427 Project

Max Barshay

5/19/2021

Overview

For this project, I am going to experiment with the differences in permutation testing when sampling with and without replacement. In particular, I am going to experiment with the power of the tests.

The question I want to answer is:

In what situations does a permutation test do better than the corresponding test that resamples with replacement?

Code

This one repititon functions runs one repition of my simulation. More specifically, this function performs a permutation test with or without replacement and determines if the null hypothesis (which asks if the populations are the same) is rejected.

```
one_repition <- function(distribution, x_param, y_param, n_x, n_y, reps, alpha, replacement){
  if (distribution == "normal"){
    x_sample <- rnorm(n_x, x_param, 1)
    y_sample <- rnorm(n_y, y_param, 1)
  }
  else if (distribution == "exponential"){
    x_sample <- rexp(n_x, x_param)
    y_sample <- rexp(n_y, y_param)
  }
  else if (distribution == "t"){
    x_sample <- rt(n_x, x_param)
    y_sample <- rt(n_y, y_param)
  }
  obs_difference <- abs(mean(x_sample) - mean(y_sample))
  perm_diffs <- numeric(reps)
  one_sample <- c(x_sample, y_sample)
  if (replacement){
      for (i in 1:reps){
        x_sample <- sample(one_sample, n_x)
        y_sample <- setdiff(one_sample, x_sample)
        perm_diff <- abs(mean(x_sample) - mean(y_sample))
        perm_diffs[i] <- perm_diff
    }
  }
```

```
  else if (!replacement){
      for (i in 1:reps){
        x_sample <- sample(one_sample, n_x, replace=T)
        y_sample <- sample(one_sample, n_y, replace=T)
        perm_diff <- abs(mean(x_sample) - mean(y_sample))
        perm_diffs[i] <- perm_diff
    }
  }
  p_value <- mean(perm_diffs > obs_difference)
  if (p_value < alpha){
    return (1)
  } else {
    return (0)
  }
}
```

This function just runs the above function many times and computes the estimated power of the procedure. This can also be used to compute type one error depending on the parameters that we use.

```
many_repititions <- function(distribution, x_param, y_param, n_x, n_y, reps, alpha, replacement, meta_r
  counter <- 0
  for (i in 1:meta_reps){
    result <- one_repition(distribution, x_param, y_param, n_x, n_y, reps, alpha, replacement)
    counter <- counter + result
  }
  return (counter / meta_reps)
}
```

One thing that I would like to point out before going on is that this algorithm is very slow. There is an unavoidable nested for loop based on how the simulation has to work and so to run 10,000 reps in both the inner and outer loops amount to running 10,000*10,000 = 100,000,000 repititions which takes too long.

We know that observed power is a sample proportion and so in the worst case scenario my estimate is going to have standard error of $\frac{1}{\sqrt{n}}$. There is some propogation of errors that I am ignoring and to save time and computer power I am only going to run the loop for computing the p-value 1,000 times.

With that being said, I am going to run the power loop 1,000 times. Thus, ignoring propogation of errors, I am going to say the standard error of my power estimates are $\frac{1}{\sqrt{1000}} \approx .03$.

Examples

I will test some examples when the alternative is true.

```
many_repititions("normal", 0, .1, 30, 30, 1000, .05, T, 1000)
```

```
## [1] 0.09
```

```
many_repititions("normal", 0, .1, 30, 30, 1000, .05, F, 1000)
```

```
## [1] 0.075
```

```
many_repititions("exponential", .5, 1, 30, 30, 1000, .05, T, 1000)
```

```
## [1] 0.706
```

```
many_repititions("exponential", .5, 1, 30, 30, 1000, .05, F, 1000)
```

```
## [1] 0.728
```

```
many_repititions("t", 3, 2, 30, 30, 1000, .05, T, 1000)
```

```
## [1] 0.062
```

```
many_repititions("t", 3, 2, 30, 30, 1000, .05, F, 1000)
```

```
## [1] 0.036
```

The results are somewhat inconclusive for these distributions. I am not sure what results are there on the final iteration I make, but these are all within .03 and I have seen many swaps in terms of which one was producing higher power. I can say with a reasonable level of confidence that the decision to sample with or without replacement does not matter for these three named distributions when the alternative is true.

