Permutation and non-parametric significance tests

XDASI Fall 2021

10/28/2021

Contents

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

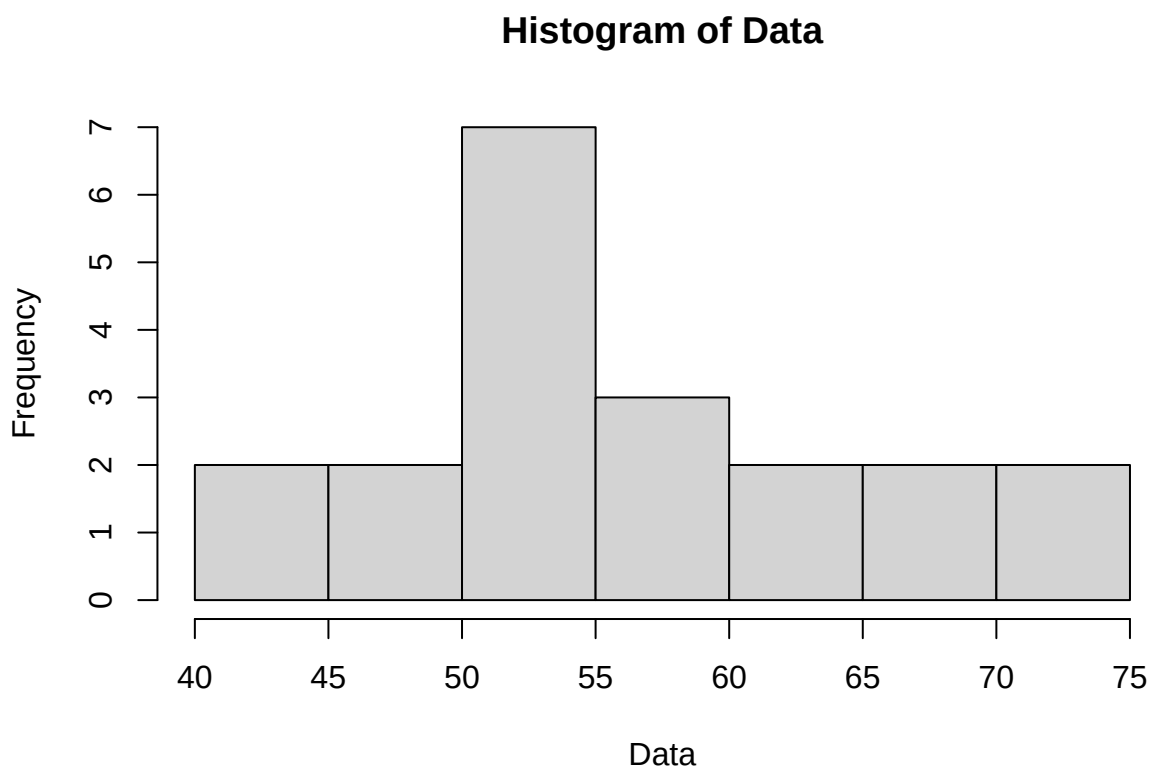
Let's return to our simple case study where a drug has been provided to 10 random patients (test subjects) and, as a control, a placebo pill was given to 10 other random patients. For each condition collected measurements and the question is:

