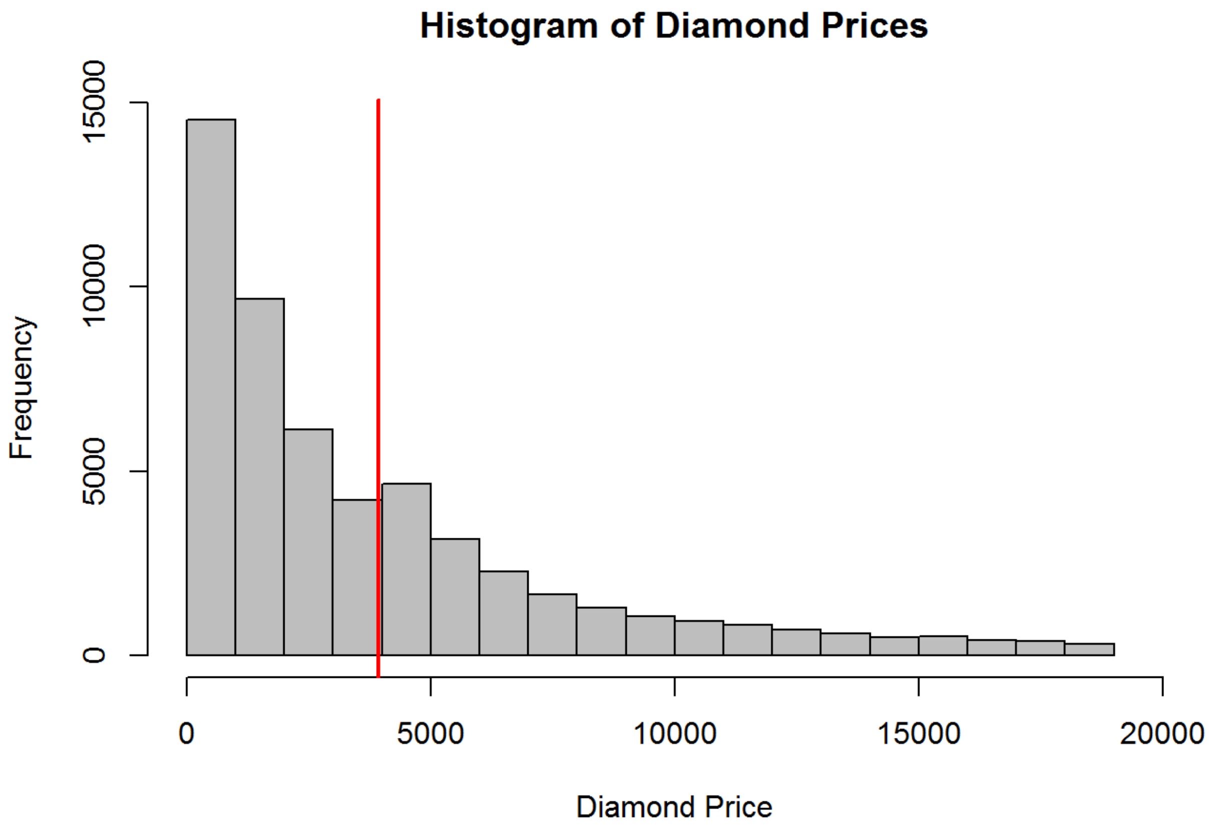
```
Diamonds %>% summarise(Mean = mean(price, na.rm = TRUE),
                             SD = sd(price, na.rm = TRUE))
```

```
##      Mean      SD
## 1 3932.8 3989.44
```

Note how we include `na.rm = TRUE` to remove missing values from the computations.

We quickly find the mean price equals 3932.8 and the standard deviation equals 3989.4. The very high standard deviation, which is higher than the mean itself, suggests great variability in diamond prices. We can plot the mean on the histogram to get a better sense of the value:

```
Diamonds$price %>% hist(,col="grey",xlim=c(0,20000),xlab="Diamond Price",
                            main="Histogram of Diamond Prices")
Diamonds$price %>% mean() %>% abline(v=.,col='red',lw=2)
```



I've added a red line showing the mean. As the mean is influenced by all values in the dataset, the histogram above suggests that the mean is being pulled towards higher values. This is what we refer to as positive skew. The mean might not be the best indicator of central tendency for such a highly skewed distribution. Fortunately, when we performed the `summary()` function, we also obtained some very useful values called quartiles.

