# Weekly Assignment 4

Sean Leggett - BDA201 Winter 2020
February 25, 2020

```
## load required libraries
library(car)
```

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

## Part 1:

What is the expected value thrown of a fair 12-sided dice?
What is the expected variance of a fair 12-sided dice?
Simulate 10,000 throws in R (using random function) from this dice and see if your answers match those above. Record the average value from the 10,000 throws and comment on the results

Answer:
The subjective probability would appear to be 6 as an expected value of a thrown 12-sided die.

However, empirical probability suggests we should calculate the actual answer. The expected value for calculating the expected value for a finite, discrete range, is the mean (average) of all options. In this case:

```
##establish object for reusability

die12 <- (1:12)
die12ev <- mean(die12)
die12ev
```

```
## [1] 6.5
```

The expected variance of a 12-sided die is calculated as follows:

```
die12var <- var(die12)
die12var
```

```
## [1] 13
```

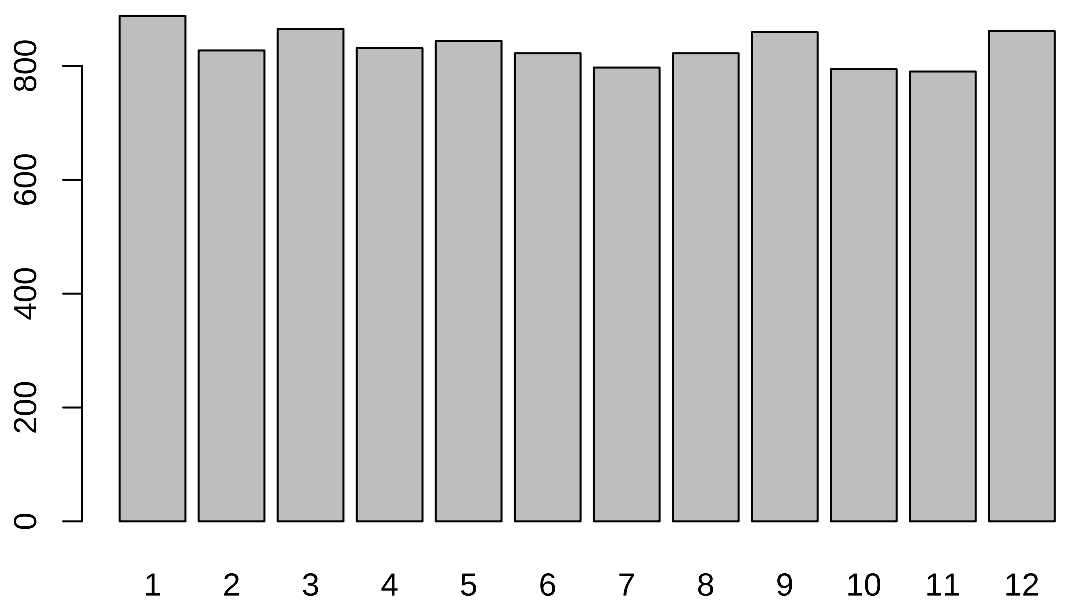
Following is a simulation of 10,000 rolls of a 12-sided die:

```
sample10K <- sample(die12, 10000, replace = TRUE)
samplemean <- mean(sample10K)
samplemean
```

```
## [1] 6.4461
```

While intuition and the human mind might not appreciate fractional means of whole numbers on a die, statistics does. When running this prior to knit, we obtained a mean of 6.4768 (will be slightly different in knitted pdf version as it will run again). This is very close to our calculated mean of 6.5. Plot of our rolls shows the following:

```
tab10K <- table(sample10K)
barplot(tab10K)
```



Fairly neat and uniform distributions across all possible outcomes. This is also a good illustration of the law of large numbers. Initially we could have great variability in the outcomes especially over say 5 or 10 rolls. However, after 10,000 rolls we see the outcomes reasoably even in distribution reinforcing the fact there is an equal chance on a roll or any number from 1 to 12.

## Part 2:

For each year from 2000 to 2010, record the following over the 12 months of the year from "HAMILTON A" station.

- Total snowfall
- Average of the Mean temp column (the sums and averages are reported at the bottom of the table).