*** 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.

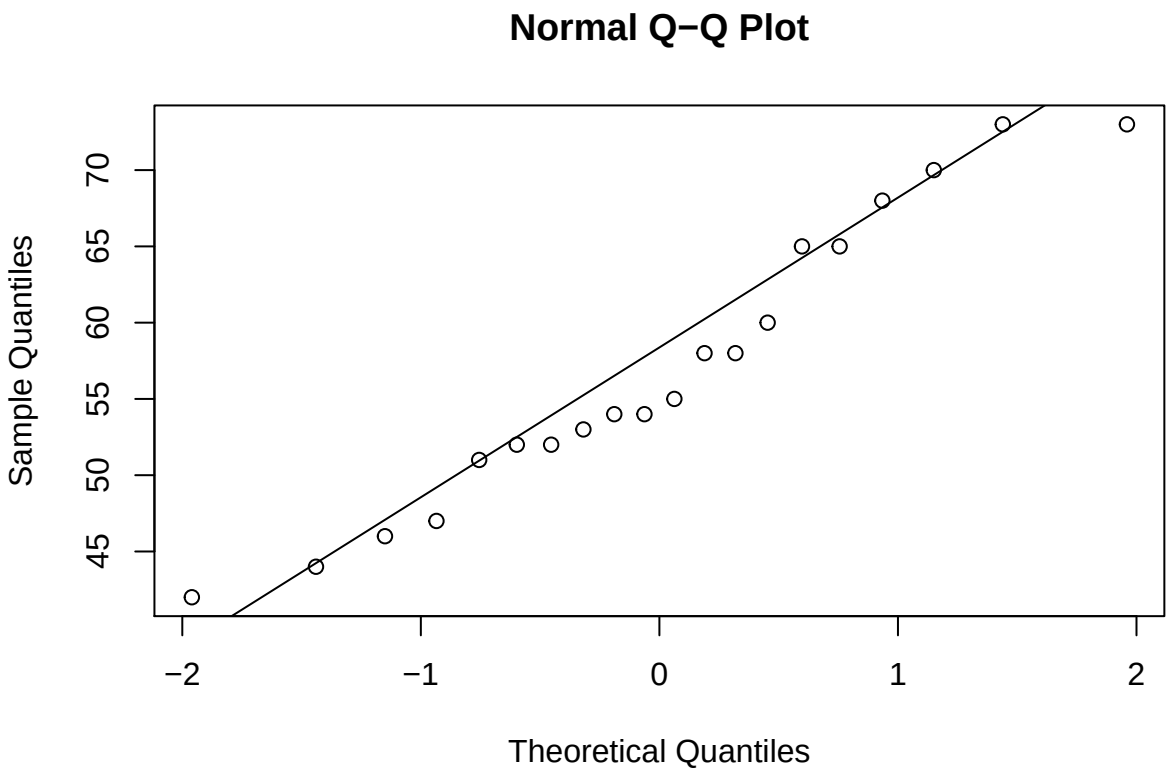
```
Placebo = c(54,51,58,44,55,52,42,47,58,46)
Drug = c(54,73,53,70,73,68,52,65,65,60)

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
```

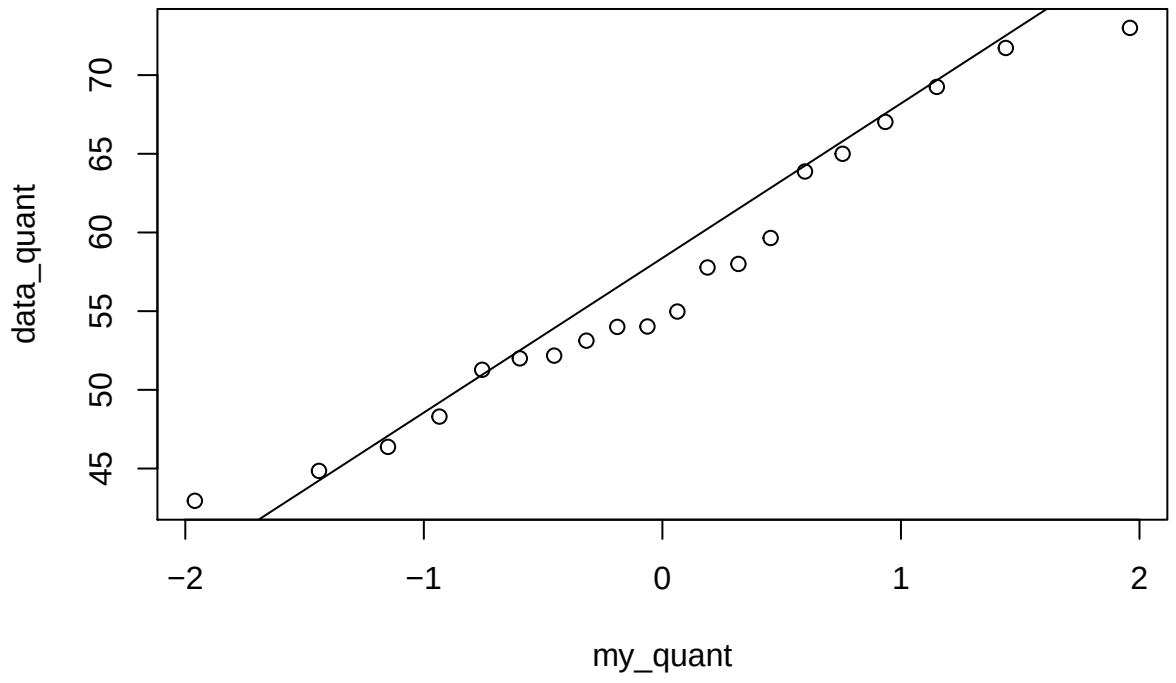
Now to plot the corresponding values from the observed data, we can use the `quantile` function. In addition to the values, the quantile function also takes the probabilites for which it should provide the values, so we can use our `my_probs` again.

```
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)
```

