Module 6

Scan: Outliers

Dr. Anil Dolgun

Last updated: 09 April, 2018

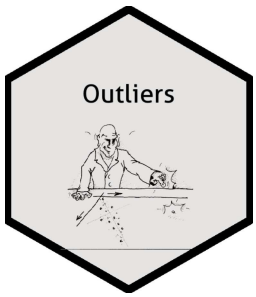
Overview

Summary

In statistics, an outlier is defined as an observation which stands far away from the most of other observations. An outlier can be a result of an measurement error and the inclusion of that error would have a great impact on the analysis results. Therefore, every data set should be scanned for possible outliers before conducting any statistical analysis. Like missing values, detecting outliers and dealing with them are also crucial in the data preprocessing. In this module, first you will learn how to identify univariate and multivariate outliers using descriptive, graphical and distance based methods. Then, you will learn different approaches to deal with these values using R.

Learning Objectives

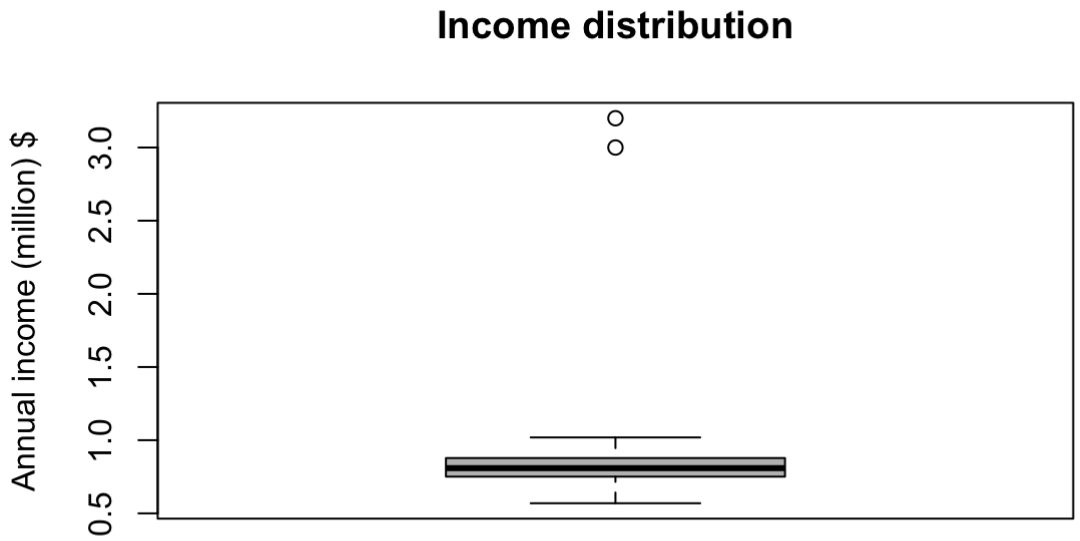
The learning objectives of this module are as follows:



- Identify the outlier(s) in the data set.
- Apply descriptive methods to identify univariate outliers
- Apply graphical approaches to scan for univariate or multivariate outliers
- Apply distance based metrics to identify univariate or multivariate outliers
- Learn commonly used approaches to handle outliers.

Outliers

In statistics, an outlier is defined as an observation which stands far away from the most of the other observations. An outlier deviates so much from other observations as to arouse suspicion that is was generated by a different mechanism (Hawkins (1980)). Let's take an example. Assume that we are interested in customer profiling and we find out that the average annual income of our customers is 800K. But, there are two customers having annual income of 3 and 3.2 million dollars. These two customers' annual income is much higher than rest of the customers (see the box plot below). These two observations will be seen as outliers.