1. Plot these two variables against time. Superimpose the mean of all 11 years over the plot as horizontal brown line.

2. Use an appropriate plot to show and verify whether the 11 values are possibly from a normal distribution.

3. Write a function that accepts a array of values and returns a array of z_scores. Use the function to display z_scores of the 11 values.

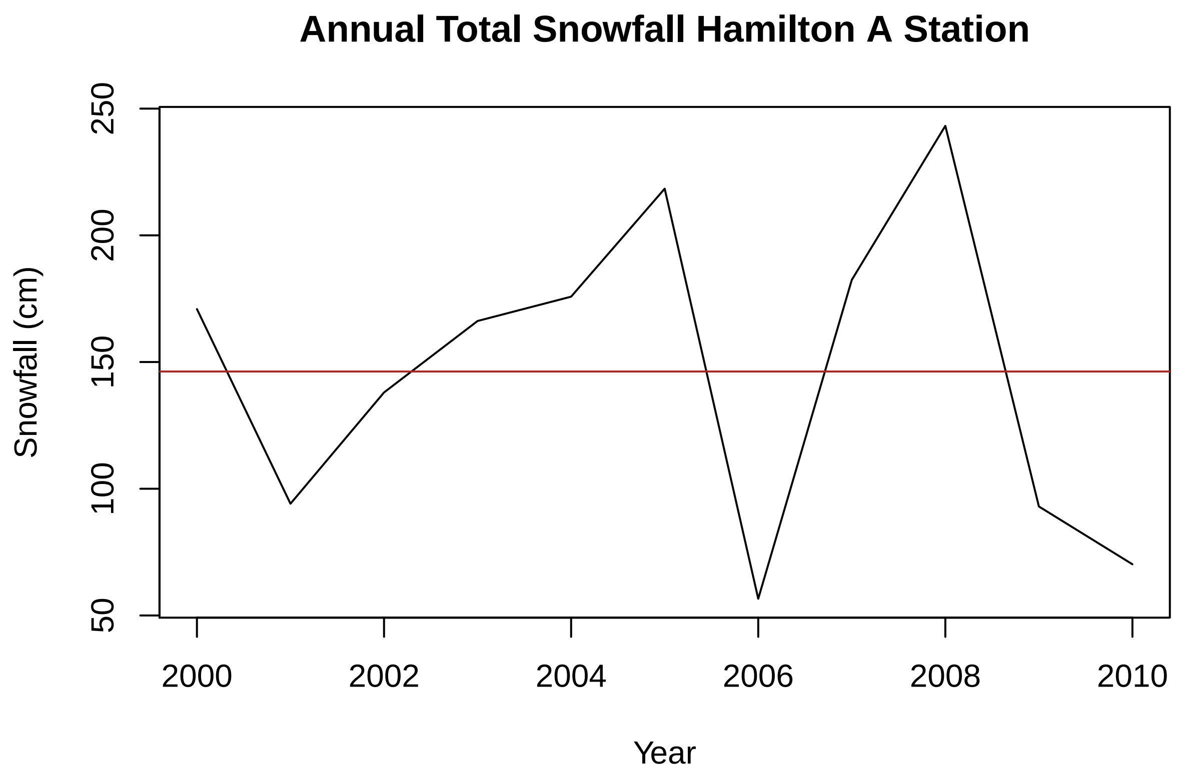
Answer:

See code below where dataframe built with variables of annual total snowfall, average mean temperature and year.

