

Tests for violations of normality

XDASI Fall 2021

10/28/2021

Contents

Is it ok to perform a t -test on my data?	1
Quantile-Quantile plots	2
Normal QQ plot	2
More general QQ plot	3
ppoints	3
quantile	4
qqplot	4
Shapiro test for normality	5
Data transformation	5
Example: Trees	7
Example: Plasma triglyceride levels	9
Histograms and QQ plots	9
Shapiro-Wilk test for normality	11
t -tests	12
95% confidence intervals	13
Manual calculations	13
Extracted from t -test output	14

Is it ok to perform a t -test on my data?

So far we have used t -tests to compare two samples. These depend on the assumption that data are normally distributed. How do we know if this is the case?

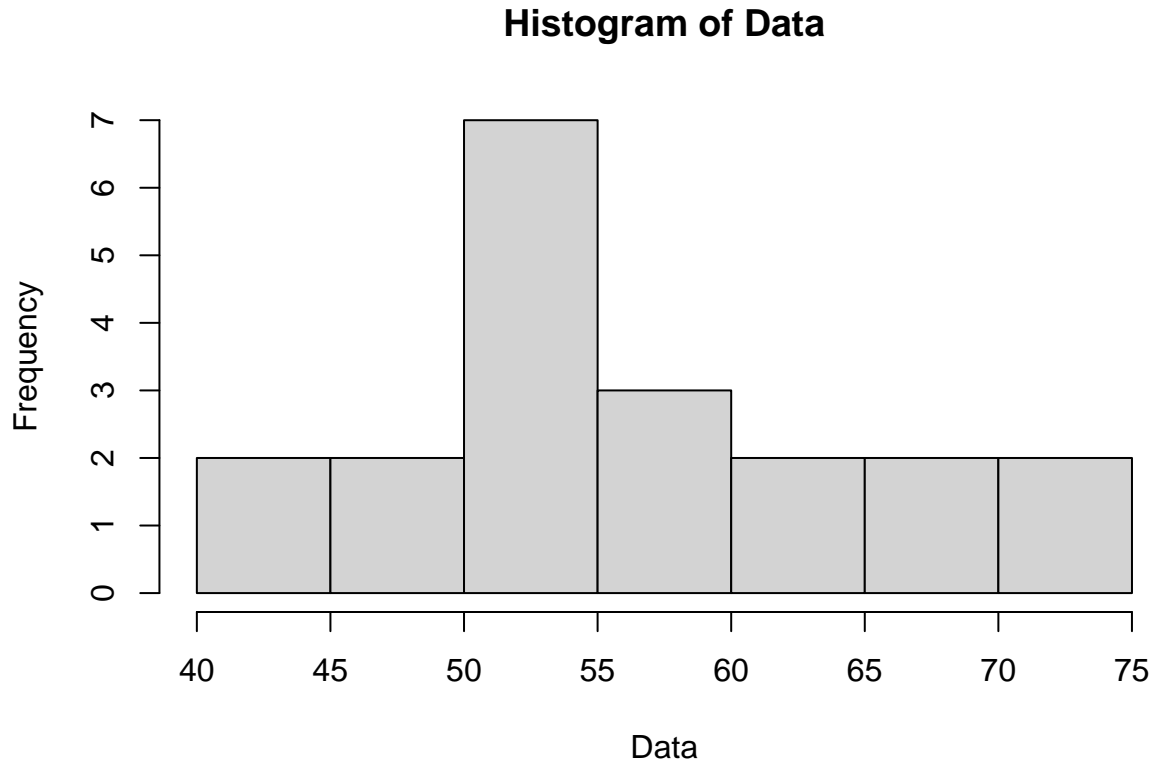
Let's return to our simple case study where a drug has been provided to 10 random patients (test subjects) and, as a control, a placebo pill was given to 10 other random patients. Measurements were collected for each condition and the question is:

Is there a significant difference between the subjects who were given the placebo and the those who were given the drug?

Let's first look at all the data together as a population. We will combine the values and draw a simple histogram.

```
Placebo = c(54,51,58,44,55,52,42,47,58,46)
Drug = c(54,73,53,70,73,68,52,65,65,60)

Data = c(Placebo, Drug)
Data_sd = sd(Data)
Data_mean = mean(Data)
hist(Data, breaks=10)
```



Do these data look “sufficiently normal” to perform regular parametric tests on them? How can we tell?

