Nonparametric statistics

Patrick Breheny

November 10, 2016

Today's lab is about nonparametric statistics. We will discuss the use of R's built-in function, `wilcox.test`, for carrying out the Wilcoxon rank sum test, then do some programming and create our own permutation test and construct a bootstrap confidence interval.

1 `wilcox.test`

Let's begin by using a built-in function in R for nonparametric testing. To illustrate its use, let's work with the lead smelter data we've discussed in class. The primary syntax of the function is essentially identical to that of `t.test`:

```
> lead <- read.delim("http://myweb.uiowa.edu/pbreheny/data/lead-iq.txt")
> t.test(IQ~Smelter, lead)


Welch Two Sample t-test

data:  IQ by Smelter
t = 1.38, df = 120.62, p-value = 0.1702
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -1.518663  8.505833
sample estimates:
 mean in group Far mean in group Near
          92.68657           89.19298

> wilcox.test(IQ~Smelter, lead)


Wilcoxon rank sum test with continuity correction

data:  IQ by Smelter
W = 2217.5, p-value = 0.1229
alternative hypothesis: true location shift is not equal to 0

> x <- with(lead, IQ[Smelter=="Near"])
> y <- with(lead, IQ[Smelter=="Far"])
> wilcox.test(x, y) ## Same thing


Wilcoxon rank sum test with continuity correction
```

```
data:  x and y
W = 1601.5, p-value = 0.1229
alternative hypothesis: true location shift is not equal to 0
```

In this particular case, as you may recall, the data is quite normal-looking, and the agreement between the parametric and nonparametric test is not surprising. However, in the presence of outliers, the two might give very different results. Let's see what happens if we add a super-genius to the "near smelter" group:

```
> x <- c(x, 210)
> t.test(x, y)$p.value

[1] 0.666513

> wilcox.test(x, y)$p.value

[1] 0.1747088
```

As we mentioned in class, the Wilcoxon rank sum test is a permutation test. In general, exact solutions to permutation tests are quite time-consuming to calculate, although there are some very clever algorithms for speeding things up when you're considering sums of permuted consecutive positive integers (i.e., what you're doing in a rank sum test). Now, this won't actually work for the lead smelter data because there are ties – eight kids, for example, have an IQ of 96 – so the ranks aren't consecutive integers anymore. So unfortunately, even if you ask for an exact test, `wilcox.test` can't give you one:

```
> wilcox.test(IQ~Smelter, lead, exact=TRUE)

Warning in wilcox.test.default(x = c(70L, 85L, 86L, 76L, 96L, 94L, 115L, :  cannot compute
exact p-value with ties


Wilcoxon rank sum test with continuity correction

data:  IQ by Smelter
W = 2217.5, p-value = 0.1229
alternative hypothesis: true location shift is not equal to 0
```

For the sake of illustration, though, let's just remove the ties by adding a small amount of random noise to all the observations (this is known as "jittering"):

```
> lead2 <- lead
> lead2$IQ <- lead2$IQ + runif(nrow(lead2), -0.1, 0.1)
> wilcox.test(IQ~Smelter, lead2, exact=TRUE)     ## Exact result


Wilcoxon rank sum test

data:  IQ by Smelter
W = 2222, p-value = 0.118
alternative hypothesis: true location shift is not equal to 0

> wilcox.test(IQ~Smelter, lead2)                 ## Approximation 1
```

```
Wilcoxon rank sum test with continuity correction

data:  IQ by Smelter
W = 2222, p-value = 0.1178
alternative hypothesis: true location shift is not equal to 0
```

```
> wilcox.test(IQ~Smelter, lead2, correct=FALSE)  ## Approximation 2


Wilcoxon rank sum test

data:  IQ by Smelter
W = 2222, p-value = 0.1172
alternative hypothesis: true location shift is not equal to 0
```