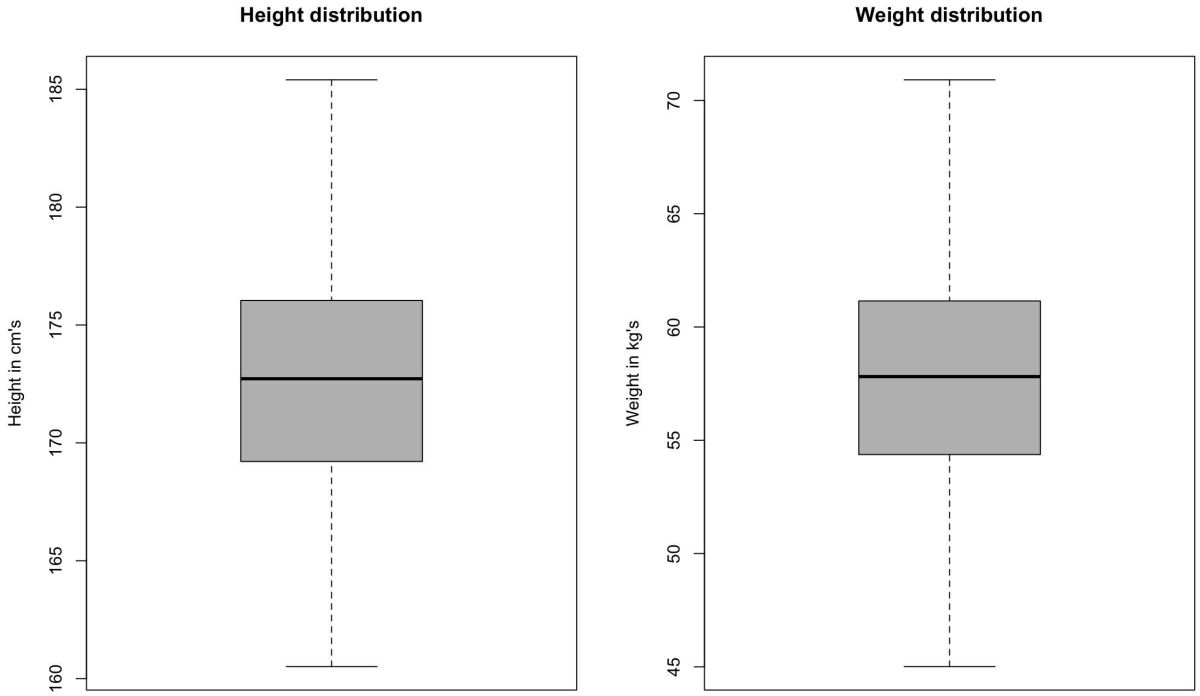


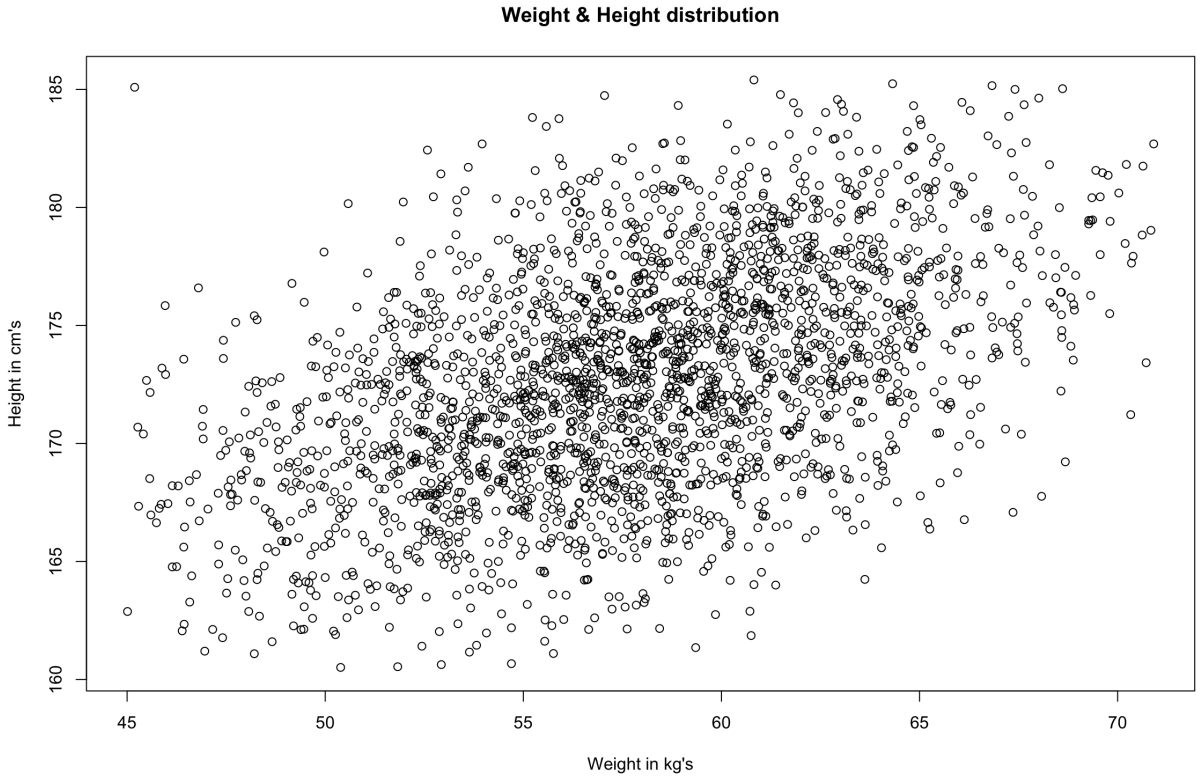
Types of Outliers

Outlier can be univariate and multivariate. **Univariate outliers** can be found when looking at a distribution of values in a single variable. The example given above is an example of a univariate outlier as we only look at the distribution of income (i.e., one variable) among our customers. On the other hand, **multivariate outliers** can be found in a n-dimensional space (of n-variables). In order to find them, we need to look at distributions in multi-dimensions.

To illustrate multivariate outliers, let's assume that we are interested in understanding the relationship between height and weight. Below, we have univariate and bivariate distribution for Height and Weight. When we look at the univariate distributions of Height and Weight (i.e., using box plots) separately, we don't spot any abnormal cases (i.e. above and below the $1.5 \times IQR$ fence). However, when we look at the bivariate (two dimensional) distribution of Height and Weight (using scatter plot), we can see that we have one observation whose

weight is 45.19 kg and height is 185.09 (on the upper-left side of the scatter plot). This observation is far away from the most of the other weight and height combinations thus, will be seen as a multivariate outlier.





Most Common Causes of Outliers

Often an outlier can be a result of data entry errors, measurement errors, experimental errors, intentional errors, data processing errors or due to the sampling (i.e., sampling error). The following are the most common causes of outliers (taken from: https://www.analyticsvidhya.com/blog/2016/01/guide-data-exploration/ (https://www.analyticsvidhya.com/blog/2016/01/guide-data-exploration/))

- **Data Entry Errors:** Outliers can arise because of the human errors during data collection, recording, or entry.
- **Measurement Errors:** It is the most common source of outliers. This is caused when the measurement instrument used turns out to be faulty.
- **Experimental Error:** Another cause of outliers is the experimental error. Experimental errors can arise during data extraction, experiment/survey planning and executing errors.
- **Intentional Error:** This type of outlier is commonly found in self-reported measures that involves sensitive data. For example, teens would typically under report the amount of alcohol that they consume. Only a fraction of them would report actual value. Here actual values might look like outliers because rest of the teens are under reporting the consumption.
- **Data Processing Errors:** Often, due to the data sets are extracted from multiple sources, it is possible that some manipulation or extraction errors may lead to outliers in the data set.
- **Sampling error:** Sometimes, outliers can arise due to the sampling (taking samples from population) process. Typically, this type of outliers can be seen when we take a few observations as a sample.

Detecting Outliers

Outliers can drastically change the results of the data analysis and statistical modelling. Some of the unfavourable impacts of outliers are;

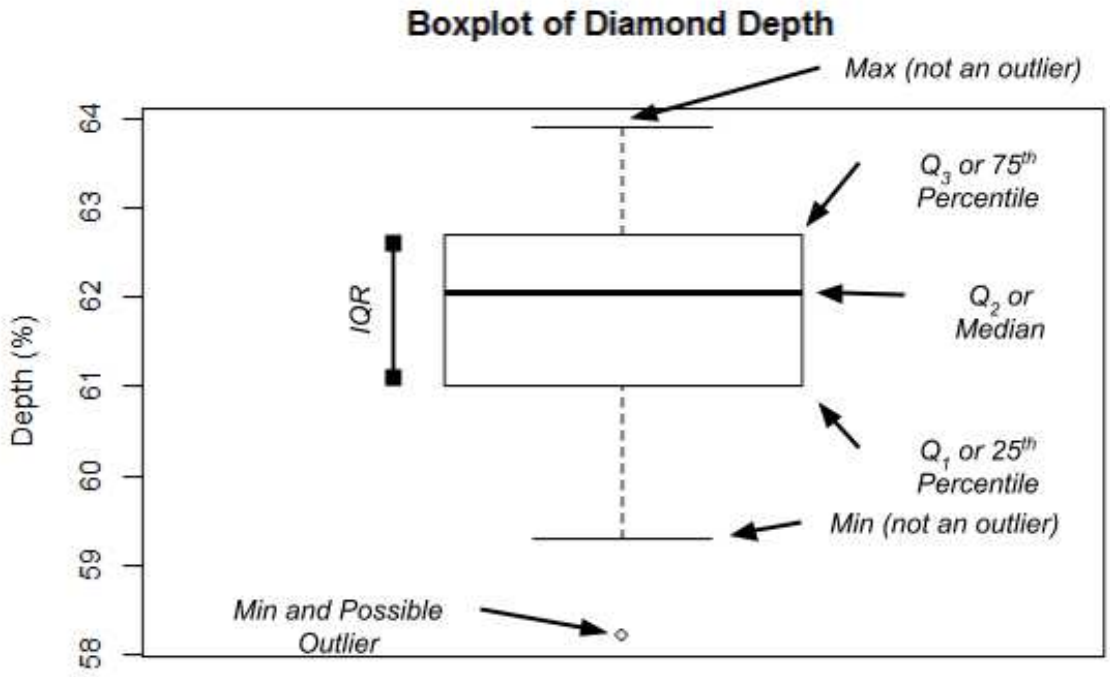
- they increase the error variance,
- they reduce the power of statistical tests,
- they can bias or influence the estimates of model parameters that may be of substantive interest.

Therefore, one of the most important tasks in data preprocessing is to identify and properly handle the outliers.

There are many methods developed for outlier detection. Majority of them deal with numerical data. This module will introduce the most basic ones with their application using R packages.

Univariate Outlier Detection Methods

One of the simplest methods for detecting univariate outliers is the use of box plots. A box plot is a graphical display for describing the distribution of the data using the median, the first (Q1) and third quartiles (Q3), and the inter-quartile range ($IQR = Q3 - Q1$). Below is an illustration of a typical box plot (taken from Dr. James Baglin's Intro to Stats website (https://astral-theory-157510.appspot.com/secured/MATH1324_Module_02.html#box_plots)).