Quantile-Quantile plots

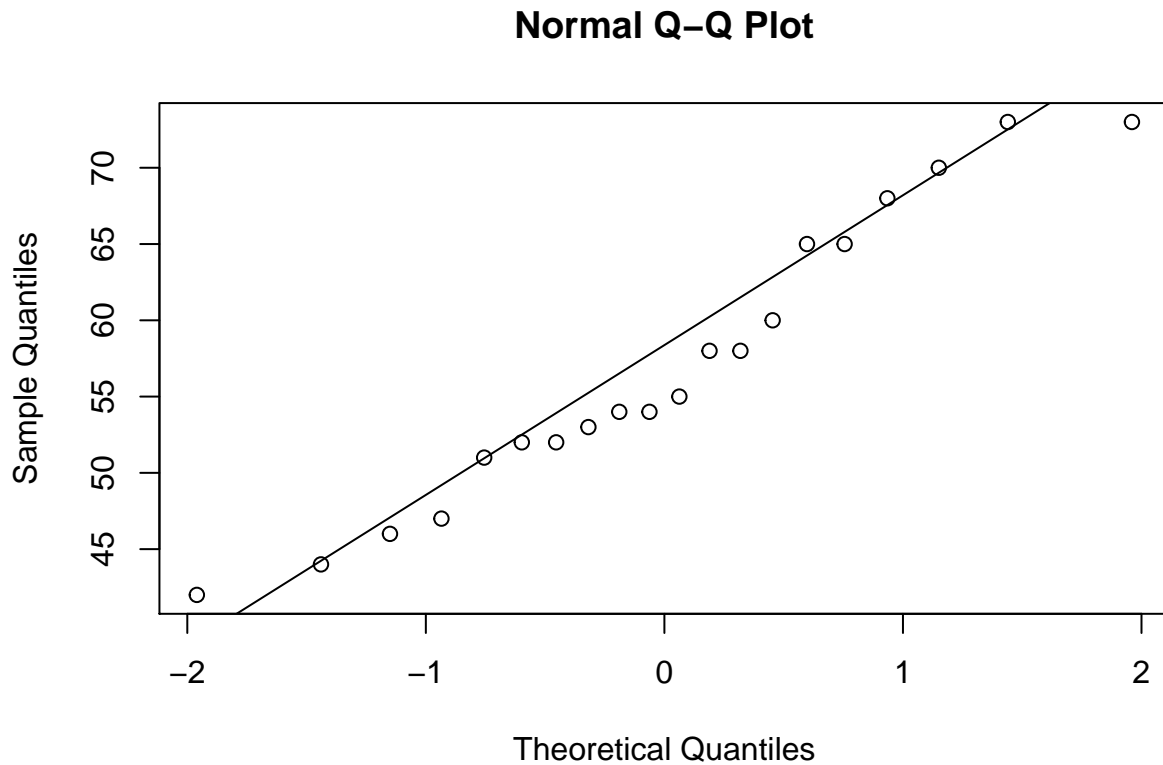
Normal QQ plot

We can check graphically to see how well an observed set of values matches a theoretical normal distribution using the `qqnorm` and `qqline` functions.

The theoretical distribution is generated using the same number of datapoints as the sample data, but distributed evenly across probability quantiles for a standard normal distribution. The corresponding z -scores are then plotted against the sample data.

The closer the observed data points fall on the “ideal” line, the more closely they resemble normally distributed data. Of course, due to sampling variation, the actual data will usually not fall exactly on the line, but random samples drawn from a normal distribution should follow the line at least approximately.

```
qqnorm(Data)
qqline(Data)
```



More general QQ plot

A more generic function is `qqplot` function, which can be used to compare data to theoretical values based on any distribution we want. To make such a plot requires a few steps:

ppoints The `ppoints` function takes the number of values you want, which should match the length of the dataset you want to test. The function returns the same number of equally spaced quantiles for the cumulative probability between 0 and 1.