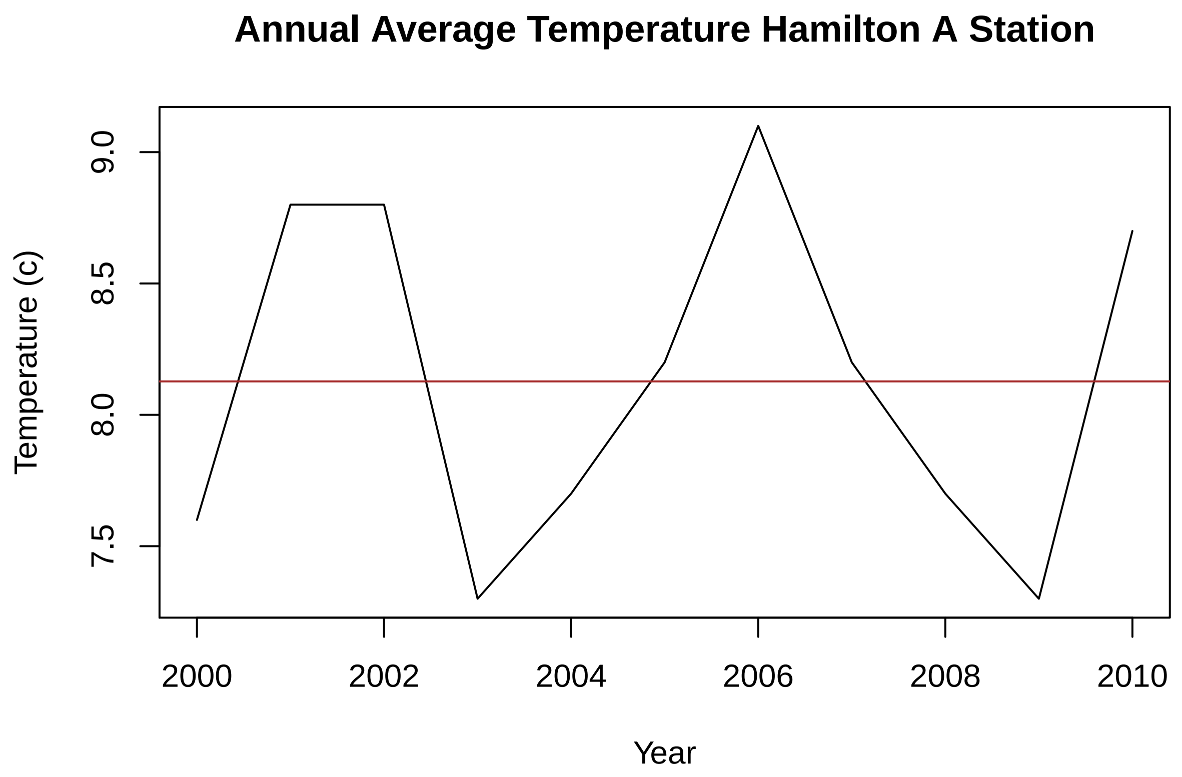
```r
## create dataframe from webpage summary data.
HamA <- data.frame("Year" = c(2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010),
                   "Snow(cm)" = c(170.9, 94.1, 138, 166.2, 175.8, 218.4, 56.6, 182.4, 243.2, 93, 70.2),
                   "AvgTemp(c)" = c(7.6, 8.8, 8.8, 7.3, 7.7, 8.2, 9.1, 8.2, 7.7, 7.3, 8.7)
                   )
HamA
```

```
##    Year Snow.cm. AvgTemp.c.
## 1  2000    170.9        7.6
## 2  2001     94.1        8.8
## 3  2002    138.0        8.8
## 4  2003    166.2        7.3
## 5  2004    175.8        7.7
## 6  2005    218.4        8.2
## 7  2006     56.6        9.1
## 8  2007    182.4        8.2
## 9  2008    243.2        7.7
## 10 2009     93.0        7.3
## 11 2010     70.2        8.7
```

```r
## plot snowfall with mean abline
plot(x = HamA$Year,
     y = HamA$Snow.cm.,
     type = "l",
     main = "Annual Total Snowfall Hamilton A Station",
    xlab = "Year",
    ylab = "Snowfall (cm)")
abline(h = mean(HamA$Snow.cm.),
       lty = 1,
       col = "brown")
```

