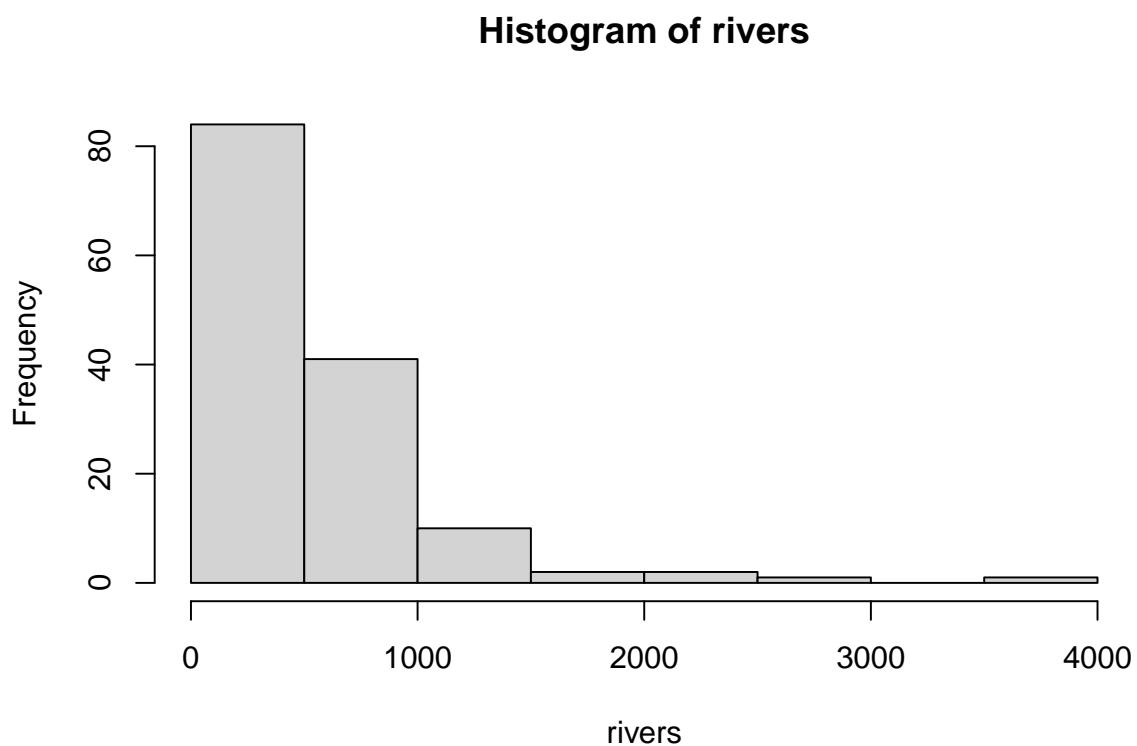


Contents

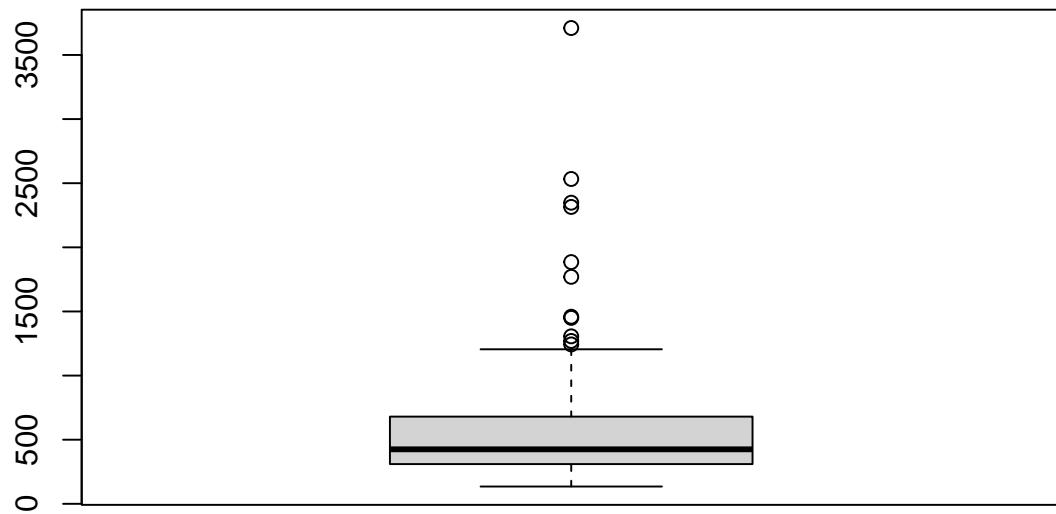
1	Exercise 5.1	2
2	Exercise 5.2	7
3	Exercise 5.3	11