I wonder what happens when the null hypothesis is true. In this case the estimates we have are the type 1 error esimates. Lets see how the estimators compare here.

```
many_repititions("normal", 0, 0, 30, 30, 1000, .05, T, 1000)
```

```
## [1] 0.052
```

```
many_repititions("normal", 0, 0, 30, 30, 1000, .05, F, 1000)
```

```
## [1] 0.059
```

```
many_repititions("exponential", 1, 1, 30, 30, 1000, .05, T, 1000)
```

```
## [1] 0.057
```

```
many_repititions("exponential", 1, 1, 30, 30, 1000, .05, F, 1000)
```

```
## [1] 0.043
```

```
many_repititions("t", 2, 2, 30, 30, 1000, .05, T, 1000)
```

```
## [1] 0.045
```

```
many_repititions("t", 2, 2, 30, 30, 1000, .05, F, 1000)
```

```
## [1] 0.037
```

From these initital tests, it does not seem like there is a significant difference in the results between situations where we choose to perform the permutation test with replacement and without regardless of if the null hypothesis is true or not. One possible reason for this could be the fact that these distributions do not have many outliers. I tried to pick the t distribution with 2 degrees of freedom so it would have many outliers, but I don't see any definitive differences between using the permutation test with or without repalacement.

Outliers

The only thing that comes to mind that could force there to be a difference between these two methods is if the distributions that we are looking at have many outliers.

However, I am not certain what will happen in this case. On one hand, if we are sampling with replacement and there is one extremely large value, then we may never select this value when resampling to one group which would make the mean of this group smaller. On the other hand, if we are sampling with replacement we may select this one extremely large value many times to one group as opposed to the once guranteed when sampling without replacement which would make the mean of that group larger.

If I had to guess, I would say these opposing forces are going to cancel each other out and we are not going to see anything that interesting. But, maybe I am just a pessimist.

I will need to adjust some code. Here are the new funcitons:

```
one_repition_outliers <- function(reps, alpha, replacement, x_sample_input, y_sample_input){
  obs_difference <- abs(mean(x_sample_input) - mean(y_sample_input))
  perm_diffs <- numeric(reps)
  one_sample <- c(x_sample_input, y_sample_input)
  n_x <- length(x_sample_input)
  n_y <- length(y_sample_input)
  if (replacement){
      for (i in 1:reps){
        x_sample <- sample(one_sample, n_x)
        y_sample <- setdiff(one_sample, x_sample)
        perm_diff <- abs(mean(x_sample) - mean(y_sample))
        perm_diffs[i] <- perm_diff
    }
  }
  else if (!replacement){
      for (i in 1:reps){
        x_sample <- sample(one_sample, n_x, replace=T)
        y_sample <- sample(one_sample, n_y, replace=T)
        perm_diff <- abs(mean(x_sample) - mean(y_sample))
        perm_diffs[i] <- perm_diff
    }
  }
  p_value <- mean(perm_diffs > obs_difference)
  if (p_value < alpha){
    return (1)
  } else {
    return (0)
```

```
  }
}

many_repititions_outliers <- function(reps, alpha, replacement, x_sample, y_sample, meta_reps){
  counter <- 0
  for (i in 1:meta_reps){
    result <- one_repition_outliers(reps, alpha, replacement, x_sample, y_sample)
    counter <- counter + result
  }
  return (counter / meta_reps)
}
```

Generating Samples with Outliers

```
x_sample_init <- rnorm(9,0,1)
x_sample <- c(x_sample_init, 1000)
y_sample_init <- rnorm(9,1,1)
y_sample <- c(y_sample_init, 1000.01) # set difference kills duplicates
```

```
many_repititions_outliers(1000, .05, T, x_sample, y_sample, 1000)
```

```
## [1] 0
```

```
many_repititions_outliers(1000, .05, F, x_sample, y_sample, 1000)
```

```
## [1] 0
```

In this case of skewed data, where we have two normal distributions with varying means, we are pretty much never going to reject the null hypothesis. The reason being that the estimated p-value which is the proportion of permuted differences that are larger than the observed difference is going to be large because the observed difference has the outlier in both distributions here so the observed difference is going to be small. But then it gets shuffled where the groups have uneven large numbers in which there permuted difference is going to be large, or they get lucky and have the same number of large numbers in which case the actual parameters of the normal distributions come into play. I think more often then not considering both replacement and without there will not be the same number of large numbers per group.

