11.10: Testing Non-normal Data with Wilcoxon Tests

Okay, suppose your data turn out to be pretty substantially non-normal, but you still want to run something like a t-test? This situation occurs a lot in real life: for the AFL winning margins data, for instance, the Shapiro-Wilk test made it very clear that the normality assumption is violated. This is the situation where you want to use Wilcoxon tests.

Like the t-test, the Wilcoxon test comes in two forms, one-sample and two-sample, and they're used in more or less the exact same situations as the corresponding t-tests. Unlike the t-test, the Wilcoxon test doesn't assume normality, which is nice. In fact, they don't make any assumptions about what kind of distribution is involved: in statistical jargon, this makes them ***nonparametric tests***. While avoiding the normality assumption is nice, there's a drawback: the Wilcoxon test is usually less powerful than the t-test (i.e., higher Type II error rate). I won't discuss the Wilcoxon tests in as much detail as the t-tests, but I'll give you a brief overview.

11.10.1 sample Wilcoxon test

I'll start by describing the ***two sample Wilcoxon test*** (also known as the Mann-Whitney test), since it's actually simpler than the one sample version. Suppose we're looking at the scores of 10 people on some test. Since my imagination has now failed me completely, let's pretend it's a "test of awesomeness", and there are two groups of people, "A" and "B". I'm curious to know which group is more awesome. The data are included in the file `awesome.Rdata`, and like many of the data sets I've been using, it contains only a single data frame, in this case called `awesome`. Here's the data:

```
load("./rbook-master/data/awesome.Rdata")
print( awesome )
```

```
##    scores group
## 1     6.4     A
## 2    10.7     A
## 3    11.9     A
## 4     7.3     A
## 5    10.0     A
## 6    14.5     B
## 7    10.4     B
## 8    12.9     B
## 9    11.7     B
## 10   13.0     B
```

As long as there are no ties (i.e., people with the exact same awesomeness score), then the test that we want to do is surprisingly simple. All we have to do is construct a table that compares every observation in group A against every observation in group B. Whenever the group A datum is larger, we place a check mark in the table:

		group B				
		14.5	10.4	12.4	11.7	13.0
group A	6.4	.	.	.	.	.
	10.7	.	✓	.	.	.
	11.9	.	✓	.	✓	.
	7.3	.	.	.	.	.
	10.0	.	.	.	.	.

We then count up the number of checkmarks. This is our test statistic, W.[200] The actual sampling distribution for W is somewhat complicated, and I'll skip the details. For our purposes, it's sufficient to note that the interpretation of W is qualitatively the same as the interpretation of t or z. That is, if we want a two-sided test, then we reject the null hypothesis when W is very large or very small; but if we have a directional (i.e., one-sided) hypothesis, then we only use one or the other.

The structure of the `wilcox.test()` function should feel very familiar to you by now. When you have your data organised in terms of an outcome variable and a grouping variable, then you use the `formula` and `data` arguments, so your command looks like this:

```
wilcox.test( formula = scores ~ group, data = awesome)
```

```
##
##  Wilcoxon rank sum test
##
## data:  scores by group
## W = 3, p-value = 0.05556
## alternative hypothesis: true location shift is not equal to 0
```

Just like we saw with the `t.test()` function, there is an `alternative` argument that you can use to switch between two-sided tests and one-sided tests, plus a few other arguments that we don't need to worry too much about at an introductory level. Similarly, the `wilcox.test()` function allows you to use the `x` and `y` arguments when you have your data stored separately for each group. For instance, suppose we use the data from the `awesome2.Rdata` file:

```
load( "./rbook-master/data/awesome2.Rdata" )
score.A
```

```
## [1]  6.4 10.7 11.9  7.3 10.0
```

```
score.B
```

```
## [1] 14.5 10.4 12.9 11.7 13.0
```

When your data are organised like this, then you would use a command like this:

```
wilcox.test( x = score.A, y = score.B )
```

```
##
##  Wilcoxon rank sum test
##
## data:  score.A and score.B
## W = 3, p-value = 0.05556
## alternative hypothesis: true location shift is not equal to 0
```

The output that R produces is pretty much the same as last time.

11.10.2 sample Wilcoxon test

What about the ***one sample Wilcoxon test*** (or equivalently, the paired samples Wilcoxon test)? Suppose I'm interested in finding out whether taking a statistics class has any effect on the happiness of students. Here's my data:

```
load( "./rbook-master/data/happy.Rdata" )
print( happiness )
```

```
##    before after change
## 1      30     6    -24
## 2      43    29    -14
## 3      21    11    -10
## 4      24    31      7
## 5      23    17     -6
## 6      40     2    -38
## 7      29    31      2
## 8      56    21    -35
## 9      38     8    -30
## 10     16    21      5
```

What I've measured here is the happiness of each student `before` taking the class and `after` taking the class; the `change` score is the difference between the two. Just like we saw with the t-test, there's no fundamental difference between doing a paired-samples test using `before` and `after`, versus doing a one-sample test using the `change` scores. As before, the simplest way to think about the test is to construct a tabulation. The way to do it this time is to take those change scores that are positive valued, and tabulate them against all the complete sample. What you end up with is a table that looks like this:

		all differences								
		-24	-14	-10	7	-6	-38	2	-35	-30	5
positive differences	7	.	.	.	✓	✓	.	✓	.	.	✓
	2	.	.	.	.	.	.	✓	.	.	.
	5	.	.	.	.	.	.	✓	.	.	✓

Counting up the tick marks this time, we get a test statistic of V=7. As before, if our test is two sided, then we reject the null hypothesis when V is very large or very small. As far of running it in R goes, it's pretty much what you'd expect. For the one-sample version, the command you would use is

```
wilcox.test( x = happiness$change,
             mu = 0
)
```

```
##
##  Wilcoxon signed rank test
##
## data:  happiness$change
## V = 7, p-value = 0.03711
## alternative hypothesis: true location is not equal to 0
```

As this shows, we have a significant effect. Evidently, taking a statistics class does have an effect on your happiness. Switching to a paired samples version of the test won't give us different answers, of course; but here's the command to do it:

```
wilcox.test( x = happiness$after,
             y = happiness$before,
             paired = TRUE
)
```

```
##
##  Wilcoxon signed rank test
##
## data:  happiness$after and happiness$before
## V = 7, p-value = 0.03711
## alternative hypothesis: true location shift is not equal to 0
```