We can then use the `qnorm` function to get the predicted z -scores of the theoretical normal distribution for the corresponding quantiles.

```
my_probs = ppoints(length(Data))
my_probs
```

```
## [1] 0.025 0.075 0.125 0.175 0.225 0.275 0.325 0.375 0.425 0.475 0.525 0.575
## [13] 0.625 0.675 0.725 0.775 0.825 0.875 0.925 0.975
```

```
my_quant = qnorm(my_probs)
my_quant
```

```
## [1] -1.95996398 -1.43953147 -1.15034938 -0.93458929 -0.75541503 -0.59776013
## [7] -0.45376219 -0.31863936 -0.18911843 -0.06270678 0.06270678 0.18911843
## [13] 0.31863936 0.45376219 0.59776013 0.75541503 0.93458929 1.15034938
## [19] 1.43953147 1.95996398
```

quantile Now to plot the corresponding values from the observed data distribution, we can use the `quantile` function.

In addition to the sample data, the `quantile` function also takes the (evenly spaced) cumulative probabilities that we have stored in `my_probs`, which it will use to infer values based on the observed data.

```
data_quant = quantile(Data, my_probs)
data_quant
```

```
## 2.5% 7.5% 12.5% 17.5% 22.5% 27.5% 32.5% 37.5% 42.5% 47.5% 52.5%
## 42.950 44.850 46.375 48.300 51.275 52.000 52.175 53.125 54.000 54.025 54.975
## 57.5% 62.5% 67.5% 72.5% 77.5% 82.5% 87.5% 92.5% 97.5%
## 57.775 58.000 59.650 63.875 65.000 67.025 69.250 71.725 73.000
```

qqplot When we now plot these against each other, we get a very similar plot to what we saw with `qqnorm`, except now we are using imputed values based on the input data.

Again, values based on the observed data are plotted on the y-axis, and normalized values for the theoretical data are plotted on the x-axis.

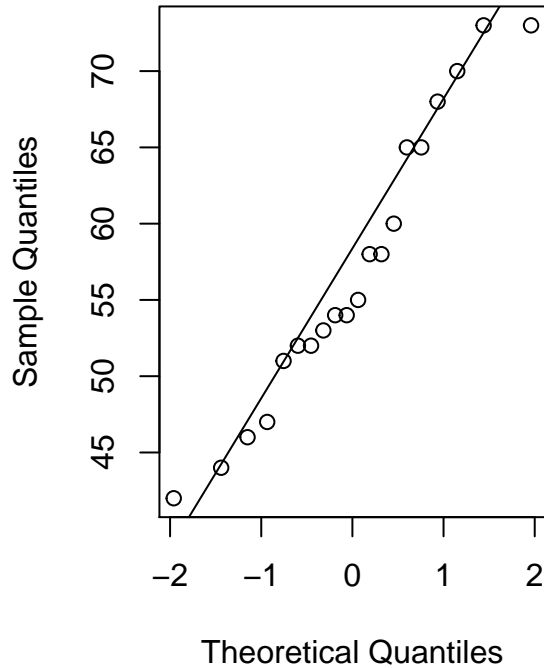
For normally distributed data, `qqnorm` and `qqplot` should give very similar results (though they will usually not be exactly the same since `qqnorm` goes through an extra imputation step). Both are shown below for easier comparison.

```
par(mfrow = c(1,2))

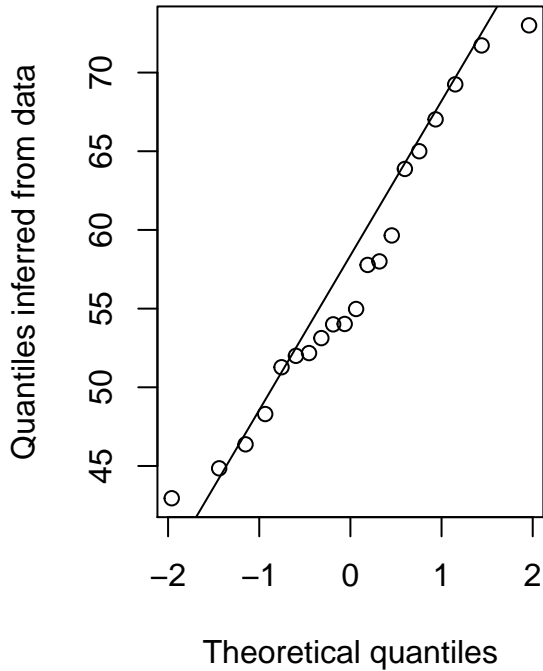
## qqnorm plot
qqnorm(Data)
qqline(Data)

## qqplot plot
qqplot(my_quant, data_quant,
       xlab="Theoretical quantiles", ylab="Quantiles inferred from data",
       main="General Q-Q Plot")
qqline(Data)
```

