Tests for violations of normality

XDASI Fall 2021

10/28/2021

Testing for violations of normality

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?

Drug-Placebo study

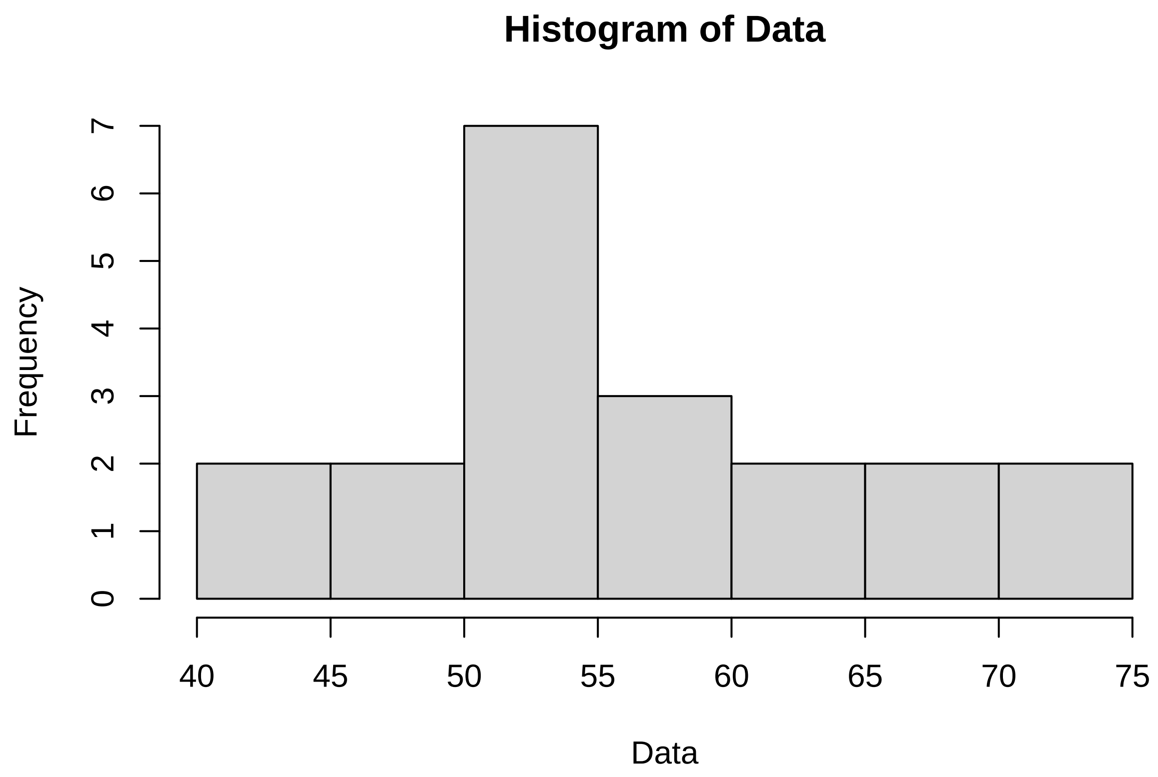
Let's return to our simple case study where a drug has been provided to 10 random patients (test subjects) and, as a control, a placebo pill was given to 10 other random patients. For each condition collected measurements and the question is:

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.