1 Exercise 5.1

```
rivers <- read.table("rivers.txt")
rivers <- rivers[,1]
hist(rivers)
```

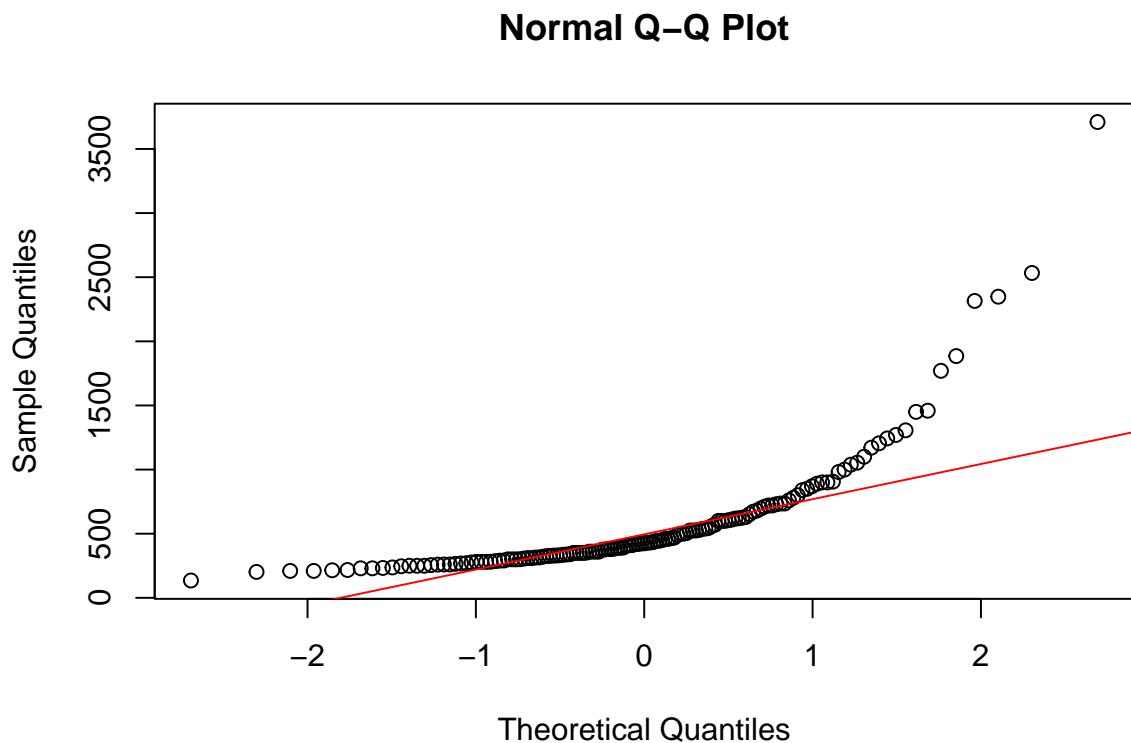


```
boxplot(rivers)
```



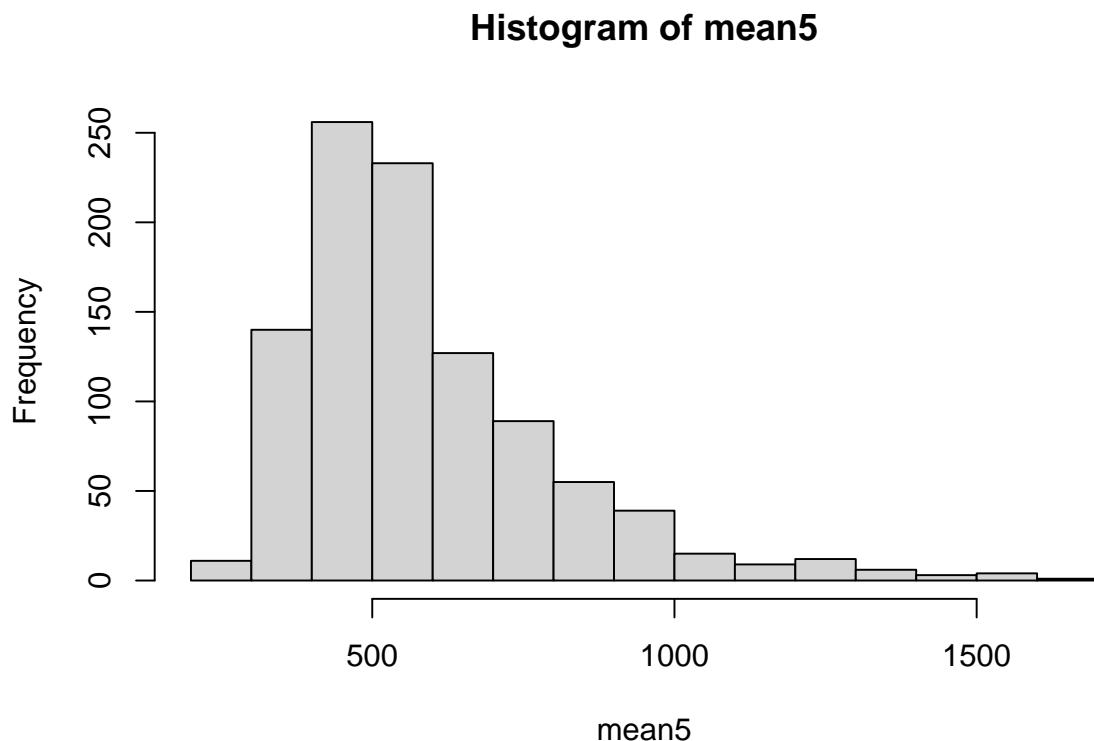
Even without a qqplot, the data is clearly not normal, based on both the histogram and boxplot. Either way, to make a q-q plot to compare it to the normal distribution, we will use `qqnorm()`.

```
qqnorm(rivers)
qqline(rivers, col="red")
```