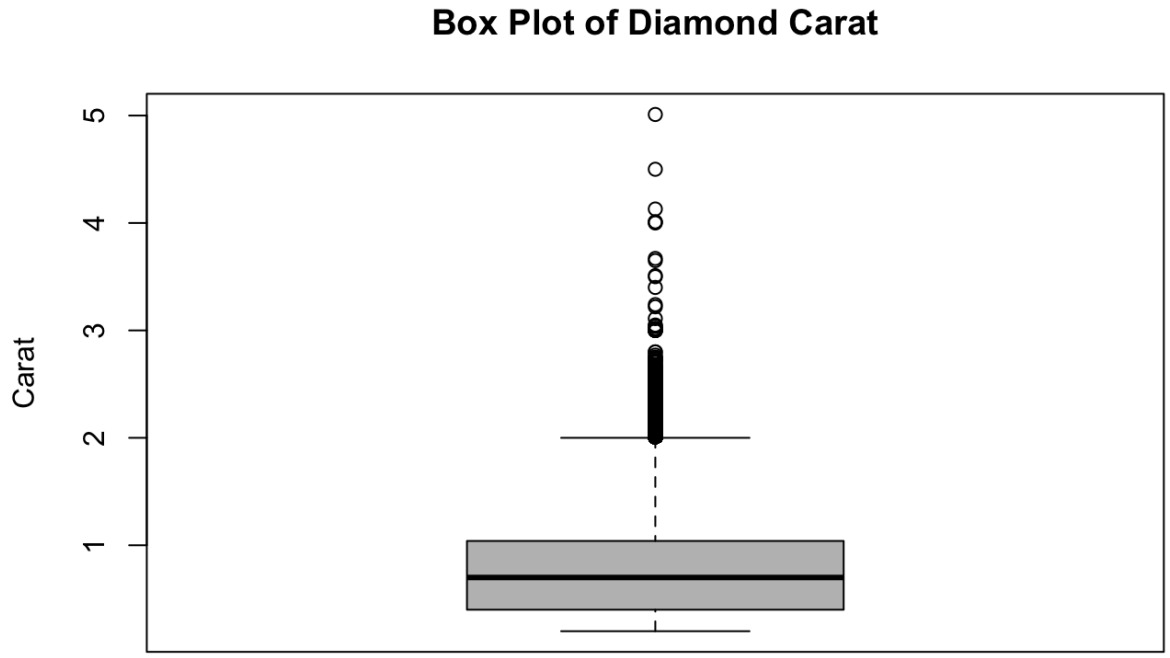
In the box plot, the "**Tukey's method of outlier detection**" is used to detect outliers. According to this method, outliers are defined as the values in the data set that fall beyond the range of $-1.5 \times IQR$ to $1.5 \times IQR$. These $-1.5 \times IQR$ and $1.5 \times IQR$ limits are called "outlier fences" and any values lying outside the outlier fences are depicted using an "o" or a similar symbol on the box plot.
In order to illustrate the box plot, we will use the Diamonds.csv (data/Diamonds.csv) data set available under the data repository.

```
Diamonds <- read.csv("data/Diamonds.csv")
head(Diamonds)
```

```
##   carat       cut color clarity depth table price    x    y    z
## 1  0.23     Ideal     E     SI2  61.5    55   326 3.95 3.98 2.43
## 2  0.21   Premium     E     SI1  59.8    61   326 3.89 3.84 2.31
## 3  0.23      Good     E     VS1  56.9    65   327 4.05 4.07 2.31
## 4  0.29   Premium     I     VS2  62.4    58   334 4.20 4.23 2.63
## 5  0.31      Good     J     SI2  63.3    58   335 4.34 4.35 2.75
## 6  0.24 Very Good     J    VVS2  62.8    57   336 3.94 3.96 2.48
```

We can use `boxplot()` function (under Base graphics) to get the box plot of the `carat` variable:

```
Diamonds$carat %>%  boxplot(main="Box Plot of Diamond Carat", ylab="Carat", c
ol = "grey")
```



According to the Tukey's method, the `carat` variable seems to have many outliers. In the next section, we will discuss different approaches to handle these outliers.

There are also distance based methods to detect univariate outliers. One of them is to use the z- scores (i.e., normal scores) method. In this method, a standardised score (z-score) of all observations are calculated using the following equation:

$$z_i = \frac{X_i - \bar{X}}{S}$$

In the equation below, X_i denotes the values of observations, $\bar{X}$ and S are the sample mean and standard deviation, respectively. An observation is regarded as an outlier based on its z-score, if the absolute value of its z**-score is greater than 3**.

In order to illustrate the z- score approach, we will use the "**outliers package**". The outliers package provides a number of useful functions to systematically extract outliers. Among those, the `scores()` function will calculate the z-scores (in addition to the t, chi-square, IQR, and Median absolute deviation scores) for the given data. Note that, there are many alternative functions in R for calculating z-scores. You may also use these functions and detect the outliers.

```
library(outliers)

z.scores <- Diamonds$carat %>%  scores(type = "z")
z.scores %>% summary()
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -1.2614 -0.8395 -0.2066  0.0000  0.5107  8.8860
```

From the `summary()` output, we can see that the minimum z score is -1.26 and the maximum is 8.88.

Using `which()`, we can also find the locations of the z-scores whose absolute value is greater than 3.

```
# Finds the locations of outliers in the carat variable

which( abs(z.scores) >3 )
```

```
##   [1] 13758 14139 15685 16284 16440 16638 17182 17197 17467 17561 17692
##  [12] 17898 17957 19082 19340 19347 19599 19867 19895 19922 19947 20298
##  [23] 20463 20839 20852 21567 21626 21759 21863 22005 22400 22414 22429
##  [34] 22441 22541 22579 22609 22671 22742 22832 22871 22873 22998 23072
##  [45] 23092 23098 23215 23219 23235 23496 23540 23581 23613 23645 23775
##  [56] 23784 23878 23890 23940 23961 24115 24125 24132 24148 24182 24186
##  [67] 24259 24274 24276 24279 24298 24329 24415 24435 24484 24495 24524
##  [78] 24546 24581 24597 24605 24613 24617 24628 24674 24731 24745 24785
##  [89] 24814 24817 24841 24863 24864 24991 25017 25054 25056 25062 25077
## [100] 25114 25138 25140 25141 25143 25155 25192 25193 25203 25227 25231
## [111] 25238 25259 25268 25276 25279 25284 25339 25340 25393 25402 25410
## [122] 25416 25436 25441 25454 25457 25459 25461 25480 25481 25484 25497
## [133] 25526 25530 25532 25533 25540 25564 25579 25580 25627 25628 25649
## [144] 25650 25676 25723 25728 25730 25733 25736 25738 25739 25760 25763
## [155] 25769 25771 25776 25779 25791 25793 25804 25816 25819 25836 25851
## [166] 25860 25875 25878 25900 25901 25911 25922 25951 25965 25980 25999
## [177] 26000 26006 26044 26048 26068 26083 26086 26089 26091 26101 26105
## [188] 26108 26119 26124 26128 26130 26138 26139 26140 26178 26202 26213
## [199] 26257 26267 26270 26271 26288 26289 26314 26321 26379 26400 26405
## [210] 26412 26432 26445 26448 26451 26456 26460 26463 26465 26468 26469
## [221] 26475 26476 26486 26495 26497 26505 26521 26524 26526 26529 26535
## [232] 26536 26537 26538 26539 26553 26554 26555 26560 26561 26569 26570
## [243] 26572 26585 26589 26593 26601 26605 26608 26609 26617 26631 26645
## [254] 26653 26658 26660 26743 26744 26745 26750 26752 26757 26760 26767
## [265] 26768 26793 26797 26804 26805 26807 26816 26818 26819 26820 26824
## [276] 26865 26867 26871 26873 26876 26883 26886 26887 26905 26909 26910
## [287] 26916 26917 26923 26925 26932 26933 26934 26938 26941 26943 26949
## [298] 26957 26959 26964 26972 26975 26978 27000 27006 27008 27015 27017
## [309] 27018 27019 27023 27024 27025 27027 27028 27066 27067 27071 27081
## [320] 27082 27086 27095 27096 27108 27119 27120 27123 27131 27136 27147
## [331] 27162 27164 27177 27179 27180 27186 27198 27203 27205 27210 27213
## [342] 27221 27228 27239 27240 27247 27254 27257 27261 27266 27273 27285
## [353] 27303 27315 27325 27335 27337 27341 27353 27354 27355 27396 27397
## [364] 27404 27406 27408 27413 27416 27417 27421 27422 27425 27426 27429
## [375] 27430 27431 27434 27442 27449 27450 27455 27457 27464 27477 27490
## [386] 27497 27514 27515 27516 27517 27518 27519 27522 27523 27524 27541
## [397] 27542 27544 27551 27552 27556 27560 27562 27564 27567 27583 27584
## [408] 27587 27591 27599 27613 27617 27621 27622 27625 27630 27631 27632
## [419] 27634 27635 27639 27640 27650 27651 27658 27664 27673 27675 27676
## [430] 27680 27682 27685 27686 27727 27728 27732 27740 27745 27750
```

```
# Finds the total number of outliers according to the z-score

length (which( abs(z.scores) >3 ))
```

```
## [1] 439
```

