Permutation tests:

Permutation tests are a large group of statistical procedures.

Most typical tests can also be done as permutation tests. For example:

Two sample tests (e.g., t-test, MWU test)

Three sample, two way designs, blocked designs, (e.g., ANOVA)

Paired tests (e.g, paired t-test, Wilcoxon signed rank test)

Contingency tables (e.g., χ^2 goodness of fit test)

etc.

All of these can be done using permutations tests.

Permutation tests make no assumptions about the distribution, and are considered "non-parametric" or "distribution-free" tests.

But what is a permutation test?

A simple example: two sample test.

Suppose we want to find out if the means of two samples are significantly different.

We go out and collect some data, and then follow this procedure:

1) Calculate the difference in means.

2) We pool our data (combine both samples).

Why? Because H_0: $\mu_1 = \mu_2$ implies that both samples come from the same population, so we can combine our data into one big sample.

3) We now take two samples (at random) from this combined samples.

i) we makes sure they're of the appropriate size (i.e., use $n_1 = n_1$ and $n_2 = n_2$, or in other words, our "new" samples must be the same size as the original samples)

ii) we calculate the difference in means for these two "new" samples.

iii) if H_0 is true, then most of the time this difference should be small.

But occasionally, we would get larger differences.

4) We repeat (3) as many times as we can (see below), and count how often the differences are larger than our original observed difference.

There are some variations on this - we can also use the test statistic, for instance (see the two way ANOVA example below).

5) If our original observed difference falls within the largest 5% of the differences, (if α = 0.05) we reject H_0.

Or, equivalently, we could calculate the number of differences that are greater than our observed differences, and divide this by the total number of differences. This would give us an actual p-value:

$$p\text{-}value = \frac{\text{\# of differences greater than the observed difference}}{\text{total number of differences calculated}}$$

Some comments:

Technically, permutation tests are non-parametric tests.

They do not make any assumptions about the distribution of the data.

However, depending on the purpose of the test, some assumptions (e.g., equal variances), may still apply.

Note that in one sense, permutation tests are really simple as the basic idea/theory is actually pretty straight forward.

If you want to calculate power, or compare permutation tests to other tests, this can get complicated.

Permutation tests can also be used to calculate such things as:

equivalence of distributions

distribution of various estimates

others

Note that the total number of permutations can get very large very quickly:

Suppose we want to do a two sample test and have n_1 = 16, n_2 = 17:

We combine this into a single sample of size 16 + 17 = 33

How many different ways can we get two samples of n_1 = 16 and n_2 = 17 from this combined data set?

We now use the binomial coefficient:

$$\binom{33}{16} = 1{,}166{,}803{,}110$$

Obviously, this already large number would explode if you had, for example, four samples (e.g, a one way ANOVA) of similar sample size (say, 16).

Because of this, it's often impossible to calculate all possible permutations. Instead, we randomly "sample" from the combined data and use this.

Other permutation tests:

One nice thing about permutation tests is that (as mentioned above) they are easy to adapt to other situations. For example, ANOVA:

One way ANOVA is quite simple (just combine all groups instead of just two).

Two way or blocked designs are more complicated:

You frequently still wind up calculating various Sums of Squares, etc.

Permutation tests with ANOVA have an advantage over traditional non-parametric techniques which are often not very powerful (with the exception of Kruskal-Wallis).

You can adapt permutation tests to many different ANOVA designs.

Here is an example of a two way ANOVA done with a permutation test (there is some disagreement as to the best way to approach this).

1) Calculate the two way ANOVA as always.

2) Get the F-values for each effect and the interaction.

(remember that with a fixed effects two way ANOVA we have three values of F, one for each of the main effects and one for the interaction)

3) Combine the data into one big data set.

4) Sample from this new data set (re-arrange the data randomly)

5) Calculate the "new" two way ANOVA

6) Repeat steps 4 & 5 as many times as desired (e.g., 5000).

7) For each "new" ANOVA count the number of times each F value is bigger than our original F-value.

8) Divide by the total number of trials. This gives us the *p-value* for each of the effects and the interaction.

If you're really interested, check out (examples are posted that you can run):

http://www.uvm.edu/~dhowell/StatPages/More_Stuff/Permutation %20Anova/PermTestsAnova.html

An example of a two sample permutation test using R:

Note that to really do permutation tests in R (without a "canned" package) requires basic programming skills.

One reason statisticians really like R is that it isn't just a statistics package. It's a full programming language (like C, C++, Fortran, basic, etc.).

Let's stick with a two sample test. Let's use an example from the midterm about boomslangs.

We have the lengths (in cm) of two subspecies of boomslang, A and B.