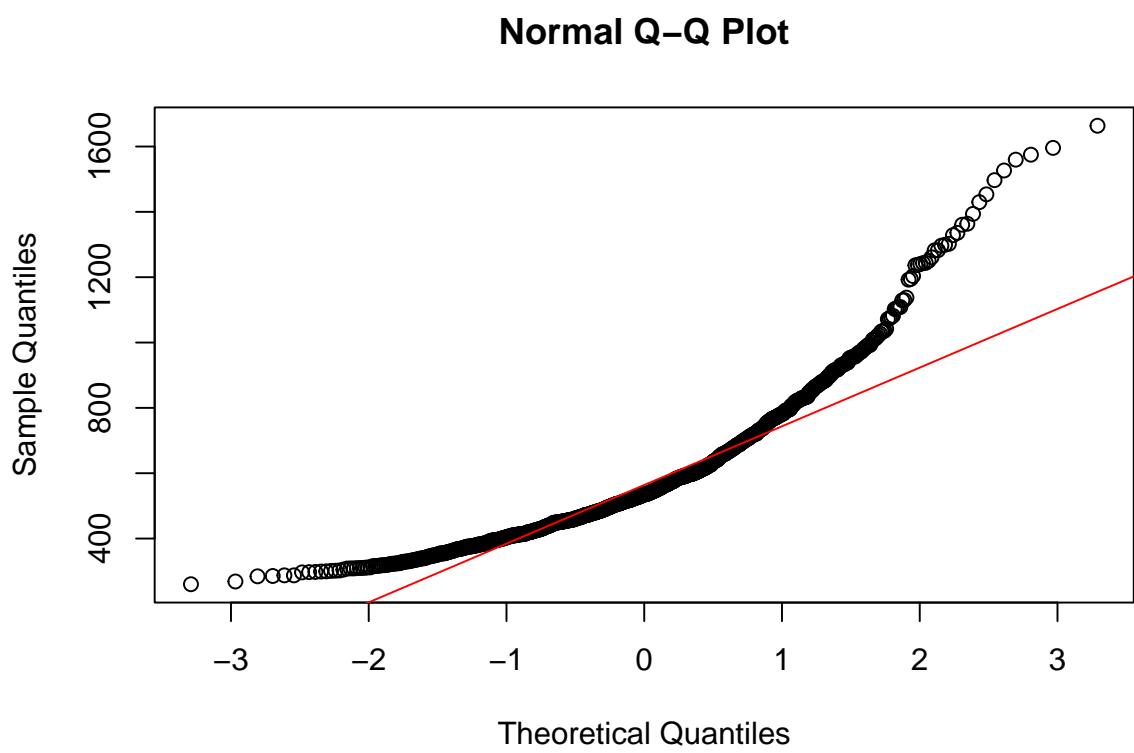


With 5 samples, the sample mean still has a long right tail (and a short left tail).

```
set.seed(123)
mean5 <- numeric(1000)
for (i in 1:1000) mean5[i] <- mean(sample(rivers, size=5))
hist(mean5)
```

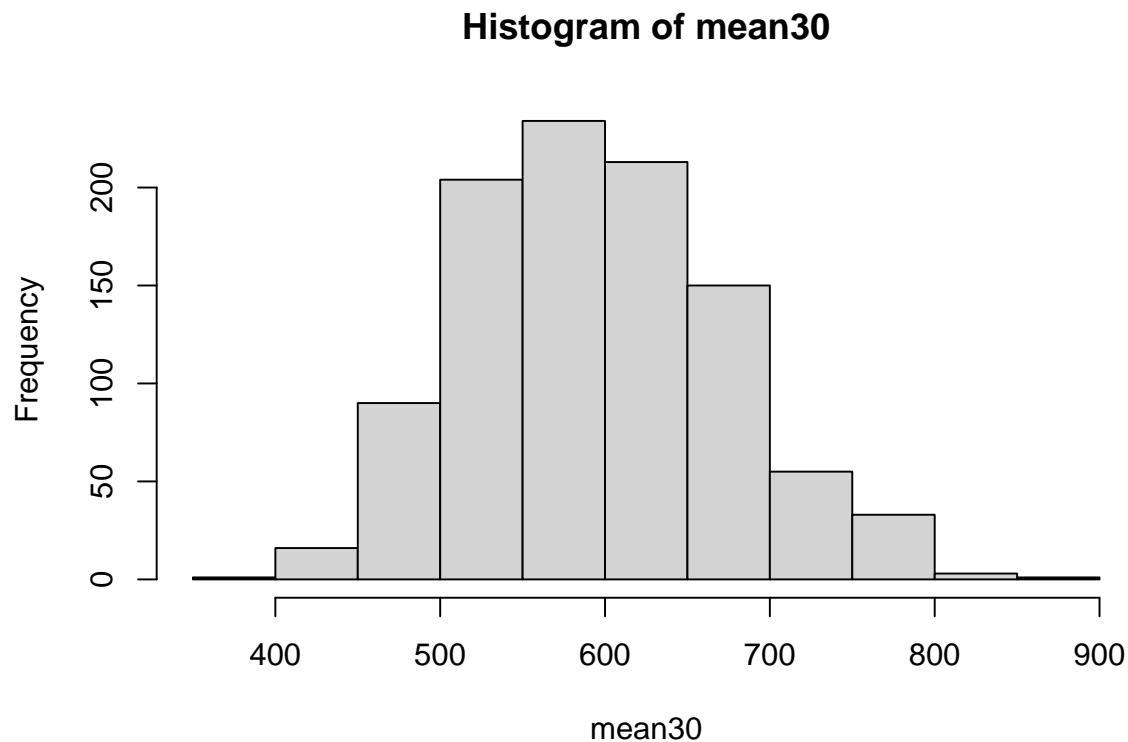


```
qqnorm(mean5)
qqline(mean5, col="red")
```