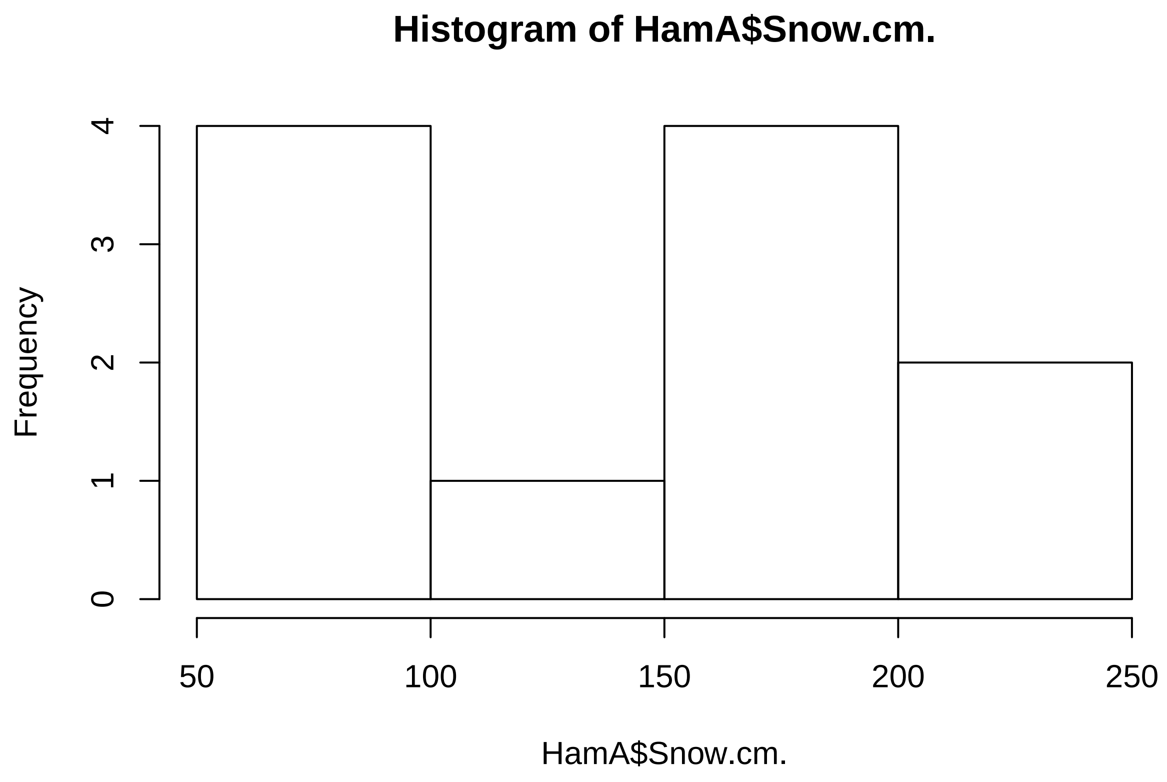


```r
## plot average temperature with mean abline
plot(x = HamA$Year,
     y = HamA$AvgTemp.c.,
     type = "l",
     main = "Annual Average Temperature Hamilton A Station",
    xlab = "Year",
    ylab = "Temperature (c)")
abline(h = mean(HamA$AvgTemp.c.),
       lty = 1,
       col = "brown")
```

