Parametric and non-parametric tests for one-sample, paired, and two-sample tests

Contents

Review

Monday, Feb. 24

- Random samples
- Standard normal distribution
  - What is a z-score?
  - What are the quantiles?
  - How do you capture height vs. area?
- Sampling distribution of the sample mean
- Standard error
- Confidence intervals
- What is a z-score?
- t-statistics vs. z-statistics : when is each appropriate?
- P values : one- vs. two-sided
- What does qt(0.975) represent?

Wednesday, Feb. 26

- Parametric vs. non-parametric tests
- Assumptions of parametric tests
- Checking for normality
  - Visualization: histograms, QQ-plots
  - Shapiro-Wilk test
- Transformations - when to use which one?
  - Log, arcsin, sqrt
- Non-parametric tests
  - Sign test (Alt: Wilcoxon signed-rank test)
    - What are they for? Assumptions?
  - Mann-Whitney U test (Equivalent: Wilcoxon rank-sum test)
    - What are they for? Assumptions?
- Power of non-parametric tests

- Options to consider when choosing a test

One-sample, paired-sample, and two-sample t-tests

This worksheet illustrates basic concepts and practical application of significance tests for different experimental scenarios.

To illustrate the various types of t-test, we will use a simple case study where a drug was administered to 10 random patients (or test subjects) and its effect was compared to that of a placebo pill. Three different experimental designs are illustrated:

- Treatment sample vs. control population (one-sample test)
- Two random samples: treatment vs. control (two-sample test)
- One sample measured before and after treatment (paired test)

In all three scenarios, measurements were collected to answer the question:

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

Sample data

To generate sample data for this exercise, I drew two random samples, each from a normal population:

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

Population data

In order to perform a one-sample t-test, we technically need to have some data from a control population for comparison, so that we can test our samples against the true population mean. Here, however, we do not have the population data, so what do we do? We can actually reverse engineer a control sample population, using our best estimates of the population parameters using the control sample we do have!

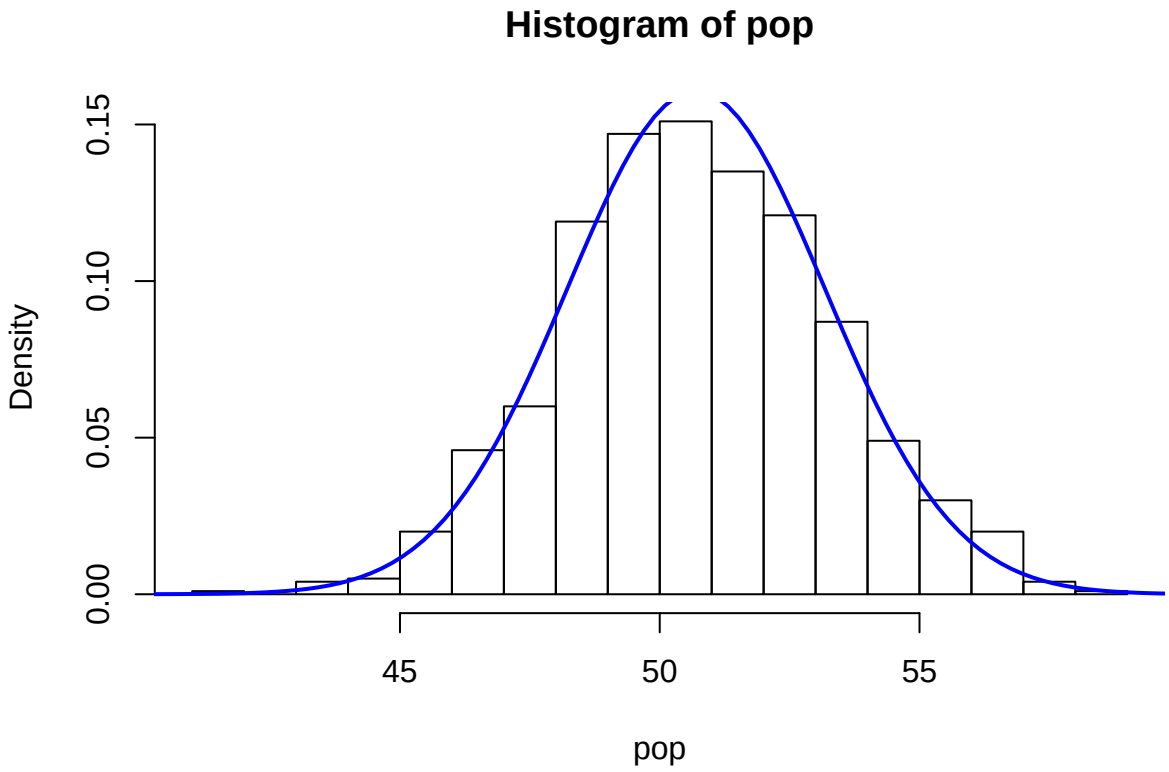
Recall from Chapter 11 that our best estimate for the true population mean μ is the sample mean $\bar{Y}$. Our best estimate for the true population variance (Section 11.5) is:

$$\sigma^2 = \frac{(n-1)s^2}{\chi^2}$$

That looks pretty simple, but what do we use for Chi-squared? If we were computing a 95% CI for the population SD, we would use $\chi_{\alpha=0.025}$ and $\chi_{\alpha=0.975}$. Instead, here we want a representative value from the sampling distribution of χ^2, so we will use the 50th quantile (the midway point in the density distribution).

Using this information, we can sample from a normal distribution with our estimated parameters and pretend that this is our control population data. It will probably be closer overall to the Placebo sample than the original population, since we have to use the sample statistics to estimate the population parameters, but that's ok for now.

```r
## estimate the population parameters

## mean
pop_mean_est = mean(Placebo)

## s.d. -- use the sample variance and the Chi-square distribution
## sigma^2 = df * s^2 / chisq_est(0.5,9)

# a) compute chi-squared estimate
chisq_est = qchisq(0.5, length(Placebo)-1)

# b) compute pop sd estimate
pop_sd_est = sqrt( (length(Placebo)-1)*sd(Placebo) / chisq_est )

## check the estimated parameters
paste("Chi-square = ",round(chisq_est,4),
      " ; Pop. SD est. = ",round(pop_sd_est,4))
## [1] "Chi-square =  8.3428  ; Pop. SD est. =  2.4833"

paste("Population mean estimate = ",round(pop_mean_est,3))
## [1] "Population mean estimate =  50.7"

## simulate a population using the estimated population mean and sd
pop = rnorm(1000,mean=pop_mean_est,sd=pop_sd_est)

## visualize the results
hist(pop, breaks=20, freq=FALSE)
xfit = seq(40,60,length=100)
yfit = dnorm(xfit,mean=pop_mean_est,sd=pop_sd_est)
lines(xfit,yfit,col="blue",lwd = 2)
```




```
## check the parameters from the simulated population
paste("Pop. mean (simulated) = ",round(mean(pop),3),
      " ; Pop. SD (simulated) = ",round(sd(pop),4))
## [1] "Pop. mean (simulated) =  50.715  ; Pop. SD (simulated) =  2.5665"
```

Compare the estimated vs. simulated population parameters. They are close, but not identical.

Confidence Intervals for the Population Mean

The standard error is very helpful because it gives us an idea of how close our data are to the actual mean. We can use the SE to help define **Confidence Intervals (CI)** for the the actual population means from which the samples were drawn.

Here, we assume that we have **random samples** and that the measurements are **normally distributed in the population**. The estimated distribution of the sample means will then follow a ***t*-distribution**, which is similar to a standard normal distribution but with heavier tails. As the sample size increases, the tails of the *t*-distribution become smaller and it resembles closely the *Z*-distribution.

Q1.a: Calculate the 95% CI for the mean of each population that the samples are drawn from.

- Estimate the mean and SE for the Placebo and Drug samples.
- Use the `qt()` *quantile function* to identify the critical *t*-score for the the 95% CI with the appropriate *df*.
- Use the estimated means, SEs, and t_{crit} to get the 95% CI's for the two samples.

```
## t_critical: # 2.26 = t-score for 97.5% area
# Placebo and Drug are the same length so this works for both
t_0.975 = qt(p = .975, df = length(Placebo)-1)
```