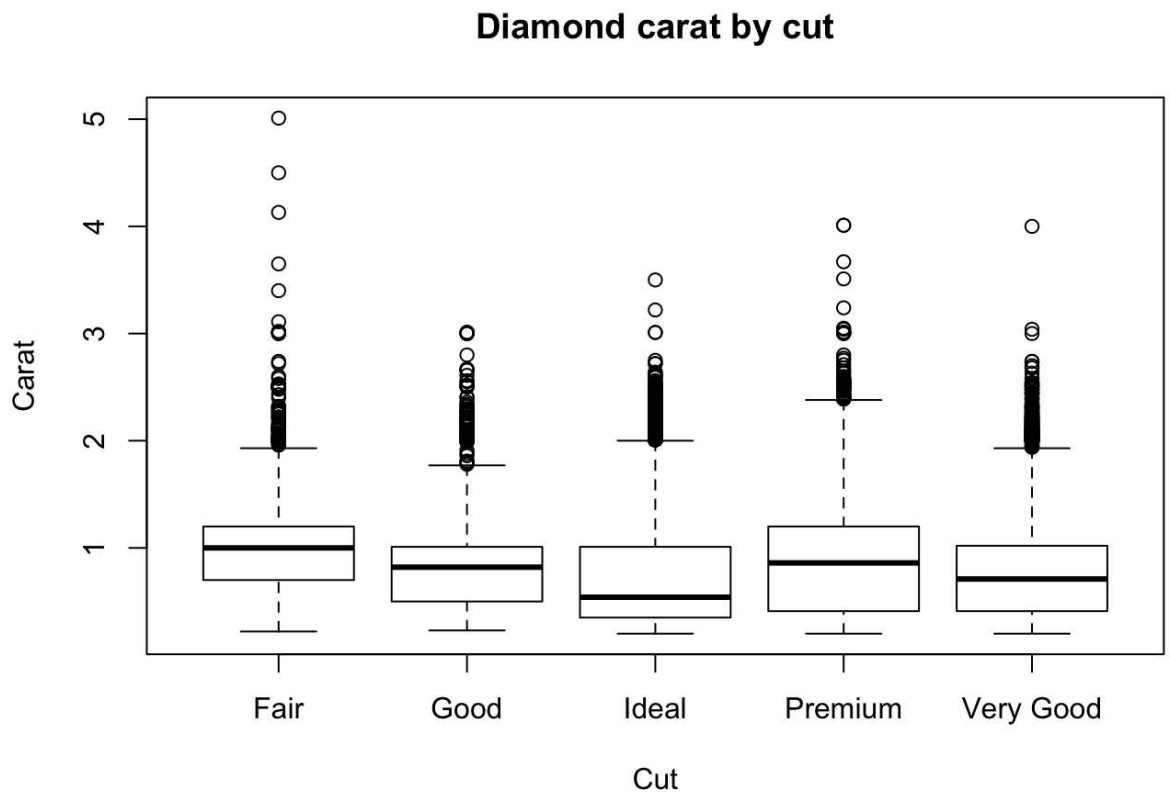
According to the z-score method, the `carat` variable has 439 outliers.

There are also many statistical tests to detect outliers. Some of them are the Chi-square test for outliers, the Cochran's test, the Dixon test and the Grubbs test, In this course, we won't cover these statistical testing approaches used for detecting outliers. You may refer to the `outliers` package manual (https://cran.r-project.org/web/packages/outliers/outliers.pdf) which includes useful functions for the commonly used outlier tests as well as the distance based approaches.

Multivariate Outlier Detection Methods

When we have only two variables, the bivariate visualisation techniques like bivariate box plots and scatter plots, can easily be used to detect any outliers.
To illustrate the bivariate box plot, we will assume that we are only interested in one numerical `carat` variable and a one factor (quantitative) `cut` variable in the Diamonds data set.

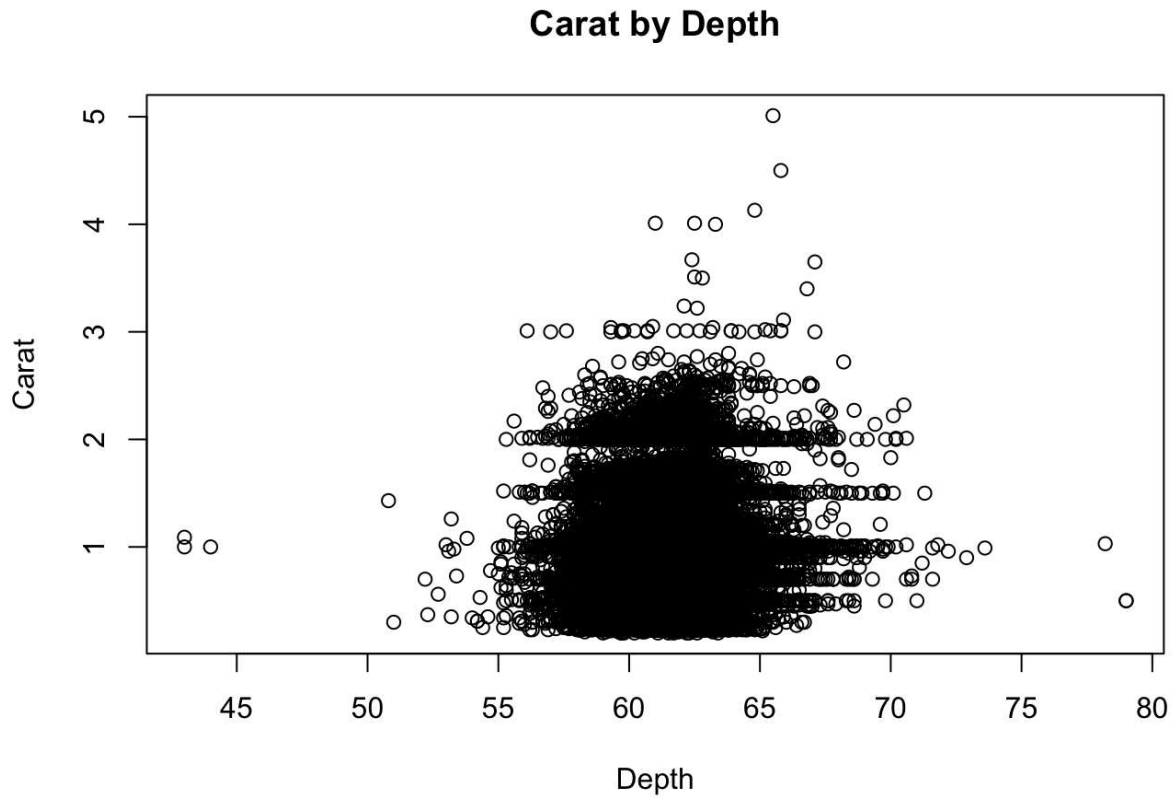
```
boxplot(Diamonds$carat ~ Diamonds$cut, main="Diamond carat by cut", ylab = "C
arat", xlab = "Cut")
```