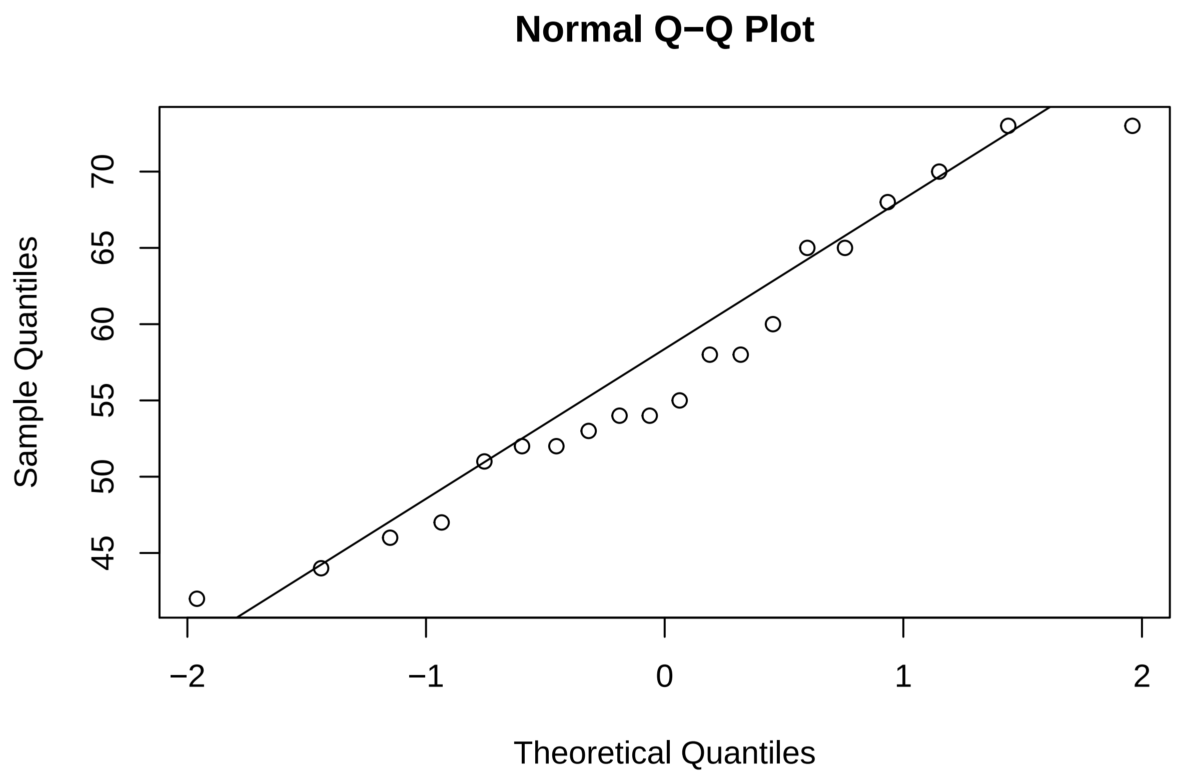
```
Placebo = c(54,51,58,44,55,52,42,47,58,46)
Drug = c(54,73,53,70,73,68,52,65,65,60)

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



Quantile-Quantile plot We can check to see how well the theoretical values of a set of values match the observed. The `qqnorm` and `qqline` function create a plot assuming the data is normally distributed.

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



A more generic function is `qqplot` function where you can provide the theoretical values based on any distribution we want.

The `ppoints` function takes the number of values you want and it returns the probability spanned equally. We have 20 values so let's pick the 20 probabilities. We will then use the `qnorm` function to get the predicted values of the theoretical normal distribution.

