Notes for Stat 430 – Design of Experiments – Lab #4 Oct. 1, 2012.

Agenda:
Power calculations – power.t.test
Power – Exercise 2.34
Normality testing – QQNorm
Normality testing – Shapiro.test

Useful files: http://www.sfu.ca/~jackd/Stat430/Wk04_Code.txt
All lab material found at : http://www.sfu.ca/~jackd/Stat430/

Power Calculations

We're used to doing is testing null hypotheses through p-values and alpha.
P-value*: The chance that a random sample would produce as much evidence against the null as the observed sample did, if the null were true.*

We compare this to alpha,
Alpha: *the acceptable chance of falsely rejecting the null hypothesis for a test.*

By default, we assume the null and see how likely such a sample is under the null.

We could also assume an alternative and see how likely it would be to reject the null under that alternative.

- If the alternative is close to the null, there would be a small chance of (correctly) rejecting the null.

- If the alternative is far from the null, it's highly likely that the null would be rejected, assuming this alternative.

This chance of correctly rejecting the null is called the POWER, and it is calculated as 1 - beta, where beta is the chance of a Type II error
Type II Error: *Failing to reject the null when the null is actually false*

Finding or using power isn't as simple as finding the p-value. For something as simple as `power.t.test()` we need four of the following five...

`n` (The sample size in EACH group. Must be equal or use the smallest one)
`delta` (The difference between the two means)
`power`
`sd` (Standard deviation)
`sig.level` (Alpha. The chance of rejecting the null when the null is true)'

First, look at the help file
`?power.t.test`

The definition includes `n=NULL, delta=NULL, sd=1, sig.level=0.05, power=NULL`
That means that if we don't put anything in for sd, it will default to 1.
If we don't put anything in for sig.level, it will default to 0.05. The rest have no default.

Next, let's use the examples:
This uses as many defaults as possible: sd = 1, alpha = 0.05, two-sample, two-tailed

```
power.t.test(n = 20, delta = 1)
```

Which gives a power of 0.9886; you would have a 98.96% chance of rejecting the null if the true difference was 1 sd, and each group has 20 units. You can find the t-score of two samples of size 20 with means one standard deviation apart, and verify this with `pt()`

Changing the values reveals more about t-test power. If we reduce the sample size to 2 each

```
power.t.test(n = 2, delta = 1)
```

…we get almost no power, (0.09 ish) when sample size is very small.

Also power can never get smaller than alpha/2 for two-tailed tests.

```
power.t.test(n = 20, delta = 0.5)
power.t.test(n = 20, delta = 0.2)
power.t.test(n = 20, delta = 0.01)
power.t.test(n = 20, delta = 0)
```

(One-tailed tests left in the code)

We can also use it the other way. If we have a specific power level and significance level that we desire, and an estimate of the standard deviation and effect size we expect, we can find the required sample size.

For example…

```
power.t.test(power = .90, sig.level= 0.05, delta = 1, sd = 1)
```

…yeilds n = 22.02111. A little more 22 units per group will give you exactly 0.9 power.

It takes fewer sample units to detect a larger effect relative to the standard deviation.

```
power.t.test(power = .90, sig.level = 0.05, delta = 2, sd = 1)
power.t.test(power = .90, sig.level = 0.05, delta = 1, sd = 1/2)
```

If the sig.level decreases, that makes it harder for the null to be rejected in general, not just when it's false.

```
power.t.test(power = .90, sig.level = 0.05, delta = 1, sd = 1)
power.t.test(power = .90, sig.level = 0.01, delta = 1, sd = 1)
```

And the same for when a higher power is needed

```
power.t.test(power = .95, sig.level = 0.01, delta = 1, sd = 1)
```

Power - Exercise 2.34

Using the paired data in exercise 2.34 (8th ed hardcover) (Unpaired analysis in code)

```
karl = c(1.186,1.151,1.322,1.339,1.200,1.402,1.365,1.537,1.559)
leh  = c(1.061,0.992,1.063,1.062,1.065,1.178,1.037,1.086,1.052)
```

If the real difference in the means were the same as the difference between the means of the these samples. What would be the power of the two-sided test alpha = 0.05 ?

Step 1: Get delta, n, and sd

```
D = mean(karl) - mean(leh)
n.paired = length(karl)
std.dev = sd( karl - leh)
```

Step 2: Get the power. Making sure the data is treated like paired data.

```
power.t.test( n=n.paired, delta=D, sd=std.dev, sig.level=0.05,
type="paired")
```

(Additional notes on normality tests in code)

Normality Test - qqnorm

A Q-Q plot, or quantile-quantile plot, arranges the values of a dataset in increasing order. On the x-axis is the number of standard deviations above or below the mean that a value of this quantile would be if the distribution were normal.

Breaks from a straight line mean breaks from normality, but small breaks at the lower and upper ends are less significant.

```
from.norm = rnorm(100, mean=10, sd = 3)

par(mfrow = c(1,2))
hist(from.norm)
qqnorm(from.norm)
qqline(from.norm)
```

In this example, the 100 values are at the 0.5%, 1.5%, 2.5%, ... , 99.5% quantiles.
So the 3rd value, at the 2.5% quantile, should be 1.96 standard deviations below the mean, assuming normality. Its x-position is that exactly.
The y-position is its actual value, which is about 6 (From normal, 2 std below 10)

If the distribution is positively skewed, the qq line bends downward

```
from.exp = rexp(100)

hist(from.exp)
qqnorm(from.exp)
qqline(from.exp)
```

If the distribution is negatively skewed, the qq line bends upward

```
from.beta = rbeta(100, shape1=8, shape2=2)

hist(from.beta)
qqnorm(from.beta)
qqline(from.beta)
```

If there is more or less kurtosis (peakness) than normal distribution, an s-shape appears

The qqplot is ALWAYS increasing because it's using the sorted values.
qqnorm is a normality check, not a symmetry check, it can find breaks from normality in symmetric but non-normal distributions.

Shapiro-Wilks test

There's not much to this test. It only takes in one thing: a list of numbers

```
shapiro.test(from.norm)
```

And gives out two things:
Shapiro and Wilk's W statistic: Perfectly normal is 1, and the farther from normal something gets, the lower this is, down to a value of 0.

The p-value of the test.

```
shapiro.test(from.exp)
shapiro.test(from.beta)
```

It's easier to detect small differences in large sample sizes

```
shapiro.test( rt(n=10,df=20)) # 10 values from a t-distribution with 20 df

shapiro.test( rt(n=30,df=20)) # 30 values
shapiro.test( rt(n=100,df=20)) # 100 values
shapiro.test( rt(n=1000,df=20)) # 1000 values
shapiro.test( rt(n=5000,df=20)) # 5000 values (the max for this test)
```

The W-statistic remains quite high, showing that t with 20 df is quite close to a normal, but eventually the p-value gets very small anyway.