Multiple Comparisons

Brooke Coneeny

# Familywise Error Rate

When completing a statistical test, we have a null hypothesis which we decide to either reject or accept based off of our observed data. A occurs when is true but we rejected it. In the cases of multiple comparisons, we have multiple individual hypotheses which leads us towards a , which is the probability of making any Type 1 errors.

FWER = P (at least one of ,…, is falsely rejected)

Similarly, a 95% confidence level interval for a parameter may fail to cover 5% of the time. Constructing multiple 95% confidence intervals { for multiple different parameters , the chance at least one of the intervals fails to include the parameter is greater than 5%.

P (Making an error)

P (Not making an error)

P (Not making an error in k tests)

P (Making at least 1 error in k tests)

For example, before a presidential election, polls are conducted in all 50 states and 95% confidence intervals are reported for the proportion of people in each state who favor the incumbent candidate.

A question we may ask ourselves is: what is the probability that at least one of these intervals fails to contain the true proportion? This is asking what the familywise error rate is for this situation.

FWER = = 0.923

This is a relatively high chance that at least one of our intervals will fail to contain the true proportion. Another question we may ask ourselves is: What level should be used to have a FWER of 0.05? This is the same as asking: What level should be used to have the probability of making no errors 0.95?

Solving for we get that . This value for is much smaller than our original value of 0.05, making it more difficult to reject hypotheses.

# K means tests

A simple hypothesis test compares means. To complete this we use a simple t-test. However, increasing *k* creates a more complex problem because of the increase in type 1 error that occurs. As we showed above, the FWER is equal to , which is why as *k* increases the FWER also increases. To complete this multiple mean comparison test, we use an ANOVA F-test.

For an experiment with treatments, there are

* pairwise comparisons to make
* numerous contrasts

Because there is a larger number of tests being performed, it is easier to find these type 1 errors and get high FWER values. Two well known methods to combat this issue are the Bonferroni Method and Holm-Bonferroni method.

# Bonferroni Method

The Bonferroni method guarantees by decreasing the level for all the individual tests to . Therefore we reject hypotheses only when .

P(any Type 1 error) P(Type 1 error for test i)

* Works even if the test statistics are not independent
* Rejects a null if the comparionswise p-value is less than
* Works OK when is small, very conservative if is large
* Ideal when interested in a smaller number of planned contrasts or pairwise comparisons

# Holm-Bonferroni Method

The Bonferroni correction can be overly conservative and there are other tests with more power such as the Holm-Bonferroni Method. Holm-Bonferroni suggests that we begin by sorting the tests in order of their obtained p-values.

This method can be thought out as an algorithm:

1. Sort the -values into increasing order, label these and denote the corresponding null hypotheses as

2. Let be the smallest index such that . If no such exists, then reject all hypotheses. If , then do not reject any hypothesis.

3. Otherwise, reject the null hypotheses and accept the null hypotheses

# Example Problem

Supposed we have 5 different hypothesis tests with p-values of:

Test 1 (p = 0.07)

Test 2 (p = 0.002)

Test 3 (p= 0.011)

Test 4 (p = 0.003)

Test 5 (p = 0.04)

### Using Bonferroni

Which tests should be accepted or rejected when the FWER is ?

* Since there are five tests, the Bonferroni Method states that a hypothesis can be rejected if . Therefore, tests 2 and 4 should be rejected, while tests 1,3 and 5 should be accepted.

### Using Holm-Bonferroni

Which tests should be accepted or rejected when the FWER is ?

* Start by arranging the p-values in increasing order: 0.002, 0.003, 0.011, 0.04, 0.07
* Need to determine the small such that the -th -value in the list is greater than
* The rounded values of ’s: 0.01, 0.0125, 0.017, 0.025, 0.05
* The smallest *k* is because
* We reject the hypotheses corresponding to the first three -values in the ordered list and accept the remaining. Therefore Tests 2, 3, and 4 are rejected while Tests 1 and 5 are accepted

# Boole’s Inequality

Boole’s inequality states that for a collection of countable events, the probability that at least one of the events happens is no greater than the sum of the probabilities of the events in the collection

### Proof (using induction):

For n=1:

P () P ()

For a collection of n events :

P

Recall that

We apply it to and

Using the associativity of the union , we get that:

P() = P() + P() - P()

Since P () , by the first axiom of probability we have,

P () P () + P ()

and therefore

P()

# Stein’s Paradox

One of the most basic statistical computations is using observed averages to predict the future. For example, a baseball player who gets 7 hits in 20 at bats is said to have a 0.350 batting average, which means we would predict him to get 35 more hits in his next 100 at bats.

Let us consider a sample of 45 major league baseball players, each with a batting average between 0 and 1.

The first step in Stein’s method is to calculate the average of averages, . We then ‘shrink’ all the individual averages towards this overall average; reducing players who are hitting above and increasing players that are hitting below. This shrunken average can be denoted for each player as

OR for

* is the observed average for an individual
* is the average of the ’s
* is a value between 0 and 1 estimated from data
* .

If then all of the ’s are estimated by their values. This would occur if all of the ’s were very close to each other.

The quantity is the amount which the player’s average differs from the overall average. Therefore, this equation says that the estimator differs from the overall average by , where , is the shrinking factor.

In our example, , and . Therefore

OR

If a player in this sample was hitting 0.400, the Stein theorem would say that his true batting ability can be better estimated as

The true average of a batter can be designated as . To compare to our estimators we can calculate the total squared error, given by for each player i. In our sample data, the total squared error using ’s is whereas Stein’s estimates give an error of . This would imply that the Stein estimate is much more accurate.