```
my_probs = ppoints(20)
my_quant = qnorm(my_probs) #, mean=Data_mean, sd=Data_sd)
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
```

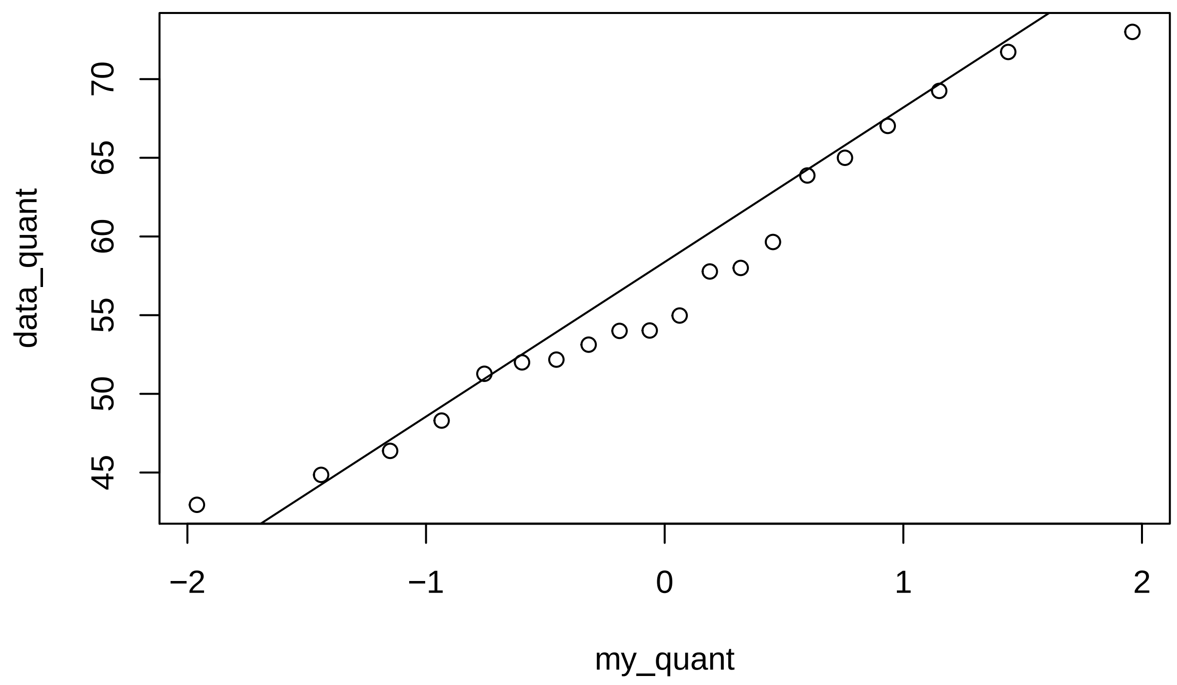
Now to plot the corresponding values from the observed data, we can use the `quantile` function. In addition to the values, the quantile function also takes the probabilites for which it should provide the values, so we can use our `my_probs` again.

```
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
```

Now we are ready to plot the observed data (on the y-axis) vs. the theoretical data (on the x-axis).