Using the univariate box plot approach, we can easily detect outliers in `carat` for a given `cut`.
Another bivariate visualisation method is the scatter plots. These plots are used to visualise the relationship between two quantitative variables. They are also very useful tools to detect

obvious outliers for the two dimensional data (i.e., for two continuous variables).
The `plot()` function will be used to get the scatter plot and detect outliers in `carat` and `depth` variables:

```
Diamonds %>% plot(carat ~ depth, data = ., ylab="Carat", xlab="Depth", main=
"Carat by Depth")
```



According to the scatter plot, there are some possible outliers on the lower left (the data points where both carat and depth is small) and lower right hand side of the scatter (the data points with larger depth but small carat values).

Scatter plots are useful methods to detect abnormal observations for a given pair of variables. However, when there are more than two variables, scatter plots can no longer be used. For such cases, multivariate distance based methods of outlier detection can be used.
The Mahalanobis distance is the most commonly used distance metric to detect outliers for the multivariate setting. The Mahalanobis distance is simply an extension of the univariate z-score, which also accounts for the correlation structure between all the variables. Mahalanobis distance follows a Chi-square distribution with n (number of variables) degrees of freedom, therefore **any Mahalanobis distance greater than the critical chi-square value is treated as outliers**.
I will not go into details of calculation of the Mahalanobis distance. We will use the `MVN` package to get these distances as it will also provide us the useful Mahalanobis distance vs. Chi-square quantile distribution plot (QQ plot) and contour plots. For more information on this package and its capabilities please refer to the paper on https://cran.r-project.org/web/packages/MVN/vignettes/MVN.pdf (https://cran.r-project.org/web/packages/MVN/vignettes/MVN.pdf)

To illustrate the usage of the MVN package to detect multivariate outliers, we will use a subset of the Iris data (data/iris.csv), which is versicolor flowers, with the first three variables (`Sepal.Length` , `Sepal.Width` and `Petal.Length`).

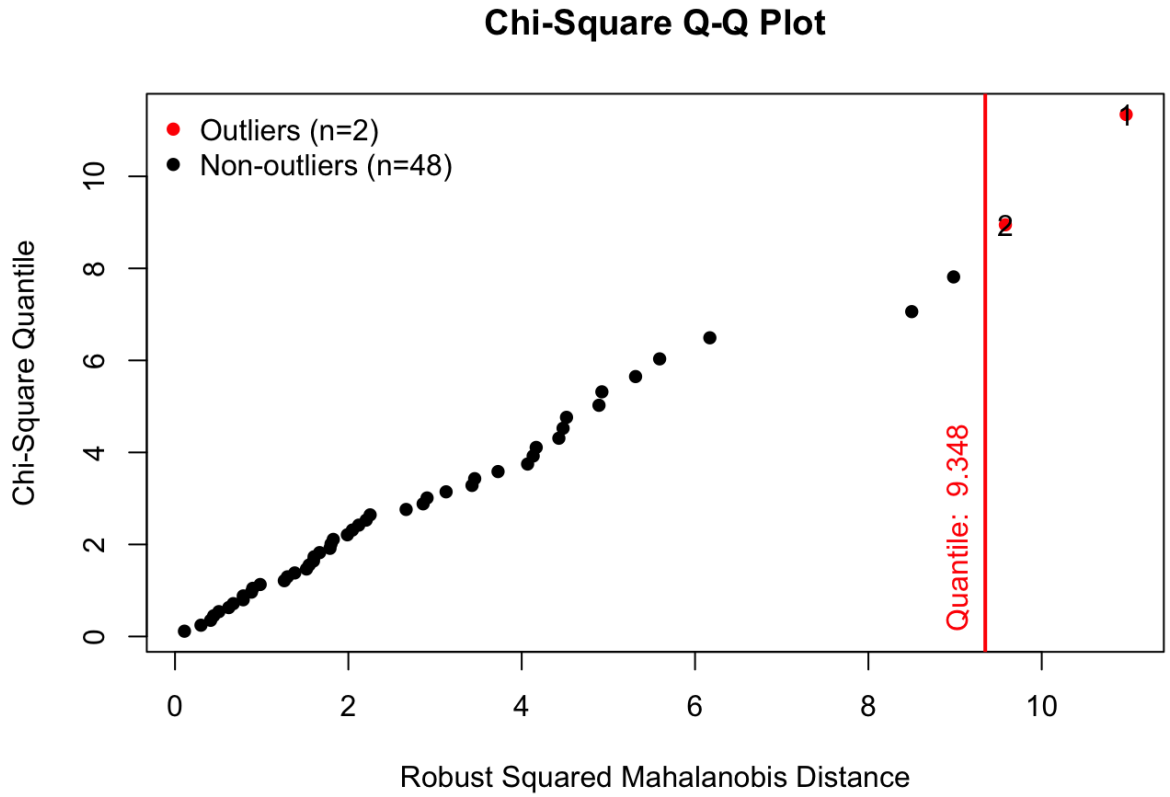
First let's read the data using:

```
# load iris data and subset using versicolor flowers, with the first three va
riables

iris <- read.csv("data/iris.csv")

versicolor <- iris %>%  filter( Species == "versicolor" ) %>%  select(Sepal.L
ength, Sepal.Width,Petal.Length )

head(versicolor)
```

```
##   Sepal.Length Sepal.Width Petal.Length
## 1          7.0         3.2          4.7
## 2          6.4         3.2          4.5
## 3          6.9         3.1          4.9
## 4          5.5         2.3          4.0
## 5          6.5         2.8          4.6
## 6          5.7         2.8          4.5
```

The `mvn()` function under MVN package (Korkmaz, Goksuluk, and Zararsiz (2014)) lets us to define the multivariate outlier detection method using the `multivariateOutlierMethod` argument. When we use `multivariateOutlierMethod = "quan"` argument, it detects the multivariate outliers using the chi-square distribution critical value approach mentioned above. The `showOutliers = TRUE` argument will depict any multivariate outliers and show them on the QQ plot.

```
# Multivariate outlier detection using Mahalanobis distance with QQ plots

results <- mvn(data = versicolor, multivariateOutlierMethod = "quan", showOut
liers = TRUE)
```



As we can see on the QQ plot, the Mahalonobis distance suggests existence of two outliers for this subset of the iris data.