In this case, with $n = 124$, both approximations are very accurate. At the same time, it didn't take very long to compute the exact result, so there is little reason in this case to prefer the appoximation. The computational burden increases dramatically, however, with larger sample sizes. For example, with $n = 400$, the exact result takes thousands of times longer to compute than the approximation:

```
> x <- runif(200)
> y <- runif(200)
> system.time(wilcox.test(x, y, exact=TRUE))

   user  system elapsed
  2.946   0.172   3.121

> system.time(wilcox.test(x, y))

   user  system elapsed
  0.002   0.000   0.001
```

Finally, as we discussed in class, one can invert the Wilcoxon rank sum test to obtain a semiparametric confidence interval for the difference in location for two groups:

```
> x <- with(lead, IQ[Smelter=="Near"])
> y <- with(lead, IQ[Smelter=="Far"])
> wilcox.test(x, y, mu=3) ## This is the test


Wilcoxon rank sum test with continuity correction

data:  x and y
W = 1374.5, p-value = 0.007337
alternative hypothesis: true location shift is not equal to 3
```

```
> wilcox.test(x, y+3)     ## we are inverting


Wilcoxon rank sum test with continuity correction

data:  x and y + 3
W = 1374.5, p-value = 0.007337
alternative hypothesis: true location shift is not equal to 0
```

```
> wilcox.test(x, y, conf.int=TRUE)


Wilcoxon rank sum test with continuity correction

data:  x and y
W = 1601.5, p-value = 0.1229
alternative hypothesis: true location shift is not equal to 0
95 percent confidence interval:
 -8.000016  1.000021
sample estimates:
difference in location
             -3.999971
```

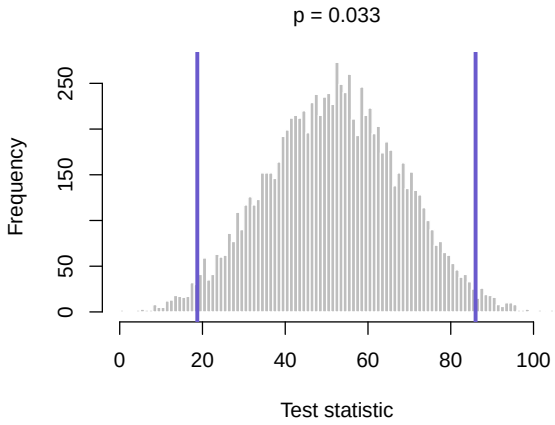
It is again worth noting that this interval is quite similar to the confidence interval we get from the t test for the difference in mean IQ between the two groups.

2 The Wilcoxon signed rank test

In class, we discussed the idea of nonparametric testing for two-sample studies, but what about one-sample studies like our cystic fibrosis crossover experiment? The idea of conditioning on the observed responses and putting them into the same urn doesn't apply here. How can we construct a nonparametric test in the one-sample case?

Frank Wilcoxon had an idea for this situation as well, and his idea was this: consider instead conditioning on the *absolute values* of the observations. Under the null, these values are just as likely to be positive as negative (i.e., a patient is just as likely to do better on the drug as on the placebo), so we can randomly assign each observation a $+/-$ sign under the null. Let's implement this idea in R:

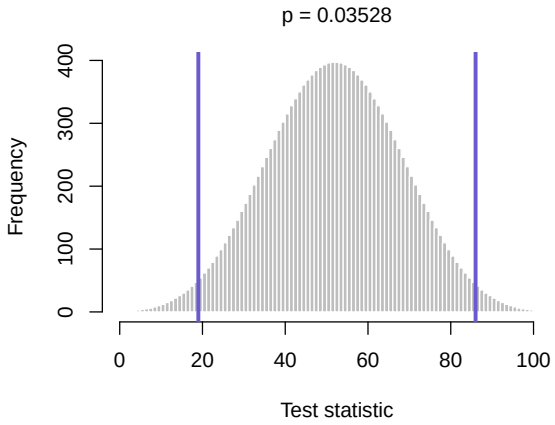
```
> cf <- read.delim("http://myweb.uiowa.edu/pbreheny/data/cystic-fibrosis.txt")
> Diff <- apply(cf, 1, diff)
> r <- rank(abs(Diff))  ## Absolute ranks
> Obs <- sum(r[Diff>0]) ## Sum of positive ranks
> n <- nrow(cf)
> N <- 10000
> ts <- numeric(N)
> for (i in 1:N) {
+   s <- rbinom(n, 1, 0.5) ## Random signs
+   ts[i] <- sum(r[s==1])
+ }
> hist(ts, col="gray", border="white", xlim=c(0,105), breaks=0:105,
+      main="", xlab="Test statistic")
> v <- c(mean(ts) - (Obs-mean(ts)), Obs)
> abline(v=v, col="slateblue", lwd=3)
> p <- sum(ts <= v[1] | ts >= v[2])/N
> mtext(paste("p =",p), line=1)
```



In this particular case, the sample size is small enough where we can work out the exact test ourselves:

```
> l <- rep(list(0:1), n)
> X <- as.matrix(expand.grid(l))
> N <- nrow(X)
> ts <- numeric(N)
> pb <- txtProgressBar(1, N, style=3)
> for (i in 1:N) {
+   ts[i] <- sum(r[X[i,]==1])
+   setTxtProgressBar(pb, i)
+ }
```

```
> tab <- table(ts)
> obs <- as.character(Obs)
> v <- as.numeric(names(tab)[tab==tab[obs]])
> hist(ts, col="gray", border="white", xlim=c(0,105), breaks=0:105,
+      main="", xlab="Test statistic")
> abline(v=v, col="slateblue", lwd=3)
> p <- sum(tab[tab <= tab[obs]])/N
> mtext(paste("p =", round(p, 5)), line=1)
```



The purpose of the above programming was mainly pedagogical, since the `wilcox.test` function can be used for Wilcoxon signed rank tests as well:

```
> wilcox.test(Diff)


Wilcoxon signed rank test

data:  Diff
V = 86, p-value = 0.03528
alternative hypothesis: true location is not equal to 0
```

Note that we obtain exactly the same p-value from `wilcox.test` that we got from our do-it-yourself version, which is reassuring.

As with the rank sum test, the signed rank test can be inverted to form a semiparametric confidence interval:

```
> wilcox.test(Diff, mu=4) ## Test of location


Wilcoxon signed rank test

data:  Diff
V = 83, p-value = 0.05798
alternative hypothesis: true location is not equal to 4

> wilcox.test(Diff-4)     ## Same thing


Wilcoxon signed rank test

data:  Diff - 4
V = 83, p-value = 0.05798
alternative hypothesis: true location is not equal to 0

> wilcox.test(Diff, conf.int=TRUE)
```

```
Wilcoxon signed rank test

data:  Diff
V = 86, p-value = 0.03528
alternative hypothesis: true location is not equal to 0
95 percent confidence interval:
   3.5 251.0
sample estimates:
(pseudo)median
         113.5
```

What is this a confidence interval for? As the output indicates, it's for something called the "pseudomedian". OK, but what if we want a confidence interval for the actual median? This is a perfect time to use the bootstrap:

```
> B <- 10000
> b <- numeric(B)
> for (i in 1:B) {
+   x <- sample(Diff, replace=TRUE)
+   b[i] <- median(x)
+ }
> quantile(b, c(.025, .975))                ## Percentile interval

 2.5% 97.5%
 13.5 185.0

> median(Diff) + qnorm(c(.025, .975))*sd(b) ## "z" interval

[1]  26.9154 192.0846
```

Generally, the bootstrap percentile interval is superior to the bootstrap z interval. It is possible to construct other, even more accurate bootstrap confidence intervals as well, although such intervals lie beyond the scope of this course. If you are interested, however, there is a nice `R` package, `boot`, that will construct such intervals (known as BC_a intervals) for you.