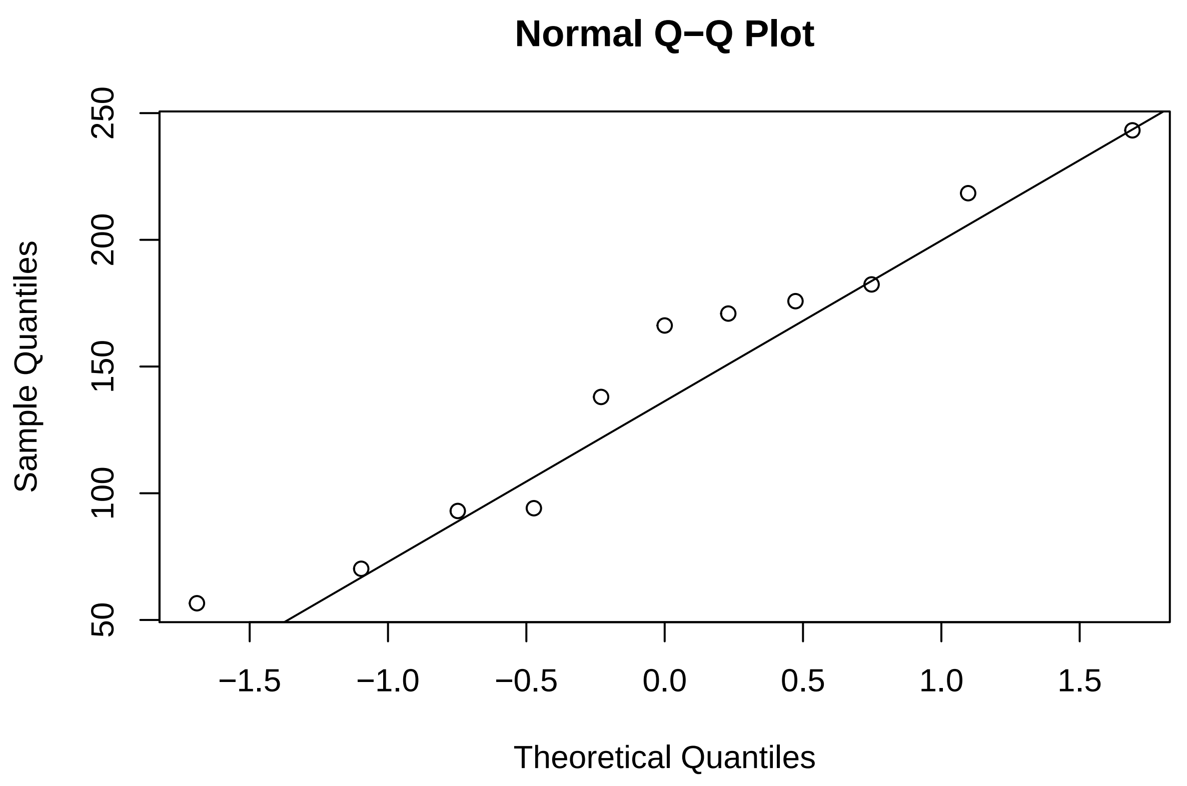


In terms of validating whether distributions are normal we tried a few different things with interesting results. The histogram does not look anything like a normal distribution but the qqPlot surprised us. These are relatively small samples however.

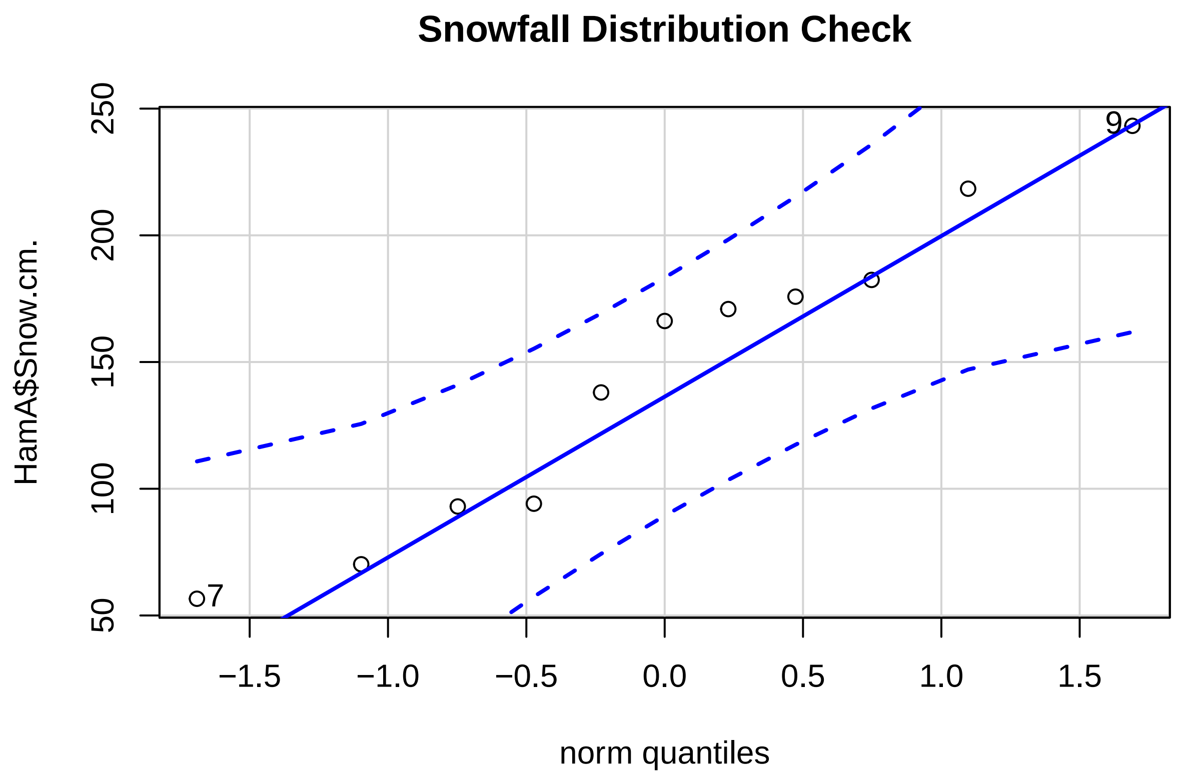
```
hist(HamA$Snow.cm.)
```



```
qqnorm(HamA$Snow.cm.)
qqline(HamA$Snow.cm.)
```