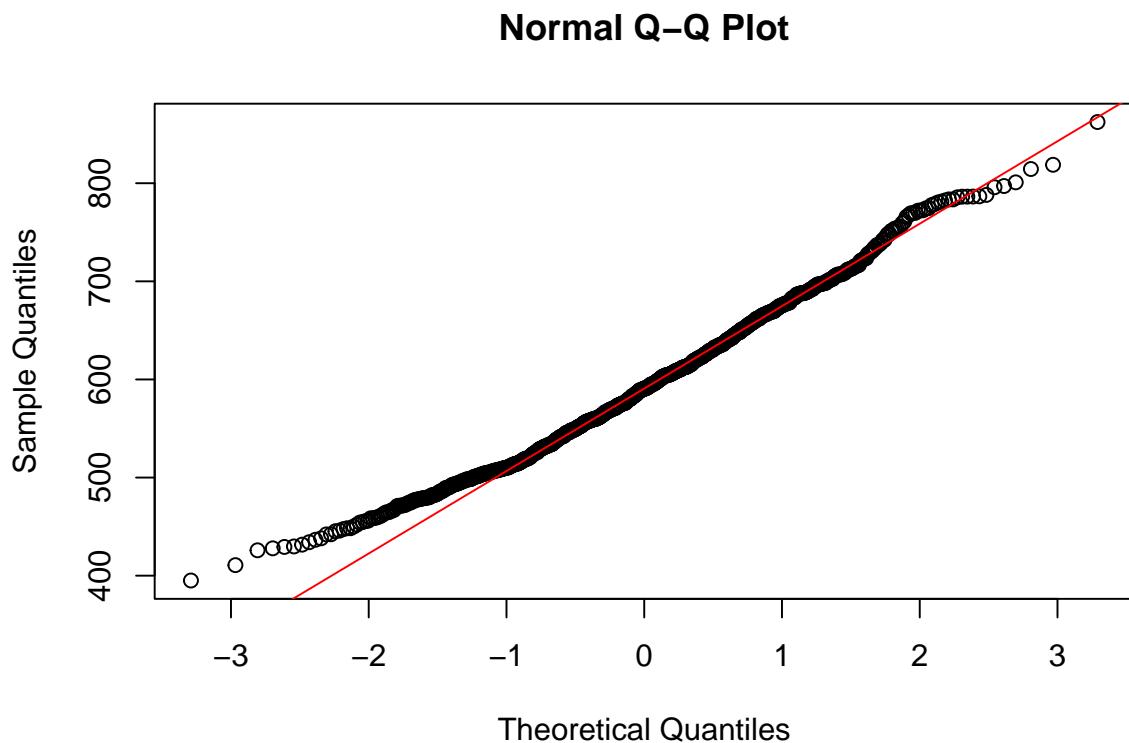


With 30 samples, the sample mean is more normal, but the short left tail persists.

```
mean30 <- numeric(1000)
for (i in 1:1000) mean30[i] <- mean(sample(rivers, size=30))
hist(mean30)
```



```
qqnorm(mean30)
qqline(mean30, col="red")
```



2 Exercise 5.2

This exercise involves looking at a dataset that is not in base R, and is not read from a text file - it is a dataset created in an **external package**.

To install packages that bring in functionality not available in base R, we use the `install.packages()` function. To import **all** the contents of the package, we use `library()`. This includes any functions that may override other function definitions.

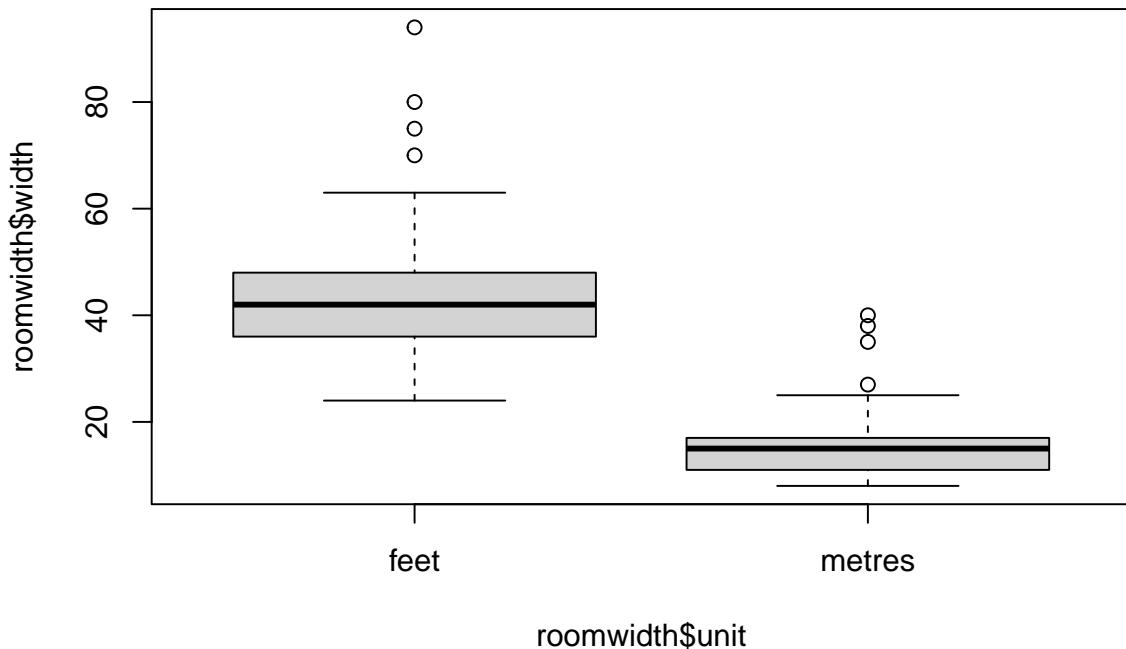
```
install.packages("HSAUR")
# library(HSAUR)
```

