Wald-Wolfowitz (Runs Test)

Ryan Honea

9/6/2017

Intro To Wald-Wolfowitz Test

Introduction to the Test

Assume you are sitting in a classroom with Professor X, and his tests seem to have a pattern to them (i.e. not random), and you're curious about testing it. You observe the following sequence of T/F answers:

$$T \quad F \quad T \quad F \quad T \quad F \quad T \quad F \quad T \quad F$$

Upon observing this pattern, a student might decide to continue that pattern in hopes of little effort.

Wald-Wolfowitz Solution

Abraham Wald and Jacob Wolfowitz developed a procedure using simple counting techniques to calculate the probability of a sequence of observations occurring.

- Count the number of “runs” or the number of times the same observations occur one after the other (10 runs in the case of the example)
- Count the number of times observation one and observation two appear
- Use equations (1) and (2) depending on an even or odd number of runs

Wald-Wolfowitz Equations for Test

$$P(r = 2k+1) = \frac{\binom{n-1}{k-1}\binom{m-1}{k} + \binom{n-1}{k}\binom{m-1}{k-1}}{\binom{m+n}{n}} \tag{1}$$

$$P(r = 2k) = \frac{2\binom{n-1}{k-1}\binom{m-1}{k-1}}{\binom{m+n}{n}} \tag{2}$$

Crafting the Test in R

Runs Counting

First step is to be able to determine the number of runs that occur in a test

```
num_runs <- function(seq) {
  sum = 1
  for (i in 1:(length(seq) - 1)) {
    if (seq[i] != seq[i+1]) sum = sum + 1
  }
  return(sum)
}
```

Even/Odd Calculations

Next step is to be able to determine the p-value based on the number of runs and number of occurrences of observation 1 and observation 2.

```
even_calc <- function(runs, n, m) {
  k = runs/2
  numerator = choose(n-1, k-1)*choose(m-1,k-1)
  denominator = choose(m + n, n)
  return(2*numerator/denominator)
}
```

Even/Odd Calculations

```
odd_calc <- function(runs, n, m) {
  k = runs%/%2
  numerator = choose(n-1,k)*choose(m-1,k-1) +
    choose(n-1,k-1)*choose(m-1,k)
  denominator = choose(m + n, n)
  return(numerator/denominator)
}
```

Full Function

Using the previous functions, we can create the full test

```
ww.test <- function(seq) {
  seq = factor(seq)
  levels(seq) <- c(0,1)
  runs = num_runs(seq)
  n = length(seq[seq == 0])
  m = length(seq[seq == 1])
  if (runs %% 2 == 0) {
    p.value = even_calc(runs, n, m)
  } else {
    p.value = odd_calc(runs, n, m)
  }
  return(p.value)
}
```

Testing Function on Initial Problem

So now we have the tools to see how likely it is that Professor Xavier's sequence of True Falses shows up!

```
tf_questions <- c('T','F','T','F','T','F','T','F','T','F')
cat("P-value:",ww.test(tf_questions))
```

```
## P-value: 0.007936508
```

And so perhaps this is truly a pattern and the results are not appearing randomly.

Testing the Test on the Standard Normal Distribution

Converting the Standard Normal to a 1/0 Sequence

- Simulate *n* observations from the Standard Normal
- Any element below the mean is a 0, any element above the mean is a 1, otherwise ignore
- Perform test on that sequence of observations

$n = 30$ Example

```
for (i in 1:5) {
  sim <- rnorm(30)
  sim[sim < 0] = '-'
  sim[sim > 0] = '+'
  sim = sim[sim != 0]
  cat("P-value =", ww.test(sim), "\n")
}
```

```
## P-value = 0.12075
## P-value = 0.1328837
## P-value = 0.12075
## P-value = 0.02460809
## P-value = 0.1431592
```

Success of Test for $n = 30$

We can test how often this test is successful by repeating this simulation

```
rejected = 0
sims = 100
for (i in 1:sims) {
  sim <- rnorm(30)
  sim[sim < 0] = '-'
  sim[sim > 0] = '+'
  sim = sim[sim != 0]
  if (ww.test(sim) <= 0.05) rejected = rejected + 1
}
cat("Success:",(sims-rejected)/sims)
```

```
## Success: 0.81
```

Success of Test for Increasing n