Normal Q-Q Plot



General Q-Q Plot



Shapiro test for normality

The Shapiro test performs a **goodness of fit** test using the mean and standard deviation of the data. The null hypothesis is that the data are normally distributed.

```
shapiro.test(Data)
```

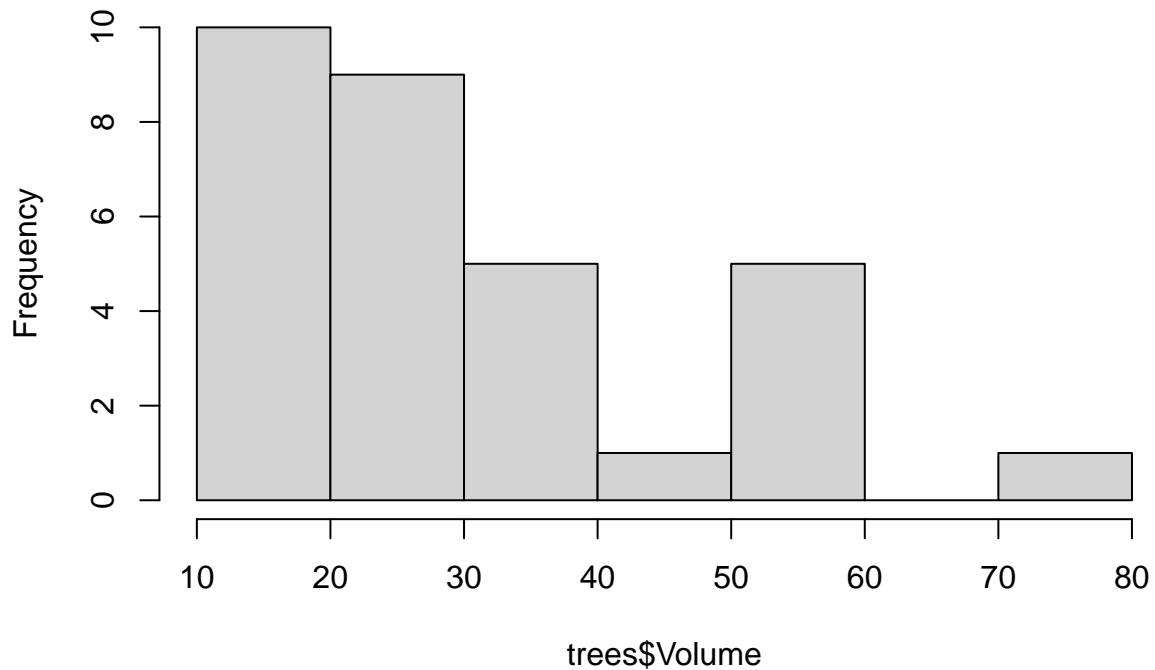
```
##  
## Shapiro-Wilk normality test  
##  
## data: Data  
## W = 0.95068, p-value = 0.3775
```

Data transformation

In our example, there is not enough evidence to reject the H_0 that the data is normally distributed. So let's look at a different dataset. The R dataset `trees` provides `Girth`, `Height`, and `Volume` of different Black Cherry Trees. Let's look at the histogram of the volume data:

```
hist(trees$Volume)
```

Histogram of trees\$Volume



Hm. This dataset doesn't look like it is normally distributed. Performing a `shapiro.test` confirms this.