Objective: calculate whether the observed difference in means is significant using a permutation test.

H_0: $\mu_1 = \mu_2$, H_1: $\mu_1 = \mu_2$, $\alpha = 0.05$

Subspecies A: 150 151 125 126 124 116 92 119 145 140 99 76 119 88 137 74 151 66 118 151

Subspecies B: 133 152 184 129 144 111 136 140 143 115 119 161 88 159 125 99 114 112 123 118 151 150 123

How would we do a permutation test on these data?

Step 1: Read the data into R:

```
booma <- scan(nlines = 1)
150 151 125 126 124 116  92 119 145 140  99  76 119  88 137  74 151  66 118 151

boomb <- scan(nlines = 2)
133 152 184 129 144 111 136 140 143 115 119 161  88 159 125  99 114 112 123
118 151 150 123
```

Step 2: Calculate the observed difference in the means:

```
diff <- mean(booma) - mean(boomb)
diff
```

Step 3: Combine the datasets into one big dataset:

```
boom <- c(booma, boomb)
```

Step 4: Decide on how many samples (permutations) we want:

```
n_perm <- 1000

# this should probably be set higher than 1000
```

Step 5: Now we need to do some real programming. We set up a "loop" that will take two samples of size n_1 and n_2, then calculate the means, and finally calculate the difference. Our loop will then keep track of how often this difference is bigger than our observed difference (step 2).

I've annotated the loop below (note that it is standard practice to indent stuff inside the loop):

```
permdiff = NULLl

# this sets up an “empty variable”.  All this does is let R know that “permdiff”
# exists, but it tells R to keep it empty (otherwise R would give us errors
# below when we refer to “permdiff”

for (i in 1 : n_perm) {

        # This sets up the loop.  “for” says” do everything after the “{“,
        # letting “i” equal 1 to n_perm. For example, with n_perm = 1000,
        # this will run through the loop 1000 times.

        newboom <- sample(boom, length(boom))

        # Basically what this does is re-arrange the combined sample (step 3).
        # We tell “sample” to sample from “boom”, and to take a sample
        # that's the same length as “boom”.  All this does is re-arrange
        # our sample.

        aboom <- newboom[1 : length(booma)]
        bboom <- newboom[(length(booma) + 1) : length(boom)]

        # Now we take this re-arranged sample (newboom) and divide it into two
        # samples, each of which is the same length as the original samples.
        #
        # Note that the first sample is obviously of length “booma”, and the
        # second sample is then everything else (from “booma + 1” to “boom”)
        #

        permdiff[i] <- mean(aboom) - mean(bboom)

        # Calculates the difference in the means of our “new” samples.
        # We will have n_perm of these differences (e.g, if n_perm = 1000,
        # then we will have 1000 differences)

}

# We close the loop and are done.  Everything above will run “n_perm” times.
```

Step 6: Calculate how many of our differences (stored in “permdiff” above) are bigger (or equal) to our actual observed differences.

```
ndiffs <- sum(abs(permdiff) >= abs(diff))

# This is a logical expression.  It is true if |permdiff| ≥ |diff|, and
# then has the value 1.  So for every difference that we get from our loop
# that is bigger than our observed difference, we count “1”.
# We can look at it by doing:

ndiffs
```

Step 7: Get the actual p-value:

```
p <- ndiffs/n_perm

# We just figure out the proportion of our sampled differences
# that are greater than our observed difference.

p
```

And, of course, as usual we can compare this *p-value* with *α* and make our decision as usual.

Note that the p-value will change every time you run the test.

This is normal, since you are randomly sampling from the combined data set, and every time you run the test, you will have slightly different samples.

However, this DOES mean you probably should use a reasonable size for the loop (1000, as illustrated above is probably too small).

Try something like 5000 or even 10,000.

Careful: if your calculations are complicated and you run through the loop too many times may need to wait a very long time to get an answer (even on today's fast computers).

There ***is*** a reason we need supercomputers!

Try setting "n_perm" to about 100,000 in the loop above.

Depending on the speed of your computer you may need to wait anywhere from 15 seconds to over a minute before you get an answer.

Our calculations are actually quite simple.

The above code was changed and adapted from a web page on permutation tests at:

http://spark.rstudio.com/ahmed/permutation/ (There is a bug or two in the original code on this page, so be careful if you copy and paste).

You will have one of these on your homework assignment. You'll need to adapt the above to the homework (it really shouldn't be too difficult). The above code is posted on the web page (you can just copy it into R-studio and run it.

We won't go into permutation tests in any more detail than this as it would take quite a bit more time, and this should be sufficient to understand the basics.