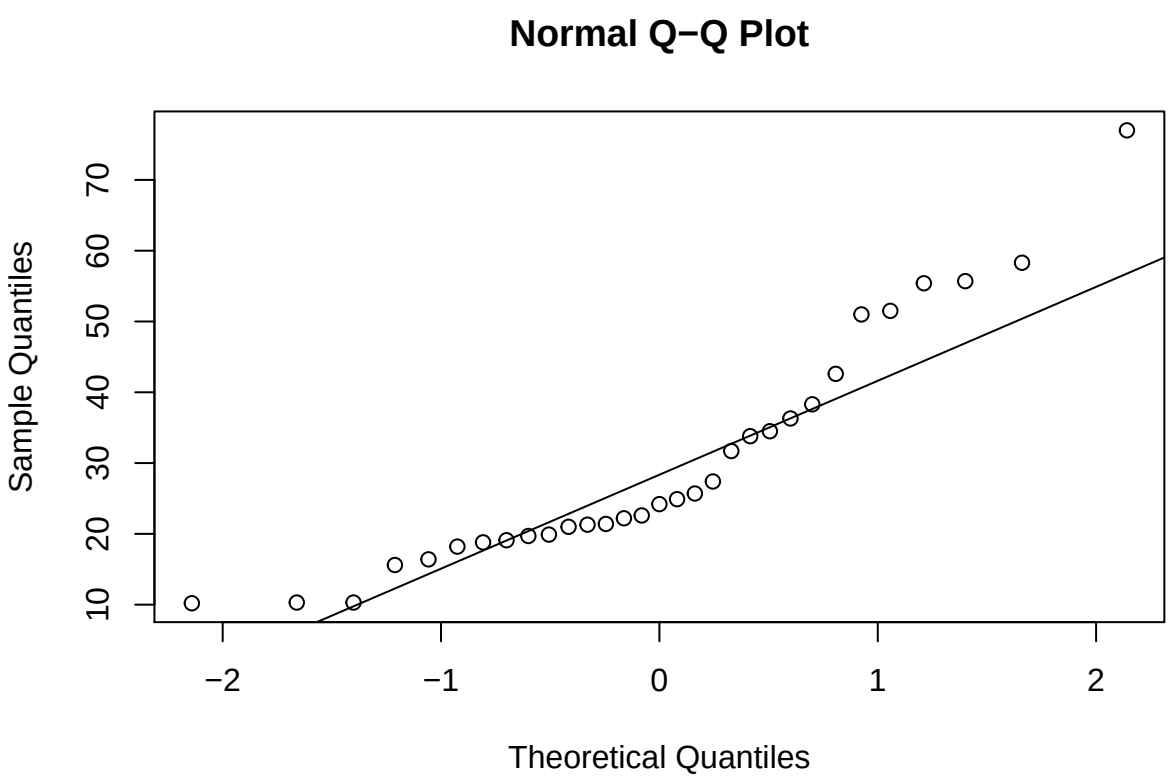
Hm. This dataset doesn't look like it is normally distributed. Performing a `shapiro.test` confirms this.

```
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.

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

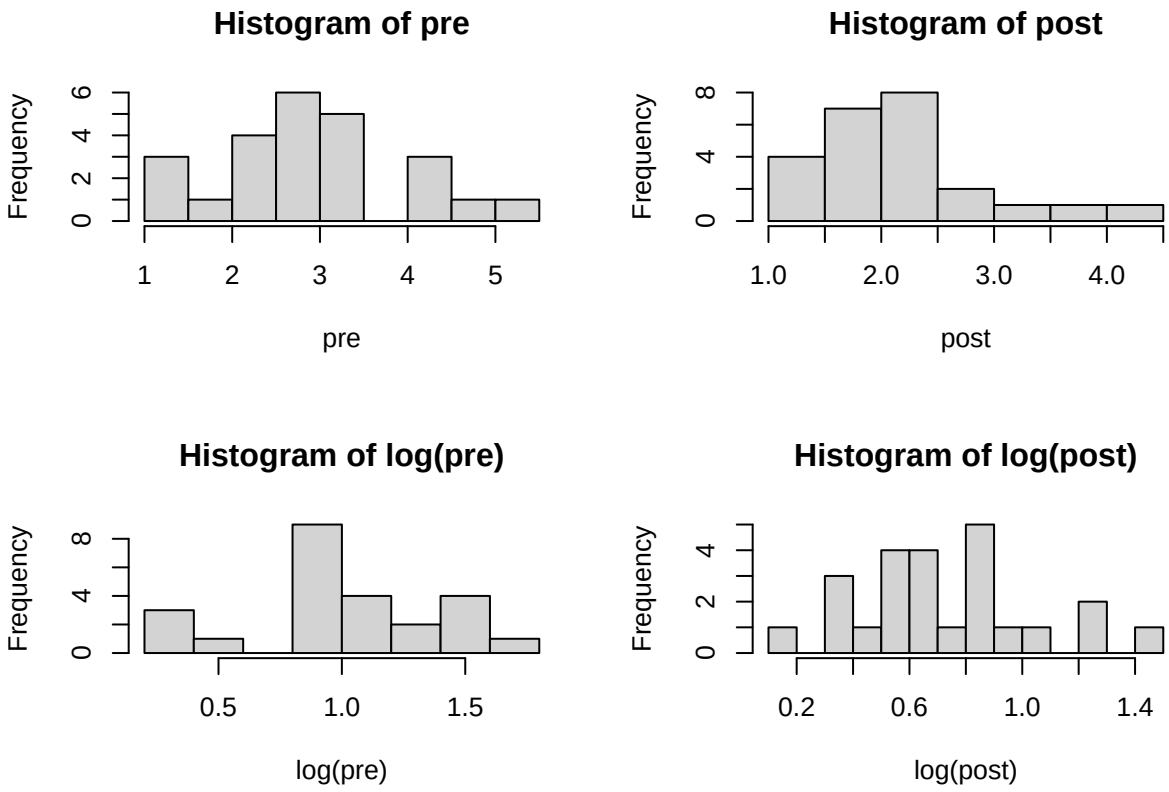
- histograms
- a test for normality
- a `t.test()` on the transformed data
- the 95% CI for the data in their original units
  - we can compute by hand (hard), or extract from `t.test()$conf.int[1:2]` (easy)
  - don't forget to back-transform using `exp()`

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(2.55,3.38,2.37,4.11,3.27,2.58,4.20,3.22,5.10,2.62,3.06,1.23,2.27,2.24,1.39,2.63,2.61,4.30,1.46,3
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
#########################################################################
```

```
# 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 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
```