Quartiles and the Median

Quartiles are values that break a distribution into four parts, so that 25% of the data values fall within each interval. We refer to these values as Q_1, Q_2 and Q_3. Q_2 is also referred to as the median. Data can also be broken into percentages. So, we could call Q_1 the 25th percentile, Q_2 the 50% percentile and Q_3 as the 75th percentile.

The median, Q_2 or the 50% percentile, is a measure of central tendency. The median splits an ordered dataset in half, with 50% of values above and below the median.

The calculation of the median depends on whether there are an even or odd number of data points. Consider the two following datasets. Scenario 1 has 7 measurements and scenario 2 has 6 measurements:

Scenario 1	**2, 4, 9, 8, 6, 5, 3**
Ordered	2, 3, 4, **5**, 6, 8, 9
Location of Median	$(n+1)/2$ = $(7+1)/2$ = 4th

Therefore, the median is the 4th ordered value, Median $= 5$. In R:

```
x <- c(2, 3, 4, 5, 6, 8, 9)
x %>% summary()
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.000   3.500   5.000   5.286   7.000   9.000
```

Scenario 1	**2, 4, 9, 8, 6, 5**
Ordered	2, 4, **5**, **6**, 8, 9
Location of Median	Average of the $(n/2)$ and $(n/2)+1$ observations, so the average of the 3rd and 4th

The median is the average of the 3rd and 4th ordered observation so, Median $= (5+6)/2 = 5.5$. In R:

```
y <- c(2, 4, 9, 8, 6, 5)
y %>% summary()
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.000   4.250   5.500   5.667   7.500   9.000
```

Q_1 and Q_3 are calculated in a similar fashion after the dataset is split at the median, top and bottom 50%. Q_1 is the median of the bottom 50% (i.e. 25th percentile) and Q_3 is median of the top 50% (i.e. 75th percentile).

Q_1 and Q_3 when n = odd (take median value)

- Q_1 = Median of bottom 50%: For example, Median of 2, 3, 4, 5 = average of 2nd and 3rd value = (3+4)/2 = 3.5
- Q_3 = Median of top 50%: For example, Median of 5, 6, 8, 9 = average of 2nd and 3rd value = (6+8)/2 = 7

Note how the median is included in both halves.

Q_1 and Q3 when n = even

- Q_1 = Median of bottom 50%: For example, Median of 2, 4, 5 = 2nd value = 4
- Q_3 = Median of top 50%: For example, Median of 6, 8, 9 = 2nd value = 8.

Note how the median is not included because the median is not an actual data point.

Note also that R does not use this method, which I have included as a simple instructional example. R has 9 different methods to calculate quartiles. We will stick to the default method produced by R. You just need to know conceptually what it represents.

Box Plots

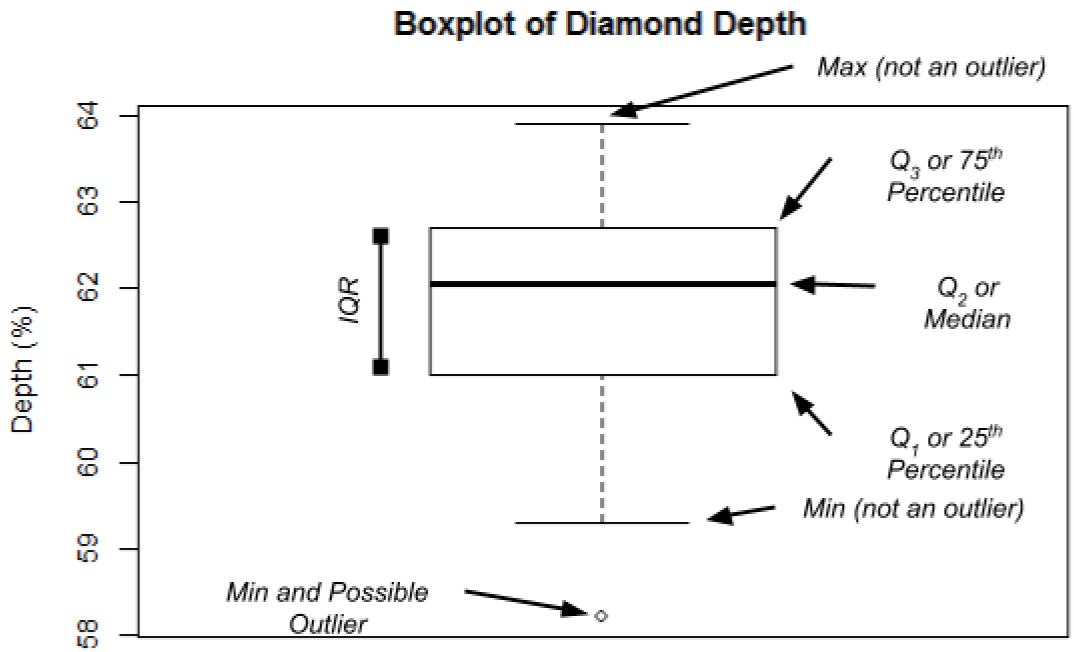
Box Plots are used to depict the quartiles of distribution. The following info-graphic puts all the concepts of quartiles, medians and percentiles together. Let’s first dissect a box plot of diamond depth from the small sample:

Here are the quartiles:

```
Diamonds_sample$depth %>% summary()
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   58.20   61.05   62.05   61.78   62.67   63.90
```

Now let’s discuss the anatomy of the basic box plot below.



The **interquartile range (IQR)** is the middle 50% of data and is depicted as the "box" in the box plot. The IQR is also a measure of variation.

$$IQR = Q_3 - Q_1$$

Box plots also include suspected **outliers**, depicted using an "o" or a similar symbol. Outliers are values that fall beyond the **outlier fences**. The outlier fences are defined as the following:

$$\text{Lower outlier} < Q_1 - 1.5 * IQR$$

$$\text{Upper outlier} > Q_3 + 1.5 * IQR$$

Using the depth data summary, we find:

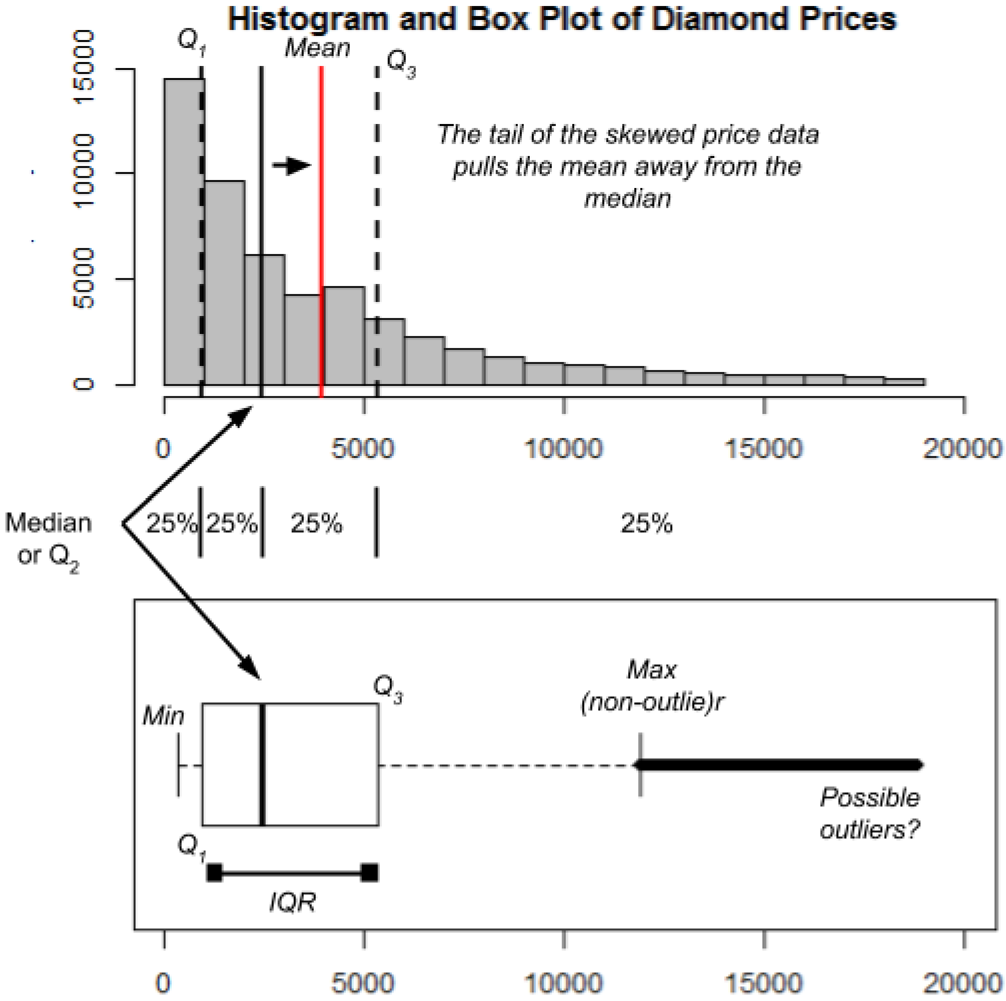
$$IQR = 62.675 - 61.05 = 1.625$$

$$\text{Lower outlier} < 61.05 - (1.5 * 1.625) = 58.6125$$

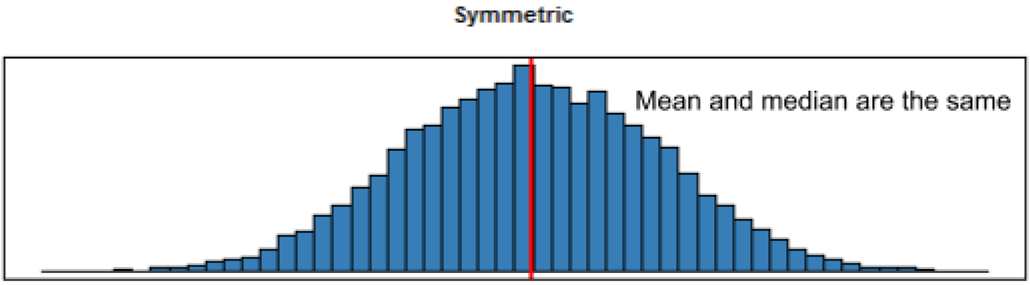
$$\text{Upper outlier} > 62.675 + (1.5 * 1.625) = 65.1125$$

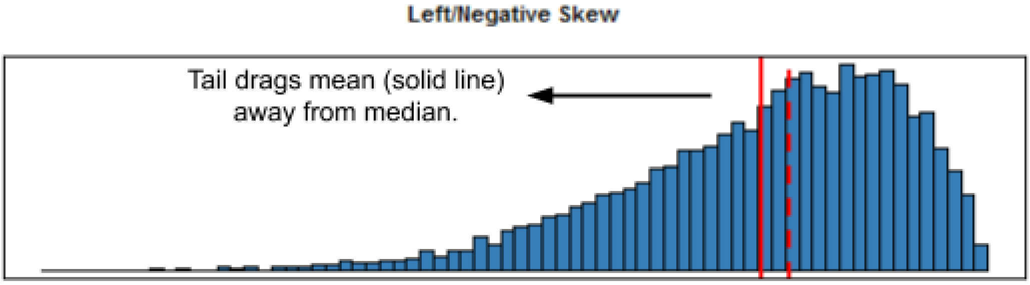
Outliers are unusual cases that should be investigated by the researcher. Don't automatically remove outliers until you have a good reason to do so. For example, an outlier might be a data entry error or a measurement that should have been excluded from the investigation. Removing outliers because they don't look "nice" is not appropriate. When you remove outliers for any reason, this should be made clear when you report your results.