```
#plot(my_quant, data_quant)
#abline(a=0,b=1)
qqplot(my_quant, data_quant)
qqline(Data)
```



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 being that the data is normally distributed.

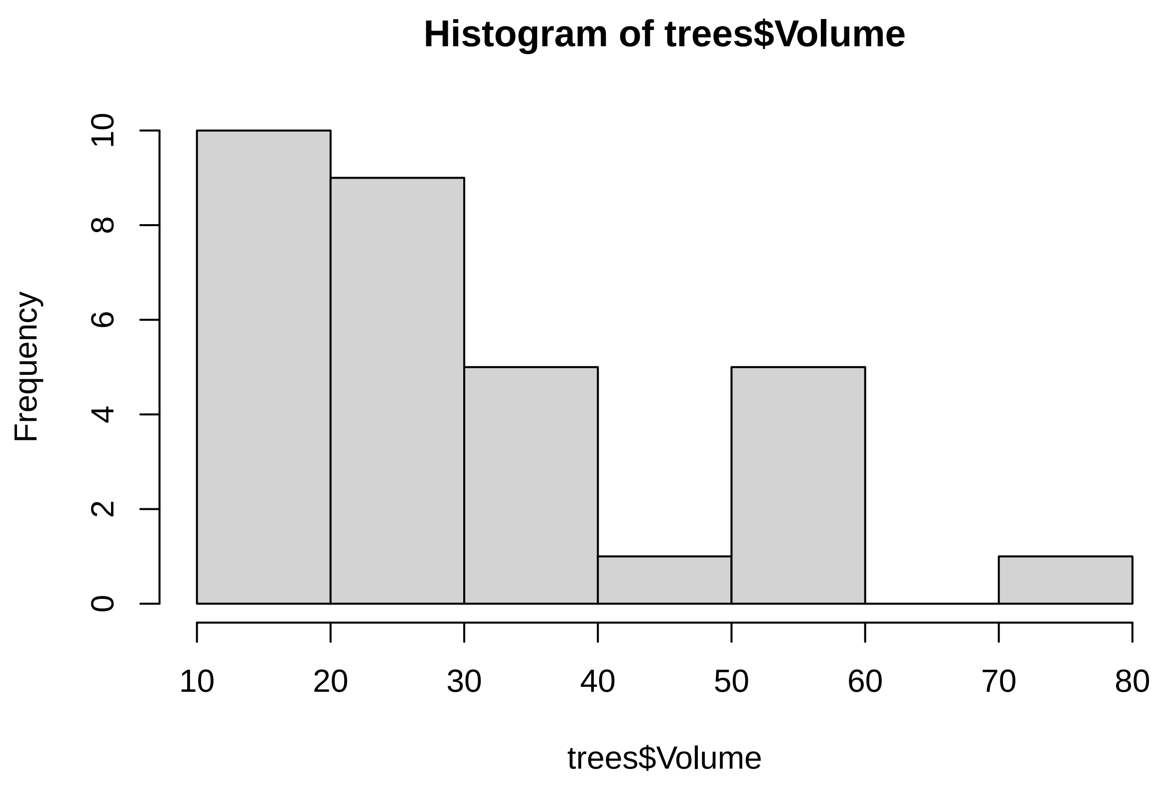
```
shapiro.test(Data)
```

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

Data tranformation

In our example, there is not enough evidence to reject the H_o 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)
```



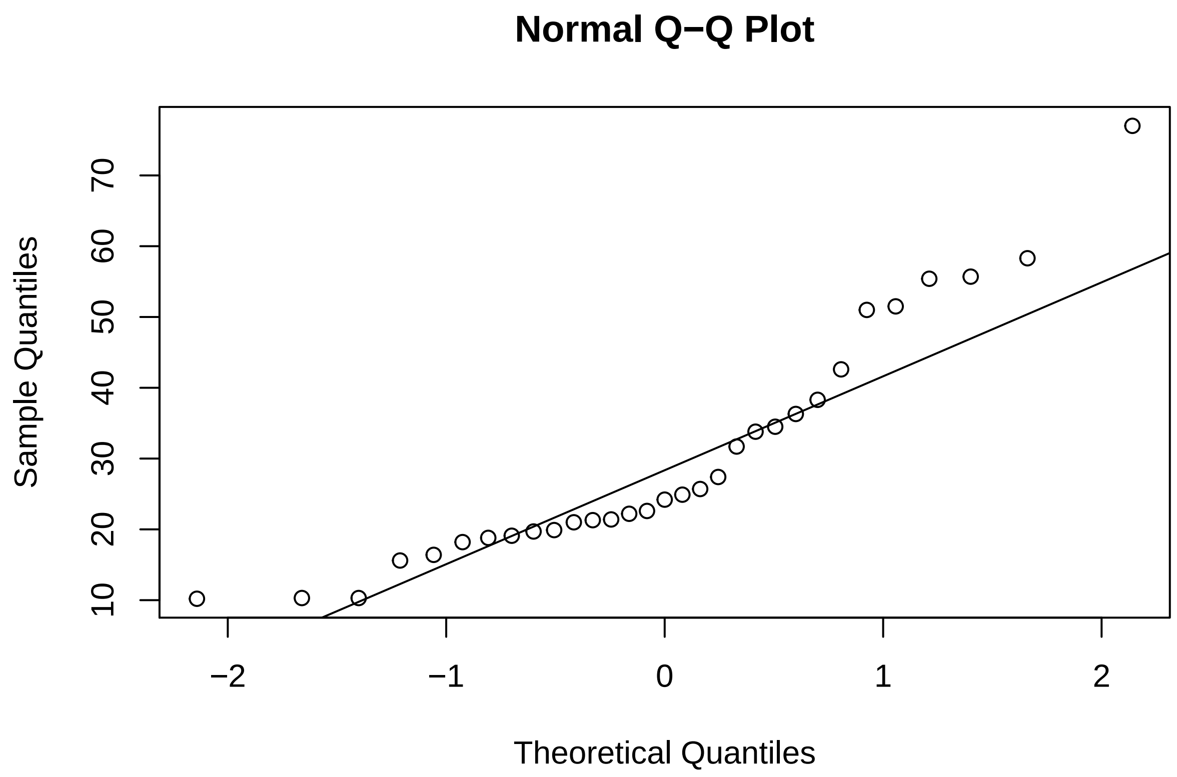
Hm. This dataset doesn't look like it is normally distributed. Performing a `shapiro.test` confirms this.

```
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)
```