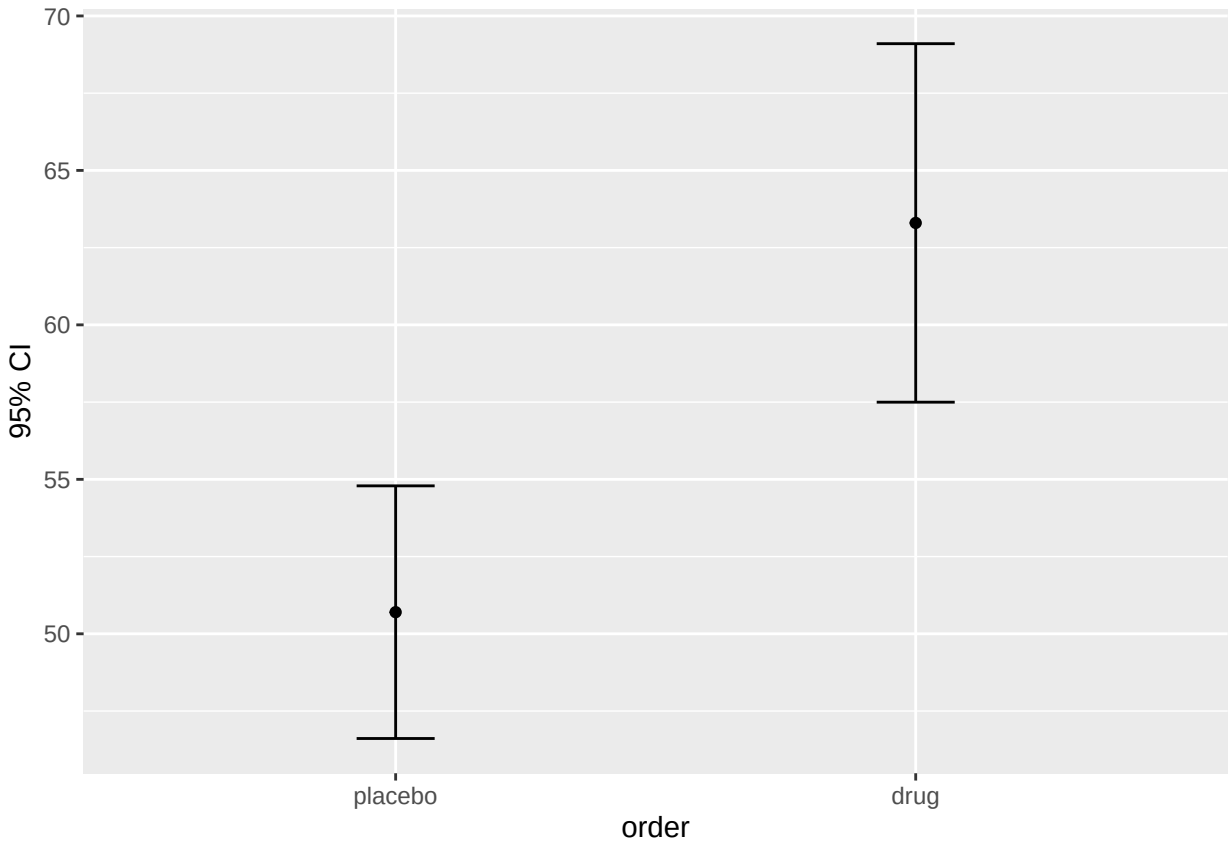
```r
paste0("t_crit = ",round(t_0.975,4))  # 2.26 = z-score for 97.5% area
## [1] "t_crit = 2.2622"

## placebo
Pmean = mean(Placebo)
Pse=sd(Placebo)/sqrt(length(Placebo))

# 95% CI
c(Pmean - Pse * t_0.975, Pmean + Pse * t_0.975)
## [1] 46.6107 54.7893

## drug
Dmean = mean(Drug)
Dse=sd(Drug)/sqrt(length(Drug))

# 95% CI
c(Dmean - Dse * t_0.975, Dmean + Dse * t_0.975)
## [1] 57.49772 69.10228
```

We can check our answers by extracting the CI estimates from the `t.test(...)` function. You can inspect the `t.test` object using `str()`; it's a list containing a bunch of information about the results of the t-test. The precise expression to get the CI is `t.test()$confint[1:2]`.