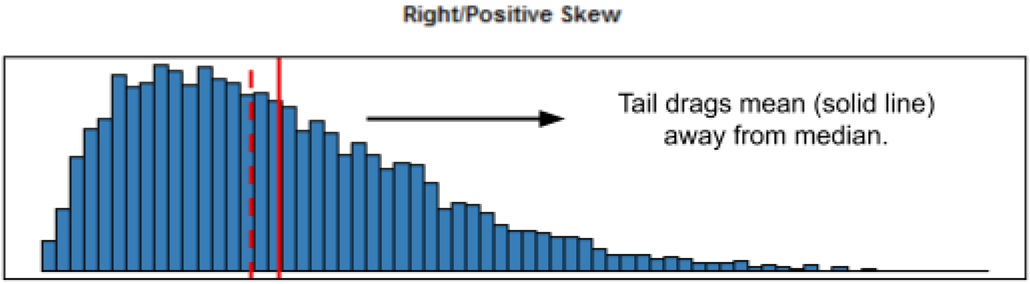
Now let's put histograms, box plots, and measures of central tendency together:



The mean and median will be the same when the data have a symmetrical distribution. The median is said to be more robust (less sensitive to unusual cases) than the mean. Therefore, the median is often preferred when a variable's distribution is skewed or has many outliers present. This is the case for the highly skewed diamond price data. The median appears to give a better account of the central tendency of the data, while the mean seems unduly influenced by the long tail. The following info-graphic summarises the effect of skewness on the mean and median. The mean is the solid red line, and the median is the dashed red line. The tail of a skewed distribution always draws the mean towards it, because the mean takes all values into account.







Note that these effects are generally true for large datasets. In small datasets, skewness is often poorly estimated, so the relationship among the mean, median and skewness can behave in unusual ways.