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** using functions such as the `log()`, `sqrt()`, or `arcsin()`. This can make data look more normal, so that parametric tests may be performed on them.

Log transform 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 presssure, and the latency period for infectious diseases.

We can use the *Shapiro-Wilk* test to see whether the data follow a normal distribution. Let's investigate the data shown below for plasma triglyceride levels using:

- histograms
- a test for normality
- a `t.test()` on the transformed data
- the 95% CI for the data in their original units
  - we can compute by hand (hard), or extract from `t.test()$conf.int[1:2]` (easy)
  - don't forget to back-transform using `exp()`

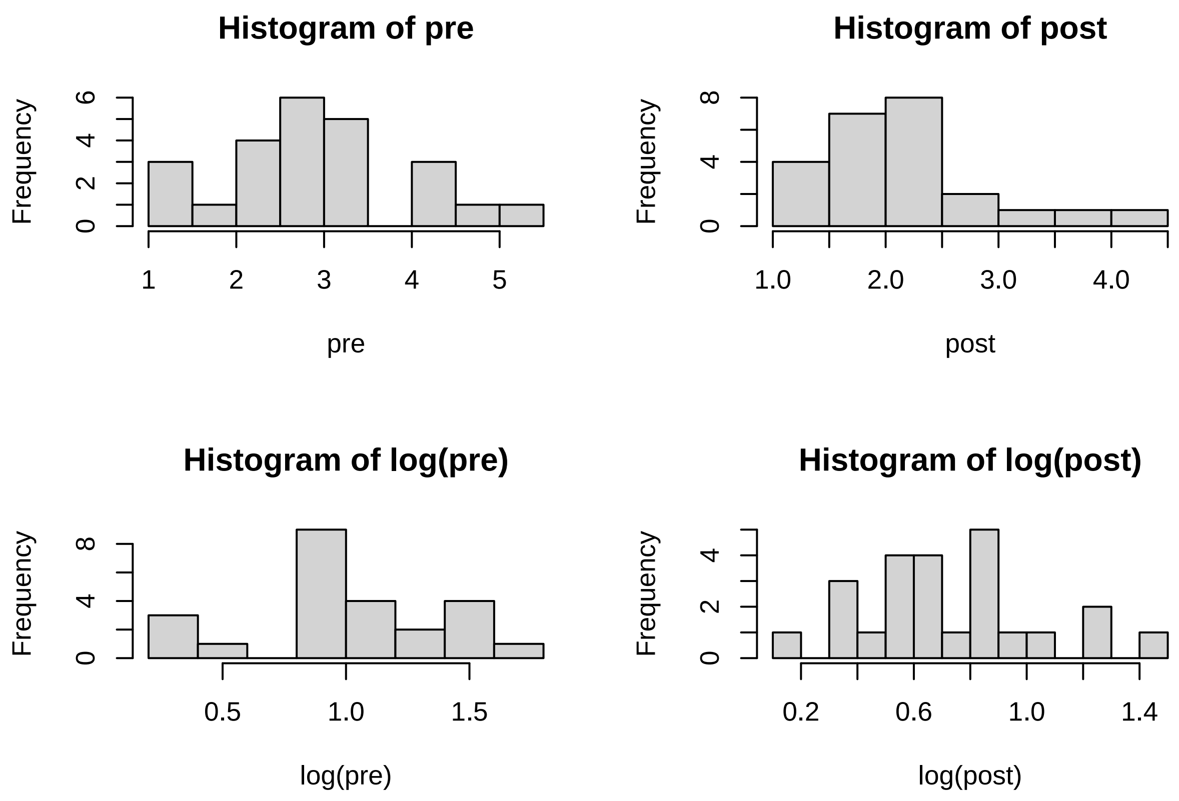
Note: By default, the `log()` function uses the natural log. You can specify other bases with the `base` parameter; convenience functions such as `log10()` and `log2()` are also available.

```
######################################################################
# plasma triglyceride levels in the population (mg/ml)
# borderline high = 2-4 vs. normal < 2
# testing before and after diet and exercise changes (expect a decrease)
pre = c(2.55,3.38,2.37,4.11,3.27,2.58,4.20,3.22,5.10,2.62,3.06,1.23,2.27,2.24,1.39,2.63,2.61,4.30,1.46,
post = c(1.59,3.51,1.44,2.32,1.75,1.67,1.90,1.37,2.72,1.80,2.40,2.01,2.41,1.38,1.18,4.31,2.09,2.32,2.63
######################################################################
```