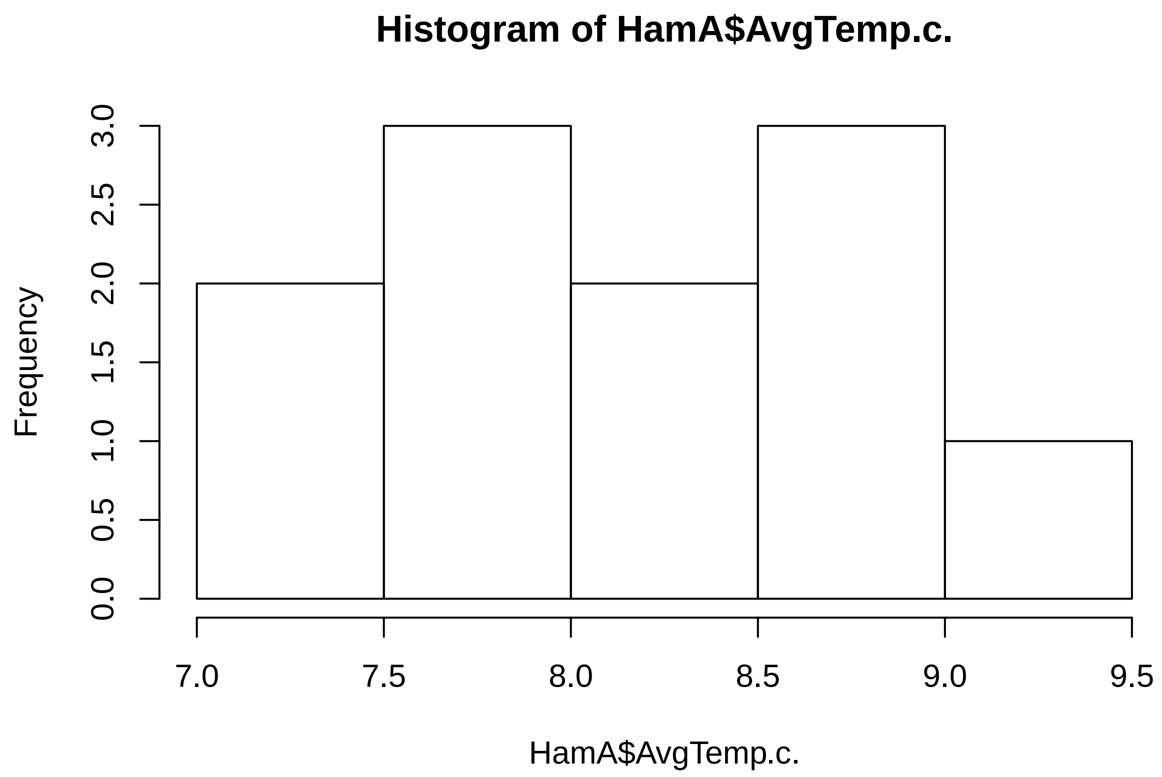
```r
qqPlot(HamA$Snow.cm.,
       main = "Snowfall Distribution Check")
```