Comparing Groups

Descriptive statistics and visualisations are often used to compare groups. The following examples show how the basic descriptive R functions and plots can be quickly extended to assist with comparing the distributions of a quantitative variable across qualitative groupings. The example will consider comparing diamond price distributions across colour. First, lets get a descriptive statistics table using the `summarise()` function from the `dplyr` package.

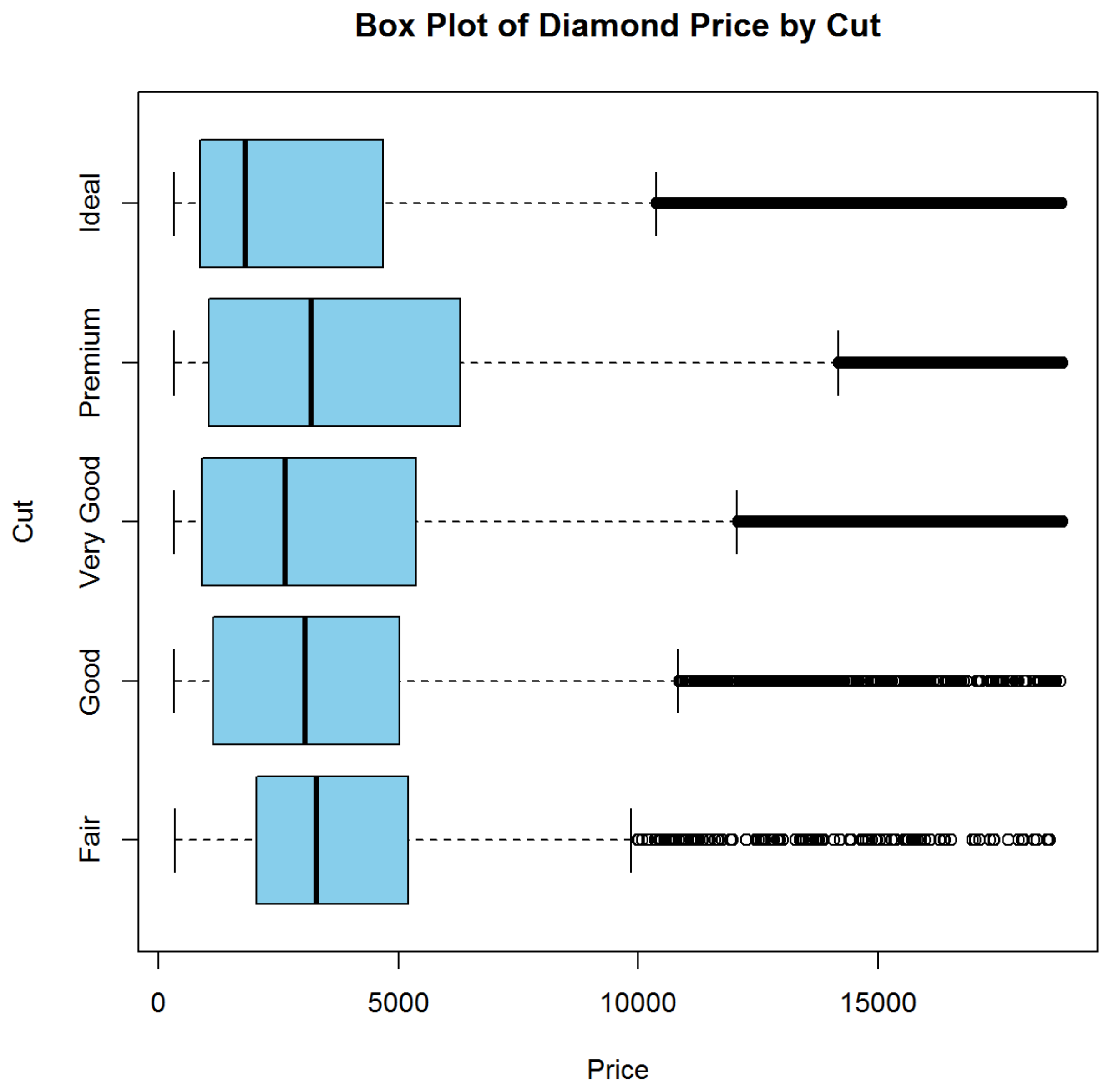
The table will include min, Q_1, median, Q_3, max, mean, standard deviation, n and missing value count. I've included quite a few descriptive statistics so you can get a sense of how to customise the table. Essentially, the table can incorporate any descriptive functions in R provided the function results in a single value. Take note how we have to include the option `na.rm = TRUE` to most of these functions. If we didn't, and missing values were present, the function would fail.

```
Diamonds %>% group_by(cut) %>% summarise(Min = min(price,na.rm = TRUE),
                                         Q1 = quantile(price,probs = .25,
  na.rm = TRUE),
                                         Median = median(price, na.rm = T
  RUE),
                                         Q3 = quantile(price,probs = .75,
  na.rm = TRUE),
                                         Max = max(price,na.rm = TRUE),
                                         Mean = mean(price, na.rm = TRUE
  ),
                                         SD = sd(price, na.rm = TRUE),
                                         n = n(),
                                         Missing = sum(is.na(price)))
```

```
## # A tibble: 5 x 10
##   cut         Min    Q1 Median    Q3   Max  Mean    SD     n Missing
##   <ord>     <int> <dbl>  <dbl> <dbl> <int> <dbl> <dbl> <int>   <int>
## 1 Fair        337 2050.   3282  5206. 18574 4359. 3560.  1610       0
## 2 Good        327 1145    3050. 5028  18788 3929. 3682.  4906       0
## 3 Very Good   336  912    2648  5373. 18818 3982. 3936. 12082       0
## 4 Premium     326 1046    3185  6296  18823 4584. 4349. 13791       0
## 5 Ideal       326  878    1810  4678. 18806 3458. 3808. 21551       0
```