Q1.b: CI estimates using the t-test function

Do this for both the Drug and the Placebo samples. Also visualize the two confidence intervals using a simple plot.

```r
# confidence intervals
ci1 = t.test(Placebo, mu=mean(pop))$conf.int[1:2]  # placebo
ci2 = t.test(Drug, mu=mean(pop))$conf.int[1:2]  # drug
c(ci1,ci2)
## [1] 46.61070 54.78930 57.49772 69.10228

#################################################################
# visualize confidence intervals with a quick plot (ggplot2)
data.df = data.frame(group = c("placebo","drug"),
                     mean = c(mean(Placebo),mean(Drug)),
                     ci_low = c(ci1[1],ci2[1]),
                     ci_hi  = c(ci1[2],ci2[2]) )
# this is needed to plot groups in the desired order
# (default is alphabetical)
data.df$order = factor(data.df$group, as.character(data.df$group))
qplot(x    = order,
      y    = mean, ylab="95% CI",
      data = data.df) +
  geom_errorbar(aes(ymin  = ci_low,
                    ymax  = ci_hi,
                    width = 0.15))
```



Q1.c: Are the results the same as what you calculated above? Using the rules of thumb given in your textbook, can you conclude by eye that the samples come from different populations? Why or why not?

One-sample t-test

Q2.a: Run a t-test for each sample against:

- the mean of the simulated control population, and
- the population mean estimate from just the Placebo sample.

```
# Placebo sample
t.test(Placebo, mu=mean(pop))
##
##  One Sample t-test
##
## data:  Placebo
## t = -0.0081716, df = 9, p-value = 0.9937
## alternative hypothesis: true mean is not equal to 50.71477
## 95 percent confidence interval:
##  46.6107 54.7893
## sample estimates:
## mean of x
##      50.7
t.test(Placebo, mu=mean(Placebo))
```

```
##
##  One Sample t-test
##
## data:  Placebo
## t = 0, df = 9, p-value = 1
## alternative hypothesis: true mean is not equal to 50.7
## 95 percent confidence interval:
##  46.6107 54.7893
## sample estimates:
## mean of x
##      50.7

# Drug sample
t.test(Drug, mu=mean(pop))
##
##  One Sample t-test
##
## data:  Drug
## t = 4.9066, df = 9, p-value = 0.0008399
## alternative hypothesis: true mean is not equal to 50.71477
## 95 percent confidence interval:
##  57.49772 69.10228
## sample estimates:
## mean of x
##      63.3
t.test(Drug, mu=mean(Placebo))
##
##  One Sample t-test
##
## data:  Drug
## t = 4.9124, df = 9, p-value = 0.0008333
## alternative hypothesis: true mean is not equal to 50.7
## 95 percent confidence interval:
##  57.49772 69.10228
## sample estimates:
## mean of x
##      63.3
```

Q2.b: How similar are the results between the tests using the "true" (simulated) population mean and the ones using the sample mean?

Q2.c: Does it make sense to do a `t.test` of the Placebo sample against its own mean?

Comparison of sample means

We already know that the distribution of the sample mean is normally distributed, and that the sum or difference of two normally distributed variables is also normally distributed.

Therefore, the **difference in sample means** must also be normally distributed, so we can use t-statistics to test whether the means are the same, and we can estimate (with 95% confidence) the true difference in the means of the two populations.

95% CI for the difference in sample means

To calculate the 95% CI for $\bar{\mu_D} - \bar{\mu_P}$ using the t-distribution, we will need several things:

- the observed mean difference
- the SE of the difference
- the degrees of freedom
- the critical t-value

In this example, the standard error is the same whether we use the **pooled variance** formula (for equal variances) or the simpler formula from **Welch's approximate** t-test (for unequal variances), which uses individual sample variances:

$$SE_{\bar{Y}_1 - \bar{Y}_2} = \sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}$$

If we treat these as **independent samples**, the degrees of freedom are: $df = n_1 + n_2 - 2$.

Q3.a: Compute the 95% CI for the difference in means

Compute the CI by hand using the formula for independent samples.

```
# mean difference
mean_diff = mean(Drug) - mean(Placebo)  # 12.6
mean_diff
## [1] 12.6

# SE of the difference
se_diff = sqrt ( var(Placebo)/length(Placebo) +
                 var(Drug) /length(Drug) )
se_diff
## [1] 3.13794

# critical t-value
t_crit = abs(qt(p=0.025, df=18))  # same as qt(p=0.975, df=18)
t_crit
## [1] 2.100922

# 95% CI
c(mean_diff - t_crit * se_diff, mean_diff +  t_crit * se_diff)
## [1]  6.007433 19.192567
```

Notice that the confidence interval does not span 0, which means it is extremely unlikely that the true difference between the Drug and Placebo populations is 0.

Let's check this using a t-test.