Since all we need is a single dataset from the package, we can import that directly - one such way is shown below. The double colon `::` means that we are accessing from the HSAUR package the definition of the `roomwidth` variable.

```
roomwidth <- HSAUR::roomwidth
str(roomwidth)

## 'data.frame':    113 obs. of  2 variables:
##   $ unit : Factor w/ 2 levels "feet","metres": 2 2 2 2 2 2 2 2 2 ...
##   $ width: num  8 9 10 10 10 10 10 11 11 ...
```

```
boxplot(roomwidth$width ~ roomwidth$unit)
```

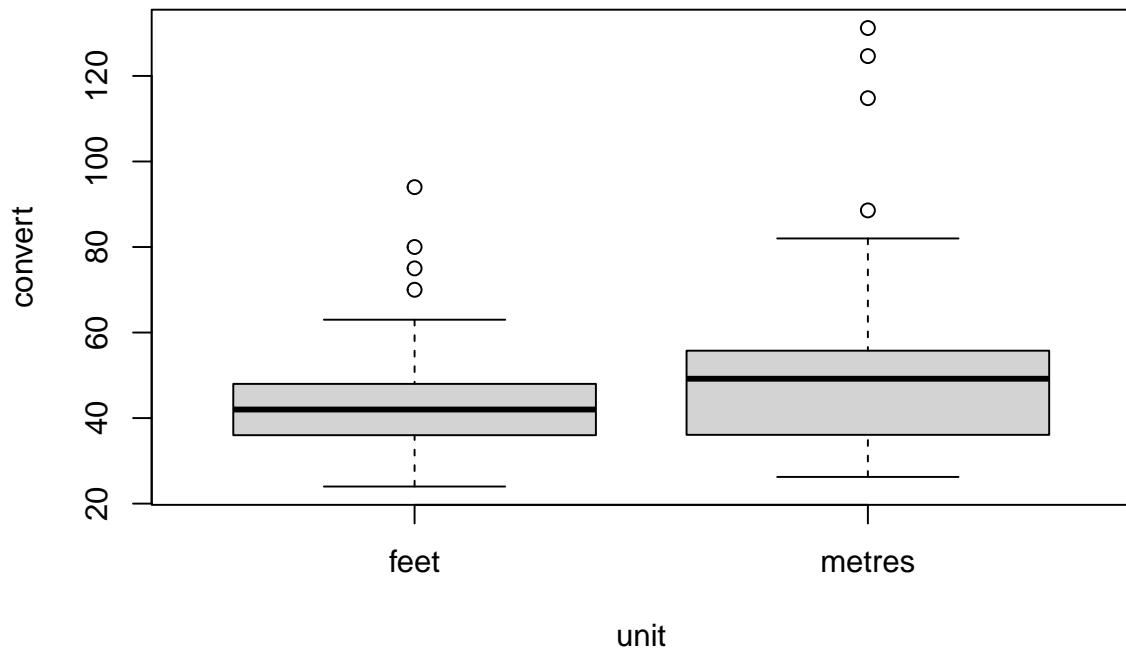


```
# equivalently, boxplot(width ~ unit, roomwidth)
```

There is nothing particularly ‘wrong’ with this plot, in my opinion. It just displays the data you asked it to.