Next, let’s create a side-by-side box plot.

```
Diamonds %>% boxplot(price ~ cut,data = ., main="Box Plot of Diamond Pric
  e by Cut",
                     ylab="Cut", xlab="Price",horizontal=TRUE, col = "sky
  blue")
```



Using this plot, confirm the following features:

- Ideal has the smallest median price
- Premium has the highest IQR
- All price distributions are positively skewed
- All price distributions have many suspected outliers
- Fair has the highest Q_1
- Premium has the highest Q_3

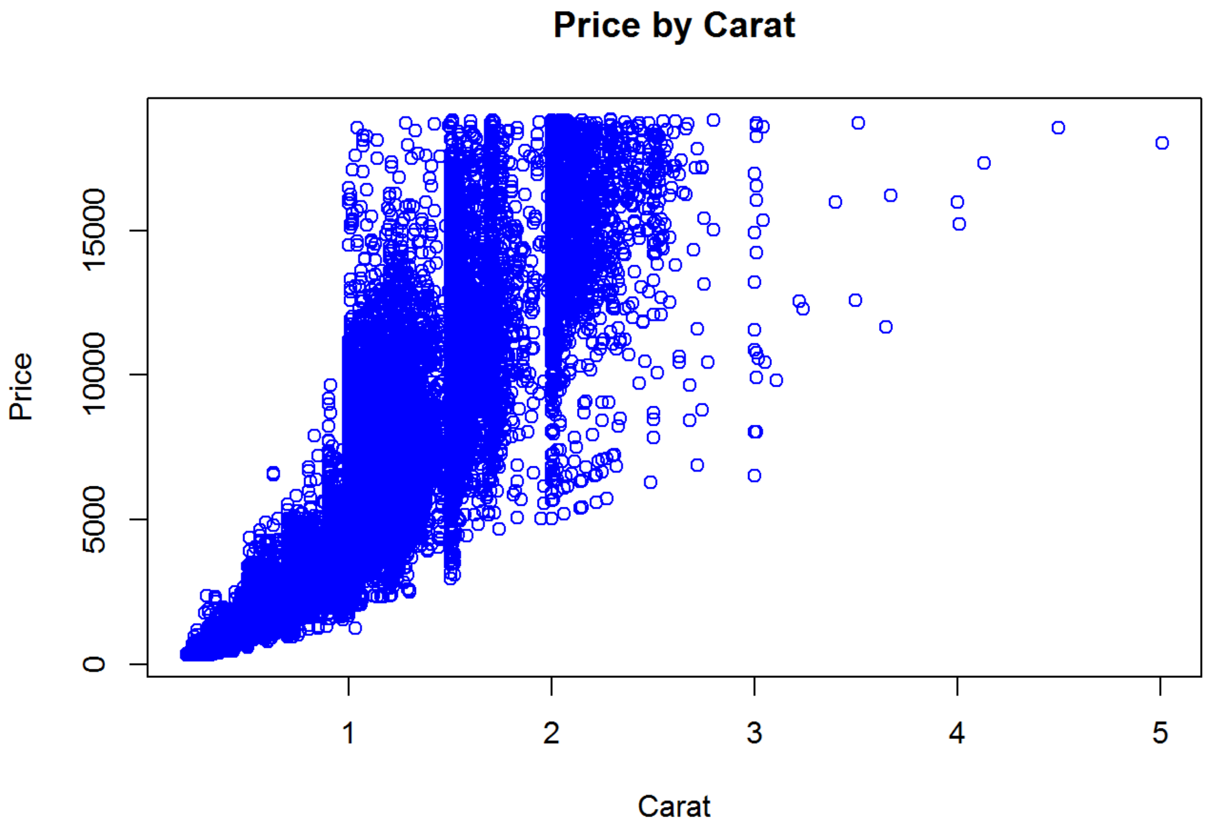
Scatter Plots

Scatter plots are used to visualise the relationship between two quantitative variables. One variable is plotted on the x axis and one variable on the y. The bivariate, or paired data, are indicated on the plot using a point. Let's look at some examples. Let's first consider the relationship between diamond price and carat. Here's an excerpt from the data showing 10 sets of bivariate data, x, y.