Two-sample t-test

The **null hypothesis** is that the difference between the two populations is 0. We know this because if we were to plot a distribution of random variables with the same mean, we would expect to get a mean difference of 0.

$$H_o : \mu_{Drug} = \mu_{Placebo}$$
$$H_A : \mu_{Drug} \neq \mu_{Placebo}$$

Since we expect that we have random samples drawn from a normal distribution, it is valid to use t-statistics to test our null hypothesis.

Recall that the t-score for the difference in the means is computed in the same way as the z-score for a normal distribution: it is simply the **difference**, **standardized** by the standard error. We already computed the mean and SE above, so this will be very easy!

Q3.b: Compute the t-statistic and p-value for the difference in means

First, calculate by hand the test statistic (the t-score for the observed difference) and use it to find the probability that our null hypothesis is true (the p-value for our test statistic).

Next, perform a two-sample test using the `t.test()`. Try both the standard version and Welch's approximate t-test.

```
# compute the test statistic for the mean difference
t_score = mean_diff / se_diff
t_score
## [1] 4.015373

# degrees of freedom for 2-sample test
df_2sample = length(Placebo)+length(Drug)-2

# compute a two-sided p-value
2*pt(q = t_score, df = df_2sample, lower.tail = F)
## [1] 0.0008115625

# perform two versions of a 2-sample test using R
t.test(Drug, Placebo, var.equal = T)  # standard test
##
##  Two Sample t-test
##
## data:  Drug and Placebo
## t = 4.0154, df = 18, p-value = 0.0008116
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##   6.007433 19.192567
## sample estimates:
## mean of x mean of y
##      63.3      50.7
t.test(Drug, Placebo, var.equal = F)  # Welch's approximation
##
##  Welch Two Sample t-test
##
## data:  Drug and Placebo
## t = 4.0154, df = 16.171, p-value = 0.0009803
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##   5.95359 19.24641
## sample estimates:
```

```
## mean of x mean of y
##      63.3      50.7
```

Q3.c: How do the results of the manual calculation and the two t-tests compare?

Q3.d: What is similar and what is different about these results in comparison with the one-sample tests above?

Paired t-test

What if the data are taken from the **same individuals**? For example, we could test the same patients before and after treatment. In this kind of experimental design, we say that the data are **paired**. If there is no difference between the two measurements for each individual – for example, a new drug for blood pressure has no measurable benefit – then we would expect that our before and after values would be about the same on average.

The **paired t-test** is performed in the same way as the one-sample t-test, except we use the **mean difference between paired measurements** from the two samples, $\bar{X}_D$, to compute a test statistic. Our **null hypothesis** is usually that the mean difference, D_o, is zero (we could set it to something else if our null hypothesis is that the difference between them is something else ...).

For paired data, we assume that the two sets of measurements are arranged in the same order as the corresponding individuals (because we have good record-keeping practices!) The test statistic is:

$$t^* = \frac{\bar{X}_D - D_0}{\frac{s_D}{\sqrt{n}}} = \frac{\bar{X}_D - D_0}{SE_D} = \frac{\sqrt{n}}{s_D}(\bar{X}_D - D_0)$$

where $\bar{X_D}$ is the mean of the pairwise differences, s_D is the standard deviation of the pairwise differences, and D_0 is what we are testing (which in this case is 0).

To compute this t-statistic, we can simply subtract one vector from the other to obtain **pair-wise differences** for each individual, take the mean, and divide by the standard error. We then find the p value in the usual way.

Study the above equation to convince yourself that the **form** of the t-statistic above is the same as that for a **one-sample** t-test – we simply substitute $\bar{X}_D$ for $\bar{X}$, D_o for μ_o, and s_D for s.

Q4.a: Perform a paired t-test

First get a p-value using a manual calculation:

```
## manual p-value using the t-statistic

# paired mean difference
pair_diff = Drug-Placebo

# SE of the difference
se_diff = sd(pair_diff)/sqrt(length(pair_diff))

# observed t-statistic
t_stat = mean(pair_diff)/se_diff
t_stat
```

```
## [1] 4.108696

# probability
2*pt(t_stat, df=9, lower.tail = F)
## [1] 0.002642154
```

Here is the manual calculation:

```
# manual p-value using the t-statistic
pair_diff = Drug-Placebo

se_diff = sd(pair_diff)/sqrt(length(pair_diff))
t_stat = mean(pair_diff)/se_diff
t_stat
## [1] 4.108696
2*pt(t_stat, df=9, lower.tail = F)
## [1] 0.002642154
```

Now use the `t.test()` function to perform a one-sample test that compares the paired difference vector against the expected mean difference.

```
# R command for one-sample t-test of the difference in means
t.test(pair_diff, mu=0)
##
##  One Sample t-test
##
## data:  pair_diff
## t = 4.1087, df = 9, p-value = 0.002642
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##   5.662718 19.537282
## sample estimates:
## mean of x
##      12.6
```

Q4.b: Perform a paired t-test

Above you just performed a **one-sample** t-test using the **paired differences** between the two samples.