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

```
# your answer here
# data are not normally distributed
```

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
```

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

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

```r
# 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

# 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
```

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

```r
##########################################################################
# 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
```

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
```

Non-parametric (rank-based) tests for paired data

When the sample data do not follow a normal distribution, we can use tests that look at the data in terms of their ranks (or ranked differences) instead of their magnitudes.

For paired data, our null hypothesis is that the paired differences between the groups should end up with about the same number being positive or negative. This works well when the two differences have a similar skew and variance.

Sign test

Instead of comparing **means** or mean differences, the **sign test** compares the **median** of a sample to a hypothesized value. For paired data, under the null hypothesis we would expect the medians of two samples to be the same. The hypotheses to be tested are:

H_o: The median difference between the groups is zero.

H_A: The median difference between the groups is NOT zero.

The sign test simply takes the signs of the paired differences and counts up the number of negative and positive deviations from zero. This generates binary data, so the p-value is then equivalent to the binomial probability for the observed number of negative deviations when the expected probability is 0.5.

The steps are:

- Calculate the pairwise differences between the samples.
- Count the number of nonzero differences n.
- The test statistic, let's call it k, is the smaller number of the positive and negative differences.
- Calculate the binomial probability of observing k out of n differences given an expected probability of $\pi = 0.5$.

Compute the binomial probability for the "pre" and "post" triglyceride samples. You can use the `ifelse()` function to find the lower number between the negative and positive differences.

```r
# sign test for the triglyceride data

# difference in the two samples
tri_diff = post - pre

neg_count = sum(tri_diff < 0)
neg_count
```

```
## [1] 18
```

```r
pos_count = sum(tri_diff > 0)
test_stat = ifelse(neg_count < pos_count, neg_count, pos_count)

# binomial probability of seeing neg_count or more
prob_obs = pbinom(test_stat, length(tri_diff), prob=0.5)

# total two-tailed probability
2*prob_obs
```

```
## [1] 0.02265584
```

Wilcoxon signed-rank test

The Wilcoxon signed-rank test is similar to the sign test except that it takes the ***sum of all the signed ranks*** (see below), and it can easily be performed directly in R.

One caveat of this test is that it assumes a symmetric distribution about the median difference, which limits its applicability. The test is often used when this assumption is violated, so keep this in mind and try not to make the same mistake. In such cases, it's probably better to use resampling methods (see below).

Here we will run the test on the log-transformed data, which is more symmetrical than the original data.

Procedure The steps are:

- Calculate the N differences between the pairs.
- Rank the *absolute* values of the n non-zero differences. Assign the average if there is a tie.
- Also record the sign of the differences, $sgn(x_{2i} - x_{1i})$ for $i \in \{1..N\}$.
- Compute the test statistic W, defined as:

$$W = \sum_{i=1}^{N} sgn[sgn(x_{2i} - x_{1i}) * R_i]$$

```r
# compute paired log differences
log_tri_diff = log(post) - log(pre)
rank_diff = rank(abs(log_tri_diff))

# Calculate the Wilcoxon W statistic
w_stat = sum(sign(log_tri_diff) * rank_diff)
w_stat
```

```
## [1] -164
```

Normal approximation W has a specific expected distribution under the null. The population mean and variance are:

$$\mu_W = 0;\ \sigma_W^2 = \frac{n(n+1)(2n+1)}{6}$$

When the number of items sampled is more than ~20, we can use a normal approximation to compute a z-score, $z = W/\sigma_W$, from which we can obtain a p-value.