```
##     ID Carat Price
## 1    1  0.23   326
## 2    2  0.21   326
## 3    3  0.23   327
## 4    4  0.29   334
## 5    5  0.31   335
## 6    6  0.24   336
## 7    7  0.24   336
## 8    8  0.26   337
## 9    9  0.22   337
## 10 10  0.23   338
```

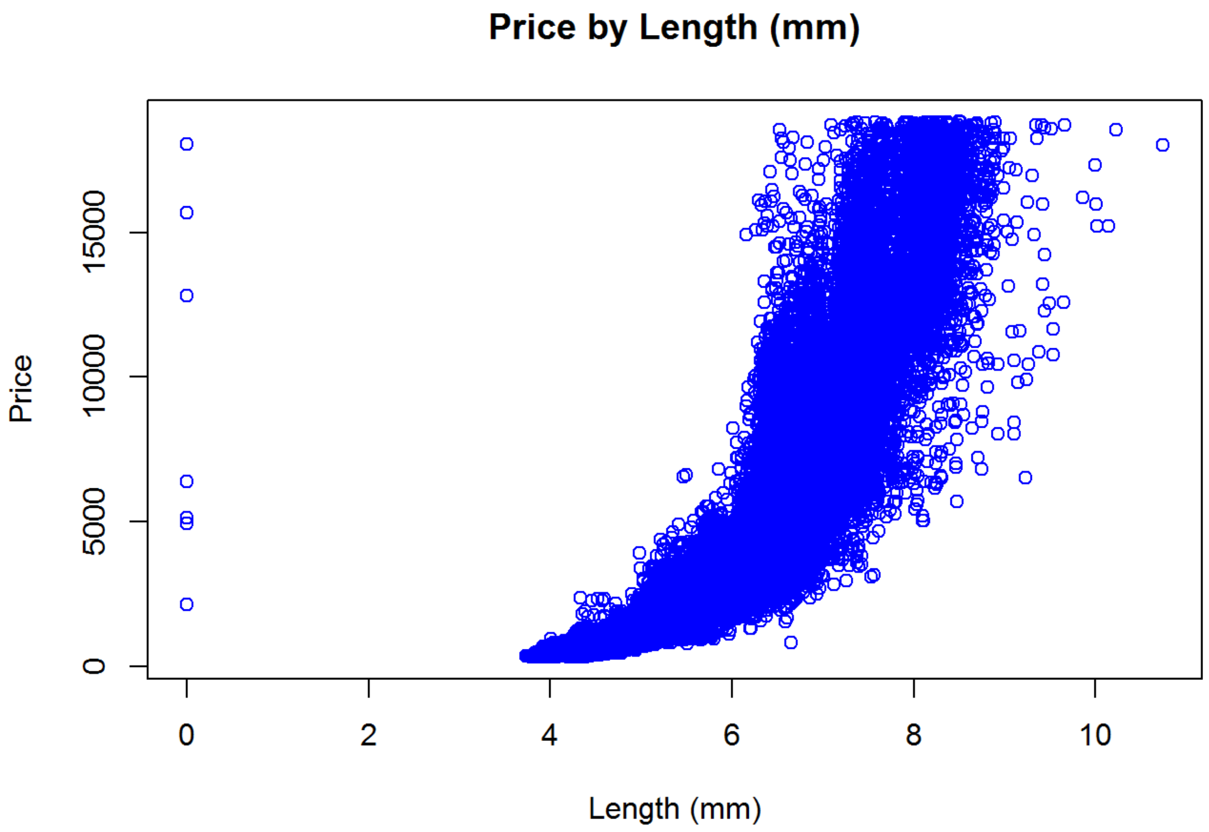
A point is plotted at the intersection between the x and y value for each data point. For example, for ID = 1, a point is plotted where carat = 0.23 intersects with a price value of 326. This is repeated for all 54,000 data points. Fortunately, you don't have to do this by hand. Using R:

```
Diamonds %>% plot(price ~ carat, data = .,ylab="Price", xlab="Carat",
                  col="blue",main="Price by Carat")
```