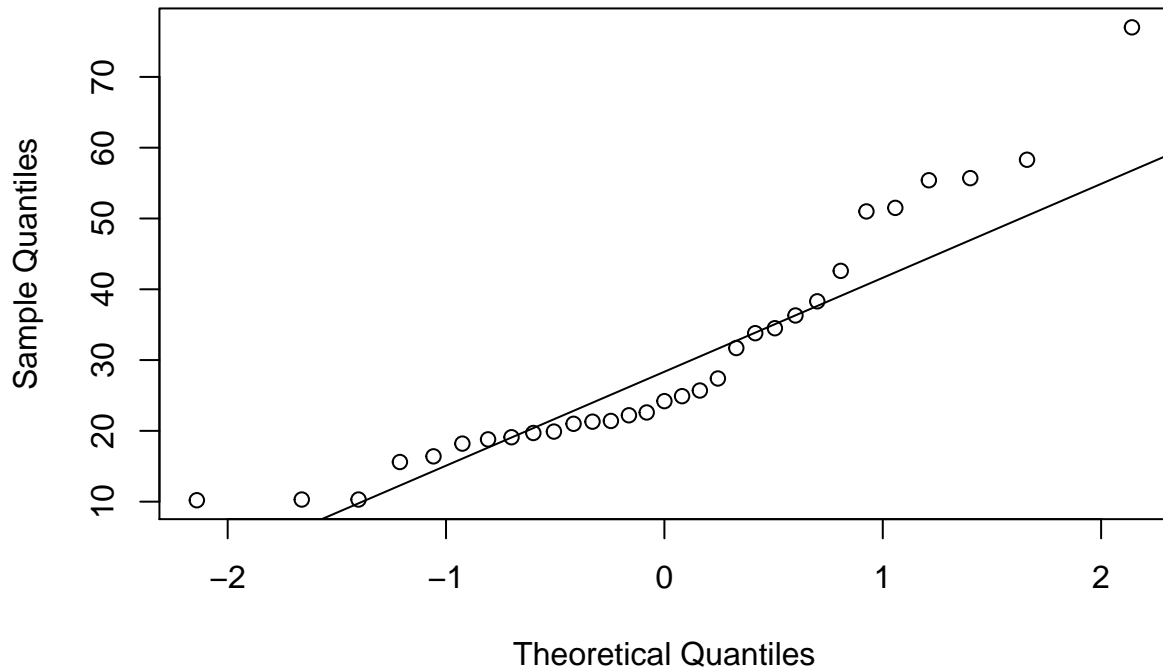
```
shapiro.test(trees$Volume)
```

```
##  
## Shapiro-Wilk normality test  
##  
## data: trees$Volume  
## W = 0.88757, p-value = 0.003579
```

And the QQ plot:

```
qqnorm(trees$Volume)  
qqline(trees$Volume)
```

Normal Q-Q Plot



We can see from the Shapiro test and the QQplot that these data don't seem to fit a normal distribution. In a QQplot, this is usually quite obvious because parts of the data veer pretty far off from the ideal line.

When the data do not look to be sufficiently normal, we can try performing tests on data that have been **transformed** in such a way that they look more normal, so that parametric tests may be performed on them. Some functions that can accomplish this include the `log()`, `sqrt()`, or `arcsin()`. The log and square-root transformations are the most common.

Any transformation that accomplishes the goal is ok, as long as you back-transform at the end in order to report statistics such as 95% confidence intervals (see Plasma Triglycerides example below).

Example: Trees

In biology it is common that multiple factors influence our measurements. The effects may be additive or multiplicative. We know from probability theory that to find the cumulative probability of several independent variables, we can multiply them (product rule). This type of data often gives rise to log-distributed measurements. Taking the log of these stretches out the small values and compresses the larger ones, rendering the data more normal-looking.

Many examples follow a log-normal distribution, such as exponential growth (cell count doubles with each division), systolic blood pressure, and the latency period for infectious diseases.

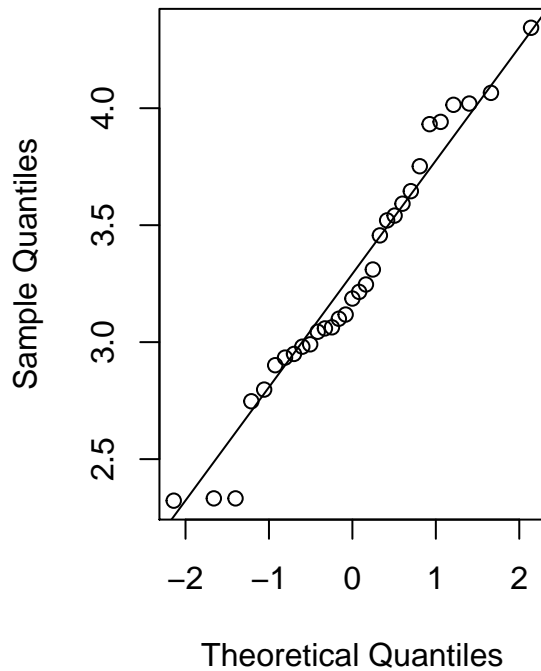
How does the `trees` dataset look when we transform it using the `log` or `sqrt` functions?

```
par(mfrow=c(1,2))  
qqnorm(log(trees$Volume))
```