To remedy this, I can create just one sample with large outliers. This way the observed difference will be larger and so we are more likely to get a smaller p-value.

```
x_sample_init2 <- rnorm(7,0,1)
x_sample2 <- c(x_sample_init2, 1000, 1000.01, 1000.02)
y_sample2 <- rnorm(10, 1, 1)
```

```
many_repititions_outliers(1000, .05, T, x_sample2, y_sample2, 1000)
```

```
## [1] 0
```

```
many_repititions_outliers(1000, .05, F, x_sample2, y_sample2, 1000)
```

```
## [1] 0
```

Again, we get zero here meaning that we never reject the null hypothesis. This could be because 3 (the number of very large numbers I am injecting in) is odd and so there cannot be situations in which we put the same number of large numbers in each group which would lead to a smaller absolute difference in means.

I did some debugging by printing output and I saw that the problem is that in the cases where all 3 large numbers go to one group we are getting a difference in means of about 300. But all the ones where 3 go into one group are likely to be just above the observed difference because the normal distribution that we add the large values to has a higher mean than the other normal distribution that we do not add noise to. I can illustrate this changes when I switch the means of the normal distributions.

```
x_sample_init3 <- rnorm(7,1,1) # swtiched 0 to 1
x_sample3 <- c(x_sample_init2, 1000, 1000.01, 1000.02)
y_sample3 <- rnorm(10, 0, 1) # switched 1 to 0
```

```
many_repititions_outliers(1000, .05, T, x_sample3, y_sample3, 1000)
```

```
## [1] 0
```

```
many_repititions_outliers(1000, .05, F, x_sample3, y_sample3, 1000)
```

```
## [1] 0
```

That changed things up a lot. It seems crazy that switching that 0 to a 1 makes that power go from roughly 0 to 1, but it makes sense given what I said in the previous section.

One thing that I have not tried yet is very uneven sample sizes. For example, maybe if one sample size is much larger than the other and one has very extreme outliers then there could be a difference between simulating with and without replacement.

```
x_sample4 <- c(1000, 1000.01, 1000.02)
y_sample4 <- rnorm(50, 0, 1)
```

```
many_repititions_outliers(1000, .05, T, x_sample4, y_sample4, 1000)
```

```
## [1] 1
```

```
many_repititions_outliers(1000, .05, F, x_sample4, y_sample4, 1000)
```

```
## [1] 1
```

The problem with doing these extreme outlier cases is that the power is likely going to be 1. The reason being that the inital observed difference in means is going to be very large (since one distribution has much larger values than the other). Any reshuffle, either with or without replacement, will not lead to as big of a difference in means, therefore the p-value will always be very small and so we will always reject.

I can alleviate this by continuing to have uneven group sizes, however now I can make the initial distribution have the same number of extreme outliers.

Now, since the simulated normal values are centered at 0, these two samples should have a small observed difference in means and more small p-values.

```r
x_sample5 <- c(1000, 1000.01)
y_sample5 <- c(rnorm(48, 0, 1), c(2000.02, 2000.03))
```

```r
many_repititions_outliers(1000, .05, T, x_sample5, y_sample5, 1000)
```

```
## [1] 0.982
```

```r
many_repititions_outliers(1000, .05, F, x_sample5, y_sample5, 1000)
```

```
## [1] 1
```

In the repitition that I just did, we see that the difference in these estimates is larger than the .03 standard error that I would expect. However, due to propogation of errors, I am not sure if this is a significant result.

Conclusion

In conclusion, I think that I underestimated how difficult this study would be for a variety of reasons.

First, the fact that this simulation is inherently a two-stage process makes the code to run it very slow. As a result, I was forced between either running very few simulations precisely or running many but having high margins of error. I chose the latter but as a result, I am very hestitant to claim that any results that I found were not due to simulation margin of error.

Second, the fact that simulating from common distributions gives approximately the same results for both simulating with and without replacement. This forced me to use very odd distributions with many outliers, but as you have seen above this lead to many other problems. I think that solving these other problems taught me more than this simulation study taught me about permutation tests with and without replacement.

Overall, given these caveats, I would say that in most normal situations it does not matter if the permutation test is done with our without replacement in terms of power. However, just looking at the theory of a permutation test and what the null hypothesis is saying it makes much more sense to sample without replacement. Therefore, since I did not find evidence sampling with replacement incresases power, I would recommend continuing to sample without replacement for permutation tests.