It is important to understand that this gives the same result as a **two-sample** t-test with the **paired** option! So you don't actually need to take the paired differences to do this test – you can just plug in the two samples and choose the paired option

Use the syntax for a **two-sample paired test** to verify that the two methods are equivalent. Compare the results using `var.equal = T` and `var.equal = F`.

```
# t.test(treatment, control, ...)
t.test(Drug, Placebo, paired = T, var.equal = T)
##
##  Paired t-test
##
## data:  Drug and Placebo
## t = 4.1087, df = 9, p-value = 0.002642
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##   5.662718 19.537282
```

```
## sample estimates:
## mean of the differences
##                    12.6
t.test(Drug, Placebo, paired = T, var.equal = F)
##
##  Paired t-test
##
## data:  Drug and Placebo
## t = 4.1087, df = 9, p-value = 0.002642
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##   5.662718 19.537282
## sample estimates:
## mean of the differences
##                    12.6
```

From these results, you can see that specifying equal or unequal variances makes no difference for a **paired test**. This makes sense because a paired test only evaluates one set of values!

Note: when performing any t-test with two samples, you must specify the **test set first**, and the **control set second** – otherwise you get a t-score that is on the opposite side of the distribution. This will not change your p-value, however your **confidence interval** will have the wrong sign!!!

What if the data are not normally distributed?

Transformation

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:

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

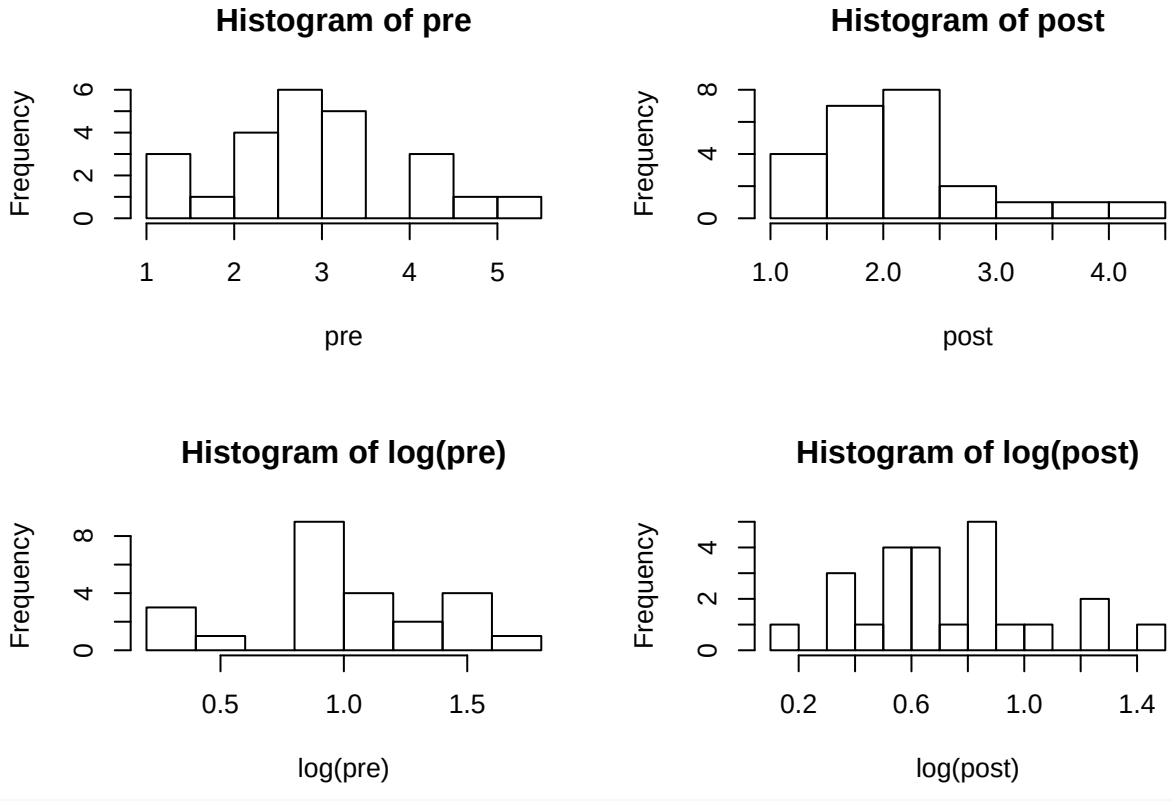
Q5.a: Exploratory data analysis

We will examine the data using graphical methods and statistics:

- histograms
- QQ plots
- 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()`

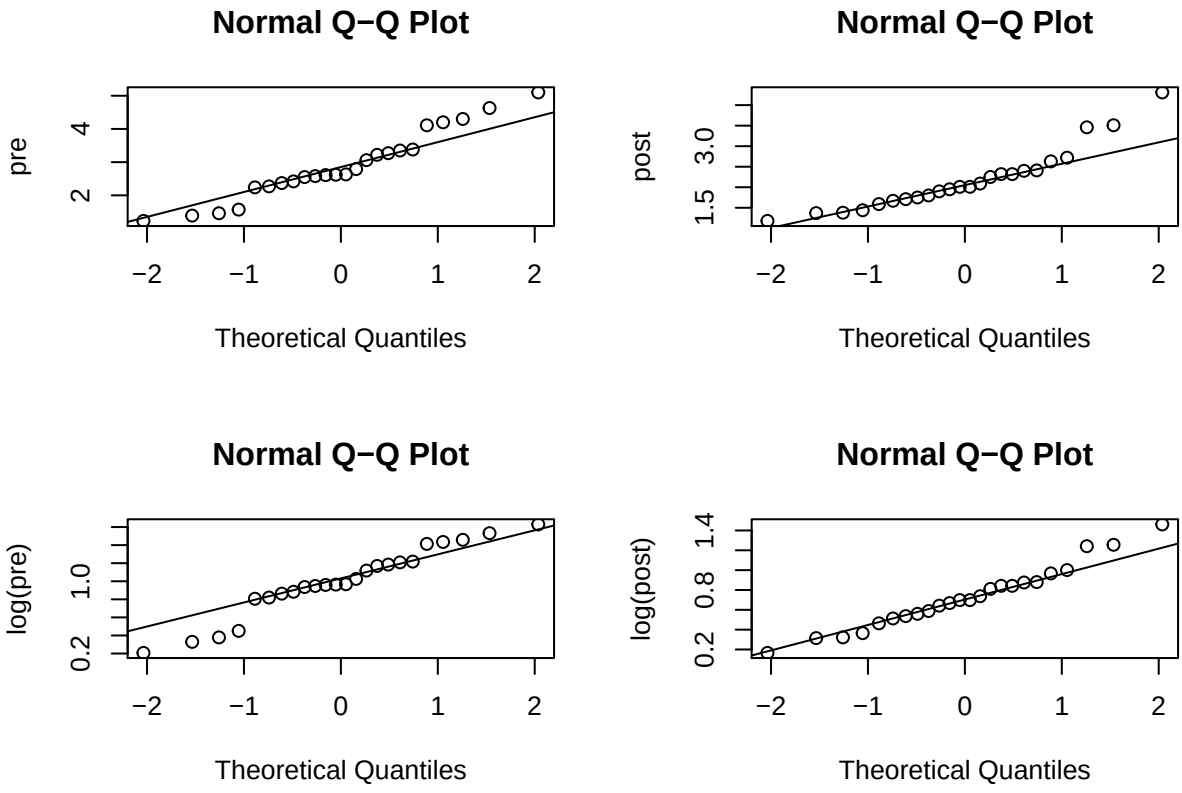
Note: By default, the `log()` function uses the natural log. You can specify other bases with the `base` parameter; convenience functions such as `log10()` and `log2()` are also available.

```
################################################################
## histograms (breaks = 10)
par(mfrow=c(2,2))

# raw data
hist(pre,breaks=10)
hist(post,breaks=10)

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




```
################################################################
## QQ-plots: use `qqnorm(data, ylab="data"; qqline(data)`
par(mfrow=c(2,2))
```

```r
# QQ-plot - raw data for pre and post
qqnorm(pre,ylab="pre"); qqline(pre)
qqnorm(post,ylab="post"); qqline(post)