If we would like to see the list of possible multivariate outliers, we can use `result$multivariateOutliers` to select only the results related to outliers as follows:

```
results$multivariateOutliers
```

```
##   Observation Mahalanobis Distance Outlier
## 1           1               10.974    TRUE
## 2           2                9.580    TRUE
```

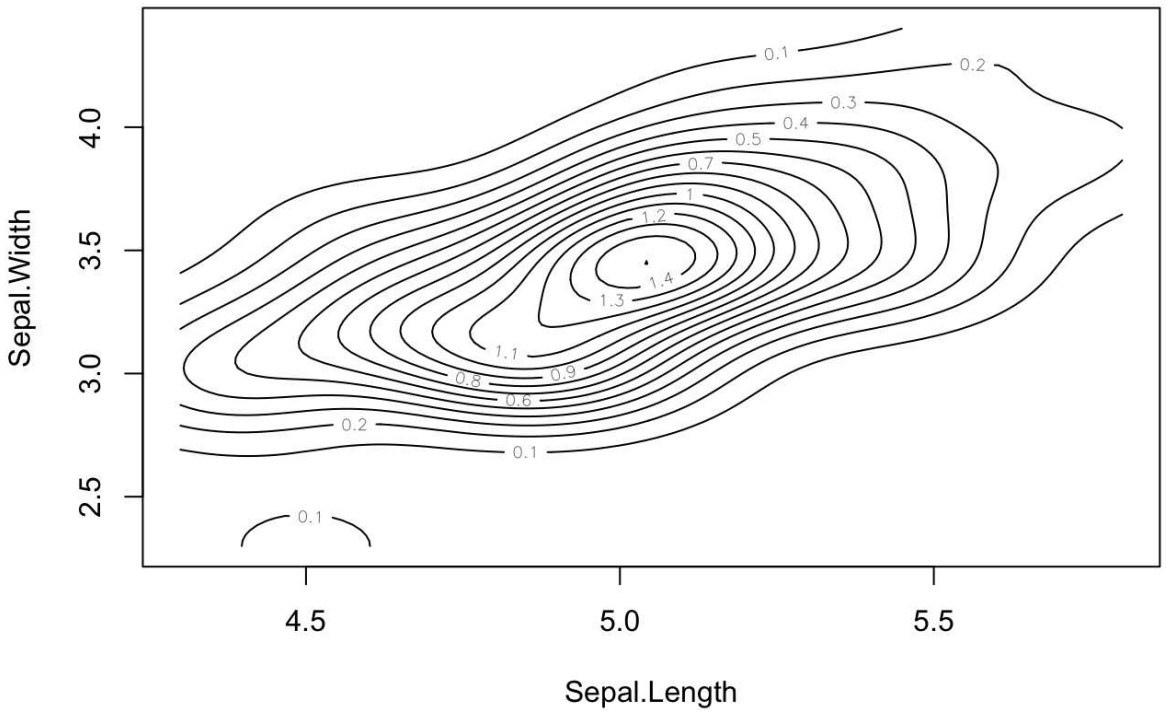
This output is very useful in terms of providing the locations of outliers in the data set. In our example the 1st and 2nd observations are the suggested outliers for this subset of iris data.

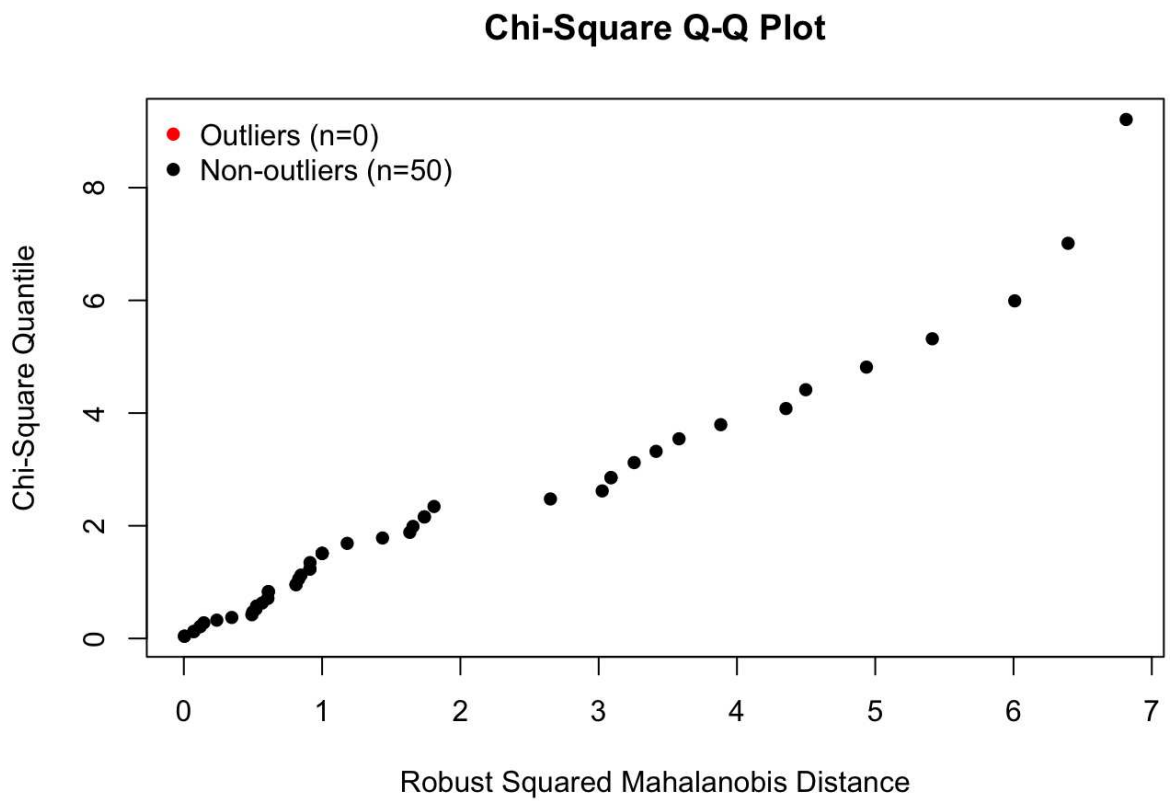
The `mvn()` function has also different plot options to help discovering possible multivariate outliers. For example to get the bivariate contour plots, we can use `multivariatePlot = "contour"` argument. However note that contour plots can only be used for two variables.
Let's illustrate the contour plots on the subset of setosa flowers, with the first two variables, `Sepal.Length` and `Sepal.Width`.

```
# load iris data and subset using setosa flowers, with the first two variable
s

setosa <- iris %>%  filter( Species == "setosa" ) %>%  select(Sepal.Length, S
epal.Width)
head(setosa)
```

```
##   Sepal.Length Sepal.Width
## 1          5.1         3.5
## 2          4.9         3.0
## 3          4.7         3.2
## 4          4.6         3.1
## 5          5.0         3.6
## 6          5.4         3.9
```

```
# Multivariate outlier detection using Mahalanobis distance with contour plot
s

results <- mvn(data = setosa, multivariateOutlierMethod = "quan", multivariat
ePlot = "contour")
```





From the contour plot, we can see one anomality on the left hand side of the plot, however, according to the QQ plot, this abnormal case doesn't seem to be an outlier as its Mahanolobis distance didn't exceed the threshold chi-square value.