Below we use the normal approximation to compute a p-value for the log-transformed data.

```
# w stat
n = length(log_tri_diff)
sigma_w = n*(n+1)*(2*n+1) / 6
z_w = w_stat / sqrt(sigma_w)
2*pnorm(z_w)
```

```
## [1] 0.01913671
```

Wilcoxon signed-rank test in R

Use the `wilcox.test()` function to test the original and the log-transformed triglyceride data. Don't forget that we are doing paired tests here.

Note: The R implementation of this test uses something called a V-statistic, which is a little different than the W-statistic: it is the **sum of the ranks** assigned to the differences with a **positive** sign. The test gives the same result when samples are entered in either order.

The normal approximation for the V-statistic has parameters:

$$\mu = n(n+1)/4\ ;\ \sigma = n(n+1)(2n+1)/24$$

The p-values from both the W and the V are the same.

```
# computing the V statistic
diff_neg = rank_diff[log_tri_diff < 0]
diff_neg
```

```
##  [1] 10 15 16 19  9 22 23 20  7  6 11  3  5 18  8 12  4 24
```

```
t_neg = sum(diff_neg)
t_neg
```

```
## [1] 232
```

```
diff_pos = rank_diff[log_tri_diff > 0]
diff_pos
```

```
## [1]  1 13  2 14 17 21
```

```
t_pos = sum(diff_pos)
t_pos
```

```
## [1] 68
```

```
v_stat = min(t_neg,t_pos)
v_stat
```

```
## [1] 68
```

```
# v stat
mean_v = n*(n+1)/4
sigma_v = n*(n+1)*(2*n+1) / 24
z_v = (v_stat - mean_v) / sqrt(sigma_v)
2*pnorm(z_v)
```

```
## [1] 0.01913671
```

Run the test in R, using both the raw and the log-transformed data.

```
# test raw data
wilcox.test(post, pre, paired=T)
```

```
##
##  Wilcoxon signed rank exact test
##
## data:  post and pre
## V = 62, p-value = 0.01051
## alternative hypothesis: true location shift is not equal to 0
```

```
# test log data
wilcox.test(log(post), log(pre), paired=T)
```

```
##
##  Wilcoxon signed rank exact test
##
## data:  log(post) and log(pre)
## V = 68, p-value = 0.01787
## alternative hypothesis: true location shift is not equal to 0
```

How do the manual calculations compare to the results of the R tests? Why might these differ?

What do you notice about the differences in the results for the non-transformed vs. the transformed data? Which version is more appropriate?

Unpaired data: Mann-Whitney-Wilcoxon rank-sum test

What if our data are **not normal** and also **not paired**?

We can use a **Mann-Whitney U-test** or the equivalent **Wilcoxon rank-sum** test. Some people therefore like to refer to this test as a **Mann-Whitney-Wilcoxon** test. The test is implemented in R as `wilcox.test()`, with the (default) unpaired option.

This test is simlar to the paired signed-rank test, but instead of measuring paired differences and counting positive and negative differences, we compute the ***sum of ranks for each group in the combined data***. This makes intuitive sense because if the data are drawn from a homogeneous population, we would expect the values of the measurements from two random samples to be relatively well interleaved (as if we had shuffled a deck of cards). So, we would expect that the overall sum of ranks should be about the same.

Procedure

The hypotheses to be tested are:

H_o: The sample distributions are the same.

H_A: The sample distributions are NOT the same.

The steps are:

- Combine the data, and rank from lowest to highest.
- Assign ties the midrank between them (the average of the ranks).
- Compute the sum of ranks R_1 for Sample 1.
- Compute the U-statistic for Samples 1 and 2.
- The sum of the ranks that is higher in magnitude is the test statistic (these will vary depending on the number of individuals in each sample).
- Compute the p-value using the quantile function for the U-distribution.

Note: The sum of ranks for a sequential set of numbers starting with 1 is $N(N+1)/2$. Check this out for yourself on some small sets of numbers (e.g. 1,2,3...). Intuitively, then, if all the measurements from Sample 1 are much smaller than those from Sample 2, then the minimum sum of ranks for Sample 1 would be $R_1 = n_1(n_1+1)/2$.