# QQ-plot - log data for pre and post
qqnorm(log(pre),ylab="log(pre)"); qqline(log(pre))
qqnorm(log(post),ylab="log(post)"); qqline(log(post))
```




```r
########################################################################
## Shapiro-Wilk tests - `shapiro.test(your_data_here)`
shapiro.test(pre)    # pre-treatment
##
##  Shapiro-Wilk normality test
##
## data:  pre
## W = 0.95695, p-value = 0.3802
shapiro.test(post)   # post-treatment
##
##  Shapiro-Wilk normality test
##
## data:  post
## W = 0.89421, p-value = 0.01626

shapiro.test(log(pre))    # pre-treatment
##
##  Shapiro-Wilk normality test
##
## data:  log(pre)
```

```
## W = 0.95163, p-value = 0.2938
shapiro.test(log(post))  # post-treatment
##
##  Shapiro-Wilk normality test
##
## data:  log(post)
## W = 0.97437, p-value = 0.7742
```

Q5.b: What do you conclude from your inspection of the normality of the data? Does the log transform make the data more normal?

Q5.c: Perform paired t-tests on the raw and 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
```

Q5.d: What are the most striking differences in the results of the t-tests? Which t-test is more appropriate, and why?

Q5.e: 95% confidence intervals

First, we compute the 95% CI for the post-treatment by hand for illustration:

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

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

Now extract the 95% CI's for the log-transformed data (pre and post) using the t-test command, comparing against the mean of the data itself. Don't forget to back-transform!

```
############################################################
## post-treatment t-test
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

# get the CI data
ci95_post_log = t_post$conf.int[1:2]

# back-transform
ci95_post = exp(ci95_post_log)
ci95_post
## [1] 1.810811 2.363145

############################################################
# pre-treatment 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
```

```r
# get the CI data
ci95_pre_log = t_pre$conf.int[1:2]
ci95_pre_log
## [1] 0.8362093 1.1560005

# back-transform
ci95_pre = exp(ci95_pre_log)
ci95_pre
## [1] 2.307603 3.177201
```

Q5.f: Comparison of results

How do the CI's from manual calculations and the `t.test()` compare? What can you conclude from the 95% CI's of the two samples?

Non-parametric (rank-based) tests

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.

Tests for paired data: Sign and Wilcoxon signed-rank tests

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. 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 $\pi = 0.5$.

Below we compute the binomial probability for the "pre" and "post" triglyceride samples. We 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
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, 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.

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
```

W and V statistics

Below you will 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
```

Q6.a: Wilcoxon signed-rank test in R

Run the paired tests in R, using both the raw and the log-transformed data. The syntax is `wilcox.test(test, ctl, paired=T)`.

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

Q6.b: How do the manual calculations compare to the results of the R tests? Why might these differ?

Q6.c: What is different about the results for the raw and transformed data? Which version is more appropriate?

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

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:

- Assign ranks to the combined data, 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 U-value that is **higher** in magnitude is the test statistic, and it needs to be tested against an upper-tail critical value.
- 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

# ranks
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

# rank sums for each sample
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"]
```

Note: Using the **lower** U value as the test statistic and comparing to a **lower-tail** critical value is equivalent to using the **higher** U value and comparing against an **upper-tail** critical value.

R use the lower U value and calls it W. We do this below for easier comparison to the results of the `wilcox.test()` function.

```
# test statistic
w_pre = t_pre - r_min
w_pre
##   pre
## 415.5
w_post = t_post - r_min
w_post
##  post
## 160.5

# use the minimum U-stat (called W here) as the test statistic
w_stat = min(w_pre,w_post)
w_stat
## [1] 160.5

# parameters for distribution
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. We multiply by 2 to get the final p-value.

```
# z-score
z_w = (w_stat - mu_w)/sd_w
z_w
## [1] -2.629006

# these are equivalent
2*pnorm(z_w,lower.tail=T)
## [1] 0.008563493
2*pnorm(w_stat, mean=mu_w, sd=sd_w,lower.tail=T)
## [1] 0.008563493
```

Q7.a: Perform Wilcoxon rank-sum tests

Check the above calculations using the `wilcox.test()` function. Run the test on both the raw and the log-transformed data. The syntax is the same as before, except now we are using **independent** samples.

```
# 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
```

Q7.b: How do the manual results compare with the results from the R functions? Why are they not identical?

Q7.c: How do the results from the raw and transformed data compare?

Conclusions

Q8: Finally, 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.