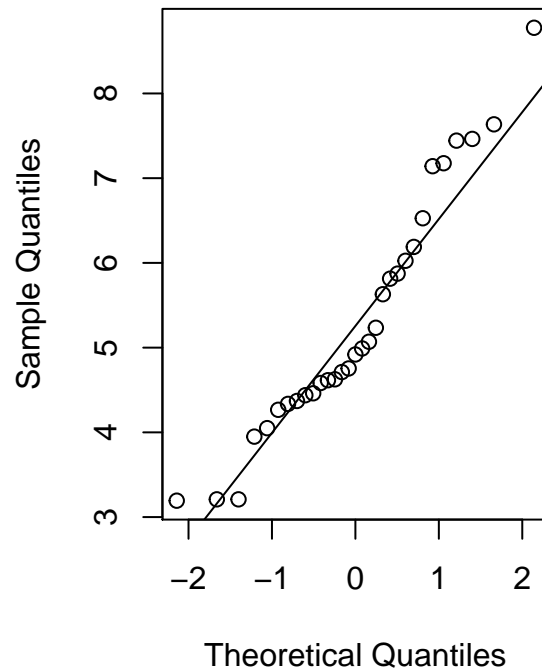
```
qqline(log(trees$Volume))

qqnorm(sqrt(trees$Volume))
qqline(sqrt(trees$Volume))
```

Normal Q-Q Plot



Normal Q-Q Plot



How do these transformations stand up in a test for normality?

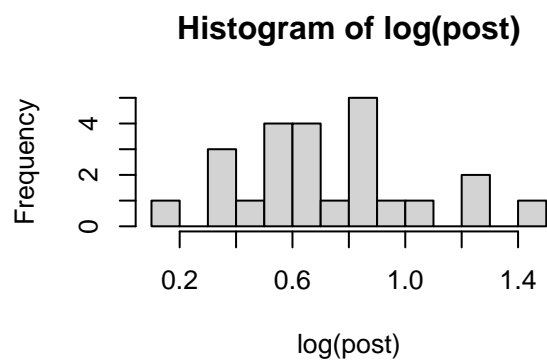
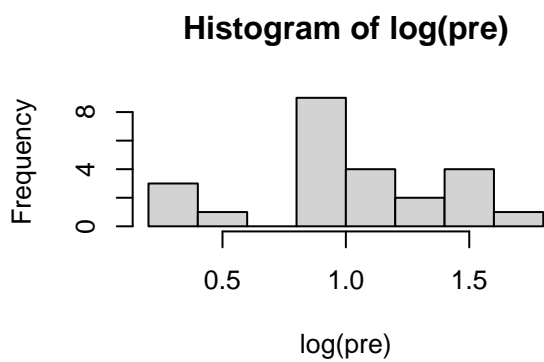
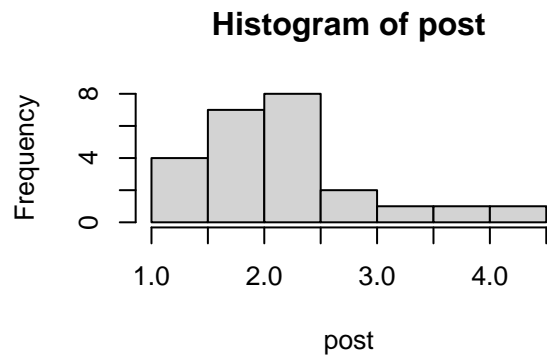
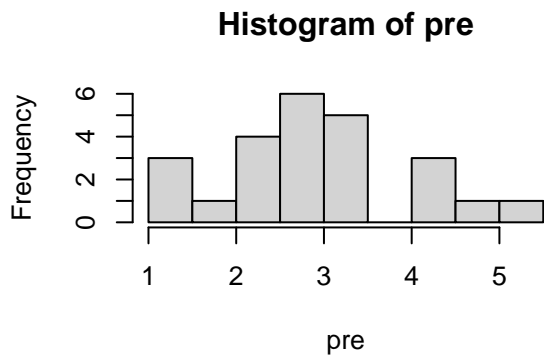
```
shapiro.test(log(trees$Volume))
```

```
##
## Shapiro-Wilk normality test
##
## data:  log(trees$Volume)
## W = 0.96427, p-value = 0.3766
```

```
shapiro.test(sqrt(trees$Volume))
```

```
##
## Shapiro-Wilk normality test
##
## data:  sqrt(trees$Volume)
## W = 0.94105, p-value = 0.08825
```