Stein shows us through this paradox that when the number of means is greater than two, estimating each of them by their own average is an inadmissible procedure for no matter the value of the true means, there are estimations with smaller risk, such as the Stein estimate, which produces a smaller sum of squared error than using the averages.

## Stein’s Paradox Simulation

We will first go through the simulation where . We are going to use

and (shrinking towards 0)

#Set 3 theta values and pick a known value for V  
theta = c(0,1,2)  
V=1  
K=3  
  
#Approximate expectation of the sum of squares   
#Do this by averaging the observed sum of squares  
#For a large number of simulated Y1,Y2,Y3 values  
#Using the given theta's as mean values  
nsim = 100000  
  
#Create sums of squares using Yi's and using stein estimate   
ss0 = ss1 = rep(0,nsim)  
  
#Conduct nsim simulations  
for(iter in 1:nsim){  
 #Generate y|theta,V ~ N(theta,V)  
 y =theta + sqrt(V)\*rnorm(K)  
 #y is a k-vector, with E(yi)=thetai, i=1,..,k  
 #Save sum of squares for yi's  
 ss0[iter]=sum((y-theta)^2)  
 #Compute james-stein estimates   
 #k=3 so shrink towars mu = 0  
 Bhat = (K-2)\*V/sum(y^2)  
 thetahat = Bhat\*0 + (1-Bhat)\*y  
 #Store sum of squares for thetahats  
 ss1[iter] = sum((thetahat - theta)^2)  
 #Print out every 1000th iteration  
 if(iter/1000 == round(iter/1000)) cat(iter,", ")  
}  
  
#Compare average sum of squares values   
mean(ss0); mean(ss1)  
  
#Compute ratio: ss1 should be smaller, so ratio should always be >1  
mean(ss0)/mean(ss1)

We get a ratio of 1.09 with theta = c(0,1,2) meaning the expected sum of squares using the ’s is about 9% larger than using the Stein estimate and shrinking towards = 0

Now let’s look at the *k* = 4 example. We are going to use:

and

There is not much improvement in small *k* problems but it should always show some improvement, no matter the ’s

#Set 4 theta values and pick a known value for V  
theta = c(-1,0,1,2)  
V=1  
K=4  
  
#Approximate expectation of the sum of squares   
#Do this by averaging the observed sum of squares  
#For a large number of simulated Y1,Y2,Y3,Y4 values  
#Using the given theta's as mean values  
nsim = 100000  
  
#Create sums of squares using Yi's and using stein estimate   
ss0 = ss1 = rep(0,nsim)  
  
#Conduct nsim simulations  
for(iter in 1:nsim){  
 #Generate y|theta,V ~ N(theta,V)  
 y = theta + sqrt(V)\*rnorm(K)  
 #y is a k-vector, with E(yi)=thetai, i=1,..,k  
 #Save sum of squares for yi's  
 ss0[iter]=sum((y-theta)^2)  
 #Compute james-stein estimates   
 ybar = mean(y)  
 Bhat = (K-3)\*V/sum((y-ybar)^2)  
 thetahat = Bhat\*ybar + (1-Bhat)\*y  
 #Store sum of squares for thetahats  
 ss1[iter] = sum((thetahat - theta)^2)  
 #Print out every 1000th iteration  
 if(iter/1000 == round(iter/1000)) cat(iter,", ")  
}  
  
#Compare average sum of squares values   
mean(ss0); mean(ss1)  
  
#Compute ratio: ss1 should be smaller, so ratio should always be >1  
mean(ss0)/mean(ss1)

In this example we got a ratio of 1.07, which means the expected sum of squares using the ’s is about 7% larger than using the Stein estimate

#Set 4 theta values and pick a known value for V  
theta = c(71.3,70.7,72.8,70.1)  
V=6.0/5  
K=4  
  
#Approximate expectation of the sum of squares   
#Do this by averaging the observed sum of squares  
#For a large number of simulated Y1,Y2,Y3,Y4 values  
#Using the given theta's as mean values  
nsim = 100000  
  
#Create sums of squares using Yi's and using stein estimate   
ss0 = ss1 = rep(0,nsim)  
  
#Conduct nsim simulations  
for(iter in 1:nsim){  
 #Generate y|theta,V ~ N(theta,V)  
 y = theta + sqrt(V)\*rnorm(K)  
 #y is a k-vector, with E(yi)=thetai, i=1,..,k  
 #Save sum of squares for yi's  
 ss0[iter]=sum((y-theta)^2)  
 #Compute james-stein estimates   
 ybar = mean(y)  
 Bhat = (K-3)\*V/sum((y-ybar)^2)  
 thetahat = Bhat\*ybar + (1-Bhat)\*y  
 #Store sum of squares for thetahats  
 ss1[iter] = sum((thetahat - theta)^2)  
 #Print out every 1000th iteration  
 if(iter/1000 == round(iter/1000)) cat(iter,", ")  
}  
  
#Compare average sum of squares values   
mean(ss0); mean(ss1)  
  
#Compute ratio: ss1 should be smaller, so ratio should always be >1  
mean(ss0)/mean(ss1)

In this example we got a ratio of 1.1, which means the expected sum of squares using the ’s is about 10% larger than using the Stein estimate

# Sources

<https://statweb.stanford.edu/~joftius/slides/testing.pdf>

<http://www.stat.uchicago.edu/~yibi/teaching/stat222/2017/Lectures/C05.pdf>

<http://courses.ieor.berkeley.edu/ieor165/lecture_notes/ieor165_lec19.pdf>

Efrom & Morris 1977: Stein’s Paradox

Everson - Steins Paradox