The suggested solution: This plot gives the wrong suggestion that the estimates using “metres” would be lower than those using “feet”. The fact is that one metre is about 3.28 feet.

```
cfactor <- ifelse(roomwidth$unit == "feet", 1, 3.28)
roomwidth$convert <- roomwidth$width * cfactor
boxplot(convert ~ unit, roomwidth)
```



```

roomwidth.feet <- roomwidth$convert[roomwidth$unit == "feet"]
roomwidth.metres <- roomwidth$convert[roomwidth$unit == "metres"]
mu.feet <- mean(roomwidth.feet)
mu.metres <- mean(roomwidth.metres)
sd.feet <- sd(roomwidth.feet)
sd.metres <- sd(roomwidth.metres)
n.feet <- length(roomwidth.feet)
n.metres <- length(roomwidth.metres)
stats.feet <- data.frame(mu=mu.feet, sd=sd.feet, n=n.feet)
stats.metres <- data.frame(mu=mu.metres, sd=sd.metres, n=n.metres)
rbind(feet = stats.feet, metres=stats.metres)

```

```

## # A tibble: 2 x 3
##       mu     sd     n
##   <dbl> <dbl> <int>
## 1  43.7  12.5    69
## 2  52.6  23.4    44

```

Based on the standard deviations of the estimations in each unit, the measurement in feet has lower standard deviation, and therefore suggests that the measurement in feet was a more accurate estimate than in metres.

To get the confidence intervals of each dataset, since we do not have the true variances/standard deviations of the population, the sample mean follows a t-distribution. The t-distribution must specify the degrees of freedom, which is 1 less than the number of data points.

The derivation is similar to confidence intervals using the standard normal distribution.

```
t <- qt(0.975, stats.feet$n - 1) * stats.feet$sd / sqrt(stats.feet$n)
c(stats.feet$mu - t, stats.feet$mu + t)
```

```
## [1] 40.69344 46.69786
```

```
t.test(roomwidth.feet)
```

```
##
##  One Sample t-test
##
## data: roomwidth.feet
## t = 29.043, df = 68, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
## 40.69344 46.69786
## sample estimates:
## mean of x
## 43.69565
```

```
t <- qt(0.975, stats.metres$n - 1) * stats.metres$sd / sqrt(stats.metres$n)
c(stats.metres$mu - t, stats.metres$mu + t)
```

```
## [1] 45.42982 59.67927
```

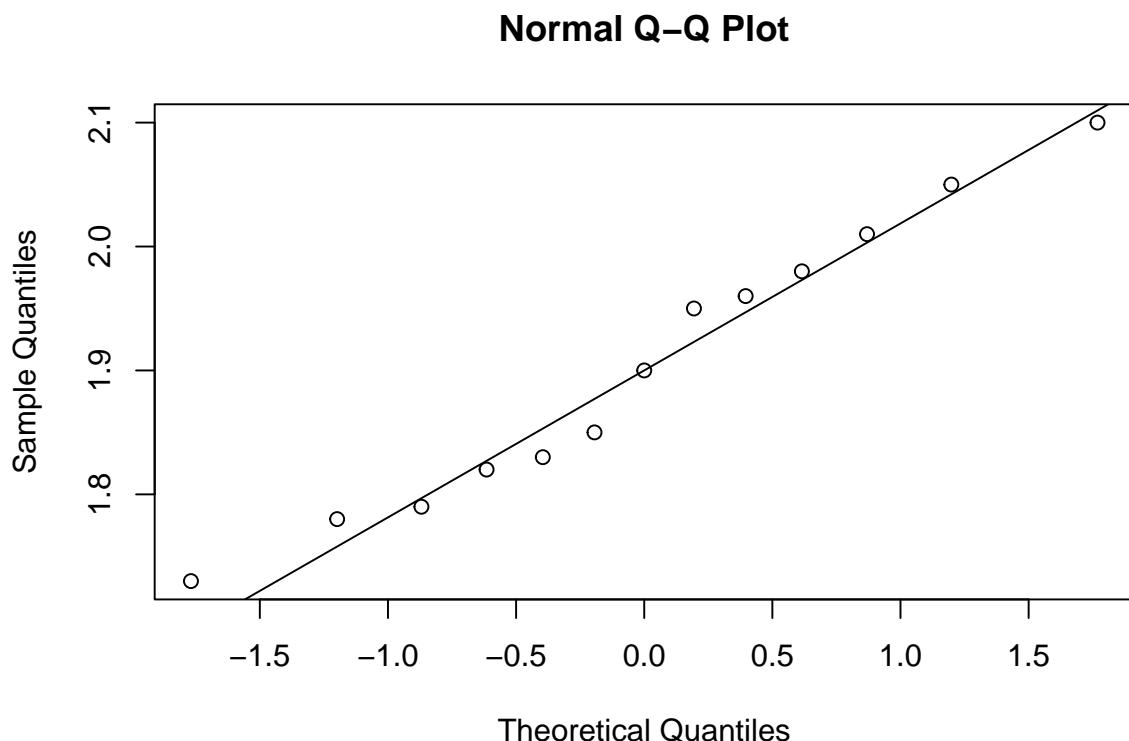
```
t.test(roomwidth.metres)
```

```
##
##  One Sample t-test
##
## data: roomwidth.metres
## t = 14.876, df = 43, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
## 45.42982 59.67927
## sample estimates:
## mean of x
## 52.55455
```