The log transformation looks promising (and a lot better than the sqrt transform). It's not perfect, but certainly closer to normality than the original data, and suitable for proceeding with statistical methods that require normally distributed data.



```

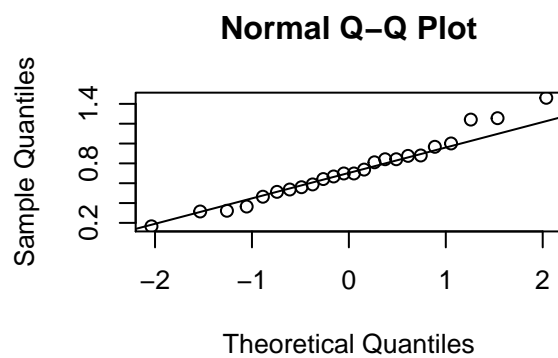
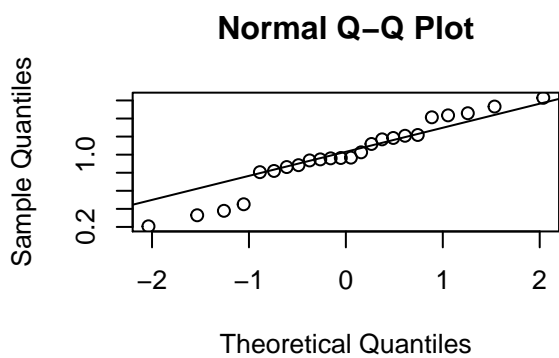
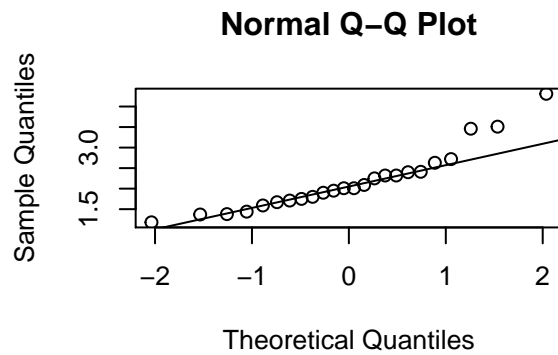
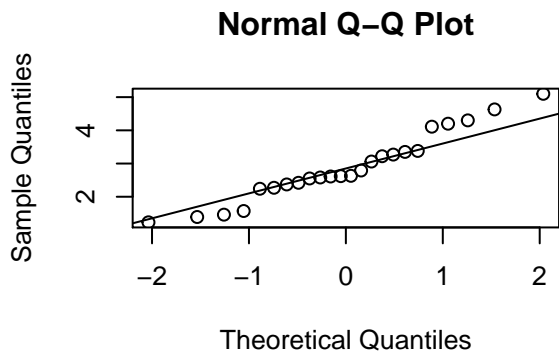
par(mfrow=c(2,2))
qqnorm(pre)
qqline(pre)

qqnorm(post)
qqline(post)

qqnorm(log(pre))
qqline(log(pre))

qqnorm(log(post))
qqline(log(post))

```



Shapiro-Wilk test for normality

Do these samples look approximately normally distributed?

```
# Shapiro-Wilk tests
shapiro.test(pre)
```

```
##
## Shapiro-Wilk normality test
##
## data:  pre
## W = 0.95695, p-value = 0.3802
```

```
shapiro.test(post)
```

```
##
## Shapiro-Wilk normality test
##
## data:  post
## W = 0.89421, p-value = 0.01626
```

```
# Shapiro-Wilk tests
shapiro.test(log(pre))
```

```
##
## Shapiro-Wilk normality test
##
## data: log(pre)
## W = 0.95163, p-value = 0.2938
```

```
shapiro.test(log(post))
```

```
##
## Shapiro-Wilk normality test
##
## data: log(post)
## W = 0.97437, p-value = 0.7742
```