Some applications of univariate and multivariate outlier detection can extend from fraud detection (i.e., unusual purchasing behaviour of a credit card owner), medicine (i.e. detection of unusual symptoms of a patient), sports statistics (i.e., abnormal performance for players may indicate doping), measurement errors (i.e., abnormal values could provide an indication of a measurement error), etc. Like missing value analysis, univariate and multivariate outlier analyses are also broader concepts. There are also different distance-based and probabilistic methods (like clustering analysis, genetic algorithm, etc.) that can be used to detect outliers in the high dimensional data sets. There is a huge theory behind the outlier detection methods and outlier analysis would be a stand alone topic of another course. Interested readers may refer to the "Outlier analysis, by Charu C. Aggarwal" for the theory behind the outlier detection methods (Aggarwal (2015)).

Approaches to Handling Outliers

Most of the ways to deal with outliers are similar to the methods of missing values like deleting them or imputing some values (i.e., mean, median, mode) instead. There are also other approaches specific to dealing with outliers like capping, transforming, and binning them. Here, we will discuss the common techniques used to deal with outliers. Some of the methods mentioned in this Module (like transforming and binning) will be covered in the next module (Module 7: Transform), therefore I won't go into the details of transforming and binning here.

Excluding or Deleting Outliers

Some authors recommend that if the outlier is due to data entry error, data processing error or outlier observations are very small in numbers, then leaving out or deleting the outliers would be used as a strategy. When this is the case, we can exclude/delete outliers in a couple different ways.

To illustrate, let's revisit the outliers in the `carat` variable. Remember that we already found the locations of the outliers in the `carat` variable using `which()` function. Intuitively, we can exclude these observations from the data using:

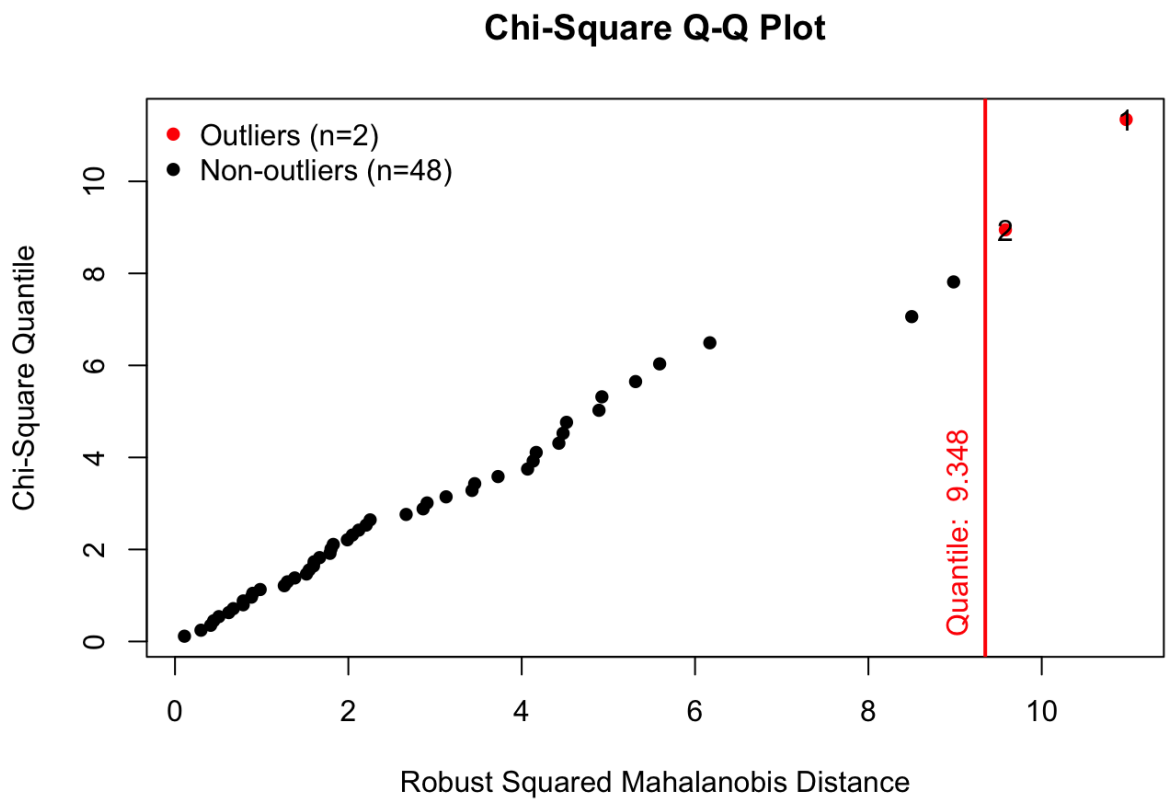
```
Carat_clean<- Diamonds$carat[ - which( abs(z.scores) >3 )]
```

Let's see another example on the previous `versicolor` data:

```
versicolor <- iris %>%  filter( Species == "versicolor" ) %>%  select(Sepal.L
ength, Sepal.Width,Petal.Length )

head(versicolor)
```

```
##   Sepal.Length Sepal.Width Petal.Length
## 1          7.0         3.2          4.7
## 2          6.4         3.2          4.5
## 3          6.9         3.1          4.9
## 4          5.5         2.3          4.0
## 5          6.5         2.8          4.6
## 6          5.7         2.8          4.5
```

```
results <- mvn(data = versicolor, multivariateOutlierMethod = "quan", showOut
liers = TRUE)
```




```
results$multivariateOutliers
```

```
##   Observation Mahalanobis Distance Outlier
## 1           1               10.974    TRUE
## 2           2                9.580    TRUE
```