3 Exercise 5.3

```
coffee <- c(1.95, 1.78, 2.10, 1.82, 1.73, 2.01, 1.83,
           1.90, 2.05, 1.85, 1.96, 1.98, 1.79)

qqnorm(coffee)
qqline(coffee)
```



```
shapiro.test(coffee)
```

```
##
##  Shapiro-Wilk normality test
##
## data: coffee
## W = 0.96491, p-value = 0.8271
```

The q-q plot does not show any significant deviations of the sample quantiles from the theoretical quantiles.

The Shapiro-Wilk test is a statistical test that the given data is normal. The alternative hypothesis is that the data is not normal. In this case, since $p = 0.8271 > 0.1$ (or whatever level of significance you want), there is insufficient statistical evidence that the data is not normal.

```
n <- length(coffee)
xbar <- mean(coffee)
s <- sd(coffee)
n; xbar; s
```

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

```
## [1] 1.903846
```

```
## [1] 0.1140569
```

Let X represent the amount of espresso poured. From the data, we suppose that $X \sim N(1.904, 0.114^2)$ using \bar{x} and s as estimates for the mean and standard deviation. Thus, with a sample size of 13, we have $\bar{X} \sim N(1.904, 0.114^2/\sqrt{13})$.

We can actually use this distribution to get the confidence interval assuming the data is normal using the quantile function. The t-test doesn't accept parameterisation using the standard deviation, so we can't get the quantiles directly.

```
alpha <- 0.1
c(qnorm(alpha/2, xbar, s/sqrt(n)), qnorm(1-alpha/2, xbar, s/sqrt(n)))
```

```
## [1] 1.851813 1.955879
```

If not, we can get the z-statistic as usual.

```
alpha = 0.1
z <- qnorm(1-alpha/2)
t <- qt(1-alpha/2, n-1)

c(xbar - s/sqrt(n)*z, xbar + s/sqrt(n)*z)
```

```
## [1] 1.851813 1.955879
```

```
c(xbar - s/sqrt(n)*t, xbar + s/sqrt(n)*t)
```

```
## [1] 1.847466 1.960226
```

```
t.test(coffee, conf.level=1-alpha)
```

```
##  
##  One Sample t-test  
##  
## data: coffee  
## t = 60.184, df = 12, p-value = 2.928e-16  
## alternative hypothesis: true mean is not equal to 0  
## 90 percent confidence interval:  
## 1.847466 1.960226  
## sample estimates:  
## mean of x  
## 1.903846
```

For the one-sided interval, note that one side is the minimum (or maximum) value the variable can take. In practice, this should be 0, since this is a physical quantity (we can't have negative espresso volumes), but we assumed it follows a normal distribution which has infinite range.

```
alpha = 0.1  
z <- qnorm(1-alpha)  
t <- qt(1-alpha, n-1)  
  
c(-Inf, xbar + s/sqrt(n)*z)  
  
## [1] -Inf 1.944386  
  
c(-Inf, xbar + s/sqrt(n)*t)  
  
## [1] -Inf 1.946748  
  
t.test(coffee, conf.level=1-alpha, alternative="less")
```

```
##  
##  One Sample t-test  
##  
## data: coffee  
## t = 60.184, df = 12, p-value = 1  
## alternative hypothesis: true mean is less than 0  
## 90 percent confidence interval:  
## -Inf 1.946748  
## sample estimates:  
## mean of x  
## 1.903846
```