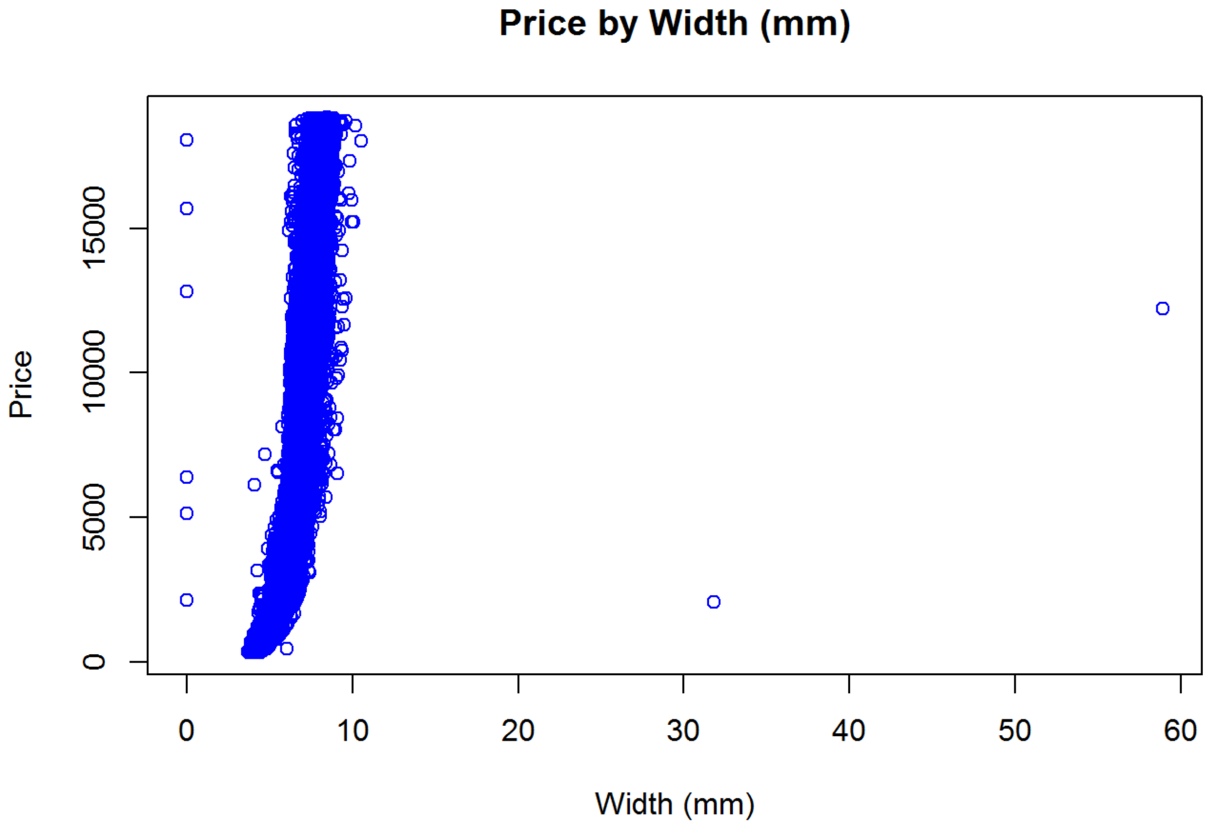
The scatter plot for price by carat suggests that as carat increases, price also tends to increase. However, there is still lots of variability that other diamond characteristics may help explain. Weight is not everything. Let's have a look at a few more examples:

```
Diamonds %>% plot(price ~ x, data = ., ylab="Price", xlab="Length (mm)",
                  col="blue",main="Price by Length (mm)")
```



Notice the 0 values. Scatter plots are a great way to explore your data to find possible irregularities.

```
Diamonds %>% plot(price ~ y, data = ., ylab="Price", xlab="Width (mm)",
                  col="blue",main="Price by Width (mm)")
```



Once again, there appears to be some outliers that need following up.