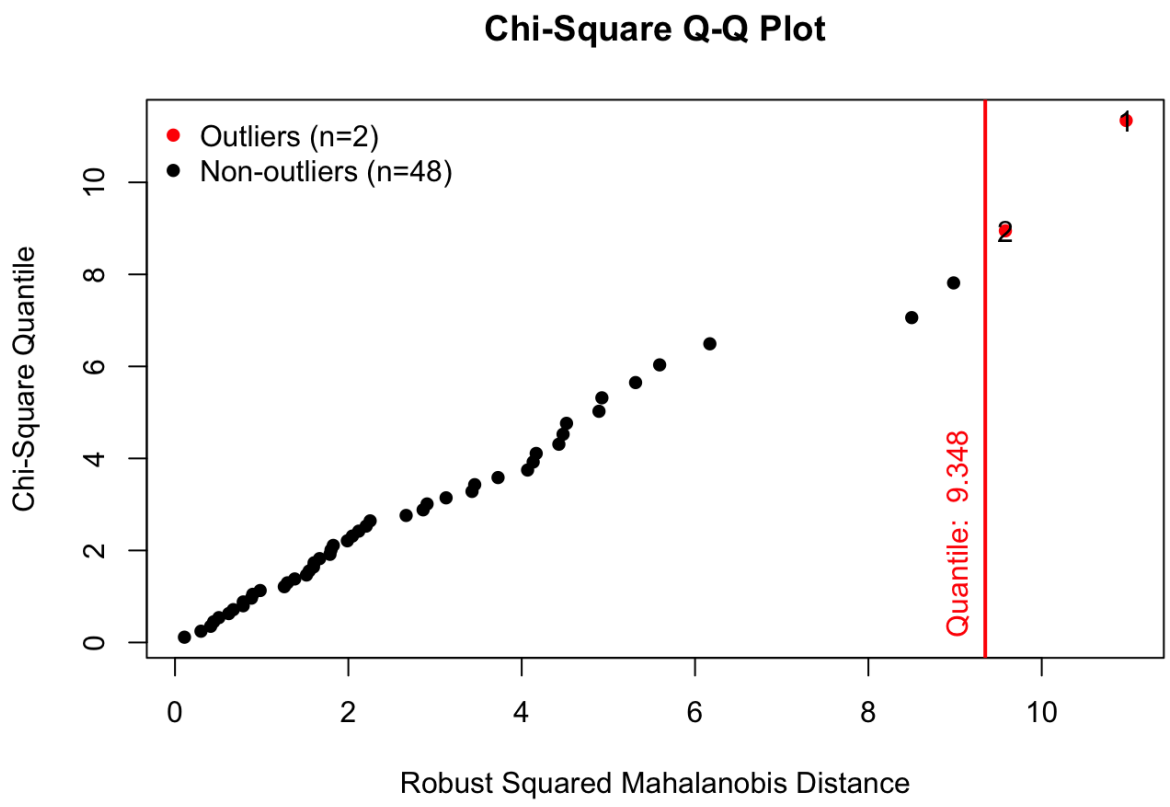
The Mahalonobis distance method provided us the locations of outliers in the data set. In our example the 1st and 2nd observations are the suggested outliers. Using the basic filtering and subsetting functions, we can easily exclude these two outliers:

```
# Exclude 1th and 2nd observations

versicolor_clean <- versicolor[ -c(1,2), ]

# Check the dimension and see outliers are excluded

dim(versicolor_clean)
```

```
## [1] 48  3
```

Note that, the `mvn()` function also has an argument called `showNewData = TRUE` to exclude the outliers. One can simply detect and remove outliers using the following argument:

```
versicolor_clean2 <- mvn(data = versicolor, multivariateOutlierMethod = "qua
n", showOutliers = TRUE, showNewData = TRUE)
```




```
# Prints the data without outliers

dim(versicolor_clean2$newData)
```

```
## [1] 48  3
```

Imputing

Like imputation of missing values, we can also impute outliers. We can use mean or median imputation methods to replace outliers. Before imputing values, we should analyse whether the outlier is a result of data entry/processing error. If the outlier is due to a data entry/processing error, we can go with imputing values.

Let's replace the two outlier values in `carat` variable with its mean by using Base R functions:

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

Diamonds$carat[ which( abs(z.scores) >3 )] <- mean(Diamonds$carat, na.rm = TR
UE)
```

Replacing outliers with the median or a user specified value can also be done using a similar approach. Note that, you may also prefer to write your own functions to deal with outliers.

Capping (a.k.a Winsorising)

Capping or winsorising involves replacing the outliers with the nearest neighbours that are not outliers. For example, for missing values that lie outside the outlier fences on a box-plot, we can cap it by replacing those observations outside the lower limit with the value of 5th percentile and those that lie above the upper limit, with the value of 95th percentile.

In order to cap the outliers we can use a user-defined function as follows (taken from: Stackoverflow (https://stackoverflow.com/questions/13339685/how-to-replace-outliers-with-the-5th-and-95th-percentile-values-in-r?utm_medium=organic&utm_source=google_rich_qa&utm_campaign=google_rich_qa)):

```
# Define a function to cap the values outside the limits

cap <- function(x){
    quantiles <- quantile( x, c(.05, .95 ) )
    x[ x < quantiles[1] ] <- quantiles[1]
    x[ x > quantiles[2] ] <- quantiles[2]
    x
}
```

To illustrate capping we will use the Diamond data set. In order to cap the outliers in the `carat` variable, we can simply apply our user-defined function to the `carat` variable as follows:

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

carat_capped <- Diamonds$carat %>% cap()
```

We can also apply this function to a data frame using `sapply` function. Here is an example of applying `cap()` function to a subset of the Diamonds data frame:

```
# Take a subset of Diamonds data using quantitative variables

Diamonds_sub <- Diamonds %>%  select(carat, depth, price)

# See descriptive statistics

summary(Diamonds_sub)
```

```
##      carat            depth           price
##  Min.   :0.2000   Min.   :43.00   Min.   :  326
##  1st Qu.:0.4000   1st Qu.:61.00   1st Qu.:  950
##  Median :0.7000   Median :61.80   Median : 2401
##  Mean   :0.7979   Mean   :61.75   Mean   : 3933
##  3rd Qu.:1.0400   3rd Qu.:62.50   3rd Qu.: 5324
##  Max.   :5.0100   Max.   :79.00   Max.   :18823
```

```
# Apply a user defined function "cap" to a data frame

Diamonds_capped <- sapply(Diamonds_sub, FUN = cap)

# Check summary statistics again

summary(Diamonds_capped)
```

```
##      carat            depth           price
##  Min.   :0.3000   Min.   :59.30   Min.   :  544
##  1st Qu.:0.4000   1st Qu.:61.00   1st Qu.:  950
##  Median :0.7000   Median :61.80   Median : 2401
##  Mean   :0.7812   Mean   :61.74   Mean   : 3806
##  3rd Qu.:1.0400   3rd Qu.:62.50   3rd Qu.: 5324
##  Max.   :1.7000   Max.   :63.80   Max.   :13107
```

Transforming and binning values:

Transforming variables can also eliminate outliers. Natural logarithm of a value reduces the variation caused by outliers. Binning is also a form of variable transformation. Transforming and binning will be covered in details in the next module (Module 7: Transform).

Additional Resources and Further Reading

As mentioned before, univariate and multivariate outlier analysis are broader concepts. Interested readers may refer to the "Outlier analysis, by Charu C. Aggarwal" for the theory behind the outlier detection methods (Aggarwal (2015)). Another useful resource is "R and Data Mining: Examples and Case Studies" by Yanchang Zhao (also available here (https://cran.r-project.org/doc/contrib/Zhao_R_and_data_mining.pdf)). Chapter 7 of this book covers univariate and multivariate outlier detection methods, outlier detection using clustering and outlier detection in time series.

The `outliers` package manual (https://cran.r-project.org/web/packages/outliers/outliers.pdf) includes useful functions for the commonly used outlier tests and the distance based approaches. This package can be used to detect univariate outliers.

I find the MVN package useful for detecting multivariate outliers as it provides many alternative visualisations in addition to the distance based metrics. The MVN package also includes different types of univariate and multivariate tests for normality. For more information and capabilities of the MVN package please refer to the paper by Korkmaz, Goksuluk, and Zararsiz (2014), which is available here (https://cran.r-project.org/web/packages/MVN/vignettes/MVN.pdf).

There are also other R packages for outlier detection and all might give different results. This blog by Antony Unwin (http://blog.revolutionanalytics.com/2018/03/outliers.html) compares different outlier detection methods available in R using the `OutliersO3` package.

References

Aggarwal, Charu C. 2015. "Outlier Analysis." In *Data Mining*, 237–63. Springer.

Hawkins, Douglas M. 1980. *Identification of Outliers*. Vol. 11. Springer.

Korkmaz, Selcuk, Dincer Goksuluk, and Gokmen Zararsiz. 2014. "MVN: An R Package for Assessing Multivariate Normality." *The R Journal* 6 (2): 151–62.