Computing the U-statistic

The U_statistic measures the difference between the observed and minimum ranks. Note that the sum of the U-values equals the product of the length of the two samples: $U_1 + U_2 = n_1 n_2$.

We can use a simplified method to compute the U-statistics:

$$U_1 = R_1 - \frac{n_1(n_1+1)}{2} = R_1 - minR_1$$

$$U_2 = R_2 - \frac{n_2(n_2+1)}{2} = n_1 n_2 - R_1$$

where n_1 and n_2 are the size of the two groups.

The population mean and variance (again, ignoring the term for ties for now) are:

$$\mu_U = \frac{n_1 n_2}{2} \; ; \; \sigma_U^2 = \frac{n_1 n_2}{12}(n_1 + n_2 + 1)$$

Below we will perform the Mann-Whitney-Wilcoxon test on the triglyceride data, pretending that they are not actually paired. First, we compute the statistics by hand:

```
n = length(pre)  # n is the same for both sets
r_min = n*(n+1)/2 # minimum rank

Data = c(log(pre),log(post))
DataRank = rank(abs(Data))
DataRank
```

```
##  [1] 28.0 40.0 24.0 43.0 38.0 29.0 44.0 37.0 48.0 31.0 36.0  2.0 21.0 19.0  5.0
## [16] 32.5 30.0 45.0  7.0 39.0 35.0 27.0 47.0  8.0  9.0 42.0  6.0 22.5 12.0 10.0
## [31] 14.0  3.0 34.0 13.0 25.0 16.5 26.0  4.0  1.0 46.0 18.0 22.5 32.5 20.0 11.0
## [46] 16.5 15.0 41.0
```

```
DataRankSums = tapply(DataRank,rep(c("pre","post"),each=n),sum)
DataRankSums
```

```
##  post   pre
## 460.5 715.5
```

```
t_pre  = DataRankSums["pre"]
t_post  = DataRankSums["post"]

w_pre = t_pre - r_min
w_pre
```

```
##   pre
## 415.5
```

```
w_post = t_post - r_min
w_post
```

```
##  post
## 160.5
```

```
mu_w = n^2 / 2
mu_w
```

```
## [1] 288
```

```
sigma_w = (n^2/12) * (2*n + 1)
sd_w = sqrt(sigma_w)
```

Normal approximation

Again, when the number of samples is ~20 or more, we can use the normal approximation. We have $n_1 = n_2 = 24$, so let's just calculate a z-score get our p-value.

```
z_w = (w_post - mu_w)/sd_w
z_w
```

```
##      post
## -2.629006
```

```
# these are equivalent
2*pnorm(z_w)
```

```
##        post
## 0.008563493
```

```
2*pnorm(w_post, mean=mu_w, sd=sd_w)
```

```
##        post
## 0.008563493
```

Check the above calculations using the `wilcox.test()` function. Try running the test on both the raw and the log-transformed data.

```
# raw data
wilcox.test(post,pre, paired=FALSE)
```

```
## Warning in wilcox.test.default(post, pre, paired = FALSE): cannot compute exact
## p-value with ties
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  post and pre
## W = 160.5, p-value = 0.008821
## alternative hypothesis: true location shift is not equal to 0
```

```
# log-transformed data
wilcox.test(log(post),log(pre), paired=FALSE)
```

```
## Warning in wilcox.test.default(log(post), log(pre), paired = FALSE): cannot
## compute exact p-value with ties
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  log(post) and log(pre)
## W = 160.5, p-value = 0.008821
## alternative hypothesis: true location shift is not equal to 0
```

How do the manual results compare with the results from the R functions? Why are they not identical?

How do the results from the raw and transformed data compare?

Conclusions

Please reflect on the results from the different tests we have performed here. Summarize in a short paragraph below what you have learned from the analysis of the triglyceride dataset.