Histograms

```
# check distributions of original data with histograms (breaks = 10)
par(mfrow=c(2,2))
hist(pre,breaks=10)
hist(post,breaks=10)

# check distributions after log-transformation
hist(log(pre),breaks=10)
hist(log(post),breaks=10)
```



Shapiro-Wilk

```
# 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 your inspection of the normality of the data?

```
# your answer here
# data are not normally distributed but log transform helps
```

t-tests

Perform *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?

```
# your answer here
```

95% confidence intervals by hand

Compute the 95% CI for the post-treatment and pre-treatment samples. Do one of these by hand first (don't forget to back-transform!)

```



## compute the 95% CI for the two samples in their original units

#####################################################################
# pre-treatment -- by hand!
mean_prime = mean(log(pre))
sd_prime = sqrt( sum( (log(pre) - mean_prime)^2 /(length(pre)-1)) )
se_prime = sd_prime/sqrt(length(pre))
tcrit = qt(0.975,df=length(pre))
tcrit
## [1] 2.063899
ci_pre = c(mean_prime - tcrit*se_prime, mean_prime + tcrit*se_prime)
ci_pre
## [1] 0.8365772 1.1556327
ci_pre_orig = exp(ci_pre)
ci_pre_orig
## [1] 2.308452 3.176032
```

```
# 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
```

Pre-treatment

```
#########################################################################
# 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
```

Post-treatment

```
## [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
```

```
# post-treatment from t-test
t_post = t.test(log(post),mu=mean(log(pre)))
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_log
```

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

```
ci95_post = exp(ci95_post_log)
ci95_post
```

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

95% confidence intervals from *t*-tests

Now use the CI values from one-sample *t*-tests to get both of these automatically (don't forget to back-transform!)

```
####################################################################
# 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
```

```
t_post = t.test(log(post),mu=mean(log(post)))
t_post
```

```
##
##  One Sample t-test
##
## data:  log(post)
## t = 0, df = 23, p-value = 1
## alternative hypothesis: true mean is not equal to 0.7268842
## 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_log
```

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

```
ci95_post = exp(ci95_post_log)
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?

```
# your answer here
```