Wald-Wolfowitz Runs Test in R

Ryan Honea

9/22/2017

Introduction to the Wald-Wolfowitz Runs Test

Suppose one wants to test the randomness of a sequence of 2 different symbols. For example, you suspect that your professor has a pattern that is being used when creating their True-False test, and that they aren't randomly deciding when they will have a true value, and when they will have a false value. For example, say you observe on a test the following pattern

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

which appears to be far from randomly distributed.

Abraham Wald and Jacob Wolfowitz devised a method to test whether or not a series of sequences (or runs) are truly randomly distributed. The test uses basic counting principles to calculate the probability of such a series occurring. Where r is the number of runs, the below formulas determine those probabalities for an odd number of runs and an even number of runs.

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

Crafting the Test in R

The first step in running this test given any data set then is to determine the number of runs which is done by the function below. Note that the sum of runs starts at 1 because a sequence that is all the same is still a single run.

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

Next, one must determine whether or not there is an even or odd number of runs which can be done with a simple if statement One also needs to know n and m, i.e. the number of occurrences of each variable. Using R, this is easy. This allows us to calculate the p-value for even or odd numbers of runs.

```
p_calc <- function(runs, n, m) {
  if (runs %% 2 == 0) {
     k = runs/2
    numerator <- choose(n-1, k-1)*choose(m-1,k-1)
    denominator <- choose(m + n, n)
    return(2*numerator/denominator)
  } else {
    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)
  }
}
```

In order to read any sequence as different factors, a function must converse a sequence into two levels, and then using the levels function in R, one can convert everything to a 0 and 1 for standardization purposes. The below code is the test function completed.

```
ww.test <- function(seq) {
  left <- 0
  right <- 0
  seq <- factor(seq)
  levels(seq) <- c(0,1)
  runs <- num_runs(seq)
  n <- length(seq[seq == 0])
  m <- length(seq[seq == 1])
  for (i in 1:runs) {
    left <- left + p_calc(i, n, m)
  }
  for (i in runs:(n+m)) {
    right <- right + p_calc(i, n, m)
  }
  return(2*min(left, right))
}
```

And so if we were to test this on our original values, we get the following

```
answers = c('T','T','F','F','T','T','F','F','T','T')
cat("P-Value: ", ww.test(answers))
```

```
## P-Value:  0.8095238
```

And so it isn't entirely unlikely that this sequence would appear. However, what if the below sequence appeared?

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

```
answers = c('T','F','T','F','T','F','T','F','T','F')
cat("P-Value: ", ww.test(answers))
```

```
## P-Value:  0.01587302
```

With a p-value this low, one could reasonably assume that this is not randomly distributed and so repeating the pattern wouldn't necessarily be an awful idea.

We can also test the other extreme

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

```
answers = c('T','T','T','T','T','F','F','F','F','F')
cat("P-Value: ", ww.test(answers))
```

```
## P-Value:  0.01587302
```

which yields the same as the first extreme which is to be expected.

Utilizing the Wald-Wolfowitz Test on Observations from Two Distributions

Now, assume that one is sampling from two distributions and is curious about whether or not these distributions are the same (without knowing the true distribution). So,

$$x_1, ..., x_n \sim F(x) y_1, ..., y_n \sim G(y)$$

Therefore, we have

$$H_0 : F(x) = G(y) H_a : F(x) \neq G(y)$$

To test this, we sample n and m observations from $G(x)$ and $G(y)$ respectively, label them by the distribution they were sampled from, and order them by number. In doing so, one gains a sequence of labels that can be tested on using the Wald-Wolfowitz test. For example,

$$1, 4, 2, 9, 5 \sim F(x) 3, 0, 8, 7, 6 \sim G(x)$$

which ordered is

$$(0, y) \quad (1, x) \quad (2, x) \quad (3, y) \quad (4, x) \quad (5, x) \quad (6, y) \quad (7, y) \quad (8, y) \quad (9, x)$$

or

$$y \quad x \quad x \quad y \quad x \quad x \quad y \quad y \quad y \quad x$$

If the two distributions are the same, the observations should be well interspersed with one another with the ideal being complete alternation (i.e. x, y, x, y, x…). So if they are not the same then, the number of runs should be low as the observations would be more seperated where the most extreme case would be no alternation (i.e. x, x, x, y, y, y). Therefore, we need to find c such that $P(R < c) \leq \alpha$ in order to obtain a rejection region.

With this information, we can create a new function that takes two samples as opposed to one (such as earlier), and assesses the null hypothesis that the distributions are equal.

```r
ww.dist.test <- function(s_1, s_2) {
  n = length(s_1)
  m = length(s_2)
  seq <- rbind(cbind(s_1,rep(0,n)),cbind(s_2,rep(1,m)))
  seq <- seq[order(seq[,1]),]
  runs = num_runs(seq[,2])
  p = 0
  for (i in 2:runs) {
    p = p + p_calc(i, n, m)
  }
  cat(runs,":",p,"\n")
}
```

We will test this on three different cases. One case of an equal distribution, and two cases of different distributions.

```r
ww.dist.test(rnorm(25), rnorm(25))
```

```
## 29 : 0.8406756
```

```r
ww.dist.test(rnorm(25), rgamma(25,5,5))
```

```
## 19 : 0.03084717
```

```r
ww.dist.test(rnorm(25), rexp(25, 5))
```

```
## 15 : 0.001152096
```

So, it appears to be working as intended.