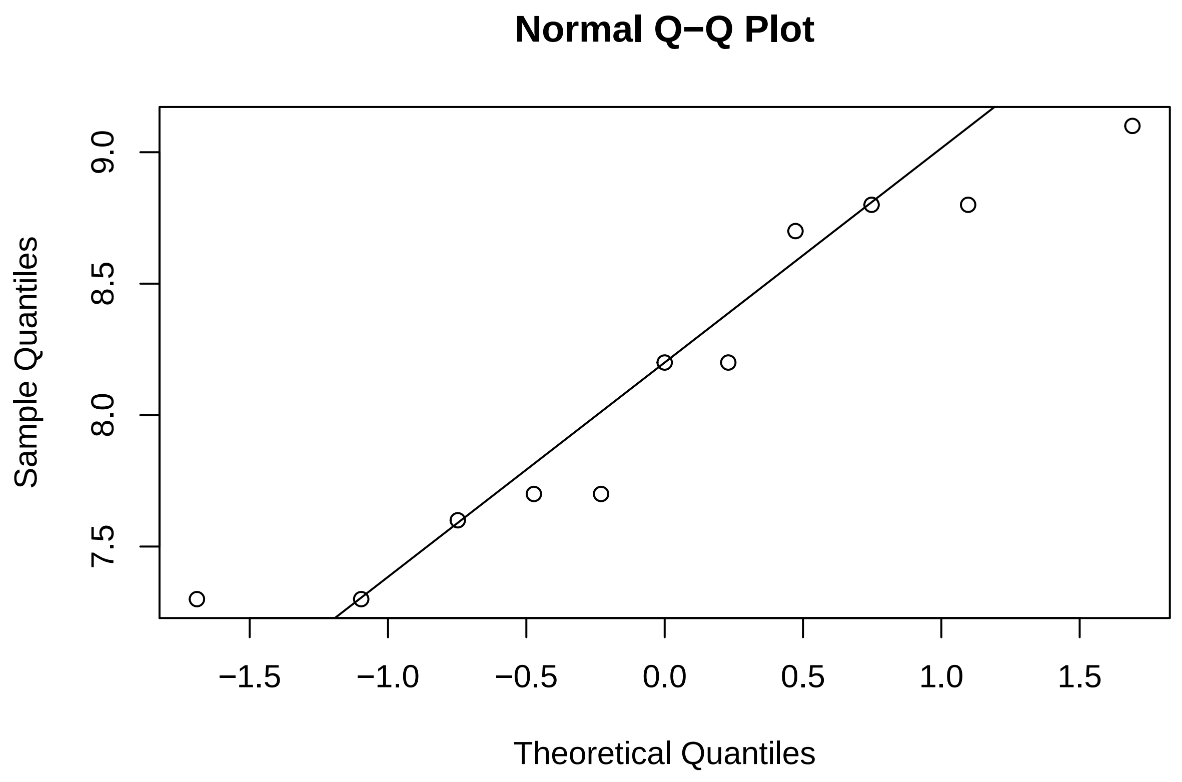
```
## [1] 9 7
```

And same exercise for temperatures. . .