What do you conclude from inspection of the normality of the data?

```
# the "post" data are not normally distributed
# log transformation makes it more normal, but the "pre" group gets a bit worse

# however according to the Shapiro-Wilk test, both are within acceptable limits
# for continuing on with a t-test on the log-transformed data
```

t-tests

Let's compare the results of *t*-tests using the original and the transformed data.

```
## do t-tests using the original and transformed data
t.test(post,pre,paired=T)
##
## Paired t-test
##
## data: post and pre
## t = -2.8642, df = 23, p-value = 0.008771
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
## -1.2321311 -0.1987022
## sample estimates:
## mean of the differences
## -0.7154167
t.test(log(post),log(pre),paired=T)
##
## Paired t-test
##
## data: log(post) and log(pre)
## t = -2.8498, df = 23, p-value = 0.009067
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
## -0.46464693 -0.07379456
## sample estimates:
## mean of the differences
## -0.2692207
```

What are the most striking differences in the results of the *t*-tests? Which *t*-test is more appropriate, and why?

```
# results are actually similar, though of course the t-stats, CIs, and mean differences  
# are not the same  
  
# the transformed test is more correct based on normality tests above
```

95% confidence intervals

Compute the 95% CI for the post-treatment and pre-treatment samples using the log-transformed data.

Manual calculations First, do these “by hand” using the formulas you know for the 95%CI (don’t forget to back-transform into the original units!).

```
# ===== #  
# pre-treatment -- by hand!  
  
# compute standard error  
mean_prime = mean(log(pre))  
sd_prime = sqrt( sum( (log(pre) - mean_prime)^2 / (length(pre)-1)) )  
se_prime = sd_prime/sqrt(length(pre))  
  
# get t-critical  
tcrit = qt(0.975,df=length(pre))  
tcrit  
## [1] 2.063899  
  
# CI in log units  
ci_pre = c(mean_prime - tcrit*se_prime, mean_prime + tcrit*se_prime)  
ci_pre  
## [1] 0.8365772 1.1556327  
  
# CI in original units  
ci_pre_orig = exp(ci_pre)  
ci_pre_orig  
## [1] 2.308452 3.176032  
  
# ===== #  
# 95%CI post-treatment -- by hand!  
mean_prime = mean(log(post))  
sd_prime = sqrt( sum( (log(post) - mean_prime)^2 / (length(post)-1)) )  
se_prime = sd_prime/sqrt(length(post))  
tcrit = qt(0.975,df=length(post))  
tcrit  
## [1] 2.063899  
  
ci_post = c(mean_prime - tcrit*se_prime, mean_prime + tcrit*se_prime)  
ci_post  
## [1] 0.5940810 0.8596873  
ci_post_orig = exp(ci_post)
```

```
ci_post_orig
## [1] 1.811366 2.362422
```

Extracted from *t*-test output Now use the CI values from one-sample *t*-tests to get both of these automatically (again, don't forget to back-transform!) To get these, simply perform 1-sample tests for each group against a group mean.

(Note: these tests should not be used for computing p-values, but the CI for each group is given in the output.)

```
# ===== #
# 95%CI pre-treatment from t-test
t_pre = t.test(log(pre),mu=mean(log(pre)))
t_pre
```

```
##
## One Sample t-test
##
## data: log(pre)
## t = 0, df = 23, p-value = 1
## alternative hypothesis: true mean is not equal to 0.9961049
## 95 percent confidence interval:
## 0.8362093 1.1560005
## sample estimates:
## mean of x
## 0.9961049
```

```
ci95_pre_log = t_pre$conf.int[1:2]
ci95_pre_log
```

```
## [1] 0.8362093 1.1560005
```

```
ci95_pre = exp(ci95_pre_log)
ci95_pre
```

```
## [1] 2.307603 3.177201
```

```
# ===== #
# 95%CI post-treatment from t-test
t_post = t.test(log(post),mu=mean(log(post)))
t_post
```

```
##
## One Sample t-test
##
## data: log(post)
## t = -4.184, df = 23, p-value = 0.000356
## alternative hypothesis: true mean is not equal to 0.9961049
## 95 percent confidence interval:
## 0.5937748 0.8599936
## sample estimates:
## mean of x
## 0.7268842
```

```
ci95_post_log = t_post$conf.int[1:2]
ci95_post = exp(ci95_post_log)
ci95_post_log
```

```
## [1] 0.5937748 0.8599936
```

```
ci95_post
```

```
## [1] 1.810811 2.363145
```

Did you get the same CI for the post-treatment sample when computing by hand and using the `t.test()`?
What can you conclude from the 95% CI's of the two samples?

yes, the CIs are exactly the same whether computed by hand or extracted from a t-test.

the CI's for the two samples barely overlap so the groups are most likely significantly different. Th