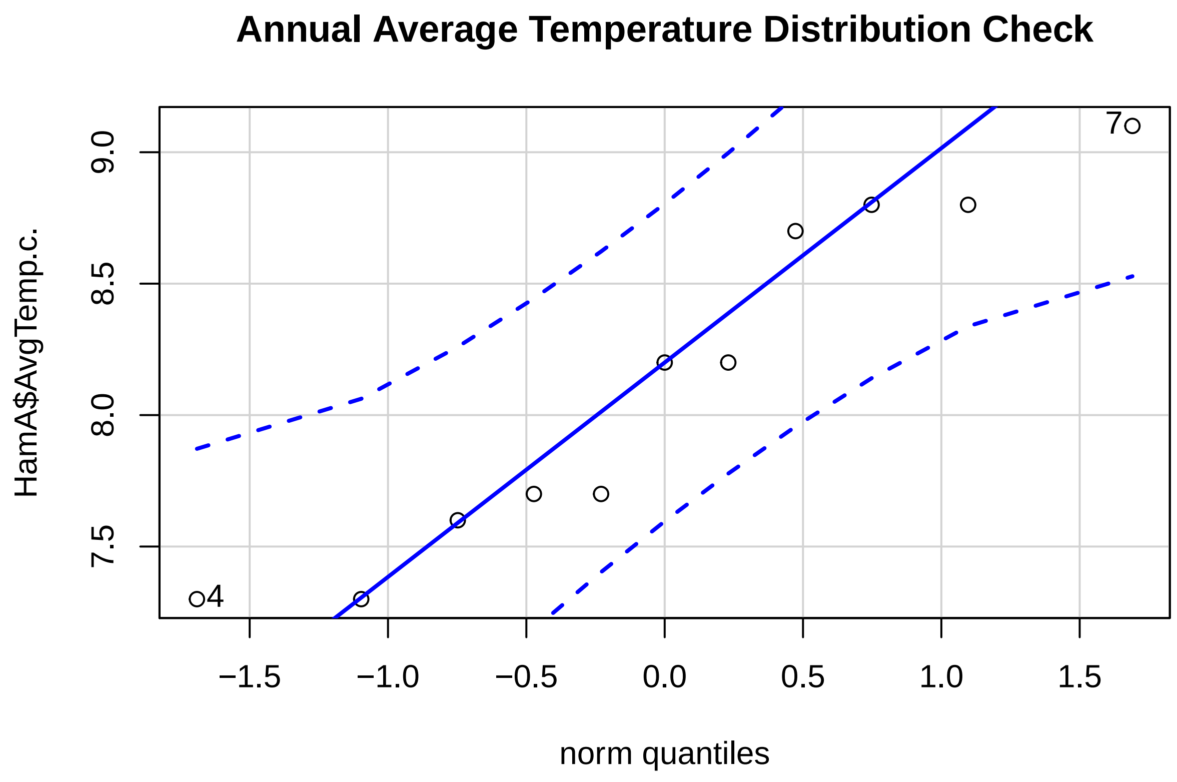
```r
hist(HamA$AvgTemp.c.)
```



```
qqnorm(HamA$AvgTemp.c.)
qqline(HamA$AvgTemp.c.)
```



```r
qqPlot(HamA$AvgTemp.c.,
       main = "Annual Average Temperature Distribution Check")
```



```
## [1] 7 4
```

While we expected the average temperatures to fit closely to a normal distribution (intuitively) it was surprising to see a similar result with snowfall. QQPlots are definitely a valuable tool.

Now a function to calculate and display z_scores.
The formula for calculating z scores requires population mean and population standard deviation. In this instance, we do not have population statistics and must rely upon the sample data to infer.

The formula we use is data point minus sample mean, divided by sample standard deviation.

We also interpret the question as requiring z scores for both snowfall and average temperature.

Other options exist to achieve this but we have tried to stay within the scope of the current lessons and assignment.

```
## function to accept array and calculate z scores
z_scores_array <- function(var_array) {
  x <- mean(var_array) #assign sample mean
  y <- sd(var_array)   #assign sample standard deviation
  #iterate through array calculating z score
  for (i in (var_array)) {
                w <- (i - x)/y
                print(w)
    }

}
```

```r
#create arrays from existing dataframe
snowarray <- array(HamA$Snow.cm.)
temparray <- array(HamA$AvgTemp.c.)
```

```r
#apply function to snowfall data
snowz <- z_scores_array(snowarray)
```

```
## [1] 0.4038906
## [1] -0.8547107
## [1] -0.1352758
## [1] 0.3268669
## [1] 0.484192
## [1] 1.182322
## [1] -1.469262
## [1] 0.5923531
## [1] 1.588746
## [1] -0.8727375
## [1] -1.246385
```

```r
#apply function to temperature data
tempz <- z_scores_array(temparray)
```

```
## [1] -0.8134234
## [1] 1.037816
## [1] 1.037816
## [1] -1.276233
## [1] -0.6591535
## [1] 0.1121963
## [1] 1.500626
## [1] 0.1121963
## [1] -0.6591535
## [1] -1.276233
## [1] 0.8835461
```

Reference Material:
1. Course Content
2. Sams Teach Yourself R in 24 Hours, Andy Nicholls, Richard Pugh, Aimee Gott. Sams, 2016.